МОСКОВСКИЙ КОМИТЕТ ОБРАЗОВАНИЯ
ЛИЦЕЙ №1533 (ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ)

ВЫПУСКНАЯ РАБОТА

(специальность «Технология программирования»)

учащихся группы П-11.2

Татаринова Андрея Сергеевича,

Федосеева Василия Олеговича,

Устинова Всеволода Дмитриевича

ИНСТРУМЕНТАЛЬНО - ПРОГРАММНЫЙ КОМПЛЕКС РАЗРАБОТКИ МУЛЬТИАГЕНТНЫХ СИСТЕМ

Москва — 2002

November 14, 2008 12:17 AM -- 1

СОДЕРЖАНИЕ

Тезисы работы 8

Введение 10
0.1. Описание результата разработки 10
0.2. Анализ предметной области. Цель и задачи работы 11

Глава 1. Архитектура комплекса и взаимодействие модулей 13
1.1. Архитектура программного комплекса 13
1.2. Консоль системы 15

Глава 2. Мультиагентная система и Сервер агентов 17

Глава 3. Мультиагентная среда и физический модуль 19
3.2. Перемещение объектов в мультиагентной среде 20
3.3. Обработка столкновений между объектами 22
3.4. Физический модуль и мультиагентная система 27

Глава 4. Визуализация мультиагентной среды и графический модуль 28
4.1. Задачи графического модуля 28
4.2. Представление мультиагентной среды в графическом модуле 29
4.3. Ландшафт 33
4.4. Вспомогательные графические классы 34
4.5. Подготовка данных для рендеринга 37
4.6. Рендеринг объектов мультиагентной среды 39
4.7. Рендеринг ландшафта. Уровни детализации 40
4.8. Графический модуль и мультиагентная система 43

Глава 5. Система подключения языков программирования 44
5.1. Компоновщик языков 44
5.2. Поддержка языка Smalltalk 45

Глава 6. Интегрированная среда разработки для языка Smalltalk 48
6.1. Возможности Интегрированной среды разработки 48
6.2. Реализация Интегрированной среды разработки 49

Глава 7. Выводы 51
7.1. Направления дальнейшей разработки 51

Список литературы 52

Приложение П1. Исходные тексты программ 53
Файл 3dConsoleExtensions\Commands.h 53
Файл 3dConsoleExtensions\Graphics.h 53
Файл 3dConsoleExtensions\System.h 53
Файл 3dsloader\3DSLoader.h 54
Файл AgentsServer\Agents.h 54
Файл AgentsServer\AgentsServer.h 55
Файл AgentsServer\AgentsServerCore.h 56
Файл AgentsServer\System.h 57

November 14, 2008 12:17 AM -- 2

Файл AICore\AICore.h 57
Файл AICore\ASystem.h 58
Файл AICore\Agents.h 59
Файл AICore\Console.h 59
Файл AICore\Creators.h 59
Файл AICore\MessageLanguages.h 60
Файл AICore\Ship.h 60
Файл AICore\System.h 61
Файл AICore\Ufo.h 61
Файл Common\Utilities\1Module.h 62
Файл Common\Utilities\Module.h 62
Файл Common\Utilities\ObjCont.h 63
Файл Console\Commands.h 67
Файл Console\Console.h 67
Файл Console\ConsoleCore.h 68
Файл Console\System.h 69
Файл Console\lexem.h 69
Файл Console\resource.h 70
Файл Console3dCommands\Commands.h 70
Файл Console3dCommands\Globals.h 70
Файл Console3dCommands\Graphics.h 70
Файл Console3dCommands\Keyboard.h 71
Файл Console3dCommands\MainLoop.h 71
Файл Console3dCommands\System.h 71
Файл ConsoleExtensions\Commands.h 71
Файл ConsoleExtensions\ConsoleExtensions.h 72
Файл ConsoleExtensions\ConsoleExtensionsCore.h 72
Файл ConsoleExtensions\Language.h 73
Файл ConsoleExtensions\System.h 73
Файл ConsoleUI\Env.h 73
Файл ConsoleUI2\Globals.h 73
Файл ConsoleUI2\Old3dUI.h 73
Файл ConsoleUI2\Startup.h 74
Файл ConsoleUI2\System.h 74
Файл ConsoleUI2\Termination.h 74
Файл ConsoleUI2\Utilities.h 74
Файл ConsoleUI2\Work.h 75
Файл ConsoleUI2\resource.h 75
Файл ConsoleUIDll\ConsoleUI.h 75
Файл ConsoleUIDll\ConsoleUICore.h 75
Файл ConsoleUIDll\ConsoleUIDll.h 76
Файл ConsoleUIDll\ConsoleWindow.h 76
Файл ConsoleUIDll\System.h 77
Файл ConsoleUIDll\resource.h 77
Файл Core2D\Core2D.h 77
Файл Core2D\Core2DCore.h 77
Файл Core2D\Core2Dint.h 78
Файл Core2D\MainWindow.h 78
Файл Core2D\System.h 79
Файл Core2D\console.h 79
Файл Core2D\resource.h 79
Файл Core3D\core3d.h 80
Файл CPPVM\CPPVM.h 87
Файл CPPVM\CPPVMCore.h 87
Файл CPPVM\system.h 89
Файл ExeptionEngine\Exeption.h 89
Файл ExeptionEngine\ExeptionUtils.h 92
Файл ExeptionEngine\resource.h 92
Файл LAgents\AgentMessage.h 92

Файл LAgents\AgentMessageInt.h 93
Файл LAgents\Agents.h 93
Файл LAgents\LAgent.h 94
Файл LAgents\LAgentCore.h 95
Файл LAgents\LAgentSimulator.h 96
Файл LAgents\System.h 96
Файл LAgents\LServer.h 97
Файл Landscape\landscap.h 97
Файл Linker\IVM.h 99
Файл Linker\LinkerCore.h 100
Файл Linker\linker.h 100
Файл Linker\system.h 101
Файл Linker\vm templates.h 101
Файл LinkerDataTypes\all.h 105
Файл LinkerDataTypes\defs.h 110
Файл LinkerDataTypes\small.h 110
Файл LinkerDataTypes\system.h 110
Файл LinkerDataTypes\utilities.h 111
Файл Loader\Commands.h 117
Файл Loader\LoaderCore.h 117
Файл Loader\System.h 118
Файл Loader\loader.h 118
Файл Loader\resource.h 119
Файл Loader\resource1.h 119
Файл LRenederer\LRenderer.h 119
Файл LRenederer\LRendererCore.h 120
Файл LRenederer\System.h 120
Файл Map\Console.h 120
Файл Map\DataTypes.h 120
Файл Map\Map.h 121
Файл Map\MapCore.h 122
Файл Map\System.h 122
Файл Memory\memory.h 123
Файл Physics\physics.h 124
Файл Physics\physint.h 126
Файл Physics\utils.h 127
Файл PhysicsBase\physbase.h 128
Файл Renderer\extdefs.h 131
Файл Renderer\renderer.h 131
Файл Renderer\rendint.h 137
Файл SmallCompiler\Console.h 140
Файл SmallCompiler\SmallCompiler.h 140
Файл SmallCompiler\SmallCompilerCore.h 140
Файл SmallCompiler\SmallSyntax.h 141
Файл SmallCompiler\syntaxcore.h 141
Файл SmallCompiler\system.h 142
Файл SmallCompiler\tokencore.h 142
Файл SmallDebugger\DebuggerCore.h 146
Файл SmallDebugger\system.h 146
Файл SmallDebugger\Debugger.h 147
Файл SmalltalkEditor\Commands.h 147
Файл SmalltalkEditor\Editor.h 147
Файл SmalltalkEditor\EditorCore.h 148
Файл SmalltalkEditor\System.h 149
Файл SmalltalkEditor\ThreadGate.h 149
Файл SmalltalkEditor\ThreadGateCore.h 150
Файл SmalltalkEditor\Builder\About.h 151
Файл SmalltalkEditor\Builder\DebugForm.h 151
Файл SmalltalkEditor\Builder\DocumentForm.h 152

Файл SmalltalkEditor\Builder\EditorInt.h 152
Файл SmalltalkEditor\Builder\ErrorMessagesForm.h 153
Файл SmalltalkEditor\Builder\Form.h 154
Файл SmalltalkEditor\Builder\Globals.h 154
Файл SmalltalkEditor\Builder\Main.h 154
Файл SmalltalkEditor\Builder\MainFrm.h 155
Файл SmalltalkEditor\Builder\Options.h 156
Файл SmalltalkEditor\Builder\SearchReplace.h 157
Файл SmallVM\DebugObjectsTree.h 158
Файл SmallVM\SmallVM.h 158
Файл SmallVM\SmallVMCore.h 159
Файл SmallVM\system.h 163
Файл SmallVM\Standard Smalltalk Library\Collection Protocols.h 163
Файл SmallVM\Standard Smalltalk Library\Fundamental Protocols.h 165
Файл SmallVM\Standard Smalltalk Library\Numeric Protocols.h 167
Файл SmallVM\Standard Smalltalk Library\Stream Protocols.h 169
Файл Sound\Commands.h 170
Файл Sound\Sound.h 170
Файл Sound\SoundCore.h 170
Файл Sound\System.h 171
Файл Sound\dmutil.h 171
Файл Sound\dsutil.h 172
Файл Sound\dxutil.h 174
Файл Sound\resource.h 175
Файл StorageMaster\objstock.h 175
Файл StorageMaster\stockint.h 177
Файл SystemExtentions1\Console.h 178
Файл SystemExtentions1\Extern.h 178
Файл SystemExtentions1\ExternCore.h 179
Файл SystemExtentions1\System.h 179
Файл Utilities\Module.h 179
Файл Utilities\Utilities.h 180
Файл 3dConsoleExtensions\Commands.cpp 182
Файл 3dConsoleExtensions\Graphics.cpp 186
Файл 3dConsoleExtensions\Module.cpp 188
Файл 3dConsoleExtensions\System.cpp 189
Файл 3dsloader\3DSLoader.cpp 189
Файл AgentsServer\AgentsServerCore.cpp 194
Файл AgentsServer\Module.cpp 201
Файл AgentsServer\System.cpp 201
Файл AICore\AICore.cpp 202
Файл AICore\ASystem.cpp 205
Файл AICore\Console.cpp 206
Файл AICore\Creators.cpp 206
Файл AICore\Module.cpp 206
Файл AICore\Ship.cpp 207
Файл AICore\System.cpp 210
Файл AICore\Ufo.cpp 210
Файл Console\Commands.cpp 215
Файл Console\Console.cpp 218
Файл Console\ConsoleCore.cpp 218
Файл Console\Lexem.cpp 224
Файл Console\Module.cpp 227
Файл Console\System.cpp 227
Файл Console3dCommands\Commands.cpp 227
Файл Console3dCommands\Globals.cpp 230
Файл Console3dCommands\Graphics.cpp 231
Файл Console3dCommands\Keyboard.cpp 233
Файл Console3dCommands\MainLoop.cpp 235

Файл Console3dCommands\Module.cpp....235
Файл Console3dCommands\System.cpp....236
Файл ConsoleExtensions\Commands.cpp....236
Файл ConsoleExtensions\ConsoleExtensionsCore.cpp....239
Файл ConsoleExtensions\Language.cpp....241
Файл ConsoleExtensions\Module.cpp....241
Файл ConsoleExtensions\System.cpp....241
Файл ConsoleRecursiveDescent\Module.cpp....242
Файл ConsoleUI\Main.cpp....243
Файл ConsoleUI2\Globals.cpp....244
Файл ConsoleUI2\Main.cpp....244
Файл ConsoleUI2\Old3dUI.cpp....245
Файл ConsoleUI2\Startup.cpp....249
Файл ConsoleUI2\System.cpp....251
Файл ConsoleUI2\Termination.cpp....251
Файл ConsoleUI2\Utilities.cpp....251
Файл ConsoleUI2\Work.cpp....252
Файл ConsoleUIDll\ConsoleUICore.cpp....253
Файл ConsoleUIDll\ConsoleUIDll.cpp....253
Файл ConsoleUIDll\ConsoleWindow.cpp....254
Файл ConsoleUIDll\System.cpp....255
Файл Core2D\Console.cpp....256
Файл Core2D\Core2D.cpp....257
Файл Core2D\Core2DCore.cpp....257
Файл Core2D\MainWindow.cpp....258
Файл Core2D\System.cpp....262
Файл Core2D\module.cpp....263
Файл Core3D\core3d.cpp....263
Файл CPPVM\CPPVMCore.cpp....288
Файл CPPVM\cppvm.cpp....291
Файл CPPVM\module.cpp....291
Файл CPPVM\system.cpp....292
Файл ExeptionEngine\Exeption.cpp....292
Файл ExeptionEngine\ExeptionCore.cpp....292
Файл ExeptionEngine\ExeptionUtils.cpp....295
Файл LAgents\AgentMessage.cpp....297
Файл LAgents\LAgentSimulator.cpp....298
Файл LAgents\Module.cpp....299
Файл LAgents\system.cpp....300
Файл LAgents\LAgentCore.cpp....300
Файл LAgents\LServer.cpp....301
Файл Landscape\landscap.cpp....304
Файл Linker\linker.cpp....310
Файл Linker\linkercore.cpp....310
Файл Linker\module.cpp....312
Файл Linker\system.cpp....312
Файл LinkerDataTypes\all.cpp....313
Файл LinkerDataTypes\defs.cpp....323
Файл LinkerDataTypes\small.cpp....323
Файл LinkerDataTypes\system.cpp....323
Файл LinkerDataTypes\utilities.cpp....324
Файл Loader\Commands.cpp....326
Файл Loader\LoaderCore.cpp....327
Файл Loader\loader.cpp....332
Файл LRenederer\LRendererCore.cpp....332
Файл LRenederer\Module.cpp....332
Файл LRenederer\system.cpp....333
Файл Map\Console.cpp....333
Файл Map\DataTypes.cpp....334

Файл Map\MapCore.cpp 334
Файл Map\Module.cpp 339
Файл MapConverter\MapConverter.cpp 340
Файл Memory\memory.cpp 340
Файл Physics\physics.cpp 344
Файл Physics\utils.cpp 358
Файл PhysicsBase\physbase.cpp 359
Файл Renderer\renderer.cpp 368
Файл SmallCompiler\Console.cpp 415
Файл SmallCompiler\Module.cpp 416
Файл SmallCompiler\SmallCompilerCore.cpp 417
Файл SmallCompiler\system.cpp 418
Файл SmallCompiler\tokencore.cpp 418
Файл SmallCompiler\syntaxcore.cpp 426
Файл SmallDebugger\DebuggerCore.cpp 433
Файл SmallDebugger\Module.cpp 435
Файл SmallDebugger\system.cpp 435
Файл SmalltalkEditor\Commands.cpp 436
Файл SmalltalkEditor\EditorCore.cpp 436
Файл SmalltalkEditor\Module.cpp 440
Файл SmalltalkEditor\System.cpp 441
Файл SmalltalkEditor\Test.cpp 441
Файл SmalltalkEditor\ThreadGateCore.cpp 442
Файл SmalltalkEditor\Builder\About.cpp 445
Файл SmalltalkEditor\Builder\DebugForm.cpp 446
Файл SmalltalkEditor\Builder\DocumentForm.cpp 446
Файл SmalltalkEditor\Builder\EditorInt.cpp 453
Файл SmalltalkEditor\Builder\ErrorMessagesForm.cpp 455
Файл SmalltalkEditor\Builder\Form.cpp 455
Файл SmalltalkEditor\Builder\Globals.cpp 455
Файл SmalltalkEditor\Builder\Main.cpp 456
Файл SmalltalkEditor\Builder\MainFrm.cpp 456
Файл SmalltalkEditor\Builder\Options.cpp 461
Файл SmalltalkEditor\Builder\SearchReplace.cpp 462
Файл SmallVM\DebugObjectsTree.cpp 463
Файл SmallVM\SmallVMCore.cpp 465
Файл SmallVM\System.cpp 474
Файл SmallVM\module.cpp 474
Файл SmallVM\Standard Smalltalk Library\Collection Protocols.cpp 475
Файл SmallVM\Standard Smalltalk Library\Fundamental Protocols.cpp 475
Файл SmallVM\Standard Smalltalk Library\Numeric Protocols.cpp 477
Файл SmallVM\Standard Smalltalk Library\Stream Protocols.cpp 482
Файл Sound\Commands.cpp 483
Файл Sound\Module.cpp 484
Файл Sound\SoundCore.cpp 484
Файл Sound\System.cpp 487
Файл Sound\dmutil.cpp 488
Файл Sound\dsutil.cpp 494
Файл Sound\dxutil.cpp 509
Файл StorageMaster\objstock.cpp 516
Файл SystemExtentions1\Console.cpp 523
Файл SystemExtentions1\ExternCore.cpp 524
Файл SystemExtentions1\Module.cpp 524
Файл SystemExtentions1\system.cpp 525
Файл Utilities\Utilities.cpp 525
Файл xxxcompiler\XXXCompiler.cpp 528

Приложение П2-а. Описание интерфейсов компонентов 529
Файл IAgent.h - базовый интерфейс для Агентов 529

Файл AgentsServer.h - интерфейс Сервера агентов 530
Файл AgentsServerAgents.h - взаимодействие между агентами....531

Приложение П2-б. Исходные тексты на языке Smalltalk....532
Файл Scr\1.smt....532
Файл Scr\1Pimp.smt 532
Файл Scr\Cactus-try.smt 534
Файл Scr\Test2.smt....535
Файл Scr\block.smt....535
Файл Scr\test.smt 535
Файл Scr\x.smt....536
Файл Scr\Bump.smt....536
Файл Scr\Center.smt 537
Файл Scr\Pimp.smt 539
Файл Scr\Pimp flyover.smt....539
Файл Scr\Ufo.smt 540

Приложение П2-в. Исходные тексты программ для консоли....543
Файл Scr\editor....543
Файл Scr\load3d 543
Файл Scr\loadall 543
Файл Scr\loadide 543
Файл Scr\smalltalk....543
Файл Scr\startup 544
Файл Scr\LSA3D....544
Файл Scr\LSmt 544
Файл Scr\LoadSmalltalk....544
Файл Scr\loadServAnd3d 544
Файл Common\Utilities\Module.txt....545
Файл ConsoleRecursiveDescent\Zabor.txt 546
Файл ConsoleUI2\ExeptionDump.txt....548
Файл meshes\ufo\3dsdump.txt....548
Файл meshes\ufo\xxxdump.txt....550
Файл meshes\ball\3dsdump.txt 550
Файл Scr\3dsdump.txt....551
Файл Scr\Agents.txt....551
Файл Scr\Modules.txt 551
Файл Scr\ScreenSettings.txt....551
Файл Scr\Test.txt 552
Файл Scr\continue.txt 552
Файл Scr\testsmt.txt....552
Файл Scr\vw.txt 552
Файл Scr\all.txt....552

Приложение П3. Диаграммы основных классов....553
П3.1. Диаграммы классов физического модуля....553
П3.2. Диаграммы классов графического модуля 556

November 14, 2008 12:17 AM -- 8

Тезисы работы

Цель работы – создать программный комплекс, позволяющий без лишних трудозатрат разрабатывать и исследовать мультиагентные системы. *В состав комплекса входят:* компилятор, отладчик, интегрированная среда разработки языка *Smalltalk*, средства поддержки работы мультиагентной системы и средства трехмерной визуализации, предусмотрено подключение компонентов-расширений комплекса, в том числе дополнительных языков программирования.

Целевой аудиторией дипломной работы являются программисты, работающие в сфере разработки методов искусственного интеллекта. Также комплекс может использоваться для обучения программированию. Базовым языком алгоритмического моделирования был выбран *Smalltalk*, так как значительная часть целевой аудитории (а именно студенты и преподаватели МИРЭА) используют этот язык в качестве основного языка программирования.

Мультиагентные системы – наиболее развитая современная технология реализации методов искусственного интеллекта. Мультиагентная система состоит из агентов и объектов. Основное свойство агентов – способность принятия решений, объекты статичны и таким свойством не обладают. В нашем комплексе агенты помещаются в мультиагентную среду, построенную как упрощенный вариант мира с реальными физическими законами.

Комплекс построен на *компонентной архитектуре*. Все компоненты представляют собой *Plug-in* модули, подключаемые к главной программе. Загрузка и взаимодействие модулей построены по идеологии *COM*. У каждого модуля есть свой интерфейс – структура, через которую происходят все вызовы функций модуля. Загрузка, инициализация и выгрузка компонентов осуществляется с помощью специального модуля-загрузчика, который загружается главной программой. Главная программа не несет в себе никаких функций, кроме предоставления базового пользовательского интерфейса и первичной загрузки и инициализации основных модулей комплекса.

Компоненты системы *взаимодействуют* друг с другом тремя способами: прямыми вызовами, через модуль-компоновщик и через модуль-консоль. Через модуль-компоновщик также идет работа с объектами, реализованными на внешних (подключаемых) языках системы и межъязыковое взаимодействие. Разработанный интерфейс компоновщика позволяет подклю-

November 14, 2008 12:17 AM -- 9

чать другие языки программирования. В настоящее время поддерживаются *Smalltalk* как встроенный транслятор, а также откомпилированные модули (*EXE*, *DLL*) на любом другом языке. Через консоль реализуется прямой контроль системы пользователем.

Мультиагентная система связана с комплексом с помощью *Сервера агентов*, но модули системы имеют право работать с агентами непосредственно через модуль-компоновщик. Сервер агентов также несет на себе функции по созданию, инициализации и поддержке работы агентов. Сервер агентов также обеспечивает взаимодействие агентов друг с другом.

Для физических взаимодействий агентов создан *физический модуль,* который позволяет поместить агентов в мультиагентную среду, являющуюся некоторым упрощением мира с реальными физическими законами. Физический модуль позволяет агентам перемещаться в мультиагентной среде, а также взаимодействовать друг с другом на физическом уровне, то есть сталкиваться, входить в прямой физический контакт.

Для поддержки языка *Smalltalk* в комплексе был реализован *транслятор* этого языка и интегрированная среда разработки. Транслятор позволяет *компилировать* исходные тексты программы в байт-код, используемый при исполнении методов объектов, созданных на этом языке. Это происходит в два прохода: первый — лексический анализ с помощью конечного автомата, второй — синтаксический анализ методом рекурсивного спуска. В интегрированную среду входят также редактор исходных текстов на языке *Smalltalk* и отладчик. Реализованы такие функции, как подсветка синтаксиса языка, пошаговое выполнение программы, просмотр локальных переменных, использование точек останова программы. Реализована возможность *динамической рекомпиляции* — изменения исходного текста программы прямо во время ее работы.

Консоль является модулем, позволяющим принимать и обрабатывать команды пользователя. Набор команд определяется тем, какие модули комплекса загружены в данный момент.

Графический модуль построен на базе библиотеки *OpenGL* и позволяет пользователю наблюдать за действиями агентов в мультиагентной среде. Он содержит набор классов функций, позволяющих визуализировать агентов, среду, в которой они находятся, а также перемещаться по этой среде, наблюдая за разными агентами с разных позиций.

Возможность в любой момент наблюдать за действиями агентов и корректировать разработанные алгоритмы уменьшает трудозатраты на создание средств отладки для мультиагентной системы и позволяет программисту сконцентрироваться на главной задаче – разработке методов и алгоритмов искусственного интеллекта.

November 14, 2008 12:17 AM -- 10

Введение

В настоящее время разработка искусственного интеллекта является важной научной и инженерной задачей. Однако программистам, занимающимся разработкой алгоритмов искусственного интеллекта, приходится тратить много времени на создание всевозможных сопутствующих программ, которые облегчили бы им процесс изучения поведения разработанного ими искусственного интеллекта в мультиагентных системах.

Мультиагентные системы – наиболее развитая современная технология реализации методов искусственного интеллекта. Мультиагентная система состоит из агентов и объектов. Основное свойство агентов – способность принятия решений, объекты статичны и таким свойством не обладают. В нашем комплексе агенты помещаются в мультиагентную среду, построенную как упрощенный вариант мира с реальными физическими законами.

Возможность в любой момент наблюдать за действиями агентов и корректировать разработанные алгоритмы уменьшает трудозатраты на создание средств отладки для мультиагентной системы и позволяет программисту сконцентрироваться на главной задаче – разработке методов и алгоритмов искусственного интеллекта.

0.1. Описание результата разработки

Цель работы – создать программный комплекс, позволяющий без лишних трудозатрат разрабатывать и исследовать мультиагентные системы. В состав комплекса входят: компилятор, отладчик, интегрированная среда разработки языка *Smalltalk*, средства поддержки работы мультиагентной системы и средства трехмерной визуализации, предусмотрено подключение компонентов-расширений комплекса, в том числе дополнительных языков программирования.

Целевой аудиторией дипломной работы являются программисты, работающие в сфере разработки методов искусственного интеллекта. Также комплекс может использоваться для обучения программированию. Базовым языком алгоритмического моделирования был выбран *Smalltalk*, так как значительная часть целевой аудитории (а именно студенты и преподаватели МИРЭА) используют этот язык в качестве основного языка программирования.

November 14, 2008 12:17 AM -- 11

В задачи, поставленные перед авторами диплома, входит разработка средств, позволяющих эффективно работать над созданием различных алгоритмов искусственного интеллекта, реализуя искусственный интеллект в виде агентов. Программист, использующий комплекс, не должен заботиться о том, как он будет наблюдать за действиями написанных им агентов, или как он будет отлаживать их интеллект.

Основным средством разработки искусственного интеллекта в комплексе является интегрированная среда разработки программ на языке *Smalltalk*, позволяющая эффективно создавать и отлаживать программы, написанные на этом языке, а также компилятор этого языка, преобразовывающий текст программы в байт-код, используемый при выполнении методов объектов, написанных на *Smalltalk*.

Одним из модулей комплекса является Сервер агентов – ядро любой мультиагентной системы. Через него агенты взаимодействуют друг с другом.

Также в состав комплекса будет включен физический модуль, позволяющий поместить агентов в виртуальный мир с заданными условиями и организовать их взаимодействие с этим миром.

Важной частью системы является графический модуль, позволяющий программисту наблюдать за действиями агентов, созданных им.

0.2. Анализ предметной области. Цель и задачи работы

В настоящее время перед многими программистами стоит задача алгоритмического моделирования искусственного интеллекта. В различных учебных заведениях, где изучаются и разрабатываются методы и алгоритмы искусственного интеллекта, необходимо использование удобной системы, позволяющей учащемуся сконцентрироваться на разработке мультиагентных систем и интеллекта агентов и не отвлекаться на создание, например, графических средств исследования созданных им алгоритмов.

Вместе со многими играми поставляются программы, позволяющие создать виртуальный мир, в котором будет происходить действие игры, задать сюжетную линию, описать поведение противников (например, *QuakeC* для игры *Quake* 2 или *AuroraToolset* для игры *NeverWinter Nights*). Хотя эти программы используют уже готовое графическое и физическое ядро от игры и на них можно смоделировать искусственный интеллект, они не являются полностью универсальными, то есть схема поведения персонажа задается только в контексте сюжета иг-

November 14, 2008 12:17 AM -- 12

ры. К тому же в чистом виде мультиагентные системы в играх не реализовываются, так как в этом нет необходимости. Персонажи в играх агентами не являются, так как не способны принимать решения, а могут только откликаться на действия игрока.

Поэтому перед авторами дипломной работы была поставлена задача не только разработать редактор и компилятор языка *Smalltalk*, но и создать набор сопутствующих библиотек, позволяющих эффективно работать над созданием алгоритмов искусственного интеллекта. В задачи, поставленные перед авторами, входило создание сервера агентов, обязательной части любой мультиагентной системы. Этот сервер является модулем комплекса и позволяет не реализовывать всю структуру общения агентов друг с другом и со средой, в которой они существуют.

Также задачей, поставленной перед авторами, является создание полноценного графического и физического ядра, которое позволило бы агентам существовать в виртуальной среде по реальным законам физики, а пользователю нашего комплекса – наблюдать за действиями агентов в реальном времени. Графическое ядро должно быть универсальным, то есть не быть рассчитанным только, например, на отображение ландшафтов или на прорисовку помещений. Также и графическое, и физическое ядро должны быть масштабируемыми, то есть ненужные для данной задачи функции можно было бы отключить.

November 14, 2008 12:17 AM -- 13

Глава 1.
Архитектура комплекса
и взаимодействие модулей

1.1. Архитектура программного комплекса

Программно – инструментальный комплекс построен на компонентной архитектуре (рис.1.1.). Все компоненты представляют собой модули, подключаемые к главной программе. Загрузка и взаимодействие модулей построены по идеологии COM. У каждого модуля есть свой интерфейс – структура, через которую происходят все вызовы функций модуля. Компоненты системы *взаимодействуют* друг с другом тремя способами: прямыми вызовами, через модуль-компоновщик и через модуль-консоль. Главная программа не несет в себе никаких функций, кроме предоставления базового пользовательского интерфейса и первичной загрузки и инициализации основных модулей комплекса.

Рис.1.1. Структура системы

Центральным звеном архитектуры комплекса является модуль-загрузчик, позволяющий подключать к главной программе различные компоненты, входящие в состав комплекса и используемые в нем. На рисунке он показан совокупностью стрелок. Каждый компонент пред-

November 14, 2008 12:17 AM -- 14

ставляет собой динамически подключаемую библиотеку, созданную, следуя определенным правилам. Каждая библиотека должна содержать в себе описание экспортируемого объекта системы и предоставлять главной программе структуру – интерфейс данного объекта. Также библиотека должна экспортировать рад функций, позволяющих получить информацию о компоненте, получить объект системы, а также выполнить начальную инициализацию компонента.

Модуль-загрузчик позволяет загрузить динамически подключаемую библиотеку в память и получить у нее указатели на служебные функции. По требованию любого компонента модуль-загрузчик возвращает ему указатель на интерфейс любого объекта системы. Экспорт объекта системы из динамически подключаемой библиотеки осуществляется по идеологии *COM*. Главной программе известен лишь интерфейс компонента, а библиотека экспортирует функцию, создающую в памяти объект системы и возвращающую главной программе указатель на его интерфейс. Также возможно подключение компонентов, полностью построенных на технологии *COM*. Также компоненты экспортируют функции, которые вызываются во время ключевых моментов работы компонента – при загрузке, инициализации, выгрузке компонента из памяти.

Компоненты могут не экспортировать объект в систему или вообще не содержать объектов, например – модули расширения консоли.

Основной способ связи компонентов системы – прямые вызовы нужных функций. Модуль – загрузчик позволяет компоненту системы получить указатель на интерфейс нужного ему модуля и работать с ним. Также существует низкоуровневый модуль – аллокатор, позволяющий структурированно хранить данные, общие для нескольких компонентов. Через этот модуль – аллокатор компоненты могут работать с общими данными.

Основными компонентами, работающими с пользователем, являются консоль и её подключаемые пользовательские интерфейсы. Консоль представляет собой командный интерпретатор и средство стандартизированного вывода информации. Об алгоритмах консоли будет рассказано позднее.

Главная программа при запуске создаёт только модуль-загрузчик, и консоль (и предоставляет консоли пользовательский интерфейс). Все остальные функции комплекса выполняются компонентами системы.

November 14, 2008 12:17 AM -- 15

1.2. Консоль системы

Консоль является модулем, позволяющим принимать и обрабатывать команды пользователя. Набор команд определяется тем, какие модули комплекса загружены в данный момент (рис.1.2).

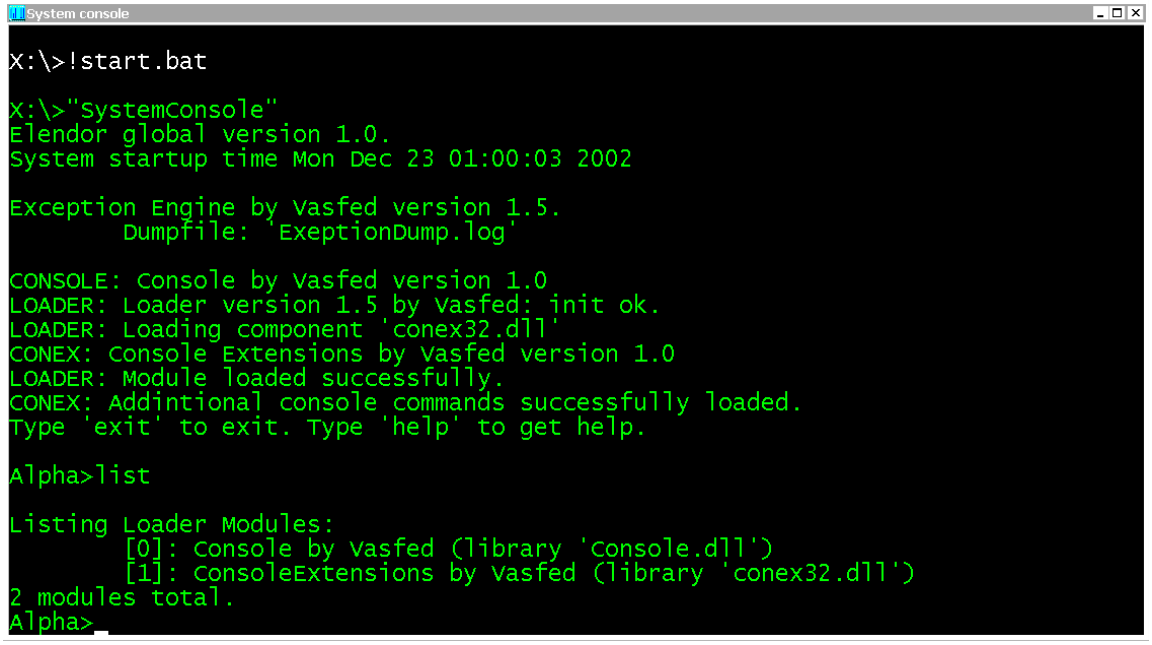


Рис.1.2. Консоль системы

Командный интерпретатор работает с выражениями вида*: «<команда> [параметры]».* При загруженных расширениях консоли, поддерживаются пакетные файлы и «префиксы» команды – переадресация команды в качестве параметров к задаваемой команде.

Консоль содержит в себе таблицу команд с информацией о загруженных командах. Там хранятся такие данные как ключевое слово команды, указатель на функцию, которая эту команду обрабатывает, название компонента, который экспортировал команду и служебные параметры команды.

Любой компонент может экспортировать команду в консоль. При этом модуль должен указать консоли, в каком виде команде будут передаваться параметры: в виде разобранных и расшифрованных лексем или в виде исходной строки.

Разбор команды производится с помощью конечного автомата (этот алгоритм описан в главе 3). Причем при сворачивании лексемы она сразу расшифровывается. Потом консоль ищет команду с ключевым словом, совпадающим с первой лексемой, и вызывает функцию,

November 14, 2008 12:17 AM -- 16

обрабатывающую её. Если такой команды не найдено, управление передается обработчику по умолчанию, если он не установлен, то выводится сообщение об ошибке.

Расширения консоли экспортируют в консоль команды для обработки пакетных файлов, обработчик по умолчанию, который пытается обработать команду и команды для установки префикса команд.

1.2.1. Основные команды консоли

Основные команды консоли приведены в таблице 1. Другие команды, экспортированные компонентами, описаны в документации по соответствующим компонентам.

Таблица 1. Основные команды консоли

N	Ключевое слово	Описание	Модуль
1	*load*	Загрузка компонента	Модуль-загрузчик
2	*init*	Инициализация непроинициализированных компонентов	Модуль-загрузчик
3	*batch*	Исполнение пакетного файла	Расширения консоли
4	*setprefix*	Установка префикса команд	Расширения консоли
5	*BackToConsole*	Сброс префикса команд	(Встроенная команда)
6	*smt*	Компиляция и выполнение команды на языке *Smalltalk*	Компилятор *Smalltalk*
7	*start3d*	Запуск графики и физики	Графический модуль
8	*pause3d*	Пауза/восстановление работы графики	Графический модуль
9	*stop3d*	Остановка графической подсистемы и физики	Графический модуль
10	*exit*	Выход из программы	Пользовательский интерфейс

November 14, 2008 12:17 AM -- 17

Глава 2.
Мультиагентная система и Сервер агентов

Логическое взаимодействие агентов определяется их искусственным интеллектом и происходит, главным образом, через сервер агентов.

Сервер агентов – основной компонент, соединяющий комплекс с мультиагентной системой. Также сервер агентов отвечает за поддержку существования агентов и их создание. Но сначала о самих агентах.

Сервер агентов также несет на себе функции по созданию, инициализации и поддержке работы агентов. Сервер агентов также обеспечивает взаимодействие агентов друг с другом (рис. 2.1).

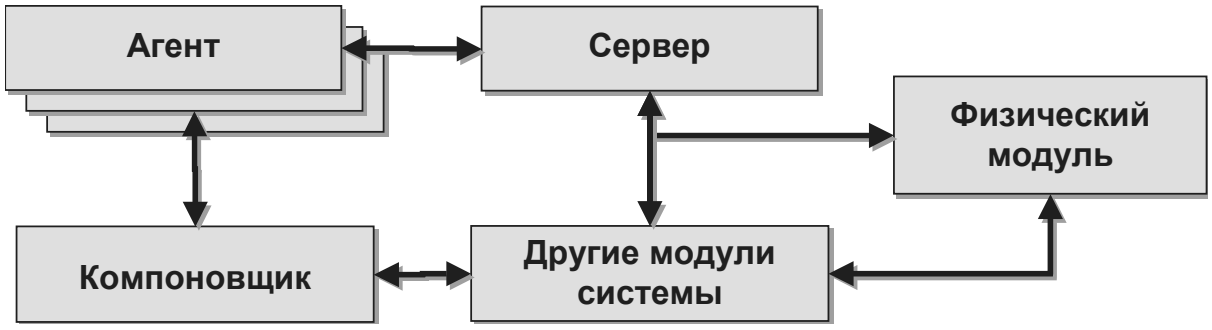


Рис. 2.1. Взаимодействие Агентов через Сервер агентов.

Все агенты, независимо от способа реализации, наследуются из *IAgent*, основного интерфейса агентов, что обеспечивает возможность одинаковой работы со всеми типами агентов. *IAgent* не является «классическим» интерфейсом, так как содержит в себе данные, обязательные для всех агентов (положение в виртуальном пространстве, индекс в сервере агентов, физическом модуле и др.), причем эти данные изменяются системой, а не агентами. Кроме того *IAgent* содержит в себе объявления процедур, также обязательных для реализации в агентах. Полное описание *IAgent* приведено в приложении П2.

Искусственный интеллект агентов обязан являться пошаговым, но так как время между «шагами» мало (менее 0.05 секунды, зависит от компьютера пользователя), то можно считать

November 14, 2008 12:17 AM -- 18

процесс принятия решений непрерывным. Агенты влияют на своё положение в пространстве только посредством изменения своей скорости или ускорения. Агенты получают данные как пассивно, так и активно, то есть могут сами запрашивать необходимые им данные из сервера агентов, а часть данных поступает к ним автоматически, например, сообщения, которыми могут обмениваться агенты.

Вернемся теперь к серверу агентов. Как было сказано выше, сервер агентов – промежуточное звено между комплексом и мультиагентной системой, этим объясняется разделение его функций на специфические для агентов, специфические для комплекса и «нейтральные», которые могут использовать как агенты, так и компоненты комплекса. Функции, специфические для агентов, в основном, представляют собой переадресацию в иные модули комплекса, например, в модуль физики.

«Специфические для комплекса» функции сервера агентов предназначены для манипуляции, причем агенты рассматриваются как обычные данные. Создание агентов, с точки зрения сервера агентов, происходит двумя способами: с помощью механизма «создателей агентов» или обычной регистрацией созданного каким-либо способом агента в сервере агентов. «Создатели агентов» - специальные функции, зарегистрированные в сервере агентов, которые по запросу сервера агентов обязаны только создать нового агента. «Создатели» имеют имена, как правило совпадающие с названием типа агентов, создаваемого с их помощью, при регистрации им присваиваются уникальные номера, которые в дальнейшем считаются синонимами имени. Типы агентов различаются по этим номерам, но если агент не был создан с помощью механизма «создателей», он будет считаться агентом неопределенного типа и другие агенты, обладающие таким же искусственным интеллектом, не смогут определить его «своим» и взаимодействовать с ним даже через сообщения. Независимо от пути создания, агенту присваивается уникальный номер, вся работа с агентом производится по этому номеру.

Сервер агентов также отвечает за предоставление каждому агенту «права хода», это происходит во время, когда графический модуль готовится к формированию следующего кадра. Сервер агентов при этом последовательно вызывает метод «*Go*» у всех агентов системы. В случае ошибки, сервер не прекращает выполнения. Агенты имеют право инициировать исключительную ситуацию «пропустить ход», которая вообще не считается ошибкой.

Полный перечень функций сервера агентов приведен в приложении П2. Компоненты, взаимодействующие и с комплексом, и с агентами, но не через сервер, считаются частью сервера агентов.

November 14, 2008 12:17 AM -- 19

Глава 3.
Мультиагентная среда и физический модуль

3.1. Задачи и структура физического модуля

Все агенты мультиагентной системы помещаются в среду, в которой смоделированы основные физические законы. В этой среде агенты существуют и действуют. Моделирование такой среды нужно для того, чтобы узнать, как будут действовать агенты в условиях, приближенных к реальным.

Для моделирования таких условий был разработан и реализован физический модуль (см. рис.1.1). В основу математической модели физического модуля было положено утверждение, что все объекты мультиагентной среды – материальные точки, и все силы, действующие на них, приложены к их центру тяжести. Объекты могут взаимодействовать друг с другом посредством столкновений, при этом между ними нет сил трения, и удары - абсолютно упругие. Агент может быть представлен в виде одного такого объекта или в виде иерархической структуры этих объектов.

Сам объект представлен как сетка треугольных граней (полигонов), задающих его геометрическую форму. Поскольку все объекты считаются материальными точками, для расчетов, проводимых физическим модулем, форма не важна – она используется лишь для графического отображения. Сам массив полигонов описывается классом *CMesh*. Этот класс является основным для описания объектов в мультиагентной среде. Помимо графической информации в этом классе содержится информация о центре тяжести объекта.

Для использования физического модуля нужно через модуль-загрузчик получить указатель на интерфейс класса *CEnvironment*. Этот класс содержит в себе информацию о параметрах моделируемой мультиагентной среды, таких как ускорение свободного падения. Также он содержит экземпляр класса *CPhysTable* – таблицу, в которой каждому объекту сопоставляется набор базовых классов, используемых физическим модулем при расчетах. Также в этой таб-

November 14, 2008 12:17 AM -- 20

лице содержится указатель на текущий ландшафт, задающий рельеф местности мультиагентной среды.

Каждому объекту соответствует набор классов, описывающих состояние объекта в мультиагентной среде: скорость, ускорение, начальная позиция на текущем этапе движения, таймер, показывающий время, прошедшее с начала текущего этапа движения, материал, из которого сделан объект, а также классы, позволяющие производить проверку на столкновение объектов друг с другом и с ландшафтом.

3.2. Перемещение объектов в мультиагентной среде

Объект может двигаться равномерно или равноускоренно, при этом все его движение подразделяется на этапы – такие участки пути, на которых ускорение объекта постоянно. Если агент решает изменить свое ускорение, информация об этом поступает в физический модуль, который производит смену этапов движения.

Формула, выражающая зависимость координат центра тяжести объекта от времени – это кинематическая формула равноускоренного движения:

$$C = P_0 + t*V_0 + (A + g)*t^2 / 2, \qquad (3.1)$$

где C – это вектор относительно мировой системы координат, задающий положение центра тяжести объекта, P_0 – начальная позиция объекта на данном этапе движения, V_0 – вектор начальной скорости на текущем участке пути, A – вектор, задающий ускорение объекта, g – ускорение свободного падения, t – время, прошедшее с момента начала текущего этапа движения.

Зависимость скорости объекта от времени выражается формулой

$$V = V_0 + t*(A + g), \qquad (3.2)$$

где V_0 – начальная скорость на данном участке пути, V – мгновенная скорость в данный момент.

При изменении ускорения совершается переход на новый этап движения. При этом текущая позиция центра тяжести объекта становится начальной на новом этапе движения, текущая мгновенная скорость, рассчитанная по формуле (3.2), также становится начальной на данном этапе движения, и значение таймера обнуляется, и дальше ведется отсчет времени только для нового этапа движения.

November 14, 2008 12:17 AM -- 21

Такая система разделения движения на этапы сделана из-за того, что формула зависимости положения центра тяжести от времени (3.1) верна, только если ускорение на всем пути неизменно. Если ускорение по какой-то причине меняется, формула перестает давать верные результаты.

Для реализации перемещения объектов в мультиагентной среде было реализовано несколько базовых классов, содержащих информацию, нужную для расчетов физического модуля.

В объектах класса *CSpeed* (диаграмма П3.1[1]) содержится информация о скорости объекта. Класс *CSpeed* содержит три вектора – вектор мгновенной скорости в данный момент времени, вектор начальной скорости на данном этапе движения, а также вектор циклической скорости (информация об углах вращения по трем осям координат).

В объектах класса *CAcceleration* (диаграмма П3.2) содержится информация о текущем векторе ускорения объекта. К каждому объекту может быть приложено несколько сил сразу, поэтому объекту сопоставляются сразу несколько ускорений, а при подстановке в формулы берется результирующий вектор (векторная сумма всех ускорений).

Класс *CPosition* (диаграмма П3.3) содержит в себе информацию о начальной позиции объекта на текущем этапе движения.

Класс *CPhysMaterial* (диаграмма П3.4) содержит в себе информацию о свойствах материала, из которого сделан объект, таких как плотность вещества, коэффициент упругости, объем и масса объекта.

Класс *CTimer* (диаграмма П3.5) – это таймер, позволяющий вести отсчет времени в секундах, используя системный таймер. Этот класс содержит в себе информацию о времени с момента включения компьютера, а также собственно время, прошедшее с момента «включения» таймера (то есть с момента фиксации начального времени). Из этого класса унаследован класс *CTimeDelta* (диаграмма П3.6) – таймер, снабженный дополнительными функциями и данными, используемыми физическим модулем. Этот класс содержит в себе также информацию о значении показаний таймера на момент предыдущей итерации главного цикла работы физического модуля. Эти данные нужны при обработке столкновений, когда столкнувшиеся объекты возвращаются на свои предыдущие позиции.

[1] Диаграммы классов приведены в приложении 3.

3.3. Обработка столкновений между объектами

Существует набор классов, позволяющих производить проверку на столкновение между объектами. Хотя все объекты в мультиагентной среде считаются материальными точками, в действительности же они имеют форму. Для приближенного определения, произошло столкновение между объектами или нет, используются геометрические формы, представляющие собой упрощенный вариант формы самого объекта. Каждый объект имеет набор таких форм с разным уровнем приближения к действительной своей форме (рис.3.1).

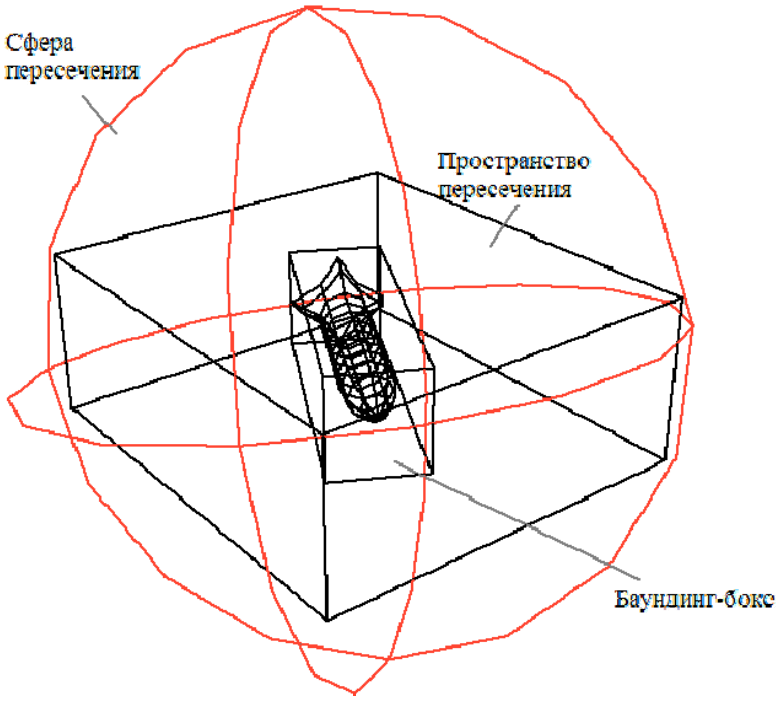


Рис.3.1. Различные уровни аппроксимации формы объекта

Самым низким уровнем приближения обладает сфера пересечения, радиус которой хранится в классе *CIntersectionSphere* (диаграмма 3.7). Сфера пересечения охватывает сам объект и небольшое пространство вокруг него. Центр этой сферы совпадает с центром тяжести объекта. Проверка на пересечение двух таких сфер не требует емких вычислений, поэтому это – первый этап проверки на столкновение двух объектов. Если пересеклись две сферы пересечения, то есть вероятность, что пересеклись и сами объекты (столкнулись, но сам факт столкновения, касания зафиксировать сложно, так как объект может «проскочить», проникнуть в другой объект на каком-то участке своего движения, это и будет считаться в дальнейшем фактом столкновения). Для того чтобы узнать, произошло ли проникновение одной сфе-

November 14, 2008 12:17 AM -- 23

ры пересечения в другую сферу, необходимо сравнить расстояние между центрами тяжести двух объектов и сумму радиусов сфер этих объектов.

Если было зафиксировано столкновение двух сфер пересечения, происходит следующий этап проверки на столкновение между объектами – проверка на столкновение двух пространств пересечения. Пространство пересечения – это прямоугольный параллелепипед, ребра которого параллельны осям координат мировой системы отсчета. Центр этого параллелепипеда совпадает с центром координат объекта. Информация о размерах этого параллелепипеда содержится в объектах класса *CIntersectionSpace* (диаграмма 3.8). Пространство пересечения задается таким образом, чтобы учитывать все возможные для данного объекта вращения относительно своего центра тяжести. Чтобы проверить, не пересеклись ли два таких пространства, необходимо проверить, не пересеклись ли ребра задающих их прямоугольных параллелепипедов по осям *X*, *Y* и *Z* (рис.3.2).

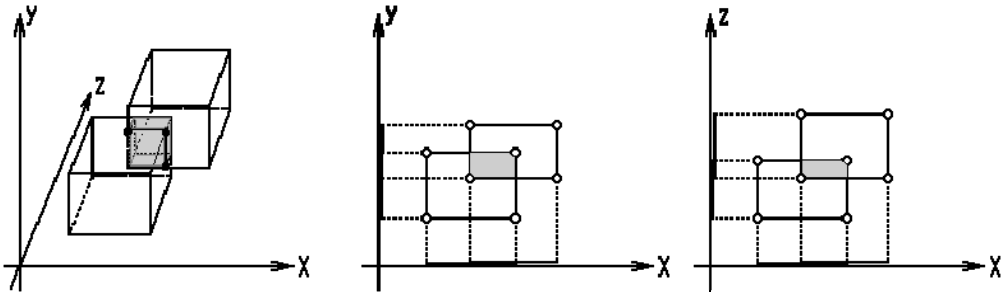


Рис.3.2. Проверка на столкновение двух пространств пересечения

Следующий (и последний) этап проверки на столкновение между двумя объектами – проверка на пересечение между двумя баундинг-боксами – «охватывающими коробками» (от английского термина *bounding box*). Баундинг-бокс представляет собой прямоугольный параллелепипед наименьшего объема, полностью охватывающий данный объект. Информация об этом параллелепипеде содержится в классе *CBoundingBox* (диаграмма 3.9). Этот параллелепипед может быть произвольно ориентирован в пространстве (то есть не обязательно его ребра должны быть параллельны осям координат), и при вращении самого объекта вращается вместе с ним. Баундинг-бокс является следующим, наиболее полным приближением к форме самого объекта.

Баундинг-бокс задается точкой (одной из вершин параллелепипеда) и тремя векторами относительно этой точки, по которым строятся все ребра параллелепипеда (рис.3.3).

November 14, 2008 12:17 AM -- 24

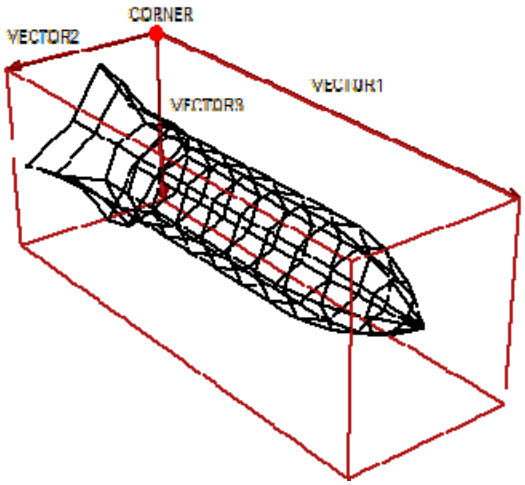


Рис.3.3. *Bounding Box* – «охватывающий параллелепипед»

Для того чтобы узнать, пересеклись ли два баундинг-бокса, нужно проверить, не пересеклось ли хотя бы одно ребро первого баундинг-бокса с какой-либо гранью второго. Для этого сначала находится точка пересечения прямой, содержащей ребро одного баундинг-бокса, и плоскости, содержащей грань другого баундинг-бокса. Эта точка находится из формулы, полученной из решения системы уравнений:

$$\begin{cases} X = X_0 + Xd * T \\ Y = Y_0 + Yd * T \\ Z = Z_0 + Zd * T \end{cases} \text{ Параметрическое уравнение прямой}$$
$$aX + bY + cZ + d = 0 \quad \text{Параметрическое уравнение плоскости} \qquad (3.3)$$

где X, Y, Z – координаты точки пересечения прямой и плоскости; T – параметр в параметрическом уравнении прямой; a, b, c, d – параметры в параметрическом уравнении плоскости; $X0$, $Y0$, $Z0$ – координаты начальной точки прямой (в качестве этой точки берется начальная точка проверяемого ребра), Xd, Yd, Zd – вектор направления прямой (в качестве этого вектора берется вектор, задающий ребро относительно начальной точки). Из этой системы получаем формулу для параметра T:

$$T = -(a * X_0 + b * Y_0 + c * Z_0 + d) / (a * Xd + b * Yd + c * Zd) \qquad (3.4)$$

Если знаменатель получившейся дроби равен нулю, то прямая параллельна плоскости. Подставив получившийся параметр T в уравнение прямой, получаем точку пересечения прямой и плоскости. Теперь нужно проверить, принадлежит ли эта точка ребру и грани, проверяемых на пересечение. Если значение параметра T лежит в пределах от нуля до единицы, то

November 14, 2008 12:17 AM -- 25

точка пересечения принадлежит ребру. Чтобы узнать, принадлежит ли точка пересечения грани, нужно проверить, равна ли площадь грани сумме площадей треугольников, образованных точкой пересечения и четырьмя ребрами граней.

Если в конце всех вычислений мы получили точку, принадлежащую и ребру, и грани, то зафиксировано их пересечение.

Для проверки на столкновение двух баундинг-боксов нужно проверить на пересечение каждое из двенадцати ребер одного баундинг-бокса с каждой из шести граней второго баундинг-бокса. Этот алгоритм содержит больше всего вычислений, поэтому он применяется на последнем этапе проверки на столкновение двух объектов, когда вероятность того, что они столкнулись, достаточно велика. Если зафиксировано пересечение двух баундинг-боксов, то считается, что объекты столкнулись.

Если зафиксировано столкновение двух объектов, то его необходимо обработать. Так как все объекты считаются материальными точками, то деформироваться они не могут, и приобрести вращательное движение также не могут. Поэтому при столкновении объектов они получают новые скорости, с которыми и продолжают движение. Эти скорости рассчитываются с применением законов сохранения импульса и законов сохранения энергии:

$$\begin{cases} m_1 * V_1 + m_2 * V_2 = m_1 * V'_1 + m_2 * V'_2 & \text{– закон сохранения импульса} \\ \dfrac{m_1 V_1^2}{2} + \dfrac{m_2 V_2^2}{2} = \dfrac{m_1 V'^2_2}{2} + \dfrac{m_2 V'^2_2}{2} & \text{– закон сохранения энергии,} \end{cases} \qquad (3.5)$$

где m_1, m_2 – массы столкнувшихся объектов; V_1, V_2 – векторы скоростей объектов до столкновения; V'_1 и V'_2 – векторы скоростей объектов после столкновения, которые и нужно найти.

Поскольку столкновения между объектами считаются абсолютно упругими, то система (3.5) верна. Из этих законов получены формулы скоростей объектов после столкновения:

$$V'_1 = (2 * m_2 * V_2 + V_1 * (m_1 - m_2)) / (m_1 + m_2) \qquad (3.6)$$
$$V'_2 = (2 * m_1 * V_1 + V_2 * (m_2 - m_1)) / (m_1 + m_2) \qquad (3.5)$$

По этим формулам рассчитываются векторы скоростей объектов после столкновения.

Несколько иным образом происходит проверка на столкновение объектов с ландшафтом и обработка такого столкновения. Ландшафт представлен в виде триангулированой регулярной сетки высот (то есть ландшафт – это двумерный массив высот, соединенных в треугольники, если спроецировать вершины этих треугольников на горизонтальную плоскость, то по-

November 14, 2008 12:17 AM -- 26

лучится регулярная сетка) (рис.3.4). Для того чтобы узнать, столкнулся ли объект с ландшафтом, необходимо проверить, не оказалась ли одна из вершин его баундинг-бокса ниже уровня ландшафта в этой точке. Для этого у проверяемой точки берутся координаты по осям *X* и *Y*, и определяется, какому треугольнику на плоскости они принадлежат. Потом через точку (*X, Y*, 0) проводится вертикальная прямая, и по описанному выше алгоритму находится точка пересечения этой прямой с треугольником (рис.3.4). *Z*-координата этой точки и считается высотой ландшафта в точке (*X, Y*).

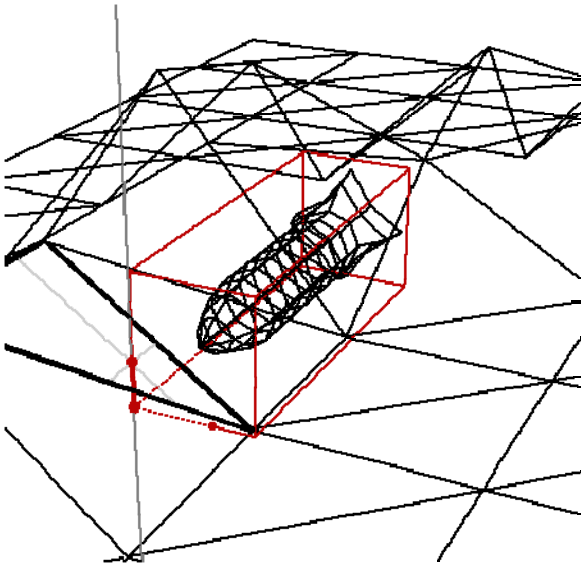


Рис.3.4. Проверка на столкновение объекта с ландшафтом

Далее производится проверка всех восьми вершин баундинг-бокса, и если *Z*-координата какой-либо из них окажется меньше высоты ландшафта в этой позиции, то считается, что произошло столкновение объекта с ландшафтом.

При обработке столкновения объекта с ландшафтом считается, что масса ландшафта близка к бесконечности, а значит, скорость его после столкновения близка к нулю. Поэтому по модулю скорость столкнувшегося с ним объекта по модулю не меняется, а меняется только ее направление. За новый вектор скорости берется вектор, полученный при отражении предыдущего вектора скорости от плоскости, содержащей треугольник, под который было зафиксировано проникновение баундинг-бокса. Отраженный вектор рассчитывается по формуле

$$R = S * N * 2 * N - S, \qquad (3.8)$$

November 14, 2008 12:17 AM -- 27

где R – отраженный вектор, S – вектор скорости до столкновения, N – нормаль к плоскости треугольника ландшафта, с которым зафиксировано столкновение. Причем $S * N$ – скалярное произведение векторов.

3.4. Физический модуль и мультиагентная система

Описанные выше методы позволяют объектам существовать в среде, представляющей собой упрощенный вариант среды с реальными физическими законами. По сути, объекты физического модуля (представленные классом *CMesh*, являющимся основным классом и в графическом модуле) – это «тела» агентов, оболочки, в которых они существуют в мультиагентной среде. Взаимодействия объектов, обрабатываемые физическим модулем, являются физическими взаимодействиями агентов между собой.

November 14, 2008 12:17 AM -- 28

Глава 4.
Визуализация мультиагентной среды и графический модуль

4.1. Задачи графического модуля

Для того чтобы программист, разрабатывающий мультиагентную систему, не тратил времени на создание средств, позволяющих ему наблюдать за действиями агентов в мультиагентной среде и отлаживать написанные им методы и алгоритмы, был разработан и реализован графический модуль (см. рис. 1.1).

Графический модуль основан на трехмерной графической библиотеке *OpenGL*. Графический модуль позволяет выводить на экран трехмерное изображение той части мультиагентной среды, которую видит с данной позиции наблюдатель. Библиотека *OpenGL* была выбрана из-за того, что она проста в изучении и использовании, а также *OpenGL* – всеплатформенная библиотека, что позволяет сделать визуализацию мультиагентной среды менее зависимой от конфигурации компьютера, на котором она происходит (то есть не обязательно нужно использовать трехмерный ускоритель последней модели, если это не нужно для выполнения данной задачи).

Для того чтобы использовать графический модуль, нужно через модуль - загрузчик получить указатель на интерфейс класса *CRenderer*. Этот класс содержит в себе данные и методы, используемые для рендеринга трехмерной графики.

Каждому агенту мультиагентной системы сопоставляется свой объект, являющийся «телом» агента, его представлением в мультиагентной среде. Графический модуль позволяет выводить на экран эти объекты, а также рисовать такие элементы самой мультиагентной среды, как рельеф местности.

4.2. Представление мультиагентной среды в графическом модуле

Все объекты, с которыми работает графический модуль, представлены как набор геометрических фигур. Основная такая фигура – это треугольник. С помощью сеток треугольников строятся все объекты мультиагентной среды. Сам треугольник состоит из трех вершин. Вершина описана классом *CVertex* (диаграмма 4.1). В каждой вершине содержится информация о ее координате относительно центра тяжести, о нормали к этой вершине, о ее цвете, о текстурных координатах этой вершины и о ее весовом коэффициенте. Треугольник был выбран наименьшей единицей представления геометрической формы, так как треугольник всегда лежит в одной плоскости, поэтому не возникает проблем при расчете нормали к грани.

И координата вершины в пространстве, и нормаль к этой вершине задаются в объектах класса *CPoint3D* (диаграмма 4.2). В этом классе содержится информация о координатах точки или вектора по трем осям – *X*, *Y* и *Z*. Класс *CPoint3D* используется и как точка в трехмерном пространстве, и как вектор. Как вектор этот класс обладает набором функций, позволяющих вычислить длину вектора, скалярное и векторное произведение двух векторов, нормализовать (привести к единичной длине) вектор. Также возможны операции с векторами, такие как сложение и вычитание векторов, умножение вектора на число.

У каждой вершины существует своя нормаль, и, несмотря на то, что все три вершины треугольника лежат в одной плоскости, не обязательно эти нормали должны быть коллинеарны друг другу и перпендикулярны этой плоскости (рис.4.1). Эти нормали задаются для того, чтобы при расчете освещения сымитировать гладкую скругленную поверхность. Объекты, форма которых близка к сферической, или имеет скругленные участки, представляется в виде набора треугольников, но эти треугольники не способны передать округленность формы. Для этого у каждой вершины объекта задаются нормали, при расчете освещения позволяющие сделать модель более гладкой на вид.

November 14, 2008 12:17 AM -- 30

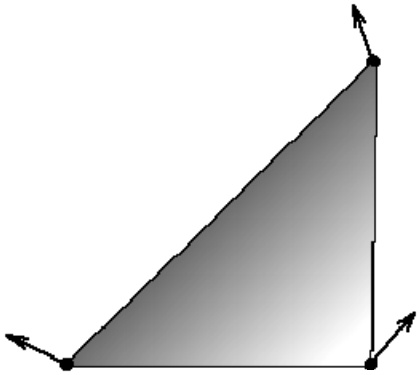


Рис.4.1. Нормали при вершинах треугольника

Цвет вершины может либо браться из глобальных настроек материала, либо задаваться отдельно для каждой вершины. В объектах класса *CColor* (диаграмма 4.3) хранится информация о цвете вершины. Цвет задается четырьмя параметрами – количество красного, зеленого, синего, а также альфа-канал (уровень прозрачности). Цвет (но не материал) у каждой вершины может быть свой.

Треугольник может быть не только освещен и окрашен в какой-либо цвет. На него может быть также наложена текстура. Текстура (от английского слова *texture* – ткань) представляет собой двумерную картинку, накладываемую на треугольник (рис.4.2).

Каждой вершине текстурированного треугольника соответствует пара текстурных координат (рис.4.3). Информация о текстурных координатах содержится в объектах класса *CTexCoord* (диаграмма 4.4). Текстурные координаты вершины – это точки на текстуре, по которым «вырезается» накладываемое на треугольник изображение.

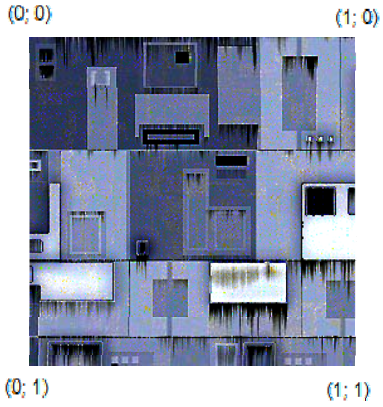


Рис.4.2. Текстура

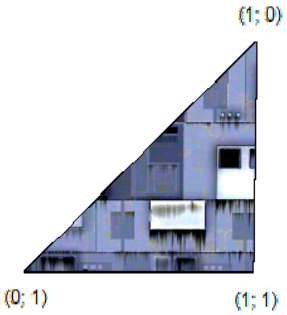


Рис.4.3. Текстурированный треугольник

November 14, 2008 12:17 AM -- 31

Сам треугольник описывается классом *CPolygon* (диаграмма 4.5). Максимальное количество вершин полигона (от английского *polygon* - многоугольник) – три, но может быть и две вершины (отрезок), и одна вершина (точка). Помимо информации о вершинах в объектах класса *CPolygon* содержится информация об идентификаторе текстуры, наложенной на треугольник. Если этот идентификатор равен нулю, треугольник рисуется без текстуры.

Все объекты мультиагентной среды построены из треугольников (рис.4.4). Класс *CMesh* (диаграмма 4.6) содержит в себе информацию об объекте – положение объекта в мировой системе координат (положение центра тяжести объекта), массив треугольников, задающих геометрическую форму объекта, массив источников света, закрепленных за объектом и перемещающихся вместе с ним, текущую матрицу преобразований объекта (эта информация хранится в классе *CBone*, диаграмма 4.7), а также информацию о материале, из которого сделан объект.

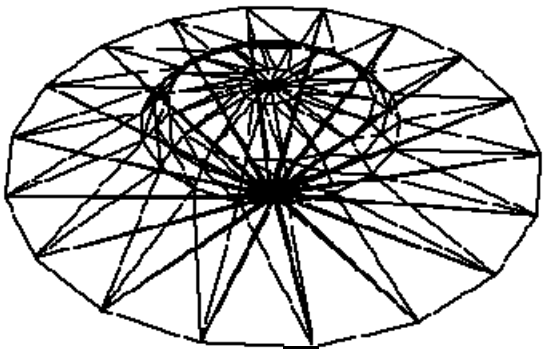


Рис.4.4. Пример объекта мультиагентной среды

При рендеринге объектов графический модуль работает с двумя массивами переменных типа *CMesh*. В одном из них хранится информация о первоначальном положении вершин объекта, а в другой записываются объекты с координатами вершин, преобразованными с помощью текущей матрицы преобразований. Эти объекты и рисуются на экране. Матрица преобразований представляет собой квадратную матрицу размера 4*x*4. Эта матрица хранится в объектах класса *CMatrix*4*x*4 (диаграмма 4.8), принадлежащих классу *CBone*. Перед рендерингом все вершины всех объектов умножаются на эту матрицу:

$$\begin{vmatrix} M_{11} & M_{21} & M_{31} & M_{41} \\ M_{12} & M_{22} & M_{32} & M_{42} \\ M_{13} & M_{23} & M_{33} & M_{43} \\ M_{14} & M_{24} & M_{34} & M_{44} \end{vmatrix} * \begin{vmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{vmatrix} = \begin{vmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{vmatrix} \qquad (4.1)$$

November 14, 2008 12:17 AM -- 32

Где **M –** матрица преобразования (M_{11}, …, M_{44} – элементы этой матрицы); X_0, Y_0, Z_0 – координаты исходной вершины; X_1, Y_1, Z_1 – координаты преобразованной вершины.

Матрица **M** является матрицей поворота объекта, и все вращения объекта происходят посредством изменения этой матрицы. При повороте создается матрица преобразования, соответствующая повороту на заданный угол, и происходит матричное умножение текущей матрицы преобразований объекта на эту матрицу. Использование матриц преобразования поворота нужно для того, чтобы в любой момент знать положение каждой вершины объекта. Информация о положении вершин объекта нужна для проведения точной проверки на столкновение между двумя объектами, когда рассматриваются не аппроксимированные формы объектов, а их точные формы, описанные массивом треугольников в классе *CMesh*. Если использовать вращения объектов, реализованные в *OpenGL*, информация о преобразованных вершинах останется недоступной. Если в определении координат вершин объекта после вращения нет надобности, то возможно включение стандартных алгоритмов вращения объектов с использованием функций *OpenGL*.

Каждый объект сделан из своего материала. Физические свойства материала описаны в объектах класса *CPhysMaterial* (диаграмма 3.4), с которыми работает физический модуль. Графические же свойства материала, влияющие на его изображение, описаны в объектах класса *CGraphMaterial* (диаграмма 4.9). В этом классе хранятся значения коэффициента рассеивания лучей света, коэффициента отражения, хранится цвет всего объекта, а также информация о том, излучает ли объект свет, и информация о том, как цвет задается для объекта – для всех его вершин сразу (при этом цвет берется из материала) или для каждой вершины отдельно (при этом цвет берется из объектов класса *CVertex* для каждой вершины).

Агент в мультиагентной среде может быть представлен не только как самостоятельный объект, но и как иерархическая структура объектов. При этом выстраивается дерево объектов – каждый объект класса *CMesh* содержит в себе указатель на объект – родитель и массив указателей на объекты – потомки. Для связи двух объектов класса *CMesh* служит класс *CBone* (от английского *bone* - кость) (диаграмма 4.7), в котором хранится позиция центра тяжести объекта относительно объекта – родителя (если объект является корнем дерева, то позиция указывается относительно мировой системы координат), а также матрица преобразования вращения относительно объекта – родителя (если объект является корнем дерева, то вращение производится относительно мировой системы координат).

Такая структура построения дерева объектов позволяет создавать скелетные модели и анимировать их.

November 14, 2008 12:17 AM -- 33

4.3. Ландшафт

Важной частью мультиагентной среды является рельеф местности (ландшафт), где живут и действуют агенты. Ландшафт также построен из треугольных граней. Ландшафт задается в виде массива высот. При рендеринге из этого массива строится регулярная сетка высот, которая затем триангулируется (то есть вершины соединяются в треугольники) (рис.4.5).

Массив высот описан в объектах класса *CFloatMatrix* (диаграмма 4.10). Этот массив называется также картой высот. Из этого класса унаследован класс *CHorizont* (диаграмма 4.11), содержащий в себе дополнительную информацию, делающую карту высот рельефом. В этом классе хранится размер одного тайла ландшафта (от английского слова *tile* – черепица, кафель. Используется в значении «клетка». Размер одного тайла – расстояние между двумя ближайшими проекциями вершин на горизонтальную плоскость). Также в этом классе содержится информация об идентификаторах глобальной текстуры, накладываемой на ландшафт, и текстуры детализации.

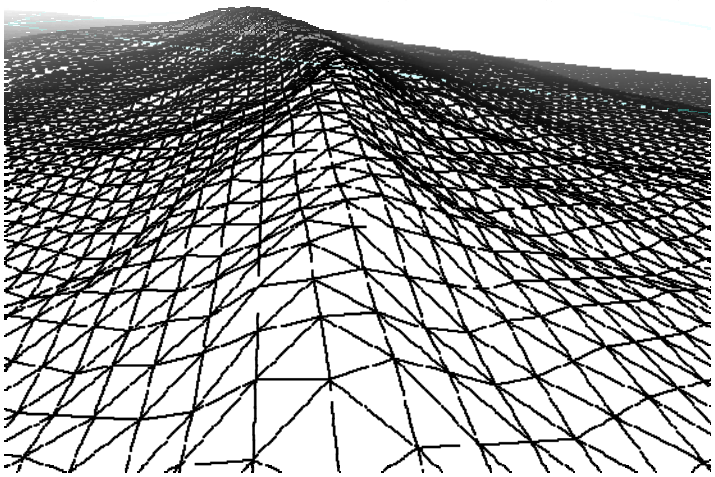


Рис.4.4. Ландшафт

Каждому массиву высот ландшафта сопоставляется такой же по размерам двумерный массив, содержащий нормали к вершинам ландшафта. Этот массив описан в классе *CPointMatrix* (диаграмма 4.12). Нормали к вершинам ландшафта считаются при инициализации ландшафта, после загрузки карты высот ландшафта из файла (алгоритмы описаны в разделе 4.5 – Подготовка данных для рендеринга).

November 14, 2008 12:17 AM -- 34

Общая информация о ландшафте хранится в классе *CLandscape* (диаграмма 4.13). Ландшафт может состоять из нескольких карт высот, то есть содержать в себе несколько экземпляров класса *CHorizont*. Каждой карте высот соответствует свой массив нормалей, то есть экземпляр класса *CPointMatrix*.

4.4. Вспомогательные графические классы

Помимо классов, описывающих различные геометрические объекты, существует также набор классов, необходимых для работы графического модуля, но не влияющих на работу мультиагентной системы. Эти классы описывают источники освещения, камеры, текстуры. Также существуют классы для описания атмосферы и воды.

Важным элементом при визуализации мультиагентной среды является освещение. Всего в мультиагентной среде может быть создано и размещено до восьми источников освещения (эти ограничения накладываются в библиотеке *OpenGL*, которая поддерживает максимум восемь источников освещения). Источники освещения описываются в объектах класса *CLightSource* (диаграмма 4.14). Источники освещения в мультиагентной среде могут быть трех типов. К первому типу относятся бесконечно удаленные источники, для которых важно лишь направление на источник света, а сам источник, по сути, является плоскостью, излучающей свет. Ко второму типу относятся точечные источники освещения, испускающие лучи света во все стороны с одинаковой интенсивностью. Для этих источников важна их позиция. К третьему типу относятся точечные направленные источники света. Для них задается не только позиция относительно мировой системы координат, но и вектор направления лучей света и угол максимального рассеивания лучей (рис.4.6). Для каждого источника освещения можно настроить цвет лучей, испускаемых им.

Источник освещения может находиться в глобальной системе координат или быть закрепленным за каким-либо объектом мультиагентной среды. В этом случае источники света, закрепленные за объектом, описываются в экземплярах класса *CMesh*, то есть объекты класса *CLightSource* хранятся в массиве источников освещения объекта. При изменении позиции объекта или его ориентации в пространстве источник освещения также меняет позицию и направление (если оно для него задано). При задании такого источника освещения позиция его указывается относительно центра тяжести объекта мультиагентной среды, за которым он закреплен.

November 14, 2008 12:17 AM -- 35

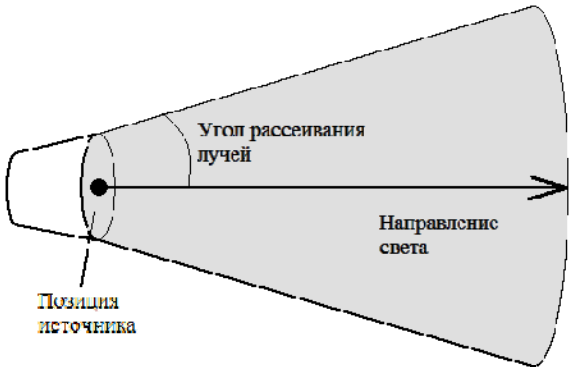


Рис.4.6. Точечный направленный источник освещения

Камеры наблюдения являются важным элементом работы с графическим модулем, так как они позволяют наблюдателю перемещаться по мультиагентной среде, наблюдая за ее различными частями с разных позиций и под разными углами. Параметры камеры наблюдения описаны в классе *CCamera* (диаграмма 4.15). Камера задается своей позицией в мировой системе координат, точкой в мировой системе координат, за которой камера наблюдает (камера фокусируется на эту точку, и она является точкой фокусировки для камеры), а также вектором, задающим ось аппликат локального ортогонального базиса камеры, фактически, этот вектор задает ориентацию камеры относительно локальной оси координат, задаваемой вектором направления наблюдения (это вектор от точки позиции камеры до точки, на которую камера сфокусирована) (рис.4.7).

Также для камеры задается «пирамида видимости» - область, по форме являющаяся усеченной четырехугольной пирамидой. Это та область, в пределах которой объекты являются видимыми, а за ее пределами – нет. Эта пирамида задается двумя плоскостями отсечения – ближней и дальней (отсекается все, что находится ближе к наблюдателю, чем ближняя плоскость отсечения, и дальше, чем дальняя плоскость отсечения) (рис.4.7). Эти плоскости перпендикулярны вектору направления наблюдения и задаются вещественными числами, определяющими расстояние от точки наблюдения до плоскости. Также для полного определения этой пирамиды на ближней плоскости отсечения задается прямоугольник с центром в точке пересечения этой плоскости с вектором направления наблюдения. Ребра «пирамиды видимости» (английский термин, обозначающий эту пирамиду - *frustum*) лежат на прямых, проходящих через точку позиции камеры и вершины этого прямоугольника.

November 14, 2008 12:17 AM -- 36

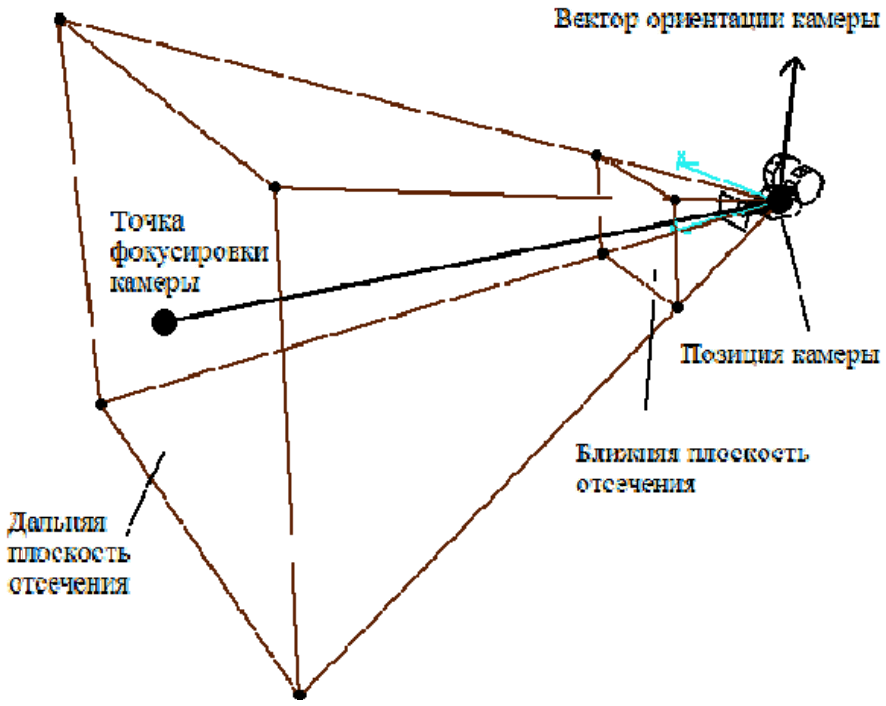


Рис.4.7. Камера наблюдения

Графический модуль содержит в себе функции для работы с атмосферой и с водой. Настройки атмосферы задаются в объекте класса *CAtmosphere* (диаграмма 4.16), находящемся среди данных графического модуля. Этот класс содержит в себе информацию о цвете тумана, о константе густоты тумана (английский термин *density*), используемой в функции расчета интенсивности тумана в зависимости от удаления, о способе расчета интенсивности тумана (в зависимости от удаленности точки от наблюдателя или в зависимости от высоты точки – объемный туман). Также в этом классе описаны настройки отображения неба – цвет неба, идентификатор текстуры, накладываемый на прямоугольник, изображающий небо, скорость изменения текстурных координат вершин прямоугольника (это нужно для создания эффекта движения облаков, если они изображены на текстуре). Прямоугольник, изображающий небо, расположен в горизонтальной плоскости на постоянном удалении от камеры, и всегда перемещается вместе с ней.

Параметры отображения воды описаны в экземпляре класса *CWater* (диаграмма 4.17), также находящемся среди данных графического модуля. В этом классе задается высота уровня моря, цвет воды и идентификатор текстуры, накладываемой на поверхность воды, а также размеры сетки треугольников, задающих расположение воды (эта сетка аналогична сетке

November 14, 2008 12:17 AM -- 37

треугольников рельефа местности, только высоты всех ее вершин одинаковы) и скорость изменения текстурных координат этих вершин (для создания эффекта течения воды).

Важным элементом создания изображения являются текстуры. Текстура – это двумерная картинка, накладываемая на рисуемую грань для придания этой грани видимости того, что она сделана из какого-либо материала (рис.4.2 и рис.4.3). Текстура описывается в классе *CColorMatrix* (диаграмма 4.18). В этом классе хранится двумерный массив текселей (элементов текстуры, содержащих информацию о цвете. При наложении текстуры на поверхность тексель отличается от пикселя, хотя их задачи схожи – они несут в себе информацию о цвете одного элемента изображения). Тексель представляет собой массив из четырех вещественных чисел, принимающих значение от нуля до единицы, обозначающих количество красного, зеленого и синего в цвете, а также альфа – канал (прозрачность текселя, если этот показатель равен нулю, то тексель абсолютно прозрачный, а если единице, то - абсолютно непрозрачный). Также текстура содержит в себе свой идентификатор – целое число, под которым она зарегистрирована в *OpenGL*.

4.5. Подготовка данных для рендеринга

Графический модуль содержит множество настроек, тем или иным образом влияющих на изображение, выдаваемое им. Большинство этих настроек описываются в специальном инициализационном файле с расширением *.ini*. В этом файле, прежде всего, описаны настройки окна, в котором будет происходить отображение. Это может быть либо просто окно, либо полноэкранный режим (тогда задаются не размеры окна, а разрешение экрана в этом режиме).

Также в инициализационном файле задаются настройки атмосферы и воды, задаются камеры, используемые при наблюдении за мультиагентной средой (поддерживается использование нескольких камер с возможностью переключения между ними), источники освещения, находящиеся в мультиагентной среде (до восьми источников освещения), задаются параметры шрифта, используемого при отображении текста на экране (инициализируются такие параметры, как толщина шрифта, его размер, наклон, перечеркнут ли шрифт, а также задается сам используемый шрифт).

Также в инициализационном файле задается информация о загрузке текстур – имя файла формата *.bmp*, в котором находится двумерное изображение текстуры, а также идентификатор, который присваивается текстуре после ее инициализации. При инициализации текстуры

November 14, 2008 12:17 AM -- 38

в *OpenGL* задается этот идентификатор, и дальше вся работа с текстурой через *OpenGL* идет только с использованием этого идентификатора. Поэтому в текстурированных треугольниках (см. пункт 4.2) хранится только информация об идентификаторе текстуры.

В инициализационном файле задается также информация о загружаемом ландшафте – количество загружаемых карт высот, а также уровень детализации, используемый при рендеринге ландшафта (см. пункт 4.7). При загрузке самой карты высот указываются такие параметры, как имя файла формата *.bmp*, содержащего карту высот (высота в конкретной вершине определяется градиентом серого – чем цвет ближе к белому, тем ландшафт в данной точке выше, а чем ближе к черному, тем ниже; каждому пикселю в файле карты высот соответствует своя вершина рельефа), размер одного тайла, информация об общем цвете ландшафта, а также информация о наложении текстур на ландшафт – идентификаторы глобальной текстуры и текстуры детализации, а также количество повторений наложения текстуры детализации на глобальную текстуру.

После того, как карта высот загружена, происходит расчет нормалей для каждой вершины и заполнение массива нормалей для этой карты высот. Для этого рассчитываются нормали для шести треугольников, которым принадлежит данная вершина (рис.4.8) (если вершина находится на каком-либо краю ландшафта, то недостающие нормали считаются направленными вертикально вверх). Нормаль к треугольнику считается как векторное произведение двух векторов, задающих этот треугольник. Векторное произведение считается по формуле (4.2):

$$\begin{aligned} X_3 &= Y_1 * Z_2 - Z_1 * Y_2 \\ Y_3 &= X_1 * Z_2 - X_1 * Y_2 \\ Z_3 &= X_1 * Y_2 - X_2 * Y_1 \end{aligned}, \qquad (4.2)$$

Где X_1, Y_1, Z_1 и X_2, Y_2, Z_2 – координаты двух исходных векторов, задающих треугольник, а X_3, Y_3, Z_3 – координаты вектора, перпендикулярного плоскости, содержащей исходный треугольник.

Нормаль к данной вершине берется как векторная сумма получившихся шести нормалей к треугольникам, содержащим эту вершину, а затем нормализуется.

November 14, 2008 12:17 AM -- 39

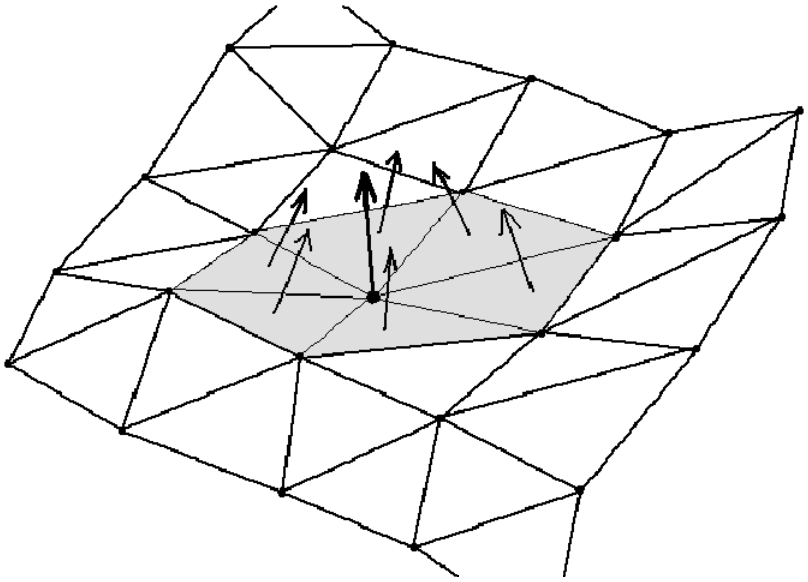


Рис.4.8. Расчет нормали к вершине ландшафта

С помощью модуля хранения данных (см. пункт 1.4) в память загружаются все объекты класса *CMesh* и *CPolygon*, которые хранятся в файлах внутреннего формата модуля хранения данных. Также подготавливаются дополнительные массивы объектов классов *CMesh* и *CPolygon*, такие же по размерам, как и основные. Затем графический модуль подготавливает эти массивы для рендеринга – для каждого элемента главного массива переменных типа *CMesh* производится настройка указателей на массивы треугольников (объектов класса *CPolygon*), соответствующих сеткам полигонов этих объектов. Подготавливаются дополнительные массивы – в них заводятся буферные экземпляры классов *CMesh* и *CPolygon*, в которые будет записываться информация о вершинах объектов, трансформированных с помощью матриц преобразования.

4.6. Рендеринг объектов мультиагентной среды

При рендеринге объектов мультиагентной среды графическому модулю передается два массив объектов типа *CMesh* – массив, содержащий исходные координаты вершин объектов, и массив, в котором будут храниться вершины, измененные с помощью матрицы преобразований объекта.

Перед рендерингом графический модуль трансформирует вершины треугольников всех объектов, которые нужно отобразить на экране. Преобразование вершин происходит по формуле (4.1). После этого объекты класса *CMesh* и объекты класса *CPolygon* с измененными вершинами передаются в функцию рендеринга объектов.

November 14, 2008 12:17 AM -- 40

Качество графики, используемой при рендеринге, зависит от выставленного в инициализационном файле уровня рендеринга – параметра, отвечающего за функции, используемые при отображении мультиагентной среды.

Самым низким уровнем рендеринга является уровень, на котором рисуются лишь каркасные модели всех объектов. Для каждой вершины задается только ее положение в мировой системе координат, а также ее цвет. Вершины в треугольниках соединяются линиями.

Следующий уровень рендеринга – это уровень закрашенных моделей. Для каждой вершины также задается ее позиция и цвет, но теперь на экране рисуются закрашенные треугольники.

На третьем уровне рендеринга включается освещение. Для вершин треугольников, из которых состоят объекты мультиагентной среды, становятся важными их нормали.

Следующие два режима рендеринга используют текстуры при отображении, поэтому для каждой вершины теперь задаются текстурные координаты. В первом из этих режимов используются просто текстуры, а во втором подключаются такие эффекты, как линейная фильтрация текселей при размерах текселя, больших, чем размеры пикселя, и *mip-mapping* при размерах текселя, меньших, чем размеры пикселя (*mip* – это аббревиатура трех латинских слов, *multim im parvo*, означающих «многое в малом»). При мип-маппинге создается набор текстур с линейными размерами, в два раза меньшими, чем у предыдущей текстуры. Отсюда возникает ограничение в *OpenGL* на размеры текстуры – они должны являться степенями двойки.

В последнем режиме рендеринга используются функции, выходящие за рамки стандартной библиотеки *OpenGL* и поддерживающиеся уже конкретными видеоадаптерами. Это такие функции, как использование объемного тумана или применение мультитекстурирования (наложения на грань нескольких текстур). Эти функции не входят в стандартный набор функций библиотеки *OpenGL*, а входят в различные расширения этой библиотеки, поддерживаемые конкретными видеоускорителями. Поэтому на разных платформах часть этих функций может не работать.

4.7. Рендеринг ландшафта. Уровни детализации

Ландшафт содержит в себе наибольшее количество треугольников, поэтому рендеринг ландшафта занимает наибольшее количество времени. Для оптимизации рендеринга ландшафта используются алгоритмы построения уровней детализации.

November 14, 2008 12:17 AM -- 41

При рендеринге ландшафта треугольники, находящиеся на большом удалении от позиции наблюдателя, по своим размерам, занимаемым ими на экране, становятся близки к одному или нескольким пикселям. Такие маленькие треугольники не имеет смысла рисовать, так как рассмотреть такой треугольник в деталях невозможно. Поэтому на значительном удалении от наблюдателя рисуются треугольники большего размера, которые позволяют сэкономить время, тратящееся на вызовы большого числа функций для задания вершин треугольников маленьких размеров.

Метод увеличения размеров треугольников, удаленных от камеры, называется методом построения уровней детализации (английский термин *LoD – Levels of Detail*, уровни детализации).

При расчете уровней детализации точки, задающие «пирамиду видимой области» камеры наблюдения, проецируются на горизонтальную плоскость, и по точкам проекции на этой плоскости выбирается прямоугольник наименьшей площади. Этот прямоугольник (рис.4.9) задает область ландшафта, видимую с данной позиции камеры. Для этой области и рассчитываются уровни детализации.

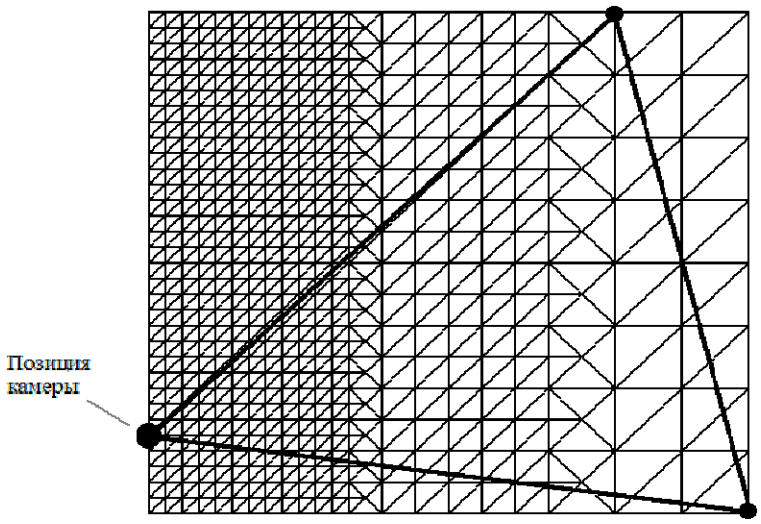


Рис.4.9. Уровни детализации для карты высот

При рендеринге ландшафта выбирается ось координат на горизонтальной плоскости, вдоль которой будут рассчитываться уровни детализации (это может быть либо ось абсцисс, либо ось ординат). Затем полученный прямоугольник вдоль выбранной оси разбивается на количество частей, равных

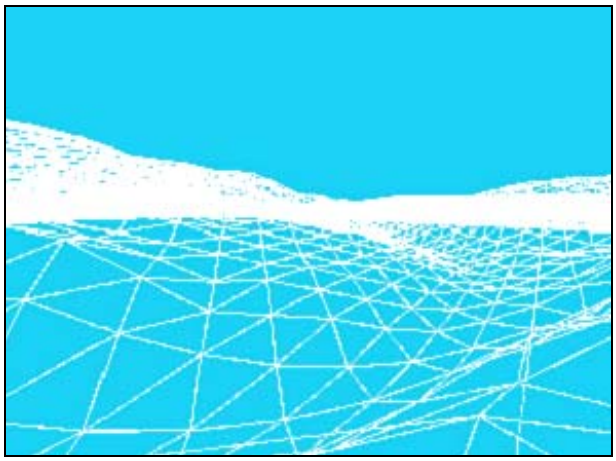
$$N = 1 + log_2 L, \qquad (4.3)$$

November 14, 2008 12:17 AM -- 42

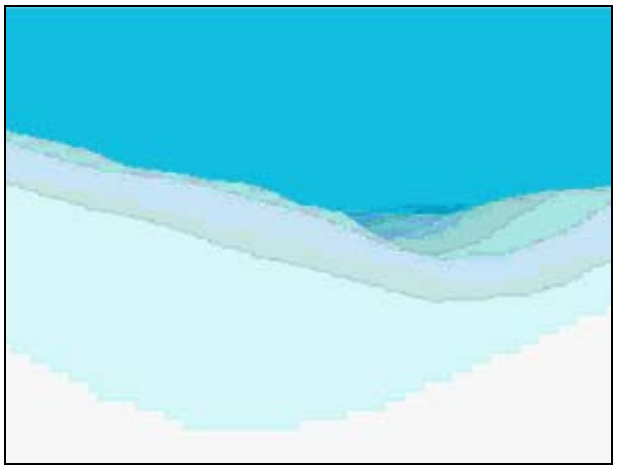
где N – количество частей, на которые разбивается прямоугольник, а L – значение показателя уровня детализации. Это значение должно являться степенью двойки, чтобы при расчете значения N по формуле (4.3) получались целые числа.

Дальше идет прорисовка всех полученных частей прямоугольника, при этом первой рисуется та часть, которая ближе всего к наблюдателю. Размер треугольников в этой части равен исходному размеру треугольников ландшафта. Затем происходит рендеринг остальных частей прямоугольника видимости, при этом рендеринг идет в порядке удаления частей от наблюдателя, и для каждой новой части прямоугольника линейные размеры треугольников, используемых при визуализации, возрастают вдвое. Линейные размеры треугольников при визуализации последней части прямоугольника равны исходным размерам, умноженным на показатель уровня детализации.

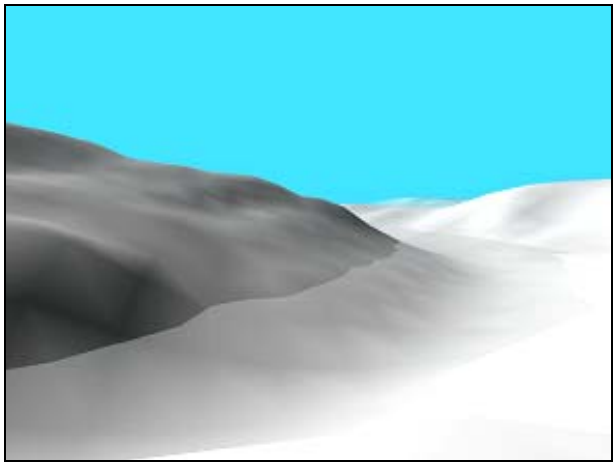
Влияние выбора уровня детализации на качество изображения представлено на рис. 4.10.



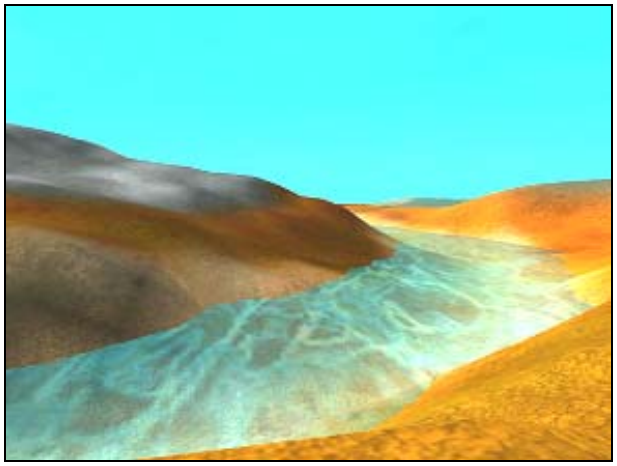
а) Проволочная модель



б) С закраской граней



в) С использованием освещения



г) С мультитекстурированием

Рис. 4.10. Различные уровни детализации ландшафта.

November 14, 2008 12:17 AM -- 43

4.8. Графический модуль и мультиагентная система

Графический модуль позволяет визуализировать агенты, входящие в состав мультиагентной системы, а также статичные детали самой мультиагентной системы. Агенты существуют в мультиагентной среде, являющейся неким геометрическим пространством, и сами агенты являются геометрическими объектами, задающими их внешний вид в мультиагентной среде. Эти геометрические объекты являются, по сути, «телами» агентов в мультиагентной среде, их представлениями в ней.

Также графический модуль позволяет визуализировать рельеф местности, являющийся важной частью мультиагентной системы и среды. При рендеринге ландшафта применяется техника оптимизации с использованием уровней детализации ландшафта.

Глава 5.
Система подключения языков программирования

5.1. Компоновщик языков

Компоновщик языков – это независимый модуль Системы, ответственный за межъязыковое взаимодействие. Через модуль-компоновщик также идет работа с объектами, реализованными на внешних (подключаемых) языках системы и межъязыковое взаимодействие. Разработанный интерфейс компоновщика позволяет подключать другие языки программирования. В настоящее время поддерживаются Smalltalk как встроенный транслятор, а также откомпилированные модули (EXE, DLL) на любом другом языке. Необходимость Компоновщика вызвана различными форматами вызова процедур в различных языках программирования, различной трактовкой переменных и возможными более глубокими различиями в структуре языков.

Мультиагентная система связана с комплексом с помощью Сервера агентов, но модули системы имеют право работать с агентами непосредственно через модуль-компоновщик.

Модуль-компоновщик реализуется классом *CLinker* и сопутствующими. Он хранит в себе индексированный массив описаний зарегистрированных в нём классов *SClass* и индексированный массив строк. Любая строка, которая используется в модулях, связанных с Компоновщиком, и является неизменяемой (прежде всего все названия объектов, классов, функций и т.д.) должна быть зарегистрирована в нём. Если такая строка уже есть в массиве строк, то выдаётся её номер, если нет – то добавляется и тоже выдаётся её новый номер. Таким образом удаётся значительно оптимизировать работу Системы, так как вместо посимвольного сравнения строк, сравнивается два целых числа. Описания классов хранят в себе своё имя в виде целого числа, массив классов – родителей, из которых этот класс унаследован, массив описаний функций *Func* и описаний статических функций *StaticFunc*. Каждый объект *Func* и объект *StaticFunc* хранят в себе описание своих параметров *CParamsTypes*, своё название, название своего класса, так называемые селекторы класса и этой функции в своей Виртуаль-

ной Машине (ВМ) и указатель на саму ВМ. Объект *CParamsTypes* хранит в себе массив типов параметров, а если параметр – объект, то тогда ещё и *smart pointer* (дословно – «умный» указатель) на описание класса объекта *SClass*. Указатель на ВМ – это указатель на интерфейс *IVM*, который содержит лишь самые необходимые Компоновщику от любой ВМ функции: функции получения селекторов классов и функций по имени и собственно функции вызова. «Селекторные» функции имеют права возвращать любое целое число. Это число при вызове будет передаваться вместе с именем чтобы можно было не делать поиск внутри ВМ, а воспользоваться Селекторами. Итак, когда компоновщик вызывает функцию по имени у некоторого объекта, этот объект выдаёт *smart pointer* на свой *SClass* (Этот указатель назван *SClassRef*). По этому указателю берётся *SClass* и ищется *CFunc* с нужным именем и вызывается функция ВМ, которой передаются указатель на объект (если функция нестатическая), имя и селектор класса и функции и ссылка на параметры. У объекта можно получить его *SClassRef* потому что любой объект, способный к межъязыковому взаимодействию, обязан наследоваться из класса *CObj*, обеспечивающему кроме хранения информации об объекте для Компоновщику, ещё и *Garbage Collection* – «сборку мусора», то есть удаление объектов, но которые нет больше ссылок в Системе. Все ссылки на объекты Межъязыкового Взаимодействия (объекты МВ, фактически наследованные из *CObj* и зарегистрированные в Системе) должны храниться в специальных *smart pointer* на *CObj – CObjRef*. Такой указатель фиксирует количество ссылок на объект, а когда ссылок не осталось – объект удаляется.

5.2. Поддержка языка Smalltalk

Для поддержки языка Smalltalk в комплексе был реализован *транслятор* этого языка и интегрированная среда разработки. Транслятор позволяет *компилировать* исходные тексты программы в байт-код, используемый при исполнении методов объектов, созданных на этом языке. Это происходит в два прохода: первый — лексический анализ с помощью конечного автомата, второй — синтаксический анализ методом рекурсивного спуска (рис. 5.1). В интегрированную среду входят также редактор исходных текстов на языке Smalltalk и отладчик. Реализованы такие функции, как подсветка синтаксиса языка, пошаговое выполнение программы, просмотр локальных переменных, использование точек останова программы. Реализована возможность *динамической рекомпиляции* — изменения исходного текста программы прямо во время ее работы.

November 14, 2008 12:17 AM -- 46

Задача лексического анализа (рис. 5.2) – разобрать исходную программу на блоки-лексемы, с которыми работать значительно удобнее, чем с исходным текстом. Обычно лексема имеет два поля – тип и значение, часто к ним прибавляются ссылки на таблицу имен и «координаты» лексемы внутри исходного текста, чтобы облегчить отчет о синтаксических ошибках программы.

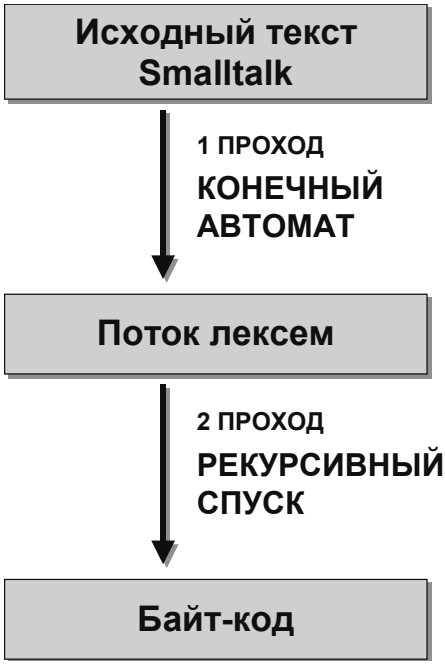


Рис.5.1. Общий характер трансляции с языка Smalltalk.

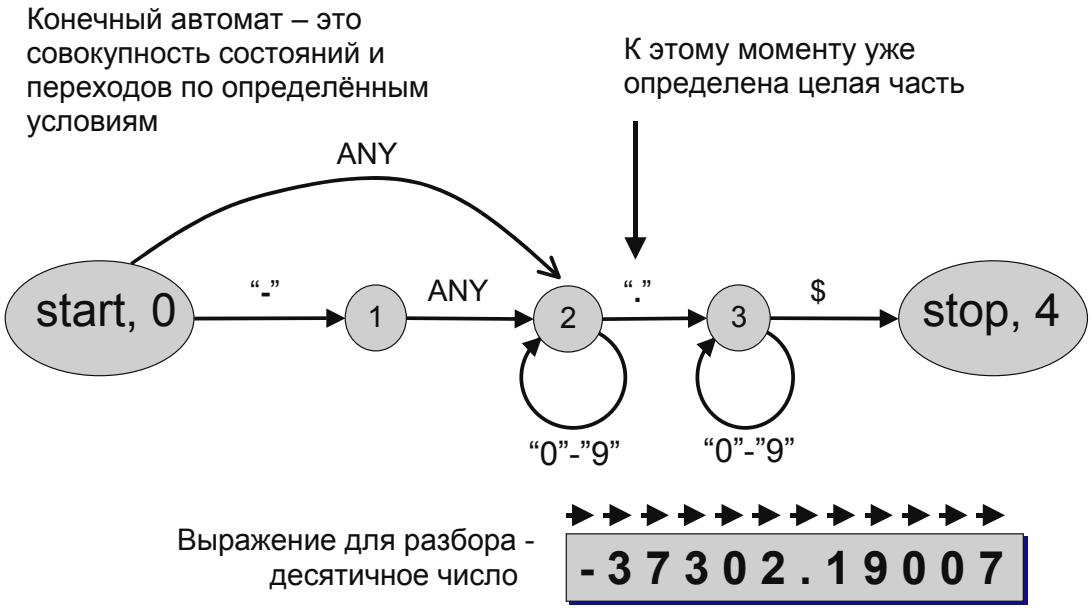


Рис.5.2. Фаза лексического анализа.

November 14, 2008 12:17 AM -- 47

Задача фазы синтаксического анализа (рис.5.3) – сформировать дерево программы и на основе него результирующий байт-код. В данной работе используется синтаксический анализатор рекурсивного спуска. Такие анализаторы легко пишутся по грамматике в стандартной форме Бэкуса-Наура, но их непросто модифицировать при ее изменении. Действующий анализатор опирался на стандарт языка Smalltalk, содержащий описание грамматики в форме Бэкуса-Наура.

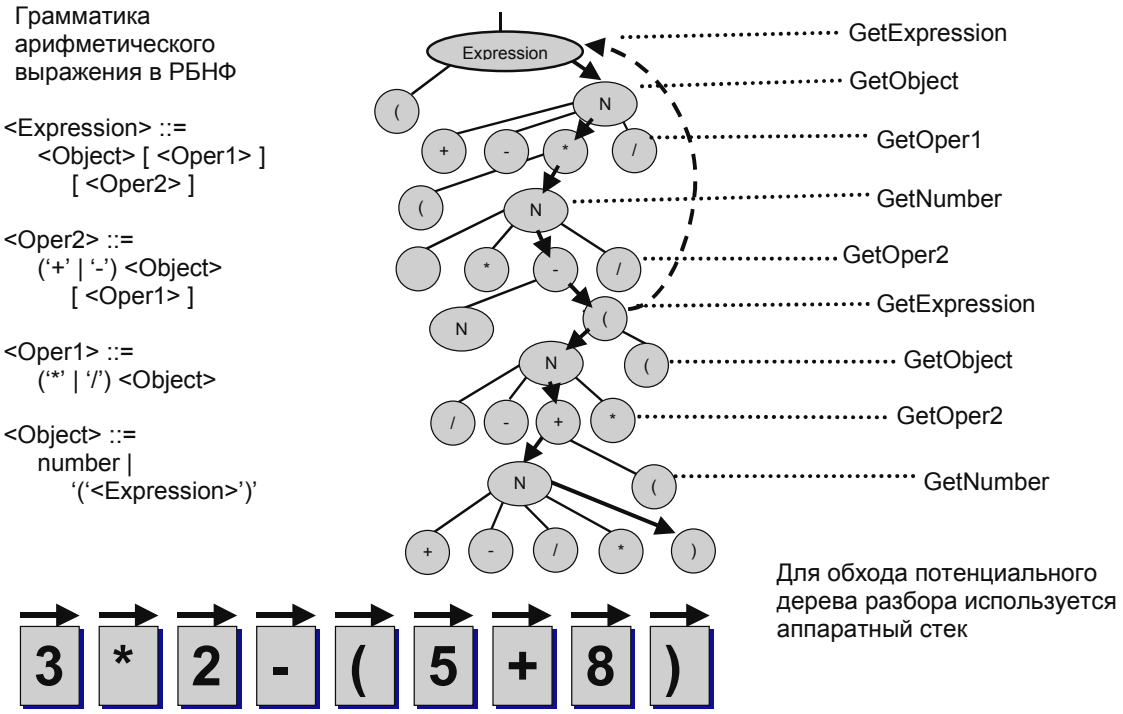


Рис.5.1. Фаза синтаксического анализа.

Задача виртуальной машины – выполнять байт-код, сформированный синтаксическим анализатором и таким образом связывать программы на языке Smalltalk с другими модулями, используя Компоновщик языков. Виртуальная машина поддерживает отладку программы, ее прерывание, просмотр переменных и ее обновление при динамической рекомпиляции.

November 14, 2008 12:17 AM -- 48

Глава 6.
Интегрированная среда разработки для языка Smalltalk

6.1. Возможности Интегрированной среды разработки

В процессе создания мультиагентной системы основной задачей является описание поведения агентов на языке моделирования и отладка написанного кода. Конечно, можно пользоваться «подручными» средствами, такими как стандартный текстовый редактор, а об ошибках судить только по поведению агента, но такой подход нельзя назвать оптимальным. Он не просто не даёт преимуществ пользователю комплекса, но и, наоборот, лишь заставляет его отвлекаться на второстепенные детали.

Интегрированная среда разработки (ИСР) – это набор средств для написания кода интеллекта агента и отладки этого кода. Она включает в себя текстовый редактор с функцией подсветки синтаксиса. Текстовый редактор позволяет выполнять стандартные операции над файлами и текстом, такие как набор текста, копирование и вставка блоков текста, поиск и замена в тексте, открытие и сохранение файлов в формате "*txt*". При чём имеется возможность одновременно редактировать и отлаживать один и более файлов. Когда эта функция подсветки синтаксиса активирована, текст анализируется, и отдельные слова подсвечиваются различными настраиваемыми цветами в зависимости от их значения этих слов. Это могут быть слова, обозначающие любые виды объектов, или любые виды методов, или константы. В режиме отладки ИСР позволяет ставить и удалять точки остановок, исполнять текст по шагам, входить в процедуры и выходить из них одной кнопкой, и просматривать текущие значения локальных переменных. Если переменная – объект, написанный на языке *Smalltalk*, то можно здесь же посмотреть значения его переменных, развернув его, как ещё одну ветку дерева локальных переменных. Ещё одной функцией ИСР является возможность вставки шаблонов для наиболее распространённых и однотипных элементов языка, таких как декла-

November 14, 2008 12:17 AM -- 49

рация класса. И наконец, ИСР позволяет одной кнопкой скомпилировать активный файл или все открытые файлы, а компилятор ещё динамически добавит изменения в Систему.

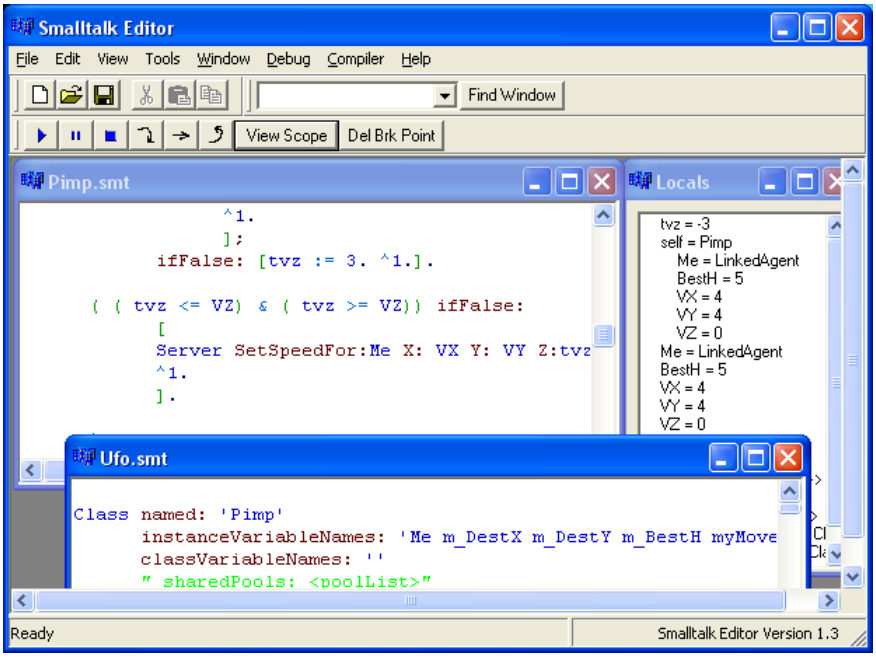


Рис.6.1. Интегрированная среда разработки для языка *Smalltalk*.

6.2. Реализация Интегрированной среды разработки

Среда разработки состоит из двух основных частей: Редактор и Модуль ИСР. Редактор создан с использованием *Borland C++ Builder* и представляет собой *DLL*, которая не может быть непосредственно загружена в Систему. Редактор загружается Модулем ИСР и запускается в отдельном потоке *Windows*, что позволяет производить процесс отладки независимо от основного потока, который, в котором работает большинство остальных компонентов, в частности, ответственных за работу агентов и развитие виртуальной среды. Поэтому во время отладки основной поток останавливается, а Редактор-отладчик продолжает работать. Модуль ИСР работает с Редактором через интерфейс *IBCEditor*, который позволяет в достаточной степени управлять Редактором. Для реализации межпроцессного взаимодействия используется механизм Отложенного Вызова Функции, который заключается в том, что в одном процессе сохраняется информации о функции, которая должна быть вызвана, а в другом уже вызывается. Функции могут быть как глобальными, статическими, так и функциями какого-либо класса. Механизм Отложенного Вызова используется через интерфейс *IDelayedExecutor* и реализуется классом *CThreadGate*. Редактор также может вызывать функции Модуля ИСР с

November 14, 2008 12:17 AM -- 50

помощью интерфейса *ISystem*. Таким образом, Редактор и Модуль ИСР работают совместно. Например, подсветка синтаксиса полностью реализована в Редакторе, а дерево локальных переменных формируется в Модуле ИСР, который получает исходные данные из Модуля Отладчика *Smalltalk*, который фактически является частью Модуля Виртуальной Машины *Smalltalk*, но не обязателен. Процедуры управления исполнением, такие как пауза или исполнение по шагам, Модуль ИСР фактически «переадресует» в Модуль Отладчика *Smalltalk*, а команды компиляции – в Модуль Компилятора, добавляя лишь название файла.

Несмотря на удобство Интегрированной среды разработчика, некоторые команды можно выполнить лишь из консоли, а вся текущая информация о процессе работы системы должна выводиться и выводится именно в неё. Поэтому её тоже можно считать частью Интегрированной среды разработчика.

November 14, 2008 12:17 AM -- 51

Глава 7. Выводы

В рамках данного проекта были разработаны:

– Структура взаимодействия модулей комплекса,

– Сервер Агентов,

– Модули визуализации и взаимодействия агентов со средой,

– Виртуальная машина, компилятор *Smalltalk* и модуль-компоновщик,

– Интегрированная среда разработки,

– Различные служебные модули.

7.1. Направления дальнейшей разработки

– Разработка мультиагентных систем;

– Разработка компонентов:

  – Поддержки других языков программирования,

  – Модули расширения,

  – Графический модуль на основе *DirectX*,

  – Другие модули;

– Разработка игровых программ.

Список литературы

1. Смольянинова В.А. Справочник по языку *Smalltalk*. Изд-во МИРЭА, 2000.
2. Д.Грис. Конструирование компиляторов для цифровых вычислительных машин. М., Мир, 1974.
3. А.Ахо, Р.Сети, Д.Ульман. Компиляторы. Принципы, технологии, инструменты. М., *Addison-Wesley*, Вильямс, 2001.
4. К.Бек. Экстремальное программирование. *Microsoft Press*, 2001.
5. Дейл Роджерс. Основы *COM*. 2-е издание. *Microsoft Press*, 2001.
6. *OpenGL Programming Guide* . *Addison-Wesley*, 1999.
7. Б.Страуструп. Язык программирования Си++. 3-е изд. М., Бином, 2001.
8. Материалы сайтов *smalltalk.org*, *opengl.org*, *nvidia.com*, *sgi.com*, *gametutorials.com*, *gnu.org*.
9. *MSDN Programming Reference*, *January* 2002. *Microsoft*, 2002.

Приложение П1.
Исходные тексты программ

Файл 3dConsoleExtensions\Commands.h

```
// Commands.h - header
// created 08.11.2002 by header creator
#pragma once
#ifndef __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__

void AddConsoleCommands (IConsole*);

#endif //__COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл 3dConsoleExtensions\Graphics.h

```
// Graphics.h - header
// created 08.11.2002 by header creator
#pragma once
#ifndef __GRAPHICS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __GRAPHICS_HEADER_FILE_BY_VASFED_INCLUDED__

#include "System.h"

extern bool Render;

void StartRender   (IConsole*, int, void*);
void StopRender    (IConsole*, int, void*);
void Init ();

#endif //__GRAPHICS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл 3dConsoleExtensions\System.h

```
// System.h - header
// created 08.11.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\StorageMaster\stockint.h"
#include "..\Renderer\rendint.h"
#include "..\Physics\physint.h"
#include "..\Loader\Loader.h"
#include "..\AgentsServer\AgentsServer.h"
#include "..\Map\Map.h"
#include "..\Console\Console.h"
#include "..\ConsoleExtensions\ConsoleExtensions.h"

extern ILoader*            Loader;
extern IConsole*           Console;

extern IStorageMaster *    Master;
extern IRenderer *         Renderer;
extern IEnvironment *      Environment;
extern IAgentsMap*         Map;
extern IAgentsServer*      Server;
extern IConsoleExtensions* Conex;

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл 3dsloader\3DSLoader.h

```
#include <stdio.h>
#include <stdlib.h>
#include <exeption.h>
#include "..\core3d\core3d.h"
#include "..\StorageMaster\stockint.h"
#include "..\Loader\Loader.h"
#include "..\Memory\memory.h"

#define PRIMARY              0x4D4D
#define OBJECTINFO           0x3D3D
#define VERSION              0x0002
#define EDITKEYFRAME         0xB000
#define MATERIAL            0xAFFF
#define OBJECT              0x4000
#define MATNAME              0xA000
#define MATDIFFUSE           0xA020
#define MATMAP               0xA200
#define MATMAPFILE           0xA300
#define OBJECT_MESH          0x4100
#define OBJECT_VERTICES      0x4110
#define OBJECT_FACES        0x4120
#define OBJECT_MATERIAL     0x4130
#define OBJECT_UV           0x4140

struct SChunk
{
unsigned short int myID;
unsigned int       myLength;
unsigned int       myBytesRead;
};

struct SIndices
{
unsigned short myA, myB, myC, myIsVisible;
};

IStorageMaster * Stm;
ILoader * Loader;

SChunk * CurrentChunk;
SChunk * TempChunk;
FILE *   ImportFile;
char *   SaveFileName, * ImportFileName;

CColor      Color (1.0F, 1.0F, 1.0F, 1.0F);
float       R, G, B, A;
float       X, Y, Z;
int         TexID = 0;
char        MapFile [256];
CMesh       Mesh;
int         MeshNum = 0;
CPoint3D *  Points;
int         PointNum = 0;
CTexCoord * TexCoords;
int         CoordNum = 0;
SIndices *  Faces;
CPolygon *  Polys;
int         FaceNum = 0;

void ReadChunk                (SChunk * chunk);
int  GetString                (char * str);
void Import3DSFile            (char * filename);
void ProcessNextChunk         (SChunk * prevChunk);
void ProcessNextMaterialChunk (SChunk * prevChunk);
void ProcessNextObjectChunk   (SChunk * prevChunk);

void ReadVertices             (SChunk * prevChunk);
void ReadIndices              (SChunk * prevChunk);
void ReadTexCoords            (SChunk * prevChunk);
void ComputeNormals           ();
void SmoothNormals            ();
```

Файл AgentsServer\Agents.h

```
// Agents.h - header
// created 21.10.2002 by header creator

#pragma once
#ifndef __AGENTSSERVERAGENTS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __AGENTSSERVERAGENTS_HEADER_FILE_BY_VASFED_INCLUDED__

#include <objbase.h>
#include "..\AICore\Agents.h"

/*
This part AgentServer is visible to agents
```

```
and is the gate from multiagent system to other system.

This interface can be taken only from IAgentsServer::GetAgentsInterface
*/

// {C71528C6-E54C-4fba-9F59-4238EB7AFE80}
static const GUID IID_IAgentsServerAgents =
{ 0xc71528c6, 0xe54c, 0x4fba, { 0x9f, 0x59, 0x42, 0x38, 0xeb, 0x7a, 0xfe, 0x80 } };

interface IAgentsServerAgents
{
    //Communication between agents:
    virtual IAgent*     GetAgent                (int)                                   PURE;
    virtual int         SendMessage             (AgentMessage)                          PURE;

    //Map:
    virtual inline int GetMapUnitCode           (float x, float y) const                        PURE;
    virtual inline int*GetPMapUnitData          (float x, float y) const                        PURE;
    virtual inline int GetMapUnitData           (float x, float y, int index = 0) const         PURE;
    virtual inline int GetMapUnitDataSize       (float x, float y) const                        PURE;
    virtual inline voidModifyMapUnitData        (float x, float y, int index, int newvalue)     PURE;

    virtual inline int      GetMapSizeX             (void)  const PURE;
    virtual inline int      GetMapSizeY             (void)  const PURE;
    virtual inline float    GetMapSquareSize        (void)  const PURE;
    virtual float           GetLandscapeLevel       (float x, float y, int reserved = 0)   PURE;

    //Physics
    virtual void        SetSpeed                (int agent, float x, float y, float z = 0)PURE;
    virtual void        SetAcceleration         (int agent, float x, float y, float z = 0)PURE;
    virtual void        GetSpeed                (int agent, float*x, float*y, float*z = 0)PURE;
    virtual void        GetAcceleration         (int agent, float*x, float*y, float*z = 0)PURE;
    virtual float       GetFallingAcceleration (void)                                    PURE;
    virtual void        AddToAcceleration       (int agent, float x, float y, float z = 0)PURE;
    virtual void        AddToSpeed              (int agent, float x, float y, float z = 0)  PURE;
    virtual void        SetAgentPosition        (int agent, float x = 10, float y = 10, float z = 10) PURE;

    //User interface
    //(not all implemented yet)
    virtual void        PlaySound               (int agent, int soundcode)                PURE;
    virtual void        PlaySound               (int agent, char* soundname)              PURE;
    virtual void        PlayAnimation           (int agent, int animation, void* params)  PURE;
    virtual void        SayToUser               (int agent, char* str)                    PURE;
    virtual void        SayToUser               (char* format, ...)                       PURE;
};

#endif //__AGENTSSERVERAGENTS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AgentsServer\AgentsServer.h

```
// AgentsServer.h - header
// created 10.09.2002 by header creator
#pragma once
#ifndef __AGENTSSERVER_HEADER_FILE_BY_VASFED_INCLUDED__
#define __AGENTSSERVER_HEADER_FILE_BY_VASFED_INCLUDED__

#include "Objbase.h"
#include "..\AICore\Agents.h"
#include <stdio.h>
#include "Agents.h"
// {E0F58370-14AC-4a17-9B6B-2B620B105F17}
static const GUID IID_IAgentsServer =
{ 0xe0f58370, 0x14ac, 0x4a17, { 0x9b, 0x6b, 0x2b, 0x62, 0xb, 0x10, 0x5f, 0x17 } };

typedef IAgent* (*PAgentCreator)(int);

interface IAgentsServer: public IUnknown
{
    virtual int                 ok                      (void)          PURE;
    virtual void                Terminate               (void)          PURE;

    virtual int                 AddAgent                (IAgent*, float x = 0, float y = 0, float z =
0)PURE;
    virtual int                 RemoveAgent             (int)           PURE;
    virtual IAgent*             GetAgent                (int)           PURE;
    virtual int                 FindAgent               (IAgent*)       PURE;
    virtual void                ReHash                  (void)          PURE;
    virtual int                 GetNumAgents            (void)          PURE;

    virtual int                 RegisterAgentCreator    (PAgentCreator func, char* ainame = "unknown", char*
                                                         programmer = "unknown", int reserved = 0) PURE;
    virtual int                 RegisterAgentCreator    (char* reserved, char* reserved1 = "unknown", char*
                                                         reserved2 = "unknown")                    PURE;
    virtual int                 UnRegisterAgentCreator (int index)     PURE;
    virtual int                 FindAgentCreator        (PAgentCreator)PURE;
    virtual int                 FindAgentCreator        (char* command)PURE;
    virtual int                 FindAgentCreatorByName  (char* ainame) PURE;
    virtual int                 GetMaxCreators          (void)          PURE;
    virtual int                 CreateAgent             (int agentCreator = 0, float x = 0, float y = 0,
                                                         float z = 0, int misc = 0)    PURE;
    virtual void                PerformAgentsStep       (void)          PURE;
```

```
    virtual void               InitializeAgents       (void)         PURE;
    virtual void               SetAgentPosition       (int agent,
                                                       float x = 10, float y = 10, float z = 10) PURE;
    virtual IAgentsServerAgents*GetAgentsInterface    (void)         PURE;

    //Se
    virtual const char*        GetAgentCreatorName    (int ai) PURE;
};

#endif //__AGENTSSERVER_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AgentsServer\AgentsServerCore.h

```
//Agents Server header file

#ifndef __AGENTS_SERVER_CORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __AGENTS_SERVER_CORE_HEADER_FILE_BY_VASFED_INCLUDED__

#ifdef AGENTSSERVER_EXPORTS
#define AS_API __declspec(dllexport)
#else
#define AS_API __declspec(dllimport)
#endif

#include "AgentsServer.h"
#include "..\Map\Map.h"
#include "Agents.h"

struct AgentCreatorRef
{
    int index;
    int type;
    PAgentCreator func;
    char command [128];
    char ainame [128];
    char programmer[128];

    enum Types {Unknown, Function, Interpeted};
};

class CAgentsServer:public IAgentsServer, public IAgentsServerAgents
{
    public:
        CAgentsServer();
        ~CAgentsServer();

        void Terminate();
        int ok();

        int AddAgent    (IAgent*, float, float, float);
        int RemoveAgent (int);

        IAgent* GetAgent   (int);
        int     FindAgent  (IAgent*);
        int     GetNumAgents (void);

        int Grow();
        int GrowFor(int);
        int GrowTo(int);

        IAgentsServerAgents* GetAgentsInterface ();

//Now functions for agents
        float   GetLandscapeLevel  (float x, float y, int horizont = 0);
        int     GetLandscapeType   (float x, float y);
        void    SetSpeed           (int agent, float x, float y, float z = 0);
        void    SetAcceleration    (int agent, float x, float y, float z = 0);
        void    AddToAcceleration      (int agent, float x, float y, float z = 0);
        void    AddToSpeed             (int agent, float x, float y, float z = 0);
        void    GetSpeed               (int agent, float*x, float*y, float*z = 0);
        void    GetAcceleration        (int agent, float*x, float*y, float*z = 0);
        void    SetAgentPosition       (int agent, float x = 10, float y = 10, float z = 10);


        float   GetFallingAcceleration (void);
        int     SendMessage        (AgentMessage);

        void    PlaySound          (int agent, int soundcode);
        void    PlaySound          (int agent, char* soundname);
        void    PlayAnimation      (int agent, int animation, void* params);
        void    SayToUser          (int agent, char* str);
        void    SayToUser          (char* format, ...);
        float   GetMaxR            (void);

    //Map aggregation:
        inline int     GetMapUnitCode     (float x, float y) const;
        inline int*    GetPMapUnitData (float x, float y) const;
        inline int     GetMapUnitData     (float x, float y, int index = 0) const;
        inline int     GetMapUnitDataSize (float x, float y) const;
        inline void    ModifyMapUnitData  (float x, float y, int index, int newvalue);
```

```
        inline int      GetMapSizeX     (void)  const;
        inline int      GetMapSizeY     (void)  const;
        inline float    GetMapSquareSize(void) const;


        void            PerformAgentsStep       (void);
        void            InitializeAgents        (void);
        void            ReHash                  (void);
        //Agent creation

        int             RegisterAgentCreator    (PAgentCreator func,
                                                 char* programmer = "unknown", char* name = "unknown",
                                                 int reserved = 0);

        int             RegisterAgentCreator    (char*, char*, char*);
        int             UnRegisterAgentCreator (int);
        int             FindAgentCreator        (PAgentCreator);
        int             FindAgentCreator        (char* command);
        int             FindAgentCreatorByName (char* ainame);

        int             CreateAgent             (int agentCreator = 0,
                                                 float x = 0, float y = 0, float z = 0, int misc = 0);

        int             GetMaxCreators          (void);

        int     GrowCreators    (void);
        int     GrowCreatorsFor (int);
        int     GrowCreatorsTo  (int);
        int     FindFreeCreatorPlace ();

        //Se
        const char*     GetAgentCreatorName     (int ai);

        ULONG STDMETHODCALLTYPE AddRef ();
        ULONG STDMETHODCALLTYPE Release ();
        HRESULT STDMETHODCALLTYPE QueryInterface (const IID &iid, void ** ptr);

    protected:
        IAgent** m_agents;
        AgentCreatorRef** m_creators;


        volatile long m_refs;
        int m_agentsSize;
        int m_creatorsSize;

        int m_numAgents;

        int m_ef; //error flag, for preventing error dump from being huge and unreadable
};

#endif
```

Файл AgentsServer\System.h

```
// System.h - header
// created 11.09.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Loader\Loader.h"
#include "..\AICore\Agents.h"
#include "..\Map\Map.h"
#include "..\StorageMaster\stockint.h"
#include "..\Physics\Physint.h"
#include "..\Console\Console.h"
#include "AgentsServer.h"
#include "..\Sound\Sound.h"

extern ILoader*         Loader;
extern IAgentsMap*      Map;
extern IStorageMaster*  Master;
extern IEnvironment*    Environment;
extern IAgentsServer*   Server;
extern IConsole*        Console;
extern ISoundEngine*    Sound;

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AICore\AICore.h

```
//AgentsCore.h - Core of MAS, defines some meat on CAgent's skeleton...

#ifndef __AGENTS_CORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __AGENTS_CORE_HEADER_FILE_BY_VASFED_INCLUDED__
```

```
#ifdef AICORE_EXPORTS
#define AIC_API __declspec(dllexport)
#else
#define AIC_API __declspec(dllimport)
#endif

#include "..\Utilities\Utilities.h"
#include "..\AICore\Agents.h"
#include "..\Map\Map.h"
#include "..\AgentsServer\AgentsServer.h"

class AIC_API CMovingAgent: public IAgent
{
    public:
        CMovingAgent(int x = 10, int y = 10, bool mustRegister = true);
        ~CMovingAgent();

        int     Go ();
        void    Init();
        int     MessageProc(AgentMessage* reserved){throw 2; return false;};
        int     Ok(){return true;};
        void    Ping    (int agent);

        void    PerformMovement ();
        void    CheckHealth     () throw (agentPassTurn);
        void    CheckPosition   () throw (agentPassTurn);

        //------
        //Movement functions:
        void MovementStop       (void);
        void MovementRandom1    (void);
        void MovementRandom2    (void);
        void MovementGoto       (int x, int y);
        void MovementPatrol     (int x1, int y1, int x2, int y2);
        //-------------------------------------------------

        //Variables

        int m_action;
        int m_globalState;
        int m_subState;

        int m_movementState;
        bool m_isWaiting;

        int m_oldX, m_oldY;
        int m_vectorCode, m_searchDirection;
        int m_destinationX, m_destinationY;

        //
        float   m_health;
        float   m_strength;
        float   m_speed;
        int     m_alignment;

        //modifiers:
        float   m_ill;
        unsigned char*  m_mapMask;
        int     m_mapMaskCx;
        int     m_mapMaskCy;

        enum ActionCodes    {AC_NONE,   AC_WALK, AC_FIGHT, AC_OTHER};
        enum MovementCodes  {MSC_NONE,  MSC_RANDOM1, MSC_RANDOM2, MSC_GOTO, MSC_PATROL};
        enum GlobalStates   {GS_NONE,   GS_ATTACK, GS_DEFEND, GS_RUNAWAY, GS_EXPLORE, GS_RANDOM};
        enum GlobalSubstates{GSS_NONE, GSS_CANCHANGE, GSS_TILLDEATH, GSS_RANDOM};
};

#endif
```

Файл AICore\ASystem.h

```
// ASystem.h - header
// created 21.10.2002 by header creator
#pragma once
#ifndef __ASYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __ASYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\AgentsServer\Agents.h"
#include "..\Map\Map.h"
#include "..\Utilities\Utilities.h"

extern IAgentsServerAgents*       Server;

void    AgentsInit  ();

#endif //__ASYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AICore\Agents.h

```
// Agents.h - base interfaces for AI
// created 10.09.2002 by header creator
#pragma once
#ifndef __AGENTS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __AGENTS_HEADER_FILE_BY_VASFED_INCLUDED__

#include <objbase.h>
#include "MessageLanguages.h"

// {935FB5CA-D30C-4b5f-8236-7A0637D5173E}
static const GUID IID_IAgentsFactory =
{ 0x935fb5ca, 0xd30c, 0x4b5f, { 0x82, 0x36, 0x7a, 0x6, 0x37, 0xd5, 0x17, 0x3e } };

class agentPassTurn {}; //- agent exception

//----------------------------
//additional structures and classes:

struct AgentMessage
{
    int     sender;     // Agent who send this message (handle in Server)
    int     receiver;   // 0-> depends on type: broadcast, or local (to nearest agent...)
    int     type;       // Serving type = Broadcast, Private etc. see MessageCodes
    int     ainame;     // - who can understand it? (target AI name)
    int     code;       // user-defined message type
    float   x;
    float   y;
    float   z;
    void*   data;       // anything, deleted, if freedata is 1
    int     data1;      // anything
    int     datasize;   // size of data pointed by data
    bool    freedata;   // indictes, if *data should be freed by deleting it
    enum Types {zMT_NONE =0, zMT_BROADCAST =1, zMT_SOUND =2, zMT_TARGETED =3, zMT_TOUSER =10};
};

interface IAgent
{
    virtual int     Go() PURE;
    virtual int     Ok() PURE;
    virtual void    Init    () PURE;
    virtual int     MessageProc(AgentMessage* msg) PURE;
    virtual void    CollisionDetected (void* reserved){};

    //Vector    m_vector;
    float   m_x, m_y, m_z;

    bool    m_shouldBeProcessed;
    bool    m_ShouldExit;

    //Handles (often all the hashes are similar)
    int     m_index;    //in AgentsServer
    int     m_mesh;     //in Renderer
    int     m_physHash; //in Physics module

    int     m_ainame;   // to make different AI recognizeable
};

#endif //__AGENTS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AICore\Console.h

```
// Console.h - header
// created 11.10.2002 by header creator
#pragma once
#ifndef __CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "system.h"

void AddAgentCommands (IConsole* console, bool replace = true);

#endif //__CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AICore\Creators.h

```
// Creators.h - header
// created 31.10.2002 by header creator
#pragma once
#ifndef __CREATORS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CREATORS_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\AgentsServer\AgentsServer.h"
```

```
IAgent* CreateNewShip  (int);
IAgent* CreateNewUfo   (int);

void RegisterCreators (IAgentsServer* server);

#endif //__CREATORS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AICore\MessageLanguages.h

```
// MessageLanguages.h - header
// created 23.10.2002 by header creator
#pragma once
#ifndef __MESSAGELANGUAGES_HEADER_FILE_BY_VASFED_INCLUDED__
#define __MESSAGELANGUAGES_HEADER_FILE_BY_VASFED_INCLUDED__

enum AgentMessageLanguages
    {
    AML_SIMPLE,
    AML_UFO_RADIO_TALKING,
    AML_SHIP_TALKING,
    AML_MESSAGE_FROM_SYSTEM,
    AML_SOUND,
    AML_ATTACK
    };

#endif //__MESSAGELANGUAGES_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AICore\Ship.h

```
// Ship.h - header
// created 21.10.2002 by header creator
#pragma once
#ifndef __SHIP_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SHIP_HEADER_FILE_BY_VASFED_INCLUDED__

#include "Agents.h"

class CShip: public IAgent
{
    public:
        CShip ();
        ~CShip ();

        int     Go ();
        void    Init();
        int     MessageProc(AgentMessage*){return false;};
        int     Ok(){return true;};
        void    Ping    (int agent);

        void    PerformMovement ();
        void    CheckHealth     () throw (agentPassTurn);
        void    CheckPosition   () throw (agentPassTurn);

        //------
        //Movement functions:
        void MovementStop       (void);
        void MovementRandom1    (void);
        void MovementRandom2    (void);
        void MovementGoto       (int x, int y);
        void MovementPatrol     (int x1, int y1, int x2, int y2);
        //-------------------------------------------------

        //Variables

        int m_action;
        int m_globalState;
        int m_subState;

        int m_movementState;
        bool m_isWaiting;

        int m_oldX, m_oldY;
        int m_vectorCode, m_searchDirection;
        int m_destinationX, m_destinationY;

        //
        float   m_health;
        float   m_strength;
        float   m_speed;
        int     m_alignment;

        //modifiers:
        float   m_ill;
        unsigned char*  m_mapMask;
        int     m_mapMaskCx;
        int     m_mapMaskCy;
        int m_vectorX, m_vectorY;
```

```
        enum ActionCodes    {AC_NONE,   AC_WALK, AC_FIGHT, AC_OTHER};
        enum MovementCodes  {MSC_NONE,  MSC_RANDOM1, MSC_RANDOM2, MSC_GOTO, MSC_PATROL};
        enum GlobalStates   {GS_NONE,   GS_ATTACK, GS_DEFEND, GS_RUNAWAY, GS_EXPLORE, GS_RANDOM};
        enum GlobalSubstates{GSS_NONE, GSS_CANCHANGE, GSS_TILLDEATH, GSS_RANDOM};
};

#endif //__SHIP_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AICore\System.h

```
// System.h - header
// created 11.09.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Loader\Loader.h"
#include "..\AgentsServer\AgentsServer.h"
#include "..\Console\Console.h"
#include "..\Utilities\Utilities.h"

extern ILoader*        Loader;
extern IAgentsServer*  AServer;
extern IConsole*       Console;

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AICore\Ufo.h

```
// Ufo.h - header
// created 21.10.2002 by header creator
#pragma once
#ifndef __UFO_HEADER_FILE_BY_VASFED_INCLUDED__
#define __UFO_HEADER_FILE_BY_VASFED_INCLUDED__

#include "Agents.h"
#include "ASystem.h"

enum UfoMessageTypes
    {
    UFOMT_NONE,
    UFOMT_DIED,
    UFOMT_FOUNDOBJECT,
    UFOMT_FLYHERE,
    UFOMT_MISC
    };

//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------

class CUfo: public IAgent
{
    public:
        CUfo ();
        ~CUfo ();

        int     Go              ();
        int     MessageProc (AgentMessage* msg);
        void    Init            ();
        int     Ok              ();
        void    Ping            (int);


        void PerformMovement   ();
        void AnalyzeLandscape  ();
        void CheckDatabase     ();

        void CallOthers        ();

        void FlyTo             (float x, float y, float z);
        void FlyAround         (float x, float y, float z);
        void CheckHeight       (bool);

        void Die               ();


    protected:

        int m_interestMin;
        int m_interestTimeMax;

        int m_currentInterest;
        int m_currentInterestTimeStart;
        int m_currentInterestTime;

        float m_maxAcceleration; //per direction, Ex. Vector (Acc, Acc, Acc) -> sqrt3*acc = module
```

```
        int m_state, m_oldstate;
            enum MyStates {State_None, State_Flyto, State_Browsing, State_Exploring, State_Exploiting,
State_Exploding, State_Died};

        int* m_database;
        int m_databaseSize;

        float m_health;

        float m_dx, m_dy, m_dz;

        float m_currentAngle;
        int m_direction;

        float m_preferedFlyingHeight;
        float m_preferedExploringHeight;

        float m_experience;
        float m_strenght;
        float m_dexterity;
        float m_wisdom;
        float m_charisma;
        float m_skill;

        float m_heightSensibility;
        int m_heightFlag;
        int m_speedFlag;

        int m_time;

        float m_ox, m_oy, m_oz;
        int m_oldX, m_oldY;
        float m_oldVx, m_oldVy, m_oldVz;
};

#endif //__UFO_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Common\Utilities\1Module.h

```
// Module.h - header
// created 11.09.2002 by header creator
#pragma once
#ifndef __MODULE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __MODULE_HEADER_FILE_BY_VASFED_INCLUDED__

#define DEFINE_IUNKNOWN() \
    public:\
    ULONG STDMETHODCALLTYPE AddRef ();\
    ULONG STDMETHODCALLTYPE Release ();\
    HRESULT STDMETHODCALLTYPE QueryInterface (const IID &iid, void ** ptr);\
    volatile long m_refs;

#define IMPLEMENT_IUNKNOWN(classname,intname,iid) \
    ULONG STDMETHODCALLTYPE classname::AddRef ()\
        {return InterlockedIncrement (&m_refs);}\
    ULONG STDMETHODCALLTYPE classname::Release ()\
        {if (InterlockedDecrement (&m_refs) == 0){delete this;return 0;}return m_refs;}\
    HRESULT STDMETHODCALLTYPE classname::QueryInterface (const IID &Iid, void ** ptr)\
        {     if (Iid == IID_IUnknown) {*ptr = static_cast <IUnknown*> (this);}\
        else {if (Iid == iid)          {*ptr = static_cast <intname*> (this);}\
        else {*ptr = NULL; return E_NOINTERFACE;}}\
        reinterpret_cast <IUnknown *> (*ptr) -> AddRef ();\
        return S_OK;}

#endif //__MODULE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Common\Utilities\Module.h

```
// Module.h - header
// created 11.09.2002 by header creator
#pragma once
#ifndef __MODULE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __MODULE_HEADER_FILE_BY_VASFED_INCLUDED__

#define DEFINE_IUNKNOWN() \
    public:\
    ULONG STDMETHODCALLTYPE AddRef ();\
    ULONG STDMETHODCALLTYPE Release ();\
    HRESULT STDMETHODCALLTYPE QueryInterface (const IID &iid, void ** ptr);\
    volatile long m_refs;

#define IMPLEMENT_IUNKNOWN(classname,intname,iid) \
    ULONG STDMETHODCALLTYPE classname::AddRef ()\
        {return InterlockedIncrement (&m_refs);}\
```

```
    ULONG STDMETHODCALLTYPE classname::Release ()\
        {if (InterlockedDecrement (&m_refs) == 0){delete this;return 0;}return m_refs;}\
    HRESULT STDMETHODCALLTYPE classname::QueryInterface (const IID &Iid, void ** ptr)\
        {       if (Iid == IID_IUnknown)   {*ptr = static_cast <IUnknown*> (this);}\
                else if (Iid == iid)          {*ptr = static_cast <intname*> (this);}\
                else {*ptr = NULL; return E_NOINTERFACE;}\
        reinterpret_cast <IUnknown *> (*ptr) -> AddRef ();\
        return S_OK;}

#define BEGIN_CLASS_INTERFACE_MAP(classname) \
    ULONG STDMETHODCALLTYPE classname::AddRef ()\
        {return InterlockedIncrement (&m_refs);}\
    ULONG STDMETHODCALLTYPE classname::Release ()\
        {if (InterlockedDecrement (&m_refs) == 0){delete this;return 0;}return m_refs;}\
    HRESULT STDMETHODCALLTYPE classname::QueryInterface (const IID &Iid, void ** ptr)\
        {     if (Iid == IID_IUnknown) {*ptr = static_cast <IUnknown*> (this);}

#define DECLARE_CLASS_INTERFACE(interfacename,iid) else if (Iid == iid) {*ptr = static_cast
<interfacename*> (this);}
#define END_CLASS_INTERFACE_MAP() else {*ptr = NULL; return E_NOINTERFACE;} reinterpret_cast <IUnknown *>
(*ptr) -> AddRef (); return S_OK;}

#endif //__MODULE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Common\Utilities\ObjCont.h

```
// ObjectsContainer.h - header
// created 27.09.2002 by header creator
#ifndef __OBJECTSCONTAINER_HEADER_FILE_BY_VASFED_INCLUDED__
#define __OBJECTSCONTAINER_HEADER_FILE_BY_VASFED_INCLUDED__

#define CCont CObjectsContainer

#include <stdio.h>
#include <exeption.h>

template <typename T>
class  CObjectsContainer
    {
    protected:
    T* myObjects;
    int myNumber;
    int myQuant;

    public:
    CObjectsContainer();
    CObjectsContainer (int quant);
    CObjectsContainer (const CObjectsContainer& cont);
    CObjectsContainer<T>& operator= (const CObjectsContainer& cont);
    virtual ~CObjectsContainer();

    /*class checkedPointer //definition of this class is here because this ******* compiler don't want to
eat it anywhere else :(
        {
        protected:
        const CObjectsContainer<T>* myContainer;
        int myPos;

        public:
        //typedef CObjectsContainer<T>::rangeError rangeError;

        checkedPointer ( const CObjectsContainer<T>* container, int pos = 0) throw ( BadParam) :
            myContainer ( container),
            myPos ( pos)
            { ASSERTPARAM ( container); }

        checkedPointer ( const checkedPointer& p) throw ( BadParam) :
            myContainer ( p.rCont()),
            myPos ( p.rIndex())
            { ASSERTPARAM ( myContainer); }

        bool ok() const throw()
            {
            return myContainer? 1 : 0;
            }

        const CObjectsContainer<T>* rCont() const { return myContainer; }
        int rIndex() const { return myPos; }

        const T* rpObject() const
            {
            return &myContainer->at ( myPos);
            }
        const T* operator-> () const  { return rpObject(); }
        operator const T*() const  { return rpObject(); }
        const T& rObject() const  { return *rpObject(); }
        const T& operator* () const  { return rObject(); }

        checkedPointer& operator++ () throw()
            {
            myPos++;
            return *this;
```

```
            }

        checkedPointer operator++ (int) throw()
            {
            checkedPointer temp ( *this);
            myPos++;
            return temp;
            }

        checkedPointer& operator-- () throw()
            {
            myPos--;
            return *this;
            }

        checkedPointer operator-- (int) throw()
            {
            checkedPointer temp ( *this);
            myPos--;
            return temp;
            }

        checkedPointer& operator+= ( int pos) throw()
            {
            myPos += pos;
            return *this;
            }

        checkedPointer& operator-= ( int pos) throw()
            {
            myPos -= pos;
            return *this;
            }

        operator bool() throw() { return myContainer->validIndex ( myPos); }
        bool operator! () const throw() { return !myContainer->validIndex ( myPos); }

        const T& at ( int pos) const  //pos may be negative
            {
            return *checkedPointer ( myContainer, myPos + pos);
            }
        const T& operator[] ( int pos) const  { return at ( pos); } //pos may be negative
        };
*/
    int rNumber() const;
    int rQuant() const;
    const T* rObjects() const;
    const T& front() const; //throws if myNumber == 0
    const T& back() const; //throws if myNumber == 0
    T& front(); //throws if myNumber == 0
    T& back(); //throws if myNumber == 0

    virtual bool ok() const;

    virtual bool operator== (const CObjectsContainer& cont) const;
    virtual int search (const T& obj);
    bool contains (const T& obj);

    virtual bool validIndex ( int i) const;
    virtual T& at ( int pos);
    virtual const T& at ( int pos) const;
    virtual T& operator[] (int n);
    virtual const T& operator[] (int n) const;

    virtual int add (const T& obj);
    virtual CObjectsContainer& del (int pos);

    int increase();
    virtual int incTo (int size);
    int incFor (int delta);
    };

//---------------------------------------------------------------------------------------
#define CCont CObjectsContainer
#define CStore CObjectsStore

//---------------------------------------------------------------------------------------

template <class T>
class  CObjectsStore : public CObjectsContainer<T> //the only difference is increasing of this container
by one objects
    {
    public:
    CObjectsStore();
    CObjectsStore (int quant);
    CObjectsStore (const CObjectsStore& cont);
    CObjectsStore<T>& operator= (const CObjectsStore& cont);
    virtual ~CObjectsStore();

    int increase();
    };

//---------------------------------------------------------------------------------------

template <class T>
CObjectsContainer<T>::CObjectsContainer (const CObjectsContainer& cont) :
    myObjects (NULL),
```

```
    myNumber (cont.rNumber()),
    myQuant (cont.rQuant())
    { if (!cont.rObjects()) return;
      myObjects = new T[cont.rQuant()];
      for (int i = 0; i < myNumber; i++)
        myObjects[i] = cont[i]; }

template <class T>
CObjectsContainer<T>& CObjectsContainer<T>::operator= (const CObjectsContainer& cont)
    { myNumber = cont.rNumber();
      myQuant = cont.rQuant();
      if (!cont.rObjects()) { myObjects = NULL; return *this; }
      myObjects = new T[myQuant];
      for (int i = 0; i < myNumber; i++)
        myObjects[i] = cont[i];
      return *this; }

template <class T>
CObjectsContainer<T>::CObjectsContainer() :
    myObjects (NULL),
    myNumber (0),
    myQuant (0)
    { }

template <class T>
CObjectsContainer<T>::CObjectsContainer (int quant) :
    myObjects (quant? new T [quant] : NULL),
    myNumber (0),
    myQuant (quant)
    { }

template <class T>
CObjectsContainer<T>::~CObjectsContainer()
    { if (myObjects) delete[] myObjects; myObjects = NULL; }

template <class T>
int CObjectsContainer<T>::rNumber() const { return myNumber; }
template <class T>
int CObjectsContainer<T>::rQuant() const { return myQuant; }
template <class T>
const T* CObjectsContainer<T>::rObjects() const { return myObjects; }

template < class T>
const T& CObjectsContainer<T>::front() const
    { Assert ( ok());
    if ( !myNumber)
        Throw (badParams));
    return myObjects[0];
    }

template < class T>
const T& CObjectsContainer<T>::back() const
    { Assert ( ok());
    if ( !myNumber)
        Throw (badParams));
    return myObjects[myNumber-1];
    }

template < class T>
T& CObjectsContainer<T>::front()
    { Assert ( ok());
    if ( !myNumber)
        Throw (badParams));
    return myObjects[0];
    }

template < class T>
T& CObjectsContainer<T>::back()
    { Assert ( ok());
    if ( !myNumber)
        Throw (badParams));
    return myObjects[myNumber-1];
    }

template <class T>
bool CObjectsContainer<T>::ok() const
    { return 0 <= myNumber && myNumber <= myQuant && (myQuant? !!myObjects : 1); }

template <class T>
bool CObjectsContainer<T>::operator== (const CObjectsContainer& cont) const
    { Assert (ok());
    if (myNumber != cont.rNumber()) return 0;
    for (int i = 0; i < myNumber; i++)
        if ( !(myObjects[i] == cont[i])) return 0;
    return 1;
    }

template <class T>
int CObjectsContainer<T>::search (const T& obj)
    { Assert (ok());
    for (int i = 0; i < myNumber; i++)
        if (myObjects[i] == obj) return i;
    return -1;
    }
template <class T>
bool CObjectsContainer<T>::contains (const T& obj)
    { return search (obj) >= 0? 1 : 0; }
```

```
template < class T>
bool CObjectsContainer<T>::validIndex ( int i) const throw()
    { return 0 <= i && i < myNumber; }

template < class T>
T& CObjectsContainer<T>::at ( int pos)
    { Assert ( ok());
    if ( !validIndex ( pos))
        throw;// ( rangeError());
    return myObjects[ pos];
    }
template < class T>
const T& CObjectsContainer<T>::at ( int pos) const
    { Assert ( ok());
    if ( !validIndex ( pos))
        throw;// ( rangeError());
    return myObjects[ pos];
    }

template <class T>
T& CObjectsContainer<T>::operator[] ( int n)
    { Assert ( ok());
      if ( n < 0 ||  n >= myNumber) throw;// rangeError ( "Container [n]", POSITION);
      return myObjects[n];  }
template <class T>
const T& CObjectsContainer<T>::operator[] (int n) const
    { Assert ( ok());
      if ( n < 0 ||  n >= myNumber) throw;//"Container [n]", POSITION);
      return myObjects[n];  }

template <class T>
int CObjectsContainer<T>::add (const T& obj)
    { Assert (ok());
      if (myNumber == myQuant) increase();
      myObjects[myNumber++] = obj;
      return myNumber - 1; }

template <class T>
CObjectsContainer<T>& CObjectsContainer<T>::del (int pos)
    { Assert (ok());
      Assert (0 <= pos && pos < myNumber);
      for (int i = pos; i < myNumber - 1; i++)
        myObjects[i] = myObjects[i+1];
      myNumber--;
      return *this; }

template <class T>
int CObjectsContainer<T>::increase()
    {
    Assert (ok());
    incFor (myQuant/2? myQuant/2 : 1);
    return myQuant;
    }

template <class T>
int CObjectsContainer<T>::incTo (int size)
    {
    Assert (ok());
    AssertC (size >= myNumber, badParams);
    T* t = new T [size];
    for (int i = 0; i < myNumber; i++)
        t[i] = myObjects[i];
    if (myObjects) delete[] myObjects;
    myObjects = t; //!!! better bit copy
    return myQuant = size;
    }
template <class T>
int CObjectsContainer<T>::incFor (int delta) { return incTo (myQuant + delta ); }

//-------------------------------------------------------------------------------------

template <class T>
CObjectsStore<T>::CObjectsStore()
    { }

template <class T>
CObjectsStore<T>::CObjectsStore (int quant) :
    CObjectsContainer<T> (quant)
    { }

template <class T>
CObjectsStore<T>::CObjectsStore (const CObjectsStore& store) :
    CObjectsContainer<T> (store)
    { }

template <class T>
CObjectsStore<T>& CObjectsStore<T>::operator= (const CObjectsStore<T>& store)
    { * ( (CObjectsContainer<T>*) this) = (store); return *this; } //not tested, can't set a breakpoint

template <class T>
CObjectsStore<T>::~CObjectsStore()
    { }

template <class T>
int CObjectsStore<T>::increase()
    {
```

```
    Assert (ok());
    incFor (1);
    return myQuant;
    }

#endif //__OBJECTSCONTAINER_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Console\Commands.h

```
// Commands.h - header
// created 12.09.2002 by header creator
#pragma once
#ifndef __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__

#include "ConsoleCore.h"
#include "System.h"

void InitCommands (CConsole*);

#endif //__COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Console\Console.h

```
// ConsoleUI.h - header file
// created 09.09.2002 by header creator
#pragma once
#ifndef __CONSOLEUI_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CONSOLEUI_HEADER_FILE_BY_VASFED_INCLUDED__

#include <objbase.h>
#include <stdarg.h>

struct Lexem1
{
    int type;
    int datalen;
    int line;

    char data[128];
    int  idata; //for numbers
    double ddata;

    enum LexemTypes
        {
        LT_Unknown = 0,
        LT_Number, //int
        LT_Float,
        LT_String,
        LT_ConstString,
        LT_ConstChar,
        LT_Operator, //or delimitter...
        };
};

// {11E1FE8E-CC2D-4567-AF6C-462CA72D9580}
static const GUID IID_IConsole =
    { 0x11e1fe8e, 0xcc2d, 0x4567, { 0xaf, 0x6c, 0x46, 0x2c, 0xa7, 0x2d, 0x95, 0x80 } };

interface IVariableEngine;
interface IConsole;

typedef int (__cdecl *VprintfP)(const char*, va_list);
typedef int (__cdecl *CommandProcP) (IConsole*, int, void*);
typedef int (__cdecl *UnhandledCommandProc)(IConsole*,int,void*);
//-----------------------------------------------------------------------------
interface IConsole: public IUnknown
{
    //Text output:
    virtual void       SetOutput (VprintfP vprintfP = NULL /*NULL = default*/) PURE;
    virtual VprintfP   GetOutput (void) PURE;

    virtual int    SetDisplayLevel (int level) PURE;
    virtual int    GetDisplayLevel (void) PURE;

    virtual int    printf  (const char*, ...)     PURE;
    virtual int    vprintf (const char*, va_list) PURE;
    virtual void   outline (const char*)          PURE;

    virtual int    printf  (int level, const char*, ...)      PURE;
    virtual int    vprintf (int level, const char*, va_list)  PURE;

    //Commands functuons
    virtual void       AddCommand     (const char* keyword,
                                       CommandProcP proc, bool lexems, char* moduleName = NULL) PURE;
    virtual void       DeleteCommand  (const char* keyword)  PURE;

    virtual void       RaiseCommand   (const char* cmd, bool u = false)      PURE;
```

```
        //Called when user typed a command (with arguments)

    //General
    virtual void               Reset          (void) PURE;
    virtual void               GetVersion     (int*,int*) PURE;

    virtual void               SetUnknownCommandHandler (UnhandledCommandProc, bool lexems = true) PURE;
    virtual void               SetConsoleMode  (char* firstCommand = NULL /*no prefix*/) PURE;

    //Obsolete
    virtual int GetDisplay (void) PURE;
    virtual void SetDisplay (int) PURE;
    //new
    virtual void SetPrompt (char*) PURE;
    virtual void DoPrompt  (void)  PURE;
    virtual bool GetPr () PURE;
};

#endif //__CONSOLEUI_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Console\ConsoleCore.h

```
// ConsoleCore.h - header
// created 12.09.2002 by header creator
#pragma once
#ifndef __CONSOLECORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CONSOLECORE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Common\Utilities\Module.h"
#include "Console.h"
#include "lexem.h"

static const int VERSION1 = 1;
static const int VERSION2 = 4;

struct Command
{
    char*           keyword;
    char*           module;
    CommandProcP    proc;
    bool            wantLexems;
};

#define CHAR_BUFFER    1024
#define LEXEM_BUFFER   128
#define DEFAULT_OUTPUT StdOut
#define DEFAULT_INCREMENT 10
#define LEXEM_WARNINGSASERRORS false

class CConsole: public IConsole
{
    public:
        DEFINE_IUNKNOWN ();

        void        SetOutput       (VprintfP vprintfP = NULL /*NULL = default*/);
        VprintfP    GetOutput       (void);

        int         printf          (const char*, ...);
        int         vprintf         (const char*, va_list);
        int         printf          (int, const char*, ...);
        int         vprintf         (int, const char*, va_list);

        int         SetDisplayLevel (int level);
        int         GetDisplayLevel (void);

        void        outline         (const char*);
        void        AddCommand      (const char* keyword, CommandProcP proc, bool lexems = true, char*
module = NULL);
        void        DeleteCommand   (const char* keyword);
        void        RaiseCommand    (const char* cmd, bool);
        void        Reset           (void);
        void        GetVersion      (int*v1, int*v2);
        int         FindKeyword     (const char*);
        int         FindFreeSpace   (void);

        void        Grow            (void);
        void        GrowFor         (int);
        void        GrowTo          (int);

        void SetPrompt  (char*);
        void DoPrompt   (void);
        bool GetPr ();

        CConsole ();
        ~CConsole ();

        void                SetDisplay (int);
        int                 GetDisplay(void);

        void                SetUnknownCommandHandler (UnhandledCommandProc, bool lexems = true);
        void                SetConsoleMode  (char* firstCommand = NULL /*no prefix*/);
```

```
//  protected:

        Command**   m_commands;
        int         m_sizeCommmands;
        int m_display;


        VprintfP    m_output;

        HWND        m_window;

        Lexem       m_lexemBuffer   [LEXEM_BUFFER];
        char        m_charBuffer    [CHAR_BUFFER]; //ASCIIZ
        int         m_numLexems;
        int         m_curChar;
        bool        m_lexemErrors;
        bool        m_lexemWarnings;

        volatile bool   m_Prompt;
        char    m_PromptStr[128];


        int     m_displayLevel;

        UnhandledCommandProc m_unhandledCommandHandler;
        bool m_unhandledCommandHandlerWantLexems;
        char*       m_commandPrefix;


        void StoreLexem         (Lexem lexem);
        void LexemError         (int lineno, char* text);
        void LexemWarning       (int lineno, char* text);
        char GetNextChar        (void);
        int  LexemAnalyzer      (void);
};

#endif //__CONSOLECORE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Console\System.h

```
// System.h - header
// created 13.09.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Loader\Loader.h"
#include "Console.h"
//#include "..\Linker\Linker" or Vm... - Tankisto's

extern ILoader*    Loader;
extern IConsole*   Console;
//extern ILinker*  Linker;

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Console\lexem.h

```
#ifndef __LEXEM_ANALIZATOR_HEADER_FILE__BY__VASFED_INCLUDED__
#define __LEXEM_ANALIZATOR_HEADER_FILE__BY__VASFED_INCLUDED__

struct Lexem
{
    int type;
    int datalen;
    int line;

    char data[128];
    int  idata; //for numbers
    double ddata;

    enum LexemTypes
        {
        LT_Unknown = 0,
        LT_Number, //int
        LT_Float,
        LT_String,
        LT_ConstString,
        LT_ConstChar,
        LT_Operator, //or delimitter...
        };
    void AddChar(char ch);
    void Reset();
};

#endif
```

Файл Console\resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Console.rc
//
#define IDI_ICON1                       101

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        102
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Файл Console3dCommands\Commands.h

```
// Commands.h - header
// created 03.12.2002 by header creator
#pragma once
#ifndef __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Console\Console.h"

void AddCommands (IConsole* console);

#endif //__COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Console3dCommands\Globals.h

```
// Globals.h - header
// created 25.09.2002 by header creator
#pragma once
#ifndef __GLOBALS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __GLOBALS_HEADER_FILE_BY_VASFED_INCLUDED__

#include "System.h"

extern int      ScrX, ScrY;
extern int      CurrentAgentsCount;
extern int      MaxAgentsCount;
extern int      AgentNum;
extern CPoint3D Target;
extern char     szProgName  [];
extern char     szWndName   [];
extern int      Render;
extern int      RenderLevel;
extern int      PhysicLevel;

extern bool     StartedNormally;
extern int      NoErrors;

namespace StartupOptions
{
    extern int scrX;
    extern int scrY;
    extern bool fullscreen;

    extern int renderLevel;
    extern int physicsLevel;

    extern char batchFile[256];
    extern bool batch;
};

#endif //__GLOBALS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Console3dCommands\Graphics.h

```
// Graphics.h - header
// created 25.09.2002 by header creator
#pragma once
#ifndef __GRAPHICS_HEADER_FILE_BY_VASFED_INCLUDED__
```

```
#define __GRAPHICS_HEADER_FILE_BY_VASFED_INCLUDED__

void InitScene     (char* filename = "meshes.xxx");
void RedrawProc    ();

void InitGraphics  ();

bool ShouldRender   ();
void StartRender   ();
void StopRender    ();

void StartPhysics  ();

void RenderProc (void*);

#endif //__GRAPHICS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Console3dCommands\Keyboard.h

```
// Keyboard.h - header
// created 25.09.2002 by header creator
#pragma once
#ifndef __KEYBOARD_HEADER_FILE_BY_VASFED_INCLUDED__
#define __KEYBOARD_HEADER_FILE_BY_VASFED_INCLUDED__

#include "Globals.h"
void KeyboardProc (HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);

#endif //__KEYBOARD_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Console3dCommands\MainLoop.h

```
// MainLoop.h - header
// created 06.12.2002 by header creator
#pragma once
#ifndef __MAINLOOP_HEADER_FILE_BY_VASFED_INCLUDED__
#define __MAINLOOP_HEADER_FILE_BY_VASFED_INCLUDED__

extern bool MainLoopPause;
extern bool MainLoopShouldStop;

void MainLoop (void*);

#endif //__MAINLOOP_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Console3dCommands\System.h

```
// System.h - header
// created 03.12.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "../Loader/Loader.h"
#include "..\Console\Console.h"

#include "../StorageMaster/stockint.h"
#include "../Renderer/rendint.h"
#include "../Physics/physint.h"
#include "..\AgentsServer\AgentsServer.h"
#include "..\Map\Map.h"

extern ILoader*            Loader;
extern IConsole*           Console;

extern IStorageMaster *    Master;
extern IRenderer *         Renderer;
extern IEnvironment *      Environment;
extern IAgentsMap*         Map;
extern IAgentsServer*      Server;

bool CheckModulesLoaded ();
bool LoadNotLoadedModules ();

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleExtensions\Commands.h

```
// Commands.h - header
```

```
// created 16.10.2002 by header creator
#pragma once
#ifndef __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Console\Console.h"

bool ExtendConsole     (IConsole* console);
bool UnExtendConsole   (IConsole* console);

#endif //__COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleExtensions\ConsoleExtensions.h

```
// ConsoleExtensions.h - header
// created 16.10.2002 by header creator
#pragma once
#ifndef __CONSOLEEXTENSIONS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CONSOLEEXTENSIONS_HEADER_FILE_BY_VASFED_INCLUDED__

#include <objbase.h>
#include "..\Console\Console.h"

// {634770AE-0E97-4709-95F0-D0DB66238428}
static const GUID IID_IConsoleExtensions =
{ 0x634770ae, 0xe97, 0x4709, { 0x95, 0xf0, 0xd0, 0xdb, 0x66, 0x23, 0x84, 0x28 } };

struct CommandDef
{
    char* keyword;
    void* proc;
    bool lexems;
    bool attached;
};

#define DEFINE_COMMAND_LIST(list) extern CommandDef CommandList_##list[];
#define BEGIN_COMMAND_LIST(list) CommandDef CommandList_##list [] = {{"", NULL, false, false}
#define DEFINE_COMMAND(keyword,proc,lexems) ,{keyword, proc, lexems, false}
#define END_COMMAND_LIST() };
#define sizearr(arr) (sizeof (arr)/sizeof (arr[0]))
#define CM(list) (CommandList_##list)
#define CMS(list) (sizearr(CommandList_##list) - 1)

#define COMMAND_PROC_SIMPLE(proc) void proc (IConsole* c, int, void*)
#define COMMAND_PROC_LEXEMS(proc) void proc (IConsole* c, int ln, Lexem1* data)
#define COMMAND_PROC_STRING(proc) void proc (IConsole* c, int, char* str)

interface IConsoleExtensions: public IUnknown
{
    virtual bool __cdecl   AttachCommands  (CommandDef* commands, int numcommands, IConsole* console =
NULL /*default console*/, bool replace = false, char* sourceModuleName = NULL) PURE;
    virtual bool __cdecl   DetachCommands  (CommandDef* commands, int numcommands, IConsole* console =
NULL) PURE;

    virtual bool __cdecl   Extend          (IConsole* console = NULL) PURE;
    virtual bool __cdecl   UnExtend        (IConsole* console = NULL) PURE;
    virtual IConsole* __cdecl GetConsole   (void) PURE;
};

#endif //__CONSOLEEXTENSIONS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleExtensions\ConsoleExtensionsCore.h

```
// ConsoleExtensionsCore.h - header
// created 16.10.2002 by header creator
#pragma once
#ifndef __CONSOLEEXTENSIONSCORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CONSOLEEXTENSIONSCORE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "ConsoleExtensions.h"
#include "..\Common\Utilities\Module.h"

class CConsoleExtensions: public IConsoleExtensions
{
    public:
    DEFINE_IUNKNOWN ();
    CConsoleExtensions ();
    ~CConsoleExtensions();

    bool __cdecl   AttachCommands  (CommandDef* commands, int, IConsole* console = NULL /*default
console*/, bool replace = false, char* module = NULL);
    bool __cdecl   DetachCommands  (CommandDef* commands, int, IConsole* console = NULL);

    bool __cdecl   Extend          (IConsole* console = NULL);
    bool __cdecl   UnExtend        (IConsole* console = NULL);
    IConsole* __cdecl GetConsole       (void);
};
```

```
#endif //__CONSOLEEXTENSIONSCORE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleExtensions\Language.h

```
// Language.h - header
// created 21.10.2002 by header creator
#pragma once
#ifndef __LANGUAGE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __LANGUAGE_HEADER_FILE_BY_VASFED_INCLUDED__

void RegisterCompiler  ();

#endif //__LANGUAGE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleExtensions\System.h

```
// System.h - header
// created 16.10.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Loader\Loader.h"
#include "..\Console\Console.h"
#include "ConsoleExtensions.h"

extern ILoader*            Loader;
extern IConsole*           Console;
extern IConsoleExtensions* Extensions;

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUI\Env.h

```
// Env.h - header
// created 12.09.2002 by header creator
#pragma once
#ifndef __ENV_HEADER_FILE_BY_VASFED_INCLUDED__
#define __ENV_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Loader\Loader.h"

//System environment:

extern ILoader* Loader;

#endif //__ENV_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUI2\Globals.h

```
// Globals.h - header
// created 16.11.2002 by header creator
#pragma once
#ifndef __GLOBALS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __GLOBALS_HEADER_FILE_BY_VASFED_INCLUDED__

extern int LamerMode;

#endif //__GLOBALS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUI2\Old3dUI.h

```
// Old3dUI.h - header
// created 16.11.2002 by header creator
#pragma once
#ifndef __OLD3DUI_HEADER_FILE_BY_VASFED_INCLUDED__
#define __OLD3DUI_HEADER_FILE_BY_VASFED_INCLUDED__

void Init ();

#endif //__OLD3DUI_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUI2\Startup.h

```
// Startup.h - header
// created 16.11.2002 by header creator
#pragma once
#ifndef __STARTUP_HEADER_FILE_BY_VASFED_INCLUDED__
#define __STARTUP_HEADER_FILE_BY_VASFED_INCLUDED__

void ShowSplash ();
void HideSplash ();

bool CreateLoader  ();
bool CreateConsole (bool notifyLoader = true);

bool StartKernel ();


#endif //__STARTUP_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUI2\System.h

```
// System.h - header
// created 05.11.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Loader\LoaderCore.h"
#include "..\Console\Console.h"

extern ILoader*        Loader;
extern IConsole*       Console;

//additional modules, may not be present
#ifndef NO_MODULES
    #include "..\Sound\Sound.h"
    #include "..\Renderer\rendint.h"
    #include "..\Physics\physint.h"
    #include "..\StorageMaster\StockInt.h"
    #include "..\AgentsServer\AgentsServer.h"
    #include "..\Map\Map.h"
    //#include "..\SmallCOmi"

    extern ISoundEngine*   Sound;
    extern IRenderer*      Renderer;
    extern IEnvironment*   Environment;
    extern IStorageMaster* Master;
    extern IAgentsServer*  Server;
    extern IAgentsServerAgents* ServerA;
    extern IAgentsMap*     Map;
    //extern ISmallCompiler*   Compiler;
#endif

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUI2\Termination.h

```
// Termination.h - header
// created 16.11.2002 by header creator
#pragma once
#ifndef __TERMINATION_HEADER_FILE_BY_VASFED_INCLUDED__
#define __TERMINATION_HEADER_FILE_BY_VASFED_INCLUDED__

#endif //__TERMINATION_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUI2\Utilities.h

```
// Utilities.h - header
// created 17.11.2002 by header creator
#pragma once
#ifndef __UTILITIES_HEADER_FILE_BY_VASFED_INCLUDED__
#define __UTILITIES_HEADER_FILE_BY_VASFED_INCLUDED__

int __cdecl printf (int level, unsigned int resourceId, ...);
int __cdecl printf (unsigned int resourceId, ...);

#endif //__UTILITIES_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUI2\Work.h

```
// Work.h - header
// created 16.11.2002 by header creator
#pragma once
#ifndef __WORK_HEADER_FILE_BY_VASFED_INCLUDED__
#define __WORK_HEADER_FILE_BY_VASFED_INCLUDED__

bool StartMainLoop (); //creates process
void PauseMainLoop ();
void StopMainLoop ();

bool GetMainLoopShouldStop ();
bool GetMainLoopPaused ();
bool WasMainLoopStarted ();

#endif //__WORK_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUI2\resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by ConsoleUI2.rc
//
#define IDI_ICON1                       101
#define IDS_LoaderDllName               105
#define IDS_MessageExEn                 106
#define IDS_ConsoleDllName              107
#define IDS_ExitCommand                 108
#define IDS_ProgName                    109
#define IDS_DefaultPrompt               110
#define IDS_Modules                     113

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        106
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1003
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Файл ConsoleUIDll\ConsoleUI.h

```
// ConsoleUI.h - header
// created 25.09.2002 by header creator
#pragma once
#ifndef __CONSOLEDLLUI_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CONSOLEDLLUI_HEADER_FILE_BY_VASFED_INCLUDED__

#include "objbase.h"

//REM: it is only window for Console

// {20473D9F-547D-4db3-A768-26F5EBC2E404}
static const GUID IID_IConsoleUI =
{ 0x20473d9f, 0x547d, 0x4db3, { 0xa7, 0x68, 0x26, 0xf5, 0xeb, 0xc2, 0xe4, 0x4 }};

interface IConsoleUI: public IUnknown
{
    virtual void ShowWindow (int cmdshow) PURE;
    virtual void BringToTop (void) PURE;
};

#endif //__CONSOLEDLLUI_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUIDll\ConsoleUICore.h

```
// ConsoleUICore.h - header
// created 25.09.2002 by header creator
#pragma once
#ifndef __CONSOLEUICORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CONSOLEUICORE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "ConsoleWindow.h"
#include "ConsoleUI.h"
```

```
#include "..\Common\Utilities\Module.h"

class CConsoleUI: public IConsoleUI
{
    public:
        CConsoleUI();
        ~CConsoleUI();
        void ShowWindow (int cmdshow);
        void BringToTop ();
        void Init ();

        CConsoleWindow* m_window;
        DEFINE_IUNKNOWN ();
};

#endif //__CONSOLEUICORE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUIDll\ConsoleUIDll.h

```
// ConsoleUIDll.h : main header file for the ConsoleUIDll DLL
//

#pragma once

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"      // main symbols

// CConsoleUIDllApp
// See ConsoleUIDll.cpp for the implementation of this class
//

class CConsoleUIDllApp : public CWinApp
{
public:
    CConsoleUIDllApp();

// Overrides
public:
    virtual BOOL InitInstance();

    DECLARE_MESSAGE_MAP()
};
```

Файл ConsoleUIDll\ConsoleWindow.h

```
#pragma once
#include "afxwin.h"
#include "resource.h"
#include "afxcmn.h"

// CConsoleWindow dialog

class CConsoleWindow : public CDialog
{
    DECLARE_DYNAMIC(CConsoleWindow)

public:
    CConsoleWindow(CWnd* pParent = NULL);   // standard constructor
    virtual ~CConsoleWindow();

// Dialog Data
    enum { IDD = IDD_ConsoleDialog };

protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

    DECLARE_MESSAGE_MAP()
public:
    CListBox m_List;
    CEdit m_Edit;
    CButton m_Button;
    afx_msg void OnBnClickedEnter();
    afx_msg void OnSize(UINT nType, int cx, int cy);
    BOOL OnInitDialog(void);
    afx_msg void OnChar(UINT nChar, UINT nRepCnt, UINT nFlags);
    afx_msg void OnClose();
    afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
    HHOOK m_hook;
};
```

Файл ConsoleUIDll\System.h

```
// System.h - header
// created 25.09.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Loader\Loader.h"
#include "..\Console\Console.h"
#include "ConsoleWindow.h"

extern ILoader*    Loader;
extern IConsole*   Console;

extern CConsoleWindow* MainWindow;

extern HINSTANCE HInstance; //used for loading resources

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл ConsoleUIDll\resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by ConsoleUIDll.rc
//
#define IDD_ConsoleDialog               130
#define IDC_Enter                       1008
#define IDC_LIST1                       1009
#define IDC_EDIT1                       1010
#define IDR_ACCELERATOR1                5000
#define IDC_EDIT2                       5001
#define IDC_RICHEDIT21                  5002

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        5001
#define _APS_NEXT_COMMAND_VALUE         32773
#define _APS_NEXT_CONTROL_VALUE         5003
#define _APS_NEXT_SYMED_VALUE           5000
#endif
#endif
```

Файл Core2D\Core2D.h

```
// Core2D.h : main header file for the Core2D DLL
//

#pragma once

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"      // main symbols

// CCore2DApp
// See Core2D.cpp for the implementation of this class
//

class CCore2DApp : public CWinApp
{
public:
    CCore2DApp();

// Overrides
public:
    virtual BOOL InitInstance();

    DECLARE_MESSAGE_MAP()
};
```

Файл Core2D\Core2DCore.h

```
// Core2DCore.h - header
```

```
// created 19.09.2002 by header creator
#pragma once
#ifndef __CORE2DCORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CORE2DCORE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "Core2Dint.h"
#include "system.h"
#include "..\Common\Utilities\Module.h"

#include "MainWindow.h"

class CCore2D: public ICore2D
{
    public:
        CCore2D();
        ~CCore2D();

        void Init ();

    void    ShowWindow (int cmdshow, int x = 0, int y = 0, int cx = 0, int cy = 0);
    HDC     GetDC       ();
    void    DrawFrame   ();

    DEFINE_IUNKNOWN ();

    protected:
        CMainWindow* m_mw;

};

#endif //__CORE2DCORE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Core2D\Core2Dint.h

```
// CORE_2D_INTERFACES.h - header
// created 19.09.2002 by header creator
#pragma once
#ifndef __CORE_2D_INTERFACES_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CORE_2D_INTERFACES_HEADER_FILE_BY_VASFED_INCLUDED__

#include "objbase.h"

// {295072BB-7874-4ce6-A176-688755DAF941}
static const GUID IID_ICore2D =
{ 0x295072bb, 0x7874, 0x4ce6, { 0xa1, 0x76, 0x68, 0x87, 0x55, 0xda, 0xf9, 0x41 } };

interface ICore2D: public IUnknown
{
    virtual void    ShowWindow  (int cmdshow, int x = 0, int y = 0, int cx = 0, int cy = 0) PURE;
    virtual HDC     GetDC       (void) PURE;
    virtual void    DrawFrame   (void) PURE;
};

#endif //__CORE_2D_INTERFACES_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Core2D\MainWindow.h

```
#pragma once

#include "resource.h"
#include "afxwin.h"
#include "afxcmn.h"

// CMainWindow dialog

class CMainWindow : public CDialog
{
    DECLARE_DYNAMIC(CMainWindow)

public:
    CMainWindow(CWnd* pParent = NULL);   // standard constructor
    virtual ~CMainWindow();

// Dialog Data
    enum { IDD = IDD_MainDialog };

protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

    DECLARE_MESSAGE_MAP()
public:
    afx_msg void OnSize(UINT nType, int cx, int cy);
    afx_msg void OnClose();
    afx_msg void OnBnTurnOnTimer();
    afx_msg void OnBnClickedTurnoff();
    afx_msg void OnDeltaposSpin1(NMHDR *pNMHDR, LRESULT *pResult);
    afx_msg void OnTimer(UINT nIDEvent);
```

```
    BOOL OnInitDialog(void);
    CStatic m_zoomValueBuddy;
    CSpinButtonCtrl m_zoomSpinner;
    int m_zoom;
    void DrawFrame(void);
    afx_msg void OnBnClickedDrawoneframe();
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg void OnPaint();
    afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
    afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg BOOL OnMouseWheel(UINT nFlags, short zDelta, CPoint pt);
    bool m_draw;

    afx_msg void OnMouseMove(UINT nFlags, CPoint point);
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    void ProcessMouse (CPoint point);
    CStatic m_MPDispl;
};
```

Файл Core2D\System.h

```
// System.h - header
// created 19.09.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Loader\Loader.h"
#include "..\AICore\Agents.h"
#include "..\Console\Console.h"
#include "..\Map\Map.h"
#include "..\AgentsServer\AgentsServer.h"
#include "Core2Dint.h"
//#include "..\UserInterface\UI.h"

extern ILoader*         Loader;
extern IAgentsMap*      Map;
extern IAgentsServerAgents* AServer;
extern IAgentsServer*   Server;
extern IConsole*        Console;
extern ICore2D*         Core2D;
//extern IMFCUI*            UI;

extern HINSTANCE        Instance;

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Core2D\console.h

```
// console.h - header
// created 21.09.2002 by header creator
#pragma once
#ifndef __CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "system.h"

void ConfigureConsole ();

#endif //__CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Core2D\resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Core2D.rc
//
#define IDD_MainDialog                  3000
#define IDC_TurnOn                      3000
#define IDC_TurnOff                     3001
#define IDC_SPIN1                       3003
#define IDC_ZoomValueDisplay            3004
#define IDC_DrawOneFrame                3005
#define IDC_MousePosDispl               3006

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        3001
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         3007
#define _APS_NEXT_SYMED_VALUE           3002
```

```
#endif
#endif
```

Файл Core3D\core3d.h

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** CORE3D.H                                                          ***/
/*** ОБЪЯВЛЕНИЕ БАЗОВЫХ ГРАФИЧЕСКИХ КЛАССОВ. ДИНАМИЧЕСКИ КОМПОНУЕМАЯ    ***/
/*** БИБЛИОТЕКА, ИСПОЛЬЗУЕМАЯ ГРАФИЧЕСКИМ И ФИЗИЧЕСКИМ МОДУЛЯМИ        ***/
/*** СОДЕРЖИТ ОПИСАНИЯ КЛАССОВ:                                        ***/
/***      CCOLOR                                                       ***/
/***      CPOINT3D                                                     ***/
/***      CMATRIX4X4                                                   ***/
/***      CLIGHTSOURCE                                                 ***/
/***      CICON                                                        ***/
/***      CPICTURE                                                     ***/
/***      CBILLBOARD                                                   ***/
/***      CCOLORMATRIX                                                 ***/
/***      CTEXCOORD                                                    ***/
/***      CGRAPHMATERIAL                                               ***/
/***      CVERTEX                                                      ***/
/***      CPOLYGON                                                     ***/
/***      CBONE                                                        ***/
/***      CMESH                                                        ***/
/***      CSTANDARDPRIMITIVE                                           ***/
/***                                                                   ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ          ***/
/***                                                                   ***/
/***                                                                   ***/
/*************************************************************************/
/*************************************************************************/

#include <windows.h>
#include <assert.h>
#include <stdio.h>
#include <io.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdarg.h>
#include <time.h>
#include <math.h>
#include <exeption.h>

#ifdef CORE3D_EXPORTS
#define CORE3D_API __declspec(dllexport)
#else
#define CORE3D_API __declspec(dllimport)
#endif

#ifndef _CORE3D_H_
#define _CORE3D_H_

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CCOLOR ////////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

// Цвет состоит из четырех компонент - компоненты красного,
// зеленого, синего и альфа-канала
#define COLOR_FLOATS 4

class CORE3D_API CColor
{
public:
float myRGBA [COLOR_FLOATS];

friend class CRenderer;

/*************************************************************************/

CColor  ();
CColor  (float * rgba);
CColor  (float r, float g, float b, float a);
~CColor ();

void    Init        (float r, float g, float b, float a);
void    Init        (float * rgba);
void    Init        (CColor color);
float & RetFloat    (int n);
void    operator =  (CColor& that);
void    operator =  (float * color);
float & operator () (int n);
float * RetPointer  ();
};

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CPOINT3D //////////////////////////////
```

```
 ////////////////////////////////////////////////////////////////////////
 ////////////////////////////////////////////////////////////////////////

// Положение точки задается тремя координатами - по осям X, Y и Z
#define NUM_COORDS 3

class CORE3D_API CPoint3D
{
private:
float myXYZ [NUM_COORDS];

friend class CRenderer;

/*************************************************************************/

public:
CPoint3D  ();
CPoint3D  (float x, float y, float z);
CPoint3D  (float xyz [3]);
~CPoint3D ();

void     Init        (float x, float y, float z);
void     Init        (float xyz [3]);
void     Init        (CPoint3D &point);
int      ok          (int n);
float &  retfloat    (int n);
float &  operator () (int n);
void     AddToFloat  (int n, float f);
void     AddFloats   (float f1, float f2, float f3);
void     AddToFloats (float f1, float f2, float f3);
void     AddToFloats (CPoint3D &point);
void     operator () (int n, float f);
float *  RetPointer  ();
bool     operator == (CPoint3D &point);
bool     operator != (CPoint3D &point);
void     operator =  (float vert [3]);
void     operator =  (CPoint3D &that);
void     operator += (float vert [3]);
void     operator -= (float vert [3]);
void     operator *= (float vert [3]);
void     operator /= (float vert [3]);
void     operator += (CPoint3D &that);
void     operator -= (CPoint3D &that);
void     operator *= (CPoint3D &that);
void     operator /= (CPoint3D &that);
void     operator += (float n);
void     operator -= (float n);
void     operator *= (float n);
void     operator /= (float n);
CPoint3D operator -  (CPoint3D &that);
CPoint3D operator +  (CPoint3D &that);
CPoint3D operator *  (CPoint3D &that);
CPoint3D operator /  (CPoint3D &that);
CPoint3D operator -  (int n);
CPoint3D operator +  (int n);
CPoint3D operator *  (int n);
CPoint3D operator /  (int n);
CPoint3D operator -  (float n);
CPoint3D operator +  (float n);
CPoint3D operator *  (float n);
CPoint3D operator /  (float n);

// Несколько функций для CPoint3D как для вектора
CPoint3D & CrossProduct (CPoint3D &point);
float      Normalize    ();
float      DotProduct   (CPoint3D &point);
float      RetLength    ();
};

/*************************************************************************/
/*************************************************************************/
/********************* CLASS CMATRIX4X4 **********************************/
/*************************************************************************/
/*************************************************************************/

#define MATRIXSIZEX 4
#define MATRIXSIZEY 4

class CORE3D_API CMatrix4x4
{
private:
float myMatrix [MATRIXSIZEX * MATRIXSIZEY];

/*************************************************************************/

public:
CMatrix4x4  ();
~CMatrix4x4 ();

void       Init             (CMatrix4x4& m);
void       operator =       (CMatrix4x4& m);
float &    RetFloat         (int x, int y);
float &    operator ()      (int x, int y);
float *    RetPointer       ();
void       MakeSimpleMatrix ();
int        IfSimpleMatrix   ();
void       MakeRotXMatrix   (float a);
```

```
void       MakeRotYMatrix   (float a);
void       MakeRotZMatrix   (float a);
float      DotProduct       (float a1, float a2, float a3, float a4, float b1, float b2, float b3, float
b4);
void       DotProduct       (CMatrix4x4 &matrix);
CPoint3D & DotProduct       (CPoint3D &point);
void       Transpone        ();
};

// Я использую три типа источников освещения:
// Либо свет исходит из одной точки одинаково во всех направлениях,
// либо источник света - бесконечно удаленный (задается или положение, или направление)
const LIGHT_OMNI = 1;
// Источник света - "фонарик" (задается и положение, и направление)
const LIGHT_SPOT = 2;

// И в OpenGL, и в Direct3D максимальное количество источников света равно восьми
#define GLMAXLIGHTS 8

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CLIGHTSOURCE ///////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

class CORE3D_API CLightSource
{
public:
CColor   myColor;
CColor   myPosition; // У CColor 4 параметра, а столько нужно
                     //для задания позиции источника освещения в OpenGL (X, Y, Z и W)
CPoint3D myDirection;
int      myLightNum, myLightType;
float    myParam;

/*************************************************************************/

public:
CLightSource  ();
CLightSource  (int lighttype, int lightnum);
~CLightSource ();

void operator =   (CLightSource& that);
void Init         (int lighttype, int lightnum);
int  ok           (int type);
void SetColor     (float r, float g, float b, float a);
void SetColor     (float * rgba);
void SetColor     (CColor &color);
void SetPosition  (float x, float y, float z, float w);
void SetPosition  (float * xyzw);
void SetPosition  (CColor &pos);
void SetDirection (float x, float y, float z);
void SetDirection (float * xyz);
void SetDirection (CPoint3D &dir);
void SetCutoff    (float cutoff);
};

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////////// CLASS CICON //////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

// Иконки нужны для формирования шрифтов и простых одноцветных картинок

class CORE3D_API CIcon
{
protected:
int    mySizeX, mySizeY, myByteSizeX;
char * myBitmap;
int    myID;

/*************************************************************************/

public:
CIcon  ();
CIcon  (int x, int y);
~CIcon ();
void   Init        (int x, int y);
void   SetID       (int n);
int    RetID       ();
int    RetSizeX    ();
int    RetSizeY    ();
int    ok          (int x, int y);
char & RetByte     (int x, int y);
char & operator () (int x, int y);
char * RetPointer  ();
void   operator =  (CIcon &that);
};

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CPICTURE ///////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

class CORE3D_API CPicture : public CIcon
```

```
{
private:
CColor   myColor;
CPoint3D myPosition;

/*************************************************************************/

public:
CPicture  ();
CPicture  (int x, int y, float px, float py, float pz);
~CPicture ();

void       Init     (int x, int y, float px, float py, float pz);
void       SetPos   (float px, float py, float pz);
void       SetColor (float r, float g, float b, float a);
CPoint3D & RetPos   ();
CColor &   RetColor ();
};

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CBILLBOARD ////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

class CORE3D_API CBillBoard
{
private:
CPoint3D myCenter;
CColor   myColor;
float    mySizeX, mySizeY;
int      myTexID;
CPoint3D myFreedomVector;

friend class CRenderer;

/*************************************************************************/

public:
CBillBoard  ();
~CBillBoard ();

void       Init             (float x, float y, float z, float r, float g, float b, float a, float sizex,
float sizey, int id);
void       SetTexID         (int id);
int        RetTexID         ();
void       SetCenter        (float x, float y, float z);
void       SetCenter        (float * xyz);
void       SetCenter        (CPoint3D & point);
CPoint3D & RetCenter        ();
void       SetColor         (float r, float g, float b, float a);
CColor &   RetColor         ();
void       SetFreedomVector (float x, float y, float z);
CPoint3D & RetFreedomVector ();
void       SetSize          (float x, float y);
float      RetSizeX         ();
float      RetSizeY         ();
float      RetMinX          ();
float      RetMaxX          ();
float      RetMinY          ();
float      RetMaxY          ();
};

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CCOLORMATRIX //////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

// Матрица цветов может быть использована как текстура, спрайт или биллборд

class CORE3D_API CColorMatrix
{
public:
float *      myMatrix;
CPoint3D     myPosition;
int          mySizeX, mySizeY;
unsigned int myID;

/*************************************************************************/

public:
CColorMatrix  ();
CColorMatrix  (int x, int y);
CColorMatrix  (CColorMatrix &cm);
~CColorMatrix ();

int        RetX        ();
int        RetY        ();
int        RetID       ();
void       SetPosition (float x, float y, float z);
CPoint3D & RetPosition ();
int        ok          (int x, int y);
int        ok          (int x, int y, int z);
void       Init        (int x, int y);
void       Init        (CColorMatrix &cm);
void       New         ();
```

```
void       Delete      ();
float &    RetElement  (int x, int y, int z);
float &    operator () (int x, int y, int z);
float *    RetPointer  (int n);
float *    operator () (int n);
void       Copy        (float * matrix);
void       AddTexel    (int x, int y, CColor &color);
void       AddTexel    (int x, int y, float r, float g, float b, float a);
void       SetNewID    (int id);
void       operator =  (float * matrix);
void       operator =  (CColorMatrix &that);
};

    ////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CTEXCOORD /////////////////////////
 ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////

// Текстурные координаты в данной вершине

class CORE3D_API CTexCoord
{
public:
float myX, myY;

/*********************************************************************/

public:
CTexCoord  ();
CTexCoord  (float x, float y);
~CTexCoord ();

void  Init (float x, float y);
};

    ////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////
  ////////////////////// CLASS CGRAPHMATERIAL ////////////////////////
 ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////

#define COLOR_PER_VERTEX    1
#define COLOR_FROM_MATERIAL 2

// В этом классе описаны свойства материала, важные для графического модуля

class CORE3D_API CGraphMaterial
{
private:
float myDiffuse, myShininess, myEmission;
float myRGBA [COLOR_FLOATS];
int   myColorSource;

friend class CRenderer;

/*********************************************************************/

public:
CGraphMaterial  ();
CGraphMaterial  (float d, float s, float e);
~CGraphMaterial ();

void    SetColor        (float r, float g, float b, float a);
float * RetColor        ();
void    SetColorSource  (int source);
int     RetColorSource  ();
void    Init            (float d, float s, float e);
void    SetDiffuseCoeff (float d);
void    SetShininess    (float s);
void    SetEmission     (float e);
float   RetDiffuseCoeff ();
float   RetShininess    ();
float   RetEmission     ();
void    operator =      (CGraphMaterial & mat);
};

    ////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CVERTEX ///////////////////////////
 ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////

class CORE3D_API CVertex
{
public:
CPoint3D  myPoint, myNormal;
CColor    myColor;
CTexCoord myCoord;
float     myWeight;

/*********************************************************************/

public:
CVertex  ();
~CVertex ();
```

```
void        Init        (float x, float y, float z, float n1, float n2, float n3,
                         float r, float g, float b, float a, float cx, float cy, float w);
void        SetPoint    (float x, float y, float z);
void        SetNormal   (float x, float y, float z);
void        SetColor    (float r, float g, float b, float a);
void        SetTexCoord (float x, float y);
void        SetWeight   (float w);
CPoint3D &  RetPoint    ();
CPoint3D &  RetNormal   ();
CColor &    RetColor    ();
CTexCoord & RetTexCoord ();
float       RetWeight   ();
void        operator =  (CVertex &that);
};

#define MAXVERTEXES 3

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CPOLYGON //////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

// Число вершин в одном полигоне ограничено тремя, т.к. для рассчета освещения
// по модели Гуро необходимы грани с тремя вершинами, и три вершины всегда
// однозначно задают плоскость.

class CORE3D_API CPolygon
{
public:
CVertex myVertexArray [MAXVERTEXES];
int     myVertexNum, myCurrVert;
int     myTexPointer;

/*************************************************************************/

public:
CPolygon  ();
CPolygon  (CVertex vertarray [3]);
~CPolygon ();

void      Init          ();
void      Init          (CVertex vertarray [3]);
int       ok            (int n);
int       ok            ();
void      operator =    (CPolygon &that);
void      AddArray      (CVertex vertarray [3]);

void      SetVertex     (float x, float y, float z, float n1, float n2, float n3,
                         float r, float g, float b, float a, float cx, float cy, float w);

CVertex & retvert       (int n);
CVertex & operator ()   (int n);
void      SetTexPointer (int n);
int       RetTexPointer ();
};

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CBONE /////////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

// "Кость" необходима для создания иерархических моделей и использования анимации

class CORE3D_API CBone
{
private:
CPoint3D   myCenter;
CMatrix4x4 myTransform;

/*************************************************************************/

public:
CBone  ();
~CBone ();

void         SetCenter          (float x, float y, float z);
void         Move               (float x, float y, float z);
CPoint3D &   RetCenter          ();
CMatrix4x4 & RetTransformMatrix ();
void         operator =         (CBone &that);
};

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CMESH /////////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

class CORE3D_API CMesh
{
private:
CLightSource ** myLights;
CPolygon **     myPolyArray;
int             myMaxPolys, myCurPoly, myMaxLights, myCurLight;
CGraphMaterial  myMaterial;
```

```
CBone           myBone;

CMesh *         myParent;
CMesh **        myChildren;
int             myMaxChildren, myCurChild;
int             myID;

/*************************************************************************/

public:
CMesh  ();
CMesh  (int maxobj, int maxlight);
~CMesh ();

void             Init               (int maxobj, int maxlight);
void             New                ();
void             Delete             ();
int              RetMaxPolys        ();
int              RetMaxLights       ();
int              RetCurPolys        ();
int              RetCurLights       ();
int              RetID              ();
void             SetID              (int id);
int              polyOk             ();
int              polyOk             (int n);
int              lightOk            ();
int              lightOk            (int n);
void             SetMaterial        (float d, float s, float e);
void             SetMaterial        (CGraphMaterial &material);
CGraphMaterial & RetMaterial        ();
void             SetCenter          (float x, float y, float z);
void             Move               (float x, float y, float z);
CPoint3D &       RetCenter          ();
CMatrix4x4 &     RetTransformMatrix ();
CBone &          RetBone            ();
void             Rotate             (CMatrix4x4 &m);
void             Rotate             (float x, float y, float z);
void             SetAngle           (float x, float y, float z);
void             SetMatrix          (CMatrix4x4 &m);
void             Clone              (CMesh * mesh);
void             Clone              (CMesh * mesh, CPolygon * polys);
void             operator =         (CMesh &that);

void             AddPoly            (CPolygon * poly);
void             AddPolys           (CPolygon * polys, int n);
CPolygon *       RetPoly            (int n);
void             AddLight           (CLightSource * light);
void             AddLights          (CLightSource * light, int n);
CLightSource *   RetLight           (int n);

void             SetParent          (CMesh * parent);
void             InitChildren       (int num);
void             SetChild           (CMesh * child);
int              RetCurChildren     ();
int              RetMaxChildren     ();
CMesh *          RetParent          ();
CMesh *          RetChild           (int num);
};
```

```
    ////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////
  ////////////////////  CLASS CSTANDARDPRIMITIVE  ////////////////////
 ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
```

```
// Типы стандартных примитивов
#define STDPR_SPHERE 1
#define STDPR_CUBE   2
#define STDPR_CONE   3
#define STDPR_TORUS  4
#define STDPR_TEAPOT 5

class CORE3D_API CStandardPrimitive
{
private:
CPoint3D myCenter;
CColor   myColor;
float *  myParams;
int      myType, myParamNum;

/*************************************************************************/

public:
CStandardPrimitive  ();
CStandardPrimitive  (int type, float * params);
~CStandardPrimitive ();

void       Init (int type, float * params);
void       InitSphere (float * params);
void       InitCube (float * params);
void       InitCone (float * params);
void       InitTorus (float * params);
void       InitTeapot (float * params);
int        RetType ();
float      RetParam (int nparam);
void       SetCenter (float x, float y, float z);
CPoint3D & RetCenter ();
```

```
void       SetColor (float r, float g, float b, float a);
CColor &   RetColor ();
void       Move (float x, float y, float z);
};

#endif
```

Файл CPPVM\CPPVM.h

```
#if !defined (CPPVM_INTERFACE_FILE_LOADED)
#define CPPVM_INTERFACE_FILE_LOADED

#pragma once

#include "..\LinkerDataTypes\stdafx.h"
#include <objbase.h>
#include <memory>

#include "..\Linker\IVM.h"

typedef SParam (CObj::*pObjFunc) (const CParams&);
typedef SParam (*pStatFunc) ( const CParams&);

struct ObjFunc
    {
    virtual SParam operator() ( const CParams& params) = 0;
    };

// {527B3AB0-022A-47a3-B38F-52937E764B68}
static const GUID IID_ICPPVM =
{ 0x527b3ab0, 0x22a, 0x47a3, { 0xb3, 0x8f, 0x52, 0x93, 0x7e, 0x76, 0x4b, 0x68 } };

interface ICPPVM : public IVM
{
    virtual SClassRef registerClass ( int className, int varName) = 0;
    virtual SClassRef registerClass ( const char* className, const char* varName) = 0;
    virtual SClassRef registerBaseClass ( int className, int baseClassName) = 0;
    virtual SClassRef registerBaseClass ( const char* className, const char* baseClassName) = 0;

    virtual Func* registerFunc ( SClassRef cl, int funcName, pObjFunc func, const CParamsTypes& types) =
0;
    virtual Func* registerFunc ( SClassRef cl, const char* funcName, pObjFunc func, const CParamsTypes&
types) = 0;

    virtual StaticFunc* registerStaticFunc ( SClassRef cl, int funcName, pStatFunc func, const
CParamsTypes& types) = 0;
    virtual StaticFunc* registerStaticFunc ( SClassRef cl, const char* funcName, pStatFunc func, const
CParamsTypes& types) = 0;

    virtual StaticFunc* registerObjectFuncAsStatic ( SClassRef cl, int funcName, std::auto_ptr<ObjFunc>
pFunc, const CParamsTypes& types) = 0;
    virtual StaticFunc* registerObjectFuncAsStatic ( SClassRef cl, const char* funcName,
std::auto_ptr<ObjFunc> pFunc, const CParamsTypes& types) = 0;
};

#endif
```

Файл CPPVM\CPPVMCore.h

```
#if !defined (CPPVMCORE_FILE_LOADED)
#define CPPVMCORE_FILE_LOADED

#pragma once

#ifdef CPPVM_EXPORTS
#define CPPVM_API __declspec(dllexport)
#else
#define CPPVM_API __declspec(dllimport)
#endif

#include "CPPVM.h"
#include "system.h"
#include <memory>

//--------------------------------------------------------------------------

class StatFunc : public ObjFunc
    {
    pStatFunc myPFunc;

    public:
    StatFunc ( pStatFunc pFunc = 0);
    ~StatFunc();

    SParam operator() ( const CParams& params);
    };
```

```
class ObjFuncPtr
    {
    private:
    std::auto_ptr<ObjFunc> myPFunc;

    public:
    ObjFuncPtr ( std::auto_ptr<ObjFunc> pFunc = std::auto_ptr<ObjFunc> ());
    ObjFuncPtr ( const ObjFuncPtr& right);
    ObjFuncPtr& operator= ( const ObjFuncPtr& right);

    std::auto_ptr<ObjFunc>& getPFunc() const;

    SParam operator() ( const CParams& params);
    };

//---------------------------------------------------------------------------

struct CPPFunc
    {
    int mySelInL;
    int myName;
    int myClassName;
    pObjFunc myFunc;
    CParamsTypes myTypes;

    CPPFunc (int className = 0, int funcName = 0, const pObjFunc& func = NULL);
    virtual ~CPPFunc();

    CPPFunc& addParamType ( SParamTypeInfo t = SParamTypeInfo());

    virtual bool operator== (const CPPFunc& func) const;
    };

struct CPPStaticFunc
    {
    int myName;
    int myClassName;
    ObjFuncPtr myFunc;
    CParamsTypes myTypes;

    CPPStaticFunc ( int className = 0, int funcName = 0, const ObjFuncPtr& func = ObjFuncPtr());
    virtual ~CPPStaticFunc();

    CPPStaticFunc& addParamType ( SParamTypeInfo t = SParamTypeInfo());

    virtual bool operator== ( const CPPStaticFunc& func) const;
    };

//---------------------------------------------------------------------------

struct CPPClass
    {
    int myClassName;
    SClassRef myClassRef;
    CStore<CPPFunc> myFuncs;
    CStore<CPPStaticFunc> myStaticFuncs;

    CPPClass (int className = 0);
    virtual ~CPPClass();

    virtual int getFuncSelector (int funcName);
    virtual int getStaticFuncSelector (int funcName);

    virtual bool operator== (const CPPClass& cl) const;
    };

//---------------------------------------------------------------------------

class CPPVM_API CCPPVM : public ICPPVM
    {
    protected:
    CCont<CPPClass*> myClasses;

    public:
    CCPPVM();

    SParam staticCall ( int className, int classSel, int funcName, int funcSel, const CParams& params);
    SParam call ( CObj* obj, int className, int classSel, int funcName, int funcSel, const CParams&
params);

    DEFINE_IUNKNOWN();

    CCont<CPPClass*> getClasses() { return myClasses; } //bad Style

    pObjFunc rFuncPtr (Func* func);
    pObjFunc rFuncPtr (int className, int classSelector, int funcName, int funcSelector); //uses selectors
if non zero otherwise get 'em from names
    ObjFuncPtr& rStatFuncPtr (StaticFunc* func);
    ObjFuncPtr& rStatFuncPtr (int className, int classSelector, int funcName, int funcSelector); //uses
selectors if non zero otherwise get 'em from names

    int getClassSelector (int className);
    int getFuncSelector (int className, int funcName);
    int getStaticFuncSelector (int className, int funcName);

    CPPClass* getClassPtr ( int className);
    CPPClass* getClassPtrByIndex ( int index) { return myClasses[index]; }
```

```
    SClassRef registerClass ( int className, int varName);
    SClassRef registerClass ( const char* className, const char* varName);
    SClassRef registerBaseClass ( const char* className, const char* baseClassName);
    //SClassRef registerBaseClass ( const char* className, const char* baseClassName);

    Func* registerFunc ( SClassRef cl, const char* funcName, pObjFunc func, const CParamsTypes& types);
    //Func* registerFunc ( SClassRef cl, const char* funcName, pObjFunc func, const CParamsTypes& types);

    StaticFunc* registerStaticFunc ( SClassRef cl, const char* funcName, pStatFunc func, const
CParamsTypes& types);
    //StaticFunc* registerStaticFunc ( SClassRef cl, const char* funcName, pStatFunc func, const
CParamsTypes& types);

    StaticFunc* registerObjectFuncAsStatic ( SClassRef cl, int funcName, std::auto_ptr<ObjFunc> pFunc,
const CParamsTypes& types);
    StaticFunc* registerObjectFuncAsStatic ( SClassRef cl, const char* funcName, std::auto_ptr<ObjFunc>
pFunc, const CParamsTypes& types);


    SClassRef registerBaseClass(int,int);

    Func* registerFunc(SClassRef,int,pObjFunc,const CParamsTypes &);
    StaticFunc* registerStaticFunc(SClassRef,int,pStatFunc,const CParamsTypes &);

    };

//extern CCPPVM* CPPVM;

//--------------------------------------------------------------------------------

SClassRef RegisterClass ( SClassRef* cl, int className, int varName);
SClassRef RegisterClass ( SClassRef* cl, const CString& className, const CString& varName);
SClassRef RegisterBaseClass ( SClassRef* cl, const CString& baseClassName);
SClassRef RegisterBaseClass ( SClassRef* cl, int baseClassName);
Func* RegisterFunc  ( SClassRef*  cl, const CString& name, pObjFunc func, const CParamsTypes& types);
StaticFunc* RegisterStaticFunc  ( SClassRef*  cl, const CString& name, pStatFunc func, const CParamsTypes&
types);

//--------------------------------------------------------------------------------

#endif
```

Файл CPPVM\system.h

```
#ifndef CPPVM_SYSTEM_FILE_LOADED
#define CPPVM_SYSTEM_FILE_LOADED

#pragma once

#include "..\Loader\Loader.h"
#include "..\Linker\linker.h"
#include "cppvm.h"
#include "..\Console\Console.h"

namespace CPPVM
{
    extern ILoader* Loader;
    extern ILinker* Linker;
    extern ICPPVM* CPPVM;
    extern IConsole* Console;
};

#endif
```

Файл ExeptionEngine\Exeption.h

```
//Exeption Engine Version 1.5 Copyrigth (c) by Vasfed, 2002

//link with Exeption.lib, and make Exeption.dll accessible to make it working

#ifndef __EXEPTION_ENGINE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __EXEPTION_ENGINE_HEADER_FILE_BY_VASFED_INCLUDED__

#include <stdio.h>

#ifdef EXEPTION_EXPORTS
    #define EXEPTION_API __declspec(dllexport)
    #else
    #define EXEPTION_API __declspec(dllimport)
    #endif

#ifndef __PROGRAMMER
    #define __PROGRAMMER "Unknown"
    #endif

#ifndef __MODULE_NAME
```

```
    #define __MODULE_NAME "Unknown"
    #endif

#define Try try
#define TryT(comments) Message("Entering try block, comments:'" comments "'"); try
#define TryCT(typeOfMessage,comments) MessageC("Entering try block, comments:'" comments "'",
typeOfMessage); try

#define Throw               throw((Exeption (__PROGRAMMER, __MODULE_NAME, __FILE__, __FUNCTION__,
__LINE__))

#define ThrowT(text)        throw (Exeption (__PROGRAMMER, __MODULE_NAME, __FILE__, __FUNCTION__, __LINE__,
general, text));
#define ThrowC(type)        throw (Exeption (__PROGRAMMER, __MODULE_NAME, __FILE__, __FUNCTION__, __LINE__,
type, "No description"));
#define ThrowCT(type,text) throw (Exeption (__PROGRAMMER, __MODULE_NAME, __FILE__, __FUNCTION__, __LINE__,
type, text));
#define ThrowCTP(type,format,param) throw (Exeption (1, __PROGRAMMER, __MODULE_NAME, __FILE__,
__FUNCTION__, __LINE__, type, text, param));

#define Assert(expression)             if(!(expression))      throw (Exeption (2, __PROGRAMMER,
__MODULE_NAME, __FILE__, __FUNCTION__, __LINE__, assert, #expression));
#define AssertT(expression,desc)       if(!(expression))      throw (Exeption (__PROGRAMMER,
__MODULE_NAME, __FILE__, __FUNCTION__, __LINE__) (assert, "assertion failed: '%s', (%s)", #expression,
desc));
#define AssertC(expression,code)       if(!(expression))      throw (Exeption (__PROGRAMMER,
__MODULE_NAME, __FILE__, __FUNCTION__, __LINE__) (code, "assertion failed:'%s'", #expression));
#define AssertTC(expression,code,desc) if(!(expression))      throw (Exeption (__PROGRAMMER,
__MODULE_NAME, __FILE__, __FUNCTION__, __LINE__) (code, "assertion failed:'%s', (%s)", #expression,
desc));

#undef assert

#define Catch() catch (Exeption exeption){ RegisterExeptionCatching (&exeption, __PROGRAMMER,
__MODULE_NAME, __FILE__, __FUNCTION__, __LINE__);
#define CatchN(ExeptionObjectName) catch (Exeption ExeptionObjectName){ RegisterExeptionCatching
(&ExeptionObjectName, __PROGRAMMER, __MODULE_NAME, __FILE__, __FUNCTION__, __LINE__);
#define CatchT(comments) catch (Exeption exeption){ RegisterExeptionCatching (&exeption, __PROGRAMMER,
__MODULE_NAME, __FILE__, __FUNCTION__, __LINE__, comments);
#define CatchNT(ExeptionObjectName,comments) catch (Exeption ExeptionObjectName){ RegisterExeptionCatching
(&ExeptionObjectName, __PROGRAMMER, __MODULE_NAME, __FILE__, __FUNCTION__, __LINE__, comments);

#define Handled()          RegisterExeptionHandling (&exeption, __PROGRAMMER, __MODULE_NAME, __FILE__,
__FUNCTION__, __LINE__);
#define HandledN(exeption) RegisterExeptionHandling (&exeption, __PROGRAMMER, __MODULE_NAME, __FILE__,
__FUNCTION__, __LINE__);
#define HandledT(comments) RegisterExeptionHandling (&exeption, __PROGRAMMER, __MODULE_NAME, __FILE__,
__FUNCTION__, __LINE__, comments);
#define HandledNT(exeption,comments) RegisterExeptionHandling (&exeption, __PROGRAMMER, __MODULE_NAME,
__FILE__, __FUNCTION__, __LINE__, comments);

//Now defines for Messaging

#define Message(text) CMsg (__PROGRAMMER, __MODULE_NAME, __FILE__, __FUNCTION__, __LINE__, CMsg::simple,
text)
#define MessageC(text,typeName) CMsg (__PROGRAMMER, __MODULE_NAME, __FILE__, __FUNCTION__, __LINE__,
CMsg::##typeName, text)

#define Trace (CMsg(1, __PROGRAMMER, __MODULE_NAME, __FILE__, __FUNCTION__, __LINE__))
#define TraceEx (CMsg(0, __PROGRAMMER, __MODULE_NAME, __FILE__, __FUNCTION__, __LINE__))

//-------------------------------------------------------------------------------------

//Header itself:
enum ExeptionTypes
    {
    unknown,
    general,
    badParams,
    runtime,
    assert,
    memoryLeak,
    memoryDamage,
    programResourcesLeak,
    programFail,
    systemResourcesLeak, //ex. кончились свободные threads
    systemFail
    };

//-------------------------------------------------------------------------------------

struct EXEPTION_API Exeption
{
        Exeption(const char* programmer = NULL, const char* module = NULL,
            const char* file = NULL, const char* function = NULL, int line = NULL);
                //constructor for using with defines

        Exeption(int, const char* programmer, const char* module,
            const char* file, const char* function, int line,
            int type, const char* text);
                //constructor for manual use
        Exeption (Exeption&);
        ~Exeption();

        void Handle();
        int PrintText (char* stringToWriteTo);
        int PrintText (FILE* file = stdout);
```

```
        Exeption& operator ()();
        Exeption& operator ()(int code);
        Exeption& operator ()(const char* text, int code = general);
        Exeption& operator ()(int code, const char* format, ...);

        enum Types
            {
            unknown,
            general,
            badParams,
            runtime,
            assert,
            memoryLeak,
            memoryDamage,
            programResourcesLeak,
            programFail,
            systemResourcesLeak, //ex. кончились свободные threads
            systemFail
            };

        static const char* TypesStr[];
        int   m_creationTime; //using timeGetTime

    //protected:
        char* m_programmer;
        char* m_file;
        char* m_module;
        char* m_function;
        int   m_line;

        char* m_text;

        int m_type;

        bool m_handled;
        bool m_textWasOrdered, m_textShouldBeDeleted;
};

//---------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------

struct EXEPTION_API CMsg // очень часто бывает так, что исключение - перебор, а сказать надо
{
    CMsg (int, const char*, const char*, const char*,  const char* function, const int);
            //for using in defines

    CMsg (const char* programmer = NULL, const char* module = NULL,
            const char* file = NULL, const char* function = NULL, const int line = NULL, int type =
simple,
            const char* text = "No text");
            //for manual use
    ~CMsg();

    operator ()(int code = simple);
    operator ()(int code, const char* format, ...);
    operator ()(const char* format, ...); //for 'Trace'



    void PrintText (char* stringToWriteTo);
    void PrintText (FILE* textfile);

    enum Type
        {
        // оно же warningLevel
        simple,
        hint,
        trace,
        warning,
        error, //non critical
        preexeption,
        unknown
        };

    static const char* CodeStr[];

    const char*
        m_programmer,
        *m_module,
        *m_file,
        *m_function;

    bool m_textWasOrdered;
    char* m_text;
    int m_type;

    const int
        m_line;

    int m_creationTime; //using timeGetTime(), not clock()!
    bool noPos;
};

//---------------------------------------------------------------------------------------

//Global functions:
```

```
void    EXEPTION_API RegisterExeptionCatching (Exeption*, const char* programmer = "Unknown",
                              const char* module = "Unknown",
                              const char* file = "Unknown",  const char* function = "Unknown",
                              int line = -1,
                              const char* text = "NoComments");

void    EXEPTION_API RegisterExeptionHandling (Exeption*, const char* programmer = "Unknown",
                              const char* module = "Unknown",
                              const char* file = "Unknown",  const char* function = "Unknown",
                              int line = -1,
                              const char* text = "NoComments");

int     EXEPTION_API SetDumpFile           (FILE* textfile = stdout, bool closeOnEnd = false);
void    EXEPTION_API BeginDump             (void);
void    EXEPTION_API EndDump               (void);
void    EXEPTION_API CreateDumpFile           (const char* filename = "ExeptionDump.log");
void    EXEPTION_API GetDumpFileName       (char* str);
EXEPTION_API FILE*  GetDumpFileHandle      (void);
void    EXEPTION_API PromptExeptionEngine  (void);

//Not working yet, does nothing:

typedef void (*PMessageHandlerFunction)(CMsg*);
typedef void (*PExeptionHandlingFunction)(Exeption*);

void EXEPTION_API SetMessageHandler               (PMessageHandlerFunction);
void EXEPTION_API SetExeptionHandler          (PExeptionHandlingFunction);
void EXEPTION_API SetUnhandledExeptionHandler (PExeptionHandlingFunction);

#endif
```

Файл ExeptionEngine\ExeptionUtils.h

```
//ExeptionUtils.h - header file for Exeption utilities
//such like dumps, error messages, tracing

#ifndef __EXEPTION_UTIL_HEADER_FILE_BY_VAFFED_INCLUDED__
#define __EXEPTION_UTIL_HEADER_FILE_BY_VAFFED_INCLUDED__

#include <Exeption.h>
#include <stdio.h>

void UnhandledExeption (Exeption*); //Registers, if an Exeption was not handled (formally)
void RegisterExeption  (Exeption*); //before handling Exeption it is Registered,
                                    //if dump is exist, exeption will be dumped
void RegisterMessage       (CMsg*); //called automatically

//functions from 'exeption.h' are implemented in Utils too,
//but their prototypes were placed there for convinience

#endif
```

Файл ExeptionEngine\resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Exeption.rc

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        101
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Файл LAgents\AgentMessage.h

```
// AgentMessage.h - header
// created 12.12.2002 by header creator
#pragma once
#ifndef __AGENTMESSAGE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __AGENTMESSAGE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\AICore\Agents.h"
#include "..\SmallVM\SmallVM.h"

#include "AgentMessageInt.h"
```

```
class CAgentMessage : public CObj, public AgentMessage
{
private:
    static SClassRef sClass;

public:
    CAgentMessage();

    static bool selfReg();

    SParam rSender (const CParams&);
    SParam rReceiver (const CParams&);
    SParam rType (const CParams&);
    SParam rCode (const CParams&);
    SParam rY (const CParams&);
    SParam rX (const CParams&);
    SParam rZ (const CParams&);

    SParam rAiName (const CParams&);

    SParam setX        (const CParams&); //1: float
    SParam setY        (const CParams&); //float
    SParam setZ        (const CParams&);
    SParam setCode     (const CParams&);
    SParam setType     (const CParams&);
    SParam setReceiver (const CParams&);
    SParam setSender   (const CParams&);

    SParam Post    (const CParams&);
};

#endif //__AGENTMESSAGE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл LAgents\AgentMessageInt.h

```
// AgentMessageInt.h - header
// created 12.12.2002 by header creator
#pragma once
#ifndef __AGENTMESSAGEINT_HEADER_FILE_BY_VASFED_INCLUDED__
#define __AGENTMESSAGEINT_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\AiCore\Agents.h"

/*
the following structure is defined in Agents.h

struct AgentMessage
{
    int     sender;     // Agent who send this message (handle in Server)
    int     receiver;   // 0-> depends on type: broadcast, or local (to nearest agent...)
    int     type;       // Serving type = Broadcast, Private etc. see MessageCodes
    int     ainame;     // - who can understand it? (target AI name)
    int     code;       // user-defined message type
    float   x;
    float   y;
    float   z;

    void*   data;       // anything, deleted, if freedata is 1
    int     data1;      // anything
    int     datasize;   // size of data pointed by data
    bool    freedata;   // indictes, if *data should be freed by deleting it

    enum Types {
                zMT_NONE        = 0,
                zMT_BROADCAST   = 1,
                zMT_SOUND       = 2,
                zMT_TARGETED    = 3,
                zMT_TOUSER      = 10};
};
*/

#endif //__AGENTMESSAGEINT_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл LAgents\Agents.h

```
// Agents.h - base interfaces for AI
// created 10.09.2002 by header creator
#pragma once
#ifndef __AGENTS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __AGENTS_HEADER_FILE_BY_VASFED_INCLUDED__

#include <objbase.h>
#include "MessageLanguages.h"
#include "..\LinkerDataTypes\stdafx.h"

// {935FB5CA-D30C-4b5f-8236-7A0637D5173E}
static const GUID IID_IAgentsFactory =
```

```
{ 0x935fb5ca, 0xd30c, 0x4b5f, { 0x82, 0x36, 0x7a, 0x6, 0x37, 0xd5, 0x17, 0x3e } };

class agentPassTurn {}; //- agent exception

interface IAgentsFactory: public IUnknown
{
    virtual int CreateNewAgent (int x = -1 /*-1=random*/, int y = -1) PURE;
    virtual void Step() PURE;
    virtual void Init() PURE;

};
//-----------------------------
//additional structures and classes:

struct AgentMessage
{
    int     sender;     // Agent who send this message (handle in Server)
    int     receiver;   // 0-> depends on type: broadcast, or local (to nearest agent...)
    int     type;       // Serving type = Broadcast, Private etc. see MessageCodes
    int     ainame;     // - who can understand it? (target AI name)
    int     code;       // user-defined message type
    float   x;
    float   y;
    float   z;
    void*   data;       // anything, deleted, if freedata is 1
    int     data1;      // anything
    int     datasize;   // size of data pointed by data
    bool    freedata;   // indictes, if *data should be freed by deleting it
    enum Types {zMT_NONE       = 0, zMT_BROADCAST = 1, zMT_SOUND     = 2, zMT_TARGETED  = 3, zMT_TOUSER
    = 10};
};

interface IAgent
{
    virtual int    Go() PURE;
    virtual int    Ok() PURE;
    virtual void   Init    () PURE;
    virtual int    MessageProc(AgentMessage* msg) PURE;

    virtual void   Ping    (int agent) PURE;

    virtual void   CollisionDetected (void* reserved){};

    //Vector    m_vector;
    float   m_x, m_y, m_z;

    bool    m_shouldBeProcessed;
    bool    m_ShouldExit;

    //Handles (often all the hashes are similar)
    int     m_index;    //in AgentsServer
    int     m_mesh;     //in Renderer
    int     m_physHash; //in Physics module

    int     m_ainame;   // to make different AI recognizeable
};

#endif //__AGENTS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл LAgents\LAgent.h

```
// LAgent.h - header
// created 03.11.2002 by header creator
#pragma once
#ifndef __LAGENT_HEADER_FILE_BY_VASFED_INCLUDED__
#define __LAGENT_HEADER_FILE_BY_VASFED_INCLUDED__

class ILinkedAgent : public IAgent, public CObj
{
    static SClassRef sClass;

    int     Go()
        { return this->call ("Go"); }

    SParam lGo ( const CParams& p)
        { return cGo(); }

    virtual cGo() {};

    int     Ok()
        { return this->call ("Ok"); }

    SParam lOk ( const CParams& p)
        { return cOk(); }

    virtual cOk() {};

    void    Init()
        { this->call ("Init"); }

    SParam lInit ( const CParams& p)
        { cInit(); }
```

```
    virtual void cInit() {};

    int     MessageProc(AgentMessage* msg)
        { return this->call ("MessageProc:", PARAMS reinterpret_cast<void*> ( msg)); }

    SParam lMessageProc ( const CParams& p)
        { return cMessageProc ( reinterpret_cast<AgentMessage*> ( p[0].rPVoid())); }

    virtual int cMessageProc ( AgentMessage* msg) {};

    void    Ping    (int agent)
        { this->call ("Ping:", PARAMS agent); }

    SParam lPing ( const CParams& p)
        { return cPing ( p[0].rInt()); }

    virtual cPing ( int agent) {};

    void    CollisionDetected (void* reserved)
        { this->call ("CollisionDetected:", PARAMS reserved); }

    SParam lCollisionDetected ( const CParams& p)
        { return cCollisionDetected ( p[0].rPVoid()); }
    virtual cCollisionDetected ( void* reserved) {};


    SParam rX(const CParams&)
        { return m_x;}
    SParam rY(const CParams&)
        { return m_y;}
    SParam rZ(const CParams&)
        { return m_z;}

    SParam rIndex(const CParams&)
        { return m_index;}
    SParam rSHBC(const CParams&)
        { return m_shouldBeProcessed;}
    SParam setSHBP(const CParams& par)
        { return m_shouldBeProcessed = par[0];}


    bool    m_shouldBeProcessed;
    bool    m_ShouldExit;

    //Handles (often all the hashes are similar)
    int     m_index;    //in AgentsServer
    int     m_mesh;     //in Renderer
    int     m_physHash; //in Physics module

    int     m_ainame;   // to make different AI recognizeable
};

#endif //__LAGENT_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл LAgents\LAgentCore.h

```
// LAgent.h - header
// created 03.11.2002 by header creator
#pragma once
#ifndef __LAGENT_HEADER_FILE_BY_SE_INCLUDED__
#define __LAGENT_HEADER_FILE_BY_SE_INCLUDED__

#include "..\AICore\Agents.h"
#include "..\SmallVM\SmallVM.h"

IAgent* CreateAgent ( int i);

class CAgentData : public CObj, public IAgent
{
private:
    static SClassRef sClass;

public:
    CAgentData();

    static bool selfReg();

    SParam/*eBool*/    operator==             ( const CParams& pars); //1: 0 - eAny

    SParam rX(const CParams&);
    SParam rY(const CParams&);
    SParam rZ(const CParams&);

    SParam rIndex(const CParams&);
    SParam rSHBP(const CParams&);
    SParam setSHBP(const CParams& par);

};

class CLinkedAgent : public CAgentData
{
```

```
private:
    CObjRef myAgent;
public:
    CLinkedAgent();
    CLinkedAgent ( const char* str);

    CObjRef rLinkedAgent() { return myAgent; }

    int     Go();
    int     Ok();
    void    Init();
    int     MessageProc(AgentMessage* msg);
    void    CollisionDetected (void* reserved);
};

#endif //__LAGENT_HEADER_FILE_BY_SE_INCLUDED__
```

Файл LAgents\LAgentSimulator.h

```
// LAgentSimulator.h - header
// created 10.12.2002 by header creator
#pragma once
#ifndef __LAGENTSIMULATOR_HEADER_FILE_BY_VASFED_INCLUDED__
#define __LAGENTSIMULATOR_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\AICore\Agents.h"
#include "..\SmallVM\SmallVM.h"

class CAgentData : public CObj, public IAgent
{
private:
    static SClassRef sClass;

public:
    CAgentData();

    static bool selfReg();

    SParam rX(const CParams&);
    SParam rY(const CParams&);
    SParam rZ(const CParams&);

    SParam rIndex(const CParams&);
    SParam rSHBP(const CParams&);
    SParam setSHBP(const CParams& par);

};

class CLinkedAgent : public CAgentData
{
private:
    CObjRef myAgent;
public:
    CLinkedAgent();
    CLinkedAgent ( const char* str);

    int     Go();
    int     Ok();
    void    Init();
    int     MessageProc(AgentMessage* msg);
    void    CollisionDetected (void* reserved);
};

#endif //__LAGENTSIMULATOR_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл LAgents\System.h

```
// System.h - header
// created 03.11.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\CPPVM\CPPVM.h"
#include "..\Linker\Linker.h"
#include "..\SmallVM\SmallVM.h"
#include "..\AgentsServer\AgentsServer.h"
#include "..\console\console.h"

namespace LA
    {
        extern IAgentsServer* AServer;
        extern IAgentsServerAgents* Server;
        extern ISmallVM* SmVM;
        extern ICPPVM* CPPVM;
        extern ILinker* Linker;
        extern IConsole* Console;
    };
```

```
#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл LAgents\LServer.h

```
#include "..\AgentsServer\Agents.h"
#include "..\Linker\Linker.h"
#include "system.h"

class CLinkedAgentsServerAgents : public CObj
{
private:
    static SClassRef sClass;
public:
    CLinkedAgentsServerAgents();

    static bool selfReg();

    SParam/*self*/     CreateAgentByName      ( const CParams& pars); //1: 0 - eStr.
    SParam/*self*/     CreateAgentByAI        ( const CParams& pars); //1: 0 - eInt.
    SParam/*eInt*/     RemoveAgent            ( const CParams& pars); //1: 0 - eInt.
    SParam/*LAgent**/  GetAgent               ( const CParams& pars); //1: 0 - eInt.
    SParam/*eInt*/     FindAgent              ( const CParams& pars); //1: 0 - LAgent*.
    SParam/*LAgent*/   GetAgentOfNumber       ( const CParams& pars); //2: 0 - eStr 1 - eInt
    SParam/*self*/     ReHash                 ( const CParams& pars); //0.
    SParam/*eInt*/     GetNumAgents           ( const CParams& pars);//0.
    SParam/*eInt*/     GetNumAgentsOf         ( const CParams& pars); //1: 0 - eStr.

    SParam/*eInt*/     RegisterAgentCreator   ( const CParams& pars);//1: 0 - eStr
    SParam/*eInt*/     FindAgentCreator       ( const CParams& pars);//1: 0 - eStr
    SParam/*eInt*/     GetMaxCreators         ( const CParams& pars);//0.
    SParam/*eStr*/     GetAgentCreatorName    ( const CParams& pars); //1: 0 - eInt

    ////Communication between agents:
    //virtual IAgent*      GetAgent              (int)                                        PURE;
    //virtual int          SendMessage           (AgentMessage)                               PURE;
    //
    ////Map:
    //virtual inline int   GetMapUnitCode        (float x, float y) const                     PURE;
    //virtual inline int*  GetPMapUnitData       (float x, float y) const                     PURE;
    //virtual inline int   GetMapUnitData        (float x, float y, int index = 0) const      PURE;
    //virtual inline int   GetMapUnitDataSize    (float x, float y) const                     PURE;
    //virtual inline void  ModifyMapUnitData     (float x, float y, int index, int newvalue) PURE;

    //virtual inline int       GetMapSizeX           (void)  const PURE;
    //virtual inline int       GetMapSizeY           (void)  const PURE;
    //virtual inline float GetMapSquareSize      (void)  const PURE;
    //virtual float        GetLandscapeLevel     (float x, float y, int reserved = 0)   PURE;
    //
    ////Physics
    //virtual void     SetSpeed              (int agent, float x, float y, float z = 0) PURE;
    //virtual void     SetAcceleration       (int agent, float x, float y, float z = 0) PURE;
    //virtual void     GetSpeed              (int agent, float*x, float*y, float*z = 0) PURE;
    //virtual void     GetAcceleration       (int agent, float*x, float*y, float*z = 0) PURE;
    //virtual float    GetFallingAcceleration (void)                                    PURE;
    //virtual void     AddToAcceleration     (int agent, float x, float y, float z = 0) PURE;
    //virtual void     AddToSpeed            (int agent, float x, float y, float z = 0)  PURE;
    //
    ////User interface
    ////(not all implemented yet)
    //virtual void     PlaySound             (int agent, int soundcode)                 PURE;
    //virtual void     PlayAnimation         (int agent, int animation, void* params)   PURE;
    //virtual void     SayToUser             (char* format, ...)                        PURE;

    SParam lSayToUser ( const CParams& par); //1 - LinkedAgent, 2 - string
    SParam lSetSpeed ( const CParams& par); //1 - LinkedAgent, 2 - float x, 3 - float y, 4 - float z
    SParam lSetAcceleration ( const CParams& par); //1 - LinkedAgent, 2 - float x, 3 - float y, 4 - float
z
    SParam lGetSpeedX ( const CParams& par); //1 - LinkedAgent
    SParam /*float*/ lGetSpeedY ( const CParams& par); //1 - LinkedAgent
    SParam /*float*/ lGetSpeedZ ( const CParams& par); //1 - LinkedAgent
    SParam /*float*/ lGetAccelerationX ( const CParams& par); //1 - LinkedAgent
    SParam /*float*/ lGetAccelerationY ( const CParams& par); //1 - LinkedAgent
    SParam /*float*/ lGetAccelerationZ ( const CParams& par); //1 - LinkedAgent
    SParam /*float*/ lGetFallingAcceleration ( const CParams&); //1 - LinkedAgent
    SParam /*float*/ lSetAgentPosition ( const CParams& par); // 4:  0 - LinkedAgent 1 - float 2 - float
3- float

    SParam /*float*/ GetLandscapeLevelAtXY ( const CParams& par); //1 - float x 2 - float y

};
```

Файл Landscape\landscap.h

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
```

```
/*** LANDSCAP.H                                                              ***/
/*** ОБЪЯВЛЕНИЕ БАЗОВЫХ КЛАССОВ ДЛЯ СОЗДАНИЯ И ОТОБРАЖЕНИЯ ЛАНДШАФТОВ ***/
/*** СОДЕРЖИТ КЛАССЫ:                                                        ***/
/***      CFLOATMATRIX                                                     ***/
/***      CPOINTMATRIX                                                     ***/
/***      CHORIZONT                                                        ***/
/***      CLANDSCAPE                                                       ***/
/***                                                                       ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ              ***/
/***                                                                       ***/
/***                                                                       ***/
/*****************************************************************************/
/*****************************************************************************/

#include <math.h>

#ifdef LANDSCAPE_EXPORTS
#define LANDSCAPE_API __declspec(dllexport)
#else
#define LANDSCAPE_API __declspec(dllimport)
#endif

#ifndef _LANDSCAPE_H_
#define _LANDSCAPE_H_

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CFLOATMATRIX ***********************************/
/*****************************************************************************/
/*****************************************************************************/

class LANDSCAPE_API CFloatMatrix
{
protected:
float * myMatrix;
int     mySizeX, mySizeY;

friend class CRenderer;

/*****************************************************************************/

public:
CFloatMatrix  ();
CFloatMatrix  (int sizex, int sizey);
~CFloatMatrix ();

void    Init        (int sizex, int sizey);
int     RetX        ();
int     RetY        ();
int     ok          (int x, int y);
float & RetElement  (int x, int y);
float & operator () (int x, int y);
void    Fill        (float n);
};

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CPOINTMATRIX ***********************************/
/*****************************************************************************/
/*****************************************************************************/

#define FLOATSINPOINT 3

class LANDSCAPE_API CPointMatrix
{
protected:
float * myMatrix;
int     mySizeX, mySizeY;

friend class CRenderer;

/*****************************************************************************/

public:
CPointMatrix  ();
CPointMatrix  (int sizex, int sizey);
~CPointMatrix ();

void    Init        (int sizex, int sizey);
int     RetX        ();
int     RetY        ();
int     ok          (int x, int y, int z);
float & RetElement  (int x, int y, int z);
float & operator () (int x, int y, int z);
void    Fill        (float x, float y, float z);
};

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CHORIZONT **************************************/
/*****************************************************************************/
/*****************************************************************************/

#define FLOATSINCOLOR 4

class LANDSCAPE_API CHorizont : public CFloatMatrix
{
```

```
private:
float   myColor [FLOATSINCOLOR];
int     myDetailTexID, myGlobalTexID;
float   mySquareSize;
float   myTexRepX, myTexRepY;

friend class CRenderer;

/*************************************************************************/

public:
CHorizont  ();
CHorizont  (int x, int y);
CHorizont  (int x, int y, float sqrsize);
~CHorizont ();

void    Set              (int x, int y, float sqrsize);
void    SetColor         (float r, float g, float b, float a);
float * RetColor         ();
void    SetTexParameters (float x, float y);
float   RetTexX          ();
float   RetTexY          ();
void    SetDetailTexID   (int id);
int     RetDetailTexID   ();
void    SetGlobalTexID   (int id);
int     RetGlobalTexID   ();
void    SetSquareSize    (float sqrsize);
float   RetSquareSize    ();
};

/*************************************************************************/
/*************************************************************************/
/********************* CLASS CLANDSCAPE **********************************/
/*************************************************************************/
/*************************************************************************/

const int HEIGHT_MAP = 1;
const int NORMAL_MAP = 2;

class LANDSCAPE_API CLandscape
{
private:
CHorizont *    myHorizonts;
CPointMatrix * myNormalMaps;
int            myCurHor, myHorNum;

friend class CRenderer;

/*************************************************************************/

public:
CLandscape  ();
CLandscape  (int n);
~CLandscape ();

void           Init             (int n);
int            ok               ();
int            ok               (int n);
int            InitNewHor       ();
int            InitNewHor       (int x, int y, float sqrsize);
int            RetSize          ();
void           SetColor         (int handle, float r, float g, float b, float a);
float *        RetColor         (int handle);
void           SetDetailTexID   (int handle, int id);
int            RetDetailTexID   (int handle);
void           SetGlobalTexID   (int handle, int id);
int            RetGlobalTexID   (int handle);
void           SetTexParameters (int handle, float x, float y);
float          RetTexX          (int handle);
float          RetTexY          (int handle);
void           SetSquareSize    (int handle, float sqrsize);
float          RetSquareSize    (int handle);
void           Fill             (int handle, float n);
CHorizont &    RetHorizont      (int handle);
CHorizont *    RetPHorizont     (int handle);
CPointMatrix & RetNormalMap     (int handle);
CPointMatrix * RetPNormalMap    (int handle);
};

#endif
```

Файл Linker\IVM.h

```
#if !defined (IVM_FILE_LOADED)
#define IVM_FILE_LOADED

#include "stdafx.h"
#include "..\utilities\module.h"
#include <objbase.h>

interface IVM : public IUnknown
```

```
    {
    virtual SParam staticCall (int className, int classSel, int funcName, int funcSel, const CParams&
params) = 0;
    virtual SParam call ( CObj* obj, int className, int classSel, int funcName, int funcSel, const
CParams& params) = 0;
    virtual int getClassSelector (int className) = 0;
    virtual int getFuncSelector (int className, int funcName) = 0;
    virtual int getStaticFuncSelector (int className, int funcName) = 0;
    };

//---------------------------------------------------------------------------

#endif
```

Файл Linker\LinkerCore.h

```
#if !defined (LINKERCORE_FILE_LOADED)
#define LINKERCORE_FILE_LOADED

#pragma once

#ifdef LINKER_EXPORTS
#define LINKER_API __declspec(dllexport)
#else
#define LINKER_API __declspec(dllimport)
#endif

#include "linker.h"
#include "system.h"
#include "..\utilities\module.h"

class LINKER_API CLinker : public ILinker
    {
    public:
    DEFINE_IUNKNOWN();

    private:
    CCont<SClass*> myClasses;
    CCont<CString> myStrings;
    CScope myGlobalScope;

    public:
    CLinker();
    ~CLinker();

    CCont<SClass*>& getClasses();

    bool ok() const;

    SParam call ( CObj* obj, int name, const CParams& par = CParams());
    SParam call ( CObj* obj, int classSelector, int funcSelector, const CParams& par = CParams());
//selectors in linker
    SParam staticCall ( int clasName, int name, const CParams& par = CParams());
    Func*  rFunc ( int name, int className); // this func returns all inf about func and allows to call
it.

    SClass* getClassPtr ( int className);
    SClassRef getClassRef ( int className);
    SClassRef getClassRef ( const SClass* clPtr);
    int getClassSelector ( const CString& className);
    int getClassSelector ( int className);
    int getClassName ( int selectorInLinker);

    int registerName ( const CString& name);
    int checkName ( const CString& name);
    const CString& getName ( int name);

    CScope& getGlobalScope();

    SClassRef registerClass ( int name, int varName);
    Func* registerFunc ( const Func& func);
    StaticFunc* registerStaticFunc ( const StaticFunc& func);
};

#endif
```

Файл Linker\linker.h

```
#if !defined (LINKER_FILE_LOADED)
#define LINKER_FILE_LOADED

#include "..\LinkerDataTypes\stdafx.h"
#include <objbase.h>

// {D539AAA8-C5D3-4e6c-B2EC-5E413E58D45E}
static const GUID IID_ILinker =
{ 0xd539aaa8, 0xc5d3, 0x4e6c, { 0xb2, 0xec, 0x5e, 0x41, 0x3e, 0x58, 0xd4, 0x5e } };
```

```
interface ILinker : public IUnknown
    {
    virtual CCont<SClass*>& getClasses() = 0;

    virtual SParam call ( CObj* obj, int name, const CParams& par = CParams()) = 0;
    virtual SParam call ( CObj* obj, int classSelector, int funcSelector, const CParams& par = CParams())
= 0; //selectors in linker
    virtual SParam staticCall ( int clasName, int name, const CParams& par = CParams()) = 0;
    virtual Func*  rFunc ( int name, int className) = 0; // this func returns all inf about func and
allows to call it.

    virtual SClass* getClassPtr ( int className) = 0;
    virtual SClassRef getClassRef ( int className) = 0;
    virtual SClassRef getClassRef ( const SClass* clPtr) = 0;
    virtual int getClassSelector ( const CString& className) = 0;
    virtual int getClassSelector ( int className) = 0;
    virtual int getClassName ( int selectorInLinker) = 0;

    virtual int registerName ( const CString& name) = 0;
    virtual int checkName ( const CString& name) = 0;
    virtual const CString& getName ( int name) = 0;

    virtual CScope& getGlobalScope() = 0;

    virtual SClassRef registerClass ( int name, int varName) = 0;
    virtual Func* registerFunc ( const Func& func) = 0;
    virtual StaticFunc* registerStaticFunc ( const StaticFunc& func) = 0;
    };

#endif
```

Файл Linker\system.h

```
#pragma once
#ifndef LINKER_SYSTEM_FILE_LOADED
#define LINKER_SYSTEM_FILE_LOADED

#include "..\Loader\Loader.h"
#include "linker.h"
#include "..\console\Console.h"

extern ILoader* Loader;
extern ILinker* Linker;
extern IConsole* Console;

#endif
```

Файл Linker\vm templates.h

```
#if !defined (VM_TEMPLATES_FILE_LOADED)
#define VM_TEMPLATES_FILE_LOADED

#include "system.h"
#include "stdafx.h"
//#include "basic classes.h"

//---------------------------------------------------------------------------

template <class FuncPtr>
struct TFunc
    {
    int mySelInL;
    int myName;
    int myClassName;
    FuncPtr myFunc;
    CParamsTypes myTypes;

    TFunc (int className = 0, int funcName = 0, const FuncPtr& func = 0);
    virtual ~TFunc();

    TFunc& addParamType ( SParamTypeInfo t = SParamTypeInfo());

    virtual bool operator== (const TFunc& func) const;
    };

template < class StatFuncPtr>
struct TStaticFunc
    {
    int myName;
    int myClassName;
    StatFuncPtr myFunc;
    CParamsTypes myTypes;

    TStaticFunc ( int className = 0, int funcName = 0, StatFuncPtr func = StatFuncPtr());
    virtual ~TStaticFunc();

    TStaticFunc& addParamType ( SParamTypeInfo t = SParamTypeInfo());
```

```
    virtual bool operator== ( const TStaticFunc& func) const;
    };

//---------------------------------------------------------------------------

template <class TempFunc, class TempStaticFunc>
struct TClass
    {
    int myClassName;
    SClassRef myClassRef;
    CStore<TempFunc> myFuncs;
    CStore<TempStaticFunc> myStaticFuncs;

    TClass (int className = 0);
    virtual ~TClass();

    virtual int getFuncSelector (int funcName);
    virtual int getStaticFuncSelector (int funcName);

    virtual bool operator== (const TClass& cl) const;
    };

//---------------------------------------------------------------------------

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
class TVM : public virtual IVM
    {
    protected:
    CCont<TempClass*> myClasses;

    public:
    TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>();

    CCont<TempClass*> getClasses() { return myClasses; } //bad Style

    virtual SParam staticCall ( int className, int classSel, int funcName, int funcSel, const CParams&
params) = 0;
    virtual SParam call ( CObj* obj, int className, int classSel, int funcName, int funcSel, const
CParams& params) = 0;

    virtual TempFuncPtr rFuncPtr (Func* func);
    virtual TempFuncPtr rFuncPtr (int className, int classSelector, int funcName, int funcSelector);
//uses selectors if non zero otherwise get 'em from names
    virtual TempStaticFuncPtr rStatFuncPtr (StaticFunc* func);
    virtual TempStaticFuncPtr rStatFuncPtr (int className, int classSelector, int funcName, int
funcSelector); //uses selectors if non zero otherwise get 'em from names

    virtual int getClassSelector (int className);
    virtual int getFuncSelector (int className, int funcName);
    virtual int getStaticFuncSelector (int className, int funcName);

    virtual TempClass* getClassPtr ( int className);
    virtual TempClass* getClassPtrByIndex ( int index) { return myClasses[index]; }

    SClassRef registerClass ( int className, int varName);
    SClassRef registerBaseClass ( int className, int baseClassName);

    Func* registerFunc ( SClassRef cl, int funcName, TempFuncPtr func, const CParamsTypes& types);

    StaticFunc* registerStaticFunc ( SClassRef cl, int funcName, TempStaticFuncPtr func, const
CParamsTypes& types);
    };

//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

template <class FuncPtr>
TFunc<FuncPtr>::TFunc ( int className, int funcName, const FuncPtr& func) :
    mySelInL ( 0),
    myClassName ( className),
    myName ( funcName),
    myFunc ( func)
    { }

template <class FuncPtr>
TFunc<FuncPtr>::~TFunc()
    { }

template <class FuncPtr>
TFunc<FuncPtr>& TFunc<FuncPtr>::addParamType ( SParamTypeInfo t = SParamTypeInfo())
    {
    myTypes << t;
    return *this;
    }

template <class FuncPtr>
bool TFunc<FuncPtr>::operator== ( const TFunc<FuncPtr>& func) const
    { return myClassName == func.myClassName && myName == func.myName; }

//---------------------------------------------------------------------------

template <class StatFuncPtr>
TStaticFunc<StatFuncPtr>::TStaticFunc (int className, int funcName, StatFuncPtr func) :
    myClassName (className),
```

```
    myName (funcName),
    myFunc (func)
    { }

template <class StatFuncPtr>
TStaticFunc<StatFuncPtr>::~TStaticFunc()
    { }

template <class StatFuncPtr>
TStaticFunc<StatFuncPtr>& TStaticFunc<StatFuncPtr>::addParamType ( SParamTypeInfo t = SParamTypeInfo())
    {
    myTypes << t;
    return *this;
    }

template <class StatFuncPtr>
bool TStaticFunc<StatFuncPtr>::operator== (const TStaticFunc<StatFuncPtr>& func) const
    { return myClassName == func.myClassName && myName == func.myName; }

//---------------------------------------------------------------------------

template <class TempFunc, class TempStaticFunc>
TClass<TempFunc,TempStaticFunc>::TClass ( int className) :
    myClassName ( className),
    myStaticFuncs (1),
    myFuncs (1)
    { myFuncs.add (TempFunc());
      myStaticFuncs.add (TempStaticFunc()); }

template <class TempFunc, class TempStaticFunc>
TClass<TempFunc,TempStaticFunc>::~TClass()
    { }

template <class TempFunc, class TempStaticFunc>
int TClass<TempFunc,TempStaticFunc>::getFuncSelector (int funcName)
    {
    for (int i = 1; i < myFuncs.rNumber(); i++)
        if (myFuncs[i].myName == funcName && myFuncs[i].myClassName == myClassName/*check*/) return i;
    return 0;
    }

template <class TempFunc, class TempStaticFunc>
int TClass<TempFunc,TempStaticFunc>::getStaticFuncSelector (int funcName)
    {
    for (int i = 1; i < myStaticFuncs.rNumber(); i++)
        if (myStaticFuncs[i].myName == funcName && myStaticFuncs[i].myClassName == myClassName/*check*/)
return i;
    return 0;
    }

template <class TempFunc, class TempStaticFunc>
bool TClass<TempFunc,TempStaticFunc>::operator== (const TClass& cl) const
    { return myClassName == cl.myClassName; }

//---------------------------------------------------------------------------

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::TVM() :
    myClasses (1)
    { myClasses.add ( 0); }

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
TempClass* TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::getClassPtr ( int
className)
    {
    return myClasses [ getClassSelector ( className)];
    }

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
int TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::getClassSelector (int className)
    {
    for (int i = 1; i < myClasses.rNumber(); i++)
        if (myClasses[i]->myClassName == className) return i;
    return 0;
    }

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
int TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::getFuncSelector (int className,
int funcName)
    {
    int classSel = getClassSelector (className);
    if (!classSel) return 0;
    return myClasses[classSel]->getFuncSelector(funcName);
    }

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
int TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::getStaticFuncSelector (int
className, int funcName)
    {
    int classSel = getClassSelector (className);
    if (!classSel) return 0;
    return myClasses[classSel]->getStaticFuncSelector(funcName);
```

```
    }

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
TempFuncPtr TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::rFuncPtr (Func* func)
    { return rFuncPtr (func->myClassName, func->myClassSelectorInVM, func->myName, func->mySelectorInVM);
}

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
TempFuncPtr TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::rFuncPtr (int className,
int classSel, int funcName, int funcSel)
    {
    if (!classSel) classSel = getClassSelector (className);
    if (classSel <= 0 || classSel >= myClasses.rNumber()) return 0; //!!! assert

    if (!funcSel) funcSel = myClasses[classSel]->getFuncSelector (funcName);
    if (funcSel <= 0 || funcSel >= myClasses[classSel]->myFuncs.rNumber()) return 0; //!!! assert

    return myClasses[classSel]->myFuncs[funcSel].myFunc;
    }

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
TempStaticFuncPtr TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::rStatFuncPtr
(StaticFunc* func)
    { return rStatFuncPtr (func->myClassName, func->myClassSelectorInVM, func->myName, func-
>mySelectorInVM); }

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
TempStaticFuncPtr TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::rStatFuncPtr (int
className, int classSel, int funcName, int funcSel)
    {
    if (!classSel) classSel = getClassSelector (className);
    Assert (0 < classSel && classSel < myClasses.rNumber());

    if (!funcSel) funcSel = myClasses[classSel]->getStaticFuncSelector (funcName);
    Assert (0 < funcSel && funcSel < myClasses[classSel]->myStaticFuncs.rNumber());

    return myClasses[classSel]->myStaticFuncs[funcSel].myFunc;
    }

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
SClassRef TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::registerClass ( int
className, int varName)
    {
    if ( getClassSelector ( className))
        return Linker->getClassRef ( className);

    if ( int i = Linker->getClassRef (className))
        {
        myClasses.add ( new TempClass ( className)); //will be deleted automatically
        return SClassRef ( i);
        }

    myClasses.add ( new TempClass (className)); //the same

    return myClasses.back()->myClassRef = Linker->registerClass ( className, varName);
    }

template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
SClassRef TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::registerBaseClass ( int
className, int baseClassName)
    {
    SClass* cl = Linker->getClassPtr ( className);
    SClassRef base  = Linker->getClassRef ( baseClassName);
    Assert ( cl && "can't find class for addin' a base class to it"); //!!! redo
    Assert ( (SClass*)base && "can't find class to add it as a base class"); //!!! redo
    return & cl->registerParent ( base);
    }

template < class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
StaticFunc* TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::registerStaticFunc (
SClassRef cl, int funcName, TempStaticFuncPtr func, const CParamsTypes& types)
    {
    int classSel = getClassSelector (cl->myClassName);
    Assert (0 < classSel && classSel < myClasses.rNumber()); //!!! redo

    for (int i = 1; i < myClasses[classSel]->myStaticFuncs.rNumber(); i++)
        Assert (myClasses[classSel]->myStaticFuncs[i].myName != funcName) //!!! redo

    myClasses[classSel]->myStaticFuncs.add ( TempStaticFunc (cl->myClassName, funcName, func));

    StaticFunc* fPtr =  cl->registerStaticFunc ( StaticFunc ( this, funcName, cl->myClassName, types));

    if (fPtr) return fPtr;

    myClasses[classSel]->myStaticFuncs.del ( myClasses[classSel]->myStaticFuncs.rNumber() - 1);

    return 0;
    }
```

```
template <class TempClass, class TempFunc, class TempFuncPtr, class TempStaticFunc, class
TempStaticFuncPtr>
Func* TVM<TempClass,TempFunc,TempFuncPtr,TempStaticFunc,TempStaticFuncPtr>::registerFunc ( SClassRef cl,
int funcName, TempFuncPtr func, const CParamsTypes& types)
    {
    int sel = getClassSelector (Linker->getClasses()[cl]->myClassName);
    Assert (0 < sel && sel < myClasses.rNumber()); //!!! redo

    for (int i = 1; i < myClasses[sel]->myFuncs.rNumber(); i++)
        Assert (myClasses[sel]->myFuncs[i].myName != funcName); //!!! redo

    myClasses[sel]->myFuncs.add ( TempFunc ( myClasses[sel]->myClassName, funcName, func));

    Func* fPtr = Linker->getClasses()[cl]->registerFunc ( Func ( this, funcName, cl->myClassName, types));

    if (fPtr)
        { myClasses[sel]->myFuncs[ myClasses[sel]->myFuncs.rNumber()-1].mySelInL = cl->getFuncSelector (
funcName);
        return fPtr; }

    myClasses[sel]->myFuncs.del ( myClasses[sel]->myFuncs.rNumber() - 1);

    return 0;
    }

#endif
```

Файл LinkerDataTypes\all.h

```
#if !defined (ALL_FILE_LOADED)
#define ALL_FILE_LOADED

#pragma once

//---------------------------------------------------------------------------------------------------

#include "utilities.h"
//#include "..\Linker\IVM.h"

//---------------------------------------------------------------------------------------------------

const MaxParams = 15;
const MaxOnenamedFuncsNumber = 10;

//---------------------------------------------------------------------------------------------------

#define TYPES CParamsTypes() <<
//#define PARAMS CParams() ,
#define PARAMS CParams() <<
// typedef SParam (CObj::*pObjFunc)(const CParams&);

struct SClassRef;
union UParam;
struct SParam;
struct CParams;
struct SParamTypeInfo;
struct CParamsTypes;
class CObj;
struct SObjFunc;
class CObjRef;
struct SObjFunc;
class CVM;
struct Func;
struct StaticFunc;
struct ImportFunc;
class CVariable;
typedef CVariable CVar;
class CScope;
extern CScope* GlobalScope;
struct SVTable;
struct SClass;

//---------------------------------------------------------------------------------------------------

struct SClassRef
    {
    int mySelectorInLinker;

    SClassRef (SClass* clPtr);
    explicit SClassRef (int selInL = 0);

    operator SClass*();
    operator int() const;
    operator bool() const;
    bool operator!() const;
    SClass* operator->() const;
    bool operator== (const SClassRef& cl) const;
    };

//-------------------------------------------------------------------------
```

```
union UParam
    {
    bool b;
    char c;
    int i;
    long l;
    float f;
    double d;
    const char* str;
    void* pVoid;
    CObj* pObj;

    UParam();
    UParam ( bool);
    UParam ( char);
    UParam ( int);
    UParam ( long);
    UParam ( float);
    UParam ( double);
    UParam ( const char* str);
    UParam ( void*);
    UParam ( CObj* pObj);
    };

struct SParam
    {
    enum Types { eAny = 0, eBool, eChar, eInt, eLong, eFloat, eDouble, eStr, ePVoid, ePObj };

    UParam param;
    Types type;


    SParam();
    SParam ( bool);
    SParam ( char);
    SParam ( int);
    SParam ( long);
    SParam ( float);
    SParam ( double);
    SParam ( const char*);
    SParam ( void*);
    SParam ( CObj*);
    SParam ( CObjRef&);
    ~SParam();

    SParam ( const SParam& par);
    SParam& SParam::operator= ( const SParam& par);

    bool rBool() const;
    char rChar() const;
    int  rInt () const;
    long rLong() const;
    float rFloat() const;
    double rDouble() const;
    const char* rString() const;
    void* rPVoid() const;
    CObjRef rObj() const;

    operator bool() const;
    operator char() const;
    operator int() const;
    operator long() const;
    operator float() const;
    operator double() const;
    operator const char*() const;
    operator void*() const;
    operator CObj*() const;
    operator CObjRef() const;

    SObjFunc operator() ( const CString& fName) const;
    };

struct CParams
    {
    SParam params[MaxParams];
    int num;

    CParams ( bool ok = 1);

    CParams& operator, ( bool);
    CParams& operator, ( char);
    CParams& operator, ( int);
    CParams& operator, ( long);
    CParams& operator, ( float);
    CParams& operator, ( double);
    CParams& operator, ( const char*);
    CParams& operator, ( void*);
    CParams& operator, ( CObjRef& ptr);
    CParams& operator, ( SParam& p);

    CParams& operator<< ( bool);
    CParams& operator<< ( char);
    CParams& operator<< ( int);
    CParams& operator<< ( long);
    CParams& operator<< ( float);
    CParams& operator<< ( double);
    CParams& operator<< ( const char*);
```

```
    CParams& operator<< ( void*);
    CParams& operator<< ( CObjRef& ptr);
    CParams& operator<< ( SParam& p);

    CParams& add ( const SParam& par);

    const SParam& operator[] (int i) const;
    SParam& operator[] (int i);
    };

struct SParamTypeInfo
    {
    SParam::Types myType;
    SClassRef myClassRef; //if type is an object, this member may show its class

    SParamTypeInfo ( SParam::Types t = SParam::eAny, const SClassRef& cl = SClassRef());
    };

struct CParamsTypes
    {
    SParamTypeInfo types[MaxParams];
    int num;

    CParamsTypes();
    CParamsTypes ( const CParams& params);

    CParamsTypes& operator << ( SParamTypeInfo t);
    CParamsTypes& operator , ( SParamTypeInfo t);

    int conformsTo ( const CParamsTypes& params);
    };

//-----------------------------------------------------------------------------------------------

class CObj
    {
    private:
    int myRefsNum;

    protected:
    SClassRef myClass;

    public:
    CObj();
    virtual ~CObj();

    class WrongParameter;

    virtual bool ok() const;

    const SClassRef rClass() const { return myClass; }

    int rRefsNum() const { return myRefsNum; }

    void incRefs();
    void decRefs();

    virtual SParam call (int expression, const CParams& par = CParams());
    virtual SParam call (const CString& expression, const CParams& par = CParams());
    };

struct SObjFunc;

class CObjRef
    {
    CObj* myObj;

    public:
    CObjRef();
    CObjRef ( CObj* obj);
    CObjRef ( const CObjRef& ref);
    CObjRef& operator= ( const CObjRef& ref);
    ~CObjRef();

    CObj* rObj() const { return myObj; }

    SObjFunc operator() ( int fName) const;
    SObjFunc operator() ( const CString& fName) const;
    SObjFunc operator>> ( int fName) const;
    SObjFunc operator>> ( const CString& fName) const;

    CObj* operator-> () const { return myObj; }
    operator bool() const { return myObj? 1 : 0; }

    bool operator== ( const CObjRef& ref) const;
    };

struct SObjFunc
    {
    CObjRef myObj;
    int fName;

    SObjFunc();
    SObjFunc (CObjRef n, int f);

    SParam operator() (const CParams& par = CParams());
    };
```

```
//---------------------------------------------------------------------------

#include "..\Linker\IVM.h"

struct Func
    {
    int myName, mySelectorInVM, myClassName, myClassSelectorInVM;
    CParamsTypes myTypes;
    IVM* myVM;

    Func();
    Func ( IVM* vm, int name, int className, const CParamsTypes& types);

    bool ok() const;

    SParam operator() (CObj* obj, const CParams& params) const;

    bool operator== (const Func& func) const;
    };

//---------------------------------------------------------------------------

struct StaticFunc
    {
    int myName, mySelectorInVM, myClassName, myClassSelectorInVM;
    CParamsTypes myTypes;
    IVM* myVM;

    StaticFunc();
    StaticFunc ( IVM* vm, int name, int className, const CParamsTypes& types);

    bool ok() const;

    SParam operator() (const CParams& params) const;

    bool operator== (const StaticFunc& func) const;
    };

//---------------------------------------------------------------------------------------------------------
/*
struct ImportFunc
    {
    Func* myFunc;

    ImportFunc (int clName, int fName);

    int call (CObj* Obj, ...);
    };
*/
//---------------------------------------------------------------------------------------------------------

class CVariable;
typedef CVariable CVar;

class CVariable
    {
    private:
    CObjRef myObj;
    int myName;

    public:

    CVariable (SParam p, const CString& str);
    CVariable (const CObjRef& Obj, const CString& str);
    CVariable (const CObjRef& Obj, int name = 0);
    CVariable (int name = 0);
    const CVariable& operator= (const CVariable& var);

    SObjFunc operator() ( const CString& fName) const;

    operator CObjRef&();
    const CObjRef& rObj() const;
    CObjRef& gObj();
    int& gName() { return myName; }
    int rName() const;

    bool operator== (const CVariable& var) const;
    bool operator== (CObjRef& obj) const;
    bool operator== (const CString& str) const;
    bool operator== (int str) const;

    CVariable& assign (CVariable& obj);
    CVariable& operator= (CObjRef& Obj);
    CVariable& operator= (const CString& str);
    CVariable& operator= (int str);
    };

//---------------------------------------------------------------------------

class CScope
    {
    protected:
    CCont<CVar> myVars;
    CScope* myParent;

    public:
```

```
    CScope ( CScope* parent = 0, int num = 0);

    CCont<CVar>& getVars() throw() { return myVars; }
    const CCont<CVar>& rVars() const throw() { return myVars; }

    bool check (int name);
    bool check ( const CString& str);

    CVar& gVar (int name);
    CVar& operator() (int name);
    CVar& operator() (const CString& str);

    CVar& operator[] ( int index) throw ( CCont<CVar>::rangeError) { return myVars[index]; }

    CScope& addVar (const CVar& var);
    CScope& delVar (int name);
    CScope& setParent ( CScope* scope);
    CScope* getParent() { return myParent; }
    };

//extern CScope* GlobalScope;

//---------------------------------------------------------------------------

struct SVTable
    {
    CStore<Func> functions;

    SVTable();
    SVTable& operator= (const SVTable& t);

    bool operator== (const SVTable& table) const;
    };

struct SClass : public CObj
    {
    int myClassName;
    SVTable myVTable;
    CStore<StaticFunc> myStaticFuncs;
    CStore<SClassRef> myParents;

    SClass (int clName = 0);
    virtual ~SClass();

    class AmbiguousFunctions;
    class AmbiguousStaticFunctions;

    virtual int getFuncSelector ( int name);
    virtual int getStaticFuncSelector ( int name);
    virtual Func*       findFunc ( int name, const CParams& params = CParams());
    virtual StaticFunc* findStaticFunc ( int name, const CParams& params = CParams());

    virtual SClass&     registerParent ( SClassRef cl);
    virtual Func*       registerFunc ( const Func& func);
    virtual StaticFunc* registerStaticFunc ( const StaticFunc& func);

    virtual bool checkParent ( SClassRef cl);
    virtual bool conformsTo ( SClassRef cl);

    SParam call (int expression, const CParams& par = CParams());

    virtual bool operator== (const SClass& cl) const;
    };

#endif

//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

class CObj::WrongParameter : public CStringData
    {
    public:
    WrongParameter ( const CString& str) :
        CStringData ( str)
        { }
    };

class SClass::AmbiguousFunctions
    {
    SClassRef myClass;
    int myFuncIndex1, myFuncIndex2;

    public:
    AmbiguousFunctions ( SClassRef cl, int funcIndex1, int funcIndex2) :
        myClass ( cl),
        myFuncIndex1 ( funcIndex1),
        myFuncIndex2 ( funcIndex2)
        { }

    SClassRef rClassRef() const { return myClass; }
    int rFuncIndex1() const { return myFuncIndex1; }
    int rFuncIndex2() const { return myFuncIndex2; }
    };

class SClass::AmbiguousStaticFunctions
    {
```

```
    SClassRef myClass;
    int myFuncIndex1, myFuncIndex2;

    public:
    AmbiguousStaticFunctions ( SClassRef cl, int funcIndex1, int funcIndex2) :
        myClass ( cl),
        myFuncIndex1 ( funcIndex1),
        myFuncIndex2 ( funcIndex2)
        { }

    SClassRef rClassRef() const { return myClass; }
    int rFuncIndex1() const { return myFuncIndex1; }
    int rFuncIndex2() const { return myFuncIndex2; }
    };
```

Файл LinkerDataTypes\defs.h

```
#if !defined (DATA_DEFS_FILE_LOADED)
#define DATA_DEFS_FILE_LOADED

#pragma once

#include "stdafx.h"

#pragma warning (disable: 4290)
typedef unsigned int uInt;
extern int TabSize;

#endif //DEFS_FILE_LOADED
```

Файл LinkerDataTypes\small.h

```
#pragma once
#if !defined (SMALL_DATA_TYPES_HEADER_INCLUDED_BY_SE)
#define SMALL_DATA_TYPES_HEADER_INCLUDED_BY_SE

#include "utilities.h" //class CString

class CPosition
    {
    private:
    CString myFileName;
    int myPosition;
    int myStringNumber;
    int mySymbolNumber;

    public:
    CPosition (CString fname, int pos, int strnum, int symbnum);
    CPosition();

    const CString& rFileName() const { return myFileName; }
    CString&     getFileName()       { return myFileName; }
    int            rPos()      const { return myPosition; }
    int&         getPos()            { return myPosition; }
    int            rSymb()     const { return mySymbolNumber; }
    int&         getSymb()           { return mySymbolNumber; }
    int            rStr()      const { return myStringNumber; }
    int&         getStr()            { return myStringNumber; }
    int incSymb (int x = 1) { return mySymbolNumber += x; }
    int incStr  (int x = 1) { return myStringNumber += x; }

    bool operator == ( const CPosition& position);
    };

//---------------------------------------------------------------------------

#endif //SMALL_DATA_TYPES_HEADER_INCLUDED_BY_SE
```

Файл LinkerDataTypes\system.h

```
#if !defined (LINKER_DATA_TYPES_SYSTEM_HEADER_FILE_LOADED)
#define LINKER_DATA_TYPES_SYSTEM_HEADER_FILE_LOADED

#pragma once

#include "defs.h"
#include "utilities.h"
#include "all.h"

#include "stdafx.h"

#include "..\Loader\loader.h"
```

```
struct ILinker;
class CScope;

namespace LinkerDataTypes
{
    extern ILinker* Linker;
    extern CScope* GlobalScope;
    extern ILoader* Loader;
};

void LinkerDataTypesLibInit (ILoader* loader);

//-----------------------------------------------------------------------------------------

#endif
```

Файл LinkerDataTypes\utilities.h

```
#if !defined (DATA_UTILITIES_FILE_LOADED)
#define DATA_UTILITIES_FILE_LOADED

#pragma once

#include <string>
#include <stdio.h>
#include <string.h>
#include "defs.h"
#include "..\ExeptionEngine\exeption.h"

extern int ShowDepth;
#define SHOW(action,depth) { if ( depth < ShowDepth) { action; } }
/*
#define Assert(expression) { if ( !(expression)) throw ( CException (__FILE__, __LINE__, \
                        "Assertion failed: " #expression ";")); }
*/
#define ASSERTPARAM(expression) { if ( !( expression))throw BadParam ( __FILE__, __LINE__, "Bad parameter:
" #expression "."); }

template <class T>
class CInterval
    {
    private:
    T myBegin, myEnd;

    public:
    CInterval (const T& from, const T& to) :
        myBegin (from),
        myEnd (to)
        { }

    const T& from() const { return myBegin; }
    const T& to() const { return myEnd; }
    T& from() { return myBegin; }
    T& to() { return myEnd; }
    bool includes (const T& obj) const
        { return myBegin <= obj && obj <= myEnd; }
    };

struct ExpressionPosition
    {
    const char* myFileName;
    int myLineNumber;

    ExpressionPosition ( const char* fileName = 0, int lineNumber = 0) throw();
    };

class CException
    {
    private:
    ExpressionPosition myPos;
    const char* myComments;

    public:
    CException ( const char* fileName, int line, const char* comments) throw();
    CException ( const char* comments = NULL, const ExpressionPosition& pos = ExpressionPosition())
throw();
    virtual ~CException() throw() { }

    const ExpressionPosition& rExpressionPosition() const throw() { return myPos; }
    const char* rFileName() const throw() { return myPos.myFileName; }
    const char* rComments() const throw() { return myComments; }
    int rLineNumber() const throw() { return myPos.myLineNumber; }

    virtual void printf (FILE* file = stdout) const throw();
    };

#define POSITION ( ExpressionPosition ( __FILE__, __LINE__))

class BadParam : public CException
    {
    public:
    BadParam ( const char* fileName = NULL, int line = -1, const char* expression = NULL) :
```

```
        CException ( fileName, line, expression)
        { }
    };

class CLetters
    {
    private:
    const char* myChars;
    CLetters* myLetters;
    int myLettersNumber;

    public:
    CLetters();
    CLetters (const char* chars);
    CLetters (const char* chars, const CLetters& letters1);
    CLetters (const char* chars, const CLetters& letters1, const CLetters& letters2);
    CLetters (const char* chars, const CLetters& letters1, const CLetters& letters2, const CLetters&
letters3);
    CLetters (const CLetters& letters);
    CLetters& operator= (const CLetters& let);
    ~CLetters();

    const char*&  getChars();
    const char*     rChars() const;
    CLetters*     getLetters();
    const CLetters* rLetters() const;
    int& getLettersNumber();
    int    rLettersNumber() const;

    bool ok() const;

    bool includes (char c);

    CLetters& add (const CLetters& letters);

    protected:
    int incLetters (int n = 1);
    };

inline bool Includes (const char* str, char x);

template <class T1, class T2, class T3>
bool In (T1 from, T2 obj, T3 to)
    { return from <= obj && obj <= to; }

class CString
    {
    char* myString;

    public:
    CString();
    CString (const char* str);
    CString ( std::string str);
//  explicit CString ( char ch);
    CString ( int i) throw ( BadParam); // i == 0 must be true: CString ( NULL) or CString ( 0) but not
CString ( 15).
    ~CString();
    CString (const CString& str);
    CString& operator= (const CString& str);

    operator bool() const throw() { return rLength()? 1 : 0; }
    bool operator ! () const throw() { return rLength()? 0 : 1; }
    operator int() const;
    bool operator== ( const CString& str) const;
    bool operator!= ( const CString& str) const { return ! (*this == str); }

    CString& operator << ( const CString& str);
    CString& operator << ( const char* str) { return (*this) << CString ( str); }

    const char* rString() const;
    const char* c_str() const { return rString(); }
    char*& getString();
    size_t rLength() const;
    };

class CStringData
    {
    protected:
    CString myString;

    public:
    CStringData ( const CString& str) :
        myString ( str)
        { }

    const CString& rString() const { return myString; }
    const char* rcString() const { return myString.rString(); }
    };

template <class T>
class CObjectsContainer
    {
    protected:
    T* myObjects;
    int myNumber;
    int myQuant;
```

```
    public:
    CObjectsContainer();
    CObjectsContainer (int quant);
    CObjectsContainer (const CObjectsContainer& cont);
    CObjectsContainer<T>& operator= (const CObjectsContainer& cont);
    virtual ~CObjectsContainer();

    class rangeError : public CException
        {
        public:
        rangeError ( const char* comments = 0, const ExpressionPosition& pos = ExpressionPosition ())
throw() :
            CException ( comments, pos)
            { }

        void printf ( FILE* fOut = stdout) const throw()
            {
            fprintf ( fOut, "Range error : ");
            CException::printf ( fOut);
            }
        };

    class checkedPointer //definition of this class is here because this ******* compiler don't want to
eat it anywhere else :(
        {
        protected:
        const CObjectsContainer<T>* myContainer;
        int myPos;

        public:
        //typedef CObjectsContainer<T>::rangeError rangeError;

        checkedPointer() :
              myContainer ( 0),
              myPos ( -1)
              { }

        checkedPointer ( const CObjectsContainer<T>* container, int pos = 0) throw ( BadParam) :
            myContainer ( container),
            myPos ( pos)
            { ASSERTPARAM ( container); }

        checkedPointer ( const checkedPointer& p) throw ( BadParam) :
            myContainer ( p.rCont()),
            myPos ( p.rIndex())
            { ASSERTPARAM ( myContainer); }

        bool ok() const throw()
            {
            return myContainer? 1 : 0;
            }

        const CObjectsContainer<T>* rCont() const { return myContainer; }
        int rIndex() const { return myPos; }

        const T* rpObject() const throw ( rangeError)
            {
            return &myContainer->at ( myPos);
            }
        const T* operator-> () const throw ( rangeError) { return rpObject(); }
        operator const T*() const throw ( rangeError) { return rpObject(); }
        const T& rObject() const throw ( rangeError) { return *rpObject(); }
        const T& operator* () const throw ( rangeError) { return rObject(); }

        checkedPointer& operator++ () throw()
            {
            myPos++;
            return *this;
            }

        checkedPointer operator++ (int) throw()
            {
            checkedPointer temp ( *this);
            myPos++;
            return temp;
            }

        checkedPointer& operator-- () throw()
            {
            myPos--;
            return *this;
            }

        checkedPointer operator-- (int) throw()
            {
            checkedPointer temp ( *this);
            myPos--;
            return temp;
            }

        checkedPointer& operator+= ( int pos) throw()
            {
            myPos += pos;
            return *this;
            }

        checkedPointer& operator-= ( int pos) throw()
```

```
            {
            myPos -= pos;
            return *this;
            }

        operator bool() throw() { return myContainer->validIndex ( myPos); }
        bool operator! () const throw() { return !myContainer || !myContainer->validIndex ( myPos); }

        const T& at ( int pos) const throw ( rangeError) //pos may be negative
            {
            return *checkedPointer ( myContainer, myPos + pos);
            }
        const T& operator[] ( int pos) const throw ( rangeError) { return at ( pos); } //pos may be
negative
        };

    int rNumber() const;
    int rQuant() const;
    const T* rObjects() const;
    const T& front() const throw ( rangeError); //throws if myNumber == 0
    const T& back() const throw ( rangeError); //throws if myNumber == 0
    T& front() throw ( rangeError); //throws if myNumber == 0
    T& back() throw ( rangeError); //throws if myNumber == 0

    virtual bool ok() const;

    virtual bool operator== (const CObjectsContainer& cont) const;
    virtual int search (const T& obj);
    bool contains (const T& obj);

    virtual bool validIndex ( int i) const throw();
    virtual T& at ( int pos) throw ( rangeError);
    virtual const T& at ( int pos) const throw ( rangeError);
    virtual T& operator[] (int n) throw ( rangeError);
    virtual const T& operator[] (int n) const throw ( rangeError);

    virtual int add (const T& obj);
    virtual CObjectsContainer& insert ( int pos, const T& obj);
    virtual CObjectsContainer& del (int pos);

    int increase();
    virtual int incTo (int size);
    int incFor (int delta);
    };

//---------------------------------------------------------------------------

#define CCont CObjectsContainer
#define CStore CObjectsStore

//---------------------------------------------------------------------------

template <class T>
class CObjectsStore : public CObjectsContainer<T> //the only difference is increasing of this container by
one objects
    {
    public:
    CObjectsStore();
    CObjectsStore (int quant);
    CObjectsStore (const CObjectsStore& cont);
    CObjectsStore<T>& operator= (const CObjectsStore& cont);
    virtual ~CObjectsStore();

    int increase();
    };

//---------------------------------------------------------------------------

template < class T >
class Stack
    {
    private:
        CCont<T> myObjs;

    public:
        Stack<T> ( int startSize = 100);

        Stack<T>& push ( const T& obj);
        T pop();
        T& top();
        int depth() const { return myObjs.rNumber(); }
    };

//---------------------------------------------------------------------------

template <class T>
CObjectsContainer<T>::CObjectsContainer (const CObjectsContainer& cont) :
    myObjects (NULL),
    myNumber (cont.rNumber()),
    myQuant (cont.rQuant())
    { if (!cont.rObjects()) return;
      myObjects = new T[cont.rQuant()];
      for (int i = 0; i < myNumber; i++)
        myObjects[i] = cont[i]; }

template <class T>
CObjectsContainer<T>& CObjectsContainer<T>::operator= (const CObjectsContainer& cont)
```

```
    { myNumber = cont.rNumber();
      myQuant = cont.rQuant();
      if (!cont.rObjects()) { myObjects = NULL; return *this; }
      myObjects = new T[myQuant];
      for (int i = 0; i < myNumber; i++)
        myObjects[i] = cont[i];
      return *this; }

template <class T>
CObjectsContainer<T>::CObjectsContainer() :
    myObjects (NULL),
    myNumber (0),
    myQuant (0)
    { }

template <class T>
CObjectsContainer<T>::CObjectsContainer (int quant) :
    myObjects (quant? new T [quant] : NULL),
    myNumber (0),
    myQuant (quant)
    { }

template <class T>
CObjectsContainer<T>::~CObjectsContainer()
    { if (myObjects) delete[] myObjects; myObjects = NULL; }

template <class T>
int CObjectsContainer<T>::rNumber() const { return myNumber; }
template <class T>
int CObjectsContainer<T>::rQuant() const { return myQuant; }
template <class T>
const T* CObjectsContainer<T>::rObjects() const { return myObjects; }

template < class T>
const T& CObjectsContainer<T>::front() const throw ( rangeError)
    { Assert ( ok());
    if ( !myNumber)
        throw rangeError();
    return myObjects[0];
    }

template < class T>
const T& CObjectsContainer<T>::back() const throw ( rangeError)
    { Assert ( ok());
    if ( !myNumber)
        throw rangeError();
    return myObjects[myNumber-1];
    }

template < class T>
T& CObjectsContainer<T>::front() throw ( rangeError)
    { Assert ( ok());
    if ( !myNumber)
        throw rangeError();
    return myObjects[0];
    }

template < class T>
T& CObjectsContainer<T>::back() throw ( rangeError)
    { Assert ( ok());
    if ( !myNumber)
        throw rangeError();
    return myObjects[myNumber-1];
    }

template <class T>
bool CObjectsContainer<T>::ok() const
    { return 0 <= myNumber && myNumber <= myQuant && (myQuant? !!myObjects : 1); }

template <class T>
bool CObjectsContainer<T>::operator== (const CObjectsContainer& cont) const
    { Assert (ok());
    if (myNumber != cont.rNumber()) return 0;
    for (int i = 0; i < myNumber; i++)
        if ( !(myObjects[i] == cont[i])) return 0;
    return 1;
    }

template <class T>
int CObjectsContainer<T>::search (const T& obj)
    { Assert (ok());
    for (int i = 0; i < myNumber; i++)
        if (myObjects[i] == obj) return i;
    return -1;
    }
template <class T>
bool CObjectsContainer<T>::contains (const T& obj)
    { return search (obj) >= 0? 1 : 0; }

template < class T>
bool CObjectsContainer<T>::validIndex ( int i) const throw()
    { return 0 <= i && i < myNumber; }

template < class T>
T& CObjectsContainer<T>::at ( int pos) throw ( rangeError)
    { Assert ( ok());
    if ( !validIndex ( pos))
```

```
        throw ( rangeError());
    return myObjects[ pos];
    }
template < class T>
const T& CObjectsContainer<T>::at ( int pos) const throw ( rangeError)
    { Assert ( ok());
    if ( !validIndex ( pos))
        throw ( rangeError());
    return myObjects[ pos];
    }

template <class T>
T& CObjectsContainer<T>::operator[] ( int n) throw ( rangeError)
    { Assert ( ok());
      if ( n < 0 ||  n >= myNumber) throw rangeError ( "Container [n]", POSITION);
      return myObjects[n];  }
template <class T>
const T& CObjectsContainer<T>::operator[] (int n) const throw ( rangeError)
    { Assert ( ok());
      if ( n < 0 ||  n >= myNumber) throw rangeError ( "Container [n]", POSITION);
      return myObjects[n];  }

template <class T>
int CObjectsContainer<T>::add (const T& obj)
    { Assert (ok());
      if (myNumber == myQuant) increase();
      myObjects[myNumber++] = obj;
      return myNumber - 1; }

template <class T>
CObjectsContainer<T>& CObjectsContainer<T>::insert ( int pos, const T& obj)
    {
        Assert (ok());
        if ( pos < 0 || pos > myNumber ) throw rangeError ( "Container.insert", POSITION);

        if (myNumber == myQuant) increase();

        for ( int i = myNumber; i > pos; i--)
              myObjects[i] = myObjects[i-1];
        myNumber++;
        myObjects[pos] = obj;
        return *this;
    }

template <class T>
CObjectsContainer<T>& CObjectsContainer<T>::del (int pos)
    { Assert (ok());
      Assert (0 <= pos && pos < myNumber);
      for (int i = pos; i < myNumber - 1; i++)
        myObjects[i] = myObjects[i+1];
      myNumber--;
      return *this; }

template <class T>
int CObjectsContainer<T>::increase()
    {
    Assert (ok());
    incFor (myQuant/2? myQuant/2 : 1);
    return myQuant;
    }

template <class T>
int CObjectsContainer<T>::incTo (int size)
    {
    Assert (ok());
    ASSERTPARAM (size >= myNumber);
    if ( size <= myQuant) return myQuant;

    T* t = new T [size];
    for (int i = 0; i < myNumber; i++)
        t[i] = myObjects[i];
    if (myObjects) delete[] myObjects;
    myObjects = t; //!!! better bit copy
    return myQuant = size;
    }
template <class T>
int CObjectsContainer<T>::incFor (int delta) { return incTo (myQuant + delta ); }

//---------------------------------------------------------------------------

template <class T>
CObjectsStore<T>::CObjectsStore()
    { }

template <class T>
CObjectsStore<T>::CObjectsStore (int quant) :
    CObjectsContainer<T> (quant)
    { }

template <class T>
CObjectsStore<T>::CObjectsStore (const CObjectsStore& store) :
    CObjectsContainer<T> (store)
    { }

template <class T>
CObjectsStore<T>& CObjectsStore<T>::operator= (const CObjectsStore<T>& store)
    { * ( (CObjectsContainer<T>*) this) = (store); return *this; } //not tested, can't set a breakpoint
```

```
template <class T>
CObjectsStore<T>::~CObjectsStore()
    { }

template <class T>
int CObjectsStore<T>::increase()
    {
    Assert (ok());
    incFor (1);
    return myQuant;
    }

//---------------------------------------------------------------------------

template < class T>
Stack<T>::Stack ( int startSize) :
    myObjs ( startSize)
    { }
template < class T>
Stack<T>& Stack<T>::push ( const T& obj)
    {
        myObjs.add ( obj);
        return *this;
    }
template < class T>
T Stack<T>::pop()
    {
        T res = myObjs.back();
        myObjs.del ( myObjs.rNumber() - 1);
        return res;
    }

template < class T >
T& Stack<T>::top ()
    {
        return myObjs.back();
    }

#endif //UTILITIES_FILE_LOADED
```

Файл Loader\Commands.h

```
// Commands.h - header
// created 05.11.2002 by header creator
#pragma once
#ifndef __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Console\COnsole.h"

void InitConsole (IConsole* con);

#endif //__COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Loader\LoaderCore.h

```
// LoaderCore.h - header file
// created 09.09.2002 by header creator
#pragma once
#ifndef __LOADERCORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __LOADERCORE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "Loader.h" //interface

#ifdef LOADER_EXPORTS
    #define LAPI __declspec (dllexport)
    #else
    #define LAPI __declspec (dllimport)
    #endif

#define START_COMPONENTS 10

typedef void (*DLLGETINSTANCE) (void**);
typedef void (*DLLGETINFO)     (void**);
typedef void (*DLLINIT)        (ILoader*);
typedef void (*DLLONLOAD)      (ILoader*);
typedef void (*DLLTERMINATE)   (void);
//DllOnLoad

//---------------------------------------------------------------------------

struct DllTechInfo
{
    DllInfo* info;

    HMODULE module;
    char name[128]; //short name, reserved
```

```
    IUnknown* iunknown; //for foreign
    bool initialized;
};

//-------------------------------------------------------------------------------------

class LAPI CLoader: public ILoader
{
    public:
    //IUnknown:
        ULONG STDMETHODCALLTYPE AddRef ();
        ULONG STDMETHODCALLTYPE Release    ();
        HRESULT STDMETHODCALLTYPE QueryInterface (REFIID riid, void **ppr);

    //ILoader:
        void  __cdecl Init (bool fullInit);
        void  __cdecl LoadComponent              (char* dllname);
                                  //loades component, rv = hash of it
        void  __cdecl LoadComponent           (HMODULE modulehandle);
        void  __cdecl LoadForeignComponent    (const GUID* clsid, DllInfo alias);
        const DllInfo*  __cdecl GetDllInfo    (int component);
        int  __cdecl GetMaxComponentNum       ();
        IUnknown*  __cdecl GetInterface (const IID& iid);
        IUnknown*  __cdecl GetInterface (const IID& iid, int module);
        void  __cdecl GetInterface (const IID& iid, void**);
        void  __cdecl GetInterface (const IID& iid, int module, void**);
        void  __cdecl InitModules  ();
        void  __cdecl ReInitModules ();
        int   __cdecl SearchComponent  (char* reserved, int reserved2);
        void  __cdecl UnloadComponent  (int reserved);
        void  __cdecl UnloadUnused     ();
        void  __cdecl Terminate        ();
        void  __cdecl Reset            ();

        CLoader     (bool autoload = false);
        ~CLoader    ();

        void Grow      ();
        void GrowFor   (int);
        void GrowTo    (int);

        int SearchBlank ();

    //protected:
        volatile long m_refs;
        bool m_initialized;
        int m_size;
        DllTechInfo* m_data;
        int m_growType; //by 1, by Half, by Whole -> 0, 1, 2
};

#endif //__LOADERCORE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Loader\System.h

```
// System.h - header
// created 05.11.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "LoaderCore.h"
#include "..\Console\Console.h"

extern CLoader* Loader;
extern IConsole* Console;

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Loader\loader.h

```
// loader.h - header file, Defines Loader INTERFACE
// created 09.09.2002 by header creator
#pragma once
#ifndef __LOADER_HEADER_FILE_BY_VASFED_INCLUDED__
#define __LOADER_HEADER_FILE_BY_VASFED_INCLUDED__

#include <objbase.h>
//Public Structures:
//NOT LAST VERSION!

struct DllInfo
{
    char name       [128];
    char dllname    [128];
    char programmers[128];

    bool isForeign;
```

```
    bool hasDemo;
    bool canStartSystem;
    bool isUI;
    bool reserved;
};

//Guids:
// {FAC88A00-069A-4ecf-B7B0-1903AD476D92}
static const GUID IID_ILoader =
{ 0xfac88a00, 0x69a, 0x4ecf, { 0xb7, 0xb0, 0x19, 0x3, 0xad, 0x47, 0x6d, 0x92 } };

struct ILoader: IUnknown
{
    virtual void __cdecl   Init                   (bool fullInit) PURE;
    virtual void __cdecl   LoadComponent          (char* dllname) PURE;
    virtual void __cdecl   LoadComponent          (HMODULE modulehandle) PURE;
    virtual void __cdecl   LoadForeignComponent   (const GUID* clsid, DllInfo alias) PURE; //- reserved
    virtual const DllInfo* __cdecl GetDllInfo     (int component) PURE;
    virtual int   __cdecl  GetMaxComponentNum     (void) PURE;

    virtual void  __cdecl  InitModules (void) PURE;
    virtual void  __cdecl  ReInitModules () PURE;

    virtual IUnknown*  __cdecl GetInterface (const IID& iid) PURE;
    virtual void       __cdecl GetInterface (const IID& iid, void**) PURE;
    virtual IUnknown*  __cdecl GetInterface (const IID& iid, int module) PURE;
    virtual void       __cdecl GetInterface (const IID& iid, int, void**) PURE;

    virtual int        __cdecl SearchComponent (char* reserved, int reserved2) PURE;
    virtual void       __cdecl UnloadComponent (int reserved) PURE;
    virtual void       __cdecl UnloadUnused    (void) PURE;
    virtual void       __cdecl Reset           (void) PURE;
    virtual void       __cdecl Terminate       (void) PURE;
};

#endif //__LOADER_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Loader\resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Loader.rc

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        101
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Файл Loader\resource1.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Loader1.rc

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        101
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Файл LRenederer\LRenderer.h

```
#pragma once
#ifndef __LRENDERER_HEADER_FILE_BY_SE_INCLUDED__
#define __LRENDERER_HEADER_FILE_BY_SE_INCLUDED__

#include <objbase.h>

// {D98F6AAA-504E-4620-8EC9-3D911D96BA50}
static const GUID IID_ILRenderer =
{ 0xd98f6aaa, 0x504e, 0x4620, { 0x8e, 0xc9, 0x3d, 0x91, 0x1d, 0x96, 0xba, 0x50 } };
```

```
interface ILRenderer : public IUnknown
{
};

#endif
```

Файл LRenederer\LRendererCore.h

```
#pragma once
#ifndef __LRENDERERCORE_HEADER_FILE_BY_SE_INCLUDED__
#define __LRENDERERCORE_HEADER_FILE_BY_SE_INCLUDED__

#include "lrenderer.h"
//#include "system.h"
#include "..\LinkerDatatypes\stdafx.h"

#include "..\Common\Utilities\module.h"

class CLinkedRenderer : public ILRenderer, public CObj
{
private:
    static SClassRef sClass;
public:
    DEFINE_IUNKNOWN();

    CLinkedRenderer();

    static bool selfReg();

};

#endif
```

Файл LRenederer\System.h

```
// System.h - header
// created 03.11.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_SE_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_SE_INCLUDED__

//#include "lrenderer.h"
#include "lrendercore.h"
#include "..\CPPVM\CPPVM.h"
#include "..\Linker\Linker.h"
#include "..\SmallVM\SmallVM.h"
#include "..\Renderer\RendInt.h"

namespace LR
    {
        extern CLinkedRenderer* Self;
        extern IRenderer* Renderer;
        extern ICPPVM* CPPVM;
        extern ILinker* Linker;
    };

#endif //__SYSTEM_HEADER_FILE_BY_SE_INCLUDED__
```

Файл Map\Console.h

```
// Console.h - header
// created 22.09.2002 by header creator
#pragma once
#ifndef __CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__

void ConfigureConsole ();

#endif //__CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Map\DataTypes.h

```
// DataTypes.h - header
// created 29.10.2002 by header creator
#pragma once
#ifndef __DATATYPES_HEADER_FILE_BY_VASFED_INCLUDED__
#define __DATATYPES_HEADER_FILE_BY_VASFED_INCLUDED__
```

```
    struct SMapUnit
    {
        SMapUnit    (int code = 0, int datasize = 0, int* data = NULL);
        ~SMapUnit   ();

        int     code;
        int*    data;       //deleted in destructor!
        int     datasize;
    };

    struct SMapFileHeader
    {
        unsigned long signature;   //"MAPF"
        int version1;
        int version2;

        int     sizeX;
        int     sizeY;
        float   sizeS;

        int offset;     //bytes to skip, after header there may be something else
    };

    struct SMapUnitInFile
    {
        int code;
        int datasize; //*4 (ints), values are after structure
    };

#endif //__DATATYPES_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Map\Map.h

```
#ifndef __MAP_HEADER_FILE_BY_VAFSED_INCLUDED__
#define __MAP_HEADER_FILE_BY_VAFSED_INCLUDED__

//this module uses Vasfed's ExeptionEngine

#include <objbase.h>
#include <stdio.h>
#include "..\Utilities\Utilities.h"

// {4D4D758F-790A-4209-8761-742563093631}
static const GUID IID_IAgentsMap =
{ 0x4d4d758f, 0x790a, 0x4209, { 0x87, 0x61, 0x74, 0x25, 0x63, 0x9, 0x36, 0x31 } };

enum MapUnitCode
{
    EMC_Error   = 0x0,
    EMC_Empty   = 0x0,      //белый (RGB (255, 255, 255))
    EMC_Road    = 0x1,    //коричневый  (RGB(150, 100, 0))
    EMC_Field   = 0x2,    //зеленый     (RGB (0, 255, 0))
    EMC_Forest  = 0x3,      //dark green (RGB (0, 128, 0))
    EMC_Rock    = 0x4,    //серый       (RGB (100, 100, 100))
    EMC_Water   = 0x5,    //синий       (RGB (0, 0, 255))
    EMC_House   = 0x6,    //розовый     (RGB(255, 128, 255))
    EMC_Castle  = 0x7,    //красный     (RGB (255, 0, 0))
    EMS_ForceDWORD = 0x8fffffff
};

interface IAgentsMap: public IUnknown
{
    virtual void    Init    (void)          PURE;
    virtual void    Reset   (bool confirm)  PURE;

    virtual int Load    (FILE* file)    PURE;
    virtual int Load    (char* file)    PURE;
    virtual int LoadBmp (FILE* file)    PURE;
    virtual int LoadBmp (char* file)    PURE;
    virtual int Save    (FILE* file)    PURE;
    virtual int SaveBmp (FILE* file)    PURE;
    virtual int Save    (char* file)    PURE;
    virtual int SaveBmp (char* file)    PURE;
    virtual int Ok      (void)          PURE;

    virtual inline int      GetUnitCode     (float x, float y) const PURE;
    virtual inline int*     GetPUnitData    (float x, float y) const PURE;
    virtual inline int      GetUnitData     (float x, float y, int index = 0) const PURE;
    virtual inline int      GetUnitDataSize (float x, float y) const PURE;
    virtual inline void     ModifyUnitData  (float x, float y, int index, int newvalue) PURE;

    virtual inline int      GetUnitCode     (int x, int y) const PURE;
    virtual inline int*     GetPUnitData    (int x, int y) const PURE;
    virtual inline int      GetUnitData     (int x, int y, int index = 0) const PURE;
    virtual inline int      GetUnitDataSize (int x, int y) const PURE;
    virtual inline void     ModifyUnitData  (int x, int y, int index, int newvalue) PURE;

    virtual inline int      GetSizeX        (void)  const PURE;
    virtual inline int      GetSizeY        (void)  const PURE;
    virtual inline float    GetSquareSize   (void)  const PURE;
    virtual inline void     SetSquareSize   (float newsize) PURE;
```

```
};

#endif
```

Файл Map\MapCore.h

```
// MapCore.h - header
// created 11.09.2002 by header creator
#pragma once
#ifndef __MAPCORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __MAPCORE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "Map.h"

#include "..\Utilities\Utilities.h"
#include "..\Common\Utilities\Module.h"
#include "DataTypes.h"

class CAgentsMap: public IAgentsMap
{
    public:
        CAgentsMap  (void);
        ~CAgentsMap (void);

        void    Init    (void);
        void    Reset   (bool confirm);

        int Load    (FILE* file);
        int Load    (char* file);
        int LoadBmp (FILE* file);
        int LoadBmp (char* file);
        int Save    (FILE* file);
        int SaveBmp (FILE* file);
        int Save    (char* file);
        int SaveBmp (char* file);
        int Ok      (void)      ;

        inline int      GetUnitCode     (float x, float y) const;
        inline int*     GetPUnitData    (float x, float y) const;
        inline int      GetUnitData     (float x, float y, int index = 0) const;
        inline int      GetUnitDataSize (float x, float y) const;
        inline void     ModifyUnitData  (float x, float y, int index, int newvalue);

        inline int      GetUnitCode     (int x, int y) const;
        inline int*     GetPUnitData    (int x, int y) const;
        inline int      GetUnitData     (int x, int y, int index = 0) const;
        inline int      GetUnitDataSize (int x, int y) const;
        inline void     ModifyUnitData  (int x, int y, int index, int newvalue);

        inline int      GetSizeX        (void)  const;
        inline int      GetSizeY        (void)  const;
        inline float    GetSquareSize   (void)  const;
        inline void     SetSquareSize   (float newsize);


        DEFINE_IUNKNOWN();

    protected:

        SMapUnit*   m_map;

        int         m_sizex, m_sizey;
        float       m_sizexh, m_sizeyh; //half-size, for optimizing

        float       m_ssize;            // taken from Landscape by Main programm
        int         m_errorflag;
};

#endif //__MAPCORE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Map\System.h

```
// System.h - header
// created 11.09.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Loader\Loader.h"
#include "..\AgentsServer\AgentsServer.h"

extern ILoader*         Loader;
extern IAgentsServer*   Server;

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Memory\memory.h

```
#include <assert.h>
#include <stdlib.h>
#include <windows.h>
#include <stdio.h>

#ifdef MEMORY_EXPORTS
#define MEMORY_API __declspec(dllexport)
#else
#define MEMORY_API __declspec(dllimport)
#endif

#ifndef _MEMORY_H_
#define _MEMORY_H_

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CMEMORYBLOCK ***********************************/
/*****************************************************************************/
/*****************************************************************************/

class MEMORY_API CMemoryBlock
{
protected:
HANDLE         myHandle;
char *         myPointer;
long           mySize;
int            myObjSize, myObjNum, myCurrentObjNum;
int *          myObjMask;
CMemoryBlock * myNext;

friend class CSmartPointer;

/*****************************************************************************/

public:
CMemoryBlock  ();
~CMemoryBlock ();

void   Init          ();
long   RetObjNum     ();
long   RetCurObjNum  ();
int    Allocate      (int objsize, int objnum);
void   Lock          ();
void   Unlock        ();
void   Free          ();
char * RetObject     (int n);
int    AddObject     (void * obj);
void   RemoveObject  (int n);
char * CreateObject  ();
char * CreateObjects (int n);
};

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CSMARTPOINTER **********************************/
/*****************************************************************************/
/*****************************************************************************/

class MEMORY_API CSmartPointer
{
private:
CMemoryBlock * myMemoryBlock;
int            myObjectOffset;

/*****************************************************************************/

public:
CSmartPointer  ();
CSmartPointer  (CMemoryBlock * block, int offset);
~CSmartPointer ();

int             New           (int objsize, int objnum);
void            Delete        ();
int             Grow          (int objsize, int objnum);
int             RetObjNum     ();
int             RetCurObjNum  ();
void            operator =    (CMemoryBlock &block);
void            operator =    (CSmartPointer &ptr);
void            operator +=   (int offset);
void            operator -=   (int offset);
void            operator ++   ();
void            operator --   ();
char *          RetObject     (int n);
int             AddObject     (void * obj);
void            RemoveObject  (int n);
char *          CreateObject  ();
char *          CreateObjects (int n);
void            Lock          ();
void            Unlock        ();
};
```

```
#endif
```

Файл Physics\physics.h

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** PHYSICS.H                                                         ***/
/*** ОБЪЯВЛЕНИЕ КЛАССА CENVIRONMENT И НЕКОТОРЫХ ДРУГИХ ВСПОМОГАТЕЛЬНЫХ***/
/*** КЛАССОВ. ENVIRONMENT ЯВЛЯЕТСЯ МОДУЛЕМ СИСТЕМЫ, ЗАГРУЖАЕМЫМ С ПО- ***/
/*** МОЩЬЮ LOADER'A                                                    ***/
/***                                                                   ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ          ***/
/***                                                                   ***/
/***                                                                   ***/
/*************************************************************************/
/*************************************************************************/

#include "physint.h"

#ifdef PHYSICS_EXPORTS
#define PHYSICS_API __declspec(dllexport)
#else
#define PHYSICS_API __declspec(dllimport)
#endif

#ifndef _PHYSICS_H_
#define _PHYSICS_H_

//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
//\/\/\/\/\/\/\/\/\/\/\/\/\/ CLASS CPHYSTABLE  /\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/

#define MAXACCELERATIONS 20

class PHYSICS_API CPhysTable
{
private:
CMesh **               myMeshes;
CSpeed **              mySpeeds;
CAcceleration ***      myAccelerations;
CIntersectionSphere ** myIntersectionSpheres;
CIntersectionSpace **  myIntersectionSpaces;
CBoundingBox **        myBoundingBoxes;
CTimeDelta **          myDeltas;
CPosition **           myPositions;
CPosition **           myOldPositions;
CPhysMaterial **       myMaterials;
CLandscape *           myLand;
int                    mySize, myCurPos;

/*************************************************************************/

public:
CPhysTable  ();
CPhysTable  (int size);
~CPhysTable ();

void                   Init                 (int size);
void                   NullArray            ();
int                    ok                   ();
int                    ok                   (int n);
int                    RetCurSize           ();
int                    RetMaxSize           ();
int                    SetElement           (CMesh * mesh, CSpeed * speed, CAcceleration * acceleration,
                                             CIntersectionSphere * sphere, CIntersectionSpace * box,
                                             CBoundingBox * box2, CPhysMaterial * mat);
void                   NewAcceleration      (int handle, int n);
void                   SetMesh              (int handle, CMesh * mesh);
void                   SetSpeed             (int handle, CSpeed * speed);
void                   SetAcceleration      (int handle, CAcceleration * acceleration, int n);
void                   SetIntersectionSphere (int handle, CIntersectionSphere * sphere);
void                   SetIntersectionSpace  (int handle, CIntersectionSpace * box);
void                   SetBoundingBox       (int handle, CBoundingBox * box);
void                   SetTimeDelta         (int handle, CTimeDelta * delta);
void                   SetPosition          (int handle, CPosition * pos);
void                   SetOldPosition       (int handle, CPosition * pos);
void                   SetMaterial          (int handle, CPhysMaterial * mat);
void                   SetLandscape         (CLandscape * land);
CMesh *                RetMesh              (int handle);
CSpeed *               RetSpeed             (int handle);
CAcceleration *        RetAcceleration      (int handle, int n);
CIntersectionSphere *  RetIntersectionSphere (int handle);
CIntersectionSpace *   RetIntersectionSpace  (int handle);
CBoundingBox *         RetBoundingBox       (int handle);
CTimeDelta *           RetTimeDelta         (int handle);
CPosition *            RetPosition          (int handle);
CPosition *            RetOldPosition       (int handle);
CPhysMaterial *        RetMaterial          (int handle);
CLandscape *           RetLandscape         ();
```

```
void                RetAccelSum          (int handle, CAcceleration * sum);
void                AddToTime            (int handle, float time);
void                ChangeMovingStep     (int handle);
void                ChangeAcceleration   (int handle, int n, float x, float y, float z);
void                AddToAcceleration    (int handle, int n, float x, float y, float z);
void                ChangeSpeed          (int handle, int n, float x, float y, float z);
void                AddToSpeed           (int handle, int n, float x, float y, float z);
};

//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\// CLASS CENVIRONMENT \\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//

class CEnvironment : public IEnvironment
{
protected:
CPhysTable myTable;
CPoint3D   myGravity;
int        myPhysicLevel;

SDCResult  myLastDCResult;

CTimeDelta myTimePassed;

long       myRefs;

/*************************************************************************/

public:
CEnvironment  ();
CEnvironment  (int tablesize);
~CEnvironment ();

virtual HRESULT             __stdcall QueryInterface        (const IID &iid, void ** obj);
virtual ULONG               __stdcall AddRef                ();
virtual ULONG               __stdcall Release               ();

virtual void                __stdcall SetGravity            (float x, float y, float z);
virtual CPoint3D &          __stdcall RetGravity            ();
virtual void                __stdcall SetPhysicLevel        (int pl);
virtual void                __stdcall InitTable             (int n);
virtual void                __stdcall FillTable             (CMesh * meshes, CSpeed * speeds,
CAcceleration * accels,
                                                             CIntersectionSphere * spheres,
CIntersectionSpace * boxes,
                                                             CBoundingBox * boxes2, CPhysMaterial * mats,
                                                             int num, CLandscape * land);
virtual void                __stdcall FillTable             (CSmartPointer & meshes, CSmartPointer &
speeds, CSmartPointer & accels,
                                                            CSmartPointer & spheres, CSmartPointer &
ispaces,
                                                            CSmartPointer & boxes, CSmartPointer & mats,
                                                            CSmartPointer & land);
virtual void                __stdcall MakeTable             (CMesh * meshes, CSpeed * speeds,
CAcceleration * accels,
                                                             CIntersectionSphere * spheres,
CIntersectionSpace * boxes,
                                                             CBoundingBox * boxes2, CPhysMaterial * mats,
                                                             int num, CLandscape * land);
virtual void                __stdcall MakeTable             (CSmartPointer & meshes, CSmartPointer &
speeds, CSmartPointer & accels,
                                                            CSmartPointer & spheres, CSmartPointer &
ispaces,
                                                            CSmartPointer & boxes, CSmartPointer & mats,
                                                            CSmartPointer & land);
virtual void                __stdcall AddElement            (CMesh * mesh, CSpeed * speed, CAcceleration
* accel,
                                                             CIntersectionSphere * sphere,
CIntersectionSpace * box,
                                                             CBoundingBox * box2, CPhysMaterial * mat);
virtual void *              __stdcall RetPhysTable          ();
virtual void                __stdcall SetMesh               (int handle, CMesh * mesh);
virtual void                __stdcall SetSpeed              (int handle, CSpeed * speed);
virtual void                __stdcall SetAcceleration       (int handle, CAcceleration * acceleration,
int n);
virtual void                __stdcall SetIntersectionSpace  (int handle, CIntersectionSpace * box);
virtual void                __stdcall SetBoundingBox        (int handle, CBoundingBox * box);
virtual void                __stdcall SetTimeDelta          (int handle, CTimeDelta * delta);
virtual void                __stdcall SetPosition           (int handle, CPosition * pos);
virtual void                __stdcall SetLandscape          (CLandscape * land);
virtual CMesh *             __stdcall RetMesh               (int handle);
virtual CSpeed *            __stdcall RetSpeed              (int handle);
virtual CAcceleration *     __stdcall RetAcceleration       (int handle, int n);
virtual CIntersectionSpace * __stdcall RetIntersectionSpace (int handle);
virtual CBoundingBox *      __stdcall RetBoundingBox        (int handle);
virtual CTimeDelta *        __stdcall RetTimeDelta          (int handle);
virtual CPosition *         __stdcall RetPosition           (int handle);
virtual void                __stdcall TransformBox          (int n, CMatrix4x4 &m);
virtual int                 __stdcall SphereCheck           (int n1, int n2);
virtual int                 __stdcall IntersectionCheckX    (CIntersectionSpace * box1,
CIntersectionSpace * box2);
virtual int                 __stdcall IntersectionCheckY    (CIntersectionSpace * box1,
CIntersectionSpace * box2);
virtual int                 __stdcall IntersectionCheckZ    (CIntersectionSpace * box1,
CIntersectionSpace * box2);
```

```
virtual int                  __stdcall IntersectionCheck      (int nobj1, int nobj2, CPoint3D &norm);
virtual int                  __stdcall LineAndQuadCheck       (CPoint3D & line, CPoint3D & dest,
                                                              CPoint3D & quad1, CPoint3D & quad2, CPoint3D
& quad3, CPoint3D & quad4);
virtual int                  __stdcall LineAndBoxCheck        (CPoint3D & line, CPoint3D & dest,
CBoundingBox * bbox);
virtual int                  __stdcall BoundCheck             (int nobj1, int nobj2, CPoint3D &norm);
virtual int                  __stdcall DetectCollisionBetween (int nobj1, int nobj2, CPoint3D &norm);
virtual float                __stdcall RetHorizontHeight      (int hornum, float x, float y, CPoint3D
&norm);
virtual int                  __stdcall DetectLandCollision    (int nobj, CPoint3D &norm);
virtual SDCResult &          __stdcall DetectCollision        (int nobj, CPoint3D &norm);
virtual void                 __stdcall RotateChildCenters     (CMesh * mesh, CMatrix4x4 &m);
virtual void                 __stdcall ReflectSpeed           (int n, CPoint3D &norm);
virtual void                 __stdcall ChangeMovingStep       (int handle);
virtual void                 __stdcall ChangeAcceleration     (int handle, int n, float x, float y, float
z);
virtual void                 __stdcall AddToAcceleration      (int handle, int n, float x, float y, float
z);
virtual void                 __stdcall ChangeSpeed            (int handle, int n, float x, float y, float
z);
virtual void                 __stdcall AddToSpeed             (int handle, int n, float x, float y, float
z);
virtual void                 __stdcall MoveAll                ();
virtual void                 __stdcall Move                   (int n);
virtual void                 __stdcall Rotate                 (int n);
virtual void                 __stdcall ProcessCollision       (int n);
virtual void                 __stdcall ProcessCollision       (int n, CPoint3D &point);
};

extern "C" void PHYSICS_API GetInstance (void ** ptr);

extern IConsole * Console;

#endif _PHYSICS_
```

Файл Physics\physint.h

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** RENDINT.H                                                         ***/
/*** ОПИСАНИЕ ИНТЕРФЕЙСА ФИЗИЧЕСКОГО МОДУЛЯ.                           ***/
/***                                                                   ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ          ***/
/***                                                                   ***/
/***                                                                   ***/
/*************************************************************************/
/*************************************************************************/

#include "..\core3d\core3d.h"
#include "..\physicsbase\physbase.h"
#include "..\landscape\landscap.h"
#include "..\memory\memory.h"
#include "..\console\console.h"
#include "utils.h"
#include <mmsystem.h>
#include <objbase.h>

#ifndef _PHYSINT_H_
#define _PHYSINT_H_

//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\// INTERFACE IENVIRONMENT \\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//

#define PL_NO_MOVE     1
#define PL_MOVE_ONLY   2
#define PL_SPHERE_TEST 3
#define PL_ISPACE_TEST 4
#define PL_BBOX_TEST   5

// {7867625A-A075-4159-B61E-4DF93DAC5B0F}
const IID IID_IEnvironment = {0x7867625a, 0xa075, 0x4159,
                              {0xb6, 0x1e, 0x4d, 0xf9, 0x3d, 0xac, 0x5b, 0x0f}};

interface IEnvironment : IUnknown
{
public:
virtual void                 __stdcall SetGravity             (float x, float y, float z) = 0;
virtual CPoint3D &           __stdcall RetGravity             () = 0;
virtual void                 __stdcall SetPhysicLevel         (int pl) = 0;
virtual void                 __stdcall InitTable              (int n) = 0;
virtual void                 __stdcall FillTable              (CMesh * meshes, CSpeed * speeds,
CAcceleration * accels,
                                                              CIntersectionSphere * spheres,
CIntersectionSpace * boxes,
                                                              CBoundingBox * boxes2, CPhysMaterial * mats,
                                                              int num, CLandscape * land) = 0;
```

```
virtual void                  __stdcall FillTable             (CSmartPointer & meshes, CSmartPointer &
speeds, CSmartPointer & accels,
                                                             CSmartPointer & spheres, CSmartPointer &
ispaces,
                                                             CSmartPointer & boxes, CSmartPointer & mats,
                                                             CSmartPointer & land) = 0;
virtual void                  __stdcall MakeTable             (CMesh * meshes, CSpeed * speeds,
CAcceleration * accels,
                                                              CIntersectionSphere * spheres,
CIntersectionSpace * boxes,
                                                              CBoundingBox * boxes2, CPhysMaterial * mats,
                                                             int num, CLandscape * land) = 0;
virtual void                  __stdcall MakeTable             (CSmartPointer & meshes, CSmartPointer &
speeds, CSmartPointer & accels,
                                                             CSmartPointer & spheres, CSmartPointer &
ispaces,
                                                             CSmartPointer & boxes, CSmartPointer & mats,
                                                             CSmartPointer & land) = 0;
virtual void                  __stdcall AddElement            (CMesh * mesh, CSpeed * speed, CAcceleration
* accel,
                                                              CIntersectionSphere * sphere,
CIntersectionSpace * box,
                                                              CBoundingBox * box2, CPhysMaterial * mat) =
0;
virtual void *                __stdcall RetPhysTable          () = 0;
virtual void                  __stdcall SetMesh               (int handle, CMesh * mesh) = 0;
virtual void                  __stdcall SetSpeed              (int handle, CSpeed * speed) = 0;
virtual void                  __stdcall SetAcceleration       (int handle, CAcceleration * acceleration,
int n) = 0;
virtual void                  __stdcall SetIntersectionSpace  (int handle, CIntersectionSpace * box) = 0;
virtual void                  __stdcall SetBoundingBox        (int handle, CBoundingBox * box) = 0;
virtual void                  __stdcall SetTimeDelta          (int handle, CTimeDelta * delta) = 0;
virtual void                  __stdcall SetPosition           (int handle, CPosition * pos) = 0;
virtual void                  __stdcall SetLandscape          (CLandscape * land) = 0;
virtual CMesh *               __stdcall RetMesh               (int handle) = 0;
virtual CSpeed *              __stdcall RetSpeed              (int handle) = 0;
virtual CAcceleration *       __stdcall RetAcceleration       (int handle, int n) = 0;
virtual CIntersectionSpace *  __stdcall RetIntersectionSpace  (int handle) = 0;
virtual CBoundingBox *        __stdcall RetBoundingBox        (int handle) = 0;
virtual CTimeDelta *          __stdcall RetTimeDelta          (int handle) = 0;
virtual CPosition *           __stdcall RetPosition           (int handle) = 0;
virtual void                  __stdcall TransformBox          (int n, CMatrix4x4 &m) = 0;
virtual int                   __stdcall SphereCheck           (int n1, int n2) = 0;
virtual int                   __stdcall IntersectionCheckX    (CIntersectionSpace * box1,
CIntersectionSpace * box2) = 0;
virtual int                   __stdcall IntersectionCheckY    (CIntersectionSpace * box1,
CIntersectionSpace * box2) = 0;
virtual int                   __stdcall IntersectionCheckZ    (CIntersectionSpace * box1,
CIntersectionSpace * box2) = 0;
virtual int                   __stdcall IntersectionCheck     (int nobj1, int nobj2, CPoint3D &norm) = 0;
virtual int                   __stdcall LineAndQuadCheck      (CPoint3D & line, CPoint3D & dest,
                                                             CPoint3D & quad1, CPoint3D & quad2, CPoint3D
& quad3, CPoint3D & quad4) = 0;
virtual int                   __stdcall LineAndBoxCheck       (CPoint3D & line, CPoint3D & dest,
CBoundingBox * bbox) = 0;
virtual int                   __stdcall BoundCheck            (int nobj1, int nobj2, CPoint3D &norm) = 0;
virtual int                   __stdcall DetectCollisionBetween (int nobj1, int nobj2, CPoint3D &norm) = 0;
virtual float                 __stdcall RetHorizontHeight     (int hornum, float x, float y, CPoint3D
&norm) = 0;
virtual int                   __stdcall DetectLandCollision   (int nobj, CPoint3D &norm) = 0;
virtual SDCResult &           __stdcall DetectCollision       (int nobj, CPoint3D &norm) = 0;
virtual void                  __stdcall RotateChildCenters    (CMesh * mesh, CMatrix4x4 &m) = 0;
virtual void                  __stdcall ReflectSpeed          (int n, CPoint3D &norm) = 0;
virtual void                  __stdcall ChangeMovingStep      (int handle) = 0;
virtual void                  __stdcall ChangeAcceleration    (int handle, int n, float x, float y, float
z) = 0;
virtual void                  __stdcall AddToAcceleration     (int handle, int n, float x, float y, float
z) = 0;
virtual void                  __stdcall ChangeSpeed           (int handle, int n, float x, float y, float
z) = 0;
virtual void                  __stdcall AddToSpeed            (int handle, int n, float x, float y, float
z) = 0;
virtual void                  __stdcall MoveAll               () = 0;
virtual void                  __stdcall Move                  (int n) = 0;
virtual void                  __stdcall Rotate                (int n) = 0;
virtual void                  __stdcall ProcessCollision      (int n) = 0;
virtual void                  __stdcall ProcessCollision      (int n, CPoint3D &point) = 0;
};

#endif
```

Файл Physics\utils.h

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** UTILS.H                                                           ***/
/*** ОБЪЯВЛЕНИЕ ВСПОМОГАТЕЛЬНЫХ КЛАССОВ И СТРУКТУР ДЛЯ ФИЗИЧЕСКОГО      ***/
/*** МОДУЛЯ                                                            ***/
/***                                                                   ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ           ***/
```

```
/***                                                                      ***/
/***                                                                      ***/
/**************************************************************************/
/**************************************************************************/

//\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\
//\\/\\/\\/\\/\\/\\/\\/\\/\\/\\ STRUCT SDCRESUL /\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\
//\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\

#define LANDSCAPE_COLLISIONED -1

struct SDCResult
{
bool myIsCollisionDetected;
int  myObjectNum;
};
```

Файл PhysicsBase\physbase.h

```
/**************************************************************************/
/**************************************************************************/
/***                                                                      ***/
/*** PHYSBASE.H                                                           ***/
/*** ОБЪЯВЛЕНИЕ БАЗОВЫХ ФИЗИЧЕСКИХ КЛАССОВ.  ДИНАМИЧЕСКИ КОМПОНУЕМАЯ  ***/
/*** БИБЛИОТЕКА, ИСПОЛЬЗУЕМАЯ ФИЗИЧЕСКИМ МОДУЛЕМ. ОПИРАЕТСЯ НА БИБЛИ- ***/
/*** ОТЕКУ CORE3D                                                         ***/
/*** СОДЕРЖИТ ОПИСАНИЯ КЛАССОВ:                                           ***/
/***      CTIMER                                                          ***/
/***      CTIMEDELTA                                                      ***/
/***      CPOSITION                                                       ***/
/***      CSPEED                                                          ***/
/***      CACCELERATION                                                   ***/
/***      CINTERSECTIONSPHERE                                             ***/
/***      CINTERSECTIONSPACE                                              ***/
/***      CBOUNDINGBOX                                                    ***/
/***      CPHYSMATERIAL                                                   ***/
/***                                                                      ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ          ***/
/***                                                                      ***/
/***                                                                      ***/
/**************************************************************************/
/**************************************************************************/

#include "..\core3d\core3d.h"
#include <mmsystem.h>

#ifdef PHYSBASE_EXPORTS
#define PHYSBASE_API __declspec(dllexport)
#else
#define PHYSBASE_API __declspec(dllimport)
#endif

#ifndef _PHYSBASE_H_
#define _PHYSBASE_H_

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * CLASS CTIMER  * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

class PHYSBASE_API CTimer
{
protected:
float myTime, myStartTime;

/**************************************************************************/

public:
CTimer  ();
~CTimer ();

void  Init       ();
void  GetTime    ();
float RetTime    ();
void  SetTime    (float time);
void  AddTime    (float time);
void  ReduceTime (float time);
void  operator = (CTimer & timer);
};

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * CLASS CTIMEDELTA  * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

class PHYSBASE_API CTimeDelta : public CTimer
{
```

```
private:
float  myPrevTime;

/*****************************************************************************/

public:
CTimeDelta  ();
~CTimeDelta ();

void  NullTime    ();
void  ZeroTime    ();
void  GetNewTime  ();
float RetPrevTime ();
void  operator =  (CTimeDelta & timer);
};

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CPOSITION * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

class PHYSBASE_API CPosition
{
private:
CPoint3D myStartPosition;

/*****************************************************************************/

public:
CPosition  ();
CPosition  (float x, float y, float z);
CPosition  (float xyz [3]);
CPosition  (CPoint3D &point);
~CPosition ();

void       Init        (float x, float y, float z);
void       Init        (float xyz [3]);
void       Init        (CPoint3D &point);
void       Add         (float x, float y, float z);
void       Add         (float xyz [3]);
void       Add         (CPoint3D &point);
float &    operator () (int n);
CPoint3D & RetPoint    ();
};

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CSPEED  * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

// Класс CSpeed содержит три скорости:
// Скорость в данный момент времени
const int SIMPLE_SPEED = 1;
// Начальная скорость на данном этапе пути
const int START_SPEED = 2;
// скорость вращения объекта
const int CIRCULAR_SPEED = 3;

class PHYSBASE_API CSpeed
{
private:
CPoint3D mySpeed,
         myStartSpeed,
         myCircularSpeed;

/*****************************************************************************/

public:
CSpeed  ();
CSpeed  (float x, float y, float z, float cx, float cy, float cz);
CSpeed  (float xyz [3], float cxyz [3]);
CSpeed  (CPoint3D &point, CPoint3D &cpoint);
~CSpeed ();

void       Init        (int n, float x, float y, float z);
void       Init        (int n, float xyz [3]);
void       Init        (int n, CPoint3D &point);
void       Change      (int n, float x, float y, float z);
void       Change      (int n, float xyz [3]);
void       Change      (int n, CPoint3D &point);
void       Add         (int n, float x, float y, float z);
void       Add         (int n, float xyz [3]);
void       Add         (int n, CPoint3D &point);
void       operator =  (CSpeed & that);
CPoint3D & operator () (int n);
CPoint3D & RetPoint    (int n);
};

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CACCELERATION * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

class PHYSBASE_API CAcceleration
```

```
{
private:
CPoint3D myAcceleration;

/*****************************************************************************/

public:
CAcceleration  ();
CAcceleration  (float x, float y, float z);
CAcceleration  (float xyz [3]);
CAcceleration  (CPoint3D &point);
~CAcceleration ();

void       Init        (float x, float y, float z);
void       Init        (float xyz [3]);
void       Init        (CPoint3D &point);
void       Change      (float x, float y, float z);
void       Change      (float xyz [3]);
void       Change      (CPoint3D &point);
void       Add         (float x, float y, float z);
void       Add         (float xyz [3]);
void       Add         (CPoint3D &point);
void       operator =  (CAcceleration & that);
float &    operator () (int n);
CPoint3D & RetPoint    ();
};

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CINTERSECTIONSPHERE * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

class PHYSBASE_API CIntersectionSphere
{
private:
float myRadius;

/*****************************************************************************/

public:
CIntersectionSphere  ();
CIntersectionSphere  (float radius);
~CIntersectionSphere ();

void    Init      (float radius);
float & RetRadius ();
};

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CINTERSECTIONSPACE  * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

class PHYSBASE_API CIntersectionSpace
{
private:
CPoint3D myPosition;
float    myX, myY, myZ;

/*****************************************************************************/

public:
CIntersectionSpace  ();
CIntersectionSpace  (float x, float y, float z, float x1, float y1, float z1);
CIntersectionSpace  (float pos [3], float xyz [3]);
CIntersectionSpace  (CPoint3D &point, float x, float y, float z);
~CIntersectionSpace ();

void       Init     (float x, float y, float z, float x1, float y1, float z1);
void       Init     (float pos [3], float xyz [3]);
void       Init     (CPoint3D &point, float x, float y, float z);
void       Change   (float x, float y, float z);
void       Change   (float xyz [3]);
void       Change   (CPoint3D &point);
void       Add      (float x, float y, float z);
void       Add      (float xyz [3]);
void       Add      (CPoint3D &point);
CPoint3D & RetPoint ();
float      RetX     ();
float      RetY     ();
float      RetZ     ();
};

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CBOUNDINGBOX  * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

class PHYSBASE_API CBoundingBox
{
private:
CPoint3D myCorner;
CPoint3D myVector [3];
```

```
public:
CBoundingBox  ();
~CBoundingBox ();

void       SetCorner   (float x, float y, float z);
CPoint3D & RetCorner   ();
void       SetVector   (int n, float x, float y, float z);
CPoint3D & RetVector   (int n);
CPoint3D & operator () (int n);
void       AddToCorner (float x, float y, float z);
void       AddToCorner (float xyz [3]);
void       AddToCorner (CPoint3D &point);
};

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * CLASS CPHYSMATERIAL * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

class PHYSBASE_API CPhysMaterial
{
private:
float myMass, myVolume, myImpulceCoeff;

/*************************************************************************/

public:
CPhysMaterial  ();
CPhysMaterial  (float mass, float vol, float imp);
~CPhysMaterial ();

void    Init            (float mass, float vol, float imp);
float & RetMass         ();
float & RetVolume       ();
float & RetImpulceCoeff ();
};

#endif
```

Файл Renderer\extdefs.h

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** EXTDEFS.H                                                         ***/
/*** ОПИСАНИЕ РАЗЛИЧНЫХ КОНСТАНТ И ТИПОВ ДАННЫХ ДЛЯ ПРИМЕНЕНИЯ РАС-    ***/
/*** ШИРЕНИЙ OPENGL.                                                   ***/
/***                                                                   ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ          ***/
/***                                                                   ***/
/***                                                                   ***/
/*************************************************************************/
/*************************************************************************/

#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glaux.h>
#include <mmsystem.h>

// Для использования мультитекстурирования

#define GL_TEXTURE0_ARB                    0x84C0
#define GL_TEXTURE1_ARB                    0x84C1

#define GL_COMBINE_ARB                   0x8570
#define GL_RGB_SCALE_ARB                 0x8573

typedef void (APIENTRY * PFNGLMULTITEXCOORD2FARBPROC) (GLenum target, GLfloat s, GLfloat t);
typedef void (APIENTRY * PFNGLACTIVETEXTUREARBPROC)   (GLenum target);

// Для использования объемного тумана

#define GL_FOG_COORDINATE_SOURCE_EXT       0x8450
#define GL_FOG_COORDINATE_EXT              0x8451
#define GL_FRAGMENT_DEPTH_EXT               0x8452

typedef void (APIENTRY * PFNGLFOGCOORDFEXTPROC)       (GLfloat coord);
```

Файл Renderer\renderer.h

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** RENDERER.H                                                        ***/
/*** ОБЪЯВЛЕНИЕ КЛАССА CRENDERER И НЕКОТОРЫХ ВСПОМОГАТЕЛЬНЫХ КЛАССОВ  ***/
/*** RENDERER ЯВЛЯЕТСЯ ДИНАМИЧЕСКИ ПОДКЛЮЧАЕМОЙ БИБЛИОТЕКОЙ И МОДУЛЕМ ***/
```

```
/*** СИСТЕМЫ, ЗАГРУЖАЕМЫМ С ПОМОЩЬЮ LOADER'A                                ***/
/***                                                                       ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ               ***/
/***                                                                       ***/
/***                                                                       ***/
/*****************************************************************************/
/*****************************************************************************/

#include "rendint.h"
#include "extdefs.h"

#ifdef RENDERER_EXPORTS
#define RENDERER_API __declspec(dllexport)
#else
#define RENDERER_API __declspec(dllimport)
#endif

#ifndef _RENDERER_H_
#define _RENDERER_H_

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CCAMERA ****************************************/
/*****************************************************************************/
/*****************************************************************************/

class CCamera
{
private:
CPoint3D myPosition, myDirection, myUpPoint, mySpeed;
float    myLeftX, myRightX, myLeftY, myRightY, myNearZ, myFarZ;
CPoint3D myUpperLeft, myUpperRight, myBottomLeft, myBottomRight;
bool     myIsRotateFromMouse;

void     CountSpeed ();
void     CountPoints ();

friend class CRenderer;

/*****************************************************************************/

public:
CCamera  ();
CCamera  (float ex, float ey, float ez, float lx, float ly, float lz, float ux, float uy, float uz);
~CCamera ();

void       Set              (float ex, float ey, float ez, float lx, float ly, float lz, float ux, float
uy, float uz);
void       MakeCurrent      ();
void       SetFrustum       (float leftx, float rightx, float lefty, float righty, float nearz, float
farz);
void       MoveAt           (float x, float y, float z);
void       LookAt           (float x, float y, float z);
void       Follow           (CPoint3D &point);
void       Seek             (CPoint3D &point);
void       SetPosition      (float x, float y, float z);
void       Remove           ();
void       Slide            (float k);
void       Rotate           (CPoint3D & point, float a, CPoint3D & axis);
void       RotateUp         (float a);
void       RotateZ          (float a);
int        RotationCheck    (CPoint3D & point);
void       SetViewFromMouse (int centerX, int centerY);
float      RetMinX          ();
float      RetMaxX          ();
float      RetMinY          ();
float      RetMaxY          ();
CPoint3D & RetSpeed         ();
};

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CFONT ******************************************/
/*****************************************************************************/
/*****************************************************************************/

class CFont
{
public:
HFONT myHDCFont;
int   myList;
int   myFirstLetter, myLastLetter;
char  myName [100];
int   myHeight, myWeight;
bool  myItalic, myUnderLined, myStrikeOut;

/*****************************************************************************/

public:
CFont  ();
~CFont ();

void Init (char * name, int size, int startLetter, int lastLetter);
void SetParameters (int weight, bool italic, bool underlined, bool strikeout);
};

/*****************************************************************************/
```

```
/*************************************************************************/
/********************* CLASS CTRANSPFACEBUFFER ***************************/
/*************************************************************************/
/*************************************************************************/

class CTranspFaceBuffer
{
public:
int           myBillBoardsNum, myCurBoardNum;
int           myPolygonsNum, myCurPolyNum;
CBillBoard ** myBoards;
CPoint3D *    myCenters;
CPolygon **   myPolys;

int           myTexNum, myCurTexNum;
int *         myTranspTexList;

/*************************************************************************/

public:
CTranspFaceBuffer  ();
CTranspFaceBuffer  (int bsize, int psize, int texnum);
~CTranspFaceBuffer ();

void         Init           (int bsize, int psize, int texnum);
void         AddTranspTex   (int texid);
bool         IfTransparent  (int texid);
void         Clear          ();
int          RetCurBoardNum ();
int          AddBillBoard   (CBillBoard * board);
CBillBoard * RetBillBoard   (int n);
int          RetCurPolyNum  ();
int          AddPolygon     (CPoint3D & center, CPolygon * poly);
CPoint3D &   RetPolyCenter  (int n);
CPolygon *   RetPolygon     (int n);
};

/*************************************************************************/
/*************************************************************************/
/********************* CLASS CATMOSPHERE *********************************/
/*************************************************************************/
/*************************************************************************/

class CAtmosphere
{
public:
CColor   mySkyColor;
float    myX1, myY1, myX2, myY2, myHeight;
int      myTexID;
CColor   myFogColor;
float    myVolumeFogDepth;
float    myDensity, myStart, myEnd;
bool     myIsVolumeFogEnable;
int      myTexX, myTexY;
float    myStartTexX, myStartTexY;
CPoint3D mySkySpeed;

/*************************************************************************/

CAtmosphere  ();
~CAtmosphere ();

void SetSkyColor  (float r, float g, float b, float a);
void SetSkyPlane  (float x1, float y1, float x2, float y2, float height,
                   float r, float g, float b, float a, int tx, int ty, int texId);
void SetSkySpeed  (float x, float y);

void SetFogColor  (float r, float g, float b, float a);
void SetFogParams (float density, float start, float end);
void SetVolumeFog (bool isvolume, float depth);
};

/*************************************************************************/
/*************************************************************************/
/********************* CLASS CWATER **************************************/
/*************************************************************************/
/*************************************************************************/

class CWater
{
public:
int      mySizeX, mySizeY, myTexCoordX, myTexCoordY;
float    myHeight, myTileSize, myTexPosX, myTexPosY;
CColor   myColor;
int      myGlobalTextureID, myDetailTextureID;
CPoint3D mySpeed;

float    myStartTime, myCurrentTime;

/*************************************************************************/

CWater  ();
~CWater ();

void Init (int sizex, int sizey, float height, float tileSize);
void SetColor (float r, float g, float b, float a);
void InitTexture (int globalTex, int detailTex, int tx, int ty);
```

```
void SetSpeed (float x, float y);
void Move ();
};

/*************************************************************************/
/*************************************************************************/
/********************* CLASS CRENDERER ***********************************/
/*************************************************************************/
/*************************************************************************/

#define MAXLIGHTS    8
#define MAXCAMERAS   20

class CRenderer: public IRenderer
{
private:
HDC               myDC;
HGLRC             myGLRC;
HPALETTE          myPalette;
HWND              myWindow;
WNDCLASS          myWndClass;
char *            myProgName;
char              myWindowName [256];
int               myX1, myY1, myX2, myY2;
bool              myIsFullScreen;
int               myRenderLevel;
int               myRotationWay;
int               myLoDLevel;
bool              myIsMouseCaptured;

int               myFPS, myCurrentFrames;
float             myStartTime, myCurrentTime;

void              (* KeyboardFunc )(HWND hWnd,UINT messg, WPARAM wParam,LPARAM lParam);
void              (* RedrawFunc) ();

CColor            myClearColor;
CAtmosphere       myAtmosphere;
CWater            myWater;

CLightSource *    myGlobalLights [MAXLIGHTS];
int               myCurLight;
int               myMaxTexturesNum;
CCamera           myCameras [MAXCAMERAS];
int               myCameraNum, myCurrentCamera;
CPoint3D          myPreviousCameraPosition;
CFont             myFont;
CTranspFaceBuffer myBuffer;
int               myBufferWriteMode;
float             myDefaultAmbient [4];

CPoint3D          myBufPoint;
CGraphMaterial    myBufMaterial;

long              myRefs;

// Дополнительно загружаемые функции OpenGL
PFNGLMULTITEXCOORD2FARBPROC       glMultiTexCoord2fARB;
PFNGLACTIVETEXTUREARBPROC      glActiveTextureARB;
PFNGLFOGCOORDFEXTPROC          glFogCoordfEXT;

/*************************************************************************/

public:
CRenderer  ();
CRenderer  (int x1, int y1, int x2, int y2);
~CRenderer ();

virtual HRESULT          __stdcall QueryInterface         (const IID &iid, void ** obj);
virtual ULONG            __stdcall AddRef                 ();
virtual ULONG            __stdcall Release                ();

virtual int              __stdcall MakeGLWindow           (HINSTANCE hInst, int nCmdShow, char *
progname, char * wndname);
virtual void             __stdcall ShowGLWindow           ();
virtual void             __stdcall HideGLWindow           ();
virtual int              __stdcall RetGLWindowX1          ();
virtual int              __stdcall RetGLWindowY1          ();
virtual int              __stdcall RetGLWindowX2          ();
virtual int              __stdcall RetGLWindowY2          ();
virtual void             __stdcall SetFullScreen          ();
virtual void             __stdcall SetRenderLevel         (int rl);
virtual LRESULT APIENTRY __stdcall WindowProc             (HWND hWnd,UINT messg, WPARAM wParam,LPARAM
lParam);
friend  LRESULT APIENTRY          CRendererWndProc        (HWND, UINT, WPARAM, LPARAM);
virtual void             __stdcall InitGL                 ();
virtual void             __stdcall LoadExtFuncs           ();
virtual void             __stdcall Resize                 ();
virtual void             __stdcall Resize                 (int x1, int y1, int x2, int y2, bool fullscr);
virtual void             __stdcall SetupPixelFormat       ();
virtual void             __stdcall SetupPalette           ();
virtual void             __stdcall MakeCurrent            ();
virtual void             __stdcall Clear                  ();
virtual void             __stdcall SetClearColor          (float r, float g, float b, float a);
virtual void             __stdcall SetFogParameters       (float density, float start, float end);
virtual void             __stdcall SetVolumeFogDepth      (bool isenabled, float depth);
```

```
virtual void             __stdcall SetSkyPlane            (float x1, float y1, float x2, float y2, float
height,
                                                          float r, float g, float b, float a, int tx, int
ty, int texId);
virtual void             __stdcall SetSkyColor            (float r, float g, float b, float a);
virtual void             __stdcall SetSkySpeed            (float x, float y);
virtual void             __stdcall InitTransparentBuffer  (int bsize, int psize, int texnum);
virtual void             __stdcall SetRotationWay         (int rotationway);
virtual void             __stdcall SetLoDLevel            (int lod);
virtual void             __stdcall Redraw                 ();
virtual int              __stdcall RetFPS                 ();
virtual void             __stdcall CountFPS               ();
virtual void             __stdcall SetKeyboardFunc        (void (* kbf )(HWND hWnd,UINT messg, WPARAM
wParam,LPARAM lParam));
virtual void             __stdcall PrepareMeshes          (CSmartPointer &meshes, CSmartPointer &polys,
                                                          CSmartPointer &meshes2, CSmartPointer &polys2,
CSmartPointer &lights);
virtual void             __stdcall PrepareMeshes          (CSmartPointer &meshes, CSmartPointer &polys,
                                                          CSmartPointer &meshes2, CSmartPointer &polys2,
CSmartPointer &lights, int start, int stop);
virtual void             __stdcall PrepareMeshes          (void * meshes, void * meshes2, int nmeshes,
void * polys, void * polys2, int npolys, void * lights, int nlights);
virtual void             __stdcall PrepareMeshes          (CMesh * meshes, CMesh * meshes2, int nmeshes,
CPolygon * polys, CPolygon * polys2, int npolys, CLightSource * lights, int nlight);
virtual void             __stdcall InitFont               (char * name, int height, int startLetter, int
lastLetter,
                                                          int weight, bool italic, bool underlined, bool
strikeout);
virtual void             __stdcall SetTextPosition        (int x, int y);
virtual void             __stdcall OutText                (int x, int y, float r, float g, float b, float
a, char * text, ...);
virtual void             __stdcall SetRedrawFunc          (void (* redraw) ());
virtual void             __stdcall Render                 ();
virtual int              __stdcall LoadTexture            (CColorMatrix * tex, char * filename);
virtual int              __stdcall LoadTexture            (CColorMatrix * tex, char * filename, char r,
char g, char b);
virtual int              __stdcall LoadTexture            (CColorMatrix * tex, char * filename, char r,
char g, char b, float alpha);
virtual int              __stdcall LoadHorizont           (CLandscape * land, int nhor, char * filename,
float n);
virtual int              __stdcall LoadHorizont           (CHorizont * hor, CPointMatrix * norms, char *
filename, float n);
virtual int              __stdcall LoadGeneralINI         (FILE * file);
virtual int              __stdcall LoadFontINI            (FILE * file);
virtual int              __stdcall LoadAtmosphereINI      (FILE * file);
virtual int              __stdcall LoadWaterINI           (FILE * file);
virtual int              __stdcall LoadCameraINI          (FILE * file);
virtual int              __stdcall LoadLightSourceINI     (FILE * file);
virtual int              __stdcall LoadTextureINI         (FILE * file);
virtual int              __stdcall LoadLandscapeINI       (FILE * file);
virtual int              __stdcall LoadHorizontINI        (FILE * file);
virtual int              __stdcall LoadINI                (char * filename);
virtual int              __stdcall SetCamera              (float ex, float ey, float ez, float lx, float
ly, float lz, float ux, float uy, float uz,
                                                          float x1, float x2, float y1, float y2, float
nz, float fz, bool isrotateformmouse);
virtual void             __stdcall MakeCameraCurrent      (int n);
virtual int              __stdcall RetCurCameraNum        ();
virtual void             __stdcall CameraSlide            (float k);
virtual void             __stdcall CameraRotate           (float up, float z);
virtual void             __stdcall CameraSeek             (CPoint3D &point);
virtual void             __stdcall CameraFollow           (CPoint3D &point);
virtual void             __stdcall CameraMove             (float k);
virtual void             __stdcall CameraSetPosition      (float x, float y, float z);
virtual void             __stdcall CameraSlide            (int n, float k);
virtual void             __stdcall CameraRotate           (int n, float up, float z);
virtual void             __stdcall CameraSeek             (int n, CPoint3D &point);
virtual void             __stdcall CameraFollow           (int n, CPoint3D &point);
virtual void             __stdcall CameraMove             (int n, float k);
virtual void             __stdcall CameraSetPosition      (int n, float x, float y, float z);
virtual void             __stdcall CameraOnLand           (int n, CHorizont * hor);
virtual void             __stdcall SetMaterial            (CGraphMaterial &material);
virtual void             __stdcall SetColorCurrent        (float * color);
virtual void             __stdcall SetColorCurrent        (CColor &color);
virtual void             __stdcall SetColorOfLightCurrent (CColor &color, GLenum nlight);
virtual void             __stdcall SetPosOfLightCurrent   (CColor &point, GLenum nlight);
virtual void             __stdcall SetDirOfLightCurrent   (CPoint3D &point, GLenum nlight);
virtual void             __stdcall SetCutoffOfLightCurrent (float cutoff, GLenum nlight);
virtual void             __stdcall ResetLight             (CLightSource * light);
virtual void             __stdcall SetLight               (CLightSource * light);
virtual void             __stdcall TurnOffLight           (CLightSource * light);
virtual void             __stdcall SetLights              (CLightSource * lights, int nlights);
virtual void             __stdcall SetLights              (CSmartPointer &lights);
virtual void             __stdcall ResetLights            (CLightSource * lights, int nlights);
virtual void             __stdcall ResetLights            (CLightSource ** lights, int nlights);
virtual void             __stdcall ResetLights            ();
virtual void             __stdcall TurnOffLights          (CLightSource * lights, int nlights);
virtual void             __stdcall EnableLighting         ();
virtual void             __stdcall DisableLighting        ();
virtual void             __stdcall TransformMesh          (CMesh * mesh, CMesh * mesh2, CMatrix4x4 &m);
virtual void             __stdcall TransformMeshes        (CMesh * mesh, CMesh * mesh2, int num);
virtual void             __stdcall TransformMeshes        (CSmartPointer &meshes, CSmartPointer
&meshes2);
virtual void             __stdcall RenderSprites          (CSmartPointer &sprites);
virtual void             __stdcall RenderSprites          (void * sprites, int nsprites);
virtual void             __stdcall RenderSprites          (CColorMatrix * sprites, int nsprites);
virtual void             __stdcall RenderSprite           (CColorMatrix * sprite);
```

```
virtual void            __stdcall RenderPictures        (CSmartPointer &pictures);
virtual void            __stdcall RenderPictures        (void * pictures, int n);
virtual void            __stdcall RenderPictures        (CPicture * picturess, int n);
virtual void            __stdcall RenderPicture         (CPicture * picture);
virtual void            __stdcall RenderBillBoards      (CSmartPointer &boards);
virtual void            __stdcall RenderBillBoards      (void * boards, int n);
virtual void            __stdcall RenderBillBoards      (CBillBoard * boards, int n);
virtual void            __stdcall RenderBillBoard       (CBillBoard * billboard);
virtual void            __stdcall RenderSolidBoard      (CPoint3D &ul, CPoint3D &ur, CPoint3D &bl,
CPoint3D &br);
virtual void            __stdcall RenderWireBoard       (CPoint3D &ul, CPoint3D &ur, CPoint3D &bl,
CPoint3D &br);
virtual void            __stdcall RenderLandscape       (CSmartPointer &landscape);
virtual void            __stdcall RenderLandscape       (void * landscape);
virtual void            __stdcall RenderLandscape       (CLandscape &landscape);
virtual void            __stdcall RenderHorizont        (CHorizont &horizont, CPointMatrix &norm);
virtual void            __stdcall RenderSolidHor        (CHorizont &horizont, CPointMatrix &norm);
virtual void            __stdcall RenderSolidLoDAlongX  (CHorizont &horizont, CPointMatrix &norm, int
sizex, int sizey, float sqrsize,

                                                        float rx, float ry, int startx, int starty, int
endx, int endy,

                                                        int nx, int ny, int stripSize, int allSize);
virtual void            __stdcall RenderSolidStepStripX  (CHorizont &horizont, CPointMatrix &norm, int
sizex, int sizey, float sqrsize,

                                                        float rx, float ry, int startx, int starty, int
endy, int nx, int ny);
virtual void            __stdcall RenderSolidLoDAlongY  (CHorizont &horizont, CPointMatrix &norm, int
sizex, int sizey, float sqrsize,

                                                        float rx, float ry, int startx, int starty, int
endx, int endy,

                                                        int nx, int ny, int stripSize, int allSize);
virtual void            __stdcall RenderSolidStepStripY  (CHorizont &horizont, CPointMatrix &norm, int
sizex, int sizey, float sqrsize,

                                                        float rx, float ry, int startx, int starty, int
endx, int nx, int ny);
virtual void            __stdcall RenderSolidLoDGlobal  (CHorizont &horizont, CPointMatrix &norm, int
sizex, int sizey, float sqrsize,

                                                        float rx, float ry, int startx, int starty, int
endx, int endy,

                                                        int nx, int ny, int stripSize, int allSize);
virtual void            __stdcall RenderWireHor         (CHorizont &horizont, CPointMatrix &norm);
virtual void            __stdcall RenderWireLoDAlongX   (CHorizont &horizont, int sizex, int sizey,
float sqrsize, int startx, int starty,

                                                        int endx, int endy, int nx, int ny, int
stripSize, int allSize);
virtual void            __stdcall RenderWireStepStripX  (CHorizont &horizont, int sizex, int sizey,
float sqrsize,

                                                        int startx, int starty, int endy, int nx, int
ny);
virtual void            __stdcall RenderWireLoDAlongY   (CHorizont &horizont, int sizex, int sizey,
float sqrsize, int startx, int starty,

                                                        int endx, int endy, int nx, int ny, int
stripSize, int allSize);
virtual void            __stdcall RenderWireStepStripY  (CHorizont &horizont, int sizex, int sizey,
float sqrsize,

                                                        int startx, int starty, int endx, int nx, int
ny);
virtual void            __stdcall RenderWireLoDGlobal   (CHorizont &horizont, int sizex, int sizey,
float sqrsize, int startx, int starty,

                                                        int endx, int endy, int nx, int ny, int
stripSize, int allSize);
virtual int             __stdcall SetLoDParams          (CHorizont & hor, int sizex, int sizey, float
sqrsize, int & startx, int & starty,

                                                        int & endx, int & endy, int & nx, int & ny, int
& stripSize, int & allSize);
virtual int             __stdcall CountLoDDiv           ();
virtual void            __stdcall ComputeNormals        (CHorizont &horizont, CPointMatrix &norms);
virtual void            __stdcall ComputeNormal         (CHorizont &horizont, CPointMatrix &norms, int
x, int y);
virtual float           __stdcall RetHorizontHeight     (CHorizont &hor, float x, float y);
virtual void            __stdcall RenderMeshes          (CSmartPointer &meshes, CSmartPointer
&meshes2);
virtual void            __stdcall RenderMeshes          (void * meshes, void * meshes2, int nmeshes);
virtual void            __stdcall RenderMeshes          (CMesh * meshes, CMesh * meshes2, int nmeshes);
virtual void            __stdcall RenderMesh            (CMesh * mesh);
virtual void            __stdcall SetMeshLights         (CMesh * mesh);
virtual void            __stdcall RenderPrimitives      (CSmartPointer &prims);
virtual void            __stdcall RenderPrimitives      (void * prims, int nprims);
virtual void            __stdcall RenderPrimitives      (CStandardPrimitive * prims, int nprims);
virtual void            __stdcall RenderPrimitive       (CStandardPrimitive * prim);
virtual void            __stdcall RenderPolys           (CSmartPointer &polys);
virtual void            __stdcall RenderPolys           (void * polys, int npolys);
virtual void            __stdcall RenderPolys           (CPolygon * polys, int pnum);
virtual void            __stdcall RenderPoly            (CPolygon * poly);
virtual void            __stdcall RenderTransparent     ();
virtual void            __stdcall RenderAtmosphere      ();
virtual void            __stdcall RenderWater           ();
virtual void            __stdcall RenderWireWater       ();
virtual void            __stdcall RenderSolidWater      ();
virtual void            __stdcall GetTexID              (CColorMatrix& tex);
virtual void            __stdcall BindTexture           (int id);
virtual void            __stdcall InitTex               (CColorMatrix& tex);
virtual void            __stdcall MakeMipmaps           (CColorMatrix& tex);
virtual void            __stdcall ExtSetMultiTex        (int tex1, int tex2, int scalex, int scaley);
virtual void            __stdcall ExtDisableMultiTex    ();
virtual void            __stdcall ExtMultiTexCoords     (float x, float y);
virtual void            __stdcall ExtSetFogCoord        (float height);
```

```
virtual void            __stdcall RenderTexPoly         (CPolygon * poly);
virtual void            __stdcall RenderSolidPoly       (CPolygon * poly);
virtual void            __stdcall RenderWirePoly        (CPolygon * poly);
virtual void            __stdcall PutVertex             (CPolygon * poly, int n);
};

extern "C" void RENDERER_API GetInstance (void ** ptr);

extern IStorageMaster * STM;
extern IConsole *       Console;

// Функция для обработки сообщений для окна
CRenderer * DllPointerOnRenderer;
void RENDERER_API SetCRendererPointer (CRenderer * renderer);
LRESULT APIENTRY CRendererWndProc (HWND, UINT, WPARAM, LPARAM);

#endif
```

Файл Renderer\rendint.h

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** RENDINT.H                                                         ***/
/*** ОПИСАНИЕ ИНТЕРФЕЙСА ГРАФИЧЕСКОГО МОДУЛЯ.                          ***/
/***                                                                   ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ          ***/
/***                                                                   ***/
/***                                                                   ***/
/*************************************************************************/
/*************************************************************************/

#include "..\memory\memory.h"
#include "..\core3d\core3d.h"
#include "..\landscape\landscap.h"
#include "..\console\console.h"
#include "..\StorageMaster\stockint.h"
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glaux.h>
#include <mmsystem.h>

#ifdef RENDINT_EXPORTS
#define RENDINT_API __declspec(dllexport)
#else
#define RENDINT_API __declspec(dllimport)
#endif

#ifndef _RENDINT_H_
#define _RENDINT_H_

#define RL_WIREFRAME       1
#define RL_SOLID           2
#define RL_LIGHTENED       3
#define RL_LOW_TEXTURED    4
#define RL_HIGH_TEXTURED   5
#define RL_EXTENSIONS_USED 6

#define RW_NO_ROTATING     0
#define RW_OGL_ROTATING    1
#define RW_MATRIX_ROTATING 2

#define DT_LIGHTSOURCE "LightSource"
#define DT_TEXTURE     "Texture"
#define DT_LANDSCAPE   "Landscape"

   //*********************************************************************\\
  //****************** INTERFACE IRENDERER ******************************\\
//*************************************************************************\\*/

// {3A517FD0-458D-43cf-9E9C-E556E0026D7D}
const IID IID_IRenderer = {0x3a517fd0, 0x458d, 0x43cf,
                           {0x9e, 0x9c, 0xe5, 0x56, 0xe0, 0x02, 0x6d, 0x7d}};

interface IRenderer : IUnknown
{
virtual int             __stdcall MakeGLWindow          (HINSTANCE hInst, int nCmdShow, char *
progname, char * wndname) = 0;
virtual void            __stdcall ShowGLWindow          () = 0;
virtual void            __stdcall HideGLWindow          () = 0;
virtual int             __stdcall RetGLWindowX1         () = 0;
virtual int             __stdcall RetGLWindowY1         () = 0;
virtual int             __stdcall RetGLWindowX2         () = 0;
virtual int             __stdcall RetGLWindowY2         () = 0;
virtual void            __stdcall SetFullScreen         () = 0;
virtual void            __stdcall SetRenderLevel        (int rl) = 0;
virtual LRESULT APIENTRY __stdcall WindowProc           (HWND hWnd, UINT messg, WPARAM wParam, LPARAM
lParam) = 0;
virtual void            __stdcall InitGL                () = 0;
virtual void            __stdcall LoadExtFuncs          () = 0;
virtual void            __stdcall Resize                () = 0;
```

```
virtual void            __stdcall Resize                (int x1, int y1, int x2, int y2, bool fullscr)
= 0;
virtual void            __stdcall SetupPixelFormat      () = 0;
virtual void            __stdcall SetupPalette          () = 0;
virtual void            __stdcall MakeCurrent           () = 0;
virtual void            __stdcall Clear                 () = 0;
virtual void            __stdcall SetClearColor         (float r, float g, float b, float a) = 0;
virtual void            __stdcall SetFogParameters      (float density, float start, float end) = 0;
virtual void            __stdcall SetVolumeFogDepth     (bool isenabled, float depth) = 0;
virtual void            __stdcall SetSkyPlane           (float x1, float y1, float x2, float y2, float
height,
                                                        float r, float g, float b, float a, int tx, int

ty, int texId) = 0;
virtual void            __stdcall SetSkyColor           (float r, float g, float b, float a) = 0;
virtual void            __stdcall SetSkySpeed           (float x, float y) = 0;
virtual void            __stdcall InitTransparentBuffer (int bsize, int psize, int texnum) = 0;
virtual void            __stdcall SetRotationWay        (int rotationway) = 0;
virtual void            __stdcall SetLoDLevel           (int lod) = 0;
virtual void            __stdcall Redraw                () = 0;
virtual int             __stdcall RetFPS                () = 0;
virtual void            __stdcall CountFPS              () = 0;
virtual void            __stdcall SetKeyboardFunc       (void (* kbf )(HWND hWnd,UINT messg, WPARAM
wParam,LPARAM lParam)) = 0;
virtual void            __stdcall PrepareMeshes         (CSmartPointer &meshes, CSmartPointer &polys,
                                                        CSmartPointer &meshes2, CSmartPointer &polys2,

CSmartPointer &lights) = 0;
virtual void            __stdcall PrepareMeshes         (CSmartPointer &meshes, CSmartPointer &polys,
                                                        CSmartPointer &meshes2, CSmartPointer &polys2,

CSmartPointer &lights, int start, int stop) = 0;
virtual void            __stdcall PrepareMeshes         (void * meshes, void * meshes2, int nmeshes,
void * polys, void * polys2, int npolys, void * lights, int nlights) = 0;
virtual void            __stdcall PrepareMeshes         (CMesh * meshes, CMesh * meshes2, int nmeshes,
CPolygon * polys, CPolygon * polys2, int npolys, CLightSource * lights, int nlight) = 0;
virtual void            __stdcall InitFont              (char * name, int height, int startLetter, int
lastLetter,
                                                        int weight, bool italic, bool underlined, bool

strikeout) = 0;
virtual void            __stdcall SetTextPosition       (int x, int y) = 0;
virtual void            __stdcall OutText               (int x, int y, float r, float g, float b, float
a, char * text, ...) = 0;
virtual void            __stdcall SetRedrawFunc         (void (* redraw) ()) = 0;
virtual void            __stdcall Render                () = 0;
virtual int             __stdcall LoadTexture           (CColorMatrix * tex, char * filename) = 0;
virtual int             __stdcall LoadTexture           (CColorMatrix * tex, char * filename, char r,
char g, char b) = 0;
virtual int             __stdcall LoadTexture           (CColorMatrix * tex, char * filename, char r,
char g, char b, float alpha) = 0;
virtual int             __stdcall LoadHorizont          (CLandscape * land, int nhor, char * filename,
float n) = 0;
virtual int             __stdcall LoadHorizont          (CHorizont * hor, CPointMatrix * norms, char *
filename, float n) = 0;
virtual int             __stdcall LoadGeneralINI        (FILE * file) = 0;
virtual int             __stdcall LoadFontINI           (FILE * file) = 0;
virtual int             __stdcall LoadAtmosphereINI     (FILE * file) = 0;
virtual int             __stdcall LoadWaterINI          (FILE * file) = 0;
virtual int             __stdcall LoadCameraINI         (FILE * file) = 0;
virtual int             __stdcall LoadLightSourceINI    (FILE * file) = 0;
virtual int             __stdcall LoadTextureINI        (FILE * file) = 0;
virtual int             __stdcall LoadLandscapeINI      (FILE * file) = 0;
virtual int             __stdcall LoadHorizontINI       (FILE * file) = 0;
virtual int             __stdcall LoadINI               (char * filename) = 0;
virtual int             __stdcall SetCamera             (float ex, float ey, float ez, float lx, float
ly, float lz, float ux, float uy, float uz,
                                                        float x1, float x2, float y1, float y2, float

nz, float fz, bool isrotateformmouse) = 0;
virtual void            __stdcall MakeCameraCurrent     (int n) = 0;
virtual int             __stdcall RetCurCameraNum       () = 0;
virtual void            __stdcall CameraSlide           (float k) = 0;
virtual void            __stdcall CameraRotate          (float up, float z) = 0;
virtual void            __stdcall CameraSeek            (CPoint3D &point) = 0;
virtual void            __stdcall CameraFollow          (CPoint3D &point) = 0;
virtual void            __stdcall CameraMove            (float k) = 0;
virtual void            __stdcall CameraSetPosition     (float x, float y, float z) = 0;
virtual void            __stdcall CameraSlide           (int n, float k) = 0;
virtual void            __stdcall CameraRotate          (int n, float up, float z) = 0;
virtual void            __stdcall CameraSeek            (int n, CPoint3D &point) = 0;
virtual void            __stdcall CameraFollow          (int n, CPoint3D &point) = 0;
virtual void            __stdcall CameraMove            (int n, float k) = 0;
virtual void            __stdcall CameraSetPosition     (int n, float x, float y, float z) = 0;
virtual void            __stdcall CameraOnLand          (int n, CHorizont * hor) = 0;
virtual void            __stdcall SetMaterial           (CGraphMaterial &material) = 0;
virtual void            __stdcall SetColorCurrent       (float * color) = 0;
virtual void            __stdcall SetColorCurrent       (CColor &color) = 0;
virtual void            __stdcall SetColorOfLightCurrent  (CColor &color, GLenum nlight) = 0;
virtual void            __stdcall SetPosOfLightCurrent    (CColor &point, GLenum nlight) = 0;
virtual void            __stdcall SetDirOfLightCurrent    (CPoint3D &point, GLenum nlight) = 0;
virtual void            __stdcall SetCutoffOfLightCurrent (float cutoff, GLenum nlight) = 0;
virtual void            __stdcall ResetLight            (CLightSource * light) = 0;
virtual void            __stdcall SetLight              (CLightSource * light) = 0;
virtual void            __stdcall TurnOffLight          (CLightSource * light) = 0;
virtual void            __stdcall SetLights             (CLightSource * lights, int nlights) = 0;
virtual void            __stdcall SetLights             (CSmartPointer &lights) = 0;
virtual void            __stdcall ResetLights           (CLightSource * lights, int nlights) = 0;
virtual void            __stdcall ResetLights           (CLightSource ** lights, int nlights) = 0;
virtual void            __stdcall ResetLights           () = 0;
virtual void            __stdcall TurnOffLights         (CLightSource * lights, int nlights) = 0;
virtual void            __stdcall EnableLighting        () = 0;
```

```
virtual void             __stdcall DisableLighting        () = 0;
virtual void             __stdcall TransformMesh          (CMesh * mesh, CMesh * mesh2, CMatrix4x4 &m) =
0;
virtual void             __stdcall TransformMeshes        (CMesh * mesh, CMesh * mesh2, int num) = 0;
virtual void             __stdcall TransformMeshes        (CSmartPointer &meshes, CSmartPointer &meshes2)
= 0;
virtual void             __stdcall RenderSprites          (CSmartPointer &sprites) = 0;
virtual void             __stdcall RenderSprites          (void * sprites, int nsprites) = 0;
virtual void             __stdcall RenderSprites          (CColorMatrix * sprites, int nsprites) = 0;
virtual void             __stdcall RenderSprite           (CColorMatrix * sprite) = 0;
virtual void             __stdcall RenderPictures         (CSmartPointer &pictures) = 0;
virtual void             __stdcall RenderPictures         (void * pictures, int n) = 0;
virtual void             __stdcall RenderPictures         (CPicture * picturess, int n) = 0;
virtual void             __stdcall RenderPicture          (CPicture * picture) = 0;
virtual void             __stdcall RenderBillBoards       (CSmartPointer &boards) = 0;
virtual void             __stdcall RenderBillBoards       (void * boards, int n) = 0;
virtual void             __stdcall RenderBillBoards       (CBillBoard * boards, int n) = 0;
virtual void             __stdcall RenderBillBoard        (CBillBoard * billboard) = 0;
virtual void             __stdcall RenderSolidBoard       (CPoint3D &ul, CPoint3D &ur, CPoint3D &bl,
CPoint3D &br) = 0;
virtual void             __stdcall RenderWireBoard        (CPoint3D &ul, CPoint3D &ur, CPoint3D &bl,
CPoint3D &br) = 0;
virtual void             __stdcall RenderLandscape        (CSmartPointer &landscape) = 0;
virtual void             __stdcall RenderLandscape        (void * landscape) = 0;
virtual void             __stdcall RenderLandscape        (CLandscape &landscape) = 0;
virtual void             __stdcall RenderHorizont         (CHorizont &horizont, CPointMatrix &norm) = 0;
virtual void             __stdcall RenderSolidHor         (CHorizont &horizont, CPointMatrix &norm) = 0;
virtual void             __stdcall RenderSolidLoDAlongX   (CHorizont &horizont, CPointMatrix &norm, int
sizex, int sizey, float sqrsize,

float rx, float ry, int startx, int starty, int

endx, int endy,

int nx, int ny, int stripSize, int allSize) = 0;
virtual void             __stdcall RenderSolidStepStripX  (CHorizont &horizont, CPointMatrix &norm, int
sizex, int sizey, float sqrsize,

float rx, float ry, int startx, int starty, int

endy, int nx, int ny) = 0;
virtual void             __stdcall RenderSolidLoDAlongY   (CHorizont &horizont, CPointMatrix &norm, int
sizex, int sizey, float sqrsize,

float rx, float ry, int startx, int starty, int

endx, int endy,

int nx, int ny, int stripSize, int allSize) = 0;
virtual void             __stdcall RenderSolidStepStripY  (CHorizont &horizont, CPointMatrix &norm, int
sizex, int sizey, float sqrsize,

float rx, float ry, int startx, int starty, int

endx, int nx, int ny) = 0;
virtual void             __stdcall RenderSolidLoDGlobal   (CHorizont &horizont, CPointMatrix &norm, int
sizex, int sizey, float sqrsize,

float rx, float ry, int startx, int starty, int

endx, int endy,

int nx, int ny, int stripSize, int allSize) = 0;
virtual void             __stdcall RenderWireHor          (CHorizont &horizont, CPointMatrix &norm) = 0;
virtual void             __stdcall RenderWireLoDAlongX    (CHorizont &horizont, int sizex, int sizey,
float sqrsize, int startx, int starty,

int endx, int endy, int nx, int ny, int

stripSize, int allSize) = 0;
virtual void             __stdcall RenderWireStepStripX   (CHorizont &horizont, int sizex, int sizey,
float sqrsize,

int startx, int starty, int endy, int nx, int

ny) = 0;
virtual void             __stdcall RenderWireLoDAlongY    (CHorizont &horizont, int sizex, int sizey,
float sqrsize, int startx, int starty,

int endx, int endy, int nx, int ny, int

stripSize, int allSize) = 0;
virtual void             __stdcall RenderWireStepStripY   (CHorizont &horizont, int sizex, int sizey,
float sqrsize,

int startx, int starty, int endx, int nx, int

ny) = 0;
virtual void             __stdcall RenderWireLoDGlobal    (CHorizont &horizont, int sizex, int sizey,
float sqrsize, int startx, int starty,

int endx, int endy, int nx, int ny, int

stripSize, int allSize) = 0;
virtual int              __stdcall SetLoDParams           (CHorizont & hor, int sizex, int sizey, float
sqrsize, int & startx, int & starty,

int & endx, int & endy, int & nx, int & ny, int

& stripSize, int & allSize) = 0;
virtual int              __stdcall CountLoDDiv            () = 0;
virtual void             __stdcall ComputeNormals         (CHorizont &horizont, CPointMatrix &norms) = 0;
virtual void             __stdcall ComputeNormal          (CHorizont &horizont, CPointMatrix &norms, int
x, int y) = 0;
virtual float            __stdcall RetHorizontHeight      (CHorizont &hor, float x, float y) = 0;
virtual void             __stdcall RenderMeshes           (CSmartPointer &meshes, CSmartPointer &meshes2)
= 0;
virtual void             __stdcall RenderMeshes           (void * meshes, void * meshes2, int nmeshes) =
0;
virtual void             __stdcall RenderMeshes           (CMesh * meshes, CMesh * meshes2, int nmeshes)
= 0;
virtual void             __stdcall RenderMesh             (CMesh * mesh) = 0;
virtual void             __stdcall SetMeshLights          (CMesh * mesh) = 0;
virtual void             __stdcall RenderPrimitives       (CSmartPointer &prims) = 0;
virtual void             __stdcall RenderPrimitives       (void * prims, int nprims) = 0;
virtual void             __stdcall RenderPrimitives       (CStandardPrimitive * prims, int nprims) = 0;
virtual void             __stdcall RenderPrimitive        (CStandardPrimitive * prim) = 0;
virtual void             __stdcall RenderPolys            (CSmartPointer &polys) = 0;
virtual void             __stdcall RenderPolys            (void * polys, int npolys) = 0;
virtual void             __stdcall RenderPolys            (CPolygon * polys, int pnum) = 0;
virtual void             __stdcall RenderPoly             (CPolygon * poly) = 0;
virtual void             __stdcall RenderTransparent      () = 0;
```

```
virtual void            __stdcall RenderAtmosphere      () = 0;
virtual void            __stdcall RenderWater           () = 0;
virtual void            __stdcall RenderWireWater       () = 0;
virtual void            __stdcall RenderSolidWater      () = 0;
virtual void            __stdcall GetTexID              (CColorMatrix& tex) = 0;
virtual void            __stdcall BindTexture           (int id) = 0;
virtual void            __stdcall InitTex               (CColorMatrix& tex) = 0;
virtual void            __stdcall MakeMipmaps           (CColorMatrix& tex) = 0;
virtual void            __stdcall ExtSetMultiTex        (int tex1, int tex2, int scalex, int scaley) =
0;
virtual void            __stdcall ExtDisableMultiTex    () = 0;
virtual void            __stdcall ExtMultiTexCoords     (float x, float y) = 0;
virtual void            __stdcall ExtSetFogCoord        (float height) = 0;
virtual void            __stdcall RenderTexPoly         (CPolygon * poly) = 0;
virtual void            __stdcall RenderSolidPoly       (CPolygon * poly) = 0;
virtual void            __stdcall RenderWirePoly        (CPolygon * poly) = 0;
virtual void            __stdcall PutVertex             (CPolygon * poly, int n) = 0;
};

#endif
```

Файл SmallCompiler\Console.h

```
// Console.h - header
// created 14.10.2002 by header creator
#pragma once
#ifndef __CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__

void AddCommands ();

#endif //__CONSOLE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл SmallCompiler\SmallCompiler.h

```
#if !defined (SMALLCOMPILER_INTERFACE_FILE_LOADED)
#define SMALLCOMPILER_INTERFACE_FILE_LOADED

#pragma once

#include <objbase.h>

// {BC94EBED-D2BD-4747-8D51-A888660E18DB}
static const GUID IID_ISmallCompiler =
{ 0xbc94ebed, 0xd2bd, 0x4747, { 0x8d, 0x51, 0xa8, 0x88, 0x66, 0xe, 0x18, 0xdb } };

interface ISmallCompiler : public IUnknown
{
    bool compileAsDebug;

    virtual bool compileFromFile ( const char* fileName, void* /*ISmallVM**/ smVM = 0) = 0;
    virtual bool executeString ( const char* str) = 0;
};

#endif
```

Файл SmallCompiler\SmallCompilerCore.h

```
// SmallCompilerCore.h - header
// created 15.10.2002 by header creator
#pragma once
#ifndef __SMALLCOMPILERCORE_HEADER_FILE_BY_SE_INCLUDED__
#define __SMALLCOMPILERCORE_HEADER_FILE_BY_SE_INCLUDED__

#include "SmallCompiler.h"
#include "system.h"
#include "..\Common\Utilities\Module.h"
#include "TokenCore.h"
#include "SyntaxCore.h"

class CSmallCompiler : public ISmallCompiler
    {
    private:
    CTokenContainer myTokens;
    CMessageContainer myMessages;

    DEFINE_IUNKNOWN();
    public:
    CSmallCompiler();
    ~CSmallCompiler();

    bool compileFromFile ( const char* fileName, void* smVM = 0);
    BlockPtr compileString ( const char* str);
    bool executeString ( const char* str);
```

```
    };

#endif //__SMALLCOMPILERCORE_HEADER_FILE_BY_SE_INCLUDED__
```

Файл SmallCompiler\SmallSyntax.h

```
// SmallSyntax.h - header
// created 14.10.2002 by header creator
#pragma once
#ifndef __SMALLSYNTAX_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SMALLSYNTAX_HEADER_FILE_BY_VASFED_INCLUDED__

#include <objbase.h>

// {6340C7D6-479E-4b3b-B738-747FC7BFA513}
static const GUID IID_ISmallSyntax =
{ 0x6340c7d6, 0x479e, 0x4b3b, { 0xb7, 0x38, 0x74, 0x7f, 0xc7, 0xbf, 0xa5, 0x13 } };

interface ISmallSyntax: public IUnknown
{

};

#endif //__SMALLSYNTAX_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл SmallCompiler\syntaxcore.h

```
#if !defined (SMALLCOMPILER_SYNTAX_HEADER_LOADED)
#define SMALLCOMPILER_SYNTAX_HEADER_LOADED

#pragma once

#include "System.h"
#include "TokenCore.h"

class CSyntaxAnalyzer
    {
    private:
    typedef CTokenContainer::checkedPointer pointer;

    CSmallVM* myVM;
    CTokenContainer* myTokenCont;
    pointer myPointer;
    CMessageContainer* myMesCont;

    int myLastMethodName;

    public:
    typedef CTokenContainer::rangeError unexpectedEOF;

    CSyntaxAnalyzer ( CTokenContainer* cont = 0, CSmallVM* vm = 0, CMessageContainer* mesCont = 0);

    bool analyze();

    bool eatInterchangeUnit();

    bool ok() const;

    void addMessage ( const CString& str, CMessage::MessageTypes t = CMessage::eSyntaxError);
    void addMessage ( int type, int length, const char* format, ...);
    void addMessage ( const CMessage& mess);

//  private:
    const CPosition* rTokenPos() const throw();

    void gotoNextInterchangeUnit();

    bool eatClassDefinition();

    bool eatMethodDefinition ( SmallClass* pSmCl);
    bool eatClassMethodDefinition ( SmallClass* pSmCl);
     bool eatMethodHeaderTo ( ISmallFunc* pSmFunc);
     bool eatStatementTo ( BlockPtr pBlock);
      bool eatTemplatesTo ( BlockPtr pBlock);
      bool eatExpressionTo ( BlockPtr pBlock);
       bool eatAssignmentTo ( BlockPtr pBlock);
       bool eatBasicExpressionTo ( BlockPtr pBlock);
        bool eatPrimaryTo ( BlockPtr pBlock);
        bool eatMessageTo ( BlockPtr pBlock);
         bool eatUnaryMessagesTo ( BlockPtr pBlock);
         bool eatBinaryMessagesTo ( BlockPtr pBlock);
         bool eatKeywordMessageTo ( BlockPtr pBlock);

    CCommand generateCommandTo ( BlockPtr pBlock, CCommand::action action, const CString& param) throw (
BadParam);
    int saveTopTo ( BlockPtr pBlock); //returns local index of variable
```

```
    void CSyntaxAnalyzer::updateWorkPos ( BlockPtr pBlock, pointer p = pointer()); // adds commands with
debug info if compileToDebug is true

    void eatToken ( CToken::types tokType, const CString& val, const CString& err, CMessage::MessageTypes
mesType = CMessage::eSyntaxError);

    bool checkToken (CString comments = CString());

    bool checkExclamationPoint();

    };

#endif
```

Файл SmallCompiler\system.h

```
#ifndef SMALLCOMPILER_SYSTEM_FILE_LOADED
#define SMALLCOMPILER_SYSTEM_FILE_LOADED

#pragma once

#include "..\LinkerDataTypes\stdafx.h"
#include "..\Loader\Loader.h"
#include "..\Linker\linker.h"
#include "..\SmallVM\SmallVMCore.h"
#include "..\Console\Console.h"
#include "SmallCompiler.h"

namespace SmCompiler
{
    extern ILoader* Loader;
    extern ILinker* Linker;
    extern CSmallVM* SmVM;
    extern ISmallCompiler* Compiler;
    extern IConsole* Console;
    extern CScope* GS;
};

#endif
```

Файл SmallCompiler\tokencore.h

```
#if !defined(TOKEN_FILE_LOADED)
#define TOKEN_FILE_LOADED

#pragma once

#include <stdlib.h>
//#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <stddef.h>
//#include <iostream.h>
#include <vector>

#include "system.h"

//---------------------------------------------------------------------------

class CPosition
    {
    private:
    int myFileName;
    int myPosition;
    int myStringNumber;
    int mySymbolNumber;

    public:
    CPosition (CString fname, int pos, int strnum, int symbnum);
    CPosition();

    const CString& rFileNameStr() const { return SmCompiler::Linker->getName ( myFileName); }
    int            rFileName() const { return myFileName; }
    int&         getFileName()       { return myFileName; }
    int            rPos()      const { return myPosition; }
    int&         getPos()            { return myPosition; }
    int            rSymb()     const { return mySymbolNumber; }
    int&         getSymb()           { return mySymbolNumber; }
    int            rStr()      const { return myStringNumber; }
    int&         getStr()            { return myStringNumber; }
    int incSymb (int x = 1) { return mySymbolNumber += x; }
    int incStr  (int x = 1) { return myStringNumber += x; }

    int setFileName ( const CString& str)
        {
            return myFileName = SmCompiler::Linker->registerName ( str);
        }
```

```
    bool operator == ( const CPosition& position);
    };

typedef CPosition CTokenPosition;

//---------------------------------------------------------------------------

class CMessage
    {
    private:
    int myType;
    char* myText;
    CPosition* myPos;

    public:
    enum MessageTypes { eUnknownError, eLexicalError = 1, eSyntaxError, eUnterminatedString,
eUnexpectedEOF, eSystemError, eInternalWarning };

    CMessage();
    CMessage (int type, const CPosition* pos, const char* text);
    CMessage (const CMessage& m);
    CMessage& operator= (const CMessage& m);
    CMessage (int type, const CPosition* pos, int length, const char* format, ...);
    ~CMessage();

    int              rType() const { return myType; }
    const char*     rText() const { return myText; }
    const CPosition* rPos() const { return myPos; }

    bool operator== (const CMessage&) const { return 0; } //!!! - TODO

    void print (FILE* file = stdout);
    };

//---------------------------------------------------------------------------

class CMessageContainer : public CCont<CMessage>
    {
    public:
    CMessageContainer (int n = 1);

    void print (FILE* file = stdout);
    };

//---------------------------------------------------------------------------

struct BinFilePos
    {
        long mySavedFilePos;
        CTokenPosition mySavedPos;

        bool operator == ( const BinFilePos& ) { return 0; }
    };

//---------------------------------------------------------------------------

class CSourceCode
    {
    private:
    FILE* myFile;

    CTokenPosition myCurrentPos;
    CTokenPosition myLastGivenPos;

    Stack < BinFilePos > mySavedPos;

    public:
    CSourceCode (FILE* File, CString fname = NULL);
//  CSourceCode (const CSourceCode &str);
    ~CSourceCode();

    bool ok() const;

    bool isEndOfFile() const;
    bool isEOF() const;

    char eatSymbol();

    FILE*          rFile()    const { return myFile; } //use carefully
    //FILE*&    getFile()          { return myFile; } it is too dangerous
    int            rFileNum() const { return myCurrentPos.rStr(); }
    int&          getFileNum()       { return myCurrentPos.getStr(); }
    int            rPos()       const { return myCurrentPos.rPos(); }
    int&          getPos()           { return myCurrentPos.getPos(); }
    int            rSymbolNum() const { return myCurrentPos.rSymb(); }
    int&          getSymbolNum()     { return myCurrentPos.getSymb(); }
    CString       rFileNameStr() const { return myCurrentPos.rFileName(); }
    int&          getFileName()      { return myCurrentPos.getFileName(); }
    int           setFileName ( const CString& str)
        {
            return myCurrentPos.setFileName ( str);
        }

    CTokenPosition    rLastGivenPos() const { return myLastGivenPos; }
    CTokenPosition& getLastGivenPos()       { return myLastGivenPos; }
    CTokenPosition    rCurrentPos()   const { return myCurrentPos; }
    CTokenPosition& getCurrentPos()         { return myCurrentPos; }
```

```
    CTokenPosition updateLastGivenPos();

    int skipSpaces();
    int skipComments();

    char getNextSymbol();
    char getNextNonSpaceSymbol();

    char showNextSymbol();
    char showNextNonSpaceSymbol();

    void savePos();
    void loadPos();
    };

//---------------------------------------------------------------------------

struct StringPtr
    {
    int strNum;
    CCont<CString>* contPtr;
    };

union Atom
    {
    int       Integer;
    char      Symbol;
    double    Double;
    StringPtr String;

    Atom ()              : Integer (0)   { }
    Atom (int i)         : Integer (i)   { }
    Atom (char ch)       : Symbol (ch)   { }
    Atom (double doub)   : Double (doub) { }
    Atom (StringPtr str) : String (str)     { }
    };

class CToken
    {
    public:
    enum types { eUnknown, eIdentifier, eKeyword, eBinarySelector, eReturnOperator, eAssignmentOperator,
eDot, eDotComa,
                 eOpenBracket, eCloseBracket, eOpenBlockBracket, eCloseBlockBracket, eInteger, eFloat,
eQuotedCharacter,
                 eQuotedString, eQuotedSelector, eHashedString, eKeywordSelector, eDoubleDot }; //to be
continued

    private:
    types myType;
    CTokenPosition myPos;
    Atom myData;

    public:

    CToken (types type, const CTokenPosition &pos, const Atom& data);
    CToken();

    bool operator== (const CToken& token) const;

    void print() const;

    types&  getType()          { return myType; }
    char&   getSymbol()       { return myData.Symbol; }
    int&    getInteger()      { return myData.Integer; }
    double& getDouble()       { return myData.Double; }
    StringPtr&  getStringPtr() { return myData.String; }
    StringPtr             rStringPtr() const { return myData.String; }
    char                 rSymbol()    const { return myData.Symbol; }
    const CTokenPosition* rPos()       const { return &myPos; }
    int                  rInteger()   const { return myData.Integer; }
    double                rDouble()    const { return myData.Double; }
    types                 rType()      const { return myType; }

    operator types() const throw() { return rType(); }
    operator CString() const throw ( CCont<CToken>::rangeError) { return rString(); }

    CString rString() const throw ( CCont<CToken>::rangeError)
        {
        if ( myType != eIdentifier && myType != eKeyword && myType != eBinarySelector && myType !=
eQuotedString &&
             myType != eQuotedSelector && myType != eHashedString && myType != eKeywordSelector)
                return 0;
        CCont<CString>* cont = myData.String.contPtr;
        if ( ! cont) return 0;
        int p = myData.String.strNum;
        return (*cont)[p];
        }
    };

//---------------------------------------------------------------------------

class CTokenContainer : public CCont<CToken>
    {
    public:
//  typedef CCont<CToken>::checkedPointer checkedPointer;
```

```
    CCont<CString> myTexts;

    CTokenContainer (int lecsQuant = 1);

    bool ok() const;

    void print ( int n = -1) const;
    };

//-------------------------------------------------------------------------

class CTokenAnalyzer
    {
    private:
    CSourceCode mySrc;
    CTokenContainer* myTokens;
    CMessageContainer* myMessages;
    long myTempFilePos;
    CTokenPosition myTempTokenPos;

    public:
    static CLetters UppercaseEnglishAlphabetic;
    static CLetters LowercaseEnglishAlphabetic;
    static CLetters UppercaseRussianAlphabetic;
    static CLetters LowercaseRussianAlphabetic;
    static CLetters NonCaseLetter;
    static CLetters EnglishAlphabetic;
    static CLetters RussianAlphabetic;
    static CLetters UppercaseAlphabetic;
    static CLetters LowercaseAlphabetic;
    static CLetters Letter;
    static CLetters Number;
    static CLetters IdentifierLetter;
    static CLetters BinaryLetter;
    static CLetters WhiteSpace;

    CTokenAnalyzer (CTokenContainer* containerPtr, const CSourceCode& srcStr, CMessageContainer* mess);
    CTokenAnalyzer (CTokenContainer* containerPtr, FILE* file, CString fname, CMessageContainer* mess);

    bool ok();

    bool analyze();

    bool genToken();

    CTokenAnalyzer& savePos();
    CTokenAnalyzer& loadPos();

    bool getQuotedString ( bool hashed = 0);
    bool getNumber();
    bool getIdentifier();
    bool getBinarySelector ( bool quoted = 0);
    bool getQuotedSelector();

    void addMessage ( const CMessage& mess);
    int addToken ( CToken::types type, const Atom &data);
    };

std::pair < std::vector<int>, bool> VariableList ( const CToken& tok); //tok.rType = eQuotedString. "v1 v2
myX myR"

/*
identifier ::= letter (letter | digit)*
keyword ::= identifier ':'
binarySelector ::= binaryCharacter+
returnOperator ::= '^'
assignmentOperator ::=  ':='

integer ::= decimalInteger  | radixInteger
-decimalInteger ::= digits
    digits ::= digit+
-radixInteger ::= radixSpecifier  'r' radixDigits
    radixSpecifier := digits
    radixDigits ::= (digit | uppercaseAlphabetic)+

float ::= mantissa [exponentLetter exponent]
    mantissa ::= digits '.' digits
    exponent ::= ['-']decimalInteger
    exponentLetter ::= 'e' | 'd' | 'q'

-scaledDecimal ::= scaledMantissa 's' [fractionalDigits]
    scaledMantissa ::= decimalInteger | mantissa
    fractionalDigits ::= decimalInteger

quotedCharacter ::= '$' character
quotedString ::= stringDelimiter stringBody stringDelimiter
    stringBody ::= (nonStringDelimiter | (stringDelimiter stringDelimiter)*)
    stringDelimiter ::= '''     "a single quote"
    nonStringDelimiter ::= "any character except stringDelimiter"

hashedString ::= '#' quotedString
quotedSelector ::= '#' (unarySelector  |  binarySelector  | keywordSelector)
    keywordSelector ::= keyword+
-<array literal> ::= '#(' <array element>* ')'
    <array element> ::= <literal> | identifier*/
#endif
```

Файл SmallDebugger\DebuggerCore.h

```
#pragma once
#if !defined (DEBUGGERCORE_HEADER_INCLUDED_BY_SE)
#define DEBUGGERCORE_HEADER_INCLUDED_BY_SE

#include "Debugger.h"

class Breakpoint
    {
    private:
        BreakpointPosition myPosition;

    public:
        Breakpoint();
        Breakpoint ( const BreakpointPosition& position);
        operator const BreakpointPosition() { return myPosition; }

        const BreakpointPosition& rPosition() const { return myPosition; }

        bool covers ( const BreakpointPosition& position);
        bool covers ( const SymbolPosition& position);

        bool operator == ( const Breakpoint& breakpoint);
    };

class CSmallDebugger : public ISmallDebugger
    {
    private:
        volatile bool myPause;
        volatile int myCurBreakpointIndex;
        CCont<Breakpoint> myBreakpoints;
        SymbolPosition myLastPosition;

    public:
        CSmallDebugger();

        DEFINE_IUNKNOWN();

        void updateWorkPosition ( SymbolPosition position);

        SymbolPosition getLastPosition() { return myLastPosition; }

        int setBreakpoint ( BreakpointPosition pos);
        int getBreakpointsNumber();
        BreakpointPosition getBreakpoint ( int index);
        void delBreakpoint ( int index);
        int getCurBreakpointIndex();

        int checkBreakpoint ( SymbolPosition position);

        void pauseOn();
        void pauseOff();
    };

#endif //DEBUGGERCORE_HEADER_INCLUDED_BY_SE
```

Файл SmallDebugger\system.h

```
#pragma once
#ifndef SMALLDEBUGGER_SYSTEM_HEADER_LOADED_BY_SE
#define SMALLDEBUGGER_SYSTEM_HEADER_LOADED_BY_SE

#include "..\LinkerDataTypes\stdafx.h"
#include "..\Loader\Loader.h"
#include "..\Linker\linker.h"
#include "..\SmallVM\SmallVM.h"
#include "..\Console\Console.h"
#include "..\SmallCompiler\SmallCompiler.h"
#include "Debugger.h"
#include "..\SmalltalkEditor\Editor.h"

namespace SmDebugger
{
    extern ILoader* Loader;
    extern ILinker* Linker;
    extern ISmallVM* SmVM;
    extern ISmallCompiler* Compiler;
    extern IConsole* Console;
    extern ISmallDebugger* Debugger;
    extern IEditor* Editor;
    extern CScope* GS;
};

#endif
```

Файл SmallDebugger\Debugger.h

```
#pragma once
#if !defined (DEBUGGER_HEADER_INCLUDED_BY_SE)
#define DEBUGGER_HEADER_INCLUDED_BY_SE

#include "..\LinkerDataTypes\stdafx.h"
#include <objbase.h>

struct SymbolPosition
    {
        int myFileName;
        int myPosition;
    };

struct BreakpointPosition
    {
        int myFileName;
        int myFirstPos;
        int myLastPos;
    };

// {53985F79-C510-48fd-9629-8FA8E2724733}
static const GUID IID_ISmallDebugger =
{ 0x53985f79, 0xc510, 0x48fd, { 0x96, 0x29, 0x8f, 0xa8, 0xe2, 0x72, 0x47, 0x33 } };

interface ISmallDebugger : public IUnknown
    {
        enum eCommands { eRun, eStop, ePause, eStep, eStepIn, eStepOut };
        eCommands myCommand;

        virtual void updateWorkPosition ( SymbolPosition pos) = 0;
        virtual SymbolPosition getLastPosition() = 0;

        virtual int setBreakpoint ( BreakpointPosition pos) = 0;
        virtual int getBreakpointsNumber() = 0;
        virtual BreakpointPosition getBreakpoint ( int index) = 0;
        virtual void delBreakpoint ( int index) = 0;
        virtual int getCurBreakpointIndex() = 0;

        virtual void pauseOn() = 0;
        virtual void pauseOff() = 0;
    };

#endif //DEBUGGER_HEADER_INCLUDED_BY_SE
```

Файл SmalltalkEditor\Commands.h

```
// Commands.h - header
// created 29.11.2002 by header creator
#pragma once
#ifndef __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Console\Console.h"

void AddCommands (IConsole* console);

#endif //__COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл SmalltalkEditor\Editor.h

```
// Editor.h - header
// created 29.11.2002 by header creator
#pragma once
#ifndef __EDITOR_HEADER_FILE_BY_VASFED_INCLUDED__
#define __EDITOR_HEADER_FILE_BY_VASFED_INCLUDED__

#include <objbase.h>

#include "..\SmallDebugger\debugger.h" //SymbolPosition, BreakpointPosition
#include "ThreadGate.h"

// {CAD38C27-DFF9-4c2a-B7E1-9D7E49FBB50F}
static const GUID IID_IEditor =
{ 0xcad38c27, 0xdff9, 0x4c2a, { 0xb7, 0xe1, 0x9d, 0x7e, 0x49, 0xfb, 0xb5, 0xf } };

interface IEditor: public IUnknown
{
    virtual void setCursorTo          ( SymbolPosition position) = 0;
    virtual void insertTextTo         ( const char* text, SymbolPosition position) = 0;
    virtual void output               ( const char* text) = 0;

    virtual void systemStarted        ( void) = 0; //не вызывается ещё
```

```
    virtual void stoppedAt            ( int fileName, int position, int breakpointIndex) = 0;
    virtual void systemStopped        ( void) = 0; //не вызывается ещё
    virtual void breakpointIsSet      ( int breakpointIndex) = 0; //не вызывается ещё
    virtual void breakpointIsRemoved  ( const BreakpointPosition& breakpoint) = 0; //не вызывается ещё

    virtual void Init              (void) PURE;
    virtual void Terminate         (void) PURE;
    virtual void ShowEditorWindow  (void) PURE;

    virtual IDelayedExecutor* __cdecl GetThreadGate (void) PURE;
};

#endif //__EDITOR_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл SmalltalkEditor\EditorCore.h

```
// EditorCore.h - header
// created 29.11.2002 by header creator
#pragma once
#ifndef __EDITORCORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __EDITORCORE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "Editor.h"
#include "..\Common\Utilities\Module.h"
#define __INTERFACE
#include ".\Builder\EditorInt.h"

#include "ThreadGateCore.h"
#include "editor.h"
#include "system.h"
#include <memory>

class Breakpoint
    {
    private:
        BreakpointPosition myPosition;

    public:
        Breakpoint();
        Breakpoint ( const BreakpointPosition& position);
        operator const BreakpointPosition() { return myPosition; }

        const BreakpointPosition& rPosition() const { return myPosition; }

        bool covers ( const BreakpointPosition& position);
        bool covers ( const SymbolPosition& position);

        bool operator == ( const Breakpoint& breakpoint);
    };

struct APDebugTree
    {
        std::auto_ptr < IDebugObjectsTree > tree;
        int myDepth;

        APDebugTree() : myDepth ( 0) { }
        APDebugTree ( std::auto_ptr < IDebugObjectsTree > apTree, int depth);

        APDebugTree& operator = ( const APDebugTree& aptree);

        std::auto_ptr < IDebugObjectsTree >& operator->() { return tree; }

        bool operator== ( const APDebugTree& apTree);
    };
//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------

class CEditorInterface: public IEditor, public ISystem
{
    private:
        CCont < APDebugTree> myLocalsList;

    public:
        CEditorInterface ();
        ~CEditorInterface ();

    void Init ();
    void Terminate ();


    //IEditor:

    void ShowEditorWindow ();

    void setCursorTo ( SymbolPosition position);
    void insertTextTo ( const char* text, SymbolPosition position);
    void output ( const char* text);
    //void task ( MessageData message);
    //void locals ( std::auto_ptr<IDebugObjectsTree> locals);

    void systemStarted();
```

```
    void stoppedAt ( int fileName, int position, int breakpointIndex);
    void systemStopped();
    void breakpointIsSet ( int breakpointIndex);
    void breakpointIsRemoved ( const BreakpointPosition& breakpoint);

    void __cdecl updateLocals();
    void clearLocals();
    void clickLocals ( int position);
    void redrawLocals();
    IDelayedExecutor* __cdecl GetThreadGate ();

    //ISystem
    int __cdecl printf (char*, ...);
    void __cdecl setBreakpoint ( const char* fileName, int stPos, int endPos);
    int  __cdecl getLocalsListNumber();
    void __cdecl clickLocalsAt ( int index);
    void __cdecl getLocalsAt ( int index, int* depth, const char** name, const char** value);
    void __cdecl CompileFile (char* filename, char* shortfilename);
    void __cdecl delCurBreakpoint();

    void __cdecl run();
    void __cdecl stop();
    void __cdecl pause();
    void __cdecl step();
    void __cdecl stepIn();
    void __cdecl stepOut();


    protected:
        typedef void (__cdecl* GetInstance)(void**);

        HMODULE m_EditorModule;
        IBCEditor* m_Editor;
        CThreadGate m_ThreadGate;

        DEFINE_IUNKNOWN ();
};

#endif //__EDITORCORE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл SmalltalkEditor\System.h

```
// System.h - header
// created 29.11.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\LinkerDataTypes\stdafx.h"
#include "..\Loader\Loader.h"
#include "..\Linker\linker.h"
#include "..\SmallVM\SmallVM.h"
#include "..\Console\Console.h"
#include "..\SmallCompiler\SmallCompiler.h"
#include "..\SmallDebugger\Debugger.h"
#include "Editor.h"

namespace EG
{
    extern ILoader* Loader;
    extern ILinker* Linker;
    extern ISmallVM* SmVM;
    extern ISmallCompiler* Compiler;
    extern IConsole* Console;
    extern ISmallDebugger* Debugger;
    extern IEditor* Editor;
    extern CScope* GS;
};

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл SmalltalkEditor\ThreadGate.h

```
// ThreadGate.h - header
// created 01.12.2002 by header creator
#pragma once
#ifndef __THREADGATE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __THREADGATE_HEADER_FILE_BY_VASFED_INCLUDED__

//DelayedExecutor - utility for walking throught threads
// (it makes possible to transfer function call from one thread to another)
//Be aware of cross-calling! This alpha-version does no logical test.

struct SDelayedCommand
{
    void* proc;
    int* params;
    int numparams;
```

```
    int procName;
    int returnType;
    void* EcxReg;
    char callingConversion;

    int timeCreated;
    int timeToLive; //0 - ever

    enum SupportedReturnTypes {RT_void, RT_bool, RT_char, RT_int, RT_float, RT_ptr};
    enum CallingConversions {E_CDECL, E_STDCALL, E_PASCAL, E_UNKNOWN};
};

struct SResult
{
    int type;

    union {
        bool b;
        char c;
        int  i;
        float f;
        void* p;
        } data;
};

interface IDelayedExecutor
{
    typedef void (__cdecl *VOIDPROC)(void);

    //client

    virtual bool __cdecl ReserveMemory (int numCalls = 2) = 0;
    virtual bool __cdecl StoreCallEx (SDelayedCommand) = 0;
    virtual bool __cdecl StoreCallS  (void* proc, int numParam, ...) = 0;
    virtual bool __cdecl StoreSimpleCall (VOIDPROC proc) = 0;
    virtual bool __cdecl StoreCall (void* proc, int rt, int procName, int timeToLive, void* ecx, int
numParam /*to be passed to proc*/, /*params*/...) = 0;

    //server
    virtual int  __cdecl PerformCall (/*out*/ int* procName, SResult* returnedValue) = 0;
    virtual int __cdecl ExecuteAll (bool stopOnError = false) = 0;

    //advanced:
    virtual int                __cdecl GetNumCalls (void) = 0;
    virtual SDelayedCommand*   __cdecl Command (int i) = 0;
    virtual int                __cdecl DeleteCommand (int i) = 0;

        //functions for freezing one of threads:
    virtual void __cdecl BlockInput       (bool value) = 0;
    virtual void __cdecl BlockOutput   (bool value) = 0;

    virtual void* __cdecl ConvertToVoidPtr (int, ...) = 0;

    enum ErrorMessages
        {
        ERM_FAILTURE    = false,
        ERM_OK          = true,
        ERM_UNDERFLOW   = -1,
        ERM_OVERFLOW    = -2,
        ERM_UNKNOWN     = -3,
        ERM_EXECUTIONFAILED = -4,
        ERM_UNKNOWNERROR =  -5,
        ERM_FORCEDWORD  = 0x8fffffff
        };
};

#endif //__THREADGATE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл SmalltalkEditor\ThreadGateCore.h

```
// ThreadGateCore.h - header
// created 01.12.2002 by header creator
#pragma once
#ifndef __THREADGATECORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __THREADGATECORE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "ThreadGate.h"

class CThreadGate: public IDelayedExecutor
{
    public:
        CThreadGate (int startsize = 10, int maxSize = 255);
        virtual ~CThreadGate ();

    //INTERFACE
        bool __cdecl ReserveMemory                (int numCalls = 2);
        bool __cdecl StoreCallEx                  (SDelayedCommand);
        bool __cdecl StoreCallS                   (void* proc, int numParam, ...);
        bool __cdecl StoreSimpleCall              (VOIDPROC proc);
        bool __cdecl StoreCall                    (void* proc, int rt, int procName, int timeToLive,
void*, int numParam /*to be passed to proc*/, /*params*/...);
        int  __cdecl PerformCall                  (/*out*/ int* procName, SResult* returnedValue);
```

```
        int  __cdecl ExecuteAll                    (bool stopOnError = false);
        int          __cdecl GetNumCalls    (void);
        SDelayedCommand*   __cdecl Command        (int i);
        int          __cdecl DeleteCommand  (int i);
        void __cdecl BlockInput    (bool value);
        void __cdecl BlockOutput   (bool value);
        void* __cdecl ConvertToVoidPtr (int, ...);


    protected:
        bool Push          (SDelayedCommand); //FIFO
        bool Pop           (SDelayedCommand*);

        bool Grow   (int type = 1);
        bool GrowFor(int);
        bool GrowTo (int);

        SDelayedCommand* m_data;
        volatile int m_dataSize;
        volatile int m_dataPutP;
        volatile int m_dataGetP;
        volatile int m_maxSize;

        volatile bool m_blockInput;
        volatile bool m_blockOutput;
        volatile int  m_numCalls;
};

#endif //__THREADGATECORE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл SmalltalkEditor\Builder\About.h

```
//---------------------------------------------------------------------------
#ifndef AboutH
#define AboutH
//---------------------------------------------------------------------------
#include <vcl\System.hpp>
#include <vcl\Windows.hpp>
#include <vcl\SysUtils.hpp>
#include <vcl\Classes.hpp>
#include <vcl\Graphics.hpp>
#include <vcl\Forms.hpp>
#include <vcl\Controls.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Buttons.hpp>
#include <vcl\ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TAboutBox : public TForm
{
__published:
    TPanel *Panel1;
    TImage *ProgramIcon;
    TLabel *ProductName;
    TLabel *Version;
    TLabel *Copyright;
    TLabel *Comments;
    TButton *OKButton;
private:
public:
    virtual __fastcall TAboutBox(TComponent* AOwner);
};
//---------------------------------------------------------------------------
extern PACKAGE TAboutBox *AboutBox;
//---------------------------------------------------------------------------
#endif
```

Файл SmalltalkEditor\Builder\DebugForm.h

```
//---------------------------------------------------------------------------

#ifndef DebugFormH
#define DebugFormH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
//---------------------------------------------------------------------------
class TDebugFrm : public TForm
{
__published:    // IDE-managed Components
    TListBox *List;
    void __fastcall ListDblClick(TObject *Sender);
    void __fastcall FormShow(TObject *Sender);
    void __fastcall Redraw();
    void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
```

```
    void __fastcall FormResize(TObject *Sender);
private:    // User declarations
public:     // User declarations
    __fastcall TDebugFrm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TDebugFrm *DebugFrm;
//---------------------------------------------------------------------------
#endif
```

Файл SmalltalkEditor\Builder\DocumentForm.h

```
//---------------------------------------------------------------------------

#ifndef DocumentFormH
#define DocumentFormH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <Menus.hpp>
#include <Dialogs.hpp>
//---------------------------------------------------------------------------
class TDocument : public TForm
{
__published:    // IDE-managed Components
        TRichEdit *Edit;
        TMainMenu *MainMenu;
        TMenuItem *Edit1;
        TMenuItem *View1;
        TMenuItem *ColorizeText1;
        TMenuItem *Undo1;
        TMenuItem *N1;
    TSaveDialog *SaveDialog;
    TMenuItem *Tools1;
    TMenuItem *Addclassdeclaration1;
    TMenuItem *Addmethoddeclaration1;
    TMenuItem *SearchReplace1;
    TMenuItem *SearchAgain1;
    TMenuItem *N2;
    TMenuItem *Decreaseindent1;
    TMenuItem *Increaseindent1;

        void __fastcall FormResize(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
    void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
    void __fastcall ColorizeText1Click(TObject *Sender);
    void __fastcall FormShortCut(TWMKey &Msg, bool &Handled);
    void __fastcall FormKeyPress(TObject *Sender, char &Key);
    void __fastcall Addclassdeclaration1Click(TObject *Sender);
    void __fastcall Addmethoddeclaration1Click(TObject *Sender);
    void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
    void __fastcall SetFocus();
    void __fastcall Undo1Click(TObject *Sender);
    void __fastcall SearchReplace1Click(TObject *Sender);
    void __fastcall SearchAgain1Click(TObject *Sender);
    void __fastcall Decreaseindent1Click(TObject *Sender);
    void __fastcall Increaseindent1Click(TObject *Sender);
private:    // User declarations
public:     // User declarations
        __fastcall TDocument(TComponent* Owner);
        __fastcall ~TDocument ();

        void SaveClick         (void);
        void SaveAsClick       (void);
        void CloseClick         (void);
        void OpenFile           (AnsiString filename);
        void ColorizeText       (void);

        AnsiString m_Filename;
};
//---------------------------------------------------------------------------
extern PACKAGE TDocument *Document;
//---------------------------------------------------------------------------
#endif
```

Файл SmalltalkEditor\Builder\EditorInt.h

```
//---------------------------------------------------------------------------

#ifndef EditorIntH
#define EditorIntH
//---------------------------------------------------------------------------
#include <stdarg.h>
```

```
#define interface struct
#include "..\ThreadGate.h"

struct ISystem
{
    virtual int  __cdecl printf (char*, ...) = 0;
    virtual void __cdecl setBreakpoint ( const char* fileName, int stPos, int endPos) = 0;
    virtual int  __cdecl getLocalsListNumber() = 0;
    virtual void __cdecl clickLocalsAt ( int index) = 0;
    virtual void __cdecl getLocalsAt ( int index, int* depth, const char** name, const char** value) = 0;
    virtual void __cdecl updateLocals() = 0;
    virtual void __cdecl delCurBreakpoint() = 0;
    virtual IDelayedExecutor* __cdecl GetThreadGate () = 0;

//    enum eCommands { eRun, eStop, ePause, eStep, eStepIn, eStepOut };
    virtual void __cdecl run() = 0;
    virtual void __cdecl stop() = 0;
    virtual void __cdecl pause() = 0;
    virtual void __cdecl step() = 0;
    virtual void __cdecl stepIn() = 0;
    virtual void __cdecl stepOut() = 0;

    virtual void __cdecl CompileFile (char* filename, char* shortfilename = NULL) = 0;
};

struct IBCEditor
{
    virtual void __cdecl Init (ISystem* syst) = 0;
    virtual void __cdecl Terminate () = 0;
    virtual void __cdecl ShowWindow () = 0;
    virtual void __cdecl OpenFile (char*) = 0;
    virtual void __cdecl RedrawLocals() = 0;

    virtual void __cdecl SetCursor ( const char* file, int position, int length = 0) = 0;
};

#ifndef __INTERFACE

extern ISystem* Syst;

class CEditor: public IBCEditor
{
    public:
        CEditor();
        ~CEditor ();

        //IEditor:
        void __cdecl Init (ISystem*);
        void __cdecl Terminate ();
        void __cdecl ShowWindow ();
        void __cdecl OpenFile (char*);
        void __cdecl RedrawLocals();

        void __cdecl SetCursor ( const char* file, int position, int length = 0);

        void __cdecl GetCommands ();
};

#endif

#endif
```

Файл SmalltalkEditor\Builder\ErrorMessagesForm.h

```
//---------------------------------------------------------------------------

#ifndef ErrorMessagesFormH
#define ErrorMessagesFormH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Grids.hpp>
#include <ValEdit.hpp>
//---------------------------------------------------------------------------
class TErrorMessageForm : public TForm
{
__published:   // IDE-managed Components
    TListBox *List;
    void __fastcall FormShow(TObject *Sender);
    void __fastcall FormResize(TObject *Sender);
    void __fastcall ListDblClick(TObject *Sender);
private:    // User declarations
public:     // User declarations
    __fastcall TErrorMessageForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TErrorMessageForm *ErrorMessageForm;
//---------------------------------------------------------------------------
#endif
```

Файл SmalltalkEditor\Builder\Form.h

```
//---------------------------------------------------------------------------

#ifndef FormH
#define FormH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:    // IDE-managed Components
private:    // User declarations
public:     // User declarations
    __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Файл SmalltalkEditor\Builder\Globals.h

```
//---------------------------------------------------------------------------

#ifndef GlobalsH
#define GlobalsH
//---------------------------------------------------------------------------

enum ColorNames
        {
        None,
        Default,
        Object,
        Method,
        MethodDef,
        MethodDefParam,
        FileHeader,
        SystemChar,
        Dot,
        MethodCascadeSymbol,
        Operator,
        VZ,
        Str,
        Number,
        Comment,
        ArraySymbol,
        Brackets,
        Block,
        BackColor,
        TemporaryVariablesSymbol,
        Error,
         MaxColorName
        };

extern int ColorValues[];
extern int NumColorizeKeys;
extern char ColorizeKeysCh[];
extern bool ColorizeKeysEnabled;

int MakeColor (ColorNames color);
int* MakeColorPointer (ColorNames color);

#endif
```

Файл SmalltalkEditor\Builder\Main.h

```
#ifndef __MAIN_UNIT_HEADER_FILE_INCLUDED__
#define __MAIN_UNIT_HEADER_FILE_INCLUDED__

#include "EditorInt.h"

extern CEditor Editor;

#endif // __MAIN_UNIT_HEADER_FILE_INCLUDED__
```

Файл SmalltalkEditor\Builder\MainFrm.h

```
//---------------------------------------------------------------------------
#ifndef MainH
#define MainH
//---------------------------------------------------------------------------
#include <sysutils.hpp>
#include <windows.hpp>
#include <messages.hpp>
#include <sysutils.hpp>
#include <classes.hpp>
#include <graphics.hpp>
#include <controls.hpp>
#include <forms.hpp>
#include <dialogs.hpp>
#include <stdctrls.hpp>
#include <buttons.hpp>
#include <extctrls.hpp>
#include <menus.hpp>
#include <Classes.hpp>
#include <ComCtrls.hpp>
#include <Controls.hpp>
#include <Dialogs.hpp>
#include <Menus.hpp>
#include "DebugForm.h"
#include <Grids.hpp>
#include <ValEdit.hpp>
#include <Buttons.hpp>
#include <StdCtrls.hpp>

#include "ErrorMessagesForm.h"
#include "DocumentForm.h"
#include <ToolWin.hpp>
#include <ImgList.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TMainForm : public TForm
{
__published:
    TMainMenu *MainMenu;
    TMenuItem *FileNewItem;
    TMenuItem *FileOpenItem;
    TMenuItem *FileExitItem;
    TMenuItem *WindowTileItem;
    TMenuItem *WindowCascadeItem;
    TMenuItem *WindowArrangeItem;
    TMenuItem *HelpContentsItem;
    TMenuItem *HelpHowToUseItem;
    TMenuItem *HelpAboutItem;
    TStatusBar *StatusLine;
    TOpenDialog *OpenDialog;
        TMenuItem *Preferences1;
        TMenuItem *N2;
        TMenuItem *N4;
        TMenuItem *Save1;
        TMenuItem *SaveAs1;
        TMenuItem *Close1;
        TMenuItem *WindowList1;
    TMenuItem *Debug1;
    TMenuItem *Showlocals1;
    TMenuItem *Compiler1;
    TMenuItem *Compile1;
    TMenuItem *Showbreakpoints1;
    TMenuItem *Setbreakpointcall1;
    TMenuItem *N5;
    TMenuItem *Run1;
    TMenuItem *N6;
    TCoolBar *CoolBar;
    TToolBar *ToolBar1;
    TImageList *ImageList1;
    TToolButton *ToolButton1;
    TToolButton *ToolButton2;
    TToolButton *ToolButton3;
    TToolButton *ToolButton4;
    TToolButton *ToolButton5;
    TToolButton *ToolButton6;
    TToolButton *ToolButton7;
    TToolBar *ToolBar2;
    TButton *FindWindowBtn;
    TComboBox *FindWindowNames;
    TTimer *GetSystemMessagesTimer;
    TToolBar *DebugToolBar;
    TToolButton *DebugRun;
    TToolButton *DebugPause;
    TToolButton *DebugStepInto;
    TToolButton *DebugStepOut;
    TToolButton *StepOver;
    TToolButton *DebugStop;
    TButton *ShowLocals;
    TMenuItem *Stepinto1;
    TMenuItem *StepOver1;
    TMenuItem *StepOut1;
    TMenuItem *Pauseatexecutionpoint1;
```

```
        TButton *Button1;
    TTimer *StatusBarTimer;
    TMenuItem *CompileAll1;
    TMenuItem *N7;
    TMenuItem *ShowMessagesWindow1;
    TMenuItem *FileSaveAll;
    void __fastcall FormCreate(TObject *Sender);
    void __fastcall ShowHint(TObject *Sender);
    void __fastcall FileOpen(TObject *Sender);
    void __fastcall FileSave(TObject *Sender);
    void __fastcall FileExit(TObject *Sender);
    void __fastcall EditUndo(TObject *Sender);
    void __fastcall EditCut(TObject *Sender);
    void __fastcall EditCopy(TObject *Sender);
    void __fastcall EditPaste(TObject *Sender);
    void __fastcall WindowTile(TObject *Sender);
    void __fastcall WindowCascade(TObject *Sender);
    void __fastcall WindowArrange(TObject *Sender);
    void __fastcall HelpContents(TObject *Sender);
    void __fastcall HelpSearch(TObject *Sender);
    void __fastcall HelpHowToUse(TObject *Sender);
    void __fastcall HelpAbout(TObject *Sender);
        void __fastcall FileNewItemClick(TObject *Sender);
        void __fastcall Save1Click(TObject *Sender);
        void __fastcall SaveAs1Click(TObject *Sender);
        void __fastcall Close1Click(TObject *Sender);
    void __fastcall Preferences1Click(TObject *Sender);
    void __fastcall Showlocals1Click(TObject *Sender);
    void __fastcall Setbreakpointcall1Click(TObject *Sender);
    void __fastcall Run1Click(TObject *Sender);
    void __fastcall Compile1Click(TObject *Sender);
    void __fastcall BitBtn1Click(TObject *Sender);
    void __fastcall FindWindowBtnClick(TObject *Sender);
    void __fastcall ToolButton1Click(TObject *Sender);
    void __fastcall ToolButton2Click(TObject *Sender);
    void __fastcall ToolButton3Click(TObject *Sender);
    void __fastcall FindWindowNamesKeyPress(TObject *Sender, char &Key);
    void __fastcall GetSystemMessagesTimerTimer(TObject *Sender);
    void __fastcall DebugRunClick(TObject *Sender);
        void __fastcall DebugPauseClick(TObject *Sender);
        void __fastcall DebugStopClick(TObject *Sender);
        void __fastcall DebugStepIntoClick(TObject *Sender);
        void __fastcall StepOverClick(TObject *Sender);
        void __fastcall DebugStepOutClick(TObject *Sender);
        void __fastcall DelCurBreakpoint(TObject *Sender);
    void __fastcall Button1Click(TObject *Sender);
    void __fastcall StatusBarTimerTimer(TObject *Sender);
    void __fastcall Say (AnsiString s);
    void __fastcall CompileAll1Click(TObject *Sender);
    void __fastcall ShowMessagesWindow1Click(TObject *Sender);
    void __fastcall Stepinto1Click(TObject *Sender);
    void __fastcall StepOver1Click(TObject *Sender);
    void __fastcall StepOut1Click(TObject *Sender);
    void __fastcall Pauseatexecutionpoint1Click(TObject *Sender);
    void __fastcall FileSaveAllClick(TObject *Sender);
    void __fastcall FormShow(TObject *Sender);

public:

    TDebugFrm* m_debugFrm;
    TErrorMessageForm* m_ErrorsForm;
    //Search forms

    //Other forms

    //Some functions

    TDocument* ActivateDocument (AnsiString filename);
        //activates or opesns a document
    TDocument* GotoPosition (AnsiString file, int position, int len = 0, bool open = false, bool
fromthread = false);
        //activates(or opens) document and sets selection to place
    void EnterDebugMode ();
    void LeaveDebugMode ();

    TErrorMessageForm* InvokeErrorsWindow ();

    virtual __fastcall TMainForm(TComponent* Owner);
    __fastcall ~TMainForm ();
};
//---------------------------------------------------------------------------
extern TMainForm *MainForm;
//---------------------------------------------------------------------------
#endif
```

Файл SmalltalkEditor\Builder\Options.h

```
//---------------------------------------------------------------------------
#ifndef OptionsH
#define OptionsH
//---------------------------------------------------------------------------
```

```
#include <vcl\ExtCtrls.hpp>
#include <vcl\ComCtrls.hpp>
#include <vcl\Buttons.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Controls.hpp>
#include <vcl\Forms.hpp>
#include <vcl\Graphics.hpp>
#include <vcl\Classes.hpp>
#include <vcl\SysUtils.hpp>
#include <vcl\Windows.hpp>
#include <vcl\System.hpp>
#include "CSPIN.h"
#include <Dialogs.hpp>
//---------------------------------------------------------------------------
class TOptionsDlg : public TForm
{
__published:
    TPanel *Panel1;
    TPageControl *OptionsTabSheet;
    TTabSheet *GeneralOptions;
    TTabSheet *ColorsSheet;
    TTabSheet *OtherOptions;
    TButton *LoadDefaultsBtn;
    TButton *SaveOptionsToFileBtn;
    TButton *LoadOptionsFromFileBtn;
    TGroupBox *GroupBox1;
    TCSpinEdit *AutosaveDelay;
    TCheckBox *AutosaveEnabled;
    TLabel *Label1;
    TLabel *Label2;
    TOpenDialog *OpenDialog;
    TSaveDialog *SaveDialog;
    TColorDialog *ColorDialog;
    TFontDialog *FontDialog;
    TTimer *AutosaveTimer;
    TListBox *ColorsList;
    TLabel *Label3;
    TLabel *Label4;
    TPanel *TextExample;
    TButton *ChangeColorBtn;
    TLabel *Label5;
    TCheckBox *ColorizerEnabler;
    void __fastcall ColorsSheetShow(TObject *Sender);
    void __fastcall ChangeColorBtnClick(TObject *Sender);
    void __fastcall ColorsListClick(TObject *Sender);
    void __fastcall ColorizerEnablerClick(TObject *Sender);
private:
public:
    virtual __fastcall TOptionsDlg(TComponent* AOwner);
};
//---------------------------------------------------------------------------
extern PACKAGE TOptionsDlg *OptionsDlg;
//---------------------------------------------------------------------------
#endif
```

Файл SmalltalkEditor\Builder\SearchReplace.h

```
//---------------------------------------------------------------------------

#ifndef SearchReplaceH
#define SearchReplaceH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include "DocumentForm.h"
//---------------------------------------------------------------------------
class TSearchReplaceForm : public TForm
{
__published:    // IDE-managed Components
    TEdit *FindWhat;
    TEdit *ReplaceWith;
    TRadioGroup *SearchWhere;
    TButton *Search;
    TButton *Replace;
    TButton *ReplaceAll;
    TButton *Cancel;
    TCheckBox *MatchCase;
    TCheckBox *MatchWholeWord;
    TLabel *Label1;
    TLabel *Label2;
    void __fastcall CancelClick(TObject *Sender);
    void __fastcall SearchClick(TObject *Sender);
    void __fastcall ReplaceClick(TObject *Sender);
    void __fastcall ReplaceAllClick(TObject *Sender);
    void __fastcall FormKeyPress(TObject *Sender, char &Key);
    void __fastcall FormShow(TObject *Sender);
private:    // User declarations
    int m_SearchStart;
```

```
    int m_SearchEnd;
    TDocument* doc;
public:     // User declarations
    __fastcall TSearchReplaceForm(TComponent* Owner);
    bool m_LastSearchSucceded;
    bool m_PromptError;
};
//---------------------------------------------------------------------------
extern PACKAGE TSearchReplaceForm *SearchReplaceForm;
//---------------------------------------------------------------------------
#endif
```

Файл SmallVM\DebugObjectsTree.h

```
#pragma once
#if !defined (DEBUG_OBJECTS_TREE_HEADER_INCLUDED_BY_SE)
#define DEBUG_OBJECTS_TREE_HEADER_INCLUDED_BY_SE

#include "SmallVM.h"

class CDebugObjectsTree : public IDebugObjectsTree
    {
    private:
    CObj* myMe; //собственный я
    CString myString;
    CString myName;
    CCont < std::pair< CString, CObj* > * > myObjects;

    public:
        CDebugObjectsTree ( const CString& name, CObj* obj);
        ~CDebugObjectsTree();

        bool ok();

        const char* asString();
        const char* rName();

        int number();
        std::auto_ptr<IDebugObjectsTree> operator[] ( int index);
        IDebugObjectsTree* expand(); //return this
        IDebugObjectsTree* unexpand(); //return this

        CDebugObjectsTree* addObject ( const CString& name, CObj* obj);
    };

#endif //DEBUG_OBJECTS_TREE_HEADER_INCLUDED_BY_SE
```

Файл SmallVM\SmallVM.h

```
#if !defined (SMALLVM_FILE_LOADED)
#define SMALLVM_FILE_LOADED

#pragma once

#include <objbase.h>
#include <memory>
#include "..\Linker\linker.h"

interface IDebugObjectsTree
    {
        virtual const char* asString() = 0;
        virtual const char* rName() = 0;

        virtual int number() = 0;
        virtual std::auto_ptr<IDebugObjectsTree> operator[] ( int index) = 0;
        virtual IDebugObjectsTree* expand() = 0;
        virtual IDebugObjectsTree* unexpand() = 0;
    };

// {0CA96762-3F46-4d54-8F68-62660C30692C}
static const GUID IID_ISmallVM =
{ 0xca96762, 0x3f46, 0x4d54, { 0x8f, 0x68, 0x62, 0x66, 0xc, 0x30, 0x69, 0x2c } };

interface ISmallVM : public IVM
    {
    virtual SParam staticCall (int className, int classSel, int funcName, int funcSel, const CParams&
params) = 0;
    virtual SParam call ( CObj* obj, int className, int classSel, int funcName, int funcSel, const
CParams& params) = 0;
    virtual int getClassSelector (int className) = 0;
    virtual int getFuncSelector (int className, int funcName) = 0;
    virtual int getStaticFuncSelector (int className, int funcName) = 0;
    virtual std::auto_ptr<IDebugObjectsTree> getWorkTree() = 0;
        virtual CScope* getWorkScope() = 0;
        virtual void setWorkScope ( CScope* scope) = 0;
        virtual void releaseWorkScope() = 0;
        virtual int  workScopeDepth() = 0;
```

```
    };
#endif
```

Файл SmallVM\SmallVMCore.h

```
#if !defined (SMALLVMCORE_FILE_LOADED)
#define SMALLVM_FILECORE_LOADED

#pragma once

#define SMVM_API __declspec (dllexport)

//#include <stdarg.h>
//#include <string.h>
#include <conio.h>

#include "SmallVM.h"
#include "system.h"

class SMVM_API CCommand;
class SMVM_API CSmObj;
class SMVM_API CSmObjPrototype;
struct SMVM_API SmallClass;
class SMVM_API CSmallVM;

//-----------------------------------------------------------------------------

class SMVM_API CCommand
    {
    public:
    enum action { eNothing = 0, eNew, eCall, ePush, ePushI, ePushG, ePushL, ePop, ePopG, ePopL,
eSetWorkScope, eUpdateWorkPosition, eReturn };
    enum stdTypes { eInteger = 1, eFloat, eChar, eStr, eBlock };

    private:
    action myCode;
    long myParam;

    public:
    CCommand ( action code = eNothing, int param = 0);
    ~CCommand();

    action& getCode() { return myCode; }
    long&   getParam(){ return myParam; }
    action    rCode()  const { return myCode; }
    long      rParam() const { return myParam; }

    bool operator== ( const CCommand& com);
    };

//-----------------------------------------------------------------------------

class SMVM_API Block
{
private:
    CStore<CCommand> myCommands;
    CScope myScope;

public:

    Block();
    Block ( int i); //Block(0)
    Block ( const Block& block);

    const CStore<CCommand>& rCommands() const { return myCommands; }
    CStore<CCommand>& gCommands() { return myCommands; }
    const CScope& rScope() const { return myScope; }
    CScope& gScope() { return myScope; }

    bool operator== ( const Block& block) const;

    Block* addCommand ( CCommand::action action, int parameter);
    Block* addCommand ( const CCommand& com);

    Block* addLocalVar ( int name);

    SParam call ( CScope* upper, const CParams& params);

};

//-----------------------------------------------------------------------------

class SMVM_API BlockPtr
{
private:
    int myIndex;
public:
    BlockPtr();
    BlockPtr ( int i);
```

```
    int gIndex();

    Block* operator-> ();
    Block* gBlock();
};

//---------------------------------------------------------------------------

class SMVM_API BlockObj : public CObj
{
private:
    static SClassRef sClass;

private:
    BlockPtr myBlock;
    CScope* myParent;

public:
    BlockObj ( BlockPtr block, CScope* parent);

    static bool selfReg();

    SParam call ( const CParams& params); //0
    SParam callWith ( const CParams& params); //1
    SParam callWithWith ( const CParams& params); //2
    SParam callWithWithWith ( const CParams& params); //3
};

//---------------------------------------------------------------------------

class SMVM_API ConstString
{
private:
    CString myString;
public:
    ConstString();
    ConstString ( const CString& str);
    ConstString ( const ConstString& str);
    ~ConstString()
        {

        }
    ConstString& operator = ( const ConstString& str);

    const char* rString() const;
    const CString& rStdString() const { return myString; }
    CString& gStdString() { return myString; }

    bool operator == ( const ConstString& str) const ;
};

//---------------------------------------------------------------------------

class SMVM_API ConstStringPtr
{
private:
    int myIndex;
public:
    ConstStringPtr ( int i = 0);

    int rIndex() const { return myIndex; }
    int& gIndex() { return myIndex; }

    ConstString* operator-> ();
    ConstString* gString();
};

//---------------------------------------------------------------------------

struct SMVM_API ISmallFunc
    {
    int mySelInL;
    int myName;
    int myClassName;
    BlockPtr myBlock;
    CParamsTypes myTypes;

    ISmallFunc ( int className = 0, int funcName = 0, const BlockPtr& bl = BlockPtr() );
    virtual ~ISmallFunc() = 0;

    virtual bool operator== ( const ISmallFunc& func) const;

    ISmallFunc& addParamType ( SParamTypeInfo t = SParamTypeInfo());
    };

struct SMVM_API SmallFunc : public ISmallFunc
    {

    SmallFunc ( int className = 0, int funcName = 0, const BlockPtr& bl = BlockPtr() );
    ~SmallFunc();
    };

struct SMVM_API SmallStaticFunc : public ISmallFunc
    {
    SmallStaticFunc ( int className = 0, int funcName = 0, const BlockPtr& bl = BlockPtr() );
    ~SmallStaticFunc();
    };
```

```
//-----------------------------------------------------------------------------

class SMVM_API CSmObj : public CObj
    {
    protected:
    SmallClass* myClassObject;
    CScope myScope;

    public:
    CSmObj ( SmallClass* cl = 0);
    CSmObj ( const CSmObjPrototype& objPro);

    SmallClass* getClassObject() throw() { return myClassObject; }
    SmallClass* rClassObject() const throw() { return myClassObject; }

    CScope& CSmObj::getScope() throw() { return myScope; }
    const CScope& CSmObj::rScope() const throw() { return myScope; }

    bool operator== ( const CSmObj& obj);
    };

class SMVM_API CSmObjPrototype : public CSmObj
    {
    public:
    CSmObjPrototype ( SmallClass* cl);

    CSmObjPrototype& setClassRef ( SClassRef cl) { myClass = cl; return *this; }

    bool ok() const;

    CSmObjPrototype& addVar ( const CVar& var);
    };

//-----------------------------------------------------------------------------

struct SMVM_API SmallClass
    {
    int myClassName;
    SClassRef myClassRef;
    CStore<SmallFunc> myFuncs;
    CStore<SmallStaticFunc> myStaticFuncs;

    SmallClass* myBaseClass;
    CSmObjPrototype myObjPro;
    CScope myScope;

    SmallClass ( int className = 0);

    int getFuncSelector (int funcName);
    int getStaticFuncSelector (int funcName);

    bool operator== ( const SmallClass& cl) const;

    SmallClass& setBaseClass ( SmallClass* base);
    SmallClass& regBaseClass ( int baseClassName);

    bool selfReg();

    SParam sNew ( const CParams& );

    SmallClass& addFunc ( const SmallFunc& func);
    SmallClass& addStaticFunc ( const SmallStaticFunc& func);

    SParam call ( CSmObj* obj, int funcName, int funcSel, const CParams& params);
    SParam staticCall ( int funcName, int funcSel, const CParams& params);

    SmallClass& addVar ( const CVar& v) { myScope.addVar ( v); return *this; }
    };

//-----------------------------------------------------------------------------

class SMVM_API CStack
    {
    CCont<int> myObjs;
    CCont<CObjRef> myRefs;

    public:
    CStack();
    ~CStack();

    bool ok();

    CStack& push ( int);
    CStack& pushObj ( const CObjRef& obj);
    int pop();
    CObjRef popObj();
    };

//-----------------------------------------------------------------------------

struct SMVM_API APBlock
{
    std::auto_ptr<Block> myBlock;

    APBlock();
    APBlock ( std::auto_ptr<Block> bl);
```

```
    APBlock& operator= ( const APBlock& bl);
    bool operator== ( const APBlock& bl);
    std::auto_ptr<Block>* operator-> ();
};

//---------------------------------------------------------------------------

class SMVM_API CSmallVM : public ISmallVM
    {
    protected:
    CStack myStack;
    CCont<SmallClass*> myClasses;
    CCont<APBlock> myBlocks;
    CCont<ConstString> myStrings;
    Stack < CScope*> myWorkScope;

    public:
    DEFINE_IUNKNOWN();

    class SMVM_API NotRealizedMethod;

    CSmallVM();

    std::auto_ptr<IDebugObjectsTree> getWorkTree();
        CScope* getWorkScope() { return myWorkScope.top(); }
        void setWorkScope ( CScope* scope) { myWorkScope.push ( scope); }
        void releaseWorkScope() { myWorkScope.pop(); }
        int  workScopeDepth() { return myWorkScope.depth(); }

    CStack& getStack() { return myStack; }
    CCont<SmallClass*> getClasses() { return myClasses; } //badbadbad

    SParam staticCall ( int className, int classSel, int funcName, int funcSel, const CParams& params);
    SParam call ( CObj* obj, int className, int classSel, int funcName, int funcSel, const CParams&
params);

    static SParam Obj2Par ( CObjRef& obj);
    static CObjRef Par2Obj ( const SParam& par);

    public:

    BlockPtr rFuncPtr ( Func* func);
    BlockPtr rFuncPtr ( int className, int classSelector, int funcName, int funcSelector); //uses
selectors if non zero otherwise get 'em from names
    BlockPtr rStatFuncPtr ( StaticFunc* func);
    BlockPtr rStatFuncPtr ( int className, int classSelector, int funcName, int funcSelector); //uses
selectors if non zero otherwise get 'em from names

    int getClassSelector ( int className);
    int getFuncSelector ( int className, int funcName);
    int getStaticFuncSelector ( int className, int funcName);

    SmallClass* getClassPtr ( int className);
    SmallClass* getClassPtrByIndex ( int index) { return myClasses[index]; }

    BlockPtr regBlock ( std::auto_ptr<Block> block);
        Block* getBlock ( int index);
    ConstStringPtr regConstString ( const char* str);
    ConstStringPtr regConstString ( const ConstString& str);
        ConstString* getConstString ( int index);

    SClassRef registerClass ( int className, int varName);
    SClassRef registerBaseClass ( int className, int baseClassName);

    Func* registerFunc ( SClassRef cl, int funcName, const BlockPtr& func, const CParamsTypes& types);
    StaticFunc* registerStaticFunc ( SClassRef cl, int funcName, const BlockPtr& func, const CParamsTypes&
types);
    };

//---------------------------------------------------------------------------

class SMVM_API CSmallVM::NotRealizedMethod
    {
    CString myMethodName;

    public:
    NotRealizedMethod ( const CString& string) :
        myMethodName ( string)
        { }

    const CString& rMethodName() const { return myMethodName; }
    const char* rcMethodName() const { return myMethodName.rString(); }
    };

//---------------------------------------------------------------------------

typedef SParam (SmallClass::*pSmallClassFunc) ( const CParams&);

//---------------------------------------------------------------------------

class SmallObjFunc : public ObjFunc
{
    SmallClass* myClass;
    pSmallClassFunc myFunc;

public:
    SmallObjFunc ( SmallClass* cl, pSmallClassFunc func);
```

```
    SParam operator() ( const CParams& params);
};

void RegisterStaticFuncForSmallClass ( SClassRef cl, const CString& funcName, pSmallClassFunc func, const
CParamsTypes& types);

//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

#endif
```

Файл SmallVM\system.h

```
#ifndef SMALLVM_SYSTEM_FILE_LOADED
#define SMALLVM_SYSTEM_FILE_LOADED

#pragma once

#pragma warning (disable: 4541)
#pragma warning (disable: 4251) //class needs to have dll-interface to be used by clients of class

#include "..\Loader\Loader.h"
#include "..\Linker\linker.h"
#include "..\CPPVM\CPPVM.h"
#include "SmallVM.h"
#include "..\Console\Console.h"
#include "..\SmallDebugger\Debugger.h"

class CSmallVM;

namespace SmallVM
{
    extern ILoader* Loader;
    extern ILinker* Linker;
    extern CSmallVM* SmallVM;
    extern ICPPVM* CPPVM;
    extern IConsole* Console;
    extern ISmallDebugger* Debugger;
};

#endif
```

Файл SmallVM\Standard Smalltalk Library\Collection Protocols.h

```
#if !defined (COLLECTION_PROTOCOLS)
#define COLLECTION_PROTOCOLS

#include "..\system.h"
#include "Fundamental Protocols.h"

//---------------------------------------------------------------------------
/*
Implementation specifications:
    none yet :)
*/
//---------------------------------------------------------------------------

bool CollectionProtocolsReg();

//---------------------------------------------------------------------------

//class Collection : public Object
//  {
//  private:
//  static SClassRef sClass;
//
//  public:
//
//  Collection();
//  virtual ~Collection() { }
//
//  static bool selfReg();
//
//  virtual SParam sDo ( const CParams& parama) = 0; //1 - block with 1 parameter
//
//  /* //collection
//  allSatisfy:
//  anySatisfy:
//  asArray
//  asBag
//  asByteArray
//  asOrderedCollection
//  asSet
//  asSortedCollection
```

```
//  asSortedCollection:
//  collect:
//  detect:
//  detect:ifNone:
//  //do:
//  do:separatedBy:
//  includes:
//  inject:into:
//  isEmpty
//  notEmpty
//  occurrencesOf:
//  rehash
//  reject:
//  select:
//  size
//  */
//  };
//
//class SequencedReadableCollection : public Collection
//  {
//  private:
//  static SClassRef sClass;
//
//  public:
//
//  SequencedReadableCollection();
//  virtual ~SequencedReadableCollection() { }
//
//  static bool selfReg();
//
//  virtual SParam sDo ( const CParams& parama) = 0; //1 - block with 1 parameter
//
//  /*//sequencedReadableCollection
//  ,
//  =
//  after:
//  at:
//  at:ifAbsent:
//  before:
//  copyFrom:to:
//  copyReplaceAll:with:
//  copyReplaceFrom:to:with:
//  copyReplaceFrom:to:withObject:
//  copyReplacing:withObject:
//  copyWith:
//  copyWithout:
//  do:
//  findFirst:
//  findLast:
//  first
//  from:to:do:
//  from:to:keysAndValuesDo:
//  indexOf:
//  indexOf:ifAbsent:
//  indexOfSubCollection:startingAt:
//  indexOfSubCollection:startingAt:ifAbsent:
//  keysAndValuesDo:
//  last
//  reverse
//  reverseDo:
//  with:do:
//  */
//  };
//
//class ReadableString : public Object
//  {
//  private:
//  static SClassRef sClass;
//
//  public:
//
//  ReadableString();
//  virtual ~ReadableString() { }
//
//  static bool selfReg();
//
//  virtual SParam sDo ( const CParams& parama) = 0; //1 - block with 1 parameter
//
//  /*//readableString
//  ,
//  <
//  <=
//  >
//  >=
//  asLowercase
//  asString
//  asSymbol
//  asUppercase
//  copyReplaceAll:with:
//  copyReplaceFrom:to:with:
//  copyReplacing:withObject:
//  copyWith:
//  sameAs:
//  subStrings:
//  */
//  };
//
```

```
//class SequencedCollection : public Object
//  {
//  private:
//  static SClassRef sClass;
//
//  public:
//
//  SequencedCollection();
//  virtual ~SequencedCollection() { }
//
//  static bool selfReg();
//
//  virtual SParam sDo ( const CParams& parama) = 0; //1 - block with 1 parameter
//
//  /*//sequencedCollection
//  at:put:
//  atAll:put:
//  atAllPut:
//  replaceFrom:to:with:
//  replaceFrom:to:with:startingAt:
//  replaceFrom:to:withObject:
//  */
//  };

class SString : public Object
    {
    private:
    static SClassRef sClass;

    std::string myString;

    public:

    SString();
    SString ( const char* str);
    virtual ~SString();

    const std::string& rString() const { return myString; }

    static SClassRef rClass() { return sClass; }
    static bool selfReg();

    SParam sDo ( const CParams& params); //1 - block with 1 parameter

    SParam /*SString*/ operator+  ( const CParams& params); //1 - string
    SParam /*SString*/ operator+= ( const CParams& params); //1 - string
    SParam /*eString*/ asString   ( const CParams& params); //0


    /*//string
    asString
    */
    };

#endif
```

Файл SmallVM\Standard Smalltalk Library\Fundamental Protocols.h

```
#if !defined (FUNDAMENTAL_PROTOCOLS_FILE_LOADED)
#define FUNDAMENTAL_PROTOCOLS_FILE_LOADED

#include "..\system.h"

//---------------------------------------------------------------------------

bool FundamentalProtocolsReg();

//---------------------------------------------------------------------------

extern int SmallLibraryMethodShowDepth;

//---------------------------------------------------------------------------

typedef CObj Object;

//---------------------------------------------------------------------------

class Character : public Object
    {
    private:
    static SClassRef sClass;

    protected:
    char myChar;

    public:

    Character ( char i = 0);
    virtual ~Character() { }

    char rChar() const { return myChar; }
    char rCharacter() const { return myChar; }
```

```
    static bool selfReg();
    static SClassRef rClass() { return sClass; }

    virtual SParam asLowercase( const CParams& params); //0 parameter, return SParam::eChar
    virtual SParam asCharacter ( const CParams& params); //0, ret eChar

    /*
    =
    asLowercase
    asString
    asUppercase
    codePoint
    isAlphaNumeric
    isDigit
    isLetter
    isLowercase
    isUppercase

    */
    };

class Boolean : public Object
    {
    private:
    static SClassRef sClass;

    protected:
    bool myBool;

    public:
    static SClassRef rClass() { return sClass; }

    Boolean ( bool b);

    static bool selfReg();

    bool rBool() const { return myBool; }

    SParam isTrue ( const CParams& par); // 0
    SParam isFalse ( const CParams& par); // 0

    SParam ifTrue ( const CParams& par); // 1 - block
    SParam ifFalse ( const CParams& par); // 1 - block

    SParam and ( const CParams& par);
    SParam printString ( const CParams& par);

/*
    &
    |
    and:
    eqv:
    ifFalse:ifTrue:
    ifTrue:ifFalse:
    not
    or:
    printString
    xor:
*/
    };

class Nil : public Object
    {
    private:
    static SClassRef sClass;

    protected:
    bool myBool;

    public:
    static SClassRef rClass() { return sClass; }

    Nil();

    static bool selfReg();

    SParam isEqual ( const CParams& par); // 1 - eAny
    SParam printString ( const CParams& par);
    };

class Instantiator : public Object
    {
    private:
    static SClassRef sClass;

    public:

    Instantiator();
    virtual ~Instantiator() { }

    static bool selfReg();

    virtual SParam sNew ( const CParams& params) const = 0;
    };

#endif
```

Файл SmallVM\Standard Smalltalk Library\Numeric Protocols.h

```
#if !defined (NUMERIC_PROTOCOLS)
#define NUMERIC_PROTOCOLS

#include "..\system.h"
#include "Fundamental Protocols.h"

//---------------------------------------------------------------------------
/*
Implementation specifications:
    none yet :)
*/
//---------------------------------------------------------------------------

bool NumericProtocolsReg();

//---------------------------------------------------------------------------

class Magnitude : public Object
    {
    private:
    static SClassRef sClass;

    public:

    Magnitude();
    virtual ~Magnitude() { }

    static bool selfReg();

    virtual SParam operator< ( const CParams& params) = 0; //1 parameter, return new bool
    virtual SParam operator<= ( const CParams& params) = 0; //1 parameter, return new bool
    virtual SParam operator> ( const CParams& params) = 0; //1 parameter, return new bool
    virtual SParam operator>= ( const CParams& params) = 0; //1 parameter, return new bool
    virtual /*eBool*/SParam operator== ( const CParams& params) = 0; // 1: 0 - some number, ret new bool
    virtual SParam betweenAnd ( const CParams& params) = 0; //2 parameters, return new bool
    virtual SParam sMax ( const CParams& params)  //1 parameter, return new bool
        { return operator> ( params); }
    virtual SParam sMin ( const CParams& params) //1 parameter, return new bool
        { return operator< ( params); }
    };

//---------------------------------------------------------------------------

class Number : public Magnitude
    {
    private:
    static SClassRef sClass;

    public:

    Number();
    virtual ~Number() { }

    static bool selfReg();

    // this funcs are not virtual because they check all variants of this and param, but may be overloaded
in system later
    virtual SParam operator< ( const CParams& params) = 0; //1 parameter, return new bool
    virtual SParam operator<= ( const CParams& params) = 0; //1 parameter, return new bool
    virtual SParam operator> ( const CParams& params) = 0; //1 parameter, return new bool
    virtual SParam operator>= ( const CParams& params) = 0; //1 parameter, return new bool
    virtual /*eBool*/SParam operator== ( const CParams& params) = 0; // 1: 0 - some number, ret new bool
    virtual SParam betweenAnd ( const CParams& params) = 0; //2 parameters, return new bool

    virtual SParam operator* ( const CParams& params) = 0; //1 param, return new
    virtual SParam operator- ( const CParams& params) = 0; //1 param, return new
    virtual SParam operator+ ( const CParams& params) = 0; //1 param, return new
    virtual SParam operator/ ( const CParams& params) = 0; //1 param, return new
    virtual SParam abs ( const CParams& params) = 0; //0 params, return new object with type of param
    virtual SParam /*eInt*/asInteger ( const CParams&) = 0; //0
    virtual SParam /*eDouble*/asFloat ( const CParams&) = 0; //0
    virtual SParam /*SString*/ asString ( const CParams&) = 0; //0 params, return new object with type of
param

    /*

//  *
//  +
//  - operand
//  /
    //
//  <
    =
//  >
    \\
//  abs
    asFloat
    asFloatD
    asFloatE
    asFloatQ
    asFraction
```

```
//  asInteger
    asScaledDecimal:
    ceiling
    floor
    fractionPart
    integerPart
    negated
    negative
    positive
    quo:
    raisedTo:
    raisedToInteger:
    reciprocal
    rem:
    rounded
    roundTo:
    sign
    sqrt
    squared
    strictlyPositive
    to:
    to:by:
    to:by:do:
    to:do:
    truncated
    truncateTo:

    */
    };

//---------------------------------------------------------------------------------------------------

class Rational : public Number
    {
    private:
    static SClassRef sClass;

    public:

    Rational();
    virtual ~Rational() { }

    static bool selfReg();

    virtual SParam denominator ( const CParams& params); //0 params, return new
    virtual SParam numerator ( const CParams& params); //0 params, return new

    };

class Integer : public Rational
    {
    private:
    static SClassRef sClass;

    protected:
    long myInteger;

    public:

    Integer();
    Integer ( long i);
    virtual ~Integer() { }

    static SParam sConstruct ( const CParams& par) //1 params: int, return new
        { return new Integer ( par[0].rInt()); }

    long rInteger() const { return myInteger; }

    static bool selfReg();
    static SClassRef rClass() { return sClass; }

    virtual SParam operator< ( const CParams& params); //1 parameter, return new bool
    virtual SParam operator<= ( const CParams& params); //1 parameter, return new bool
    virtual SParam operator> ( const CParams& params); //1 parameter, return new bool
    virtual SParam operator>= ( const CParams& params); //1 parameter, return new bool
    virtual /*eBool*/SParam operator== ( const CParams& params); // 1: 0 - some number, ret new bool
    virtual SParam betweenAnd ( const CParams& params); //2 parameters, return new bool

    virtual SParam operator* ( const CParams& params); //1 param, return new
    virtual SParam operator- ( const CParams& params); //1 param, return new
    virtual SParam operator+ ( const CParams& params); //1 param, return new
    virtual SParam operator/ ( const CParams& params); //1 param, return new
    virtual SParam abs ( const CParams&); //0 params, return new object with type of param
    virtual SParam /*eInt*/asInteger ( const CParams&); //0
    virtual SParam /*eDouble*/asFloat ( const CParams&); //0
    virtual SParam /*SString*/ asString ( const CParams&); //0

    //my own funcs
    virtual SParam repeat ( const CParams& par); // 1 - block
    };

class Float : public Rational
    {
    private:
    static SClassRef sClass;

    protected:
```

```
    double myFloat;

    public:

    Float();
    Float ( double d);
    virtual ~Float() { }

    static SParam sConstruct ( const CParams& par) //1 params: int, return new
        { return new Float ( par[0].rDouble()); }

    long rInteger() const { return myFloat; }
    double rFloat() const { return myFloat; }

    static bool selfReg();
    static SClassRef rClass() { return sClass; }

    virtual SParam operator< ( const CParams& params); //1 parameter, return new bool
    virtual SParam operator<= ( const CParams& params); //1 parameter, return new bool
    virtual SParam operator> ( const CParams& params); //1 parameter, return new bool
    virtual SParam operator>= ( const CParams& params); //1 parameter, return new bool
    virtual /*eBool*/SParam operator== ( const CParams& params); // 1: 0 - some number, ret new bool
    virtual SParam betweenAnd ( const CParams& params); //2 parameters, return new bool

    virtual SParam operator* ( const CParams& params); //1 param, return new
    virtual SParam operator- ( const CParams& params); //1 param, return new
    virtual SParam operator+ ( const CParams& params); //1 param, return new
    virtual SParam operator/ ( const CParams& params); //1 param, return new
    virtual SParam abs ( const CParams&); //0 params, return new object with type of param
    virtual SParam /*eInt*/asInteger ( const CParams&); //0
    virtual SParam /*eDouble*/asFloat ( const CParams&); //0
    virtual SParam /*SString*/ asString ( const CParams&); //0 params, return new object with type of
param

    };

#endif //NUMERIC_PROTOCOLS
```

Файл SmallVM\Standard Smalltalk Library\Stream Protocols.h

```
#if !defined (STREAM_PROTOCOLS_FILE_LOADED)
#define STREAM_PROTOCOLS_FILE_LOADED

#include "..\system.h"
#include "Fundamental protocols.h"

//---------------------------------------------------------------------------

bool StreamProtocolsReg();

//---------------------------------------------------------------------------

class PuttableStream : public Object
    {
    private:
    static SClassRef sClass;

    public:

    PuttableStream();
    virtual ~PuttableStream();

    static bool selfReg();

    virtual SParam cr ( const CParams&) = 0;
    virtual SParam flush ( const CParams&) = 0;
    virtual SParam space ( const CParams&) = 0;
    virtual SParam tab ( const CParams&) = 0;
    virtual SParam nextPut ( const CParams& param) = 0; //object
    virtual SParam nextPutAll ( const CParams& param) = 0; //collection
    };

//---------------------------------------------------------------------------

class Transcript : public PuttableStream
    {
    private:
    static SClassRef sClass;

    public:

    Transcript();
    ~Transcript();

    static bool selfReg();

    SParam cr ( const CParams&);
    SParam flush ( const CParams&);
    SParam space ( const CParams&);
    SParam tab ( const CParams&);
    SParam nextPut ( const CParams& param); //object
    SParam nextPutAll ( const CParams& param); //collection
```

```
    };
//---------------------------------------------------------------------------
#endif
```

Файл Sound\Commands.h

```
// Commands.h - header
// created 06.11.2002 by header creator
#pragma once
#ifndef __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__

#include "system.h"

void AddCommands (IConsole* console);

#endif //__COMMANDS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Sound\Sound.h

```
// Sound.h - header
// created 06.11.2002 by header creator
#pragma once
#ifndef __SOUND_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SOUND_HEADER_FILE_BY_VASFED_INCLUDED__

#include <objbase.h>

// {1AB69489-5A73-44cb-88DB-CC30526601FF}
static const GUID IID_ISoundEngine =
{ 0x1ab69489, 0x5a73, 0x44cb, { 0x88, 0xdb, 0xcc, 0x30, 0x52, 0x66, 0x1, 0xff } };

interface ISoundEngine: public IUnknown
{
    virtual void Reset     (void) PURE;
    virtual void Init      (int startsize) PURE;
    virtual void Terminate (void) PURE;

    virtual void   LoadResource       (char* filename)   PURE;
    virtual void   RemoveSound        (int index)        PURE;
    virtual int    FindSound          (char* name)       PURE;
    virtual int    AddSound           (char* filename, char* name) PURE;
    virtual int    AddSoundFromResource(void* reserved)   PURE;

    virtual void Play  (int index) PURE;
    virtual void Play  (char*name) PURE;
    virtual void Stop  (int index) PURE;
    virtual void Stop  (char*name) PURE;
};

#endif //__SOUND_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Sound\SoundCore.h

```
// SoundCore.h - header
// created 06.11.2002 by header creator
#pragma once
#ifndef __SOUNDCORE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SOUNDCORE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "Sound.h"
#include "..\Common\Utilities\Module.h"
#include "DMUtil.h"

struct SSound
{
    char name[128];
    CMusicSegment* segment;
};

class CSoundEngine: public ISoundEngine
{
    public:
        CSoundEngine ();
        virtual ~CSoundEngine ();

        void Terminate ();
        void Reset  ();
        void Init   (int startsize);

        void    LoadResource           (char* filename);
        void    RemoveSound            (int index);
```

```
        int     FindSound               (char* name);
        int     AddSound                (char* filename, char* name = NULL);
        int     AddSoundFromResource    (void* reserved);

        void Play (int index);
        void Play (char*name);
        void Stop (int index);
        void Stop (char*name);


        void Grow       (int a = 0);
        void GrowFor    (int n = 1);
        void GrowTo     (int n);

        int FindFreeSpace ();

        DEFINE_IUNKNOWN ();

    //protected:
        CMusicManager*  m_musicManager;
        SSound**        m_sounds;
        int             m_numSounds;
        int             m_soundsSize;

        bool            m_initialized;
};

#endif //__SOUNDCORE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Sound\System.h

```
// System.h - header
// created 06.11.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\Loader\Loader.h"
#include "SoundCore.h"
#include "..\Console\Console.h"

extern ILoader* Loader;
extern CSoundEngine* Sound;
extern IConsole* Console;

#endif //__SYSTEM_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Sound\dmutil.h

```
//-----------------------------------------------------------------------------
// File: DMUtil.h
//
// Desc:
//
// Copyright (c) 1999-2001 Microsoft Corp. All rights reserved.
//-----------------------------------------------------------------------------
#ifndef DMUTIL_H
#define DMUTIL_H

#include <dmusicc.h>
#include <dmusici.h>
#include <dsound.h>

//-----------------------------------------------------------------------------
// Classes used by this header
//-----------------------------------------------------------------------------
class CMusicManager;
class CMusicSegment;
class CMusicScript;

//-----------------------------------------------------------------------------
// Name: class CMusicManager
// Desc:
//-----------------------------------------------------------------------------
class CMusicManager
{
protected:
    BOOL                        m_bCleanupCOM;
    IDirectMusicLoader8*        m_pLoader;
    IDirectMusicPerformance8*   m_pPerformance;

public:
    CMusicManager();
    ~CMusicManager();

    inline IDirectMusicLoader8*      GetLoader()      { return m_pLoader; }
    inline IDirectMusicPerformance8* GetPerformance() { return m_pPerformance; }
    IDirectMusicAudioPath8* GetDefaultAudioPath();
```

```
    HRESULT Initialize( HWND hWnd, DWORD dwPChannels = 128, DWORD dwDefaultPathType =
DMUS_APATH_DYNAMIC_STEREO );

    HRESULT SetSearchDirectory( const TCHAR* strMediaPath );
    VOID    CollectGarbage();

    HRESULT CreateSegmentFromFile( CMusicSegment** ppSegment, TCHAR* strFileName,
                                   BOOL bDownloadNow = TRUE, BOOL bIsMidiFile = FALSE );
    HRESULT CreateScriptFromFile( CMusicScript** ppScript, TCHAR* strFileName );

    HRESULT CreateChordMapFromFile( IDirectMusicChordMap8** ppChordMap, TCHAR* strFileName );
    HRESULT CreateStyleFromFile( IDirectMusicStyle8** ppStyle, TCHAR* strFileName );
    HRESULT GetMotifFromStyle( IDirectMusicSegment8** ppMotif, TCHAR* strStyle, TCHAR* wstrMotif );

    HRESULT CreateSegmentFromResource( CMusicSegment** ppSegment, TCHAR* strResource, TCHAR*
strResourceType,
                                   BOOL bDownloadNow = TRUE, BOOL bIsMidiFile = FALSE );
};

//-----------------------------------------------------------------------------
// Name: class CMusicSegment
// Desc: Encapsulates functionality of an IDirectMusicSegment
//-----------------------------------------------------------------------------
class CMusicSegment
{
protected:
    IDirectMusicSegment8*     m_pSegment;
    IDirectMusicLoader8*      m_pLoader;
    IDirectMusicPerformance8* m_pPerformance;
    IDirectMusicAudioPath8*   m_pEmbeddedAudioPath;
    BOOL                      m_bDownloaded;

public:
    CMusicSegment( IDirectMusicPerformance8* pPerformance,
                   IDirectMusicLoader8* pLoader,
                   IDirectMusicSegment8* pSegment );
    virtual ~CMusicSegment();

    inline  IDirectMusicSegment8* GetSegment() { return m_pSegment; }
    HRESULT GetStyle( IDirectMusicStyle8** ppStyle, DWORD dwStyleIndex = 0 );

    HRESULT SetRepeats( DWORD dwRepeats );
    HRESULT Play( DWORD dwFlags = DMUS_SEGF_SECONDARY, IDirectMusicAudioPath8* pAudioPath = NULL );
    HRESULT Stop( DWORD dwFlags = 0 );
    HRESULT Download( IDirectMusicAudioPath8* pAudioPath = NULL );
    HRESULT Unload( IDirectMusicAudioPath8* pAudioPath = NULL );

    BOOL    IsPlaying();
};

//-----------------------------------------------------------------------------
// Name: class CMusicScript
// Desc: Encapsulates functionality of an IDirectMusicScript
//-----------------------------------------------------------------------------
class CMusicScript
{
protected:
    IDirectMusicScript8*      m_pScript;
    IDirectMusicLoader8*      m_pLoader;
    IDirectMusicPerformance8* m_pPerformance;

public:
    CMusicScript( IDirectMusicPerformance8* pPerformance,
                  IDirectMusicLoader8* pLoader,
                  IDirectMusicScript8* pScript );
    virtual ~CMusicScript();

    inline  IDirectMusicScript8* GetScript() { return m_pScript; }

    HRESULT CallRoutine( TCHAR* strRoutine );
    HRESULT SetVariableNumber( TCHAR* strVariable, LONG lValue );
    HRESULT GetVariableNumber( TCHAR* strVariable, LONG* plValue );
};

#endif // DMUTIL_H
```

Файл Sound\dsutil.h

```
//-----------------------------------------------------------------------------
// File: DSUtil.h
//
// Desc:
//
// Copyright (c) 1999-2001 Microsoft Corp. All rights reserved.
//-----------------------------------------------------------------------------
#ifndef DSUTIL_H
#define DSUTIL_H

#include <windows.h>
#include <mmsystem.h>
```

```
#include <mmreg.h>
#include <dsound.h>

//-----------------------------------------------------------------------------
// Classes used by this header
//-----------------------------------------------------------------------------
class CSoundManager;
class CSound;
class CStreamingSound;
class CWaveFile;

//-----------------------------------------------------------------------------
// Typing macros
//-----------------------------------------------------------------------------
#define WAVEFILE_READ   1
#define WAVEFILE_WRITE  2

#define DSUtil_StopSound(s)         { if(s) s->Stop(); }
#define DSUtil_PlaySound(s)         { if(s) s->Play( 0, 0 ); }
#define DSUtil_PlaySoundLooping(s)  { if(s) s->Play( 0, DSBPLAY_LOOPING ); }

//-----------------------------------------------------------------------------
// Name: class CSoundManager
// Desc:
//-----------------------------------------------------------------------------
class CSoundManager
{
protected:
    LPDIRECTSOUND8 m_pDS;

public:
    CSoundManager();
    ~CSoundManager();

    HRESULT Initialize( HWND hWnd, DWORD dwCoopLevel, DWORD dwPrimaryChannels, DWORD dwPrimaryFreq, DWORD
dwPrimaryBitRate );
    inline  LPDIRECTSOUND8 GetDirectSound() { return m_pDS; }
    HRESULT SetPrimaryBufferFormat( DWORD dwPrimaryChannels, DWORD dwPrimaryFreq, DWORD dwPrimaryBitRate
);
    HRESULT Get3DListenerInterface( LPDIRECTSOUND3DLISTENER* ppDSListener );

    HRESULT Create( CSound** ppSound, LPTSTR strWaveFileName, DWORD dwCreationFlags = 0, GUID
guid3DAlgorithm = GUID_NULL, DWORD dwNumBuffers = 1 );
    HRESULT CreateFromMemory( CSound** ppSound, BYTE* pbData, ULONG ulDataSize, LPWAVEFORMATEX pwfx, DWORD
dwCreationFlags = 0, GUID guid3DAlgorithm = GUID_NULL, DWORD dwNumBuffers = 1 );
    HRESULT CreateStreaming( CStreamingSound** ppStreamingSound, LPTSTR strWaveFileName, DWORD
dwCreationFlags, GUID guid3DAlgorithm, DWORD dwNotifyCount, DWORD dwNotifySize, HANDLE hNotifyEvent );
};

//-----------------------------------------------------------------------------
// Name: class CSound
// Desc: Encapsulates functionality of a DirectSound buffer.
//-----------------------------------------------------------------------------
class CSound
{
protected:
    LPDIRECTSOUNDBUFFER* m_apDSBuffer;
    DWORD                m_dwDSBufferSize;
    CWaveFile*           m_pWaveFile;
    DWORD                m_dwNumBuffers;

    HRESULT RestoreBuffer( LPDIRECTSOUNDBUFFER pDSB, BOOL* pbWasRestored );

public:
    CSound( LPDIRECTSOUNDBUFFER* apDSBuffer, DWORD dwDSBufferSize, DWORD dwNumBuffers, CWaveFile*
pWaveFile );
    virtual ~CSound();

    HRESULT Get3DBufferInterface( DWORD dwIndex, LPDIRECTSOUND3DBUFFER* ppDS3DBuffer );
    HRESULT FillBufferWithSound( LPDIRECTSOUNDBUFFER pDSB, BOOL bRepeatWavIfBufferLarger );
    LPDIRECTSOUNDBUFFER GetFreeBuffer();
    LPDIRECTSOUNDBUFFER GetBuffer( DWORD dwIndex );

    HRESULT Play( DWORD dwPriority = 0, DWORD dwFlags = 0 );
    HRESULT Stop();
    HRESULT Reset();
    BOOL    IsSoundPlaying();
};

//-----------------------------------------------------------------------------
// Name: class CStreamingSound
// Desc: Encapsulates functionality to play a wave file with DirectSound.
//       The Create() method loads a chunk of wave file into the buffer,
//       and as sound plays more is written to the buffer by calling
//       HandleWaveStreamNotification() whenever hNotifyEvent is signaled.
//-----------------------------------------------------------------------------
class CStreamingSound : public CSound
{
protected:
    DWORD m_dwLastPlayPos;
    DWORD m_dwPlayProgress;
    DWORD m_dwNotifySize;
    DWORD m_dwNextWriteOffset;
    BOOL  m_bFillNextNotificationWithSilence;

public:
```

```
    CStreamingSound( LPDIRECTSOUNDBUFFER pDSBuffer, DWORD dwDSBufferSize, CWaveFile* pWaveFile, DWORD
dwNotifySize );
    ~CStreamingSound();

    HRESULT HandleWaveStreamNotification( BOOL bLoopedPlay );
    HRESULT Reset();
};

//-----------------------------------------------------------------------------
// Name: class CWaveFile
// Desc: Encapsulates reading or writing sound data to or from a wave file
//-----------------------------------------------------------------------------
class CWaveFile
{
public:
    WAVEFORMATEX* m_pwfx;        // Pointer to WAVEFORMATEX structure
    HMMIO         m_hmmio;       // MM I/O handle for the WAVE
    MMCKINFO      m_ck;          // Multimedia RIFF chunk
    MMCKINFO      m_ckRiff;      // Use in opening a WAVE file
    DWORD         m_dwSize;      // The size of the wave file
    MMIOINFO      m_mmioinfoOut;
    DWORD         m_dwFlags;
    BOOL          m_bIsReadingFromMemory;
    BYTE*         m_pbData;
    BYTE*         m_pbDataCur;
    ULONG         m_ulDataSize;
    CHAR*         m_pResourceBuffer;

protected:
    HRESULT ReadMMIO();
    HRESULT WriteMMIO( WAVEFORMATEX *pwfxDest );

public:
    CWaveFile();
    ~CWaveFile();

    HRESULT Open( LPTSTR strFileName, WAVEFORMATEX* pwfx, DWORD dwFlags );
    HRESULT OpenFromMemory( BYTE* pbData, ULONG ulDataSize, WAVEFORMATEX* pwfx, DWORD dwFlags );
    HRESULT Close();

    HRESULT Read( BYTE* pBuffer, DWORD dwSizeToRead, DWORD* pdwSizeRead );
    HRESULT Write( UINT nSizeToWrite, BYTE* pbData, UINT* pnSizeWrote );

    DWORD   GetSize();
    HRESULT ResetFile();
    WAVEFORMATEX* GetFormat() { return m_pwfx; };
};

#endif // DSUTIL_H
```

Файл Sound\dxutil.h

```
//-----------------------------------------------------------------------------
// File: DXUtil.h
//
// Desc: Helper functions and typing shortcuts for DirectX programming.
//
// Copyright (c) 1997-2001 Microsoft Corporation. All rights reserved
//-----------------------------------------------------------------------------
#ifndef DXUTIL_H
#define DXUTIL_H

//-----------------------------------------------------------------------------
// Miscellaneous helper functions
//-----------------------------------------------------------------------------
#define SAFE_DELETE(p)       { if(p) { delete (p);     (p)=NULL; } }
#define SAFE_DELETE_ARRAY(p) { if(p) { delete[] (p);   (p)=NULL; } }
#define SAFE_RELEASE(p)      { if(p) { (p)->Release(); (p)=NULL; } }

//-----------------------------------------------------------------------------
// Name: DXUtil_GetDXSDKMediaPath() and DXUtil_FindMediaFile()
// Desc: Returns the DirectX SDK path, as stored in the system registry
//       during the SDK install.
//-----------------------------------------------------------------------------
const TCHAR* DXUtil_GetDXSDKMediaPath();
HRESULT      DXUtil_FindMediaFile( TCHAR* strPath, TCHAR* strFilename );

//-----------------------------------------------------------------------------
// Name: DXUtil_Read*RegKey() and DXUtil_Write*RegKey()
// Desc: Helper functions to read/write a string registry key
//-----------------------------------------------------------------------------
HRESULT DXUtil_WriteStringRegKey( HKEY hKey, TCHAR* strRegName, TCHAR* strValue );
HRESULT DXUtil_WriteIntRegKey( HKEY hKey, TCHAR* strRegName, DWORD dwValue );
HRESULT DXUtil_WriteGuidRegKey( HKEY hKey, TCHAR* strRegName, GUID guidValue );
HRESULT DXUtil_WriteBoolRegKey( HKEY hKey, TCHAR* strRegName, BOOL bValue );

HRESULT DXUtil_ReadStringRegKey( HKEY hKey, TCHAR* strRegName, TCHAR* strValue, DWORD dwLength, TCHAR*
strDefault );
HRESULT DXUtil_ReadIntRegKey( HKEY hKey, TCHAR* strRegName, DWORD* pdwValue, DWORD dwDefault );
HRESULT DXUtil_ReadGuidRegKey( HKEY hKey, TCHAR* strRegName, GUID* pGuidValue, GUID& guidDefault );
HRESULT DXUtil_ReadBoolRegKey( HKEY hKey, TCHAR* strRegName, BOOL* pbValue, BOOL bDefault );
```

```
//-----------------------------------------------------------------------------
// Name: DXUtil_Timer()
// Desc: Performs timer opertations. Use the following commands:
//          TIMER_RESET           - to reset the timer
//          TIMER_START           - to start the timer
//          TIMER_STOP            - to stop (or pause) the timer
//          TIMER_ADVANCE         - to advance the timer by 0.1 seconds
//          TIMER_GETABSOLUTETIME - to get the absolute system time
//          TIMER_GETAPPTIME      - to get the current time
//          TIMER_GETELAPSEDTIME  - to get the time that elapsed between
//                                  TIMER_GETELAPSEDTIME calls
//-----------------------------------------------------------------------------
enum TIMER_COMMAND { TIMER_RESET, TIMER_START, TIMER_STOP, TIMER_ADVANCE,
                     TIMER_GETABSOLUTETIME, TIMER_GETAPPTIME, TIMER_GETELAPSEDTIME };
FLOAT __stdcall DXUtil_Timer( TIMER_COMMAND command );

//-----------------------------------------------------------------------------
// UNICODE support for converting between CHAR, TCHAR, and WCHAR strings
//-----------------------------------------------------------------------------
VOID DXUtil_ConvertAnsiStringToWide( WCHAR* wstrDestination, const CHAR* strSource, int cchDestChar = -1
);
VOID DXUtil_ConvertWideStringToAnsi( CHAR* strDestination, const WCHAR* wstrSource, int cchDestChar = -1
);
VOID DXUtil_ConvertGenericStringToAnsi( CHAR* strDestination, const TCHAR* tstrSource, int cchDestChar = -
1 );
VOID DXUtil_ConvertGenericStringToWide( WCHAR* wstrDestination, const TCHAR* tstrSource, int cchDestChar =
-1 );
VOID DXUtil_ConvertAnsiStringToGeneric( TCHAR* tstrDestination, const CHAR* strSource, int cchDestChar = -
1 );
VOID DXUtil_ConvertWideStringToGeneric( TCHAR* tstrDestination, const WCHAR* wstrSource, int cchDestChar =
-1 );

//-----------------------------------------------------------------------------
// GUID to String converting
//-----------------------------------------------------------------------------
VOID DXUtil_ConvertGUIDToString( const GUID* pGuidIn, TCHAR* strOut );
BOOL DXUtil_ConvertStringToGUID( const TCHAR* strIn, GUID* pGuidOut );

//-----------------------------------------------------------------------------
// Debug printing support
//-----------------------------------------------------------------------------
VOID    DXUtil_Trace( TCHAR* strMsg, ... );
HRESULT _DbgOut( TCHAR*, DWORD, HRESULT, TCHAR* );

#if defined(DEBUG) | defined(_DEBUG)
    #define DXTRACE           DXUtil_Trace
#else
    #define DXTRACE           sizeof
#endif

#if defined(DEBUG) | defined(_DEBUG)
    #define DEBUG_MSG(str)    _DbgOut( __FILE__, (DWORD)__LINE__, 0, str )
#else
    #define DEBUG_MSG(str)    (0L)
#endif

#endif // DXUTIL_H
```

Файл Sound\resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Sound.rc
//
#define IDR_WAV                         103

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        104
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Файл StorageMaster\objstock.h

```
/*****************************************************************************/
/*****************************************************************************/
/***                                                                       ***/
/*** OBJSTOCK.H                                                            ***/
/*** ОБЪЯВЛЕНИЕ КЛАССА CSTORAGEMASTER И НЕКОТОРЫХ ВСПОМОГАТЕЛЬНЫХ           ***/
/*** КЛАССОВ. ЯВЛЯЕТСЯ ДИНАМИЧЕСКИ ПОДКЛЮЧАЕМОЙ БИБЛИОТЕКОЙ И МОДУЛЕМ       ***/
```

```
/*** СИСТЕМЫ, ЗАГРУЖАЕМЫМ С ПОМОЩЬЮ LOADER'A.                              ***/
/*** ЭТОТ МОДУЛЬ ОПИРАЕТСЯ НА МОДУЛЬ MEMORY, В КОТОРОМ ОПИСАНЫ БАЗОВЫЕ     ***/
/*** КЛАССЫ ДЛЯ РАБОТЫ С ПАМЯТЬЮ                                          ***/
/***                                                                      ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ             ***/
/***                                                                      ***/
/***                                                                      ***/
/****************************************************************************/
/****************************************************************************/

#include "stockint.h"

#ifdef OBJSTOCK_EXPORTS
#define OBJSTOCK_API __declspec(dllexport)
#else
#define OBJSTOCK_API __declspec(dllimport)
#endif

#ifndef _OBJSTOCK_H_
#define _OBJSTOCK_H_

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////  CLASS CDATABLOCK  ////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

class CDataBlock : public CMemoryBlock
{
private:
int (* Ok )(char *);
int (* Dump )(char *, FILE *);
int (* Save )(char *, FILE *);
int (* Load )(char *, FILE *);

/****************************************************************************/

public:
CDataBlock  ();
~CDataBlock ();

void   Init           (int size, int num);
void   Init           (int size, int num, int (* ok )(char *), int (* dump )(char *, FILE *));
int    RetMax         ();
int    RetObjSize     ();
void   InitFuncs      (int (* ok)(char *), int (* dump)(char *, FILE *),
                       int (* save)(char *, FILE *), int (* load)(char *, FILE *));
int    SaveData       (FILE * file);
int    LoadData       (FILE * file);
int    DumpData       (FILE * file);
int    TestData       ();
};

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////  CLASS CHANDLETABLE  ////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

class CHandleTable
{
private:
char ** myNames;
int *   myHandles;
int     myCurNum, myMaxNum;

/****************************************************************************/

public:
CHandleTable  ();
CHandleTable  (int num);
~CHandleTable ();

void   Init      (int num);
int    ok        ();
int    ok        (int n);
void   Add       (int handle, char * name);
int    RetHandle (char * name);
char * RetName   (int handle);
};

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////  CLASS CSTORAGEMANAGER  ////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

class CStorageMaster : public IStorageMaster
{
private:
CDataBlock    * myData;
CSmartPointer * myPointers;
int             myCurBlock, myMaxBlocks;
CHandleTable    myTable;

bool myFor3DOptimized;
```

```
long myRefs;

/*************************************************************************/

public:
CStorageMaster  ();
CStorageMaster  (int maxblocks);
~CStorageMaster ();

virtual HRESULT         __stdcall QueryInterface (const IID &iid, void ** obj);
virtual ULONG           __stdcall AddRef ();
virtual ULONG           __stdcall Release ();

virtual void            __stdcall Init            (int maxblocks);
virtual int             __stdcall ok              ();
virtual int             __stdcall ok              (int n);
virtual int             __stdcall SetNewDataType  (int size, int num, char * name);
virtual void            __stdcall InitFuncs       (int handle, int (* ok)(char *), int (* dump)(char *,
FILE *),
                                                  int (* save)(char *, FILE *), int (* load)(char *, FILE
*));
virtual void            __stdcall RemoveDataType  (int handle);
virtual int             __stdcall AddObject       (int handle, void * obj);
virtual CSmartPointer & __stdcall RetData         (int handle);
virtual char *          __stdcall RetObject       (int handle, int nobj);
virtual void            __stdcall RemoveObject    (int handle, int n);
virtual char *          __stdcall CreateObject    (int handle);
virtual char *          __stdcall CreateObjects   (int handle, int num);
virtual int             __stdcall RetMaxObjects   (int handle);
virtual void            __stdcall Lock            (int handle);
virtual void            __stdcall Unlock          (int handle);
virtual int             __stdcall DumpData        (int handle, char * fname);
virtual int             __stdcall SaveData        (int handle, char * fname);
virtual int             __stdcall LoadData        (int handle, char * fname);
virtual int             __stdcall TestData        (int handle);
virtual int             __stdcall SaveXXX         (int handle, char * filename, char type);
virtual int             __stdcall LoadXXX         (char * filename);

virtual int             __stdcall RetHandle       (char * name);
virtual void            __stdcall InitFuncs       (char * name, int (* ok)(char *), int (* dump)(char *,
FILE *),
                                                  int (* save)(char *, FILE *), int (* load)(char *, FILE
*));
virtual void            __stdcall RemoveDataType  (char * name);
virtual int             __stdcall AddObject       (char * name, void * obj);
virtual CSmartPointer & __stdcall RetData         (char * name);
virtual char *          __stdcall RetObject       (char * name, int nobj);
virtual void            __stdcall RemoveObject    (char * name, int n);
virtual char *          __stdcall CreateObject    (char * name);
virtual char *          __stdcall CreateObjects   (char * name, int num);
virtual int             __stdcall RetMaxObjects   (char * name);
virtual void            __stdcall Lock            (char * name);
virtual void            __stdcall Unlock          (char * name);
virtual int             __stdcall DumpData        (char * name, char * fname);
virtual int             __stdcall SaveData        (char * name, char * fname);
virtual int             __stdcall LoadData        (char * name, char * fname);
virtual int             __stdcall TestData        (char * name);
virtual int             __stdcall SaveXXX         (char * name, char * filename, char type);
};

extern "C" void OBJSTOCK_API GetInstance (void ** ptr);

extern IConsole * Console;

#endif
```

Файл StorageMaster\stockint.h

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** STOCKINT.H                                                        ***/
/*** ОПИСАНИЕ ИНТЕРФЕЙСА МОДУЛЯ ХРАНЕНИЯ ДАННЫХ.                       ***/
/***                                                                   ***/
/*** ФАЙЛ НАПИСАН ТАТАРИНОВЫМ АНДРЕЕМ СЕРГЕЕВИЧЕМ В 2002 ГОДУ          ***/
/***                                                                   ***/
/***                                                                   ***/
/*************************************************************************/
/*************************************************************************/

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include "..\memory\memory.h"
#include "..\console\console.h"
#include <objbase.h>

#ifdef STOCKINT_EXPORTS
#define STOCKINT_API __declspec(dllexport)
#else
#define STOCKINT_API __declspec(dllimport)
#endif
```

```
#ifndef _STOCKINT_H_
#define _STOCKINT_H_

    //////////////////////////////////////////////////////////////////\\
  //////////////////// INTERFACE ISTORAGEMASTER ////////////////////////\\
//////////////////////////////////////////////////////////////////////////\\*/

// {246A6CB3-9441-4ded-AC72-36DF456711B7}
const IID IID_IStorageMaster = {0x246a6cb3, 0x9441, 0x4ded,
                                {0xac, 0x72, 0x36, 0xdf, 0x45, 0x67, 0x11, 0xb7}};

interface IStorageMaster : IUnknown
{
virtual void            __stdcall Init            (int maxblocks) = 0;
virtual int             __stdcall SetNewDataType  (int size, int num, char * name) = 0;
virtual void            __stdcall InitFuncs       (int handle, int (* ok)(char *), int (* dump)(char *,
FILE *),
                                                   int (* save)(char *, FILE *), int (* load)(char *, FILE
*)) = 0;
virtual void            __stdcall RemoveDataType  (int handle) = 0;
virtual int             __stdcall AddObject       (int handle, void * obj) = 0;
virtual CSmartPointer & __stdcall RetData         (int handle) = 0;
virtual char *          __stdcall RetObject       (int handle, int nobj) = 0;
virtual void            __stdcall RemoveObject    (int handle, int nobj) = 0;
virtual char *          __stdcall CreateObject    (int handle) = 0;
virtual char *          __stdcall CreateObjects   (int handle, int num) = 0;
virtual int             __stdcall RetMaxObjects   (int handle) = 0;
virtual void            __stdcall Lock            (int handle) = 0;
virtual void            __stdcall Unlock          (int handle) = 0;
virtual int             __stdcall DumpData        (int handle, char * fname) = 0;
virtual int             __stdcall SaveData        (int handle, char * fname) = 0;
virtual int             __stdcall LoadData        (int handle, char * fname) = 0;
virtual int             __stdcall TestData        (int handle) = 0;
virtual int             __stdcall SaveXXX         (int handle, char * filename, char type) = 0;
virtual int             __stdcall LoadXXX         (char * filename) = 0;

virtual int             __stdcall RetHandle       (char * name) = 0;
virtual void            __stdcall InitFuncs       (char * name, int (* ok)(char *), int (* dump)(char *,
FILE *),
                                                   int (* save)(char *, FILE *), int (* load)(char *, FILE
*)) = 0;
virtual void            __stdcall RemoveDataType  (char * name) = 0;
virtual int             __stdcall AddObject       (char * name, void * obj) = 0;
virtual CSmartPointer & __stdcall RetData         (char * name) = 0;
virtual void            __stdcall RemoveObject    (char * name, int n) = 0;
virtual char *          __stdcall RetObject       (char * name, int nobj) = 0;
virtual char *          __stdcall CreateObject    (char * name) = 0;
virtual char *          __stdcall CreateObjects   (char * name, int num) = 0;
virtual int             __stdcall RetMaxObjects   (char * name) = 0;
virtual void            __stdcall Lock            (char * name) = 0;
virtual void            __stdcall Unlock          (char * name) = 0;
virtual int             __stdcall DumpData        (char * name, char * fname) = 0;
virtual int             __stdcall SaveData        (char * name, char * fname) = 0;
virtual int             __stdcall LoadData        (char * name, char * fname) = 0;
virtual int             __stdcall TestData        (char * name) = 0;
virtual int             __stdcall SaveXXX         (char * name, char * filename, char type) = 0;
};

#endif
```

Файл SystemExtentions1\Console.h

```
// Console.h - header
// created 14.10.2002 by header creator
#pragma once
#ifndef __CONSOLE_HEADER_FILE_BY_SE_INCLUDED__
#define __CONSOLE_HEADER_FILE_BY_SE_INCLUDED__

void CommandNewVar (IConsole* c, int ln, char* data);
void AddCommands ();

#endif //__CONSOLE_HEADER_FILE_BY_SE_INCLUDED__
```

Файл SystemExtentions1\Extern.h

```
#pragma once
#ifndef __EXTERN_HEADER_FILE_BY_SE_INCLUDED__
#define __EXTERN_HEADER_FILE_BY_SE_INCLUDED__

#include <objbase.h>

// {28F0D3F0-D97B-4a69-B394-89680FF698C5}
static const GUID IID_IExtern =
{ 0x28f0d3f0, 0xd97b, 0x4a69, { 0xb3, 0x94, 0x89, 0x68, 0xf, 0xf6, 0x98, 0xc5 } };

interface IExtern : public IUnknown
{
```

```
};

#endif
```

Файл SystemExtentions1\ExternCore.h

```
#pragma once
#ifndef __EXTERNCORE_HEADER_FILE_BY_SE_INCLUDED__
#define __EXTERNCORE_HEADER_FILE_BY_SE_INCLUDED__

#include "Extern.h"
#include "system.h"

#include "..\Common\Utilities\module.h"

class CExtern : public IExtern, public CObj
{
private:
    static SClassRef sClass;
public:
    DEFINE_IUNKNOWN();

    CExtern();

    static bool selfReg();

    static SParam NewVar ( const CParams& par);
    static /*self*/SParam cmd ( const CParams& par); // 0 - eStr
    static /*eFloat*/SParam Random ( const CParams& par); // 1 - eFloat
    static /*eInt*/SParam GetSystemTimer ( const CParams& par); // 0
};

extern CExtern* Extern;

#endif //__LAGENT_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл SystemExtentions1\System.h

```
// System.h - header
// created 03.11.2002 by header creator
#pragma once
#ifndef __SYSTEM_HEADER_FILE_BY_SE_INCLUDED__
#define __SYSTEM_HEADER_FILE_BY_SE_INCLUDED__

#include "..\Linker\Linker.h"
#include "..\CPPVM\CPPVM.h"
#include "..\SmallVM\SmallVM.h"
#include "..\Console\Console.h"
#include "externcore.h"

extern ILinker*  Linker;
extern ICPPVM*   CPPVM;
extern ISmallVM* SmallVM;
extern IConsole* Console;

#endif //__SYSTEM_HEADER_FILE_BY_SE_INCLUDED__
```

Файл Utilities\Module.h

```
// Module.h - header
// created 11.09.2002 by header creator
#pragma once
#ifndef __MODULE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __MODULE_HEADER_FILE_BY_VASFED_INCLUDED__

#define DEFINE_IUNKNOWN() \
    public:\
    ULONG STDMETHODCALLTYPE AddRef ();\
    ULONG STDMETHODCALLTYPE Release ();\
    HRESULT STDMETHODCALLTYPE QueryInterface (const IID &iid, void ** ptr);\
    volatile long m_refs;

#define IMPLEMENT_IUNKNOWN(classname,intname,iid) \
    ULONG STDMETHODCALLTYPE classname::AddRef () \
        {return InterlockedIncrement (&m_refs);} \
    ULONG STDMETHODCALLTYPE classname::Release () \
        {if (InterlockedDecrement (&m_refs) == 0){/*delete this;*/return 0;}return m_refs;} \
    HRESULT STDMETHODCALLTYPE classname::QueryInterface (const IID &Iid, void** ptr) \
        {       if (Iid == IID_IUnknown)   {*ptr = static_cast <IUnknown*> (this);} \
                else if (Iid == iid)          {*ptr = static_cast <intname*> (this);} \
                else {*ptr = NULL; return E_NOINTERFACE;} \
        reinterpret_cast<IUnknown*>(*ptr) -> AddRef (); \
        return S_OK;}
```

```
#define BEGIN_CLASS_INTERFACE_MAP(classname) \
    ULONG STDMETHODCALLTYPE classname::AddRef ()\
        {return InterlockedIncrement (&m_refs);}\
    ULONG STDMETHODCALLTYPE classname::Release ()\
        {if (InterlockedDecrement (&m_refs) == 0){/*delete this;*/return 0;}return m_refs;}\
    HRESULT STDMETHODCALLTYPE classname::QueryInterface (const IID &Iid, void ** ptr)\
        {     if (Iid == IID_IUnknown) {*ptr = static_cast <IUnknown*> (this);}

#define DECLARE_CLASS_INTERFACE(interfacename,iid) else if (Iid == iid) {*ptr = static_cast
<interfacename*> (this);}

#define END_CLASS_INTERFACE_MAP() \
        else {*ptr = NULL; return E_NOINTERFACE;}\
        reinterpret_cast <IUnknown *> (*ptr) -> AddRef ();\
        return S_OK;}

#endif //__MODULE_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл Utilities\Utilities.h

```
//Utilties.h - some additional data types and algorythms

#ifndef __UTILITIES_FOR_AI_HEADER_FILE_BY_VASFED_INCLUDED__
#define __UTILITIES_FOR_AI_HEADER_FILE_BY_VASFED_INCLUDED__

#ifdef UTILITIES_EXPORTS
#define UTIL_API __declspec(dllexport)
#else
#define UTIL_API __declspec(dllimport)
#endif
#include <Exeption.h>

void    UTIL_API randomize();
float   UTIL_API random (float);
double  UTIL_API random (double);
int     UTIL_API random (int);

struct Matrix;

struct UTIL_API Vector
{
    float x;
    float y;

    Vector (float x = 0, float y = 0);
    Vector operator +(Vector op2);
    Vector operator -(Vector op2);
    Vector& operator += (Vector op2);
    Vector& operator -= (Vector op2);

    Vector operator *(float number);
    Vector operator /(float number);
    Vector& operator *=(float number);
    Vector& operator /=(float number);
    operator == (Vector v2);

    Vector Rotare (float angle, bool isDegrees = true);
    Vector& RotareIt (float,bool isDegrees = true); //Rotare that sets this vector to result
    //operator Matrix();

    Vector& Normalize();
    void DrawTo (HDC, int x, int y, int zoom);
    float Len();
};

/*
class Matrix
{
    public:
        Matrix(int sizex = 2, int sizey = 2);
        ~Matrix();

        float& Element(int x, int y);

        Matrix& operator =(float);
        Matrix& operator ,(float);

        operator *=(float number);
        operator *=(Matrix);

        operator /=(float number);
        operator /=(Matrix);

        operator +=(Matrix);
        operator -=(Matrix);

    protected:
        int m_sizex;
        int m_sizey;

        float* m_data;
```

```
        int m_fillingIndex;
};
*/

struct UTIL_API CRectangle
{
    CRectangle ();
    CRectangle (int, int, int, int);
    CRectangle (RECT*);

    int x1;
    int y1;
    int x2;
    int y2;

    void DrawTo (HDC, int xoff, int yoff, int zoom);
};

template <typename T> class CBag
{
    public:
        CBag (T* ptr, int size, bool clear = false):m_pointer(ptr), m_numObjects (size),
m_clearOnDelete(clear) {}
        CBag (CBag<T> &s):m_pointer(s.m_pointer), m_numObjects(s.m_numObjects), m_clearOnDelete (false){}
        ~CBag (){if (m_clearOnDelete) delete m_pointer; m_pointer = NULL;}

        T& operator [](int i)
            {
            AssertC (i > 0 && i < m_numObjects, badParams);
            AssertT (m_pointer, "Null-pointer assignment detected");
            return m_pointer[i];
            }

        T* operator ->()
            {
            AssertT (m_pointer, "Null-pointer assignment detected");
            }

        void DoNotClear () {if (m_clearOnDelete) m_clearOnDelete = false;}

    //protected:
        T*  m_pointer;
        int m_numObjects;
        bool m_clearOnDelete;
};

template <typename T> class CStack
{
    public:
        CStack  (int size, bool throwOverflow = true, bool throwUnderflow = true):
            m_size (size), m_tail(0), m_data(NULL), m_throwOverflow (throwOverflow), m_throwUnderflow
(throwUnderflow)
            {
            Assert (m_size > 1);
            m_data = new T [m_size];
            Assert (m_data);
            }

        ~CStack (){if (m_data) delete m_data; m_data = NULL;}

        T&      Pop ()
            {
            Assert (m_data);
            if (!m_tail)
                {
                AssertT (m_throwUnderflow, "CStack Underflow");
                m_tail = m_size - 1;
                }
            return m_data[--m_tail];
            }

        T*      PopP ()
            {
            Assert (m_data);
            if (!m_tail)
                {
                AssertT (m_throwUnderflow, "CStack Underflow!");
                return NULL;
                }
            return &m_data[--m_tail];
            }

        void    Push(T& t)
            {
            Assert (m_data);
            if (m_tail == m_size)
                {
                AssertT (m_throwOverflow, "CStack overflow!")
                m_tail = 0;
                }
            m_data[m_tail++] = t;
            }

    protected:
        T* m_data;
        int m_tail;
        int m_size;
```

```
        bool m_throwOverflow;
        bool m_throwUnderflow;
};

class UTIL_API CThread // objects, that are children or this class are automatically runned
                //in separate thread, (method void Main() is called)
                //TO BE CREATED ONLY VIA NEW!!!
{
    public:
        CThread();
        ~CThread();

    protected:
        virtual void Main();
        virtual void Init();
        int     m_threadID;

    private:
        int StartThread();
        static void ThreadEntryPoint (void*);
};

class UTIL_API CBitMatrix
{
    public:
        CBitMatrix (int cx, int cy);
        ~CBitMatrix ();

        bool operator ()(int x, int y);

        bool Get       (unsigned int x, unsigned int y);
        void Set       (unsigned int x, unsigned int y, bool value);

        bool FastGet   (unsigned int x, unsigned int y);
        void FastSet   (unsigned int x, unsigned int y, bool val);

    protected:
        unsigned char* m_data;
        int m_dataSize;
        int m_sizex; int m_sizey;
};

#endif
```

Файл 3dConsoleExtensions\Commands.cpp

```
// Commands.cpp - implementation of Commands.h, of module 3dConsoleExtensions
// created 08.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "3dConsoleExtensions"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Commands.h"

#include <Exeption.h>      // use ExeptionEngine
#include <process.h>

#pragma warning (disable:4244)

void Init ();

int AgentsCount = 0;
int ScrX = 320, ScrY = 240;
bool FullScreen = false;
extern bool Render;
bool Started = false;
extern int NoErrors;

//-----------------------------------------------------------------------------
COMMAND_PROC_LEXEMS (InitSTM)
{
    if (ln != 1) {c->printf ("Error\n"); return;}
    if (data[0].type != data[0].LT_Number) {c->printf ("Error\n"); return;}
    Master->Init (data[0].idata);
    Master->Set3DOptimizer (true);

}
//-----------------------------------------------------------------------------

COMMAND_PROC_LEXEMS (SetNumAgents)
{
    if (ln != 1) {c->printf ("Error\n"); return;}
    if (data[0].type != data[0].LT_Number) {c->printf ("Error\n"); return;}
    AgentsCount = data[0].idata;
    Master -> SetNewDataType (sizeof (CMesh), AgentsCount, "Mesh");
    Master -> SetNewDataType (sizeof (CPolygon), AgentsCount * 216, "Polygon");
    Master -> SetNewDataType (sizeof (CAcceleration), AgentsCount, "Acceleration");
    Master -> SetNewDataType (sizeof (CSpeed), AgentsCount, "Speed");
    Master -> SetNewDataType (sizeof (CIntersectionSphere), AgentsCount, "IntersectionSphere");
```

```
    Master -> SetNewDataType (sizeof (CIntersectionSpace), AgentsCount, "IntersectionSpace");
    Master -> SetNewDataType (sizeof (CBoundingBox), AgentsCount, "BoundingBox");
    Master -> SetNewDataType (sizeof (CPhysMaterial), AgentsCount, "Material");
}

//-------------------------------------------------------------------------------------
COMMAND_PROC_STRING (LoadMap)
{
    if (!strlen (str)) return;

    Map->Reset (true);
    Map->Load (str);
}
//-------------------------------------------------------------------------------------
COMMAND_PROC_LEXEMS (SetScreenSettings)
{
    if (ln != 3) return;
    ScrX = data[0].idata;
    ScrY = data[1].idata;
    FullScreen = data[2].idata > 0? 1:0;
}

COMMAND_PROC_LEXEMS (SetFog)
{
    if (ln != 3) return;
    Renderer->SetFogParameters (data[0].ddata, data[1].ddata, data[2].ddata);
}

COMMAND_PROC_LEXEMS (SetCamera)
{
    if (ln != 16) return;
    Renderer->SetCamera (data[0].ddata, data[1].ddata, data[2].ddata,
                         data[3].ddata, data[4].ddata, data[5].ddata,
                         data[6].ddata, data[7].ddata, data[8].ddata,
                         data[9].ddata, data[10].ddata, data[11].ddata, data[12].ddata,
                         data[13].ddata, data[14].ddata, data[15].idata);

}

COMMAND_PROC_LEXEMS (SetClearColor)
{
    if (ln != 4) return;
    Renderer->SetClearColor (data[0].ddata / 255, data[1].ddata / 255, data[2].ddata / 255,
                             data[3].ddata / 255);
}

COMMAND_PROC_LEXEMS (SetGravity)
{
    if (ln != 3) return;
    Environment->SetGravity (data[0].ddata, data[1].ddata, data[2].ddata);
}

COMMAND_PROC_STRING (LoadTextures)
{
    if (!strlen (str)) return;
    int handle = Master -> SetNewDataType (sizeof (CColorMatrix), 10, "Texture");
    Master -> Lock ("Texture");
    AssertT (handle != -1, "Memory for textures is not initialized");

    Renderer -> LoadResource (Master -> RetData ("Texture"), str);
    Master -> Unlock ("Texture");
}

COMMAND_PROC_STRING (LoadLandscape)
{
    if (!strlen (str)) return;
    int handle = Master -> SetNewDataType (sizeof (CLandscape), 1, "Landscape");
    AssertT (handle != -1, "Memory for landscape is not initialized");
    Master -> Lock ("Landscape");
    Renderer -> LoadResource (Master -> RetData ("Landscape"), str);
    Master -> Unlock ("Landscape");
}

COMMAND_PROC_LEXEMS (MakeAgent)
{
    if (ln < 6) return;

    if (Server->CreateAgent (Server->FindAgentCreatorByName (data[0].data),
                         (data[2].type == data[2].LT_Number || data[2].type == data[2].LT_Float)?
                         data[2].ddata: random (Map->GetSizeX ()) - Map->GetSizeX () / 2,
                         (data[3].type == data[3].LT_Number || data[3].type == data[3].LT_Float)?
                         data[3].ddata: random (Map->GetSizeX ()) - Map->GetSizeX () / 2,
                         (data[4].type == data[4].LT_Number || data[4].type == data[4].LT_Float)?
                         data[4].ddata: random (Map->GetSizeX ()) - Map->GetSizeX () / 2,
                         data[5].idata) >= 0)
        {
        Master->LoadXXX (data[1].data);
        }
}

COMMAND_PROC_STRING (LoadAgents)
{
    c->AddCommand ("AGENT", (CommandProcP)MakeAgent, true);
    char cmd[256];
    sprintf (cmd, "batch %s", str);
    c->RaiseCommand (cmd);
```

```
    Master -> Lock ("Mesh");
    for (int b = 0; b < AgentsCount; b ++)
        Master -> CreateObject ("Mesh");
    Master -> Unlock ("Mesh");

    Master -> Lock ("Polygon");
    for (int i = 0; i < AgentsCount * 216; i ++)
        Master -> CreateObject ("Polygon");
    Master -> Unlock ("Polygon");

    c->DeleteCommand ("AGENT");
}

COMMAND_PROC_SIMPLE (MakeWindow)
{
    Renderer->Resize (0, 0, ScrX, ScrY, FullScreen);
    Renderer->MakeGLWindow (GetModuleHandle (NULL), SW_SHOWNORMAL, "Elendor", "OpenGL window");
}

COMMAND_PROC_STRING (SetLights)
{
    if (!strlen (str)) return;

    Master -> LoadXXX (str);

    Master -> Lock ("LightSource");
    Renderer -> SetLights (Master -> RetData ("LightSource"));
    Master -> Unlock ("LightSource");
}

COMMAND_PROC_SIMPLE (Prepare)
{
    Master -> LoadXXX ("..\\plants.xxx");

    Master -> Lock ("Mesh");
    Master -> Lock ("Polygon");
    Master -> Lock ("LightSource");
    Master -> Lock ("Speed");
    Master -> Lock ("Acceleration");
    Master -> Lock ("IntersectionSphere");
    Master -> Lock ("IntersectionSpace");
    Master -> Lock ("BoundingBox");
    Master -> Lock ("Material");
    Master -> Lock ("Landscape");
    Init ();

    Renderer -> PrepareMeshes (Master -> RetData ("Mesh"),
                               Master -> RetData ("Polygon"),
                               Master -> RetData ("LightSource"));

    for (int n = 0; n < AgentsCount; n ++)
        {
        IAgent* a = Server->GetAgent(n);
        if (!a) continue;
        ((CMesh*)(Master -> RetObject ("Mesh", AgentsCount + n))) -> SetCenter (a->m_x, a->m_y, a->m_z);
        }

    CSmartPointer MeshPtr = Master -> RetData ("Mesh");
    MeshPtr += AgentsCount;
    Environment -> MakeTable (MeshPtr,
                              Master -> RetData ("Speed"),
                              Master -> RetData ("Acceleration"),
                              Master -> RetData ("IntersectionSphere"),
                              Master -> RetData ("IntersectionSpace"),
                              Master -> RetData ("BoundingBox"),
                              Master -> RetData ("Material"),
                              Master -> RetData ("Landscape"));

    Master -> Unlock ("Mesh");
    Master -> Unlock ("Polygon");
    Master -> Unlock ("LightSource");
    Master -> Unlock ("Speed");
    Master -> Unlock ("Acceleration");
    Master -> Unlock ("IntersectionSphere");
    Master -> Unlock ("IntersectionSpace");
    Master -> Unlock ("BoundingBox");
    Master -> Unlock ("Material");
    Master -> Unlock ("Landscape");

    Renderer->SetRenderLevel (1);
}

COMMAND_PROC_LEXEMS (SetPhysLevel)
{
    if (ln == 0) data[0].idata = 1;
    Environment->SetPhysicLevel (data[0].idata);
}

COMMAND_PROC_LEXEMS (SetGraphLevel)
{
    if (ln == 0) data[0].idata = 1;
    Renderer->SetRenderLevel (data[0].idata);
}

COMMAND_PROC_STRING (ChangeDirectory)
{
    if (!strlen (str)) return;
```

```
    SetCurrentDirectory (str);
}

//-----------------------------------------------------------------------------

BEGIN_COMMAND_LIST (ConfigCommands)
    DEFINE_COMMAND ("STORAGEMASTER_SIZE", InitSTM, true)
    DEFINE_COMMAND ("NUMAGENTS", SetNumAgents, true)
    DEFINE_COMMAND ("AGENTSMAP", LoadMap, false)
    DEFINE_COMMAND ("SCREEN", SetScreenSettings, true)
    DEFINE_COMMAND ("FOG", SetFog, true)
    DEFINE_COMMAND ("CAMERA", SetCamera, true)
    DEFINE_COMMAND ("FOGCOLOR", SetClearColor, true)
    DEFINE_COMMAND ("GRAVITY", SetGravity, true)
    DEFINE_COMMAND ("TEXTURES", LoadTextures, false)
    DEFINE_COMMAND ("LANDSCAPE", LoadLandscape, false)
    DEFINE_COMMAND ("AGENTS", LoadAgents, false)
    DEFINE_COMMAND ("MAKEWINDOW", MakeWindow, false)
    DEFINE_COMMAND ("LIGHTS", SetLights, false)
    DEFINE_COMMAND ("COMPILE", Prepare, false)
    DEFINE_COMMAND ("PHYSICSLEVEL", SetPhysLevel, true)
    DEFINE_COMMAND ("RENDERLEVEL", SetGraphLevel, true)
    DEFINE_COMMAND ("CD", ChangeDirectory, false)
END_COMMAND_LIST();

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------

void LoadWorld (IConsole* con, int, char* str)
{

    if (!strlen (str))
        {
        con->printf ("USAGE: loadwv <world config file>");
        return;
        }

    char cmd[256];
    sprintf (cmd, "batch %s", str);

    Conex->AttachCommands (CM (ConfigCommands), CMS (ConfigCommands), Console, true, "Temporary
Commands");
    Console->RaiseCommand ("echo off");
    Renderer->SetLoDLevel (1);
    //Renderer->LoadResource (NULL, "settings.rrc");

    Console->RaiseCommand (cmd);
    Conex->DetachCommands (CM (ConfigCommands), CMS (ConfigCommands), Console);
}

void MainLoop (void*)
{
    MSG lpMsg;
    Console->printf ("\nMainloop started.\n");
    for (;;)
        {
        try {
            if (PeekMessage (&lpMsg, NULL, 0, 0, PM_NOREMOVE))
                {
                if (GetMessage(&lpMsg, NULL, 0, 0) == 0) break;
                TranslateMessage(&lpMsg);
                DispatchMessage(&lpMsg);
                }
            else
                if (Render && !NoErrors)
                    Renderer -> Render ();
            }
        Catch ()}
        catch (...)
            {
            Trace ("Unknown excetion happened.");
            }
        }
}

COMMAND_PROC_SIMPLE (Start)
{
//  if (!Started)
        {
        c->printf ("Starting rendering...\n");
        _beginthread (MainLoop, NULL, NULL);
        Started = false;
        }
}

//-----------------------------------------------------------------------------
void AddConsoleCommands (IConsole* con)
{
    #define IGNORE_EXCEPTIONS(exp) try {(exp);} catch (...){}
    #define IE IGNORE_EXCEPTIONS

    IE (con->AddCommand ("loadvw", (CommandProcP)LoadWorld, false));
    IE (con->AddCommand ("start", (CommandProcP)Start, false));


    #undef IE
```

```
    #undef IGNORE_EXCEPTIONS
}
```

Файл 3dConsoleExtensions\Graphics.cpp

```
// Graphics.cpp - implementation of Graphics.h, of module 3dConsoleExtensions
// created 08.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "3dConsoleExtensions"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Graphics.h"

#include <Exeption.h>      // use ExeptionEngine

bool Render = true;
int NoErrors = 0;
int AgentNum = 0;
extern int ScrX;
extern int ScrY;
extern int AgentsCount;

int RenderLevel = 3;
int PhysicLevel = PL_NO_MOVE;
char* StrRenderLevels[] = {"None", "Wireframe", "Solid", "Solid&Light",
                           "LowTextured", "Textured", "Best View"};
CPoint3D Target (0.0F, 0.0F, 0.0F);

void StartRender   (IConsole*, int, void*)
{

}
//-----------------------------------------------------------------------------

void StopRender    (IConsole*, int, void*)
{

}
//-----------------------------------------------------------------------------
void RedrawProc ();
void KeyboardProc (HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);

void Init ()
{
    Renderer -> SetVolumeFogDepth (false, 0.0F);
    Renderer -> InitTransparentBuffer (1, 1);
    Renderer -> MakeCurrent ();
    Renderer -> Clear ();
    Renderer -> SetKeyboardFunc (KeyboardProc);
    Renderer -> SetRedrawFunc (RedrawProc);
}

void RedrawProc ()
    {
    try {
        Master -> Lock ("Mesh");
        Master -> Lock ("Landscape");
        Master -> Lock ("BillBoard");
        Renderer -> Clear ();

        Renderer -> CameraFollow (1, reinterpret_cast <CMesh *> (Master -> RetObject ("Mesh", AgentsCount
+ AgentNum)) -> RetCenter ());
        Renderer -> CameraOnLand (1, ((CLandscape *) Master -> RetObject ("Landscape", 0)) -> RetPHorizont
(0));
        Renderer -> CameraOnLand (2, ((CLandscape *) Master -> RetObject ("Landscape", 0)) -> RetPHorizont
(0));
        Renderer -> RenderMeshes (Master -> RetData ("Mesh"));
        Renderer -> RenderLandscape (Master -> RetData ("Landscape"));
        Renderer -> RenderBillBoards (Master -> RetData ("BillBoard"));

        try {
            Server->PerformAgentsStep ();
            Environment -> MoveAll ();
            }
        catch (...)
            {
            NoErrors = NoErrors == 2? false: 2;
            if (NoErrors) Console->printf ("Error happened in MultiAgentSystem.\n");
            else Console->printf ("Error happened in MultiAgentSystem.\nRendering STOPPED\n");
            }

        for (int n = 0; n < AgentsCount; n ++)
            {
            IAgent* a = Server->GetAgent (n);
            if (!a) continue;
            CMesh* mesh = (CMesh*)(Master -> RetObject ("Mesh", AgentsCount + n));
            CPoint3D p = mesh->RetCenter ();
            a->m_x = p (0);
            a->m_y = p (1);
```

```
            a->m_z = p (2);
            }

        Master -> Unlock ("Mesh");
        Master -> Unlock ("Landscape");
        Master -> Unlock ("BillBoard");
        }
    catch (...)
        {
        NoErrors = NoErrors == 3? false: 3;
        if (NoErrors == 3)
            {
            Console->printf ("Error during rendering.\n  If often happens because of powerless of 3D
system of computer.\n  Trying to fix..\n");
            Renderer->SetRenderLevel ((--RenderLevel > 0? RenderLevel: RenderLevel = RL_WIREFRAME));
            }
        else {Console->printf ("Cannot recover after error.\n Rendering stopped. System needs
restart.\n");}
        //MessageBox (NULL, "Runtime Error during rendering", "Error", MB_OK);
        //PostQuitMessage (0);
        return;
        }
        if (NoErrors == 3)
            {
            NoErrors = true;
            Console->printf ("   ok.\n\n");
            }
    }
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
void KeyboardProc (HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam)
    {
    switch ((char) wParam)
        {
        case 'q': Renderer -> CameraSlide (1, 10.0F);
                  break;
        case 'a': Renderer -> CameraSlide (1, -10.0F);
                  break;
        //case 'z': Renderer -> CameraRotate (1, 0.0F, 0.0F, -5.0F);
        //          break;
        //case 'x': Renderer -> CameraRotate (1, -5.0F, 0.0F, 0.0F);
        //          break;
        //case 'y': Renderer -> CameraRotate (1, 0.0F, -5.0F, 0.0F);
        //          break;
        case '1': Renderer -> MakeCameraCurrent (1);
                  break;
        case '2': Renderer -> MakeCameraCurrent (2);
                  break;
        case ',': if (AgentNum > 0) AgentNum --;
                    Console->printf ("Current agent is now %d\n", AgentNum);
                  break;
        case '.': if (AgentNum < AgentsCount - 1) AgentNum ++;
                    Console->printf ("Current agent is now %d\n", AgentNum);
                  break;
        case 'f': //Trace ("FPS is %d", (int) Renderer -> RetFPS ());
                    Console->printf ("Fps is %g\n", Renderer->RetFPS ());
                  break;
        case 'i': Target.AddToFloats (-5.0F, 0.0F, 0.0F);
                  Renderer -> CameraFollow (2, Target);
                  break;
        case 'k': Target.AddToFloats (5.0F, 0.0F, 0.0F);
                  Renderer -> CameraFollow (2, Target);
                  break;
        case 'j': Target.AddToFloats (0.0F, -5.0F, 0.0F);
                  Renderer -> CameraFollow (2, Target);
                  break;
        case 'l': Target.AddToFloats (0.0F, 5.0F, 0.0F);
                  Renderer -> CameraFollow (2, Target);
                  break;
        case (char) 27: PostQuitMessage (0);
                  Console->printf ("Exited by pressing Esc");
                  break;
        case '+': if (RenderLevel < RL_EXTENSIONS_USED)
                        {
                        RenderLevel ++;
                        Renderer -> SetRenderLevel (RenderLevel);
                        Console->printf ("Setting Render level to %d (%s)", RenderLevel,
StrRenderLevels[RenderLevel]);
                        if (RenderLevel == RL_EXTENSIONS_USED)
                            {
                            Console->printf (" warning: The BestView mode is not supported by some video
cards\n");
                            }
                        }
                  break;
        case '-': if (RenderLevel > RL_WIREFRAME)
                        {
                        RenderLevel --;
                        Renderer -> SetRenderLevel (RenderLevel);
                        Console->printf ("Setting Render level to %d (%s)", RenderLevel,
StrRenderLevels[RenderLevel]);
                        }
                  break;
        case '*': if (PhysicLevel < PL_BBOX_TEST)
                        {
                        PhysicLevel ++;
```

```
                    Environment -> SetPhysicLevel (PhysicLevel);
                    }
                break;
        case '/': if (PhysicLevel > PL_NO_MOVE)
                    {
                    PhysicLevel --;
                    Environment -> SetPhysicLevel (PhysicLevel);
                    }
                break;
        case 'c':
            {
            IAgent* a = Server->GetAgent (AgentNum);
            CPoint3D np (a->m_x, a->m_y, a->m_z);
            Target = np;
            Renderer -> CameraFollow (2, Target);
            break;
            }
        case 'e': Renderer -> SetFogParameters (1.0F, 0.0F, 30.0F);
                Renderer -> SetVolumeFogDepth (true, 20.0F);
                break;
        case 'd': Renderer -> SetFogParameters (0.0035F, 0.0F, 50.0F);
                Renderer -> SetVolumeFogDepth (false, 0.0F);
                break;
        case 'r': Renderer -> Resize (0, 0, 640, 480, false);
                break;
        case 's': if (ScrX == 320 && ScrY == 240)
                    {
                    ScrX = 640; ScrY = 480;
                    Renderer -> Resize (0, 0, ScrX, ScrY, true);
                    break;
                    }
                if (ScrX == 640 && ScrY == 480)
                    {
                    ScrX = 800; ScrY = 600;
                    Renderer -> Resize (0, 0, ScrX, ScrY, true);
                    break;
                    }
                if (ScrX == 800 && ScrY == 600)
                    {
                    ScrX = 1024; ScrY = 768;
                    Renderer -> Resize (0, 0, ScrX, ScrY, true);
                    break;
                    }
                if (ScrX == 1024 && ScrY == 768)
                    {
                    ScrX = 320; ScrY = 240;
                    Renderer -> Resize (0, 0, ScrX, ScrY, true);
                    break;
                    }
        default:  break;
        }
    }
```

Файл 3dConsoleExtensions\Module.cpp

```
// Module.cpp - implementation of Module.h, of module 3dConsoleExtensions
// created 08.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "3dConsoleExtensions"

#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine

#include "..\Common\Utilities\Module.h"
#include <objbase.h>
#include "Commands.h"

class DllClass: IUnknown
{DEFINE_IUNKNOWN();};
/* {C0B11C2B-62EC-4524-8674-1F7893DA08F8}*/ static const GUID IID_IllClass = { 0xc0b11c2b, 0x62ec, 0x4524,
{ 0x86, 0x74, 0x1f, 0x78, 0x93, 0xda, 0x8, 0xf8 } };
IMPLEMENT_IUNKNOWN (DllClass, IUnknown, IID_IllClass);

    #define DLL_CLASS DllClass
    #define DLL_NAME "ConsoleExtensions2"
    #include "..\Common\Utilities\Module.c"

DLL_ONLOAD (loader)
{
    Assert (loader);
    Loader = loader;
    if (!(Console = (IConsole*)loader->GetInterface (IID_IConsole)))
        {
        loader->LoadComponent ("Console");
        loader->GetInterface (IID_IConsole, (void**)&Console);
        Assert (Console);
        }
```

```
    Console->printf ("VWM: VirtualWorldManager console extension version 0.9 by Vasfed\n");
    if (!(Master = (IStorageMaster*)loader->GetInterface (IID_IStorageMaster)))
        {
        loader->LoadComponent ("Stm32");
        loader->GetInterface (IID_IStorageMaster, (void**)&Master);
        Assert (Master);
        }
    if (!(Renderer = (IRenderer*)loader->GetInterface (IID_IRenderer)))
        {
        loader->LoadComponent ("renderer");
        loader->GetInterface (IID_IRenderer, (void**)&Renderer);
        Assert (Renderer);
        }
    if (!(Environment = (IEnvironment*)loader->GetInterface (IID_IEnvironment)))
        {
        loader->LoadComponent ("Physics");
        loader->GetInterface (IID_IEnvironment, (void**)&Environment);
        Assert (Environment);
        }
    if (!(Map = (IAgentsMap*)loader->GetInterface (IID_IAgentsMap)))
        {
        loader->LoadComponent ("Map");
        loader->GetInterface (IID_IAgentsMap, (void**)&Map);
        Assert (Map);
        }
    if (!(Server = (IAgentsServer*)loader->GetInterface (IID_IAgentsServer)))
        {
        loader->LoadComponent ("AgSrv32");
        loader->GetInterface (IID_IAgentsServer, (void**)&Server);
        Assert(Server);
        }
    if (!(Conex = (IConsoleExtensions*)loader->GetInterface (IID_IConsoleExtensions)))
        {
        loader->LoadComponent ("conex");
        loader->GetInterface (IID_IAgentsServer, (void**)&Conex);
        Assert (Conex);
        }
    AddConsoleCommands (Console);
    Console->printf ("VWM: System configured to work with Virtual World\n");
    Console->printf ("VWM: Type loadvw to load a virtual world\n");
}
```

Файл 3dConsoleExtensions\System.cpp

```
// System.cpp - implementation of System.h
// created 08.11.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "System"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include <Exeption.h>      // use ExeptionEngine

ILoader*            Loader = NULL;
IConsole*           Console = NULL;

IStorageMaster *    Master = NULL;
IRenderer *         Renderer = NULL;
IEnvironment *      Environment = NULL;
IAgentsMap*         Map = NULL;
IAgentsServer*      Server = NULL;
IConsoleExtensions* Conex = NULL;
```

Файл 3dsloader\3DSLoader.cpp

```
// 3DSLoader.cpp : Defines the entry point for the console application.
//

#define __PROGRAMMER   "Xroft"
#define __MODULE_NAME  "3DSLoader"

#include "stdafx.h"
#include "3dsloader.h"

/*********************************************************************/

void ReadChunk (SChunk * chunk)
```

```
    {
    chunk -> myBytesRead =  (int) fread (&(chunk -> myID), 1, 2, ImportFile);
    chunk -> myBytesRead += (int) fread (&(chunk -> myLength), 1, 4, ImportFile);
    Trace ("New chunk is found, chunk id is %d, chunk length is %d",
           chunk -> myID, chunk -> myLength);
    }

/*************************************************************************/

int GetString (char * str)
    {
    int index = 0;
    fread (str, 1, 1, ImportFile);

    while (*(str + index ++) != 0)
        {
        fread (str + index, 1, 1, ImportFile);
        }
    return strlen (str) + 1;
    }

/*************************************************************************/

void Import3DSFile (char * filename)
    {
    ImportFile = fopen (filename, "rb");
    AssertT (ImportFile, "Wrong file to import");
    Trace ("File %s is found", filename);

    CurrentChunk = new SChunk;
    TempChunk    = new SChunk;
    ReadChunk (CurrentChunk);
    AssertT (CurrentChunk -> myID == PRIMARY, "Invalid 3DS file format");
    Trace ("PRIMARY chunk is found");

    ProcessNextChunk (CurrentChunk);
    }

/*************************************************************************/

void ProcessNextChunk (SChunk * prevChunk)
    {
    Trace ("ProcessNextChunk started");
    char buffer [50000];
    CurrentChunk = new SChunk;
    int bytesread = 0;

    while (prevChunk -> myBytesRead < prevChunk -> myLength)
        {
        ReadChunk(CurrentChunk);
        switch (CurrentChunk -> myID)
            {
            case VERSION:
                Trace ("VERSION chunk is found");
                CurrentChunk -> myBytesRead += fread (buffer, 1, CurrentChunk -> myLength - CurrentChunk -
> myBytesRead, ImportFile);
                break;

            case OBJECTINFO:
                Trace ("OBJECTINFO chunk is found");
                ReadChunk (TempChunk);
                TempChunk -> myBytesRead += fread (buffer, 1, TempChunk -> myLength - TempChunk ->
myBytesRead, ImportFile);
                CurrentChunk -> myBytesRead += TempChunk -> myBytesRead;
                ProcessNextChunk (CurrentChunk);
                break;

            case MATERIAL:
                Trace ("MATERIAL chunk is found");
                ProcessNextMaterialChunk (CurrentChunk);
                break;

            case OBJECT:
                Trace ("OBJECT chunk is found");
                MeshNum ++;
                printf ("Mesh %d is found\n", MeshNum);
                CurrentChunk -> myBytesRead += GetString (buffer);
                ProcessNextObjectChunk (CurrentChunk);

                Mesh.Init (FaceNum, 0);
                printf ("Please enter mesh position: ");
                scanf ("%f, %f, %f", &X, &Y, &Z);
                Mesh.SetCenter (X, Y, Z);
                Mesh.RetMaterial ().Init (1.0F, 0.0F, 0.0F);
                Mesh.RetMaterial ().SetColorSource (COLOR_PER_VERTEX);
                Stm -> SetNewDataType (sizeof (CMesh), 1, "Mesh");
                Stm -> Lock ("Mesh");
                Stm -> AddObject ("Mesh", &Mesh);
                Stm -> SaveXXX ("Mesh", SaveFileName, 'n');
                Stm -> Unlock ("Mesh");

                Stm -> SetNewDataType (sizeof (CPolygon), FaceNum, "Polygon");
                Stm -> Lock ("Polygon");

                for (int n = 0; n < FaceNum; n ++)
                    {
```

```
                    Polys [n].SetVertex (Points [Faces [n].myA] (0), Points [Faces [n].myA] (2), Points
[Faces [n].myA] (1),
                                         0.0F, 0.0F, 1.0F, R, G, B, A,
                                         TexCoords [Faces [n].myA].myX, TexCoords [Faces [n].myA].myY,
1.0F);
                    Polys [n].SetVertex (Points [Faces [n].myB] (0), Points [Faces [n].myB] (2), Points
[Faces [n].myB] (1),
                                         0.0F, 0.0F, 1.0F, R, G, B, A,
                                         TexCoords [Faces [n].myB].myX, TexCoords [Faces [n].myB].myY,
1.0F);
                    Polys [n].SetVertex (Points [Faces [n].myC] (0), Points [Faces [n].myC] (2), Points
[Faces [n].myC] (1),
                                         0.0F, 0.0F, 1.0F, R, G, B, A,
                                         TexCoords [Faces [n].myC].myX, TexCoords [Faces [n].myC].myY,
1.0F);
                    Polys [n].SetTexPointer (TexID);
                    }
                ComputeNormals ();
                SmoothNormals ();
                for (n = 0; n < FaceNum; n ++)
                    {
                    Stm -> AddObject ("Polygon", &Polys [n]);
                    }
                Stm -> SaveXXX ("Polygon", SaveFileName, 'a');
                Stm -> Unlock ("Polygon");
                Stm -> RemoveDataType ("Polygon");
                break;

            case EDITKEYFRAME:
                Trace ("EDITKEYFRAME chunk is found");
                break;

            default:
                Trace ("Unknown chunk is found");
                CurrentChunk -> myBytesRead += fread (buffer, 1, CurrentChunk -> myLength - CurrentChunk -
> myBytesRead, ImportFile);
                Trace ("%d bytes read", bytesread);
                break;
            }
        prevChunk -> myBytesRead += CurrentChunk -> myBytesRead;
        }
    delete [] CurrentChunk;
    CurrentChunk = prevChunk;
    Trace ("ProcessNextChunk exited");
    }

/*************************************************************************/

void ProcessNextMaterialChunk (SChunk * prevChunk)
    {
    Trace ("ProcessNextMaterialChunk started");
    char buffer [50000];
    CurrentChunk = new SChunk;
    int bytesread = 0;

    while (prevChunk -> myBytesRead < prevChunk -> myLength)
        {
        ReadChunk(CurrentChunk);
        switch (CurrentChunk -> myID)
            {
            case MATNAME:
                Trace ("MATNAME chunk is found");
                CurrentChunk -> myBytesRead += fread (buffer, 1, CurrentChunk -> myLength - CurrentChunk -
> myBytesRead, ImportFile);
                break;

            case MATMAP:
                ProcessNextMaterialChunk (CurrentChunk);
                break;

            case MATDIFFUSE:
                CurrentChunk -> myBytesRead += fread (buffer, 1, CurrentChunk -> myLength - CurrentChunk -
> myBytesRead, ImportFile);
                printf ("Please enter background color: ");
                scanf ("%f, %f, %f, %f", &R, &G, &B, &A);
                break;

            case MATMAPFILE:
                CurrentChunk -> myBytesRead += fread (MapFile, 1, CurrentChunk -> myLength - CurrentChunk
-> myBytesRead, ImportFile);
                printf ("Please enter ID for texture %s: ", MapFile);
                scanf ("%d", &TexID);
                printf ("Please enter background color: ");
                scanf ("%f, %f, %f, %f", &R, &G, &B, &A);
                break;

            default:
                Trace ("Unknown chunk is found");
                CurrentChunk -> myBytesRead += fread (buffer, 1, CurrentChunk -> myLength - CurrentChunk -
> myBytesRead, ImportFile);
                break;
            }
        prevChunk -> myBytesRead += CurrentChunk -> myBytesRead;
        }

    delete [] CurrentChunk;
    CurrentChunk = prevChunk;
```

```
    Trace ("ProcessNextMaterialChunk exited");
    }

/*******************************************************************************/

void ProcessNextObjectChunk (SChunk * prevChunk)
    {
    Trace ("ProcessNextObjectChunk started");
    char buffer [50000];
    CurrentChunk = new SChunk;
    int bytesread = 0;

    while (prevChunk -> myBytesRead < prevChunk -> myLength)
        {
        ReadChunk(CurrentChunk);
        switch (CurrentChunk -> myID)
            {
            case OBJECT_MESH:
                Trace ("OBJECT_MESH chunk is found");
                ProcessNextObjectChunk (CurrentChunk);
                break;

            case OBJECT_VERTICES:
                Trace ("OBJECT_VERTICES chunk is found");
                ReadVertices (CurrentChunk);
                break;

            case OBJECT_FACES:
                Trace ("OBJECT_FACES chunk is found");
                ReadIndices (CurrentChunk);
                break;

            case OBJECT_UV:
                Trace ("OBJECT_UV chunk is found");
                ReadTexCoords (CurrentChunk);
                break;

            default:
                Trace ("Unknown chunk is found");
                bytesread = fread (buffer, 1, CurrentChunk -> myLength - CurrentChunk -> myBytesRead,
ImportFile);
                CurrentChunk -> myBytesRead += bytesread;
                Trace ("%d bytes read", bytesread);
                break;
            }
        prevChunk -> myBytesRead += CurrentChunk -> myBytesRead;
        }

    delete [] CurrentChunk;
    CurrentChunk = prevChunk;
    Trace ("ProcessNextObjectChunk exited");
    }

/*******************************************************************************/

void ReadVertices (SChunk * prevChunk)
    {
    prevChunk -> myBytesRead += fread (&PointNum, 1, 2, ImportFile);
    Points = new CPoint3D [PointNum];
    prevChunk -> myBytesRead += fread (Points, 1, prevChunk -> myLength - prevChunk -> myBytesRead,
ImportFile);
    Trace ("Vertices read: %d", PointNum);
    for (int n = 0; n < PointNum; n ++)
        {
        Trace ("Vertex %d is %d, %d, %d", n, (int) Points [n] (0), (int) Points [n] (1), (int) Points [n]
(2));
        }
    }

/*******************************************************************************/

void ReadIndices (SChunk * prevChunk)
    {
    prevChunk -> myBytesRead += fread (&FaceNum, 1, 2, ImportFile);
    Faces = new SIndices [FaceNum];
    Polys = new CPolygon [FaceNum];
    prevChunk -> myBytesRead += fread (Faces, 1, sizeof (SIndices) * FaceNum, ImportFile);

    Trace ("Indices read: %d", FaceNum);
    for (int n = 0; n < FaceNum; n ++)
        {
        Trace ("Indices %d are %d, %d, %d", n, (int) Faces [n].myA, (int) Faces [n].myB, (int) Faces
[n].myC);
        }
    }

/*******************************************************************************/

void ReadTexCoords (SChunk * prevChunk)
    {
    prevChunk -> myBytesRead += fread (&CoordNum, 1, 2, ImportFile);
    TexCoords = new CTexCoord [CoordNum];
    prevChunk -> myBytesRead += fread (TexCoords, 1, sizeof (CTexCoord) * CoordNum, ImportFile);

    Trace ("texture coord read: %d", CoordNum);
    for (int n = 0; n < CoordNum; n ++)
        {
```

```
        Trace ("TexCoords %d are %d, %d", n, (int) TexCoords [n].myX, (int) TexCoords [n].myY);
        }
    }

/*************************************************************************/

void ComputeNormals ()
    {
    for (int n = 0; n < FaceNum; n ++)
        {
        CPoint3D norm,
                 vec1 = Polys [n] (1).RetPoint () - Polys [n] (2).RetPoint (),
                 vec2 = Polys [n] (1).RetPoint () - Polys [n] (0).RetPoint ();
        norm = vec1.CrossProduct (vec2);
        norm.Normalize ();
        Polys [n] (0).SetNormal (norm (0), norm (1), norm (2));
        Polys [n] (1).SetNormal (norm (0), norm (1), norm (2));
        Polys [n] (2).SetNormal (norm (0), norm (1), norm (2));
        }
    }

/*************************************************************************/

void SmoothNormals ()
    {
    CPoint3D * normals [10], result (0.0F, 0.0F, 0.0F);
    int J [10], K [10];
    int num = 0;

    for (int n = 0; n < FaceNum; n ++)
        for (int i = 0; i < 3; i ++)
            {
            for (int j = n + 1; j < FaceNum; j ++)
                for (int k = 0; k < 3; k ++)
                    if (Polys [n] (i).RetPoint () (0) == Polys [j] (k).RetPoint () (0) &&
                        Polys [n] (i).RetPoint () (1) == Polys [j] (k).RetPoint () (1) &&
                        Polys [n] (i).RetPoint () (2) == Polys [j] (k).RetPoint () (2) &&
                        num < 10)
                        {
                        normals [num] = &(Polys [j] (k).RetNormal ());
                        J [num] = j; K [num] = k;
                        num ++;
                        }
            result = Polys [n] (i).RetNormal ();
            for (int l = 0; l < num; l ++)
                result += *(normals [l]);
            result /= (num + 1);
            result.Normalize ();
            Polys [n] (i).SetNormal (result (0), result (1), result (2));
            for (l = 0; l < num; l ++)
                Polys [J [l]] (K [l]).SetNormal (result (0), result (1), result (2));
            result.Init (0.0F, 0.0F, 0.0F);
            num = 0;
            }
    }

/*************************************************************************/

int _tmain(int argc, _TCHAR* argv[])
    {
    CreateDumpFile ("3dsdump.txt");
    try
        {
        AssertT (argc == 3, "Not equal number of parameters in command line");
        ImportFileName = argv [1];
        SaveFileName   = argv [2];
        HMODULE LoaderDll = LoadLibrary ("Loader.dll");
        Assert (LoaderDll);
        typedef void (*DLLGETINSTANCE)(void**);
        DLLGETINSTANCE LoaderGetInstance = (DLLGETINSTANCE)GetProcAddress (LoaderDll, "DllGetInstance");
        Assert (LoaderGetInstance);
        LoaderGetInstance ((void**)&Loader);
        Loader -> LoadComponent ("STM32.dll");
        Loader -> InitModules ();
        Loader -> GetInterface (IID_IStorageMaster, (void **) &Stm);
        Stm -> Init (5);
        Import3DSFile (ImportFileName);

        #define SAFE_RELEASE(p) if (p) p->Release();
        SAFE_RELEASE (Stm);
        SAFE_RELEASE (Loader);
        #undef SAFE_RELEASE
        EndDump ();
        }
    Catch ()};
    return 0;
    }
```

Файл AgentsServer\AgentsServerCore.cpp

```
// AgentsServerCore.cpp - implementation of AgentsServerCore.h, of module AgentsServer
// created 23.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AgentsServer"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "AgentsServerCore.h"

#include <Exeption.h>      // use ExeptionEngine
#include <stdarg.h>

#define START_SIZE 5

#pragma warning (disable:4244)

//-----------------------------------------------------------------------------

CAgentsServer::CAgentsServer ()
{
    m_agentsSize = START_SIZE;
    m_agents = new IAgent* [m_agentsSize];
    m_creators = NULL;
    m_creatorsSize = 0;
    m_numAgents = 0;
    m_refs = 0;
    ZeroMemory (m_agents, m_agentsSize * sizeof (IAgent*));
}
//-----------------------------------------------------------------------------
CAgentsServer::~CAgentsServer()
{
    if (m_agents) delete[] m_agents;
    if (m_creators)
        {
        for (int i = 0; i < m_creatorsSize; i++)
            if (m_creators [i])
                delete m_creators[i];
        delete m_creators;
        }
}
//-----------------------------------------------------------------------------
int CAgentsServer::GetNumAgents ()
    { return m_agentsSize;}
//-----------------------------------------------------------------------------
void CAgentsServer::Terminate ()
{
    if (m_agents)

    for (int i = 0; i < m_agentsSize; i++)
        if (m_agents [i])
            {
            try {m_agents[i]->m_ShouldExit = true;}
                catch (...){}
            m_numAgents--;
            }
}
//-----------------------------------------------------------------------------
int     CAgentsServer::ok          ()
    {return (m_agents && m_agentsSize && m_creators);}

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
int     CAgentsServer::AddAgent    (IAgent* a, float x, float y, float z)
{
    AssertC (a, badParams);
    for (int i = 0; i < m_agentsSize - 1; i++)
        if (!m_agents[i])
            {
            m_agents[i] = a;
            a->m_mesh = a->m_physHash = a->m_index = i;
            m_numAgents++;
            a->m_x = x;
            a->m_y = y;
            a->m_z = z;
            return i;
            }
    m_agents[i] = a;
    a->m_index = i;
    a->m_mesh = a->m_physHash = a->m_index = i;
    m_numAgents++;
    a->m_x = x;
    a->m_y = y;
    a->m_z = z;
    Grow();
    return i;
}
//-----------------------------------------------------------------------------
int     CAgentsServer::RemoveAgent (int i)
{
    if (i < 0 || i > m_agentsSize) return false;
```

```
    if (m_agents[i])
        {
        m_agents [i]->m_ShouldExit = true;
        m_agents [i]->m_index = -1;
        m_agents [i] = NULL;
        return true;
        }
    return false;
}
//-----------------------------------------------------------------------------

IAgent* CAgentsServer::GetAgent       (int i)
{
    AssertC (i >= 0 && i < m_agentsSize, badParams);
    return m_agents[i];
}
//-----------------------------------------------------------------------------

int     CAgentsServer::FindAgent      (IAgent* a)
{
    AssertC (a, badParams);
    for (int i = 0; i < m_agentsSize; i++)
        if (a == m_agents[i])
            return i;
    return -1;
}
//-----------------------------------------------------------------------------

void    CAgentsServer::ReHash     ()
{
    for (int i = 0; i < m_agentsSize; i ++)
        if (m_agents[i])
            m_agents[i]->m_index = i;
    return;
}
//-----------------------------------------------------------------------------

int     CAgentsServer::Grow       ()
{
    if (m_agentsSize > 1)
        return GrowFor (m_agentsSize / 2);
    return GrowTo (4);
}
//-----------------------------------------------------------------------------

int     CAgentsServer::GrowFor    (int s)
{
    return GrowTo (m_agentsSize + s);
}
//-----------------------------------------------------------------------------

int     CAgentsServer::GrowTo     (int s)
{
    IAgent** temp = new IAgent*[s];
    AssertC (temp, memoryLeak);
    ZeroMemory (temp, sizeof (IAgent*) * s);

    if (!m_agents)
        {
        m_agents = temp;
        m_agentsSize = s;
        return true;
        }

    for (int i = 0; i < m_agentsSize; i++)
        if (m_agents[i])
            temp[i] = m_agents[i];
    m_agents = temp;
    m_agentsSize = s;
    return true;
}
//-----------------------------------------------------------------------------
ULONG STDMETHODCALLTYPE CAgentsServer::AddRef ()
    {
    return InterlockedIncrement (&m_refs);
    }
//-----------------------------------------------------------------------------

ULONG STDMETHODCALLTYPE CAgentsServer::Release ()
    {
    if (InterlockedDecrement (&m_refs) == 0)
        {
        delete this;
        return 0;
        }
    return m_refs;
    }
//-----------------------------------------------------------------------------
HRESULT STDMETHODCALLTYPE CAgentsServer::QueryInterface (const IID &iid, void ** ptr)
    {
    if (iid == IID_IUnknown)
        {
        *ptr = static_cast <IUnknown*> (this);
        }
    else
        {
        if (iid == IID_IAgentsServer)
```

```
            {
            *ptr = static_cast <IAgentsServer*> (this);
            }
        else if (iid == IID_IAgentsServerAgents)
            {
            *ptr = static_cast <IAgentsServerAgents*> (this);
            }
            else
                {
                *ptr = NULL;
                return E_NOINTERFACE;
                }
        }
    reinterpret_cast <IUnknown *> (* ptr) -> AddRef ();
    return S_OK;
    }
//------------------------------------------------------------------------------------
//------------------------------------------------------------------------------------
//------------------------------------------------------------------------------------

float   CAgentsServer::GetLandscapeLevel   (float x, float y, int horizont)
    {
    Master->Lock ("Landscape");
    //CSmartPointer land = Master->RetData ("Landscape");
    CLandscape* land = (CLandscape*)(Master->RetData ("Landscape").RetObject (0));
    float lsx = land -> RetHorizont (0).RetX () / 2.0;
    float lsy = land -> RetHorizont (0).RetY () / 2.0;
    float ss = land -> RetHorizont (0).RetSquareSize ();

    if (x > lsx * ss ||
        x < - (lsx * ss)) return 0.0;
    if (y > lsy * ss ||
        y < - (lsy * ss)) return 0.0;

    x /= ss;
    y /= ss;

    int x1 = (int)x, y1 = (int)y;
    int x2 = x1, y2 = y1;

    if (x1 > x) x2 = x1 - 1;
        else if (x1 < x) x2 = x1 + 1;
    if (y1 > y) y2 = y1 - 1;
        else if (y1 < y) y2 = y1 + 1;


    float a = land->RetHorizont (0).RetElement (x1 + lsx, y1 + lsy);
    float b = land->RetHorizont (0).RetElement (x2 + lsx, y2 + lsy);

    Master->Unlock ("Landscape");
    return ((a+b)/2);
    }

//------------------------------------------------------------------------------------

void    CAgentsServer::SetSpeed    (int agent, float x, float y, float z)
    {
    Environment->ChangeSpeed (agent, SIMPLE_SPEED, x, y, z);
    }
//------------------------------------------------------------------------------------

void    CAgentsServer::SetAcceleration (int agent, float x, float y, float z)
    {
    Environment->ChangeAcceleration (agent, 0, x, y, z);
    }
//------------------------------------------------------------------------------------

float   CAgentsServer::GetFallingAcceleration  ()
    {
    return 9.8F; //return Environment->RetGravity ();
    }
//------------------------------------------------------------------------------------
int     CAgentsServer::SendMessage         (AgentMessage msg)
{
    switch (msg.type)
        {
        case msg.zMT_BROADCAST:
            {
            if (msg.ainame < 0)
                for (int i = 0; i < m_agentsSize; i++)
                    try
                    {
                    if (i == msg.sender) continue;
                    if (m_agents [i])
                        m_agents[i]->MessageProc (&msg);
                    } catch (...){continue;}
            else
                for (int i = 0; i < m_agentsSize; i++)
                    try
                    {
                    if (i == msg.sender) continue;
                    if (m_agents [i])
                        if (m_agents [i]->m_ainame == msg.ainame)
                            m_agents[i]->MessageProc (&msg);
                    } catch (...){continue;}
            break;
```

```
            }

        case msg.zMT_SOUND:
            for (int i = 0; i < m_agentsSize; i++)
                try
                {
                if (i == msg.sender) continue;
                if (m_agents [i])
                    m_agents[i]->MessageProc (&msg);
                } catch (...){continue;}

            break;

        case msg.zMT_TARGETED:
            if (msg.receiver == msg.sender)
                {
                SayToUser ("Agent %d tryed to post message to itself\n", msg.sender);
                break;
                }

            if (msg.receiver >= 0 && msg.receiver < m_agentsSize)
                if (m_agents [msg.receiver])
                    m_agents [msg.receiver]->MessageProc (&msg);
            break;

        case msg.zMT_TOUSER:
            SayToUser ("Agent %d 've posted message to user...\n", msg.sender);
            break;

        case msg.zMT_NONE:
        default:
            //Unknown message type..
            break;
        };

    if (msg.freedata && msg.data)
        delete msg.data;

    return true;
}

//-----------------------------------------------------------------------------

void    CAgentsServer::PlayAnimation       (int agent, int animation, void* params)
    {if (Console) Console->printf ("Agent %d plays animation %d\n", agent, animation);}

//-----------------------------------------------------------------------------

void    CAgentsServer::PlaySound           (int agent, int soundcode)
    {
    //if (Console) Console->printf ("Agent %d plays sound %d\n", agent, soundcode);
    if (Sound)
        Sound->Play (soundcode);
    }
//-----------------------------------------------------------------------------
void    CAgentsServer::PlaySound           (int agent, char* sn)
{
    if (Sound)
        Sound->Play (sn);
}
//-----------------------------------------------------------------------------

void    CAgentsServer::SayToUser           (int agent, char* str)
    {if (Console) Console->printf ("Agent %d says: %s\n", agent, str);}

//-----------------------------------------------------------------------------

void    CAgentsServer::SayToUser           (char* format, ...)
    {
    va_list list;
    va_start (list, format);
    if (Console) Console->vprintf (format, list);
    va_end (list);
    }

//-----------------------------------------------------------------------------

IAgentsServerAgents* CAgentsServer::GetAgentsInterface ()
    {return static_cast <IAgentsServerAgents*> (this);}

//-----------------------------------------------------------------------------

float CAgentsServer::GetMaxR ()
    { return 20.0;}

//-----------------------------------------------------------------------------

void    CAgentsServer::AddToAcceleration   (int agent, float x, float y, float z)
    { Environment->AddToAcceleration (agent, 0, x, y, z);}

//-----------------------------------------------------------------------------

void    CAgentsServer::AddToSpeed          (int agent, float x, float y, float z)
    { Environment->AddToSpeed (agent, SIMPLE_SPEED, x, y, z);}

//-----------------------------------------------------------------------------
```

```
void    CAgentsServer::GetSpeed           (int agent, float*x, float*y, float*z)
{
    CSpeed* speed = Environment->RetSpeed (agent);
    if (x) *x = speed->RetPoint (0).retfloat (0);
    if (y) *y = speed->RetPoint (0).retfloat (1);
    if (z) *z = speed->RetPoint (0).retfloat (2);
}

//---------------------------------------------------------------------------
void    CAgentsServer::GetAcceleration    (int agent, float*x, float*y, float*z)
{
    CAcceleration* acceleration = Environment->RetAcceleration (agent, 0);
    if (x) *x = acceleration->RetPoint ().retfloat (0);
    if (y) *y = acceleration->RetPoint ().retfloat (1);
    if (z) *z = acceleration->RetPoint ().retfloat (2);
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

int     CAgentsServer::GetMapUnitCode     (float x, float y)
    const {return Map->GetUnitCode (x, y);}

//---------------------------------------------------------------------------

int*    CAgentsServer::GetPMapUnitData    (float x, float y)
    const {return Map->GetPUnitData (x, y);}

//---------------------------------------------------------------------------

int     CAgentsServer::GetMapUnitData     (float x, float y, int index)
    const {return Map->GetUnitData (x, y);}

//---------------------------------------------------------------------------

int     CAgentsServer::GetMapUnitDataSize  (float x, float y)
    const {return Map->GetUnitDataSize (x, y);}

//---------------------------------------------------------------------------

void    CAgentsServer::ModifyMapUnitData   (float x, float y, int index, int newvalue)
    {return Map->ModifyUnitData (x, y, index, newvalue);}

//---------------------------------------------------------------------------

int     CAgentsServer::GetMapSizeX        (void)
    const {return Map->GetSizeX ();}

//---------------------------------------------------------------------------

int     CAgentsServer::GetMapSizeY        (void)
    const {return Map->GetSizeY ();}

//---------------------------------------------------------------------------

float   CAgentsServer::GetMapSquareSize       (void)
    const {return Map->GetSquareSize ();}

//---------------------------------------------------------------------------

void CAgentsServer::PerformAgentsStep     (void)
{
    int errorcounter = 0;

    for (int i = 0; i < m_agentsSize; i++)
        {
        try {
            if (m_agents[i])
                m_agents[i]->Go ();
            }
        catch (agentPassTurn)
            {continue;}
        catch (...)
            {
            if (!m_ef)
                {
                Console->printf ("Agent %d 've encountered a problem.\n", i);
                m_ef = 1;
                }
            errorcounter ++;
            continue;
            }
        }
    if (errorcounter && !m_ef)
        {
        Console->printf ("AgentsServer: there was %d errors during AI step.\n", errorcounter);
        m_ef = 1;
        }
}
//---------------------------------------------------------------------------

void CAgentsServer::InitializeAgents      (void)
{
    int errorcounter = 0;

    for (int i = 0; i < m_agentsSize; i++)
        {
        try {
```

```
        if (m_agents[i])
            m_agents[i]->Init ();
            }
        catch (...)
            {
            errorcounter ++;
            continue;
            }
        }
    if (errorcounter)
        Console->printf ("AgentsServer: there was %d errors during AI initialization.\n", errorcounter);
}
//---------------------------------------------------------------------------

int CAgentsServer::RegisterAgentCreator    (PAgentCreator func, char* ainame, char* progr, int)
{
    Assert (func);
    int place = -1;

    if ((place = FindAgentCreatorByName (ainame)) >= 0)
        return place;

    place = FindFreeCreatorPlace ();
    AgentCreatorRef* ref = new AgentCreatorRef;
    ref->func = func;
    ref->index = place;
    ref->type = ref->Function;
    strcpy (ref->ainame, ainame? ainame: "unknown");
    strcpy (ref->programmer, progr? progr: "unknown");
    m_creators [place] = ref;
    return place;
}

//---------------------------------------------------------------------------

int CAgentsServer::RegisterAgentCreator    (char* command, char* ainame, char* progr)
    {
    Assert (command);

    int place = -1;

    if ((place = FindAgentCreator (command)) >= 0)
        return place;

    place = FindFreeCreatorPlace ();
    AgentCreatorRef* ref = new AgentCreatorRef;
    Assert (ref);
    ref->index = place;
    ref->type = ref->Function;
    strcpy (ref->ainame, ainame? ainame: "unknown");
    strcpy (ref->programmer, progr? progr: "unknown");
    strcpy (ref->command, command);

    m_creators [place] = ref;

    Message ("Interpreted agent creator registered. Interpreted agents are not supported yet.");
    return place;
    }

//---------------------------------------------------------------------------

int CAgentsServer::UnRegisterAgentCreator  (int index)
{
    if (index < 0 || index >= m_creatorsSize)
        return false;
    if (m_creators [index])
        delete m_creators [index];
    m_creators [index] = NULL;
    return true;
}

//---------------------------------------------------------------------------

int CAgentsServer::FindAgentCreator        (PAgentCreator func)
{
    for (int i = 0; i < m_creatorsSize; i++)
        if (m_creators [i])
            if (m_creators [i]->func == func)
                return i;
    return -1;
}

//---------------------------------------------------------------------------

int CAgentsServer::FindAgentCreator        (char* command)
{
    for (int i = 0; i < m_creatorsSize; i++)
        if (m_creators [i])
            if (!strcmp (command, m_creators [i]->command))
                return i;
    return -1;
}

//---------------------------------------------------------------------------

int CAgentsServer::FindAgentCreatorByName  (char* ainame)
{
```

```
    for (int i = 0; i < m_creatorsSize; i++)
        if (m_creators [i])
            if (!strcmp (ainame, m_creators [i]->ainame))
                return i;
    return -1;
}

//---------------------------------------------------------------------------

int CAgentsServer::CreateAgent            (int agentCreator, float x, float y, float z, int reserved)
{
    AssertTC (agentCreator >= 0 && agentCreator < m_creatorsSize, badParams, "CreateAgent(rubbish)");

    if (m_creators [agentCreator])
        if (m_creators [agentCreator]->type = AgentCreatorRef::Function)
            if (m_creators [agentCreator]->func)
                {
                try
                    {
                    IAgent* agent = m_creators [agentCreator]->func (agentCreator);
                    if (!agent) Throw ("CreateAgent: AgentCreator failed to create agent"));
                    agent->m_ainame = agentCreator;
                    return AddAgent (agent, x, y, z);
                    }
                catch (...)
                    {Throw ("CreateAgent: Cannot create agent."));}
                }
            else {Throw ("CreateAgent: Interpreted agents are not supported yet."));}


    Throw ("CreateAgent: AgentFactory does not exist."));
}

//---------------------------------------------------------------------------

int CAgentsServer::GrowCreators ()
    {return GrowCreatorsFor (1);}

//---------------------------------------------------------------------------

int CAgentsServer::GrowCreatorsFor (int n)
    {return GrowCreatorsTo (m_creatorsSize + n);}

//---------------------------------------------------------------------------

int CAgentsServer::GrowCreatorsTo  (int newsize)
{
    if (newsize <= m_creatorsSize) return false;

    AgentCreatorRef** temp = NULL;

    try {
        temp = new AgentCreatorRef*[newsize];
        if (!temp) return false;
        ZeroMemory (temp, sizeof (AgentCreatorRef*)*newsize);

        if (m_creators)
            for (int i = 0; i < m_creatorsSize; i++)
                temp[i] = m_creators [i];
        }
    catch (...)
        {return false;}

    if (m_creators) delete m_creators;

    m_creators = temp;
    m_creatorsSize = newsize;
    return true;
}

//---------------------------------------------------------------------------

int     CAgentsServer::GetMaxCreators  (void)
    {return m_creatorsSize;}

//---------------------------------------------------------------------------

int     CAgentsServer::FindFreeCreatorPlace ()
{
    if (!m_creators)
        {
        m_creatorsSize = 0;
        Assert (GrowCreatorsTo (1));
        return 0;
        }

    int i = 0;
    for (i = 0; i < m_creatorsSize; i++)
        if (!m_creators [i]) return i;

    GrowCreatorsFor (1);
    return i;
}

//---------------------------------------------------------------------------
//Se
const char*     CAgentsServer::GetAgentCreatorName     ( int ai)
```

```
    {
        if ( !m_creators)
            return 0;
        if ( ai < 0 || ai >= m_creatorsSize)
            return 0;
        return m_creators[ai]->ainame;
    }

//-------------------------------------------------------------------------------------
void    CAgentsServer::SetAgentPosition       (int agent, float x, float y, float z)
{
    IAgent* a = GetAgent (agent);
    if (!a) return;
    Master -> Lock ("Mesh2");
    a->m_x = x;
    a->m_y = y;
    a->m_z = z;
    ((CMesh*) (Master -> RetObject ("Mesh2", a->m_mesh))) -> SetCenter (x, y, z);
    Master -> Unlock ("Mesh2");
}
```

Файл AgentsServer\Module.cpp

```
// Module.cpp - implementation of Module, of module AgentsServer
// created 23.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AgentsServer"

#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine
#include "AgentsServerCore.h"

    #define DLL_NAME "Agents Server"
    #define DLL_CLASS CAgentsServer
    #include "..\Common\Utilities\Module.c"

#include "System.h"

DLL_ONLOAD (loader)
{
    Loader = loader;
    Assert (Loader);
    Loader->GetInterface (IID_IConsole,           (void**)&Console);

    if (Console)
        {
        Console->printf ("ASERVER: AgentsServer version 1.0 by Vasfed.\n");
        }
}

DLL_INIT (loader)
{
    Loader->GetInterface (IID_IAgentsMap,      (void**)&Map);
    Loader->GetInterface (IID_IStorageMaster,  (void**)&Master);
    Loader->GetInterface (IID_IEnvironment,        (void**)&Environment);
    Loader->GetInterface (IID_IAgentsServer,   (void**)&Server);
    Loader->GetInterface (IID_ISoundEngine,        (void**)&Sound);
    Assert (Map && Master && Environment && Server && Console);
}
```

Файл AgentsServer\System.cpp

```
// System.cpp - implementation of System.h, of module AgentsServer
// created 23.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AgentsServer"

#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine

ILoader*        Loader      = NULL;
IAgentsMap*     Map         = NULL;
IStorageMaster* Master      = NULL;
IEnvironment*   Environment = NULL;
IAgentsServer*  Server      = NULL;
IConsole*       Console     = NULL;
ISoundEngine*   Sound       = NULL;
```

November 14, 2008 12:17 AM -- 202

Файл AICore\AICore.cpp

```
//AICore.cpp - implementation of Agents.

//HINT: The main file in solution... (other - just for make it be executed)

#define __PROGRAMMER "Cool Vasfed"
#define __MODULE_NAME "Agents Core"

#include "stdafx.h"
#include "..\Utilities\Utilities.h"
#include "AICore.h"
#include <Exeption.h>

#include "system.h"

//--------------------------------------------------------------------------------

#define _Map(x,y) Map->GetUnit (x, y)
#define MapMask(x,y) m_mapMask[(int)(x)+m_mapMaskCx*(int)(y)]

void    CMovingAgent::Ping (int agent)
{
    m_vectorCode = random (8);
}

int GetNearestCode (Vector v)
{
    Vector v2 = v;
    v.Normalize ();

    if (v.x > 0.3) v.x = 1;
        else if (v.x < -0.3) v.x = -1;
            else v.x = 0;
    if (v.y > 0.3) v.y = 1;
        else if (v.y < -0.3) v.y = -1;
            else v.y = 0;

    if (v == Vector (-1,-1)) return 0;
    if (v == Vector (0 ,-1)) return 1;
    if (v == Vector (1 ,-1)) return 2;
    if (v == Vector (1 , 0)) return 3;
    if (v == Vector (1 , 1)) return 4;
    if (v == Vector (0 , 1)) return 5;
    if (v == Vector (-1, 1)) return 6;
    if (v == Vector (-1, 0)) return 7;

    return 0;
}

#define SetVector(code,speed) {switch (code)\
        {\
        case 0: m_vector = Vector (-speed  ,-speed); break;\
        case 1: m_vector = Vector (0       ,-speed); break;\
        case 2: m_vector = Vector (speed   ,-speed); break;\
        case 3: m_vector = Vector (speed   , 0    ); break;\
        case 4: m_vector = Vector (speed   , speed); break;\
        case 5: m_vector = Vector (0       , speed); break;\
        case 6: m_vector = Vector (-speed  , speed); break;\
        case 7: m_vector = Vector (-speed  , 0    ); break;\
        default: \
            m_vectorCode = 0;\
            break;\
        }m_vectorCode = code;}

//--------------------------------------------------------------------------------

class movingAgentMovementDirectionFound{};

void CMovingAgent::PerformMovement ()
{
    /*
    └ухэЄ ёЄюшЄ эр (m_x, m_y), яЁшўхь т ёхЁхфшэх Єрцыр (шыш ёсюъЄ, эхтрцэю)
    ярЁьхЄЁ√ фтшцхэш  шчтхёЄэ√
    */
    if (m_movementState > 0)
        if (m_oldX == m_x)
            if (m_oldY == m_y)
            {
            MapUnit* m = NULL;
            if (m = Map->GetUnit (m_x + m_vector.x, m_y + m_vector.y))
                {
                switch (m->code & 0xf)
                    {
                    case EMC_Castle:
                    case EMC_Empty:
                    case EMC_House:
                    case EMC_Rock:
                        break;
                    default: throw agentPassTurn(); return; break;
                    };
                }
            }
```

```
m_oldX = m_x;
m_oldY = m_y;

switch (m_movementState)
    {
    case this->MSC_NONE:
        break;


    case this->MSC_RANDOM1:
        m_vector.x = random (1.0);
        m_vector.y = random (1.0);
        Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, m_vector.x, m_vector.y, random (3.0));
        break;


    case this->MSC_RANDOM2:
        {
        int v = m_vectorCode;

        MapUnit* m = NULL;

        if (m = Map->GetUnit (m_x + m_vector.x, m_y + m_vector.y))
            {
            if (m->code & 0x10)
                {
                m_vectorCode += 4;
                if (m_vectorCode >= 8) m_vectorCode -= 8;
                }
            switch (m->code & 0xf)
                {
                case EMC_Castle:
                case EMC_Empty:
                case EMC_House:
                case EMC_Rock:
                    m_vectorCode += 4;
                    if (m_vectorCode >= 8) m_vectorCode -= 8;
                    break;
                };
            }
        else {m_vectorCode += 4; if (m_vectorCode >= 8) m_vectorCode -= 8; }

        SetVector (m_vectorCode, 1);
        if (v != m_vectorCode)
        Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, m_vector.x, m_vector.y, 2.0F);
        return;
        }
        break;


    case this->MSC_GOTO:
        {
        Vector dest (m_destinationX - m_x, m_destinationY - m_y);
        if (dest.Len () < 0.1)
            {
            Console->printf ("Agent %d reached (%d,%d).", m_index, m_destinationX, m_destinationY);
            Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, 0, 0, 0);
            MovementRandom2 ();
            }

        MapMask (m_x, m_y)++;
        MapUnit* m = NULL;
        m_vectorCode = GetNearestCode (dest);
        SetVector (m_vectorCode, 1);

        try {
            int min = 8;
            int mind = 0;
            for (int i = 0; i <= 8; i++)
                {
                if (m = Map->GetUnit (m_x + m_vector.x, m_y + m_vector.y))
                    {
                    if  ((m->code&0xf) == EMC_Castle ||
                         (m->code&0xf) == EMC_Empty ||
                         (m->code&0xf) == EMC_House ||
                         (m->code&0xf) == EMC_Rock ||
                         m->code & 0x10 ||
                         MapMask (m_x + m_vector.x, m_y + m_vector.y))
                             {
                             if (MapMask (m_x + m_vector.x, m_y + m_vector.y) < min)
                                 mind = i;

                             Assert (m_searchDirection != 0);
                             m_vectorCode += m_searchDirection;
                             while (m_vectorCode <= 0) m_vectorCode += 8;
                             while (m_vectorCode >= 8) m_vectorCode -= 8;
                             SetVector (m_vectorCode, 1);
                             continue;
                             }
                    throw movingAgentMovementDirectionFound();
                    }
                    else {Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, 0, 0, 0); throw;}
                }
            }
            catch (movingAgentMovementDirectionFound)
                {
```

```
                    Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, m_vector.x, m_vector.y, random
(2.0));
                    return;
                    }
                catch (...)
                    {
                    MovementRandom2 ();
                    return;
                    }
            }
            break;


        case this->MSC_PATROL:
            throw;
            Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, 0, 0, 0);
            break;
        };
}
//---------------------------------------------------------------------------------------
void CMovingAgent::CheckHealth () throw (agentPassTurn)
{
    if (m_ill)
        m_health -= random (m_ill);

    if (m_health <= 0 && m_shouldBeProcessed)
        {
        m_shouldBeProcessed = false;
        m_vector = Vector (0, 0);
        Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, 0, 0, 0);
        Console->printf ("  Agent %d died or killed", m_index);
        }
}

//---------------------------------------------------------------------------------------

int CMovingAgent::Go ()
{
    if (!m_shouldBeProcessed)
        throw agentPassTurn ();

    CheckHealth    ();
    CheckPosition ();
    PerformMovement ();
    Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, m_vector.x, m_vector.y, 2.0F);
    //AnalyzeLandscape ();
    return true;
}

void    CMovingAgent::CheckPosition    () throw (agentPassTurn)
{
    if (m_movementState > this->MSC_NONE)
        {
        int num = m_index;

        if (m_oldX == m_x)
            if (m_oldY == m_y)
                throw agentPassTurn();

        float x = (m_x - (int)m_x);
        float y = (m_y - (int)m_y);

        if (x < 0.4 || x > 0.6)
            if (y < 0.4 || y > 0.6)
                throw agentPassTurn();
        }
}


CMovingAgent::~CMovingAgent ()
{if (m_mapMask) {delete m_mapMask; m_mapMask = NULL;}}
//---------------------------------------------------------------------------------------

void CMovingAgent::Init()
{
    Assert (this);
    m_globalState = 0;
    m_vectorCode = random (8);
    m_vector = Vector (0);
    m_action = 0;
    m_globalState = 0;
    m_subState = 0;
    m_movementState = 0;
    m_isWaiting = 0;
    m_oldX = m_y, m_oldY = m_x;
    m_searchDirection = 1;
    m_destinationX = m_x, m_destinationY = m_y;
        //
    m_health = 100;
    m_ill = 0; //random (2)? 0.3:0;
    m_strength = 100;
    m_speed = 100;
    m_alignment = random (3) - 1;
    m_shouldBeProcessed = true;

    try{
    if (m_mapMask) {delete m_mapMask; m_mapMask = NULL;}
```

```
    }catch (...){}
    m_mapMaskCx = Map->GetSizeX ();
    m_mapMaskCy = Map->GetSizeY ();
    m_mapMask = new unsigned char [m_mapMaskCx * m_mapMaskCy + 1];
    ZeroMemory (m_mapMask, m_mapMaskCx * m_mapMaskCy + 1);

    MovementGoto (40, 35);
    //MovementRandom2 ();

    //MapMask (m_x, m_y)++;
    //Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, m_vector.x, m_vector.y, 2.0F);
}
//---------------------------------------------------------------------------

CMovingAgent::CMovingAgent(int x, int y, bool m)
    {
    int a;
    m_mapMask = NULL;

    if (m)
        {
        if (a = AServer->FindAgent(this))
            AServer->RemoveAgent (a);
        AServer->AddAgent (this);
        }
    }


//---------------------------------------------------------------------------
void CMovingAgent::MovementStop       (void)
    {
    m_movementState = MSC_NONE;
    m_vector = Vector (0);
    Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, 0, 0, 0);
    }

void CMovingAgent::MovementRandom1 (void)
    {
    m_movementState = MSC_RANDOM1;
    m_vectorCode = random (8);
    SetVector (m_vectorCode, 1);
    Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, m_vector.x, m_vector.y, random (3.0));
    PerformMovement ();
    }

void CMovingAgent::MovementRandom2 (void)
    {
    m_movementState = MSC_RANDOM2;
    m_vectorCode = random (8);
    SetVector (m_vectorCode, 1);
    Environment -> ChangeSpeed (m_index, SIMPLE_SPEED, m_vector.x, m_vector.y, random (3.0));
    PerformMovement ();
    }

void CMovingAgent::MovementGoto       (int x, int y)
    {
    m_destinationX = x;
    m_destinationY = y;
    m_movementState = MSC_GOTO;
    m_searchDirection = m_x - x > 0? -1: 1;
    if (m_y - y <= 0)m_searchDirection *= -1;

    ZeroMemory (m_mapMask, m_mapMaskCx * m_mapMaskCy + 1);
    PerformMovement ();
    }

void CMovingAgent::MovementPatrol     (int x1, int y1, int x2, int y2)
{throw;}
```

Файл AICore\ASystem.cpp

```
// ASystem.cpp - implementation of ASystem.h, of module AICore
// created 21.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AICore"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "ASystem.h"

#include <Exeption.h>      // use ExeptionEngine

IAgentsServerAgents* Server = NULL;

void    AgentsInit  ()
{
    Server = AServer->GetAgentsInterface ();
    Assert (Server);
}
```

Файл AICore\Console.cpp

```
// Console.cpp - implementation of Console.h, of module AICore
// created 11.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AICore"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Console.h"
#include <Exeption.h>      // use ExeptionEngine

//-----------------------------------------------------------------------------
void AddAgentCommands (IConsole* console, bool replace)
{
    Assert (console);

    //console->AddCommand ("", func, lexems);
}
```

Файл AICore\Creators.cpp

```
// Creators.cpp - implementation of Creators.h, of module AICore
// created 31.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AICore"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Creators.h"

#include <Exeption.h>      // use ExeptionEngine

void RegisterCreators (IAgentsServer* server)
{
    server->RegisterAgentCreator (CreateNewShip, "Ship", "Vasfed", 0);
    server->RegisterAgentCreator (CreateNewUfo, "Ufo", "Vasfed", 0);
}
```

Файл AICore\Module.cpp

```
// AgentsFactoryCore.cpp - implementation of AgentsFactoryCore.h
// created 10.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AgentsFactoryCore"

#include "StdAfx.h"      // use precompiled header
#include <Exeption.h>      // use ExeptionEngine
#include "system.h"
#include "Asystem.h"
#include "console.h"
#include "creators.h"

    #define DLL_NAME "AI: Ship&Ufo"
    #include "..\Common\Utilities\Module.c"
//-----------------------------------------------------------------------------

DLL_INIT(loader)
{
    Loader = loader;
    AServer = (IAgentsServer*)Loader->GetInterface (IID_IAgentsServer);
    Loader->GetInterface (IID_IConsole, (void**)&Console);

    Assert (Console);
    AssertT (AServer, "No agents server in system");

    RegisterCreators (AServer);
    AddAgentCommands (Console, true);
    AgentsInit ();
}
//-----------------------------------------------------------------------------
```

Файл AICore\Ship.cpp

```
// Ship.cpp - implementation of Ship.h, of module AICore
// created 21.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AICore"

#include "StdAfx.h"      // use precompiled header
#include "ASystem.h"
#include "math.h"
#include "Ship.h"
#include <mmsystem.h>
#include <Exeption.h>      // use ExeptionEngine

IAgent* CreateNewShip (int)
    {   return new CShip ();}

#define _Map(x,y) Map->GetUnit (x, y)
#define MapMask(x,y) m_mapMask[(int)(x + Server->GetMapSizeX()/2)+m_mapMaskCx*(int)(y + Server-
>GetMapSizeY()/2)]

void    CShip::Ping (int agent)
{
    m_vectorCode = random (8);
}

int GetNearestCode (Vector v)
{
    Vector v2 = v;
    v.Normalize ();

    if (v.x > 0.3) v.x = 1;
        else if (v.x < -0.3) v.x = -1;
            else v.x = 0;
    if (v.y > 0.3) v.y = 1;
        else if (v.y < -0.3) v.y = -1;
            else v.y = 0;

    if (v == Vector (-1,-1)) return 0;
    if (v == Vector (0 ,-1)) return 1;
    if (v == Vector (1 ,-1)) return 2;
    if (v == Vector (1 , 0)) return 3;
    if (v == Vector (1 , 1)) return 4;
    if (v == Vector (0 , 1)) return 5;
    if (v == Vector (-1, 1)) return 6;
    if (v == Vector (-1, 0)) return 7;

    return 0;
}

#define SetVector(code,speed) {switch (code)\
        {\
        case 0: m_vectorX = -speed; m_vectorY =    -speed; break;\
        case 1: m_vectorX = 0  ; m_vectorY =   -speed; break;\
        case 2: m_vectorX = speed; m_vectorY = -speed; break;\
        case 3: m_vectorX = speed; m_vectorY =  0     ; break;\
        case 4: m_vectorX = speed; m_vectorY =  speed; break;\
        case 5: m_vectorX = 0  ; m_vectorY =    speed; break;\
        case 6: m_vectorX = -speed; m_vectorY =     speed; break;\
        case 7: m_vectorX = -speed; m_vectorY =     0    ; break;\
        default: \
            m_vectorCode = 0;\
            break;\
        }m_vectorCode = code;}

//-------------------------------------------------------------------------------------

class movingAgentMovementDirectionFound{};

void CShip::PerformMovement ()
{
    /*
    ╚ухэЄ ёЄющЄ эр (m_x, m_y), яЁшўхь т ёхЁхфшэх Єрщыр (шыш ёсюъЄ, эхтрцэю)
    ярЁьрхЄЁ√ фтшцхэш  шчтхЁЄэ√
    */
    /*if (m_movementState > 0 && m_movementState != MSC_GOTO)
        if ((int)m_oldX == (int)m_x)
            if ((int)m_oldY == (int)m_y)
                if (Server->GetMapUnitCode (m_x + m_vectorX, m_y + m_vectorY) == EMC_Water)
                    throw agentPassTurn();
    */

    m_oldX = m_x;
    m_oldY = m_y;
    Server->ModifyMapUnitData (m_x, m_y, 0, timeGetTime ());

    switch (m_movementState)
        {
        case this->MSC_NONE:
            Server->SetSpeed (m_physHash, 0, 0, 0);
            break;
```

```
case this->MSC_RANDOM1:
    m_vectorX = random (1.0);
    m_vectorY = random (1.0);
    Server->SetSpeed (m_physHash, m_vectorX, m_vectorY, 0);
    break;


case this->MSC_RANDOM2:
    {
    MapMask (m_x, m_y)++;
    int v = m_vectorCode;

    int min = 8;
    int mind = 0;
    Assert (m_searchDirection != 0);
    for (int i = 0; i <= 8; i++)
        {
        if (Server->GetMapUnitCode (m_x + m_vectorX, m_y + m_vectorY) != EMC_Water)
            {
            m_vectorCode += m_searchDirection;
            while (m_vectorCode <= 0) m_vectorCode += 8;
            while (m_vectorCode >= 8) m_vectorCode -= 8;
            SetVector (m_vectorCode, 1);
            continue;
            }
        break;
        }
    //if (m_vectorCode != v)
        {
        SetVector (m_vectorCode, 1);
        Server->SetSpeed (m_index, m_vectorX, m_vectorY, 0);
        return;
        }
    }
    break;


case this->MSC_GOTO:
    {
    Vector dest (m_destinationX - m_x, m_destinationY - m_y);
    if (dest.Len () < 0.2)
        {
        Server->SayToUser ("Agent %d reached (%d,%d).", m_index, m_destinationX, m_destinationY);
        Server->SetSpeed (m_index, 0, 0, 0);
        MovementGoto (random (128) - 64, random (128) - 64);
        //MovementRandom2 ();
        }

    MapMask (m_x, m_y)++;
    m_vectorCode = GetNearestCode (dest);
    SetVector (m_vectorCode, 1);

    try {
        int min = 8;
        int mind = 0;
        Assert (m_searchDirection != 0);
        for (int i = 0; i <= 8; i++)
            {
            int code = Server->GetMapUnitCode (m_x + m_vectorX, m_y + m_vectorY);
            {
                if (Server->GetMapUnitCode (m_x + m_vectorX, m_y + m_vectorY) != EMC_Water ||
                          MapMask (m_x + m_vectorX, m_y + m_vectorY))
                        {
                        if (MapMask (m_x + m_vectorX, m_y + m_vectorY) < min)
                            mind = i;

                        m_vectorCode += m_searchDirection;
                        while (m_vectorCode <= 0) m_vectorCode += 8;
                        while (m_vectorCode >= 8) m_vectorCode -= 8;
                        SetVector (m_vectorCode, 1);
                        continue;
                        }
                throw movingAgentMovementDirectionFound();
                }

            }
        }
        catch (movingAgentMovementDirectionFound)
            {
            Server->SetSpeed (m_index, m_vectorX, m_vectorY, 0);
            return;
            }
        catch (...)
            {
            MovementRandom2 ();
            return;
            }
    }
    break;


case this->MSC_PATROL:
    throw;
    Server->SetSpeed (m_index, 0, 0, 0);
    break;
};
```

```
    float vx, vy;
    Server->GetSpeed (m_physHash, &vx, &vy);
    if (vx != m_vectorX || vy != m_vectorY)
        Server->SetSpeed (m_physHash, m_vectorX, m_vectorY, 0);

}
//-----------------------------------------------------------------------------------------
void CShip::CheckHealth () throw (agentPassTurn)
{
    if (m_ill)
        m_health -= random (m_ill);

    if (m_health <= 0 && m_shouldBeProcessed)
        {
        m_shouldBeProcessed = false;
        m_vectorX = 0; m_vectorY = 0;
        Server->SetSpeed (m_index, 0, 0, 0);
        Server->SayToUser ("   Agent %d died or killed", m_index);
        }
}

//-----------------------------------------------------------------------------------------

int CShip::Go ()
{
    if (!m_shouldBeProcessed)
        throw agentPassTurn ();

    //CheckHealth   ();
    //CheckPosition ();

    PerformMovement ();
    //Server->SetSpeed (m_index, m_vectorX, m_vectorY, 0);
    return true;
}
//-----------------------------------------------------------------------------------------

void    CShip::CheckPosition   () throw (agentPassTurn)
{
    if (m_movementState > this->MSC_NONE)
        {
        int num = m_index;

        if ((int)m_oldX == (int)m_x)
            if ((int)m_oldY == (int)m_y)
                throw agentPassTurn();

        //float x = (m_x - (int)m_x);
        //float y = (m_y - (int)m_y);

        /*if (x < 0.4 || x > 0.6)
            if (y < 0.4 || y > 0.6)
                throw agentPassTurn();*/
        }
}


CShip::~CShip ()
{if (m_mapMask) {delete m_mapMask; m_mapMask = NULL;}}
//-----------------------------------------------------------------------------------------

void CShip::Init()
{
    Assert (this);
    m_globalState = 0;
    m_vectorCode = random (8);
    m_vectorX = 0;
    m_vectorY = 0;
    m_action = 0;
    m_globalState = 0;
    m_subState = 0;
    m_movementState = 0;
    m_isWaiting = 0;
    m_oldX = m_y, m_oldY = m_x;
    m_searchDirection = 1;
    m_destinationX = m_x, m_destinationY = m_y;
        //
    m_health = 100;
    m_ill = 0; //random (2)? 0.3:0;
    m_strength = 100;
    m_speed = 100;
    m_alignment = random (3) - 1;
    m_shouldBeProcessed = true;

    try{
        if (m_mapMask) {delete m_mapMask; m_mapMask = NULL;}
        }
    catch (...){}
    m_mapMaskCx = Server->GetMapSizeX ();
    m_mapMaskCy = Server->GetMapSizeY ();
    m_mapMask = new unsigned char [m_mapMaskCx * m_mapMaskCy + 1];
    ZeroMemory (m_mapMask, m_mapMaskCx * m_mapMaskCy + 1);

    MovementGoto (-40 * 4, 20 * 4);

    //MovementRandom2 ();
```

```
    Server->SetAcceleration (m_index, 0, 0, Server->GetFallingAcceleration ());
}
//-----------------------------------------------------------------------------

CShip::CShip()
    {
    m_mapMask = NULL;
    }
//-----------------------------------------------------------------------------


//-----------------------------------------------------------------------------
void CShip::MovementStop       (void)
    {
    m_movementState = MSC_NONE;
    m_vectorX = m_vectorY = 0;
    Server->SetSpeed (m_index, 0, 0, 0);
    }
//-----------------------------------------------------------------------------

void CShip::MovementRandom1    (void)
    {
    m_movementState = MSC_RANDOM1;
    m_vectorCode = random (8);
    SetVector (m_vectorCode, 1);
    Server->SetSpeed (m_index, m_vectorX, m_vectorY, 0);
    PerformMovement ();
    }
//-----------------------------------------------------------------------------

void CShip::MovementRandom2    (void)
    {
    m_movementState = MSC_RANDOM2;
    m_vectorCode = random (8);
    SetVector (m_vectorCode, 1);
    Server->SetSpeed (m_index, m_vectorX, m_vectorY, 0);
    m_searchDirection = random (3) - 1;
    ZeroMemory (m_mapMask, m_mapMaskCx * m_mapMaskCy + 1);
    m_oldX = -1000.0F;
    m_oldY = -1000.0F;

    PerformMovement ();
    }
//-----------------------------------------------------------------------------

void CShip::MovementGoto       (int x, int y)
    {
    m_destinationX = x;
    m_destinationY = y;
    m_movementState = MSC_GOTO;
    m_searchDirection = fabs (m_x) - fabs (x) > 0? -1: 1;
    if (fabs (m_y) - fabs (y) <= 0) m_searchDirection *= -1;
    m_oldX = m_oldY = -1000.0F;
    ZeroMemory (m_mapMask, m_mapMaskCx * m_mapMaskCy + 1);
    PerformMovement ();
    }
//-----------------------------------------------------------------------------

void CShip::MovementPatrol     (int x1, int y1, int x2, int y2)
{throw;}
```

Файл AICore\System.cpp

```
// System.cpp - implementation of System.h, of module AICore
// created 31.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AICore"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine

ILoader*       Loader      = NULL;
IAgentsServer* AServer     = NULL;
IConsole*      Console     = NULL;
```

Файл AICore\Ufo.cpp

```
// Ufo.cpp - implementation of Ufo.h, of module AICore
// created 21.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00
```

```
#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AICore"

#include "StdAfx.h"      // use precompiled header
#define _USE_MATH_DEFINES
#include "Math.h"
#include "System.h"
#include "Ufo.h"
#include "MessageLanguages.h"
#include <mmsystem.h>

#include <Exeption.h>      // use ExeptionEngine

IAgent* CreateNewUfo (int)
    {   return new CUfo ();}

CUfo::CUfo ()
    {
    m_interestMin = 1;
    m_interestTimeMax = 5000; //msec
    m_currentInterest = 0;
    m_currentInterestTimeStart = 0;
    m_currentInterestTime = 0;
    m_currentAngle = 0;
    m_maxAcceleration = 3.0; //per direction, Ex. Vector (Acc, Acc, Acc) -> sqrt3*acc = module


    m_state = State_Browsing;
    m_oldstate = m_state;

    m_database = NULL;
    m_databaseSize = 0;
    m_health = 100.0;

    m_dx = 0.0;
    m_dy = 0.0;
    m_dz = 0.0;

    m_experience = 0;
    m_strenght = 100;
    m_dexterity = 100;
    m_wisdom = 50;
    m_charisma = 60;
    m_skill = 10;
    m_heightSensibility = 2.0;
    //m_Vector; //?
    //m_acceleration;
    }
//--------------------------------------------------------------------------------------

CUfo::~CUfo ()
    {
    if (m_database) delete m_database;
    }
//--------------------------------------------------------------------------------------

int CUfo::Go ()
{
    PerformMovement ();
    //Server->SayToUser ("%g: %g: %g", m_x, m_y, m_z);
    AnalyzeLandscape ();
    //CheckDatabase ();
    return true;
}
//--------------------------------------------------------------------------------------

int CUfo::MessageProc  (AgentMessage* msg)
{
    switch (msg->type)
        {
        case msg->zMT_BROADCAST:
            switch (msg->code)
                {
                case UFOMT_NONE: break;
                case UFOMT_DIED:
                    break;
                case UFOMT_FOUNDOBJECT:
                    break;
                case UFOMT_FLYHERE:
                    if (msg->data1 > m_currentInterest)
                        {
                        FlyTo (msg->x, msg->y, 10);
                        Server->SayToUser (m_index, "Fly to interesting...");
                        }
                    break;
                case UFOMT_MISC:
                    break;

                default: break;
                }
            break;
        case msg->zMT_TARGETED:

            break;
        case msg->zMT_SOUND:

            break;
```

```
        }
    return true;
}
//-----------------------------------------------------------------------------

void CUfo::Init        ()
{
    m_interestMin = 1;
    m_interestTimeMax = 5000; //msec
    m_currentInterest = 0;
    m_currentInterestTimeStart = 0;
    m_currentInterestTime = 0;
    m_currentAngle = 0;
    m_maxAcceleration = 3.0; //per direction, Ex. Vector (Acc, Acc, Acc) -> sqrt3*acc = module
    m_oldX = 0;
    m_oldY = 0;

    m_state = State_Browsing;
    m_oldstate = m_state;

    m_database = NULL;
    m_databaseSize = 0;
    m_health = 100.0;

    m_dx = 0.0;
    m_dy = 0.0;
    m_dz = 0.0;

    m_preferedFlyingHeight = 10;
    m_preferedExploringHeight = 5;

    m_oldVx = 0.0;
    m_oldVy = 0.0;
    m_oldVz = 0.0;
    m_heightFlag = 0;

    m_heightSensibility = 1.0 + random (2.0);

    Server->SetAcceleration (m_index, 0, 0, Server->GetFallingAcceleration ());
    Server->SetSpeed       (m_index, random (10.0) - 5.0, random (10.0) - 5.0, 0);

    //Server->SayToUser ("Agent %d: (%d, %d)", m_index, (int)m_x, (int)m_y);
    if (m_index == 1)
        {
        FlyTo (-100, -440, 10);
        //Server->SayToUser (m_index, "ship!");
        }
    else
        {
        m_state = State_Browsing;
        Server->SetSpeed (m_physHash, random (8.0) - 4.0, random (8.0) - 4.0);
        }
}
//-----------------------------------------------------------------------------

int CUfo::Ok               ()
    {return true;}
//-----------------------------------------------------------------------------

void CUfo::Ping        (int)
{}
//-----------------------------------------------------------------------------


void CUfo::PerformMovement ()
{
    //!!
    m_state = State_Browsing;

    switch (m_state)
        {
        case State_None:
            break;
        case State_Died:
            throw agentPassTurn();
            break;

        case State_Browsing:
            {
            //Server->SetAcceleration (m_index, 0, 0, Server->GetFallingAcceleration ());
            //Server->SetSpeed (m_index, );
            //FlyTo ();

            float maxr = 100 * 4; //Server->GetMaxR ();
            float vx = 0, vy = 0, vz = 0;
            Server->GetSpeed (m_index, &vx, &vy, &vz);

            if (vx == 0) vx = random (8.0) - 4.0;
            if (vy == 0) vy = random (8.0) - 4.0;

            if (m_x >= maxr || m_x <= (0-maxr)) vx = -vx;
            if (m_y >= maxr || m_y <= (0-maxr)) vy = -vy;

            if (m_speedFlag != 1 && (vx != m_oldVx || vy != m_oldVy /*|| vz != m_oldVz*/))
                {
                Server->SetSpeed (m_index, vx, vy, vz);
                m_speedFlag = 1;
```

```
                m_oldVx = vx;
                m_oldVy = vy;
                m_oldVz = vz;
                }
            else {m_speedFlag = 0;}
            }
            break;

        case State_Flyto:
            {
            if ((int)m_x == (int)m_dx)
                if ((int)m_y == (int)m_y)
                    {
                    //Server->SayToUser (m_index, );
                    Server->SayToUser ("Ufo %d reached %d:%d", m_index, (int)m_x, (int)m_y);
                    if (m_oldstate != State_Flyto) m_state = m_oldstate;
                    else m_state = State_Browsing;
                    }
                {
                float   vx = m_dx - m_x,
                        vy = m_dy - m_y,
                        vz = 0;
                float speed = 4.0;
                //now normalize vector
                if (vx == 0 || vy == 0)
                    {
                    if (vx == 0) vy = speed;
                    if (vy == 0) vx = speed;
                    }
                    else
                        {
                        if (fabs (vx) >= fabs (vy))
                            {
                            vy = (vy*speed) / fabs (vx);
                            vx = vx > 0? speed: -speed;
                            }
                        else
                            {
                            vx = (vx*speed) / fabs (vy);
                            vy = vy > 0? speed : -speed;
                            }
                        }
                //Server->SayToUser ("Vector found: %f, %f", vx, vy);

                Server->GetSpeed (m_index, NULL, NULL, &vz);
                if (timeGetTime () - m_time > 300)
                    {
                    m_time = timeGetTime ();
                    Server->SetSpeed (m_index, vx, vy, vz);
                    Server->SayToUser (m_index, "new vector");
                    }
                }
            float maxr = 100*4;
            if (m_x > maxr || m_x < - maxr || m_y > maxr || m_y < -maxr)
                {
                m_state = State_Browsing;
                }
            }
            break;

        case State_Exploring:
            {
            double rxy = sqrt (pow (m_x - m_dx, 2) + pow (m_y - m_dy, 2));

            double angle = asin ((m_y - m_dy) / rxy);
            if (m_x < m_dx)
                angle = M_PI - angle;
            /*єуюы сўшЄрхЄё  ъЁштю! (possibly)*/
            float v = 2.0;
            float vx = v * sin (angle);
            float vy = v * cos (angle);
            //float vz = random (1.0);

            Server->SetAcceleration (m_index, 0, 0, Server->GetFallingAcceleration ());
            Server->SetSpeed (m_index, vx, vy, 0);
            }
            break;

        }

    CheckHeight (m_state == State_Exploring);
    m_ox = m_x;
    m_oy = m_y;
    m_oz = m_z;
}
//-----------------------------------------------------------------------------

void CUfo::AnalyzeLandscape    ()
{
    if (timeGetTime () - m_currentInterestTimeStart > m_currentInterestTime)
        {
        if (m_state != State_Browsing)
            {
            //this->FlyTo ();
            m_state = State_Browsing;
            m_currentInterest = 1;
```

```
            m_currentInterestTimeStart = timeGetTime();
            m_currentInterestTime = 2000;
            Server->SayToUser (m_index, "nadoelo..");
            return;
            }
        }
    else
    {
    if (m_oldX == m_x && m_oldY == m_y)
        return;

    m_oldX = m_x;
    m_oldY = m_y;

    IAgent* ship = Server->GetAgent (4); //!!!
    float l = 30, l2 = 10;

    if (fabs (ship->m_x - m_x) < l2)
        if (fabs (ship->m_x - m_x) < l2)
            {
            this->FlyAround (ship->m_x, ship->m_y, 10);
            m_currentInterest = random (5);
            m_currentInterestTimeStart = timeGetTime ();
            m_currentInterestTime = 5000;
            CallOthers ();
            Server->SayToUser (m_index, "Ship found");
            }
    if (fabs (ship->m_x - m_x) < l)
        if (fabs (ship->m_x - m_x) < l)
            {
            this->FlyTo (ship->m_x, ship->m_y, 10);
            m_currentInterest = 0;
            m_currentInterestTimeStart = timeGetTime ();
            m_currentInterestTime = 5000;
            Server->SayToUser (m_index, "Ship found");
            }
    }
}
//-------------------------------------------------------------------------------------

void CUfo::CheckDatabase       (){}

//-------------------------------------------------------------------------------------

void CUfo::CallOthers          ()
{
    AgentMessage msg;
    ZeroMemory (&msg, sizeof (msg));

    msg.sender = m_index;
    msg.type = msg.zMT_BROADCAST;
    msg.ainame = m_ainame;

    msg.code = UFOMT_FLYHERE;
    msg.datasize = 0;
    msg.data1 = m_currentInterest;
    msg.x = m_x;
    msg.y = m_y;
    msg.freedata = false;
    Server->SendMessage (msg);
}

//-------------------------------------------------------------------------------------

void CUfo::FlyTo               (float x, float y, float z)
    {
    m_dx = x;
    m_dy = y;
    m_dz = z;
    m_oldstate = m_state;
    m_state = State_Flyto;

    float   vx = m_dx - m_x,
            vy = m_dy - m_y,
            vz = 0;
    float speed = 4.0;
    //now normalize vector
    if (vx == 0 || vy == 0)
        {
        if (vx == 0) vy = speed;
        if (vy == 0) vx = speed;
        }
        else
            {
            if (fabs (vx) >= fabs (vy))
                {
                vy = (vy*speed) / fabs (vx);
                vx = vx > 0? speed: -speed;
                }
            else
                {
                vx = (vx*speed) / fabs (vy);
                vy = vy > 0? speed : -speed;
                }
            }
    //Server->SayToUser ("Vector found: %f, %f", vx, vy);
```

```
    Server->GetSpeed (m_index, NULL, NULL, &vz);
    Server->SetSpeed (m_index, vx, vy, vz);
    }

//-----------------------------------------------------------------------------

void CUfo::FlyAround           (float x, float y, float z)
    {
    m_dx = x;
    m_dy = y;
    m_dz = z;
    }

//-----------------------------------------------------------------------------

void CUfo::CheckHeight     (bool exploring)
{
    float dh = 0;
    if (exploring) dh = m_z - (m_preferedExploringHeight + Server->GetLandscapeLevel (m_x, m_y));
    else dh = m_z - (m_preferedFlyingHeight + Server->GetLandscapeLevel (m_x, m_y));

    float delta = fabs (dh);

    if ((delta < m_heightSensibility / 2) && m_heightFlag != 0)
        {
        m_heightFlag = 0;
        float vx, vy, vz;
        Server->GetSpeed (m_index, &vx, &vy, &vz);
        Server->SetSpeed (m_index, vx, vy, random (0.2) - 0.1);
        //Server->SayToUser (m_index, "Up/Down normalized");
        return;
        }

    float speed = delta > 10? 10 : delta > 5? 5 : delta > 2? 2 : 1;

    if (dh > m_heightSensibility && m_heightFlag != 1)
        {
        //Server->AddToAcceleration (m_index, 0, 0, -2);
        float vx, vy, vz;
        Server->GetSpeed (m_index, &vx, &vy, &vz);
        Server->SetSpeed (m_index, vx, vy, /*vz*/ - speed);
        //Server->SayToUser (m_index, "Going down");
        m_heightFlag = 1;
        }
    if (dh < -m_heightSensibility && m_heightFlag != 2)
        {
        float vx, vy, vz;
        Server->GetSpeed (m_index, &vx, &vy, &vz);
        Server->SetSpeed (m_index, vx, vy, /*vz*/ + speed);
        //Server->AddToAcceleration (m_index, 0, 0, 2);
        //Server->SayToUser (m_index, "Going up");
        m_heightFlag = 2;
        }
}

//-----------------------------------------------------------------------------
```

Файл Console\Commands.cpp

```
// Commands.cpp - implementation of Commands.h
// created 13.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Console Basic Commands"

#include "StdAfx.h"      // use precompiled header
#include "Commands.h"

#include <Exeption.h>      // use ExeptionEngine

IVariableEngine* Variables = NULL;

#define COMMAND(name) int name(IConsole* Console, int nlexem, void* data)

COMMAND(type)
{
    Console->printf ("Type:\n%s\n", data);
    return true;
}

COMMAND(exit)
{
    Console->printf ("Exited.");
    return true;
}

COMMAND(MessageB)
{
    Console->printf ("MessageBox\n");

    MessageBox (NULL,
```

```
    nlexem > 0? (((Lexem*)data)[0].type == Lexem::LT_ConstString? ((Lexem*)data)[0].data:"<invalid
param>"): "<no text>",
    nlexem > 1? (((Lexem*)data)[0].type == Lexem::LT_ConstString?((Lexem*)data)[1].data:"<invalid
param>"): "<no title>",
    NULL);

    return true;
}

/*int set (CConsole* c, int nl, Lexem* l)
{
    Assert (nl > 1);
    Assert (l[0].type = Lexem::LT_String);
    Variable* v = c->GetVariable (-l[0].idata);
    if (!v) return printf ("%d - is not a variable", l[0].data);

    int i = 1;
    if (l[i].type == Lexem::LT_Operator)
        i++;

    switch (l[i].type)
        {
        case Lexem::LT_ConstString:
            v->type = v->String;
            v->value.c = new char[strlen(l[i].data)];
            strcpy (v->value.c, l[i].data);
            v->dp = true;
            break;
        case Lexem::LT_Number:
            v->type = v->Integer;
            v->value.i = l[i].idata;

            break;
        case Lexem::LT_Float:
            v->type = v->Float;
            v->value.f = l[i].ddata;
            break;
        default: printf ("Incorrect lexem\n"); break;
        };

    printf ("Variable set\n");
}

int print(CConsole* c, int nl, Lexem* l)
{
    Assert (nl>0);
    Assert (l[0].type = Lexem::LT_String);
    Variable* v = c->GetVariable (-l[0].idata);
    if (!v) return printf ("%d - is not a variable", l[0].data);

    switch (v->type)
        {
        case v->Float:
            printf ("Float %s = %g\n", l[0].data, v->value.f);
            break;
        case v->Integer:
            printf ("Integer %s = %d\n", l[0].data, v->value.i);
            break;
        case v->Pointer:
            printf ("Pointer %s = %p\n", l[0].data, v->value.p);
            break;
        case v->String:
            printf ("String %s = '%s'\n", l[0].data, v->value.c);
            break;
        case v->Unknown:
            printf ("Unknown %s = <unknown value type>\n", l[0].data);
            break;
        default:
            AssertT (NULL, "Error: unknown type of 'Var'");
            break;
        };
}

COMMAND (add)
{
    if (nlexem < 2) return Console->printf ("too few arguments\n");
    Lexem* lex = (Lexem*)data;
    CConsole* c = (CConsole*)Console;
    float num1 = 0;
    float num2 = 0;


    if (lex[0].type == Lexem::LT_String)
        {
        Variable* var  = c->GetVariable (-lex[0].idata);
        if (!var) return printf ("%s - is not a variable\n", lex[0].data);
        num1 = ((var->type == var->Float)? var->value.f:var->value.i);
        }
    else if (lex[0].type != Lexem::LT_Number && lex[0].type != Lexem::LT_Float)
        {
        return Console->printf ("first argument is not a number\n");
        }
        else {num1 = lex[0].ddata;}

    if (lex[1].type == Lexem::LT_String)
        {
```

```
        Variable* var  = c->GetVariable (-lex[1].idata);
        if (!var) return printf ("%s - is not a variable\n", lex[1].data);
        num2 = ((var->type == var->Float)? var->value.f: var->value.i);
        }
        else if (lex[1].type != Lexem::LT_Number && lex[1].type != Lexem::LT_Float)
            return Console->printf ("second argument is not a number\n");
            else {num2 = lex[1].ddata;}


    printf ("Result is %g.\n", num1 + num2);
    return true;
}

int __cdecl NewVar (CConsole* c, int nl, Lexem* data)
{
    if (!nl) return printf ("Usage: var \"varName\"\n");
    if ((*data).type != Lexem::LT_ConstString)
        return printf ("Usage: var \"varName\"\n");
    c->NewVariable ((*data).data);
    return true;
}*/
//-----------------------------------------------------------------------------------

void Help (IConsole* c, int ln, Lexem* data)
{
    if (!ln)
        {
        c->printf (" Type 'listc' for list of loaded commands\n");
        c->printf (" Type 'help <command name>' to get help on a command\n");
        c->printf ("  (some modules may not provide help on their's commands)\n");
        return;
        }

    c->printf (" The full version 'Help on command' is not avaible in current version of Console\n"
               " To get help on command see user's manual of module\n\n");
    CConsole* con = (CConsole*)c;
    int i = con->FindKeyword (data[0].data);
    if (i <= 0) c->printf (" No such command - %s\n", data[0].data);
    else
        {
        printf ("Command '%s' is exported from module '%s'\n", con->m_commands[i]->keyword, con-
>m_commands[i]->module);
        }
}

void ListCommands (IConsole*c, int, void*)
{
    CConsole* con = (CConsole*)c;
    c->printf (" Listing console commands.\n");
    int n = 0;
    for (int i = 1; i < con->m_sizeCommmands; i++)
        if (con->m_commands [i])
            {
            c->printf (" \t'%s',\tmodule '%s'\n", con->m_commands[i]->keyword, con->m_commands[i]-
>module);
            n++;
            }
    c->printf (" There's %d commands total.\n  You may use 'help <command name>' to get help on
command\n", n);
}
//-----------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------
void DisplLevel (IConsole*c, int ln, Lexem1* data)
{
    if (!ln)
        {
        c->printf (8, "Display level is %d\n", c->GetDisplayLevel ());
        return;
        }
    else
        if (data[0].type == data[0].LT_Number)
            {
            if (data[0].idata >= 0 && data[0].idata < 100)
                {
                c->SetDisplayLevel (data[0].idata);
                }
            else c->printf (5, "CONSOLE: DisplayLevel must be 0 - 100.\n");
            }
        else
        {
        c->printf (8, "USAGE: displaylevel <number>\n");
        }
}

void InitCommands (CConsole* console)
{
    console->AddCommand ("type", type, false, "Internal console commands");
    //console->AddCommand ("exit", exit, false, "Internal commands");
    //console->AddCommand ("msgbox", MessageB, true, "Internal console commands");
    //console->AddCommand ("add", add, true, "Internal console commands");
    //console->AddCommand ("var", (CommandProcP)NewVar, true, "Internal console commands");
    //console->AddCommand ("set", (CommandProcP)set, true, "Internal console commands");
    //console->AddCommand ("print", (CommandProcP)print, true, "Internal console commands");

    //console->AddCommand ("load", (CommandProcP)LoaderLoadModule, true, "Internal console commands");
```

```
    //console->AddCommand ("init", (CommandProcP)LoaderInitModules, false, "Internal console commands");
    console->AddCommand ("help", (CommandProcP)Help, true, "Internal console commands");
    console->AddCommand ("listc", (CommandProcP)ListCommands, false, "Internal console commands");
    console->AddCommand ("displaylevel", (CommandProcP)DisplLevel, true, "Internal console commands");
}
```

Файл Console\Console.cpp

```
// Console.cpp - implementation of Console.h
// created 09.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Console"

#include "StdAfx.h"      // use precompiled header
#include "Console.h"
#include <Exeption.h>      // use ExeptionEngine
#include <conio.h>
#include <stdio.h>
#include "Env.h"
#include "..\Loader\LoaderCore.h"

//-------------------------------------------------------------------------------------
//Create loader
ILoader* Loader = NULL;

//-------------------------------------------------------------------------------------

int main (...)
{
    CreateDumpFile ("ExeptionDump.txt");
    Loader = new CLoader;

    try {

    Assert (NULL);

    delete Loader;
    }
        catch (Exeption exeption)
            {
            printf ("\nAn Exeption occured: %s module by %s, %s exeption: %s \n",
                exeption.m_module, exeption.m_programmer, exeption.TypesStr[exeption.m_type],
exeption.m_text);
            }
        catch (...)
            {
            printf ("An Unknown exeption occured in system.\n\tExecution of last command may be
failed\n");
            }

    EndDump ();
    getch ();
    return NULL;
}
```

Файл Console\ConsoleCore.cpp

```
// ConsoleCore.cpp - implementation of ConsoleCore.h
// created 13.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Console"

#include "StdAfx.h"      // use precompiled header
#include "ConsoleCore.h"
#include "system.h"
#include "commands.h"
#include <Exeption.h>      // use ExeptionEngine

#define _WIN32_WINNT 0x0500
#include <windows.h>
#include <wincon.h>

#define NUM_BASIC_COMMANDS 0

int __cdecl StdOut (const char* format, va_list list)
    {
    //Output can be dubled in file
    return vfprintf (stdout, format, list);
    }

//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------
```

```
CConsole::CConsole ()
    {
    Assert (this);
    m_commands = NULL;
    m_sizeCommmands = 0;
    m_output = DEFAULT_OUTPUT;

    Grow ();
    InitCommands (this);
    m_displayLevel = 10;
    m_unhandledCommandHandler = NULL;
    m_unhandledCommandHandlerWantLexems = true;
    m_commandPrefix = NULL;
    m_Prompt = false;
    strcpy (m_PromptStr, "System>");
    }
//-----------------------------------------------------------------------------
CConsole::~CConsole (){if (m_commands) {delete m_commands; m_commands = NULL;}}
//-----------------------------------------------------------------------------
/*void CConsole::CreateDefaultWindow   ()
{
    FreeConsole ();
    if (AllocConsole ())
        {
        //m_window = GetConsoleWindow ();
        //SetWindowText (m_window, "System Console");
        SetConsoleTitle ("System Console");
        }
}*/
//-----------------------------------------------------------------------------
void CConsole::SetOutput (VprintfP out /*NULL = default*/)
{
    if (!out) out = DEFAULT_OUTPUT;
    try {
        if (!out) throw;
        out ("", NULL); //test output
        m_output = out;
        }
    catch (...)
        {m_output = DEFAULT_OUTPUT;}
}
//-----------------------------------------------------------------------------
void        CConsole::GetVersion       (int*v1, int*v2)
{
    Assert (v1);
    *v1 = VERSION1;
    Assert (v2);
    *v2 = VERSION2;
}
//-----------------------------------------------------------------------------

VprintfP CConsole::GetOutput (void)
    {return m_output;}
//-----------------------------------------------------------------------------

int CConsole::FindKeyword      (const char* name)
{
    AssertC (name, badParams);

    //int i = FindVariable (name);
    //if (i > 0) return -(i);

    /*for (i = 0; i < m_numBasicCommands; i ++)
        if (m_basicCommands[i])
            if (!strcmp (m_basicCommands[i]->keyword, name))
                return (-i-1);
*/
    for (int i = 1; i < m_sizeCommmands; i ++)
        if (m_commands [i])
            if (!strcmp (m_commands[i]->keyword, name))
                return i;
    return 0;
}
//-----------------------------------------------------------------------------

int CConsole::FindFreeSpace    ()
{
    int s = m_sizeCommmands>1?m_sizeCommmands:1;
    for (int i = 1; i < m_sizeCommmands; i++)
        if (!m_commands[i])
            return i;
    Grow ();
    return s;
}
//-----------------------------------------------------------------------------
void CConsole::Grow(void) {return GrowFor (DEFAULT_INCREMENT);}
//-----------------------------------------------------------------------------
void CConsole::GrowFor (int s){return GrowTo (m_sizeCommmands + s);}
//-----------------------------------------------------------------------------
void CConsole::GrowTo  (int s)
{
    AssertC (s > m_sizeCommmands, badParams);

    Command** temp = new Command*[s];
    AssertC (temp, memoryLeak);

    ZeroMemory (temp, sizeof (Command*)*s);
```

```
    if (m_commands)
        for (int i = 0; i < m_sizeCommmands; i++)
            temp[i] = m_commands[i];

    if (m_commands) delete m_commands;
    m_commands = temp;
    m_sizeCommmands = s;
}
//-----------------------------------------------------------------------------
//void CConsole::GrowB (void) {return GrowFor (DEFAULT_INCREMENT);}
//-----------------------------------------------------------------------------
//void CConsole::GrowBFor  (int s){return GrowBTo (m_numBasicCommands + s);}
//-----------------------------------------------------------------------------
/*void CConsole::GrowBTo   (int s)
{
    AssertC (s > m_numBasicCommands, badParams);

    Command** temp = new Command*[s];
    AssertC (temp, memoryLeak);

    ZeroMemory (temp, sizeof (Command*)*s);
    if (m_commands)
        for (int i = 0; i < m_numBasicCommands; i++)
            temp[i] = m_basicCommands[i];

    if (m_basicCommands) delete m_basicCommands;
    m_basicCommands = temp;
    m_numBasicCommands = s;
}*/
//-----------------------------------------------------------------------------

int CConsole::printf   (const char* format, ...)
{
    va_list list;
    va_start (list, format);
    if (m_Prompt)
        {
        m_output ("\n\r", NULL);
        m_Prompt = false;
        }
    return m_output (format, list);
}
//-----------------------------------------------------------------------------

int CConsole::vprintf (const char* format, va_list list)
{
    if (m_Prompt)
        {
        m_output ("\n\r", NULL);
        m_Prompt = false;
        }
    return m_output (format, list);
}
//-----------------------------------------------------------------------------

void CConsole::outline (const char* str)
{
    if (m_Prompt)
        {
        m_output ("\n\r", NULL);
        m_Prompt = false;
        }
    this->printf ("%s\n", str);
}
//-----------------------------------------------------------------------------

void CConsole::AddCommand      (const char* keyword, CommandProcP proc, bool lexems, char* module)
{
    Assert (m_commands);
    if (FindKeyword (keyword) != NULL)
        Throw (runtime, "The keyword %s is already existed", keyword));

    int i = FindFreeSpace ();
    m_commands [i] = new Command;
    Assert (m_commands[i]);
    m_commands [i]->keyword = new char[strlen(keyword)];
    Assert (m_commands[i]->keyword);
    strcpy (m_commands[i]->keyword, keyword);
    m_commands[i]->proc = proc;
    m_commands[i]->wantLexems = lexems;
    m_commands[i]->module = module? module: "unknown";
}
//-----------------------------------------------------------------------------
/*void CConsole::AddBasicCommand      (const char* keyword, CommandProcP proc, bool lexems)
{
    if (FindKeyword (keyword) != NULL)
        Throw (runtime, "The keyword %d is already existed", keyword));

    try {
        for (int i = 0; i < m_numBasicCommands; i++)
            if (!m_basicCommands[i])
                throw (i);

        int s = m_numBasicCommands;
        GrowB ();
        throw (s);
        }
```

```
    catch (int i)
        {
        m_basicCommands [i] = new Command;
        Assert (m_basicCommands[i]);
        m_basicCommands [i]->keyword = new char[strlen(keyword)];
        Assert (m_basicCommands[i]->keyword);
        strcpy (m_basicCommands[i]->keyword, keyword);
        m_basicCommands[i]->proc = proc;
        m_basicCommands[i]->wantLexems = lexems;
        return;
        }
    Throw ("It must not be executed!"));
    return;
}
*/
//-----------------------------------------------------------------------------

void CConsole::DeleteCommand   (const char* keyword)
{
    int i = 0;
    if (i = FindKeyword (keyword) > 0)
        {
        delete m_commands [i];
        m_commands [i] = NULL;
        return;
        }
    Throw (runtime, "Trying to delete not a keyword '%s'", keyword));
}
//-----------------------------------------------------------------------------

void CConsole::RaiseCommand    (const char* cmd, bool user)
{
    AssertC (cmd, badParams);

    strcpy (m_charBuffer, cmd);

    if (!m_Prompt && user)
        {
        m_Prompt = false;
        printf ("\r%s:%s\n", m_PromptStr, cmd);
        }


    int l = strlen (cmd); if (l >=sizeof (m_charBuffer) -2) return; m_charBuffer[l] = 0; m_charBuffer[l +
1] = 0;

    m_lexemErrors  = false;
    m_lexemWarnings  = false;
    m_numLexems = 0;
    m_curChar = 0;
    try {this->LexemAnalyzer ();}
    catch (int){
        }

    if (m_commandPrefix)
        {
        int proc = this->FindKeyword (m_commandPrefix);
        if (!strcmp (m_lexemBuffer[0].data, "BackToConsole"))
            {
            m_commandPrefix = NULL;
            return;
            }

        if (proc)
            if (m_commands[proc]->proc)
            {
                m_commands[proc]->proc (static_cast <IConsole*> (this),
                    m_commands[proc]->wantLexems? m_numLexems:0,
                    m_commands[proc]->wantLexems?
                    (void*)&m_lexemBuffer[0]:
                    (void*)&m_charBuffer[0]);
                return;
            }
            else
                m_commandPrefix = NULL;
        }

    if (!m_numLexems) return;

    if (m_lexemErrors || (LEXEM_WARNINGSASERRORS && m_lexemWarnings))
        {this->printf ("Command can't be executed bacause there were errors.\n"); return;}

    if (m_lexemWarnings)
        {
        this->printf (20, "There were warnings, so command may be executed incorrectly\n");
        }
    //Now the command is converted to lexems, lexems are recognized and correct
    if (!(m_lexemBuffer [0].type == Lexem::LT_String))
        {
        //printf ("First lexem must be a name of procedure\n");
        //return;
        goto UnkwCmd;
        }
//  if (m_lexemBuffer[0].idata < 0)
//      BuiltInCommandOrVariable ();

    //External command:
```

```
    if (m_commands[m_lexemBuffer[0].idata]->proc)
        m_commands[m_lexemBuffer[0].idata]->proc (static_cast <IConsole*> (this),
            m_commands[m_lexemBuffer[0].idata]->wantLexems? m_numLexems-1:0,
            m_commands[m_lexemBuffer[0].idata]->wantLexems?
            (void*)&m_lexemBuffer[1]:
            (void*)&m_charBuffer[m_lexemBuffer[0].datalen + 1]);
    else
        { //unknown command
        UnkwCmd:
            if (m_unhandledCommandHandler)
                {
                if (!m_unhandledCommandHandler (static_cast <IConsole*> (this),
                    m_unhandledCommandHandlerWantLexems? m_numLexems:0,
                    m_unhandledCommandHandlerWantLexems? (void*)&m_lexemBuffer[0]:
                    (void*)&m_charBuffer[0]))
                    {
                    //totally unknown command
                    this->printf ("\tUnknown command.\n");
                    return;
                    }
                }
            else printf ("\tUnknown command.\n");
        }
}
//-----------------------------------------------------------------------------

void CConsole::Reset           (void)
{
    Throw ("What a <..> wants to reset Console?!"));
}
//-----------------------------------------------------------------------------
/*void CConsole::BuiltInCommandOrVariable ()
{
    Throw ("Not implemented"));
}
//-----------------------------------------------------------------------------

void CConsole::NewVariable     (const char* name)
{
    AssertC (name, badParams);
    if ((this->FindKeyword (name)) != 0)
        Throw (runtime, "Cannot create variable '%s': name conflict", name));

    Variable* v = new Variable;
    AssertC (v, memoryLeak);
    v->name = new char[strlen (name)];
    AssertC (v->name, memoryLeak);
    strcpy (v->name, name);

    v->type = 0;

    m_variables[FindFreeVariable ()] = v;

    this->printf ("Variable '%s' successfully created\n", name);
}
//-----------------------------------------------------------------------------
Variable*   CConsole::GetVariable      (const char* name)
    {return GetVariable (FindVariable (name));}
//-----------------------------------------------------------------------------
Variable*   CConsole::GetVariable      (const int number)
{
    AssertC (number >= 0 && number < m_sizeVariables, badParams);
    return m_variables[number];
}
//-----------------------------------------------------------------------------
void        CConsole::DeleteVariable   (const char* name)
    {return DeleteVariable (FindVariable (name));}
//-----------------------------------------------------------------------------
void        CConsole::DeleteVariable   (const int number)
{
    AssertC (number >= 0 && number < m_sizeVariables, badParams);

    if (m_variables[number])
        {
        if (m_variables[number]->dp && m_variables[number]->type == Variable::String)
            {
            delete m_variables[number]->value.c;
            }
        delete m_variables[number]->name;
        delete m_variables[number];
        m_variables[number] = NULL;
        return;
        }
}
//-----------------------------------------------------------------------------
int         CConsole::FindVariable     (const char* name)
{
    AssertC (name, badParams);

    for (int i = 1; i < m_sizeVariables; i++)
        if (m_variables[i])
            if (!strcmp (m_variables[i]->name, name))
                return i;
    return -1;
}
//-----------------------------------------------------------------------------
IVariableEngine*    CConsole::GetVEInterface   (void)
```

```
    {return static_cast <IVariableEngine*>(this);}
//-------------------------------------------------------------------------------------------
void CConsole::GrowVariables       (void)
    {return GrowVariablesFor (DEFAULT_INCREMENT);}
//-------------------------------------------------------------------------------------------
void CConsole::GrowVariablesTo (int s)
{
    AssertC (s > m_sizeVariables, badParams);

    Variable** temp = new Variable*[s];
    AssertC (temp, memoryLeak);

    ZeroMemory (temp, sizeof (Variable*)*s);

    if (m_variables)
        for (int i = 0; i < m_sizeVariables; i++)
            temp[i] = m_variables[i];

    if (m_variables) delete m_variables;
    m_variables = temp;
    m_sizeVariables = s;
}
//-------------------------------------------------------------------------------------------
void CConsole::GrowVariablesFor    (int s)
    {return GrowVariablesTo (s + m_sizeVariables);}
//-------------------------------------------------------------------------------------------
int  CConsole::FindFreeVariable    (void)
{
    for (int i = 1; i < m_sizeVariables; i++)
        if (m_variables[i] == NULL)
            return i;
    int s = m_sizeVariables;
    GrowVariables ();
    return s;
}
*/
//-------------------------------------------------------------------------------------------
IMPLEMENT_IUNKNOWN (CConsole, IConsole, IID_IConsole);

//-------------------------------------------------------------------------------------------
int CConsole::printf           (int l, const char* f, ...)
{
    if (l < m_displayLevel)
        {
        va_list list;
        va_start (list, f);
        if (m_Prompt)
            {
            m_output ("\n\r", NULL);
            m_Prompt = false;
            }
        int i = vprintf (f, list);
        va_end (list);
        return i;
        }
    else return -1;
}
//-------------------------------------------------------------------------------------------

int CConsole::vprintf          (int l, const char* f, va_list li)
{
    if (l < m_displayLevel)
        return vprintf (f, li);
    return -1;
}
//-------------------------------------------------------------------------------------------

int CConsole::SetDisplayLevel (int level)
{
    int o = m_displayLevel;
    m_displayLevel = level;
    return o;
}
//-------------------------------------------------------------------------------------------

int CConsole::GetDisplayLevel  (void)
    {return m_displayLevel;}
//-------------------------------------------------------------------------------------------

void CConsole::SetUnknownCommandHandler (UnhandledCommandProc handler, bool lexems)
{
    m_unhandledCommandHandler = handler;
    m_unhandledCommandHandlerWantLexems = lexems;
}
//-------------------------------------------------------------------------------------------

void CConsole::SetConsoleMode  (char* firstCommand)
{
    m_commandPrefix = firstCommand;
}
//-------------------------------------------------------------------------------------------
int CConsole::GetDisplay (void)
{
    return this->m_display;
}

void CConsole::SetDisplay (int v)
```

```
{
    m_display = v;
}

void CConsole::SetPrompt   (char* p)
{
    strcpy (m_PromptStr, p?p:"System>");
}

void CConsole::DoPrompt(void)
{
    printf ("%s", m_PromptStr);
    m_Prompt = true;
}

bool CConsole::GetPr ()
{
    return m_Prompt;
}
```

Файл Console\Lexem.cpp

```
//╥хъёхьэ√щ рэрышчрЄюЁ - с№хЄ шёїюфэє■ яЁюуЁрььє эр ыхъёхь√ ъыхЄюўэ√ь ртЄюьрЄюь.

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "CConsole::Lexem analyzer"

#include "stdafx.h"
#include <Math.h>
#include <stdlib.h>
#include "Lexem.h"
#include "ConsoleCore.h"
#include "stdarg.h"
#include <exeption.h>

#pragma warning (disable:4244)

//-----------------------------------------------------------------------------------------------------
//Implementation of Lexem structure
void Lexem::AddChar(char ch)
    {
    data[datalen] = ch;
    datalen++;
    }
void Lexem::Reset()
    {
    type = datalen = *data = line = 0;
    }
//-----------------------------------------------------------------------------------------------------

void CConsole::StoreLexem (Lexem lexem)
{
    if (!lexem.datalen) return;

    lexem.data[lexem.datalen] = '\0';
    if (m_numLexems < sizeof (m_lexemBuffer)- 1)
        m_lexemBuffer[m_numLexems++] = lexem;
    //Process lexem - number or keyword
    Lexem* l = &m_lexemBuffer[m_numLexems-1];
    l->idata = 0;

    switch (l->type)
        {
        case Lexem::LT_ConstChar:
            l->idata = l->data [0];
            break;
        case Lexem::LT_Float:
            double d;
            d = strtod (l->data, NULL);
            l->idata = d;
            l->ddata = d;
            break;
        case Lexem::LT_Number:
            sscanf (l->data, "%d", &l->idata);
            l->ddata = l->idata;
            break;
        case Lexem::LT_String:
            if ((l->idata = FindKeyword (l->data)) == 0)
                {
                //if (m_display) this->printf (9, "\tUnknown keyword: '%s', token as string\n", l->data);
                //m_numLexems--;
                l->type = Lexem::LT_ConstString;
                //Throw (runtime, "Undefined indefiner: '%s'\n", l->data));
                }
            break;
        case Lexem::LT_Operator:
            break;
        case Lexem::LT_ConstString:
            break;

        default:
```

```
            this->printf (9, "Error: unsupported lexem type, data = '%s'\n", l->data);
            m_numLexems--;
            break;
        }
    if (lexem.type = Lexem::LT_String && m_numLexems == 1)
        {
        //we may stop lexem analyse, if function do not need it
/*      if (l->idata > 0)
            if (!m_commands[l->idata]->wantLexems)
                throw (1);*/

        /*if (l->idata < 0)
            if (-l->idata < m_numBasicCommands)
                if (!m_basicCommands[-(l->idata)]->wantLexems)
                    throw (1);*/
        }
}

void CConsole::LexemWarning (int line, char* text)
{
    m_lexemWarnings = true;
    if (m_display) this->printf (10, "Lexem warning: %s, line %d\n", text, line);
}

void CConsole::LexemError (int line, char* text)
{
    m_lexemErrors = true;
    if (m_display) this->printf (9, "Lexem Error: %s, line %d\n", text, line);
}

char CConsole::GetNextChar()
{
    if (m_curChar < sizeof (m_charBuffer)-1)
        return m_charBuffer [m_curChar++];
    if (m_display)LexemWarning (-1, "Buffer onderrun, lexem truncated");
    return '\0';
}
//-----------------------------------------------------------------------------------------------
int CConsole::LexemAnalyzer()
{
    int alredyerr = false;
    int lineno = 1;

    this->m_numLexems  = 0;
    this->m_curChar = NULL;

    Lexem temp;
    temp.Reset();
    char ch = 0;
    while (ch = GetNextChar())
    {
        switch (ch)
            {
            case '#':
                LexemError (lineno, "Pragma commands aren't supported in this version");
                break;
            case '\0': StoreLexem (temp); temp.Reset (); return 1; break;
            case '\n': lineno++; case ' ': case '\t':
                StoreLexem (temp); temp.Reset(); break;
            case '.': if (temp.type == Lexem::LT_Number)
                            {
                            temp.type = Lexem::LT_Float;
                            temp.AddChar (ch);
                            break;
                            }
            case ';':
                StoreLexem (temp);
                temp.Reset();
                temp.line = lineno;
                temp.type = Lexem::LT_Operator;
                temp.AddChar (ch);
                StoreLexem(temp); temp.Reset();
                break;
            case ':': case '!': case '%': case '^': case '&': case '*':
            case '(': case ')':  case '+': case '=': case '~':
            case ',': case '.': case '/': case '<': case '>': case '?':
            case '[': case ']':case '{': case '}': case '|': case '\\':
            case '-':
                if (temp.type != Lexem::LT_Operator && temp.type != Lexem::LT_Float)
                    {
                    StoreLexem (temp);
                    temp.Reset();
                    temp.line = lineno;
                    temp.type = Lexem::LT_Operator;
                    }
                temp.AddChar (ch);
                if (temp.data[0] == '/')
                    {
                    if (temp.data[1] == '/')
                        while (ch = GetNextChar()!='\n')
                            if (ch == '\0') LexemError (lineno, "Unfinished line comment");
                    if (temp.data[1] == '*')
                        {
                            while (true)
                            {
                            while (GetNextChar()!= '*');
                            if (ch = GetNextChar() == '/') break;
```

```
                            if (ch == '\0')
                                {
                                LexemError (lineno, "Unfinished comment");
                                }
                             }
                        }
                    }
                break;
            case '0': case '1':  case '2': case '3': case '4': case '5':
                 case '6': case '7': case '8': case '9':
                 //we may have number
/*!!!*/             if (temp.type == Lexem::LT_Operator && temp.data[0] == '-')
                    temp.type = Lexem::LT_Number;
                if (temp.type == Lexem::LT_String)
                    {temp.AddChar (ch); break;}
                if (temp.type != Lexem::LT_Number && temp.type != Lexem::LT_Float)
                    {
                    StoreLexem (temp);
                    temp.line = lineno;
                    temp.Reset ();
                    temp.type = Lexem::LT_Number;
                    }
                temp.AddChar (ch);
             break;
            case '"': //constant string started
                StoreLexem (temp);
                 temp.Reset ();
                temp.line = lineno;
                temp.type = Lexem::LT_ConstString;
                while ((ch = GetNextChar()) != '"')
                    {
                    if (ch == '\0') {LexemError (lineno, "Unfinished string constant");break;}
                    if (ch == '\n') {LexemError (lineno, "Newline in constant string"); break;}
                    temp.AddChar (ch);
                     }
                break;
            case '\'': //constant characher started
                StoreLexem (temp);
                temp.Reset ();
                temp.line = lineno;
                temp.type = Lexem::LT_ConstChar;
                for (int ii = 0; (ch = GetNextChar()) != '\''; ii++)
                    {
                    if (ch == '\0') {LexemError (lineno, "Unfinished character constant");break;}
                    if (ch == '\n') {LexemError (lineno, "Newline in characher constant"); break;}
                    if (ii > 2 && !alredyerr) {LexemWarning (lineno, "Too many indefiners in characer
constant, used only first");alredyerr = true;}
                    temp.AddChar (ch);
                    }
                 StoreLexem (temp);
                temp.Reset ();
                temp.line = lineno;
                break;
            default: //part of name etc. - string
                if (temp.type != Lexem::LT_String)
                    {
                    StoreLexem (temp);
                    temp.Reset ();
                    temp.line = lineno;
                    temp.type = Lexem::LT_String;
                    }
                temp.AddChar (ch);
                break;
            }
    }
    StoreLexem (temp);
    return true;
}

//-------------------------------------------------------------------------------------------------
/*
//Test:
#include <conio.h>
int _tmain ()
{
    FillLexemBuffer ();
    getch();
}

char data[256] = "This is an    example              of string;\"Constant string \" 'characher
const'\n123+ 23456/7654= ?\n//comment\n";
int datap = 0;

char GetNextChar()
{
    if (datap < sizeof (data))
        return data[datap++];
    else return 0;
}
*/
```

Файл Console\Module.cpp

```
//Module.cpp

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Console.dll initialization"

#include <exeption.h>
#include "System.h"
#include "ConsoleCore.h"

    #define DLL_NAME "Console"
    #define DLL_CLASS CConsole
    #include "..\Common\Utilities\Module.c"


DLL_INIT(loader)
{
    Assert (loader);
    Loader = loader;
    Console = (IConsole*)Loader->GetInterface (IID_IConsole);
    Assert (Console);
}

DLL_ONLOAD (loader)
{
    printf ("CONSOLE: Console by Vasfed version 1.0\n");
}
```

Файл Console\System.cpp

```
// System.cpp - implementation of System.h
// created 13.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "System"

#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine

ILoader* Loader = NULL;
IConsole* Console = NULL;
```

Файл Console3dCommands\Commands.cpp

```
// Commands.cpp - implementation of Commands.h
// created 03.12.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "3d ConsoleCommands"

#include "Commands.h"
#include <Exeption.h>      // use ExeptionEngine

#include "Globals.h"
#include "Keyboard.h"
#include "Graphics.h"
#include "MainLoop.h"
#include <process.h>

void StartGraphics (IConsole* c, int ln, Lexem1* data)
{
    c->printf ("Starting graphics...\n");

    try
        {
        c->printf ("    StorageMaster...\n");
        Master -> Init (14);

        c->printf ("    Graphics...\n");
        InitGraphics ();

        c->printf ("    AgentsMap...\n");

        //Map->LoadBmp ("VirtualWorld\\map4.bmp");
        Map->Load ("Map.map");

        c->printf ("    Agents...\n");


        //Compiler->compileFromFile ("pimp.smt", NULL);
```

```
        int ai = Server->FindAgentCreatorByName (/*"Pimp"); */ "Ufo");
        //for (int a = 0; a < AgentsCount - 1; a++)
        /*Server->CreateAgent (ai,  -90.0F, -440.0F, 11.0F, 0);
        Server->CreateAgent (ai, -110.0F, -460.0F,  5.0F, 0);
        Server->CreateAgent (ai, random (100.0F) - 50.0F, random (100.0F) - 50.0F, 10.0, 0);
        Server->CreateAgent (ai, random (100.0F) - 50.0F, random (100.0F) - 50.0F, 10.0, 0);*/

        //Server->CreateAgent (Server->FindAgentCreatorByName ("Ship"), -100.0F, -440.0F, 1.0F, 0);

        c->printf ("    Scene...\n");
        InitScene ();

        /*
        Master->Lock ("Landscape");
        CLandscape* land = (CLandscape*)(Master->RetData ("Landscape").RetObject (0));
        Map->SetSquareSize (land->RetHorizont (0).RetSquareSize ());
        //Map->Load ("Map.map");
        Master->Unlock ("Landscape");

        Map->Save ("Map.map");
        */


        //Server->InitializeAgents ();


        Environment -> SetPhysicLevel (PhysicLevel = PL_BBOX_TEST);
        //if we've reached this place, it means that...
        StartedNormally = true;
        NoErrors = true;
        }
    catch (...)
        {
        Message ("Initialization error");
        c->printf ("System initialization failed.\n");
        c->printf ("Use console...");
        if (MessageBox (NULL, "System startup failed.\nExit?", "Failture", MB_YESNO | MB_ICONQUESTION |
MB_SYSTEMMODAL) == IDYES)
            PostQuitMessage (0);
        }
    StartRender ();
    c->printf (" Graphics startup succeded.\n\n");

}

void StartMainLoop (IConsole*c, int, void*)
{
    if (!MainLoopShouldStop)
        {
        Renderer -> ShowGLWindow ();
        c->printf ("Mainloop already started.\n");
        return;
        }

    c->printf ("Starting main loop...\n");
    MainLoopPause = true;
    MainLoopShouldStop = false;

    if (_beginthread (MainLoop, NULL, NULL) == -1L)
        {
        Renderer->HideGLWindow ();
        c->printf ("Failed to create thread for mainloop.\n");
        MainLoopShouldStop = true;
        return;
        }

    MainLoopPause = false;

    //MainLoop (0);
    //Renderer->HideGLWindow ();
}

void PauseMainLoop (IConsole*c, int ln, Lexem1* data)
{
    if ( ln == 0)
        MainLoopPause ^= 1;
    else
        {
            if ( !strcmpi ( data->data, "on"))
                MainLoopPause = true;
            else if ( !strcmpi ( data->data, "off"))
                MainLoopPause = false;
            else MainLoopPause ^= 1;
        }

    if ( MainLoopPause)
        {
        Environment -> SetPhysicLevel (PhysicLevel = PL_NO_MOVE);
        }
    else
        {
        Environment -> SetPhysicLevel (PhysicLevel = PL_BBOX_TEST);
        }
    c->printf ("3D %s.\n", MainLoopPause? "paused":"restored");
}
/*
```

```
void StopMainLoop (IConsole*c, int, void*)
{
    MainLoopShouldStop = true;
    c->printf ("3D .\n");
    Renderer -> HideGLWindow ();
}
*/
void ShowGlWin (IConsole*c, int, void*)
{
    //MainLoopShouldStop = true;
    //c->printf ("3D .\n");
    Renderer -> ShowGLWindow ();
}

void HideGlWin (IConsole*c, int, void*)
{
    //MainLoopShouldStop = true;
    //c->printf ("3D .\n");
    Renderer -> HideGLWindow ();
}

void AddAgent (IConsole* c, int ln, Lexem1* data)
{
    c->printf ("AddAgent called.\n");
    AssertT (CurrentAgentsCount < MaxAgentsCount - 1, "Can't add agent - no place");

    if (ln == 0)
        {
        strcpy (data[0].data, "Ufo");
        data[0].type = data[0].LT_ConstString;
        ln = 1;
        }

    if (ln == 1)
        {
        strcpy (data[1].data, "VirtualWorld\\Ufo.xxx");
        data[1].type = data[0].LT_ConstString;
        }


    int creator = Server->FindAgentCreatorByName (data[0].data);
    if (creator < 0) return;

    Environment -> SetPhysicLevel (PL_NO_MOVE);
    MainLoopPause = true;
    Sleep (1000);

    Master -> LoadXXX (data[1].data/*"VirtualWorld\\ufo.xxx"*/);

    c->printf ("XXX loaded.\n");
    Master -> Lock ("Mesh");
    Master -> Lock ("Mesh2");
    Master -> Lock ("Polygon");
    Master -> Lock ("Polygon2");
    Master -> Lock (DT_LIGHTSOURCE);
    Master -> Lock ("Speed");
    Master -> Lock ("Acceleration");
    Master -> Lock ("IntersectionSphere");
    Master -> Lock ("IntersectionSpace");
    Master -> Lock ("BoundingBox");
    Master -> Lock ("Material");
    Master -> Lock ("Landscape");

    try {Renderer -> PrepareMeshes (Master -> RetData ("Mesh"),
                                Master -> RetData ("Polygon"),
                                Master -> RetData ("Mesh2"),
                                Master -> RetData ("Polygon2"),
                                Master -> RetData (DT_LIGHTSOURCE), CurrentAgentsCount, CurrentAgentsCount +
1);
    c->printf ("Meshes prepared.\n");
    int agent = Server->CreateAgent (creator, 10, 10, 10, 0);

    //for (int n = 0; n < CurrentAgentsCount; n ++)
        {
        IAgent* a = Server->GetAgent(agent);
        if (!a) //continue;
            return;
        ((CMesh*) (Master -> RetObject ("Mesh2", /*n*/agent))) -> SetCenter (a->m_x, a->m_y, a->m_z);
        }

    Environment -> AddElement ((CMesh *) Master -> RetObject ("Mesh2", CurrentAgentsCount),
                                (CSpeed *) Master -> RetObject ("Speed", CurrentAgentsCount),
                                (CAcceleration *) Master -> RetObject ("Acceleration", CurrentAgentsCount),
                                (CIntersectionSphere *) Master -> RetObject ("IntersectionSphere",
CurrentAgentsCount),
                                (CIntersectionSpace *) Master -> RetObject ("IntersectionSpace",
CurrentAgentsCount),
                                (CBoundingBox *) Master -> RetObject ("BoundingBox", CurrentAgentsCount),
                                (CPhysMaterial *) Master -> RetObject ("Material", CurrentAgentsCount));

    c->printf ("Element added.\n");
    Server->GetAgent (agent)->Init ();
    }
    catch(...){}

    Master -> Unlock ("Mesh");
```

```
    Master -> Unlock ("Mesh2");
    Master -> Unlock ("Polygon");
    Master -> Unlock ("Polygon2");
    Master -> Unlock (DT_LIGHTSOURCE);
    Master -> Unlock ("Speed");
    Master -> Unlock ("Acceleration");
    Master -> Unlock ("IntersectionSphere");
    Master -> Unlock ("IntersectionSpace");
    Master -> Unlock ("BoundingBox");
    Master -> Unlock ("Material");
    Master -> Unlock ("Landscape");

    CurrentAgentsCount++;
    MainLoopPause = false;
    Environment -> SetPhysicLevel (PhysicLevel);
}

void InitializeAgents (IConsole*c, int, void*)
{
    Server->InitializeAgents ();
}

void MoveAgent (IConsole* console, int ln, Lexem1* data)
{
    if (ln <= 1) return;
    if (ln = 2)
        {
        data[1].ddata = 10;
        ln = 3;
        }
        if (ln = 3)
        {
        data[2].ddata = 10;
        ln = 4;
        }
        if (ln = 4)
        {
        data[3].ddata = 10;
        ln = 3;
        }
    Server->SetAgentPosition (data[0].idata, data[1].ddata, data[2].ddata, data[3].ddata);
}

void AddCommands (IConsole* console)
{
    Assert (console);

    //console->AddCommand ("test", (CommandProcP)Test, false, "Console 3D commands");
    console->AddCommand ("addagent",(CommandProcP)AddAgent, true, "Console 3D Commands");
    console->AddCommand ("initializeagents",(CommandProcP)InitializeAgents, true, "Console 3D Commands");
    console->AddCommand ("start3d",    (CommandProcP)StartMainLoop, false, "Console 3D Commands");
    console->AddCommand ("pause3d",    (CommandProcP)PauseMainLoop, true, "Console 3D Commands");
    //console->AddCommand ("stop3d",   (CommandProcP)StopMainLoop,  false, "Console 3D Commands");
    console->AddCommand ("show3d", (CommandProcP)ShowGlWin,  false, "Console 3D Commands");
    console->AddCommand ("hide3d", (CommandProcP)HideGlWin,  false, "Console 3D Commands");
}
```

Файл Console3dCommands\Globals.cpp

```
// Globals.cpp - implementation of Globals.h
// created 25.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Globals"

#include "StdAfx.h"      // use precompiled header
#include "Globals.h"

#include <Exeption.h>      // use ExeptionEngine

int ScrX = 1024,
    ScrY = 768;
int CurrentAgentsCount = 0;
int MaxAgentsCount = 50;
int AgentNum = 0;
CPoint3D Target (0.0F, 0.0F, 0.0F);
int RenderLevel = RL_WIREFRAME;
int PhysicLevel = PL_BBOX_TEST;

bool StartedNormally = false;
int NoErrors = false;

char szProgName    []  = "Program Name";
char szWndName     []  = "OpenGL Window";

int StartupOptions::scrX = 640;
int StartupOptions::scrY = 480;
bool StartupOptions::fullscreen = true;

int StartupOptions::renderLevel = 1;
```

```
int StartupOptions::physicsLevel = 1;

char StartupOptions::batchFile[256] = "";
bool StartupOptions::batch = false;

int Render = false;
```

Файл Console3dCommands\Graphics.cpp

```
// Graphics.cpp - implementation of Graphics.h
// created 25.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Graphics"

#include "StdAfx.h"      // use precompiled header
#include "Graphics.h"
#include "Commands.h"
#include "Keyboard.h"
#include "System.h"
#include "Globals.h"
#include <Exeption.h>      // use ExeptionEngine

#include <process.h>

void InitScene (char* filename)
    {
    Master -> SetNewDataType (sizeof (CMesh), MaxAgentsCount, "Mesh");
    Master -> SetNewDataType (sizeof (CMesh), MaxAgentsCount, "Mesh2");
    Master -> SetNewDataType (sizeof (CPolygon), MaxAgentsCount * 500, "Polygon");
    Master -> SetNewDataType (sizeof (CPolygon), MaxAgentsCount * 500, "Polygon2");
    Master -> SetNewDataType (sizeof (CAcceleration), MaxAgentsCount, "Acceleration");
    Master -> SetNewDataType (sizeof (CSpeed), MaxAgentsCount, "Speed");
    Master -> SetNewDataType (sizeof (CIntersectionSphere), MaxAgentsCount, "IntersectionSphere");
    Master -> SetNewDataType (sizeof (CIntersectionSpace), MaxAgentsCount, "IntersectionSpace");
    Master -> SetNewDataType (sizeof (CBoundingBox), MaxAgentsCount, "BoundingBox");
    Master -> SetNewDataType (sizeof (CPhysMaterial), MaxAgentsCount, "Material");

    char directory[300];
    GetCurrentDirectory (300, directory);
    SetCurrentDirectory (".\\VirtualWorld\\");

    for (int a = 0; a < CurrentAgentsCount; a ++)
        {
        Master -> LoadXXX ("ufo.xxx");
        }
    //Master -> LoadXXX ("small.xxx");
    Master -> LoadXXX ("plants.xxx");

    /*Master -> Lock ("Mesh2");
    Master -> CreateObjects ("Mesh2", MaxAgentsCount);
    Master -> Unlock ("Mesh2");
    Master -> Lock ("Polygon2");
    Master -> CreateObjects ("Polygon2", MaxAgentsCount * 500);
    Master -> Unlock ("Polygon2");*/

    Environment -> InitTable (MaxAgentsCount);

    /*Master -> Lock ("Mesh");
    Master -> Lock ("Mesh2");
    Master -> Lock ("Polygon");
    Master -> Lock ("Polygon2");
    Master -> Lock (DT_LIGHTSOURCE);
    Master -> Lock ("Speed");
    Master -> Lock ("Acceleration");
    Master -> Lock ("IntersectionSphere");
    Master -> Lock ("IntersectionSpace");
    Master -> Lock ("BoundingBox");
    Master -> Lock ("Material");
    Master -> Lock ("Landscape");

    Renderer -> PrepareMeshes (Master -> RetData ("Mesh"),
                               Master -> RetData ("Polygon"),
                               Master -> RetData ("Mesh2"),
                               Master -> RetData ("Polygon2"),
                               Master -> RetData (DT_LIGHTSOURCE));

    for (int n = 0; n < CurrentAgentsCount; n ++)
        {
        IAgent* a = Server->GetAgent(n);
        if (!a) continue;
        ((CMesh*) (Master -> RetObject ("Mesh2", n))) -> SetCenter (a->m_x, a->m_y, a->m_z);
        }

    Environment -> MakeTable (Master -> RetData ("Mesh2"),
                              Master -> RetData ("Speed"),
                              Master -> RetData ("Acceleration"),
                              Master -> RetData ("IntersectionSphere"),
                              Master -> RetData ("IntersectionSpace"),
                              Master -> RetData ("BoundingBox"),
                              Master -> RetData ("Material"),
```

```
                            Master -> RetData ("Landscape"));

    Master -> Unlock ("Mesh");
    Master -> Unlock ("Mesh2");
    Master -> Unlock ("Polygon");
    Master -> Unlock ("Polygon2");
    Master -> Unlock (DT_LIGHTSOURCE);
    Master -> Unlock ("Speed");
    Master -> Unlock ("Acceleration");
    Master -> Unlock ("IntersectionSphere");
    Master -> Unlock ("IntersectionSpace");
    Master -> Unlock ("BoundingBox");
    Master -> Unlock ("Material");
    Master -> Unlock ("Landscape");*/

    Master -> Lock ("Landscape");
    Environment -> SetLandscape ((CLandscape *) Master -> RetObject ("Landscape", 0));
    Master -> Unlock ("Landscape");
    SetCurrentDirectory (directory);
    }

//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

void RedrawProc ()
    {
    try {
        Master -> Lock ("Mesh");
        Master -> Lock ("Mesh2");
        Master -> Lock ("Landscape");
        Master -> Lock ("BillBoard");
        Renderer -> Clear ();
        if (ShouldRender ())
            {
            if (CurrentAgentsCount > 0)
                Renderer -> CameraFollow (1, reinterpret_cast <CMesh *> (Master -> RetObject ("Mesh2",
AgentNum)) -> RetCenter ());
            Renderer -> CameraOnLand (1, ((CLandscape *) Master -> RetObject ("Landscape", 0)) ->
RetPHorizont (0));
            Renderer -> CameraOnLand (2, ((CLandscape *) Master -> RetObject ("Landscape", 0)) ->
RetPHorizont (0));
            Renderer -> RenderMeshes (Master -> RetData ("Mesh"), Master -> RetData ("Mesh2"));
            Renderer -> RenderBillBoards (Master -> RetData ("BillBoard"));
            Renderer -> RenderLandscape (Master -> RetData ("Landscape"));
            Renderer -> OutText (10, 10, 1.0F, 1.0F, 1.0F, 1.0F, "FPS is %d", Renderer -> RetFPS ());
            }

        try {
            Server->PerformAgentsStep ();
            Environment -> MoveAll ();
            }
        catch (...)
            {
            NoErrors = NoErrors == 2? false: 2;
            if (NoErrors) Console->printf ("Error happened in MultiAgentSystem.\n");
            else Console->printf ("Error happened in MultiAgentSystem.\nRendering STOPPED\n");
            }

        for (int n = 0; n < CurrentAgentsCount; n ++)
            {
            IAgent* a = Server->GetAgent (n);
            if (!a) continue;
            CMesh* mesh = (CMesh*) (Master -> RetObject ("Mesh2", n));
            CPoint3D p = mesh->RetCenter ();
            a->m_x = p (0);
            a->m_y = p (1);
            a->m_z = p (2);
            }

        Master -> Unlock ("Mesh");
        Master -> Unlock ("Mesh2");
        Master -> Unlock ("Landscape");
        Master -> Unlock ("BillBoard");
        }
    catch (...)
        {
        NoErrors = NoErrors == 3? false: 3;
        if (NoErrors == 3)
            {
            Console->printf ("Error during rendering.\n  If often happens because of powerless of 3D
system of computer.\n  Trying to fix..\n");
            Renderer->SetRenderLevel ((--RenderLevel > 0? RenderLevel: RenderLevel = RL_WIREFRAME));
            }
        else {Console->printf ("Cannot recover after error.\n Rendering stopped. System needs
restart.\n");}
        //MessageBox (NULL, "Runtime Error during rendering", "Error", MB_OK);
        //PostQuitMessage (0);
        return;
        }
        if (NoErrors == 3)
            {
            NoErrors = true;
            Console->printf ("   ok.\n\n");
            }
    }
```

```
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
bool FullScreen = false;

void InitGraphics ()
    {
    char directory[300];
    GetCurrentDirectory (300, directory);
    SetCurrentDirectory (".\\VirtualWorld\\");
    Renderer -> LoadINI ("settings.ini");
    SetCurrentDirectory (directory);

    Renderer -> HideGLWindow ();
    Renderer -> SetKeyboardFunc (KeyboardProc);
    Renderer -> SetRedrawFunc (RedrawProc);
    Renderer -> MakeCameraCurrent (2);

/*  Environment -> SetPhysicLevel (PhysicLevel);*/
    Environment -> SetGravity (0.0F, 0.0F, -9.8F);
    }
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
bool g_RenderShouldStart = false;
//-----------------------------------------------------------------------------
bool ShouldRender       ()
    {
    return g_RenderShouldStart;
    }
//-----------------------------------------------------------------------------
void StartRender   ()
    {
    g_RenderShouldStart = true;
    Assert (Renderer);
    }
//-----------------------------------------------------------------------------
void StopRender    ()
    {
    g_RenderShouldStart = false;
    }
```

Файл Console3dCommands\Keyboard.cpp

```
// Keyboard.cpp - implementation of Keyboard.h
// created 25.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Keyboard"

#include "StdAfx.h"      // use precompiled header
#include "Keyboard.h"

#include <Exeption.h>      // use ExeptionEngine

char* StrRenderLevels[] = {"None", "Wireframe", "Solid", "Lightened",
                           "Low Textured", "High Textured", "Extensions Used"};

void KeyboardProc (HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam)
    {
    switch ((char) wParam)
        {
        case 'q': Renderer -> CameraSlide (1, 1.0F);
                  break;
        case 'a': Renderer -> CameraSlide (1, -1.0F);
                  break;
        case 'w': Renderer -> CameraSlide (1, 5.0F);
                  break;
        case 's': Renderer -> CameraSlide (1, -5.0F);
                  break;
        case '1': Renderer -> MakeCameraCurrent (1);
                  break;
        case '2': Renderer -> MakeCameraCurrent (2);
                  break;
        case ',': if (AgentNum > 0) AgentNum --;
                    Console->printf ("Current agent is now %d\n", AgentNum);
                  break;
        case '.': if (AgentNum < CurrentAgentsCount - 1) AgentNum ++;
                    Console->printf ("Current agent is now %d\n", AgentNum);
                  break;
        case 'f':   //Trace ("FPS is %d", (int) Renderer -> RetFPS ());
                    //Console->printf ("Fps is %d\n", Renderer -> RetFPS ());
                    ScrX = 1024; ScrY = 768;
                    Renderer -> Resize (0, 0, ScrX, ScrY, true);
                  break;
        case 'i': Renderer -> CameraMove (2, 1.0F);
                  break;
        case 'k': Renderer -> CameraMove (2, -1.0F);
                  break;
        case 'o': Renderer -> CameraMove (2, 10.0F);
                  break;
        case 'l': Renderer -> CameraMove (2, -10.0F);
```

```
                break;
        //case (char) 27: PostQuitMessage (0);
        //        Console->printf ("Terminating Graphics...\n");
        //        break;
        case '+': if (RenderLevel < RL_EXTENSIONS_USED)
                        {
                        RenderLevel ++;
                        Renderer -> SetRenderLevel (RenderLevel);
                        Console->printf ("Setting Render level to %d (%s)\n", RenderLevel,
StrRenderLevels[RenderLevel]);
                        if (RenderLevel == RL_EXTENSIONS_USED)
                            {
                            //Console->printf ("   warning: The ExtensionsUsed mode is not supported by
some video cards\n");
                            }
                        }
                break;
        case '-': if (RenderLevel > RL_WIREFRAME)
                        {
                        RenderLevel --;
                        Renderer -> SetRenderLevel (RenderLevel);
                        Console->printf ("Setting Render level to %d (%s)\n", RenderLevel,
StrRenderLevels[RenderLevel]);
                        }
                break;
        case '*': //if (PhysicLevel < PL_BBOX_TEST)
                        {
                        //PhysicLevel ++;
                        Environment -> SetPhysicLevel (PhysicLevel = PL_BBOX_TEST);
                        }
                break;
        case '/': //if (PhysicLevel > PL_NO_MOVE)
                        {
                  //  PhysicLevel --;
                        Environment -> SetPhysicLevel (PhysicLevel = PL_NO_MOVE);
                        }
                break;
        case 'c':
            {
            IAgent* a = Server->GetAgent (AgentNum);
            CPoint3D np (a->m_x, a->m_y, a->m_z);
            Renderer -> CameraFollow (2, np);
            break;
            }
        case 'e': Renderer -> SetFogParameters (1.0F, 0.0F, 30.0F);
                  Renderer -> SetVolumeFogDepth (true, 20.0F);
                  break;
        case 'd': Renderer -> SetFogParameters (0.0025F, 0.0F, 50.0F);
                  Renderer -> SetVolumeFogDepth (false, 0.0F);
                  break;
        case 'r': Renderer -> Resize (0, 0, 320, 240, false);
                  break;
        case 'y':
                    if (ScrX < 640)
                        {
                        ScrX = 640; ScrY = 480;
                        Renderer -> Resize (0, 0, ScrX, ScrY, true);
                        break;
                        }
                    if (ScrX == 640)
                        {
                        ScrX = 800; ScrY = 600;
                        Renderer -> Resize (0, 0, ScrX, ScrY, true);
                        break;
                        }
                  if (ScrX == 800)
                        {
                        ScrX = 1024; ScrY = 768;
                        Renderer -> Resize (0, 0, ScrX, ScrY, true);
                        break;
                        }
                  if (ScrX == 1024)
                        {
                        //ScrX = 320; ScrY = 240;
                        ScrX = 640; ScrY = 480;
                        Renderer -> Resize (0, 0, ScrX, ScrY, true);
                        break;
                        }
        case 'b':
            Renderer->CameraSetPosition (2, 1, 1, 20);
            Renderer->MakeCameraCurrent (2);
            break;
        case 'n':

            break;
        default:  break;
        }
    }
```

Файл Console3dCommands\MainLoop.cpp

```
// MainLoop.cpp - implementation of MainLoop.h, of module Console3dCommands
// created 06.12.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Console3dCommands"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "MainLoop.h"
#include "Globals.h"

#include <Exeption.h>      // use ExeptionEngine

bool MainLoopPause = true;
bool MainLoopShouldStop = true;

int MainLoopMaxErrors = 10;

void StartGraphics (IConsole* c, int ln, Lexem1* data);

void MainLoop (void*)
{
    Console->printf ("\nMainloop started.\n");

    StartGraphics (Console, 0, 0);

    Renderer -> ShowGLWindow ();
    Environment -> SetPhysicLevel ( PhysicLevel = PL_BBOX_TEST);
    MSG lpMsg;

    while (!MainLoopShouldStop)
        {
        while (MainLoopPause) Sleep (100);

        try
            {
            if (PeekMessage (&lpMsg, NULL, 0, 0, PM_NOREMOVE))
                {
                if (GetMessage(&lpMsg, NULL, 0, 0) == 0) break;
                TranslateMessage(&lpMsg);
                DispatchMessage(&lpMsg);
                }
            else
                Renderer -> Render ();
            }
        catch (...)
            {
            if (MainLoopMaxErrors > 0)
                {
                Trace ("GRAPHICS: Exception happened in MainLoop.");
                MainLoopMaxErrors--;
                }
            }
        }
    Renderer -> HideGLWindow ();
    MainLoopShouldStop = true;
}
```

Файл Console3dCommands\Module.cpp

```
// Module.cpp - implementation of Module.h, of module Console3dCommands
// created 03.12.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Console3dCommands"

#include "commands.h"
#include "System.h"
#include "MainLoop.h"
#include <Exeption.h>      // use ExeptionEngine

    #define DLL_NAME "Console Commands for 3D"
    #define DLL_PROGRAMMER "Vasfed"
    #include "..\Common\Utilities\Module.c"

DLL_ONLOAD (loader)
{
    Loader = loader;
    loader->GetInterface (IID_IConsole, (void**)&Console);
    if (Console)
        {
        AddCommands (Console);
        }
}
```

```
DLL_INIT (loader)
{
    Loader = loader;
    Assert (Loader);

    Loader->GetInterface (IID_IStorageMaster,  (void**) &Master);
    Assert (Master);
    Loader->GetInterface (IID_IRenderer,       (void**) &Renderer);
    Assert (Renderer);
    Loader->GetInterface (IID_IEnvironment,        (void**) &Environment);
    Assert (Environment);
    Loader->GetInterface (IID_IAgentsMap,      (void**) &Map);
    Assert (Map);
    Loader->GetInterface (IID_IAgentsServer,   (void**) &Server);
    Assert (Server);

    //Loader->GetInterface (IID_ISmallCompiler,    (void**) &Compiler);
    //Assert (Compiler);
}

DLL_ONTERMINATE ()
{
    MainLoopShouldStop = true;
    Sleep (500);
}
```

Файл Console3dCommands\System.cpp

```
// System.cpp - implementation of System.h, of module Console3dCommands
// created 06.12.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Console3dCommands"

#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine

ILoader*            Loader = NULL;
IConsole*           Console = NULL;

IStorageMaster*     Master = NULL;
IRenderer*          Renderer = NULL;
IEnvironment*       Environment = NULL;
IAgentsMap*         Map = NULL;
IAgentsServer*      Server = NULL;

//---------------------------------------------------------------------------------------

bool CheckModulesLoaded ()
{
    if (!Loader || !Console)
        {
        TraceEx ("Fatal: No Loader or Console");
        return false;
        }
    if (!Master || !Renderer || !Environment || !Map || !Server)
        return false;

    return true;
}
//---------------------------------------------------------------------------------------

bool LoadNotLoadedModules ()
{
    Throw ("Not implemented yet"));
}
```

Файл ConsoleExtensions\Commands.cpp

```
// Commands.cpp - implementation of Commands.h, of module ConsoleExtensions
// created 16.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleExtensions"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Commands.h"

#include <Exeption.h>      // use ExeptionEngine

#define MAX_BOOKMARKS 256

int BatchProcessing = 0;
```

```
int BatchEcho = 1;

int BatchGoto = -1;

int BatchShouldExit = 0;
int BatchBookmarks = 0;

struct BatchBookmark
{
    char name[128];
    int lineNo;
};

BatchBookmark Bookmarks[MAX_BOOKMARKS];

//-----------------------------------------------------------------------------
#include <io.h>
#include <fcntl.h>
int __cdecl StdOut (const char* format, va_list list)
    {
    //Output can be dubled in file
    return vfprintf (stdout, format, list);
    }

COMMAND_PROC_SIMPLE (CreateConsoleWindow)
{
  FreeConsole();        //эр тё ъшщ ёыєўрщ
  if ( AllocConsole() )
   {
     int hCrt = _open_osfhandle((long) GetStdHandle(STD_OUTPUT_HANDLE), _O_TEXT);
     *stdout = *(::_fdopen(hCrt, "w"));
     ::setvbuf(stdout, NULL, _IONBF, 0);

     *stderr = *(::_fdopen(hCrt, "w"));
     ::setvbuf(stderr, NULL, _IONBF, 0);

     hCrt = _open_osfhandle((long) GetStdHandle(STD_INPUT_HANDLE), _O_TEXT);
     *stdin = *(::_fdopen (hCrt, "rt"));
     ::setvbuf (stdin, NULL, _IONBF, 0);

     int v1 = 0, v2 = 0;
     Console->GetVersion (&v1, &v2);
     printf ("System console\nVersion %d.%d\n\n\n", v1, v2);
     Console->SetOutput (StdOut);
     return;
    }
}
/*
MSG msg;
    while (GetMessage(&msg, NULL,  0, 0))
    {
       TranslateMessage(&msg);
       DispatchMessage(&msg);
    }
   return TRUE;
*/
COMMAND_PROC_SIMPLE (StartCommandInterpreter)
{
    char cmd[256];

    while (strcmp (cmd, "exit\n"))
        {
        try {
            printf ("\n>");
            fgets (cmd, 255, stdin);
            //printf ("CMd is '%s'", cmd);
            Console->RaiseCommand (cmd);
            }
        catch (...){continue;}
        }
}

//-----------------------------------------------------------------------------

COMMAND_PROC_STRING (ProcessBatchFile)
{
    if (strlen(str) < 1) {c->printf ("Usage: batch <filename>\n"); return;}
    FILE* file = fopen (str, "rt");
    if (!file) {c->printf ("Cannot open \"%s\".\n", str); return;}
    c->printf ("Processing batch \"%s\"...\n", str);

    int show = Console->GetDisplay ();
    Console->SetDisplay (0);
    for (int i = 0; !feof (file); i++)
        {
        char string[256];
        fgets (string, 255, file);
        int l = strlen (string);
        if (string[l-1] == 10) string [l-1] = 0;

        if (BatchBookmarks >= MAX_BOOKMARKS - 1)
            {
            c->printf ("File contains too much bookmarks!\n");
            BatchProcessing--;
            return;
            }
        if (string[0] == ':')
```

```
            {
            strcpy ((Bookmarks [BatchBookmarks++].name), (&string[1]));
            //c->printf ("Bookmark '%s' found.\n", Bookmarks [BatchBookmarks - 1].name);
            Bookmarks [BatchBookmarks - 1].lineNo = i;
            }
        }

    BatchProcessing++;
    fseek (file, SEEK_SET, 0);

    for (int i = 0; !feof (file); i++)
        {
        char string[256];
        if (BatchGoto >= 0)
            {
            fseek (file, SEEK_SET, 0);
            for (int i = 0; i < BatchGoto; i++)
                fgets (string, 255, file);
            BatchGoto = -1;
            }

        if (!fgets (string, 255, file)) break;

        int l = strlen (string);
        if (string[l-1] == 10) string [l-1] = 0;
        if (string[0] == ':' || string[0] == '!') continue;

        try {if (string[0] == '@')
            {
            c->RaiseCommand (string+1);
            continue;
            }

        if (BatchEcho)
            c->printf ("a>%s\n", string);
        c->RaiseCommand (string);
        } catch (...){continue;}
        }
    Console->SetDisplay (show);
    BatchProcessing--;
}
//---------------------------------------------------------------------------

COMMAND_PROC_STRING (Echo)
{
    int l = 0;
    if (!(l = strlen (str)))
        {
        c->printf ("\tEcho is %s.\n", BatchEcho? "on":"off");
        return;
        }

    if (l == 3 && str[0] == 'o' && str [1] == 'f' && str [2] == 'f')
        {BatchEcho = false; return;}
    if (l == 2 && str[0] == 'o' && str [1] == 'n')
        {BatchEcho = true; return;}

    c->printf ("%s\n", str);
}
//---------------------------------------------------------------------------
COMMAND_PROC_STRING (sys)
{
    if (!strlen (str)) {c->printf ("Usage: system <command to windows>"); return;}
    system (str);
}
//---------------------------------------------------------------------------
COMMAND_PROC_LEXEMS (Goto)
{
    if (!BatchProcessing) {c->printf ("'goto' can be used only in batch.\n"); return;}
    if (!ln) {c->printf ("Error: Goto to nowhere"); return;}

    for (int i = 0; i <= BatchBookmarks; i++)
        {
        if (!strcmp (Bookmarks [i].name, data[0].data))
            {
            BatchGoto = Bookmarks [i].lineNo;
            return;
            }
        }
    c->printf ("'Goto' Cannot find bookmark %s!\n", data[0].data);
}
//---------------------------------------------------------------------------
COMMAND_PROC_LEXEMS(Delay)
{
    if (ln)
        if (data[0].type==data[0].LT_Number)
            Sleep (data[0].idata);
}

COMMAND_PROC_LEXEMS (Repeat)
{
    for (int i = 0; i < data[0].idata; i++)
        {
        Console->RaiseCommand (data[1].data);
        }
}
//---------------------------------------------------------------------------
```

```
//-------------------------------------------------------------------------------
COMMAND_PROC_SIMPLE (CmdExit)
{
    c->printf ("\nExiting...\n");
}

COMMAND_PROC_SIMPLE (Comment)
{}

int UnknownCommandHandler (IConsole*c, int, char* str)
{
    if (!strlen (str))
        return false;

    FILE* file = fopen (str, "rt");
    if (file)
        {
        fclose (file);

        for (int i = strlen (str); i > 0; i--)
            if (str[i] == '.')
                break;
        if (i != 0)
            {
            if (strcmp (".cs", str + i) &&
                strcmp (".bat", str + i) &&
                strcmp (".cb", str + i) &&
                strcmp (".txt", str + i)) return false;
            }


        char cmd [320];
        sprintf (cmd, "batch %s", str);
        c->RaiseCommand (cmd);
        return true;
        }
    return false;
}

char Prefix[256] = "";

COMMAND_PROC_LEXEMS (SetPrefix)
{
    if (!ln)
        {
        return;
        }

    c->printf ("Setting prefix to '%s'.\n", data[0].data);
    strcpy (Prefix, data[0].data);
    c->SetConsoleMode (Prefix);
}

BEGIN_COMMAND_LIST (Commands)
    DEFINE_COMMAND ("batch", ProcessBatchFile, false)
    DEFINE_COMMAND ("echo", Echo, false)
    DEFINE_COMMAND ("system", sys, false)
    DEFINE_COMMAND ("goto", Goto, true)
    DEFINE_COMMAND ("repeat", Repeat, true)
    DEFINE_COMMAND ("rem", Comment, false)
//DEFINE_COMMAND ("createwindow", CreateConsoleWindow, false)
//  DEFINE_COMMAND ("startcommandinterpreter", StartCommandInterpreter, false)
    DEFINE_COMMAND ("exit", CmdExit, false)
    DEFINE_COMMAND ("delay", Delay, false)
    DEFINE_COMMAND ("setprefix", SetPrefix, true)
END_COMMAND_LIST ();

//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------
bool ExtendConsole     (IConsole* console)
{
    bool r = Extensions->AttachCommands (CM (Commands), CMS (Commands), console, false, "Console
Extensions");
    Console->SetUnknownCommandHandler ((UnhandledCommandProc)UnknownCommandHandler, false);
    return r;
}
//-------------------------------------------------------------------------------

bool UnExtendConsole   (IConsole* console)
{
    bool r = Extensions->DetachCommands (CM (Commands), CMS (Commands), console);
    Console->SetUnknownCommandHandler (NULL, false);
    return r;
}
```

Файл ConsoleExtensions\ConsoleExtensionsCore.cpp

```
// ConsoleExtensionsCore.cpp - implementation of ConsoleExtensionsCore.h, of module ConsoleExtensions
// created 16.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
```

```
#define __MODULE_NAME  "ConsoleExtensions"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "ConsoleExtensionsCore.h"

#include <Exeption.h>      // use ExeptionEngine
#include "commands.h"

IMPLEMENT_IUNKNOWN (CConsoleExtensions, IConsoleExtensions, IID_IConsoleExtensions);
CConsoleExtensions::CConsoleExtensions (){}
CConsoleExtensions::~CConsoleExtensions(){}
//-----------------------------------------------------------------------------

bool __cdecl   CConsoleExtensions::AttachCommands (CommandDef* commands, int nc, IConsole* console, bool
replace, char* module)
{
    try {
        Assert (console && "AttachCommandsTo" && nc);
        for (int i = 1; i <= nc; i++)
            {
            try {
                console->AddCommand (commands[i].keyword, (CommandProcP) commands[i].proc,
commands[i].lexems, module);
                commands[i].attached = true;
                }
            catch (...)
                {
                if (replace)
                    {
                    console->DeleteCommand (commands[i].keyword);
                    console->AddCommand (commands[i].keyword, (CommandProcP) commands[i].proc,
commands[i].lexems, module);
                    commands[i].attached = true;
                    continue;
                    }
                throw 0;
                }
            }
        }
    catch (...){return false;}
    return true;
}
//-----------------------------------------------------------------------------

bool __cdecl   CConsoleExtensions::DetachCommands (CommandDef* commands, int nc, IConsole* console)
{
    try {
        if (!console) console = Console;
        Assert (console && nc && "DetachCommands");

        for (int i = 0; i <= nc; i++)
            {
            try
                {
                if (commands[i].attached)
                    {
                    console->DeleteCommand (commands[i].keyword);
                    commands[i].attached = false;
                    }
                }   catch (...) {continue;}
            }
    }
    catch (...)
        {
        return false;
        }
    return true;
}
//-----------------------------------------------------------------------------

bool __cdecl   CConsoleExtensions::Extend        (IConsole* console)
{
    if (!console) console = Console;
    if (!console) return false;
    return ExtendConsole (console);
}
//-----------------------------------------------------------------------------

bool __cdecl   CConsoleExtensions::UnExtend      (IConsole* console)
{
    if (!console) console = Console;
    if (!console) return false;
    return UnExtendConsole (false);
}
//-----------------------------------------------------------------------------
IConsole* __cdecl CConsoleExtensions::GetConsole  (void)
    {   return Console;}
//-----------------------------------------------------------------------------
```

Файл ConsoleExtensions\Language.cpp

```
// Language.cpp - implementation of Language.h, of module ConsoleExtensions
// created 21.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleExtensions"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Language.h"

#include <Exeption.h>      // use ExeptionEngine

void RegisterCompiler  ()
{

}
//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------

/*
▌Єю сєфхЄ фшэрьшўхёъш шэЄхЁяЁхЄшЁєхьщ  ч√ўхъ, ЇєэъЎшш ьюцэю ЁхфръЄшЁютрЄ№ "эр ївфє"
*/

struct SFunction
{
    char name[128];
    char** code;
    int codelines;
};
```

Файл ConsoleExtensions\Module.cpp

```
// Module.cpp - implementation of Module.h, of module ConsoleExtensions
// created 16.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleExtensions"

#include "StdAfx.h"
#include "System.h"

#include <Exeption.h>
#include "ConsoleExtensionsCore.h"

    #define DLL_NAME "ConsoleExtensions"
    #define DLL_CLASS CConsoleExtensions
    #include "..\Common\Utilities\Module.c"

DLL_ONLOAD (loader)
{
    Assert (loader);
    Loader = loader;
    Loader->GetInterface (IID_IConsole, (void**)&Console);
    Assert (Console);
    Console->printf ("CONEX: Console Extensions by Vasfed version 1.0\n");
}

DLL_INIT (loader)
{
    Assert (loader);
    Loader = loader;

    Loader->GetInterface (IID_IConsoleExtensions, (void**)&Extensions);
    Loader->GetInterface (IID_IConsole, (void**)&Console);

    Assert (Console && Extensions);

    Extensions->Extend (Console);
    Console->printf ("CONEX: Addintional console commands successfully loaded.\n");
}
```

Файл ConsoleExtensions\System.cpp

```
// System.cpp - implementation of System.h, of module ConsoleExtensions
// created 16.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleExtensions"
```

```
#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine

ILoader*           Loader      = NULL;
IConsole*          Console     = NULL;
IConsoleExtensions* Extensions = NULL;
```

Файл ConsoleRecursiveDescent\Module.cpp

```
// Module.cpp - implementation of Module.h, of module ConsoleRecursiveDescent
// created 06.12.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleRecursiveDescent"

#include "StdAfx.h"
#include "System.h"

#include <Exeption.h>
#include "..\Console\Console.h"
IConsole* Console = NULL;

    #define DLL_NAME "Console Recursive Descent Test Polygon"
    #include "..\Common\Utilities\Module.c"

void RD (IConsole* c, int ln, Lexem1* data);

DLL_ONLOAD (loader)
{
    Assert (loader);
    loader->GetInterface (IID_IConsole, (void**)&Console);
    Console->AddCommand ("rd", (CommandProcP)RD, true, "ReDes");
}
//-------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------------

Lexem1* lexems = NULL;
int current = 0;

int GetT();
int GetN();

////////////////////////////////
// Rule 1

int GetG0 (const char* s)
{
printf ("Postfix form: ");

S = s;
return GetT();
}

////////////////////////////////
// Rule 2

int GetT()
{
int val = GetN();
while (*S == '+' || *S == '-')
    {
    int op = *S++;
    int val2 = GetN();

    printf ("%c ", op);

    if (op == '+') val += val2;
    else           val -= val2;
    }

return val;
}

////////////////////////////////
// Rule 3

int GetN()
{
int val = 0;
//while (isdigit (*S)) val = val*10 + (*S++ - '0');

printf ("%d ", val);
return val;
}

////////////////////////////////
// Test
```

```
void main()
{
const char ex[] = "43+5-7948-72";

printf ("\nGetN (%s) returned %d\n\n", ex, GetG0 (ex));
}

void RD (IConsole* c, int ln, Lexem1* data)
{

}
```

Файл ConsoleUI\Main.cpp

```
// ConsoleUI.cpp - implementation of ConsoleUI
// created 13.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleUI"

#include <Exeption.h>      // use ExeptionEngine
#include <conio.h>

#include "..\Loader\LoaderCore.h"
#include "..\Console\Console.h"
//---------------------------------------------------------------------------------------

ILoader* Loader = NULL;
IConsole* Console = NULL;
//---------------------------------------------------------------------------------------

void main (int nparam, const char* vparam[])
{
    CreateDumpFile ("ExeptionDump.txt");

    try
    {
    Loader = new CLoader;
    AssertT (Loader, "Cannot create loader");

    Loader->LoadComponent ("Console.dll");
    Loader->LoadComponent ("conex32.dll");
    Loader->InitModules ();

    Loader->GetInterface (IID_IConsole, (void**)&Console);
    }
    Catch ()

        printf ("\nInitialization failed, error detected.\nPress any key to exit.");
        Handled ();
        getch ();
        if (Loader) delete Loader;
        return;
        }
    catch (...)
        {
        printf ("\nInitialization failed, unknown error.\n");
        char msg[256];
        FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS, NULL,
            GetLastError(), MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), (LPTSTR) msg, 256, NULL);
        MessageBox (NULL, msg, "Error text", NULL);
        if (Loader) delete Loader;
        }


    printf ("Console UI\nVersion 1.1\n\n");

    char cmd[256];

    if (nparam > 1)
        {
        char cmd[300];
        sprintf (cmd, "batch %s", vparam[1]);
        Console->printf ("Batch file %s:", vparam[1]);
        Console->RaiseCommand (cmd);
        }

    while (strcmp (cmd, "exit"))
        {
        try {
            printf ("\n>");
            gets (cmd);
            Console->RaiseCommand (cmd);
            }
        catch (...){continue;}
        }

    delete Loader;
    EndDump ();
```

```
}
```

Файл ConsoleUI2\Globals.cpp

```
// Globals.cpp - implementation of Globals.h, of module ConsoleUI
// created 16.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleUI"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Globals.h"

#include <Exeption.h>      // use ExeptionEngine

int LamerMode = 0;
```

Файл ConsoleUI2\Main.cpp

```
// Main.cpp - implementation of Main.h, of module ConsoleUI2
// created 05.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleUI2"

#define _WIN32_WINNT 0x500
#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine

#include <windows.h>
#include <windowsx.h>
#include <process.h>
#include <conio.h>
#include <time.h>

#include "System.h"
#include "globals.h"
#include "Startup.h"

//---------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------

void GetInterfaces ();

char progname[128];

char cmd[256];
bool run = true, c = false;

void PromptListener(void*)
{
    while (run)
    {
//  printf (".");
    if (Console)
        if (!Console->GetPr () && c)
        {
        Console->DoPrompt ();
        printf ("%s", cmd);
        }
    Sleep (500);
    }
}

void main (int nparam, const char* vparam[])
{
    ShowSplash ();

    TraceEx ("Something");

    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), FOREGROUND_GREEN|FOREGROUND_INTENSITY);
    SetConsoleTitle ("System console");

    time_t aclock;
    time( &aclock );
    LoadString (GetModuleHandle (NULL), IDS_ProgName, progname, 128);


    printf ("%s global version %d.%d.\n", progname, 1, 0);
    printf ("System startup time %s\n", ctime (&aclock));

    PromptExeptionEngine ();
    printf ("\n");
```

```
    try
        {
        StartKernel ();
        Loader->LoadComponent ("conex32.dll");
        Loader->InitModules ();
        }
    Catch ()

        printf ("\nSystem Initialization failed.\nCannot recover from error. Press any key to exit.");
        Handled ();
        getch ();
        return;
        }
    catch (...)
        {
        printf ("\nInitialization failed:");
        char msg[256];
        FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS, NULL,
            GetLastError(), MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), (LPTSTR) msg, 256, NULL);
        printf (" %s\n", msg);
        MessageBox (NULL, msg, "Error text", NULL);
        }

    #define LS(str,res) LoadString (GetModuleHandle (NULL), res, str, sizeof (str));


    char prompt[128];

    LS (prompt, IDS_DefaultPrompt);
    char exitCmd[64];
    LoadString (GetModuleHandle (NULL), IDS_ExitCommand, exitCmd, 64);

//  if (LamerMode)
        printf ("Type '%s' to exit. Type 'help' to get help.\n\n", exitCmd);


    _beginthread (PromptListener, 0, 0);

    HideSplash (); //system has already started
    Console->SetPrompt ("Alpha>");
    while (strcmp (cmd, exitCmd))
        {
        try {
            Console->DoPrompt ();
            ZeroMemory (cmd, 255);
            gets (cmd);
            c = false;
            Console->RaiseCommand (cmd);
            c = true;
            }
        catch (...){continue;}
        }
    run = false;
    Loader->Terminate ();
}

//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
```

Файл ConsoleUI2\Old3dUI.cpp

```
// Old3dUI.cpp - implementation of Old3dUI.h, of module ConsoleUI
// created 16.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleUI"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Old3dUI.h"
#include "startup.h"
#include <Exeption.h>      // use ExeptionEngine

//---------------------------------------------------------------------------
void InitScene     (char* filename = "meshes.xxx");
void RedrawProc    ();

void InitGraphics  ();

bool ShouldRender   ();
void StartRender   ();
void StopRender    ();

void StartPhysics  ();

void RenderProc (void*);
void KeyboardProc (HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);

//---------------------------------------------------------------------------
```

```
int ScrX = 320,
    ScrY = 240;
int AgentNum = 0;
CPoint3D Target (0.0F, 0.0F, 0.0F);
int RenderLevel = RL_WIREFRAME;
int PhysicLevel = PL_NO_MOVE;

bool StartedNormally = true;

char szProgName     []  = "Program Name";
char szWndName      []  = "OpenGL Window";
int Render = true;

//-------------------------------------------------------------------------------------

void Init ()
{
    Master -> Init (12);
    Master -> Set3DOptimizer (true);
    Map->Load ("Map.map");
    CreateAgents ();
    InitGraphics ();
    printf ("Now loading scene... ");
    InitScene ();
    printf ("OK.\n");
    Server->InitializeAgents ();
    Environment -> SetPhysicLevel (PhysicLevel = PL_MOVE_ONLY);
    StartRender ();
}

//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------
//Graphics:

void InitScene (char* filename)
    {
    Renderer -> MakeCameraCurrent (1);

    Master -> SetNewDataType (sizeof (CMesh), AgentsCount, "Mesh");
    Master -> SetNewDataType (sizeof (CPolygon), AgentsCount * 216, "Polygon");
    Master -> SetNewDataType (sizeof (CAcceleration), AgentsCount, "Acceleration");
    Master -> SetNewDataType (sizeof (CSpeed), AgentsCount, "Speed");
    Master -> SetNewDataType (sizeof (CIntersectionSphere), AgentsCount, "IntersectionSphere");
    Master -> SetNewDataType (sizeof (CIntersectionSpace), AgentsCount, "IntersectionSpace");
    Master -> SetNewDataType (sizeof (CBoundingBox), AgentsCount, "BoundingBox");
    Master -> SetNewDataType (sizeof (CPhysMaterial), AgentsCount, "Material");

    char directory[300];
    GetCurrentDirectory (300, directory);
    SetCurrentDirectory (".\\VirtualWorld\\");

    for (int a = 0; a < AgentsCount - 1; a ++)
        {
        Master -> LoadXXX ("ufo.xxx");
        }
    Master -> LoadXXX ("small.xxx");
    Master -> LoadXXX ("plants.xxx");


    Master -> Lock ("Mesh");
    for (int b = 0; b < AgentsCount; b ++)
        Master -> CreateObject ("Mesh");

    Master -> Unlock ("Mesh");
    Master -> Lock ("Polygon");
    for (int c = 0; c < AgentsCount * 216; c ++)
        Master -> CreateObject ("Polygon");
    Master -> Unlock ("Polygon");

    Master -> LoadXXX ("lights.xxx");

    Master -> Lock ("LightSource");
    Renderer -> SetLights (Master -> RetData ("LightSource"));
    Master -> Unlock ("LightSource");

    int handle = Master -> SetNewDataType (sizeof (CColorMatrix), 10, "Texture");
    Master -> Lock ("Texture");
    AssertT (handle != -1, "Memory for textures is not initialized");

    Renderer -> LoadResource (Master -> RetData ("Texture"), "textures.rrc");
    Master -> Unlock ("Texture");

    handle = Master -> SetNewDataType (sizeof (CLandscape), 1, "Landscape");
    AssertT (handle != -1, "Memory for landscape is not initialized");
    Master -> Lock ("Landscape");
    Renderer -> LoadResource (Master -> RetData ("Landscape"), "landscap.rrc");
    Master -> Unlock ("Landscape");

    Master -> Lock ("Mesh");
    Master -> Lock ("Polygon");
    Master -> Lock ("LightSource");
    Master -> Lock ("Speed");
    Master -> Lock ("Acceleration");
    Master -> Lock ("IntersectionSphere");
    Master -> Lock ("IntersectionSpace");
    Master -> Lock ("BoundingBox");
    Master -> Lock ("Material");
```

```
    Master -> Lock ("Landscape");
    Renderer -> PrepareMeshes (Master -> RetData ("Mesh"),
                               Master -> RetData ("Polygon"),
                               Master -> RetData ("LightSource"));

    for (int n = 0; n < AgentsCount; n ++)
        {
        IAgent* a = Server->GetAgent(n);
        if (!a) continue;
        ((CMesh*)(Master -> RetObject ("Mesh", AgentsCount + n))) -> SetCenter (a->m_x, a->m_y, a->m_z);
        }

    CSmartPointer MeshPtr = Master -> RetData ("Mesh");
    MeshPtr += AgentsCount;
    Environment -> MakeTable (MeshPtr,
                              Master -> RetData ("Speed"),
                              Master -> RetData ("Acceleration"),
                              Master -> RetData ("IntersectionSphere"),
                              Master -> RetData ("IntersectionSpace"),
                              Master -> RetData ("BoundingBox"),
                              Master -> RetData ("Material"),
                              Master -> RetData ("Landscape"));

    Master -> Unlock ("Mesh");
    Master -> Unlock ("Polygon");
    Master -> Unlock ("LightSource");
    Master -> Unlock ("Speed");
    Master -> Unlock ("Acceleration");
    Master -> Unlock ("IntersectionSphere");
    Master -> Unlock ("IntersectionSpace");
    Master -> Unlock ("BoundingBox");
    Master -> Unlock ("Material");
    Master -> Unlock ("Landscape");

    SetCurrentDirectory (directory);
    }

//---------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------

void RedrawProc ()
    {
    try {
        Master -> Lock ("Mesh");
        Master -> Lock ("Landscape");
        Master -> Lock ("BillBoard");
        Renderer -> Clear ();
        if (ShouldRender ())
            {
            Renderer -> CameraFollow (1, reinterpret_cast <CMesh *> (Master -> RetObject ("Mesh",
AgentsCount + AgentNum)) -> RetCenter ());
            Renderer -> CameraOnLand (1, ((CLandscape *) Master -> RetObject ("Landscape", 0)) ->
RetPHorizont (0));
            Renderer -> CameraOnLand (2, ((CLandscape *) Master -> RetObject ("Landscape", 0)) ->
RetPHorizont (0));
            Renderer -> RenderMeshes (Master -> RetData ("Mesh"));
            Renderer -> RenderLandscape (Master -> RetData ("Landscape"));
            Renderer -> RenderBillBoards (Master -> RetData ("BillBoard"));
            }

        try {
            Server->PerformAgentsStep ();
            Environment -> MoveAll ();
            }
        catch (...)
            {
            NoErrors = NoErrors == 2? false: 2;
            if (NoErrors) Console->printf ("Error happened in MultiAgentSystem.\n");
            else Console->printf ("Error happened in MultiAgentSystem.\nRendering STOPPED\n");
            }

        for (int n = 0; n < AgentsCount; n ++)
            {
            IAgent* a = Server->GetAgent (n);
            if (!a) continue;
            CMesh* mesh = (CMesh*)(Master -> RetObject ("Mesh", AgentsCount + n));
            CPoint3D p = mesh->RetCenter ();
            a->m_x = p (0);
            a->m_y = p (1);
            a->m_z = p (2);
            }

        Master -> Unlock ("Mesh");
        Master -> Unlock ("Landscape");
        Master -> Unlock ("BillBoard");
        }
    catch (...)
        {
        NoErrors = NoErrors == 3? false: 3;
        if (NoErrors == 3)
            {
            Console->printf ("Error during rendering.\n  If often happens because of powerless of 3D
system of computer.\n  Trying to fix..\n");
            Renderer->SetRenderLevel ((--RenderLevel > 0? RenderLevel: RenderLevel = RL_WIREFRAME));
            }
```

```
        else {Console->printf ("Cannot recover after error.\n Rendering stopped. System needs
restart.\n");}
        //MessageBox (NULL, "Runtime Error during rendering", "Error", MB_OK);
        //PostQuitMessage (0);
        return;
        }
        if (NoErrors == 3)
            {
            NoErrors = true;
            Console->printf ("   ok.\n\n");
            }
    }

//---------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------
bool FullScreen = false;

void InitGraphics ()
    {
    /*Renderer -> Resize (0, 0, 320, 240, false);
    Renderer -> MakeGLWindow (GetModuleHandle (NULL), SW_SHOWNORMAL, szProgName, szWndName);
    Renderer -> SetRenderLevel (RenderLevel);
    Renderer -> SetFogParameters (0.0035F, 0.0F, 50.0F);
    Renderer -> SetVolumeFogDepth (false, 0.0F);
    Renderer -> InitTransparentBuffer (1, 1);
    Renderer -> MakeCurrent ();
    Renderer -> Clear ();*/
    Renderer -> LoadResource (NULL, "settings.rrc");
    Renderer -> SetLoDLevel (1);
    Renderer -> SetKeyboardFunc (KeyboardProc);
    Renderer -> SetRedrawFunc (RedrawProc);
    Renderer -> SetCamera (0.0F, -5.0F, 1.0F, 0.0F, 0.0F, 0.0F, 0.0F, 0.0F, 1.0F,
                           -0.133F, 0.133F, -0.100F, 0.100F, 0.1F, 500.0F, true);
    Renderer -> SetCamera (0.01F, 0.01F, 10.01F, 0.0F, 0.0F, 10.0F, 0.0F, 0.0F, 1.0F,
                           -0.133F, 0.133F, -0.100F, 0.100F, 0.1F, 500.0F, true);
    Renderer -> SetClearColor (50.0F / 255.0F, 200.0F / 255.0F, 230.0F / 255.0F, 1.0F);

    Environment -> SetPhysicLevel (PhysicLevel);
    Environment -> SetGravity (0.0F, 0.0F, -9.8F);
    }
//---------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------
bool g_RenderShouldStart = false;
//---------------------------------------------------------------------------------------
bool ShouldRender       ()
    {
    return g_RenderShouldStart;
    }
//---------------------------------------------------------------------------------------
void StartRender   ()
    {
    g_RenderShouldStart = true;
    Assert (Renderer);
    }
//---------------------------------------------------------------------------------------
void StopRender    ()
    {
    g_RenderShouldStart = false;
    }

//---------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------

char* StrRenderLevels[] = {"None", "Wireframe", "Solid", "Solid&Light",
                           "LowTextured", "Textured", "Best View"};

void KeyboardProc (HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam)
    {
    switch ((char) wParam)
        {
        case 'q': Renderer -> CameraSlide (1, 1.0F);
                  break;
        case 'a': Renderer -> CameraSlide (1, -1.0F);
                  break;
        case '1': Renderer -> MakeCameraCurrent (1);
                  break;
        case '2': Renderer -> MakeCameraCurrent (2);
                  break;
        case ',': if (AgentNum > 0) AgentNum --;
                    Console->printf ("Current agent is now %d\n", AgentNum);
                  break;
        case '.': if (AgentNum < AgentsCount - 1) AgentNum ++;
                    Console->printf ("Current agent is now %d\n", AgentNum);
                  break;
        case 'f': //Trace ("FPS is %d", (int) Renderer -> RetFPS ());
                    Console->printf ("Fps is %g\n", Renderer->RetFPS ());
                  break;
        case 'i': Renderer -> CameraMove (2, 1.0F);
                  break;
        case 'k': Renderer -> CameraMove (2, -1.0F);
                  break;
        case (char) 27: PostQuitMessage (0);
                  Console->printf ("Exited by pressing Esc");
                  break;
        case '+': if (RenderLevel < RL_EXTENSIONS_USED)
                      {
```

```
                        RenderLevel ++;
                        Renderer -> SetRenderLevel (RenderLevel);
                        Console->printf ("Setting Render level to %d (%s)", RenderLevel,
StrRenderLevels[RenderLevel]);
                        if (RenderLevel == RL_EXTENSIONS_USED)
                            {
                            Console->printf (" warning: The BestView mode is not supported by some video
cards\n");
                            }
                        }
                    break;
        case '-': if (RenderLevel > RL_WIREFRAME)
                        {
                        RenderLevel --;
                        Renderer -> SetRenderLevel (RenderLevel);
                        Console->printf ("Setting Render level to %d (%s)", RenderLevel,
StrRenderLevels[RenderLevel]);
                        }
                    break;
        case '*': if (PhysicLevel < PL_BBOX_TEST)
                        {
                        PhysicLevel ++;
                        Environment -> SetPhysicLevel (PhysicLevel);
                        }
                    break;
        case '/': if (PhysicLevel > PL_NO_MOVE)
                        {
                        PhysicLevel --;
                        Environment -> SetPhysicLevel (PhysicLevel);
                        }
                    break;
        case 'c':
            {
            IAgent* a = Server->GetAgent (AgentNum);
            CPoint3D np (a->m_x, a->m_y, a->m_z);
            Renderer -> CameraFollow (2, np);
            break;
            }
        case 'e': Renderer -> SetFogParameters (1.0F, 0.0F, 30.0F);
                  Renderer -> SetVolumeFogDepth (true, 20.0F);
                  break;
        case 'd': Renderer -> SetFogParameters (0.0035F, 0.0F, 50.0F);
                  Renderer -> SetVolumeFogDepth (false, 0.0F);
                  break;
        case 'r': Renderer -> Resize (0, 0, 640, 480, false);
                  break;
        case 's': if (ScrX == 320 && ScrY == 240)
                        {
                        ScrX = 640; ScrY = 480;
                        Renderer -> Resize (0, 0, ScrX, ScrY, true);
                        break;
                        }
                  if (ScrX == 640 && ScrY == 480)
                        {
                        ScrX = 800; ScrY = 600;
                        Renderer -> Resize (0, 0, ScrX, ScrY, true);
                        break;
                        }
                  if (ScrX == 800 && ScrY == 600)
                        {
                        ScrX = 1024; ScrY = 768;
                        Renderer -> Resize (0, 0, ScrX, ScrY, true);
                        break;
                        }
                  if (ScrX == 1024 && ScrY == 768)
                        {
                        ScrX = 320; ScrY = 240;
                        Renderer -> Resize (0, 0, ScrX, ScrY, true);
                        break;
                        }
        case '3': Renderer -> SetLoDLevel (1);
                    break;
        case '4': Renderer -> SetLoDLevel (8);
                    break;
        default:  break;
        }
    }
```

Файл ConsoleUI2\Startup.cpp

```
// Startup.cpp - implementation of Startup.h, of module ConsoleUI
// created 16.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleUI"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "globals.h"
#include "Startup.h"
#include "Resource.h"
```

```
#include <Exeption.h>      // use ExeptionEngine

bool StartKernel ()
{
    if (!CreateLoader ()) return false;
    if (!CreateConsole(true)) return false;
    Trace ("Main: Loader and Console started.");
    return true;
}

//------------------------------------------------------------------------------

bool CreateLoader  ()
{
    try
        {
        char LoaderDllName[256];
        LoadString (GetModuleHandle (NULL), IDS_LoaderDllName, LoaderDllName, 256);

        Trace ("Main: Starting Loader from library '%s'...", LoaderDllName);

        HMODULE LoaderDll = LoadLibrary (LoaderDllName);
        AssertT (LoaderDll, "Cannot load library");

        typedef void (*DLLGETINSTANCE)(void**);
        DLLGETINSTANCE LoaderGetInstance = (DLLGETINSTANCE)GetProcAddress (LoaderDll, "DllGetInstance");
        AssertT (LoaderGetInstance, "Module does not contain Loader");

        LoaderGetInstance ((void**)&Loader);
        AssertT (Loader, "Module failed to create Loader");
        }
    catch (...)
        {
        Trace ("Main: Loader creation failed");
        return false;
        }
    return true;
}
//------------------------------------------------------------------------------
bool CreateConsole (bool nl)
{
    char consoleDllName[256];
    LoadString (GetModuleHandle (NULL), IDS_ConsoleDllName, consoleDllName, 256);
    Trace ("Main: Creating Console from library '%s'", consoleDllName);

    Loader->LoadComponent (consoleDllName);
    Loader->GetInterface (IID_IConsole, (void**)&Console);
    if (!Console) return false;

    Loader->InitModules ();
    if (nl) Loader->Init (true);
    return true;
}
//------------------------------------------------------------------------------

HWND SplashWnd = NULL;

LRESULT CALLBACK SplashWindowProc(
  HWND hwnd,      // handle to window
  UINT uMsg,      // message identifier
  WPARAM wParam,  // first message parameter
  LPARAM lParam   // second message parameter
)
{
    switch (uMsg)
        {
        case WM_CLOSE:
            return false;
            break;
        case WM_PAINT:
            PAINTSTRUCT ps;
            BeginPaint (hwnd, &ps);
            RECT rect;
            rect.top = 30;
            rect.bottom = 50;
            rect.left = 0;
            rect.right = 200;
            //DrawText (ps.hdc, "Elendor", strlen ("Elendor"), &rect, 0);

            EndPaint (hwnd, &ps);
            break;
        default:
            return true;
            //return DefDlgProc (hwnd, uMsg, wParam, lParam);
        break;
        }
    return false;
}

WNDCLASS SplashCls =
    {
    0, SplashWindowProc, 0, 0, GetModuleHandle (NULL),
    0, 0, (HBRUSH)COLOR_WINDOW, 0, "MainProgramSplashWindowClass"
    };

//------------------------------------------------------------------------------
```

```
void ShowSplash ()
{
    return;

    RegisterClass (&SplashCls);
    SplashWnd = CreateWindow ("MainProgramSplashWindowClass", "Splash Window", WS_POPUP | WS_VISIBLE |
WS_THICKFRAME, 100, 100, 300, 200, NULL, NULL, GetModuleHandle (NULL), NULL);
    if (!SplashWnd) return;
    ShowWindow (SplashWnd, SW_SHOW);
    UpdateWindow (SplashWnd);
    RedrawWindow (SplashWnd, NULL, NULL, NULL);
    Sleep (5000);
}
//---------------------------------------------------------------------------

void HideSplash ()
{
    return;
    SendMessage (SplashWnd, WM_CLOSE, 0, 0);
    UnregisterClass ("MainProgramSplashWindowClass", GetModuleHandle (NULL));
    SplashWnd = NULL;
}
```

Файл ConsoleUI2\System.cpp

```
// System.cpp - implementation of System.h, of module ConsoleUI2
// created 05.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleUI2"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine

ILoader*        Loader = NULL;
IConsole*       Console = NULL;

ISoundEngine*           Sound = NULL;
IRenderer*              Renderer = NULL;
IEnvironment*           Environment = NULL;
IStorageMaster*         Master = NULL;
IAgentsServer*          Server = NULL;
IAgentsServerAgents*    ServerA = NULL;
IAgentsMap*             Map = NULL;
//ISmallCompiler*   Compiler = NULL;
```

Файл ConsoleUI2\Termination.cpp

```
// Termination.cpp - implementation of Termination.h, of module ConsoleUI
// created 16.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleUI"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Termination.h"

#include <Exeption.h>      // use ExeptionEngine
```

Файл ConsoleUI2\Utilities.cpp

```
// Utilities.cpp - implementation of Utilities.h, of module ConsoleUI
// created 17.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleUI"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Utilities.h"

#include <Exeption.h>      // use ExeptionEngine

int __cdecl printf (int level, unsigned int resourceId, ...)
{
```

```
    char str[256];
    if (!LoadString (GetModuleHandle (NULL), resourceId, str, 256)) return NULL;
    va_list list;
    va_start (list, resourceId);
    int r = 0;
    if (Console) r = Console->vprintf (level, str, list);
    else r = vprintf (str, list);
    va_end (list);
    return r;
}

int __cdecl printf (unsigned int res, ...)
{
    va_list list;
    va_start (list, res);
    int r = printf (1, res, list);
    va_end (list);
    return r;
}
```

Файл ConsoleUI2\Work.cpp

```
// Work.cpp - implementation of Work.h, of module ConsoleUI
// created 16.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleUI"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Work.h"
#include <process.h>
#include <Exeption.h>      // use ExeptionEngine
#include <errno.h>

bool MainLoopPaused = false;
bool MainLoopShouldStop = false;
uintptr_t MainLoopProcessHandle = 1L;

void MainLoop (void*);
//-----------------------------------------------------------------------------

bool StartMainLoop ()
{
    if ((MainLoopProcessHandle = _beginthread (MainLoop, NULL, NULL)) == 1L)
        {
        if (errno == EAGAIN)
            {
            Console->printf (5, "STARTER: Cannot start MainLoop: there's too many threads in system.\n");
            Console->printf (10, "You may try to close some applications to solve this problem.\n");
            }
        else {Console->printf (5, "STARTER: Cannot start MainLoop: unknown error.\n");}
        return false;
        }
    Console->printf (6, "STARTER: MainLoop started. Process handle is 0x%X\n", MainLoopProcessHandle);
    return true;
}
//-----------------------------------------------------------------------------

void PauseMainLoop ()
{

}
//-----------------------------------------------------------------------------

void StopMainLoop ()
{

}
//-----------------------------------------------------------------------------

bool GetMainLoopShouldStop ()
{
    return MainLoopShouldStop;
}
//-----------------------------------------------------------------------------

bool GetMainLoopPaused ()
{
    return MainLoopPaused;
}
//-----------------------------------------------------------------------------
bool WasMainLoopStarted (){return (MainLoopProcessHandle == 1L);}
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------

void MainLoop (void*)
{
    MSG lpMsg;
    while (!MainLoopShouldStop)
        {
```

```
        if (MainLoopPaused)
            {
            Sleep (100);
            continue;
            }
        try {
            if (PeekMessage (&lpMsg, NULL, 0, 0, PM_NOREMOVE))
                {
                if (GetMessage(&lpMsg, NULL, 0, 0) == 0) break;
                TranslateMessage(&lpMsg);
                DispatchMessage(&lpMsg);
                }
            else
                if (NoErrors) Renderer -> Render ();
            }
        Catch ()}
        catch (...)
            {
            Trace ("Unknown excetion happened.");
            }
        }
}
```

Файл ConsoleUIDll\ConsoleUICore.cpp

```
// ConsoleUICore.cpp - implementation of ConsoleUICore.h
// created 25.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "ConsoleUICore"

#include "StdAfx.h"      // use precompiled header
#include "ConsoleUI.h"
#include "ConsoleUICore.h"

#include <Exeption.h>      // use ExeptionEngine
#include "System.h"
#include "ConsoleWindow.h"

//-----------------------------------------------------------------------------
IMPLEMENT_IUNKNOWN (CConsoleUI, IConsoleUI, IID_IConsoleUI);

CConsoleUI::CConsoleUI()
    {m_window = NULL;}

CConsoleUI::~CConsoleUI()
    {
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    if (m_window) delete m_window;}

void CConsoleUI::ShowWindow (int cmdshow)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    if (!m_window)
        Init ();
    m_window->ShowWindow (cmdshow);
}
//-----------------------------------------------------------------------------
void CConsoleUI::BringToTop (void)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    if (!m_window)
        Init ();
    m_window->BringWindowToTop ();
}
//-----------------------------------------------------------------------------
void CConsoleUI::Init ()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    Assert (!m_window);
    MainWindow = m_window = new CConsoleWindow ();
    AssertT (m_window, "Failed to create Console Window");
}
```

Файл ConsoleUIDll\ConsoleUIDll.cpp

```
// ConsoleUIDll.cpp : Defines the initialization routines for the DLL.
//
#define __PROGRAMMER "Vasfed"
#define __MODULE_NAME "ConsoleUI"

#include "stdafx.h"
#include "ConsoleUIDll.h"
#include "system.h"
#include <exeption.h>

#ifdef _DEBUG
```

```
#define new DEBUG_NEW
#endif

// CConsoleUIDllApp

BEGIN_MESSAGE_MAP(CConsoleUIDllApp, CWinApp)

END_MESSAGE_MAP()

// CConsoleUIDllApp construction

CConsoleUIDllApp::CConsoleUIDllApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}

// The one and only CConsoleUIDllApp object

CConsoleUIDllApp theApp;

// CConsoleUIDllApp initialization

BOOL CConsoleUIDllApp::InitInstance()
{
    CWinApp::InitInstance();

    HInstance = this->m_hInstance;

    return TRUE;
}
```

Файл ConsoleUIDll\ConsoleWindow.cpp

```
// ConsoleWindow.cpp : implementation file
//

#include "stdafx.h"
#include "ConsoleUIDll.h"
#include "ConsoleWindow.h"

#include "system.h"
#include <exeption.h>
#include "resource.h"
// CConsoleWindow dialog

IMPLEMENT_DYNAMIC(CConsoleWindow, CDialog)
CConsoleWindow::CConsoleWindow(CWnd* pParent)
{
    HGLOBAL dlg = NULL;
    dlg = LoadResource (HInstance,
        FindResource (HInstance, (LPCSTR)IDD_ConsoleDialog, (LPCSTR)RT_DIALOG));
    AssertT (dlg, "Resource error");
    Assert (CDialog::CreateIndirect (dlg, pParent, HInstance));
    m_hook = NULL;
}

CConsoleWindow::~CConsoleWindow()
{
    if (m_hook)
        UnhookWindowsHookEx (m_hook);
}

void CConsoleWindow::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_LIST1, m_List);
    DDX_Control(pDX, IDC_EDIT1, m_Edit);
    DDX_Control(pDX, IDC_Enter, m_Button);
}

BEGIN_MESSAGE_MAP(CConsoleWindow, CDialog)
    ON_BN_CLICKED(IDC_Enter, OnBnClickedEnter)
    ON_WM_SIZE()
    ON_WM_CLOSE()
    ON_WM_KEYDOWN()
END_MESSAGE_MAP()

// CConsoleWindow message handlers

void CConsoleWindow::OnBnClickedEnter()
{
    char cmd[256];
    #define prompt ">"

    strcpy (cmd, prompt);
    int l = strlen (cmd);
    m_Edit.GetWindowText (cmd + l, 256 - l);

    m_List.AddString (cmd);
```

```
    m_List.SetCurSel (m_List.GetCount () - 1);

    m_Edit.SetWindowText ("");
    RedrawWindow ();

    Console->RaiseCommand (cmd + 1);
}
//-----------------------------------------------------------------------------
void CConsoleWindow::OnSize(UINT nType, int cx, int cy)
{
    CDialog::OnSize(nType, cx, cy);

    this->m_List.MoveWindow (10, 10, cx-20, cy - 60, true);
    this->m_Edit.MoveWindow (10, cy - 45, cx-20, 20, true);
    this->m_Button.MoveWindow (10, cy-15, cx-20, 10, true);
}
//-----------------------------------------------------------------------------

LRESULT CALLBACK KeyboardHook(int code, WPARAM wParam, LPARAM lParam)
{
    if (wParam == VK_RETURN && code == HC_ACTION)
        {
        MainWindow->OnBnClickedEnter ();
        return true;
        }   return CallNextHookEx (MainWindow->m_hook, code, wParam, lParam);
}

BOOL CConsoleWindow::OnInitDialog(void)
{
    CDialog::OnInitDialog ();
    int Ver1 = 0, Ver2 = 0;
    Console->GetVersion (&Ver1, &Ver2);

    char str[256];
    sprintf (str, "System Console Version %d.%d", Ver1, Ver2);
    m_List.AddString (str);
    m_List.AddString ("by Vasfed, all rights reserved.");
    m_List.AddString ("");
    m_hook = SetWindowsHookEx (WH_KEYBOARD, KeyboardHook, HInstance, NULL);
    return true;
}

void CConsoleWindow::OnClose()
{
    switch (MessageBox ("Close all the program or only console?\nYes - closes all\nNo - hides
console\nCancel - no comments", "Exit", MB_DEFBUTTON2 | MB_YESNOCANCEL | MB_ICONQUESTION))
        {
        case IDYES:
            PostQuitMessage (0);
            break;
        case IDNO:
            CDialog::OnClose();
            break;
        case IDCANCEL:
            break;
        default:
            Throw ("It must not be executed"));
            break;
        };
}

void CConsoleWindow::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    CDialog::OnKeyDown(nChar, nRepCnt, nFlags);
}
```

Файл ConsoleUIDll\System.cpp

```
// System.cpp - implementation of System.h
// created 25.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "System"

#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine
#include "ConsoleUICore.h"
#include "stdarg.h"

//-----------------------------------------------------------------------------
ILoader*    Loader      = NULL;
IConsole* Console   = NULL;
HINSTANCE HInstance = NULL;
CConsoleWindow* MainWindow = NULL;
IConsoleUI* ConsoleUI = NULL;
//-----------------------------------------------------------------------------

    #define DLL_NAME "ConsoleUI"
    #define DLL_CLASS CConsoleUI
```

```
    #define DLL_NO_DLLMAIN
    #include "..\Common\Utilities\Module.c"
//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------

int ExitProc (IConsole* c, int, void*)
    {c->printf ("Exiting.."); PostQuitMessage (0); return true;}
//-------------------------------------------------------------------------------------

int __cdecl ConsoleOut (const char* format, va_list list)
{
    if (!format || !list) return false;

    char str [256];
    int a = vsprintf (str, format, list);
    int strl = strlen (str);
    for (int i = 0; i < strl; i++)
        if (str[i] == '\t') str[i] = ' ';

    char* s = strtok (str, "\n");
        if (strlen (s) >= 1) MainWindow->m_List.AddString (s);
        else MainWindow->m_List.AddString  (" ");

    while (s = strtok (NULL, "\n\0"))
        if (strlen (s) >= 1) MainWindow->m_List.AddString (s);
        else MainWindow->m_List.AddString  (" ");

    MainWindow->m_List.SetCurSel (MainWindow->m_List.GetCount () -1 );
    MainWindow->RedrawWindow ();
    return a;
}
//-------------------------------------------------------------------------------------
int Cls (IConsole* c, int, void*p)
{
    MainWindow->m_List.ResetContent ();
    if (p) if (strlen ((char*)p)) MainWindow->m_List.AddString ((char*)p);
    MainWindow->m_List.AddString ("Console'as been cleared");
    MainWindow->m_List.AddString (" ");
    MainWindow->m_List.AddString (" ");
    return true;
}

//-------------------------------------------------------------------------------------
int CreateConsoleWindow (IConsole*, int, void*)
{
    ConsoleUI->ShowWindow (SW_SHOW);
    MessageBox (0, 0, 0, 0);
    return true;
}

DLL_INIT (loader)
{
    Assert (loader);
    Loader = loader;
    Loader->GetInterface (IID_IConsole, (void**)&Console);
    Loader->GetInterface (IID_IConsoleUI, (void**)&ConsoleUI);

    Assert (Console);

    Console->AddCommand ("exit", ExitProc, false);

    //Console->DeleteCommand ("createwindow");
    //Console->AddCommand ("createwindow", (CommandProcP) CreateConsoleWindow, false);
    Console->SetOutput (ConsoleOut);
    Console->AddCommand ("cls", Cls, false);
}
```

Файл Core2D\Console.cpp

```
// Console.cpp - implementation of Console.h
// created 21.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Console"

#include "StdAfx.h"      // use precompiled header
#include "system.h"
#include "Console.h"

#include <Exeption.h>      // use ExeptionEngine

int ShowWindow (IConsole* console, int i, void* p)
{
    Core2D->ShowWindow (SW_SHOW);
    return true;
}

//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------

void ConfigureConsole ()
```

```
{
    Assert (Console);
    Console->AddCommand ("show2d", (CommandProcP)ShowWindow, false);

}
```

Файл Core2D\Core2D.cpp

```
// Core2D.cpp : Defines the initialization routines for the DLL.
//

#include "stdafx.h"
#include "Core2D.h"
#include "system.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

BEGIN_MESSAGE_MAP(CCore2DApp, CWinApp)
END_MESSAGE_MAP()

// CCore2DApp construction

CCore2DApp::CCore2DApp()
{

}

// The one and only CCore2DApp object

CCore2DApp theApp;

// CCore2DApp initialization

BOOL CCore2DApp::InitInstance()
{
    Instance = this->m_hInstance;
    CWinApp::InitInstance();

    return TRUE;
}
```

Файл Core2D\Core2DCore.cpp

```
// Core2DCore.cpp - implementation of Core2DCore.h
// created 19.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Core2DCore"

#include "StdAfx.h"      // use precompiled header
#include "Core2DCore.h"
#include "resource.h"
#include <Exeption.h>      // use ExeptionEngine

CCore2D::CCore2D():m_mw(NULL){AFX_MANAGE_STATE(AfxGetStaticModuleState());}
CCore2D::~CCore2D(){if (m_mw) delete m_mw;}

void    CCore2D::ShowWindow (int cmdshow, int x, int y, int cx, int cy)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    if (!m_mw)
        this->Init ();

    m_mw->ShowWindow (SW_SHOW);
    //if (x >= 0)m_mw->MoveWindow (x, y, cx, cy, true);
}

HDC     CCore2D::GetDC      ()
{
    return NULL;
}

void    CCore2D::DrawFrame ()
{
    if (!m_mw)
        this->Init ();
    m_mw->DrawFrame ();
}

void CCore2D::Init ()
{
    if (!m_mw)
        {
        m_mw = new CMainWindow (/*UI? UI->GetWnd (): */NULL);
        Assert (m_mw);
```

```
        }
}

IMPLEMENT_IUNKNOWN (CCore2D, ICore2D, IID_ICore2D);
```

Файл Core2D\MainWindow.cpp

```
// MainWindow.cpp : implementation file
//

#include "stdafx.h"
#include "Core2D.h"
#include "MainWindow.h"
#include "system.h"
#include <exeption.h>

// CMainWindow dialog

IMPLEMENT_DYNAMIC(CMainWindow, CDialog)
CMainWindow::CMainWindow(CWnd* pParent /*=NULL*/)
: m_draw(false)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    HGLOBAL res = LoadResource (Instance, FindResource(Instance, (LPCSTR)IDD_MainDialog,
(LPCSTR)RT_DIALOG));
    Assert (res);
    AssertT (CDialog::CreateIndirect (res, pParent, Instance), "Cannot initialize MWnd");
    m_zoom = 1;
}

CMainWindow::~CMainWindow()
{
}

void CMainWindow::DoDataExchange(CDataExchange* pDX)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    CDialog::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_ZoomValueDisplay, m_zoomValueBuddy);
    DDX_Control(pDX, IDC_SPIN1, m_zoomSpinner);
    DDX_Control(pDX, IDC_MousePosDispl, m_MPDispl);
}

BEGIN_MESSAGE_MAP(CMainWindow, CDialog)
    ON_WM_SIZE()
    ON_WM_CLOSE()
    ON_BN_CLICKED(IDC_TurnOn, OnBnTurnOnTimer)
    ON_BN_CLICKED(IDC_TurnOff, OnBnClickedTurnoff)
    ON_NOTIFY(UDN_DELTAPOS, IDC_SPIN1, OnDeltaposSpin1)
    ON_WM_TIMER()
    ON_BN_CLICKED(IDC_DrawOneFrame, OnBnClickedDrawoneframe)
    ON_WM_VSCROLL()
    ON_WM_PAINT()
    ON_WM_KEYDOWN()
    ON_WM_HSCROLL()
    ON_WM_MOUSEWHEEL()
    ON_WM_MOUSEMOVE()
    ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()

// CMainWindow message handlers

void CMainWindow::OnSize(UINT nType, int cx, int cy)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    CDialog::OnSize(nType, cx, cy);
}

void CMainWindow::OnClose()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    //CDialog::OnClose();
    ShowWindow (SW_HIDE);
    KillTimer (1);
}

void CMainWindow::OnBnTurnOnTimer()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    m_draw = true;
    SetTimer (1, 500, NULL);
}

void CMainWindow::OnBnClickedTurnoff()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    m_draw = false;
    KillTimer (1);
}
```

```
void CMainWindow::OnDeltaposSpin1(NMHDR *pNMHDR, LRESULT *pResult)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    LPNMUPDOWN pNMUpDown = reinterpret_cast<LPNMUPDOWN>(pNMHDR);
    if (m_zoom != pNMUpDown->iPos + pNMUpDown->iDelta)
        {
        m_zoom = pNMUpDown->iPos + pNMUpDown->iDelta;
        RedrawWindow();
        }
    *pResult = 0;
}

BOOL CMainWindow::OnInitDialog(void)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    CDialog::OnInitDialog ();

    m_zoomSpinner.SetBuddy (&m_zoomValueBuddy);
    m_zoomSpinner.SetRange (1, 50);
    m_zoomSpinner.SetPos (1);

    SetScrollRange (SB_VERT, 0, 100);
    SetScrollRange (SB_HORZ, 0, 100);
    SetScrollPos (SB_VERT, 0);
    SetScrollPos (SB_HORZ, 0);


    return true;
}

void CMainWindow::OnTimer(UINT nIDEvent)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    //CDialog::OnTimer(nIDEvent);
    DrawFrame ();
}
/*
EMC_Road    = 0x1,   //ъюЁшўэхт√щ  (RGB(150, 100, 0))
    EMC_Field   = 0x2,   //чхыхэ√щ      (RGB (0, 255, 0))
    EMC_Forest  = 0x3,      //dark green (RGB (0, 128, 0))
    EMC_Rock    = 0x4,   //ёхЁ√щ        (RGB (100, 100, 100))
    EMC_Water   = 0x5,   //ёшэшщ        (RGB (0, 0, 255))
    EMC_House   = 0x6,   //Ёючют√щ      (RGB(255, 128, 255))
    EMC_Castle  = 0x7,   //ъЁёэ√щ      (RGB (255, 0, 0))

*/

int MapColors[] = {RGB (0, 0, 0),
                   RGB(150, 100, 0),
                   RGB (0, 255, 0),
                   RGB (0, 128, 0),
                   RGB (100, 100, 100),
                   RGB (0, 0, 255),
                   RGB(255, 128, 255),
                   RGB (0, 0, 255),//
                   0, 0, 0
                   };

void CMainWindow::DrawFrame(void)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    CDC* dc = GetDC ();
    if (!dc) return;

    if (!Server)
        {
        Loader->GetInterface (IID_IAgentsServer, (void**)&Server);
        Assert (Server);
        AServer = Server->GetAgentsInterface ();
        }

    dc->SelectObject (GetStockObject (DC_BRUSH));
    dc->SelectObject (GetStockObject (DC_PEN));

    if (!AServer)
        {
        dc->SetTextColor (RGB (255, 255, 255));
        dc->SetBkMode (TRANSPARENT);
        dc->TextOut (10, 100, "Current configuration does not contain valid Map and/or Server");
        ReleaseDC (dc);
        return;
        }

    int X = 0, Y = 0;
    RECT rect;
    GetClientRect (&rect);

    int color = RGB (0, 255, 0);

    int xfrom = GetScrollPos (SB_HORZ) * AServer->GetMapSizeX () / 100,
        xto = rect.right/m_zoom;
    int yfrom = GetScrollPos (SB_VERT) * AServer->GetMapSizeY () / 100,
        yto = rect.bottom/m_zoom;

    if (xto >= AServer->GetMapSizeX ())
        xto = AServer->GetMapSizeX ();
```

```
    if (yto >= AServer->GetMapSizeY ())
        yto = AServer->GetMapSizeY ();

    if (xto <= 0)
        xto = 0;

    if (yto <= 0)
        yto = 0;


    int yoff = 25;

    for (int x = xfrom; x < xto; x++)
        for (int y = yfrom; y < yto; y++)
            {
            //MapUnit* u = Map->GetUnit (x, y);
            int color = 0;
            //if (!u) break;
            int code = AServer->GetMapUnitCode (x, y);
            color = MapColors [(code & 0xf)];

            dc->SetDCBrushColor (color);
            dc->SetDCPenColor (color);

            if (color)
                if (m_zoom > 1)
                    {
                    dc->Rectangle  ((x - xfrom)*m_zoom, yoff + (y - yfrom)*m_zoom,
                                (x - xfrom + 1)*m_zoom, yoff + (y - yfrom + 1)*m_zoom);
                    }
                else
                    {
                    dc->SetPixel (x - xfrom, yoff + y - yfrom, color);
                    }

            if (AServer->GetMapUnitData (x, y, 0))
                {
                if (m_zoom > 1)
                    {
                    dc->SetDCBrushColor (RGB (255, 0, 0));
                    dc->SetDCPenColor (RGB (255, 0, 0));
                    dc->Rectangle  ((x - xfrom)*m_zoom, yoff + (y - yfrom)*m_zoom,
                                (x - xfrom + 1)*m_zoom, yoff + (y - yfrom + 1)*m_zoom);
                    }
                else
                    {
                    dc->SetPixel (x - xfrom, yoff + y - yfrom, RGB (255, 0, 0));
                    }
                }
            }
    ReleaseDC (dc);
}

void CMainWindow::OnBnClickedDrawoneframe()
{
    DrawFrame ();
}

void CMainWindow::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    if (pScrollBar) return;
    switch (nSBCode)
        {
        case SB_THUMBPOSITION:
        case SB_THUMBTRACK:
            SetScrollPos (SB_VERT, nPos);
            RedrawWindow ();
            break;
        case SB_LINELEFT:
            nPos = GetScrollPos (SB_VERT);
            SetScrollPos (SB_VERT, nPos-2);
            RedrawWindow ();
            break;
        case SB_LINERIGHT:
            nPos = GetScrollPos (SB_VERT);
            SetScrollPos (SB_VERT, nPos+2);
            RedrawWindow ();
            break;
        case SB_PAGEDOWN:
            nPos = GetScrollPos (SB_VERT);
            SetScrollPos (SB_VERT, nPos+20);
            RedrawWindow ();
            break;
        case SB_PAGEUP:
            nPos = GetScrollPos (SB_VERT);
            SetScrollPos (SB_VERT, nPos-20);
            RedrawWindow ();
            break;
        default:
            CDialog::OnVScroll(nSBCode, nPos, pScrollBar);
            break;
        };
    RedrawWindow ();
    if (!m_draw) DrawFrame ();

}
```

```
void CMainWindow::OnPaint()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    CPaintDC dc(this);
    if (m_draw)
        {
        DrawFrame ();
        }
}

void CMainWindow::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{

    CDialog::OnKeyDown(nChar, nRepCnt, nFlags);
}

void CMainWindow::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    if (pScrollBar) return;
    switch (nSBCode)
        {
        case SB_THUMBPOSITION:
        case SB_THUMBTRACK:
            SetScrollPos (SB_HORZ, nPos);
            RedrawWindow ();
            break;
        case SB_LINELEFT:
            nPos = GetScrollPos (SB_HORZ);
            SetScrollPos (SB_HORZ, nPos-2);
            RedrawWindow ();
            break;
        case SB_LINERIGHT:
            nPos = GetScrollPos (SB_HORZ);
            SetScrollPos (SB_HORZ, nPos+2);
            RedrawWindow ();
            break;
        case SB_PAGEDOWN:
            nPos = GetScrollPos (SB_HORZ);
            SetScrollPos (SB_HORZ, nPos+20);
            RedrawWindow ();
            break;
        case SB_PAGEUP:
            nPos = GetScrollPos (SB_HORZ);
            SetScrollPos (SB_HORZ, nPos-20);
            RedrawWindow ();
            break;
        default:
            CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
            break;
        };
    RedrawWindow ();
    if (!m_draw) DrawFrame ();
}

BOOL CMainWindow::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)
{
    if (!zDelta) return true;

    if (nFlags & MK_CONTROL)
        {
        m_zoom += (zDelta / WHEEL_DELTA);
        int min = 1, max = 1;
        m_zoomSpinner.GetRange (min, max);

        if (m_zoom <= min) m_zoom = min;
        if (m_zoom >= max) m_zoom = max;
        m_zoomSpinner.SetPos (m_zoom);
        RedrawWindow ();
        if (!m_draw) DrawFrame ();
        return true;
        }

    return CDialog::OnMouseWheel(nFlags, zDelta, pt);
}

void CMainWindow::OnMouseMove(UINT nFlags, CPoint point)
{
    ProcessMouse (point);
    CDialog::OnMouseMove(nFlags, point);
}

void CMainWindow::ProcessMouse (CPoint point)
{
    char str [128];
    RECT rect;
    GetClientRect (&rect);

    int xfrom = GetScrollPos (SB_HORZ) * AServer->GetMapSizeX () / 100 - AServer->GetMapSizeX () / 2;
    int yfrom = GetScrollPos (SB_VERT) * AServer->GetMapSizeY () / 100 - AServer->GetMapSizeY () / 2;
    int yoff = 25;
    sprintf (str, "Mouse (%.3d;%.3d)", point.x/m_zoom + xfrom, (point.y-yoff)/m_zoom + yfrom);
    m_MPDispl.SetWindowText (str);
}

void CMainWindow::OnLButtonDown(UINT nFlags, CPoint point)
{
    ProcessMouse (point);
```

```
    CDialog::OnLButtonDown(nFlags, point);
}
```

Файл Core2D\System.cpp

```
// System.cpp - implementation of System.h
// created 19.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "System"

#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine
#include "Core2Dcore.h"
#include "Console.h"

    #define DLL_NAME "Core2D"
    #define DLL_CLASS CCore2D
    #define DLL_ISUI
/// #include "..\Common\Utilities\Module.c"

ILoader*        Loader = NULL;
IAgentsMap*     Map = NULL;
IAgentsServer*  Server = NULL;
ICore2D*        Core2D = new CCore2D;
IConsole*       Console = NULL;
//IMFCUI*           UI = NULL;
HINSTANCE       Instance = NULL;
IAgentsServerAgents* AServer = NULL;

void extern "C" __declspec (dllexport) DllInit (ILoader* loader)
{
    Assert (loader);
    Loader = loader;

    Loader->GetInterface (IID_IAgentsMap, (void**)&Map);
    Assert (Map);
    Loader->GetInterface (IID_IAgentsServer, (void**)&Server);
    Assert (Server);
    AServer = Server->GetAgentsInterface ();
    Loader->GetInterface (IID_IConsole, (void**)&Console);
    Assert (Console);
    //Loader->GetInterface (IID_IMFCUI, (void**)&UI);

    ConfigureConsole();
}

#include "..\Loader\Loader.h"
#include <string.h>

#ifndef DLL_NAME
    #define DLL_NAME "Unknown"
    #endif

#ifndef DLL_PROGRAMMERS
        #ifdef __PROGRAMMER
            #define DLL_PROGRAMMERS __PROGRAMMER
        #else
            #define DLL_PROGRAMMERS "Unknown"
        #endif
    #endif

#ifndef DLL_HASDEMO
    #define DLL_HASDEMO false
    #else
    #undef DLL_HASDEMO
    #define DLL_HASDEMO true
    #endif

#ifndef DLL_CANSTARTSYSTEM
    #define DLL_CANSTARTSYSTEM false
    #else
    #undef DLL_CANSTARTSYSTEM
    #define DLL_CANSTARTSYSTEM true
    #endif

#ifndef DLL_ISUI
    #define DLL_ISUI false
    #else
    #undef DLL_ISUI
    #define DLL_ISUI true
    #endif

DllInfo DLL_INFO = {DLL_NAME, DLL_NAME, DLL_PROGRAMMERS, false, DLL_HASDEMO, DLL_CANSTARTSYSTEM, DLL_ISUI,
false};

#ifndef DLL_CLASS
    #error Every module must have a COM-like structure! (Class:interface:IUnknown)
    #endif
```

```
extern "C" __declspec (dllexport) void __cdecl DllGetInfo     (void** ppr)
    {if (ppr) *ppr = &DLL_INFO;}
extern "C" __declspec (dllexport) void __cdecl DllGetInstance (void** ppr)
    {if (ppr) {Core2D->AddRef(); *ppr = Core2D;}}
```

Файл Core2D\module.cpp

```
// module.cpp - implementation of module.h
// created 21.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "module"

#include "StdAfx.h"      // use precompiled header
#include "module.h"

#include <Exeption.h>      // use ExeptionEngine
```

Файл Core3D\core3d.cpp

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** CORE3D.CPP                                                        ***/
/*** ╬╬╚╤╘╬╪═╚╬ ╠╫╫╬╪╬╫ ┴└╠╬╥▌┌ ╬╤╘╬╫╥╥┬, ╬╨┌▀┬╥╪═▌┌ ┬ CORE3D.H       ***/
/***                                                                   ***/
/*** └└╔╥ =└═╚╤└═ ╥└╤╙╙╚═╬╫▌╠ └═╨┼┼╠ ┬╪╨┼┼┼┬╚╫┼╠ ┬ 2002 ┼╬╨╙          ***/
/***                                                                   ***/
/***                                                                   ***/
/*************************************************************************/
/*************************************************************************/

#include "core3d.h"

#define __PROGRAMMER "Xroft"
#define __MODULE_NAME "Core3D"

#include <exeption.h>

BOOL APIENTRY DllMain(HANDLE hModule,
                      DWORD  ul_reason_for_call,
                      LPVOID lpReserved)
{
return 1;
}

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////  CLASS CCOLOR  ////////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

#define COLOR_FLOATS 4

/*************************************************************************/

CColor::CColor ()
    {
    }

/*************************************************************************/

CColor::CColor (float * rgba)
    {
    myRGBA [0] = rgba [0];
    myRGBA [1] = rgba [1];
    myRGBA [2] = rgba [2];
    myRGBA [3] = rgba [3];
    }

/*************************************************************************/

CColor::CColor (float r, float g, float b, float a)
    {
    myRGBA [0] = r;
    myRGBA [1] = g;
    myRGBA [2] = b;
    myRGBA [3] = a;
    }
```

```
/*************************************************************************/

CColor::~CColor ()
    {
    }

/*************************************************************************/

void CColor::Init (float r, float g, float b, float a)
    {
    myRGBA [0] = r;
    myRGBA [1] = g;
    myRGBA [2] = b;
    myRGBA [3] = a;
    }

/*************************************************************************/

void CColor::Init (float * rgba)
    {
    myRGBA [0] = rgba [0];
    myRGBA [1] = rgba [1];
    myRGBA [2] = rgba [2];
    myRGBA [3] = rgba [3];
    }

/*************************************************************************/

void CColor::Init (CColor color)
    {
    myRGBA [0] = color.RetFloat (0);
    myRGBA [1] = color.RetFloat (1);
    myRGBA [2] = color.RetFloat (2);
    myRGBA [3] = color.RetFloat (3);
    }

/*************************************************************************/

float &CColor::RetFloat (int n)
    {
    return myRGBA [n];
    }

/*************************************************************************/

void CColor::operator = (CColor& that)
    {
    Init (that.RetPointer ());
    }

/*************************************************************************/

void CColor::operator = (float * color)
    {
    Init (color);
    }

/*************************************************************************/

float &CColor::operator () (int n)
    {
    return RetFloat (n);
    }

/*************************************************************************/

float * CColor::RetPointer ()
    {
    return myRGBA;
    }

/*************************************************************************/
    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CPOINT3D //////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

#define NUM_COORDS 3

/*************************************************************************/

CPoint3D::CPoint3D ()
    {
    }

/*************************************************************************/

CPoint3D::CPoint3D (float x, float y, float z)
    {
    myXYZ [0] = x;
    myXYZ [1] = y;
    myXYZ [2] = z;
    }

/*************************************************************************/
```

```
CPoint3D::CPoint3D (float xyz [3])
    {
    myXYZ [0] = xyz [0];
    myXYZ [1] = xyz [1];
    myXYZ [2] = xyz [2];
    }

/*************************************************************************/

CPoint3D::~CPoint3D ()
    {
    }

/*************************************************************************/

void CPoint3D::Init (float x, float y, float z)
    {
    myXYZ [0] = x;
    myXYZ [1] = y;
    myXYZ [2] = z;
    }

/*************************************************************************/

void CPoint3D::Init (float xyz [3])
    {
    myXYZ [0] = xyz [0];
    myXYZ [1] = xyz [1];
    myXYZ [2] = xyz [2];
    }

/*************************************************************************/

void CPoint3D::Init (CPoint3D &point)
    {
    myXYZ [0] = point.RetPointer () [0];
    myXYZ [1] = point.RetPointer () [1];
    myXYZ [2] = point.RetPointer () [2];
    }

/*************************************************************************/

int CPoint3D::ok (int n)
    {
    return n < NUM_COORDS;
    }

/*************************************************************************/

float &CPoint3D::retfloat (int n)
    {
    Assert (ok (n));
    return myXYZ [n];
    }

/*************************************************************************/

float &CPoint3D::operator () (int n)
    {
    return retfloat (n);
    }

/*************************************************************************/

void CPoint3D::AddToFloat (int n, float f)
    {
    Assert (ok (n));
    myXYZ [n] = f;
    }

/*************************************************************************/

void CPoint3D::AddFloats (float f1, float f2, float f3)
    {
    myXYZ [0] = f1;
    myXYZ [1] = f2;
    myXYZ [2] = f3;
    }

/*************************************************************************/

void CPoint3D::AddToFloats (float f1, float f2, float f3)
    {
    myXYZ [0] += f1;
    myXYZ [1] += f2;
    myXYZ [2] += f3;
    }

/*************************************************************************/

void CPoint3D::AddToFloats (CPoint3D &point)
    {
    myXYZ [0] += point.RetPointer () [0];
    myXYZ [1] += point.RetPointer () [1];
    myXYZ [2] += point.RetPointer () [2];
    }
```

```
/*************************************************************************/

void CPoint3D::operator () (int n, float f)
    {
    AddToFloat (n, f);
    }

/*************************************************************************/

float * CPoint3D::RetPointer ()
    {
    return myXYZ;
    }

/*************************************************************************/

bool CPoint3D::operator == (CPoint3D & point)
    {
    return myXYZ [0] == point (0) && myXYZ [1] == point (1) && myXYZ [2] == point (2);
    }

/*************************************************************************/

bool CPoint3D::operator != (CPoint3D & point)
    {
    return myXYZ [0] != point (0) && myXYZ [1] != point (1) && myXYZ [2] != point (2);
    }

/*************************************************************************/

void CPoint3D::operator = (float vert [3])
    {
    Init (vert);
    }

/*************************************************************************/

void CPoint3D::operator = (CPoint3D &that)
    {
    myXYZ [0] = that.myXYZ [0];
    myXYZ [1] = that.myXYZ [1];
    myXYZ [2] = that.myXYZ [2];
    }

/*************************************************************************/

void CPoint3D::operator += (float vert [3])
    {
    myXYZ [0] += vert [0];
    myXYZ [1] += vert [1];
    myXYZ [2] += vert [2];
    }

/*************************************************************************/

void CPoint3D::operator -= (float vert [3])
    {
    myXYZ [0] -= vert [0];
    myXYZ [1] -= vert [1];
    myXYZ [2] -= vert [2];
    }

/*************************************************************************/

void CPoint3D::operator *= (float vert [3])
    {
    myXYZ [0] *= vert [0];
    myXYZ [1] *= vert [1];
    myXYZ [2] *= vert [2];
    }

/*************************************************************************/

void CPoint3D::operator /= (float vert [3])
    {
    myXYZ [0] /= vert [0];
    myXYZ [1] /= vert [1];
    myXYZ [2] /= vert [2];
    }

/*************************************************************************/

void CPoint3D::operator += (CPoint3D &that)
    {
    AddToFloats (that);
    }

/*************************************************************************/

void CPoint3D::operator -= (CPoint3D &that)
    {
    myXYZ [0] -= that.myXYZ [0];
    myXYZ [1] -= that.myXYZ [1];
    myXYZ [2] -= that.myXYZ [2];
    }
```

```
/*************************************************************************/

void CPoint3D::operator *= (CPoint3D &that)
    {
    myXYZ [0] *= that.myXYZ [0];
    myXYZ [1] *= that.myXYZ [1];
    myXYZ [2] *= that.myXYZ [2];
    }

/*************************************************************************/

void CPoint3D::operator /= (CPoint3D &that)
    {
    myXYZ [0] /= that.myXYZ [0];
    myXYZ [1] /= that.myXYZ [1];
    myXYZ [2] /= that.myXYZ [2];
    }

/*************************************************************************/

 void CPoint3D::operator += (float n)
    {
    myXYZ [0] += n;
    myXYZ [1] += n;
    myXYZ [2] += n;
    }

/*************************************************************************/

void CPoint3D::operator -= (float n)
    {
    myXYZ [0] -= n;
    myXYZ [1] -= n;
    myXYZ [2] -= n;
    }

/*************************************************************************/

void CPoint3D::operator *= (float n)
    {
    myXYZ [0] *= n;
    myXYZ [1] *= n;
    myXYZ [2] *= n;
    }

/*************************************************************************/

void CPoint3D::operator /= (float n)
    {
    myXYZ [0] /= n;
    myXYZ [1] /= n;
    myXYZ [2] /= n;
    }

/*************************************************************************/

CPoint3D CPoint3D::operator - (CPoint3D &that)
    {
    return CPoint3D (myXYZ [0] - that (0), myXYZ [1] - that (1), myXYZ [2] - that (2));
    }

/*************************************************************************/

CPoint3D CPoint3D::operator + (CPoint3D &that)
    {
    return CPoint3D (myXYZ [0] + that (0), myXYZ [1] + that (1), myXYZ [2] + that (2));
    }

/*************************************************************************/

CPoint3D CPoint3D::operator * (CPoint3D &that)
    {
    return CPoint3D (myXYZ [0] * that (0), myXYZ [1] * that (1), myXYZ [2] * that (2));
    }

/*************************************************************************/

CPoint3D CPoint3D::operator / (CPoint3D &that)
    {
    return CPoint3D (myXYZ [0] / that (0), myXYZ [1] / that (1), myXYZ [2] / that (2));
    }

/*************************************************************************/

CPoint3D CPoint3D::operator - (int n)
    {
    return CPoint3D (myXYZ [0] - n, myXYZ [1] - n, myXYZ [2] - n);
    }

/*************************************************************************/

CPoint3D CPoint3D::operator + (int n)
    {
    return CPoint3D (myXYZ [0] + n, myXYZ [1] + n, myXYZ [2] + n);
    }

/*************************************************************************/
```

```
CPoint3D CPoint3D::operator * (int n)
    {
    return CPoint3D (myXYZ [0] * n, myXYZ [1] * n, myXYZ [2] * n);
    }

/*************************************************************************/

CPoint3D CPoint3D::operator / (int n)
    {
    return CPoint3D (myXYZ [0] / n, myXYZ [1] / n, myXYZ [2] / n);
    }
/*************************************************************************/

CPoint3D CPoint3D::operator - (float n)
    {
    return CPoint3D (myXYZ [0] - n, myXYZ [1] - n, myXYZ [2] - n);
    }

/*************************************************************************/

CPoint3D CPoint3D::operator + (float n)
    {
    return CPoint3D (myXYZ [0] + n, myXYZ [1] + n, myXYZ [2] + n);
    }

/*************************************************************************/

CPoint3D CPoint3D::operator * (float n)
    {
    return CPoint3D (myXYZ [0] * n, myXYZ [1] * n, myXYZ [2] * n);
    }

/*************************************************************************/

CPoint3D CPoint3D::operator / (float n)
    {
    return CPoint3D (myXYZ [0] / n, myXYZ [1] / n, myXYZ [2] / n);
    }

/*************************************************************************/
// =хёъюы№ъю Їєэъ Ўшщ фы  CPoint3D ъръ фы  тхъЄюЁр

CPoint3D &CPoint3D::CrossProduct (CPoint3D & point)
    {
    return CPoint3D (myXYZ [1] * point (2) - myXYZ [2] * point (1),
                     myXYZ [2] * point (0) - myXYZ [0] * point (2),
                     myXYZ [0] * point (1) - myXYZ [1] * point (0));
    }

/*************************************************************************/

float CPoint3D::Normalize ()
    {
    double sizeSqr = myXYZ [0] * myXYZ [0] + myXYZ [1] * myXYZ [1] + myXYZ [2] * myXYZ [2];
    if (sizeSqr > 0.0000001F)
        {
        float scale = 1.0F / (float) sqrt (sizeSqr);
        myXYZ [0] *= scale; myXYZ [1] *= scale; myXYZ [2] *= scale;
        return scale;
        }
    else return 0.0F;
    }

/*************************************************************************/

float CPoint3D::DotProduct (CPoint3D &point)
    {
    return myXYZ [0] * point (0) + myXYZ [1] * point (1) + myXYZ [2] * point (2);
    }

/*************************************************************************/

float CPoint3D::RetLength ()
    {
    return (float) sqrt (myXYZ [0] * myXYZ [0] + myXYZ [1] * myXYZ [1] + myXYZ [2] * myXYZ [2]);
    }

/*************************************************************************/

/*************************************************************************/
/*************************************************************************/
/********************* CLASS CMATRIX4X4 **********************************/
/*************************************************************************/
/*************************************************************************/

/*************************************************************************/

CMatrix4x4::CMatrix4x4 ()
    {
    MakeSimpleMatrix ();
    }

/*************************************************************************/

CMatrix4x4::~CMatrix4x4 ()
    {
```

```
    }

/*************************************************************************/

void CMatrix4x4::Init (CMatrix4x4& m)
    {
    for (int y = 0; y < MATRIXSIZEY; y ++)
        for (int x = 0; x < MATRIXSIZEX; x ++)
            RetFloat (x, y) = m.RetFloat (x, y);
    }

/*************************************************************************/

void CMatrix4x4::operator = (CMatrix4x4& m)
    {
    Init (m);
    }

/*************************************************************************/

float &CMatrix4x4::RetFloat (int x, int y)
    {
    AssertT (x < MATRIXSIZEX && y < MATRIXSIZEY, "X or Y is out of range");
    return myMatrix [MATRIXSIZEX * y + x];
    }

/*************************************************************************/

float &CMatrix4x4::operator ()(int x, int y)
    {
    return RetFloat (x, y);
    }

/*************************************************************************/

float * CMatrix4x4::RetPointer ()
    {
    AssertT (!!myMatrix, "Out of memory");
    return myMatrix;
    }

/*************************************************************************/

void CMatrix4x4::MakeSimpleMatrix ()
    {
    RetFloat (0, 0) = 1.0F; RetFloat (1, 0) = 0.0F; RetFloat (2, 0) = 0.0F; RetFloat (3, 0) = 0.0F;
    RetFloat (0, 1) = 0.0F; RetFloat (1, 1) = 1.0F; RetFloat (2, 1) = 0.0F; RetFloat (3, 1) = 0.0F;
    RetFloat (0, 2) = 0.0F; RetFloat (1, 2) = 0.0F; RetFloat (2, 2) = 1.0F; RetFloat (3, 2) = 0.0F;
    RetFloat (0, 3) = 0.0F; RetFloat (1, 3) = 0.0F; RetFloat (2, 3) = 0.0F; RetFloat (3, 3) = 1.0F;
    }

/*************************************************************************/

int CMatrix4x4::IfSimpleMatrix ()
    {
    if (RetFloat (0, 0) == 1.0F && RetFloat (1, 0) == 0.0F && RetFloat (2, 0) == 0.0F && RetFloat (3, 0)
== 0.0F &&
        RetFloat (0, 1) == 0.0F && RetFloat (1, 1) == 1.0F && RetFloat (2, 1) == 0.0F && RetFloat (3, 1)
== 0.0F &&
        RetFloat (0, 2) == 0.0F && RetFloat (1, 2) == 0.0F && RetFloat (2, 2) == 1.0F && RetFloat (3, 2)
== 0.0F &&
        RetFloat (0, 3) == 0.0F && RetFloat (1, 3) == 0.0F && RetFloat (2, 3) == 0.0F && RetFloat (3, 3)
== 1.0F)
        return 1;
    return 0;
    }

/*************************************************************************/

void CMatrix4x4::MakeRotXMatrix (float a)
    {
    float c = cos (a / 57.295779F), s = sin (a / 57.295779F);
    RetFloat (0, 0) = 1.0F; RetFloat (1, 0) = 0.0F; RetFloat (2, 0) = 0.0F; RetFloat (3, 0) = 0.0F;
    RetFloat (0, 1) = 0.0F; RetFloat (1, 1) = c;    RetFloat (2, 1) = -s;   RetFloat (3, 1) = 0.0F;
    RetFloat (0, 2) = 0.0F; RetFloat (1, 2) = s;    RetFloat (2, 2) = c;    RetFloat (3, 2) = 0.0F;
    RetFloat (0, 3) = 0.0F; RetFloat (1, 3) = 0.0F; RetFloat (2, 3) = 0.0F; RetFloat (3, 3) = 1.0F;
    }

/*************************************************************************/

void CMatrix4x4::MakeRotYMatrix (float a)
    {
    float c = cos (a / 57.295779F), s = sin (a / 57.295779F);
    RetFloat (0, 0) = c;    RetFloat (1, 0) = 0.0F; RetFloat (2, 0) = s;    RetFloat (3, 0) = 0.0F;
    RetFloat (0, 1) = 0.0F; RetFloat (1, 1) = 1.0F; RetFloat (2, 1) = 0.0F; RetFloat (3, 1) = 0.0F;
    RetFloat (0, 2) = -s;   RetFloat (1, 2) = 0.0F; RetFloat (2, 2) = c;    RetFloat (3, 2) = 0.0F;
    RetFloat (0, 3) = 0.0F; RetFloat (1, 3) = 0.0F; RetFloat (2, 3) = 0.0F; RetFloat (3, 3) = 1.0F;
    }

/*************************************************************************/

void CMatrix4x4::MakeRotZMatrix (float a)
    {
    float c = cos (a / 57.295779F), s = sin (a / 57.295779F);
    RetFloat (0, 0) = c;    RetFloat (1, 0) = -s;   RetFloat (2, 0) = 0.0F; RetFloat (3, 0) = 0.0F;
    RetFloat (0, 1) = s;    RetFloat (1, 1) = c;    RetFloat (2, 1) = 0.0F; RetFloat (3, 1) = 0.0F;
    RetFloat (0, 2) = 0.0F; RetFloat (1, 2) = 0.0F; RetFloat (2, 2) = 1.0F; RetFloat (3, 2) = 0.0F;
```

```
    RetFloat (0, 3) = 0.0F; RetFloat (1, 3) = 0.0F; RetFloat (2, 3) = 0.0F; RetFloat (3, 3) = 1.0F;
    }

/*****************************************************************************/

float CMatrix4x4::DotProduct (float a1, float a2, float a3, float a4,
                              float b1, float b2, float b3, float b4)
    {
    return (a1 * b1) + (a2 * b2) + (a3 * b3) + (a4 * b4);
    }

/*****************************************************************************/

void CMatrix4x4::DotProduct (CMatrix4x4 &matrix)
    {
    static CMatrix4x4 result;
    for (int x = 0; x < MATRIXSIZEX; x ++)
        {
        for (int y = 0; y < MATRIXSIZEY; y ++)
            {
            result (x, y) = DotProduct (RetFloat (x, 0), RetFloat (x, 1),
                                        RetFloat (x, 2), RetFloat (x, 3),
                                        matrix (0, y), matrix (1, y),
                                        matrix (2, y), matrix (3, y));
            }
        }
    Init (result);
    }

/*****************************************************************************/

CPoint3D &CMatrix4x4::DotProduct (CPoint3D &point)
    {
    static CPoint3D result;
    float vec [4] = {point (0), point (1), point (2), 1.0F};
    float res [4];
    for (int y = 0; y < MATRIXSIZEY; y ++)
        {
        res [y] = DotProduct (vec [0], vec [1], vec [2], vec [3],
                              RetFloat (0, y), RetFloat (1, y), RetFloat (2, y), RetFloat (3, y));
        }
    result (0) = res [0]; result (1) = res [1]; result (2) = res [2];
    return result;
    }

/*****************************************************************************/

void CMatrix4x4::Transpone ()
    {
    CMatrix4x4 result;
    for (int row = 0; row < MATRIXSIZEY; row ++)
        {
        for (int column = 0; column < MATRIXSIZEX; column ++)
            {
            result (column, row) = RetFloat (row, column);
            }
        }
    Init (result);
    }

/*****************************************************************************/

#define GLMAXLIGHTS 8

    ////////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CLIGHTSOURCE //////////////////////////////
 ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

/*****************************************************************************/

CLightSource::CLightSource ()
    {
    }

/*****************************************************************************/

CLightSource::CLightSource (int lighttype, int lightnum) :
    myLightType (lighttype),
    myLightNum (lightnum)
    {
    }

/*****************************************************************************/

CLightSource::~CLightSource ()
    {
    }

/*****************************************************************************/

void CLightSource::operator = (CLightSource& that)
    {
    myColor = that.myColor;
    myPosition = that.myPosition;
    myDirection = that.myDirection;
```

```
    myLightNum = that.myLightNum;
    myLightType = that.myLightType;
    }

/*************************************************************************/

void CLightSource::Init (int lighttype, int lightnum)
    {
    myLightType = lighttype;
    myLightNum = lightnum;
    }

/*************************************************************************/

int CLightSource::ok (int type)
    {
    return myLightType == type;
    }

/*************************************************************************/

void CLightSource::SetColor (float r, float g, float b, float a)
    {
    myColor.Init (r, g, b, a);
    }

/*************************************************************************/

void CLightSource::SetColor (float * rgba)
    {
    myColor.Init (rgba);
    }

/*************************************************************************/

void CLightSource::SetColor (CColor &color)
    {
    myColor = color;
    }

/*************************************************************************/

void CLightSource::SetPosition (float x, float y, float z, float w)
    {
    myPosition.Init (x, y, z, w);
    }

/*************************************************************************/

void CLightSource::SetPosition (float * xyzw)
    {
    myPosition = xyzw;
    }

/*************************************************************************/

void CLightSource::SetPosition (CColor &pos)
    {
    myPosition = pos;
    }

/*************************************************************************/

void CLightSource::SetDirection (float x, float y, float z)
    {
    myDirection.Init (x, y, z);
    }

/*************************************************************************/

void CLightSource::SetDirection (float * xyz)
    {
    myDirection = (xyz);
    }

/*************************************************************************/

void CLightSource::SetDirection (CPoint3D &dir)
    {
    myDirection = dir;
    }

/*************************************************************************/

void CLightSource::SetCutoff (float cutoff)
    {
    myParam = cutoff;
    }

/*************************************************************************/

    ////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////
  ////////////////////////  CLASS CICON  /////////////////////////////
 ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
```

```
/*************************************************************************/

CIcon::CIcon ()
    {
    }

/*************************************************************************/

CIcon::CIcon (int x, int y)
    {
    Init (x, y);
    }

/*************************************************************************/

CIcon::~CIcon ()
    {
    }

/*************************************************************************/

void CIcon::Init (int x, int y)
    {
    mySizeX = x;
    mySizeY = y;
    myByteSizeX = mySizeX / 8;
    if (myByteSizeX * 8 < mySizeX) myByteSizeX ++;
    myBitmap = new char [myByteSizeX * mySizeY];
    }

/*************************************************************************/

void CIcon::SetID (int n)
    {
    myID = n;
    }

/*************************************************************************/

int CIcon::RetID ()
    {
    return myID;
    }

/*************************************************************************/

int CIcon::RetSizeX ()
    {
    return mySizeX;
    }

/*************************************************************************/

int CIcon::RetSizeY ()
    {
    return mySizeY;
    }

/*************************************************************************/

int CIcon::ok (int x, int y)
    {
    return (x < myByteSizeX && y < mySizeY);
    }

/*************************************************************************/

char &CIcon::RetByte (int x, int y)
    {
    AssertT (ok (x, y), "Array is out of range");
    return myBitmap [y * myByteSizeX + x];
    }

/*************************************************************************/

char &CIcon::operator () (int x, int y)
    {
    return RetByte (x, y);
    }

/*************************************************************************/

char * CIcon::RetPointer ()
    {
    return myBitmap;
    }

/*************************************************************************/

void CIcon::operator = (CIcon &that)
    {
    Init (that.mySizeX, that.mySizeY);
    for (int n = 0; n < myByteSizeX * mySizeY; n ++)
        myBitmap [n] = that.myBitmap [n];
    }

/*************************************************************************/
```

```
    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CPICTURE //////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

/*************************************************************************/

CPicture::CPicture ()
    {
    }

/*************************************************************************/

CPicture::CPicture (int x, int y, float px, float py, float pz)
    {
    Init (x, y, px, py, pz);
    }

/*************************************************************************/

CPicture::~CPicture ()
    {
    }

/*************************************************************************/

void CPicture::Init (int x, int y, float px, float py, float pz)
    {
    mySizeX = x;
    mySizeY = y;
    myByteSizeX = mySizeX / 8;
    if (myByteSizeX * 8 < mySizeX) myByteSizeX ++;
    myBitmap = new char [myByteSizeX * mySizeY];
    myPosition.Init (px, py, pz);
    }

/*************************************************************************/

void CPicture::SetPos (float x, float y, float z)
    {
    myPosition.Init (x, y, z);
    }

/*************************************************************************/

void CPicture::SetColor (float r, float g, float b, float a)
    {
    myColor.Init (r, g, b, a);
    }

/*************************************************************************/

CPoint3D &CPicture::RetPos ()
    {
    return myPosition;
    }

/*************************************************************************/

CColor &CPicture::RetColor ()
    {
    return myColor;
    }

/*************************************************************************/

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CBILLBOARD ////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

/*************************************************************************/

CBillBoard::CBillBoard ()
    {
    }

/*************************************************************************/

CBillBoard::~CBillBoard ()
    {
    }

/*************************************************************************/

void CBillBoard::Init (float x, float y, float z, float r, float g, float b, float a,
                       float sizex, float sizey, int id)
    {
    myCenter.Init (x, y, z);
    myColor.Init (r, g, b, a);
    mySizeX = sizex;
    mySizeY = sizey;
    myTexID = id;
    }
```

```
/*************************************************************************/
void CBillBoard::SetTexID (int id)
    {
    myTexID = id;
    }
/*************************************************************************/
int CBillBoard::RetTexID ()
    {
    return myTexID;
    }
/*************************************************************************/
void CBillBoard::SetCenter (float x, float y, float z)
    {
    myCenter.Init (x, y, z);
    }
/*************************************************************************/
void CBillBoard::SetCenter (float * xyz)
    {
    myCenter.Init (xyz);
    }
/*************************************************************************/
void CBillBoard::SetCenter (CPoint3D & center)
    {
    myCenter = center;
    }
/*************************************************************************/
CPoint3D & CBillBoard::RetCenter ()
    {
    return myCenter;
    }
/*************************************************************************/
void CBillBoard::SetColor (float r, float g, float b, float a)
    {
    myColor.Init (r, g, b, a);
    }
/*************************************************************************/
CColor & CBillBoard::RetColor ()
    {
    return myColor;
    }
/*************************************************************************/
void CBillBoard::SetFreedomVector (float x, float y, float z)
    {
    myFreedomVector.Init (x, y, z);
    }
/*************************************************************************/
CPoint3D & CBillBoard::RetFreedomVector ()
    {
    return myFreedomVector;
    }
/*************************************************************************/
void CBillBoard::SetSize (float x, float y)
    {
    mySizeX = x;
    mySizeY = y;
    }
/*************************************************************************/
float CBillBoard::RetSizeX ()
    {
    return mySizeX;
    }
/*************************************************************************/
float CBillBoard::RetSizeY ()
    {
    return mySizeY;
    }
/*************************************************************************/
float CBillBoard::RetMinX ()
    {
```

```
    return -(mySizeX / 2.0F);
    }
/*************************************************************************/
float CBillBoard::RetMaxX ()
    {
    return mySizeX / 2.0F;
    }
/*************************************************************************/
float CBillBoard::RetMinY ()
    {
    return -(mySizeY / 2.0F);
    }
/*************************************************************************/
float CBillBoard::RetMaxY ()
    {
    return mySizeY / 2.0F;
    }
/*************************************************************************/
    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CCOLORMATRIX //////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/*************************************************************************/
CColorMatrix::CColorMatrix ()
    {
    myPosition.Init (0.0F, 0.0F, 0.0F);
    myMatrix = NULL;
    }
/*************************************************************************/
CColorMatrix::CColorMatrix (int x, int y)
    {
    Init (x, y);
    }
/*************************************************************************/
CColorMatrix::CColorMatrix (CColorMatrix &cm)
    {
    Init (cm);
    }
/*************************************************************************/
CColorMatrix::~CColorMatrix ()
    {
    delete [] myMatrix;
    }
/*************************************************************************/
int CColorMatrix::RetX ()
    {
    return mySizeX;
    }
/*************************************************************************/
int CColorMatrix::RetY ()
    {
    return mySizeY;
    }
/*************************************************************************/
int CColorMatrix::RetID ()
    {
    return myID;
    }
/*************************************************************************/
void CColorMatrix::SetPosition (float x, float y, float z)
    {
    myPosition.Init (x, y, z);
    }
/*************************************************************************/
CPoint3D &CColorMatrix::RetPosition ()
    {
    return myPosition;
    }
/*************************************************************************/
```

```
int CColorMatrix::ok (int x, int y)
    {
    return x < mySizeX && y < mySizeY;
    }

/*****************************************************************************/

int CColorMatrix::ok (int x, int y, int z)
    {
    return x < mySizeX && y < mySizeY && z < COLOR_FLOATS;
    }

/*****************************************************************************/

void CColorMatrix::Init (int x, int y)
    {
    myPosition.Init (0.0F, 0.0F, 0.0F);
    mySizeX = x;
    mySizeY = y;
    New ();
    }

/*****************************************************************************/

void CColorMatrix::Init (CColorMatrix &cm)
    {
    myPosition.Init (0.0F, 0.0F, 0.0F);
    Init (cm.mySizeX, cm.mySizeY);
    Copy (cm.myMatrix);
    }

/*****************************************************************************/

void CColorMatrix::New ()
    {
    myMatrix = new float [mySizeX * mySizeY * COLOR_FLOATS];
    AssertT (!!myMatrix, "Out of memory!");
    }

/*****************************************************************************/

void CColorMatrix::Delete ()
    {
    delete [] myMatrix;
    }

/*****************************************************************************/

float &CColorMatrix::RetElement (int x, int y, int z)
    {
    AssertT (ok (x, y, z), "X, Y or Z is out of range!");
    AssertT (!!myMatrix, "Out of memory!");
    return myMatrix [(y * mySizeX + x) * COLOR_FLOATS + z];
    }

/*****************************************************************************/

float &CColorMatrix::operator () (int x, int y, int z)
    {
    return RetElement (x, y, z);
    }

/*****************************************************************************/

float * CColorMatrix::RetPointer (int n)
    {
    AssertT (!!myMatrix, "Out of memory!");
    return myMatrix + n;
    }

/*****************************************************************************/

float * CColorMatrix::operator () (int n)
    {
    return RetPointer (n);
    }

/*****************************************************************************/

void CColorMatrix::Copy (float * matrix)
    {
    AssertT (!!myMatrix, "Out of memory!");
    myMatrix = (float *) memcpy (myMatrix, matrix, mySizeX * mySizeY * COLOR_FLOATS * sizeof (float));
    }

/*****************************************************************************/

void CColorMatrix::AddTexel (int x, int y, CColor &color)
    {
    AssertT (ok (x, y), "X and Y are out of range");
    RetElement (x, y, 0) = color.RetPointer () [0];
    RetElement (x, y, 1) = color.RetPointer () [1];
    RetElement (x, y, 2) = color.RetPointer () [2];
    RetElement (x, y, 3) = color.RetPointer () [3];
    }
```

```
/*****************************************************************************/

void CColorMatrix::AddTexel (int x, int y, float r, float g, float b, float a)
    {
    AssertT (ok (x, y), "X or Y is out of range");
    RetElement (x, y, 0) = r;
    RetElement (x, y, 1) = g;
    RetElement (x, y, 2) = b;
    RetElement (x, y, 3) = a;
    }

/*****************************************************************************/

void CColorMatrix::SetNewID (int id)
    {
    myID = id;
    }

/*****************************************************************************/

void CColorMatrix::operator = (float * matrix)
    {
    Copy (matrix);
    }

/*****************************************************************************/

void CColorMatrix::operator = (CColorMatrix &that)
    {
    Delete ();
    Init (that.mySizeX, that.mySizeY);
    Copy (that.myMatrix);
    }

/*****************************************************************************/

    ////////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CTEXCOORD /////////////////////////////////
 ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

/*****************************************************************************/

CTexCoord::CTexCoord ()
    {
    }

/*****************************************************************************/

CTexCoord::CTexCoord (float x, float y) :
    myX (x),
    myY (y)
    {
    }

/*****************************************************************************/

CTexCoord::~CTexCoord ()
    {
    }

/*****************************************************************************/

void CTexCoord::Init (float x, float y)
    {
    myX = x;
    myY = y;
    }

/*****************************************************************************/

    ////////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////////
  ////////////////////// CLASS CGRAPHMATERIAL ////////////////////////////////
 ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

/*****************************************************************************/

CGraphMaterial::CGraphMaterial ()
    {
    }

/*****************************************************************************/

CGraphMaterial::CGraphMaterial (float d, float s, float e) :
    myDiffuse (d),
    myShininess (s),
    myEmission (e)
    {
    }

/*****************************************************************************/

CGraphMaterial::~CGraphMaterial ()
    {
```

```
    }
/*****************************************************************************/
void CGraphMaterial::SetColor (float r, float g, float b, float a)
    {
    myRGBA [0] = r;
    myRGBA [1] = g;
    myRGBA [2] = b;
    myRGBA [3] = a;
    }
/*****************************************************************************/
float * CGraphMaterial::RetColor ()
    {
    return myRGBA;
    }
/*****************************************************************************/
void CGraphMaterial::SetColorSource (int source)
    {
    myColorSource = source;
    }
/*****************************************************************************/
int CGraphMaterial::RetColorSource ()
    {
    return myColorSource;
    }
/*****************************************************************************/
void CGraphMaterial::Init (float d, float s, float e)
    {
    myDiffuse = d;
    myShininess = s;
    myEmission = e;
    }
/*****************************************************************************/
void CGraphMaterial::SetDiffuseCoeff (float d)
    {
    myDiffuse = d;
    }
/*****************************************************************************/
void CGraphMaterial::SetShininess (float s)
    {
    myShininess = s;
    }
/*****************************************************************************/
void CGraphMaterial::SetEmission (float e)
    {
    myEmission = e;
    }
/*****************************************************************************/
float CGraphMaterial::RetDiffuseCoeff ()
    {
    return myDiffuse;
    }
/*****************************************************************************/
float CGraphMaterial::RetShininess ()
    {
    return myShininess;
    }
/*****************************************************************************/
float CGraphMaterial::RetEmission ()
    {
    return myEmission;
    }
/*****************************************************************************/
void CGraphMaterial::operator = (CGraphMaterial & mat)
    {
    myDiffuse = mat.myDiffuse;
    myShininess = mat.myShininess;
    myEmission = mat.myEmission;
    myRGBA [0] = mat.myRGBA [0];
    myRGBA [1] = mat.myRGBA [1];
    myRGBA [2] = mat.myRGBA [2];
    myRGBA [3] = mat.myRGBA [3];
    myColorSource = mat.myColorSource;
    }
```

```
/*****************************************************************/

     ////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////////////
   ////////////////////////// CLASS CVERTEX ////////////////////////////
  ////////////////////////////////////////////////////////////////////
 ////////////////////////////////////////////////////////////////////

/*****************************************************************/

CVertex::CVertex ()
    {
    }

/*****************************************************************/

CVertex::~CVertex ()
    {
    }

/*****************************************************************/

void CVertex::Init (float x, float y, float z, float n1, float n2, float n3,
                    float r, float g, float b, float a, float cx, float cy, float w)
    {
    myPoint.Init (x, y, z);
    myNormal.Init (n1, n2, n3);
    myNormal.Normalize ();
    myColor.Init (r, g, b, a);
    myCoord.Init (cx, cy);
    myWeight = w;
    }

/*****************************************************************/

void CVertex::SetPoint (float x, float y, float z)
    {
    myPoint.Init (x, y, z);
    }

/*****************************************************************/

void CVertex::SetNormal (float x, float y, float z)
    {
    myNormal.Init (x, y, z);
    myNormal.Normalize ();
    }

/*****************************************************************/

void CVertex::SetColor (float r, float g, float b, float a)
    {
    myColor.Init (r, g, b, a);
    }

/*****************************************************************/

void CVertex::SetTexCoord (float x, float y)
    {
    myCoord.Init (x, y);
    }

/*****************************************************************/

void CVertex::SetWeight (float w)
    {
    myWeight = w;
    }

/*****************************************************************/

CPoint3D &CVertex::RetPoint ()
    {
    return myPoint;
    }

/*****************************************************************/

CPoint3D &CVertex::RetNormal ()
    {
    return myNormal;
    }

/*****************************************************************/

CColor &CVertex::RetColor ()
    {
    return myColor;
    }

/*****************************************************************/

CTexCoord &CVertex::RetTexCoord ()
    {
    return myCoord;
    }
```

November 14, 2008 12:17 AM -- 280

```
/*****************************************************************/

float CVertex::RetWeight ()
    {
    return myWeight;
    }

/*****************************************************************/

void CVertex::operator = (CVertex &that)
    {
    myPoint = that.myPoint;
    myNormal = that.myNormal;
    myColor = that.myColor;
    myCoord = that.myCoord;
    myWeight = that.myWeight;
    }

/*****************************************************************/

#define MAXVERTEXES 3
#define MAXCOORDS   4

     ////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////////////
   ////////////////////// CLASS CPOLYGON ////////////////////////////
  ////////////////////////////////////////////////////////////////////
 ////////////////////////////////////////////////////////////////////

/*****************************************************************/
/*                                                               */
/*    ╫шёыю тхЁ°шэ т юфэюь яюышуюэх юуЁрэшўхэю ЄЁхь , Є.ъ. Direct  */
/* 3D ЁрсюЄрхЄ Єюы№ъю ё ЄЁхєуюы№эшърьш, ш фы  ЁрёёўхЄр юётх•хэш    */
/* яю ьюфхыш ├єЁю эхюсїюфшь√ уЁрэш ё ЄЁхь  тхЁ°шэрьш.              */
/*                                                               */
/*****************************************************************/

/*********************************************************************/

CPolygon::CPolygon () :
    myCurrVert (0),
    myVertexNum (MAXVERTEXES)
    {
    }

/*********************************************************************/

CPolygon::CPolygon (CVertex vertarray [3]) :
    myCurrVert (0),
    myVertexNum (MAXVERTEXES)
    {
    AddArray (vertarray);
    }

/*********************************************************************/

CPolygon::~CPolygon ()
    {
    }

/*********************************************************************/

void CPolygon::Init ()
    {
    myCurrVert = 0;
    myVertexNum = MAXVERTEXES;
    }

/*********************************************************************/

void CPolygon::Init (CVertex vertarray [3])
    {
    myCurrVert = 0;
    myVertexNum = MAXVERTEXES;
    AddArray (vertarray);
    }

/*********************************************************************/

int CPolygon::ok (int n)
    {
    return n < myCurrVert;
    }

/*********************************************************************/

int CPolygon::ok ()
    {
    return myCurrVert <= MAXVERTEXES;
    }

/*********************************************************************/

void CPolygon::operator = (CPolygon &that)
    {
    AddArray (that.myVertexArray);
```

```
    myVertexNum = MAXVERTEXES;
    myCurrVert = that.myCurrVert;
    myTexPointer = that.myTexPointer;
    }

/*************************************************************************/

void CPolygon::AddArray (CVertex vertarray [3])
    {
    myVertexArray [0] = vertarray [0];
    myVertexArray [1] = vertarray [1];
    myVertexArray [2] = vertarray [2];
    }

/*************************************************************************/

void CPolygon::SetVertex (float x, float y, float z, float n1, float n2, float n3,
                          float r, float g, float b, float a, float cx, float cy, float w)
    {
    AssertT (ok (), "Can't add new vertex to polygon");
    myVertexArray [myCurrVert].Init (x, y, z, n1, n2, n3, r, g, b, a, cx, cy, w);
    myCurrVert ++;
    }

/*************************************************************************/

CVertex &CPolygon::retvert (int n)
    {
    AssertT (ok (n), "Can't return vertex");
    return myVertexArray [n];
    }

/*************************************************************************/

CVertex &CPolygon::operator () (int n)
    {
    return retvert (n);
    }

/*************************************************************************/

void CPolygon::SetTexPointer (int n)
    {
    myTexPointer = n;
    }

/*************************************************************************/

int CPolygon::RetTexPointer ()
    {
    return myTexPointer;
    }

/*************************************************************************/

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CBONE /////////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

/*************************************************************************/

CBone::CBone ()
    {
    myTransform.MakeSimpleMatrix ();
    }

/*************************************************************************/

CBone::~CBone ()
    {
    }

/*************************************************************************/

void CBone::SetCenter (float x, float y, float z)
    {
    myCenter.Init (x, y, z);
    }

/*************************************************************************/

void CBone::Move (float x, float y, float z)
    {
    myCenter.AddToFloats (x, y, z);
    }

/*************************************************************************/

CPoint3D &CBone::RetCenter ()
    {
    return myCenter;
    }

/*************************************************************************/
```

```
CMatrix4x4 &CBone::RetTransformMatrix ()
    {
    return myTransform;
    }

/*************************************************************************/

void CBone::operator = (CBone &that)
    {
    myCenter = that.myCenter;
    myTransform = that.myTransform;
    }

/*************************************************************************/

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CMESH /////////////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

/*************************************************************************/

CMesh::CMesh () :
    myMaxChildren (0),
    myCurChild (0)
    {
    myPolyArray = NULL;
    myLights = NULL;
    myParent = NULL;
    myChildren = NULL;
    }

/*************************************************************************/

CMesh::CMesh (int maxobj, int maxlight) :
    myMaxPolys (maxobj),
    myMaxLights (maxlight),
    myCurPoly (0),
    myCurLight (0),
    myMaxChildren (0),
    myCurChild (0)
    {
    myPolyArray = new CPolygon * [maxobj];
    myLights = new CLightSource * [maxlight];
    myParent = NULL;
    myChildren = NULL;
    RetTransformMatrix ().MakeSimpleMatrix ();
    }

/*************************************************************************/

CMesh::~CMesh ()
    {
    delete [] myPolyArray;
    delete [] myLights;
    delete [] myChildren;
    }

/*************************************************************************/

void CMesh::Init (int maxobj, int maxlight)
    {
    myMaxPolys = maxobj;
    myMaxLights = maxlight;
    myCurPoly = 0;
    myCurLight = 0;
    myMaxChildren = 0;
    myCurChild = 0;
    myPolyArray = new CPolygon * [maxobj];
    myLights = new CLightSource * [maxlight];
    myParent = NULL;
    myChildren = NULL;
    RetTransformMatrix ().MakeSimpleMatrix ();
    }

/*************************************************************************/

void CMesh::New ()
    {
    myPolyArray = new CPolygon * [myMaxPolys];
    myLights = new CLightSource * [myMaxLights];
    myParent = NULL;
    myChildren = NULL;
    }

/*************************************************************************/

void CMesh::Delete ()
    {
    delete [] myPolyArray;
    delete [] myLights;
    delete [] myChildren;
    }

/*************************************************************************/
```

```
int CMesh::RetMaxPolys ()
    {
    return myMaxPolys;
    }
/*************************************************************************/
int CMesh::RetMaxLights ()
    {
    return myMaxLights;
    }
/*************************************************************************/
int CMesh::RetCurPolys ()
    {
    return myCurPoly;
    }
/*************************************************************************/
int CMesh::RetCurLights ()
    {
    return myCurLight;
    }
/*************************************************************************/
int CMesh::RetID ()
    {
    return myID;
    }
/*************************************************************************/
void CMesh::SetID (int id)
    {
    myID = id;
    }
/*************************************************************************/
int CMesh::polyOk ()
    {
    return myCurPoly < myMaxPolys;
    }
/*************************************************************************/
int CMesh::polyOk (int n)
    {
    return n < myCurPoly;
    }
/*************************************************************************/
int CMesh::lightOk ()
    {
    return myCurLight < myMaxLights;
    }
/*************************************************************************/
int CMesh::lightOk (int n)
    {
    return n < myCurLight;
    }
/*************************************************************************/
void CMesh::SetMaterial (float d, float s, float e)
    {
    myMaterial.Init (d, s, e);
    }
/*************************************************************************/
void CMesh::SetMaterial (CGraphMaterial &material)
    {
    myMaterial = material;
    }
/*************************************************************************/
CGraphMaterial& CMesh::RetMaterial ()
    {
    return myMaterial;
    }
/*************************************************************************/
void CMesh::SetCenter (float x, float y, float z)
    {
    myBone.SetCenter (x, y, z);
    }
/*************************************************************************/
```

```
void CMesh::Move (float x, float y, float z)
    {
    myBone.Move (x, y, z);
    }

/*************************************************************************/

CPoint3D &CMesh::RetCenter ()
    {
    return myBone.RetCenter ();
    }

/*************************************************************************/

CMatrix4x4 &CMesh::RetTransformMatrix ()
    {
    return myBone.RetTransformMatrix ();
    }

/*************************************************************************/

CBone &CMesh::RetBone ()
    {
    return myBone;
    }

/*************************************************************************/

void CMesh::Rotate (CMatrix4x4 &m)
    {
    myBone.RetTransformMatrix ().DotProduct (m);
    for (int n = 0; n < myCurChild; n ++)
        {
        myChildren [n] -> Rotate (m);
        }
    }

/*************************************************************************/

void CMesh::Rotate (float x, float y, float z)
    {
    CMatrix4x4 mx, my, mz;
    mx.MakeRotZMatrix (x);
    my.MakeRotZMatrix (y);
    mz.MakeRotZMatrix (z);
    mx.DotProduct (my);
    mx.DotProduct (mz);
    Rotate (mx);
    }

/*************************************************************************/

void CMesh::SetAngle (float x, float y, float z)
    {
    CMatrix4x4 mx, my, mz;
    mx.MakeRotZMatrix (x);
    my.MakeRotZMatrix (y);
    mz.MakeRotZMatrix (z);
    mx.DotProduct (my);
    mx.DotProduct (mz);
    myBone.RetTransformMatrix () = mx;
    }

/*************************************************************************/

void CMesh::SetMatrix (CMatrix4x4 &m)
    {
    myBone.RetTransformMatrix () = m;
    }

/*************************************************************************/

void CMesh::Clone (CMesh * that)
    {
    myBone = that -> myBone;
    myMaterial = that -> myMaterial;
    Init (that -> myCurPoly, that -> myCurLight);
    for (int n = 0; n < that -> myCurPoly; n ++)
        {
        AddPoly (new CPolygon);
        *(myPolyArray [n]) = *(that -> myPolyArray [n]);
        }
    for (n = 0; n < that -> myCurLight; n ++)
        {
        AddLight (new CLightSource);
        *(myLights [n]) = *(that -> myLights [n]);
        }
    }

/*************************************************************************/

void CMesh::Clone (CMesh * that, CPolygon * polys)
    {
    myBone = that -> myBone;
    myMaterial = that -> myMaterial;
    Init (that -> myCurPoly, that -> myCurLight);
```

```
    for (int n = 0; n < that -> myCurPoly; n ++)
        {
        AddPoly (&polys [n]);
        myPolyArray [n] -> Init ();
        *(myPolyArray [n]) = *(that -> myPolyArray [n]);
        }
    for (n = 0; n < that -> myCurLight; n ++)
        {
        AddLight (new CLightSource);
        *(myLights [n]) = *(that -> myLights [n]);
        }
    }

/*************************************************************************/

void CMesh::operator = (CMesh &that)
    {
    myBone = that.myBone;
    myMaterial = that.myMaterial;
    Init (that.myCurPoly, that.myCurLight);
    for (int n = 0; n < that.myCurPoly; n ++)
        {
        AddPoly (that.myPolyArray [n]);
        }
    for (n = 0; n < that.myCurLight; n ++)
        {
        AddLight (that.myLights [n]);
        }
    }

/*************************************************************************/

/*************************************************************************/
// ╧ыышуюэ√

void CMesh::AddPoly (CPolygon * poly)
    {
    Assert (polyOk ());
    myPolyArray [myCurPoly] = poly;
    myCurPoly ++;
    }

/*************************************************************************/

void CMesh::AddPolys (CPolygon * polys, int n)
    {
    for (int i = 0; i < n; i ++)
        {
        AddPoly (&polys [i]);
        }
    }

/*************************************************************************/

CPolygon * CMesh::RetPoly (int n)
    {
    Assert (polyOk (n));
    return myPolyArray [n];
    }

/*************************************************************************/

/*************************************************************************/
// ╬ётх•хэшх

void CMesh::AddLight (CLightSource * light)
    {
    Assert (lightOk ());
    myLights [myCurLight] = light;
    myCurLight ++;
    }

/*************************************************************************/

void CMesh::AddLights (CLightSource * lights, int nlights)
    {
    for (int i = 0; i < nlights; i ++)
        {
        AddLight (&lights [i]);
        }
    }

/*************************************************************************/

CLightSource * CMesh::RetLight (int n)
    {
    Assert (lightOk (n));
    return myLights [n];
    }

/*************************************************************************/

/*************************************************************************/
// ─ы  рэшьрЎшш

void CMesh::SetParent (CMesh * parent)
    {
```

November 14, 2008 12:17 AM -- 286

```
    myParent = parent;
    }

/*****************************************************************************/

void CMesh::InitChildren (int num)
    {
    myMaxChildren = num;
    myCurChild = 0;
    myChildren = new CMesh * [num];
    }

/*****************************************************************************/

void CMesh::SetChild (CMesh * child)
    {
    AssertT (myChildren != NULL, "No children initialized");
    AssertT (myCurChild < myMaxChildren, "Can't access unexisting child");
    myChildren [myCurChild] = child;
    child -> SetParent (this);
    myCurChild ++;
    }

/*****************************************************************************/

CMesh * CMesh::RetParent ()
    {
    return myParent;
    }

/*****************************************************************************/

CMesh * CMesh::RetChild (int num)
    {
    AssertT (myChildren != NULL, "No children initialized");
    AssertT (num < myCurChild, "Can't access unexisting child");
    return myChildren [num];
    }

/*****************************************************************************/

int CMesh::RetCurChildren ()
    {
    return myCurChild;
    }

/*****************************************************************************/

int CMesh::RetMaxChildren ()
    {
    return myMaxChildren;
    }

/*****************************************************************************/

/*****************************************************************************/

    ////////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////////
  ////////////////////// CLASS CSTANDARDPRIMITIVE ////////////////////////////
 ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

/*****************************************************************************/

CStandardPrimitive::CStandardPrimitive ()
    {
    myParams = NULL;
    }

/*****************************************************************************/

CStandardPrimitive::CStandardPrimitive (int type, float * params) :
    myType (type)
    {
    switch (type)
        {
        case STDPR_SPHERE:
            InitSphere (params);
            break;
        case STDPR_CUBE:
            InitCube (params);
            break;
        case STDPR_CONE:
            InitCone (params);
            break;
        case STDPR_TORUS:
            InitTorus (params);
            break;
        case STDPR_TEAPOT:
            InitTeapot (params);
            break;
        default:
            break;
        }
    }
```

```
/*************************************************************************/

CStandardPrimitive::~CStandardPrimitive ()
    {
    delete [] myParams;
    }

/*************************************************************************/

void CStandardPrimitive::Init (int type, float * params)
    {
    myType = type;
    switch (type)
        {
        case STDPR_SPHERE:
            InitSphere (params);
            break;
        case STDPR_CUBE:
            InitCube (params);
            break;
        case STDPR_CONE:
            InitCone (params);
            break;
        case STDPR_TORUS:
            InitTorus (params);
            break;
        case STDPR_TEAPOT:
            InitTeapot (params);
            break;
        default:
            break;
        }
    }

/*************************************************************************/

void CStandardPrimitive::InitSphere (float * params)
    {
    myParamNum = 3;
    myParams = new float [myParamNum];
    myParams [0] = params [0];
    myParams [1] = params [1];
    myParams [2] = params [2];
    }

/*************************************************************************/

void CStandardPrimitive::InitCube (float * params)
    {
    myParamNum = 1;
    myParams = new float [myParamNum];
    myParams [0] = params [0];
    }

/*************************************************************************/

void CStandardPrimitive::InitCone (float * params)
    {
    myParamNum = 4;
    myParams = new float [myParamNum];
    myParams [0] = params [0];
    myParams [1] = params [1];
    myParams [2] = params [2];
    myParams [3] = params [3];
    }

/*************************************************************************/

void CStandardPrimitive::InitTorus (float * params)
    {
    myParamNum = 4;
    myParams = new float [myParamNum];
    myParams [0] = params [0];
    myParams [1] = params [1];
    myParams [2] = params [2];
    myParams [3] = params [3];
    }

/*************************************************************************/

void CStandardPrimitive::InitTeapot (float * params)
    {
    myParamNum = 1;
    myParams = new float [myParamNum];
    myParams [0] = params [0];
    }

/*************************************************************************/

int CStandardPrimitive::RetType ()
    {
    return myType;
    }

/*************************************************************************/

float CStandardPrimitive::RetParam (int nparam)
```

```
    {
    if (nparam >= 0 && nparam < myParamNum)
        return myParams [nparam];
    return -1.0F;
    }

/*************************************************************************/

void CStandardPrimitive::SetCenter (float x, float y, float z)
    {
    myCenter.Init (x, y, z);
    }

/*************************************************************************/

CPoint3D &CStandardPrimitive::RetCenter ()
    {
    return myCenter;
    }

/*************************************************************************/

void CStandardPrimitive::SetColor (float r, float g, float b, float a)
    {
    myColor.Init (r, g, b, a);
    }

/*************************************************************************/

CColor &CStandardPrimitive::RetColor ()
    {
    return myColor;
    }

/*************************************************************************/

void CStandardPrimitive::Move (float x, float y, float z)
    {
    myCenter.AddToFloats (x, y, z);
    }

/*************************************************************************/
```

Файл CPPVM\CPPVMCore.cpp

```
#include "CPPVMCore.h"

//-----------------------------------------------------------------------------------------------------------

StatFunc::StatFunc ( pStatFunc pFunc) :
    myPFunc ( pFunc)
    { }
StatFunc::~StatFunc()
    { }

SParam StatFunc::operator() ( const CParams& params)
    { return myPFunc ( params); }

//-----------------------------------------------------------------------------------------------------------

ObjFuncPtr::ObjFuncPtr ( std::auto_ptr<ObjFunc> pFunc) :
    myPFunc ( pFunc)
    { }
ObjFuncPtr::ObjFuncPtr ( const ObjFuncPtr& right) :
    myPFunc ( right.getPFunc())
    { }

ObjFuncPtr& ObjFuncPtr::operator= ( const ObjFuncPtr& right)
{
    myPFunc = right.getPFunc();
    return *this;
}

std::auto_ptr<ObjFunc>& ObjFuncPtr::getPFunc() const
    { return *const_cast<std::auto_ptr<ObjFunc>*> ( &myPFunc); }

SParam ObjFuncPtr::operator() ( const CParams& params)
    { Assert ( ObjFuncPtr::myPFunc.get() );
    return myPFunc->operator() ( params); }

//-----------------------------------------------------------------------------------------------------------

CPPFunc::CPPFunc ( int className, int funcName, const pObjFunc& func) :
    mySelInL ( 0),
    myClassName ( className),
    myName ( funcName),
    myFunc ( func)
    { }

CPPFunc::~CPPFunc()
```

```
    { }

CPPFunc& CPPFunc::addParamType ( SParamTypeInfo t)
    {
    myTypes << t;
    return *this;
    }

bool CPPFunc::operator== ( const CPPFunc& func) const
    { return myClassName == func.myClassName && myName == func.myName; }

//---------------------------------------------------------------------------------------------------

CPPStaticFunc::CPPStaticFunc (int className, int funcName, const ObjFuncPtr& func) :
    myClassName (className),
    myName (funcName),
    myFunc (func)
    { }

CPPStaticFunc::~CPPStaticFunc()
    { }

CPPStaticFunc& CPPStaticFunc::addParamType ( SParamTypeInfo t)
    {
    myTypes << t;
    return *this;
    }

bool CPPStaticFunc::operator== (const CPPStaticFunc& func) const
    { return myClassName == func.myClassName && myName == func.myName; }

//---------------------------------------------------------------------------------------------------

CPPClass::CPPClass ( int className) :
    myClassName ( className),
    myStaticFuncs (1),
    myFuncs (1)
    { myFuncs.add (CPPFunc());
      myStaticFuncs.add (CPPStaticFunc()); }

CPPClass::~CPPClass()
    { }

int CPPClass::getFuncSelector (int funcName)
    {
    for (int i = 1; i < myFuncs.rNumber(); i++)
        if (myFuncs[i].myName == funcName && myFuncs[i].myClassName == myClassName/*check*/) return i;
    return 0;
    }

int CPPClass::getStaticFuncSelector (int funcName)
    {
    for (int i = 1; i < myStaticFuncs.rNumber(); i++)
        if (myStaticFuncs[i].myName == funcName && myStaticFuncs[i].myClassName == myClassName/*check*/)
return i;
    return 0;
    }

bool CPPClass::operator== (const CPPClass& cl) const
    { return myClassName == cl.myClassName; }

//---------------------------------------------------------------------------------------------------

SParam CCPPVM::staticCall ( int className, int classSel, int funcName, int funcSel, const CParams& params)
    {
    return ( rStatFuncPtr ( className, classSel, funcName, funcSel)) ( params);
    }

SParam CCPPVM::call ( CObj* obj, int className, int classSel, int funcName, int funcSel, const CParams&
params)
    { Assert (obj);
    return (obj->*( rFuncPtr (className, classSel, funcName, funcSel))) (params);
    }

IMPLEMENT_IUNKNOWN (CCPPVM, ICPPVM, IID_ICPPVM);

CCPPVM::CCPPVM() :
    myClasses (1)
    { myClasses.add ( NULL); }

CPPClass* CCPPVM::getClassPtr ( int className)
    {
    return myClasses [ getClassSelector ( className)];
    }

int CCPPVM::getClassSelector (int className)
    {
    for (int i = 1; i < myClasses.rNumber(); i++)
        if (myClasses[i]->myClassName == className) return i;
    return 0;
    }

int CCPPVM::getFuncSelector (int className, int funcName)
    {
    int classSel = getClassSelector (className);
    if (!classSel) return 0;
    return myClasses[classSel]->getFuncSelector(funcName);
```

```
    }

int CCPPVM::getStaticFuncSelector (int className, int funcName)
    {
    int classSel = getClassSelector (className);
    if (!classSel) return 0;
    return myClasses[classSel]->getStaticFuncSelector(funcName);
    }

pObjFunc CCPPVM::rFuncPtr (Func* func)
    { return rFuncPtr (func->myClassName, func->myClassSelectorInVM, func->myName, func->mySelectorInVM);
}

pObjFunc CCPPVM::rFuncPtr (int className, int classSel, int funcName, int funcSel)
    {
    if (!classSel) classSel = getClassSelector (className);
    if (classSel <= 0 || classSel >= myClasses.rNumber()) return 0; //!!! assert

    if (!funcSel) funcSel = myClasses[classSel]->getFuncSelector (funcName);
    if (funcSel <= 0 || funcSel >= myClasses[classSel]->myFuncs.rNumber()) return 0; //!!! assert

    return myClasses[classSel]->myFuncs[funcSel].myFunc;
    }

ObjFuncPtr& CCPPVM::rStatFuncPtr (StaticFunc* func)
    { return rStatFuncPtr (func->myClassName, func->myClassSelectorInVM, func->myName, func-
>mySelectorInVM); }

ObjFuncPtr& CCPPVM::rStatFuncPtr (int className, int classSel, int funcName, int funcSel)
    {
    if (!classSel) classSel = getClassSelector (className);
    Assert (0 < classSel && classSel < myClasses.rNumber());

    if (!funcSel) funcSel = myClasses[classSel]->getStaticFuncSelector (funcName);
    Assert (0 < funcSel && funcSel < myClasses[classSel]->myStaticFuncs.rNumber());

    return myClasses[classSel]->myStaticFuncs[funcSel].myFunc;
    }
SClassRef CCPPVM::registerClass ( const char* className, const char* varName)
    { return registerClass ( CPPVM::Linker->registerName ( className), CPPVM::Linker->registerName (
varName)); }

SClassRef CCPPVM::registerClass ( int className, int varName)
    {
    if ( getClassSelector ( className))
        return CPPVM::Linker->registerClass ( className, varName);

    if ( int i = CPPVM::Linker->getClassRef (className))
        {
        myClasses.add ( new CPPClass ( className)); //will be deleted automatically
        return SClassRef ( i);
        }

    myClasses.add ( new CPPClass (className)); //the same

    return myClasses.back()->myClassRef = CPPVM::Linker->registerClass ( className, varName);
    }

SClassRef CCPPVM::registerBaseClass ( const char* className, const char* varName)
    { return registerBaseClass ( CPPVM::Linker->registerName ( className), CPPVM::Linker->registerName (
varName)); }
SClassRef CCPPVM::registerBaseClass ( int className, int baseClassName)
    {
    SClass* cl = CPPVM::Linker->getClassPtr ( className);
    SClassRef base  = CPPVM::Linker->getClassRef ( baseClassName);
    Assert ( cl && "can't find class for addin' a base class to it"); //!!! redo
    Assert ( (SClass*)base && "can't find class to add it as a base class"); //!!! redo
    return & cl->registerParent ( base);
    }

StaticFunc* CCPPVM::registerObjectFuncAsStatic ( SClassRef cl, const char* funcName,
std::auto_ptr<ObjFunc> pFunc, const CParamsTypes& types)
{ return registerObjectFuncAsStatic ( cl, CPPVM::Linker->registerName ( funcName), pFunc, types); }
StaticFunc* CCPPVM::registerObjectFuncAsStatic ( SClassRef cl, int funcName, std::auto_ptr<ObjFunc> pFunc,
const CParamsTypes& types)
{
int classSel = getClassSelector (cl->myClassName);
    Assert (0 < classSel && classSel < myClasses.rNumber()); //!!! redo

    for (int i = 1; i < myClasses[classSel]->myStaticFuncs.rNumber(); i++)
        Assert (myClasses[classSel]->myStaticFuncs[i].myName != funcName) //!!! redo

    myClasses[classSel]->myStaticFuncs.add ( CPPStaticFunc (cl->myClassName, funcName, ObjFuncPtr (
pFunc)));

    StaticFunc* fPtr =  cl->registerStaticFunc ( StaticFunc ( this, funcName, cl->myClassName, types));

    if (fPtr) return fPtr;

    myClasses[classSel]->myStaticFuncs.del ( myClasses[classSel]->myStaticFuncs.rNumber() - 1);

    return 0;
}

StaticFunc* CCPPVM::registerStaticFunc ( SClassRef cl, const char* funcName, pStatFunc func, const
CParamsTypes& types)
    { return registerStaticFunc ( cl, CPPVM::Linker->registerName ( funcName), func, types); }
```

```
StaticFunc* CCPPVM::registerStaticFunc ( SClassRef cl, int funcName, pStatFunc func, const CParamsTypes&
types)
    {
        return registerObjectFuncAsStatic ( cl, funcName, std::auto_ptr <ObjFunc> ( new StatFunc ( func)),
types);
    }

Func* CCPPVM::registerFunc ( SClassRef cl, const char* funcName, pObjFunc func, const CParamsTypes& types)
    { return registerFunc ( cl, CPPVM::Linker->registerName ( funcName), func, types); }
Func* CCPPVM::registerFunc ( SClassRef cl, int funcName, pObjFunc func, const CParamsTypes& types)
    {
    int sel = getClassSelector ((CPPVM::Linker->getClasses()[cl])->myClassName);
    Assert (0 < sel && sel < myClasses.rNumber()); //!!! redo

    for (int i = 1; i < myClasses[sel]->myFuncs.rNumber(); i++)
        Assert (myClasses[sel]->myFuncs[i].myName != funcName); //!!! redo

    myClasses[sel]->myFuncs.add ( CPPFunc ( myClasses[sel]->myClassName, funcName, func));

    Func* fPtr = CPPVM::Linker->getClasses()[cl]->registerFunc ( Func ( this, funcName, cl->myClassName,
types));

    if (fPtr)
        { myClasses[sel]->myFuncs[ myClasses[sel]->myFuncs.rNumber()-1].mySelInL = cl->getFuncSelector (
funcName);
        return fPtr; }

    myClasses[sel]->myFuncs.del ( myClasses[sel]->myFuncs.rNumber() - 1);

    return NULL;
    }

//-------------------------------------------------------------------------------------------------------

SClassRef RegisterBaseClass (SClassRef* cl, const CString& baseClassName)
    { return CPPVM::CPPVM->registerBaseClass ( (*cl)->myClassName, CPPVM::Linker->registerName (
baseClassName)); }

SClassRef RegisterBaseClass (SClassRef* cl, int baseClassName)
    { return CPPVM::CPPVM->registerBaseClass ( (*cl)->myClassName, baseClassName); }

SClassRef RegisterClass (SClassRef* cl, const CString& clName, const CString& varName)
    { return *cl = CPPVM::CPPVM->registerClass ( CPPVM::Linker->registerName ( clName), CPPVM::Linker-
>registerName ( varName)); }

SClassRef RegisterClass (SClassRef* cl, int clName, int varName)
    { return *cl = CPPVM::CPPVM->registerClass ( clName, varName); }

Func* RegisterFunc (SClassRef* cl, const CString& fname, pObjFunc func, const CParamsTypes& types)
    { return CPPVM::CPPVM->registerFunc ( *cl, CPPVM::Linker->registerName (fname), func, types); }

StaticFunc* RegisterStaticFunc (SClassRef* cl, const CString& fname, pStatFunc func, const CParamsTypes&
types)
    { return CPPVM::CPPVM->registerStaticFunc ( *cl, CPPVM::Linker->registerName (fname), func, types); }

//-------------------------------------------------------------------------------------------------------
```

Файл CPPVM\cppvm.cpp

```
#include "CPPVM.h"
```

Файл CPPVM\module.cpp

```
#define __PROGRAMMER    "Se"
#define __MODULE_NAME   "cppvm.dll initialization"

//#include <exeption.h>
#include "System.h"
#include "CPPVMCore.h"
#include "..\Linker\linker.h"

    #define DLL_NAME "CPPVM"
    #define DLL_CLASS CCPPVM
    #include "..\Common\Utilities\Module.c"


DLL_INIT(loader)
{
    Assert (loader);
    CPPVM::Loader = loader;

    LinkerDataTypesLibInit ( loader);

    CPPVM::Linker = (ILinker*)CPPVM::Loader->GetInterface (IID_ILinker);
    Assert (CPPVM::Linker);
    CPPVM::CPPVM = (ICPPVM*) CPPVM::Loader->GetInterface (IID_ICPPVM);
    Assert (CPPVM::CPPVM);
```

```
    loader->GetInterface (IID_IConsole, (void**)&CPPVM::Console);
    Assert (CPPVM::Console);
    CPPVM::Console->printf ("CPPVM initialized.\n");
}

DLL_ONLOAD(loader)
{
    IConsole* con = NULL;
    loader->GetInterface (IID_IConsole, (void**)&con);
    Assert (con);
    con->printf ("CPPVM: CPP Virtual Mashine by Se version 1.0\n");
}
```

Файл CPPVM\system.cpp

```
#define __PROGRAMMER   "Se"
#define __MODULE_NAME  "System"

#include "System.h"

//#include <Exeption.h>      // use ExeptionEngine

ILoader* CPPVM::Loader = 0;
ILinker* CPPVM::Linker = 0;
ICPPVM*  CPPVM::CPPVM = 0;
IConsole* CPPVM::Console = 0;
```

Файл ExeptionEngine\Exeption.cpp

```
// Exeption.cpp : Defines the entry point for the DLL application.
//

#include "stdafx.h"
#include "Exeption.h"

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

Файл ExeptionEngine\ExeptionCore.cpp

```
//ExeptionCore.cpp - implementation of Exeption and CMsg

#include "stdafx.h" //use precompiled headers

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Exeption Engine Core"

#include "Exeption.h"
#include "ExeptionUtils.h"
#include <stdarg.h>
#include <stdio.h>
#include <io.h>
#include <fcntl.h>
#include <mmsystem.h> //winmm.lib

//Constants:
const char* Exeption::TypesStr[] =        {"Unknown", "General", "Bad arguments",
                                           "Runtime", "Assertion", "Memory Leak", "Memory damage",
                                           "Program resources Leak", "Program Fail",
                                           "System resources Leak","System fail"};
const char* CMsg::CodeStr[] = {"Simple", "Hint", "Trace", "Warning", "Error", "Pre-Exeption", "Unknown"};

//class Exeption
unsigned int StartTime = timeGetTime ();

Exeption::Exeption(const char* programmer, const char* module,
                   const char* file, const char* function, int line):
           m_programmer((char*)programmer), m_module((char*)module),
           m_file((char*)file), m_line(line),
```

```
            m_text ((char*)"No description"), m_textWasOrdered (false), m_creationTime (timeGetTime()-
StartTime), m_textShouldBeDeleted(true)
{m_function = function? const_cast <char*>(function): "Unknown";}

Exeption::Exeption(int, const char* programmer, const char* module,
                   const char* file, const char* function, int line,
                   int type, const char* text):
            m_programmer((char*)programmer), m_module((char*)module),
            m_file((char*)file), m_line(line), m_type(type), m_text ((char*)text),
            m_textWasOrdered (false), m_creationTime (timeGetTime()-StartTime),
m_textShouldBeDeleted(true)
{
    m_function = function? const_cast <char*> (function): "Unknown";
    RegisterExeption (this);
}

Exeption::Exeption (Exeption& from):
            m_programmer(from.m_programmer), m_module(from.m_module),
            m_file(from.m_file), m_function (from.m_function), m_line(from.m_line), m_type(from.m_type),
m_text (from.m_text),
            m_textWasOrdered (from.m_textWasOrdered), m_creationTime (from.m_creationTime),
m_textShouldBeDeleted(true)
{}

Exeption& Exeption::operator ()()
{
    this->m_text = NULL;
    this->m_type = general;
    RegisterExeption (this);
    m_textShouldBeDeleted = false;
    return *this;
}
Exeption& Exeption::operator ()(int code)
{
    this->m_text = NULL;
    this->m_type = code;
    RegisterExeption (this);
    m_textShouldBeDeleted = false;
    return *this;
}

Exeption& Exeption::operator ()(const char* text, int code)
{
    m_text = (char*)text;
    m_type = code;
    RegisterExeption (this);
    m_textShouldBeDeleted = false;
    return *this;
}

Exeption& Exeption::operator ()(int code, const char* format, ...)
{
    m_type = code;
    m_textWasOrdered = true;
    m_text = new char[512];
    m_text[511] = 0;
    va_list list;
    va_start (list, format);
    vsprintf (m_text, format, list);
    va_end (list);
    RegisterExeption (this);
    m_textShouldBeDeleted = false;
    return *this;
}


void Exeption::Handle(){m_handled = true;}
Exeption::~Exeption()
{
    if (!m_handled)
        {
        UnhandledExeption (this);
        }

    //if (m_textWasOrdered && m_textShouldBeDeleted) ;//delete m_text;
}

int Exeption::PrintText (char* str)
{
    if (str) return sprintf (str, "(%s module by %s, file: '%s', procedure: '%s' line %d, generated at
time %d)\n\t%s Exeption: %s",
        m_module, m_programmer, m_file, m_function, m_line, m_creationTime, TypesStr[m_type], m_text);
    else return -1;
}

int Exeption::PrintText (FILE* file)
{
    if (file)
        {
        setmode (fileno (file), O_TEXT);
        fprintf (file, "(%s module by %s, file: '%s', procedure: '%s' line %d, generated at time %d)\n\t
%s Exeption: %s",
        m_module, m_programmer, m_file, m_function, m_line, m_creationTime, TypesStr[m_type], m_text);
        return true;
        }
    else return -1;
}
```

```
//CMsg
CMsg::CMsg (const char* programmer, const char* module,
        const char* file, const char* function, const int line, const int type,
        const char* text):
    m_programmer(programmer), m_module(module), m_file(file), m_line(line), m_type (type),
        m_text((char*)text), m_creationTime (timeGetTime ()-StartTime), m_textWasOrdered(false)
{
m_function = function? const_cast <char*> (function): "Unknown";
noPos = false;
RegisterMessage (this);
}

CMsg::CMsg (int pos, const char* programmer, const char* module,
        const char* file, const char* function, const int line):
    m_programmer(programmer), m_module(module), m_file(file), m_line(line), m_type (hint),
        m_text((char*)"No text"), m_creationTime (timeGetTime ()-StartTime), m_textWasOrdered(false)
{
noPos = pos? true: false;
m_function = function? const_cast <char*> (function): "Unknown";
}

CMsg::~CMsg (){if (m_textWasOrdered) delete[] m_text;}

void CMsg::PrintText (char* str)
{
    if (!str) return;

    if (noPos)
        sprintf (str, "%s", m_text);
    else sprintf (str, "(module %s by %s, file '%s' procedure '%s' line %d, time %d) %s message: %s\n",
        m_module, m_programmer, m_file, m_function, m_line, m_creationTime, CMsg::CodeStr [m_type],
m_text);
}

void CMsg::PrintText (FILE* file)
{
    if (!file) return;

    if (noPos)
        fprintf (file, "%s", m_text);
    else fprintf (file, "(module %s by %s, file '%s' procedure '%s' line %d, time %d) %s message: %s\n",
        m_module, m_programmer, m_file, m_function, m_line, m_creationTime, CMsg::CodeStr [m_type],
m_text);
}

CMsg::operator ()(int code)
{
    m_type = code;
    RegisterMessage (this);
    return true;
}

/*CMsg::operator ()(const char* text, int code)
{
    m_text = (char*) text;
    m_type = code;
    RegisterMessage (this);
    return true;
}
*/
CMsg::operator ()(int code, const char* format, ...)
{
    m_type = code;
    m_textWasOrdered = true;
    m_text = new char[256];
    va_list list;
    va_start (list, format);
    vsprintf (m_text, format, list);
    va_end(list);
    RegisterMessage (this);
    return true;
}


CMsg::operator ()(const char* format, ...)
{
    m_type = trace;
    m_textWasOrdered = true;
    m_text = new char[256];
    va_list list;
    va_start (list, format);
    vsprintf (m_text, format, list);
    va_end(list);
    RegisterMessage (this);
    return true;
}
```

Файл ExeptionEngine\ExeptionUtils.cpp

```
//Exeption Utilities implementation

#include "stdafx.h"
#include "Exeption.h"
#include "ExeptionUtils.h"

#include <stdio.h>
#include <io.h>
#include <fcntl.h>
#include <mmsystem.h> //wimmm.lib
#include <time.h>

#define DEFAULT_DUMP_FILENAME "ExeptionDump.log"

const int VersionH = 1, VersionL = 5;

unsigned int    CountMessages          = 0,
                CountExeptions         = 0,
                CountExeptionUnhandled = 0,
                CountExeptionCatching  = 0,
                CountExeptionHandling  = 0;

FILE* DumpFile = stdout;
bool fileWasOpened = false;
char fileName[256];
PMessageHandlerFunction           MessageHandler;
PExeptionHandlingFunction         ExeptionHandler;
PExeptionHandlingFunction         UnhandledExeptionHandler;

unsigned int StartupClock = 0;
time_t StartupTime;
time_t ShutdownTime;

//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------

void UnhandledExeption (Exeption* ex)
{
    CountExeptionUnhandled ++;
    char str[512];
    ex->PrintText (str);
    fprintf (DumpFile, "\n(time %d) Unhandled exeption: \n\t%s\n\tLived %d clocs ago\n", timeGetTime(),
str, (timeGetTime() - (ex->m_creationTime)));
    fflush (DumpFile);
}
//-------------------------------------------------------------------------------------

void RegisterExeption  (Exeption* ex)
{
    CountExeptions++;
    fprintf (DumpFile, "\nExeption thrown: ");
    ex->PrintText (DumpFile);
    fprintf (DumpFile, "\n");
    fflush (DumpFile);
}
//-------------------------------------------------------------------------------------

void RegisterMessage       (CMsg* message)
{
    CountMessages++;
    if (!message->noPos) fprintf (DumpFile, "\nMessage: ");
    else fprintf (DumpFile, "\n");
    message->PrintText (DumpFile);
    fflush (DumpFile);
}

//-------------------------------------------------------------------------------------
void EXEPTION_API RegisterExeptionCatching (Exeption* ex, const char* programmer,
                              const char* module, const char* file, const char* proc, int line,
                              const char* text)
{
    CountExeptionCatching ++;
    fprintf (DumpFile, "\nThe Exeption '");
    ex->PrintText(DumpFile);
    fprintf (DumpFile, "\n\tWas Catched by %s in module '%s', file '%s', procedure '%s'line %d, comments:
'%s'\n",
        programmer, module, file, proc, line, text);
    fflush (DumpFile);
}
//-------------------------------------------------------------------------------------

void EXEPTION_API RegisterExeptionHandling (Exeption* ex, const char* programmer,
                              const char* module, const char* file, const char* proc, int line,
                              const char* text)
{
    CountExeptionHandling ++;
    ex->Handle ();
    fprintf (DumpFile, "\nThe Exeption '");
    ex->PrintText(DumpFile);
    fprintf (DumpFile, "\n\tWas Handled by %s in module '%s', file '%s', procedure '%s' line %d, comments:
'%s'\n",
```

```
        programmer, module, file, proc, line, text);
    fflush (DumpFile);
}
//-------------------------------------------------------------------------------

int  EXEPTION_API SetDumpFile (FILE* file, bool close)
{
    if (!file)
        return false;
    fileWasOpened = close;
    DumpFile = file;
    return true;
}

//-------------------------------------------------------------------------------
void EXEPTION_API BeginDump ()
{
    fprintf (DumpFile, "Exeption Engine version %d.%d by Vasfed\n", VersionH, VersionL);
    fprintf (DumpFile, "Beginning dynamical synchronous dump of exeptions and messages\n");
    fprintf (DumpFile, "Computer's clock is %d     (winmm timeGetTime())\n", timeGetTime());
    fprintf (DumpFile, "System startup time %s\n", ctime (&StartupTime));
    fprintf (DumpFile, "-------------------------------------------------------------\n"
                       "DUMP:\n\n");
    fflush (DumpFile);
}

//-------------------------------------------------------------------------------
void EXEPTION_API EndDump()
{
    fprintf (DumpFile, "\n\n\n\n\nEnd of dump\n");
    fprintf (DumpFile, "-------------------------------------------------------------\n");
    fprintf (DumpFile, "Computer's clock is %d\n", timeGetTime());
    unsigned int t = timeGetTime ();
    fprintf (DumpFile, "System shutdown time is %s", ctime (&ShutdownTime));
    int d = 0, h = 0, m = 0, s = 0;

    s = (t - StartupClock) / 1000;

    if (s >= 60)
        {
        m = s / 60;
        s = s % 60;
        }

    if (m >= 60)
        {
        h = m / 60;
        m = m % 60;
        }
    if (h >= 24)
        {
        d = h / 24;
        d = d % 24;
        }

    fprintf (DumpFile, "System was up for about %d days %d hours %d minutes %d seconds (if more, there's
error)\n", d, h, m, s);
    fprintf (DumpFile, "\nStatistics: \n");
    fprintf (DumpFile, "there'as %d total events:\n", CountMessages + CountExeptions +
CountExeptionCatching + CountExeptionHandling + CountExeptionUnhandled );
    fprintf (DumpFile, "\t%d\texeptions were thrown\n", CountExeptions);
    fprintf (DumpFile, "\t%d\tmessages  were send\n", CountMessages);
    fprintf (DumpFile, "\t%d\texeptions were catched\n", CountExeptionCatching);
    fprintf (DumpFile, "\t%d\texeptions were handled\n", CountExeptionHandling);
    fprintf (DumpFile, "\t%d\texeptions were NOT handled\n", CountExeptionUnhandled);

    fprintf (DumpFile, "\nNote: If programmer was not using all the macroces, dump may be wrong\n");

    fflush (DumpFile);
    if (fileWasOpened) fclose (DumpFile);
    DumpFile =  stdout;
}
//-------------------------------------------------------------------------------

void EXEPTION_API CreateDumpFile (const char* filename)
{
    FILE* file = fopen (filename, "wt");
    if (file)
        {
        DumpFile = file;
        fileWasOpened = true;
        strcpy (fileName, filename);
        }
    else {/*error*/}
    BeginDump();
}
//-------------------------------------------------------------------------------

void EXEPTION_API SetMessageHandler (PMessageHandlerFunction func)
{
    fprintf (DumpFile, "EXCEPTIONENGINE: Call to unimplemented function \"%s\"\n", "SetMessageHandler");
}
//-------------------------------------------------------------------------------

void EXEPTION_API SetExeptionHandler (PExeptionHandlingFunction func)
{
    fprintf (DumpFile, "EXCEPTIONENGINE: Call to unimplemented function \"%s\"\n", "SetExeptionHandler");
```

```
}

//---------------------------------------------------------------------------

void EXEPTION_API SetUnhandledExeptionHandler (PExeptionHandlingFunction func)
{
    fprintf (DumpFile, "EXCEPTIONENGINE: Call to unimplemented function \"%s\"\n",
"SetUnhandledExeptionHandler");
}

//---------------------------------------------------------------------------

void EXEPTION_API GetDumpFileName (char* str)
{
    strcpy (str, fileName);
}
//---------------------------------------------------------------------------
void EXEPTION_API PromptExeptionEngine (void)
{
    printf ("Exception Engine by Vasfed version %d.%d. \n\tDumpfile: '%s'\n", VersionH, VersionL,
fileName);
}

//---------------------------------------------------------------------------

class CClassWithDestructorAndConstructor
{
    public:
    CClassWithDestructorAndConstructor ()
        {
        time (&StartupTime);
        StartupClock = timeGetTime ();
        CreateDumpFile (DEFAULT_DUMP_FILENAME);
        }

    ~CClassWithDestructorAndConstructor ()
        {
        time (&ShutdownTime);
        EndDump ();
        }
} SomeDummyObject;
//---------------------------------------------------------------------------
```

Файл LAgents\AgentMessage.cpp

```
// AgentMessage.cpp - implementation of AgentMessage.h, of module LAgents
// created 12.12.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "LAgents"

#include "StdAfx.h"
#include "System.h"
#include "AgentMessage.h"

#include <Exeption.h>

SClassRef CAgentMessage::sClass = SClassRef ( 0);

CAgentMessage::CAgentMessage()
    {
        myClass = sClass;
        isLinkedClass = true;
    }

bool CAgentMessage::selfReg()
    {
        sClass = LA::CPPVM->registerClass ("AgentMessage", "AgentMessage");

        LA::CPPVM->registerFunc ( sClass, "rX", (pObjFunc) rX, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rY", (pObjFunc) rY, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rZ", (pObjFunc) rZ, CParamsTypes());

        LA::CPPVM->registerFunc ( sClass, "rSender", (pObjFunc) rSender, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rReceiver", (pObjFunc) rReceiver, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rType", (pObjFunc) rType, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rCode", (pObjFunc) rCode, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rAiName", (pObjFunc) rAiName, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "setX:", (pObjFunc) setX, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "setY:", (pObjFunc) setY, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "setZ:", (pObjFunc) setZ, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "setCode:", (pObjFunc) setCode, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "setType:", (pObjFunc) setType, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "setReceiver:", (pObjFunc) setReceiver, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "setSender:", (pObjFunc) setSender, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "Post", (pObjFunc) Post, CParamsTypes());
        return 1;
    }

SParam CAgentMessage::rSender (const CParams&)
    {return this->sender;}
```

```
SParam CAgentMessage::rReceiver (const CParams&)
    {return this->receiver;}

SParam CAgentMessage::rType (const CParams&)
    {return this->type;}

SParam CAgentMessage::rCode (const CParams&)
    {return this->code;}

SParam CAgentMessage::rY (const CParams&)
    {return this->y;}

SParam CAgentMessage::rX (const CParams&)
    {return this->x;}

SParam CAgentMessage::rZ (const CParams&)
     {return this->z;}

SParam CAgentMessage::rAiName (const CParams&)
     {return this->ainame;}

SParam CAgentMessage::setX         (const CParams& par)
     {
     x = par[0].rFloat();
     return this;
     }

SParam CAgentMessage::setY         (const CParams& par)
     {
     y = par[0].rFloat();
     return this;
     }

SParam CAgentMessage::setZ         (const CParams& par)
     {
     z = par[0].rFloat();
     return this;
     }

SParam CAgentMessage::setCode      (const CParams& par)
     {
     code = par[0].rInt();
     return this;
     }

SParam CAgentMessage::setType      (const CParams& par)
     {
     type = par[0].rInt();
     return this;
     }

SParam CAgentMessage::setReceiver  (const CParams& par)
     {
     receiver = par[0].rInt();
     return this;
     }

SParam CAgentMessage::setSender    (const CParams& par)
     {
     if (par[0].type == par[0].ePObj)
        {
        sender = ((par[0].rObj())->call ("rIndex")).rInt();
        }
    else
        {
        sender = par[0].rInt();
        }
     return this;
     }

SParam CAgentMessage::Post     (const CParams&)
    {
    LA::Server->SendMessage (this);
    return SParam ();
    }
```

Файл LAgents\LAgentSimulator.cpp

```
// LAgentSimulator.cpp - implementation of LAgentSimulator.h, of module LAgents
// created 10.12.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "LAgents"

#include "StdAfx.h"
#include "System.h"
#include "LAgentSimulator.h"

#include <Exeption.h>
```

```
SClassRef CAgentData::sClass = SClassRef ( 0);

CAgentData::CAgentData()
    {
        myClass = sClass;
    }

bool CAgentData::selfReg()
    {
        sClass = LA::CPPVM->registerClass ("LinkedAgent", "LinkedAgent");

        LA::CPPVM->registerFunc ( sClass, "rX", (pObjFunc) CAgentData::rX, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rY", (pObjFunc) rY, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rZ", (pObjFunc) rZ, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rIndex", (pObjFunc) rIndex, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rSHBC", (pObjFunc) rSHBP, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "setSHBP", (pObjFunc) setSHBP, (TYPES SParam::eBool));

        return 1;
    }

SParam CAgentData::rX(const CParams&)
    { return m_x; }
SParam CAgentData::rY(const CParams&)
    { return m_y; }
SParam CAgentData::rZ(const CParams&)
    { return m_z; }

SParam CAgentData::rIndex(const CParams&)
    {
        ;
        return m_index;
    }
SParam CAgentData::rSHBP(const CParams&)
    { return m_shouldBeProcessed;}
SParam CAgentData::setSHBP(const CParams& par)
    { return m_shouldBeProcessed = par[0];}

//-----------------------------------------------------------------------------------------
CLinkedAgent::CLinkedAgent()
    {
        myAgent = LA::Linker->getGlobalScope()("Pimp") ("new")();
        myAgent ("InitData:") (PARAMS CObjRef ( this));
    }
CLinkedAgent::CLinkedAgent ( const char* str)
    {
        myAgent = LA::Linker->getGlobalScope()(str) ("new")();
        myAgent ("InitData:") (PARAMS CObjRef ( this));
    }

int CLinkedAgent::Go()
    {
        return (int) myAgent ("Go") ();
    }

int CLinkedAgent::Ok()
    {
        return (int) myAgent ("Ok") ();
    }

void CLinkedAgent::Init()
    {
        myAgent ("Init") ();
    }

int CLinkedAgent::MessageProc(AgentMessage* msg)
    {
        return (int) myAgent ("MessageProc:") (PARAMS (void*) msg);
    }
void CLinkedAgent::CollisionDetected (void* reserved)
    {
        myAgent ("CollisionDetected:") (PARAMS (void*) reserved);
    }
```

Файл LAgents\Module.cpp

```
// Module.cpp - implementation of Module.h, of module LAgents
// created 03.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "LAgents"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
//#include "Module.h"

#include "LAgentCore.h"
#include "LServer.h"

#include <Exeption.h>      // use ExeptionEngine
```

```
#include "..\Common\Utilities\Module.h"
// {811A46EA-3A95-4972-AC54-D93414003DA1}
static const GUID IID_Fignya =
{ 0x811a46ea, 0x3a95, 0x4972, { 0xac, 0x54, 0xd9, 0x34, 0x14, 0x0, 0x3d, 0xa1 } };
class CModule: public IUnknown
{DEFINE_IUNKNOWN();};
IMPLEMENT_IUNKNOWN (CModule, IUnknown, IID_Fignya);

    #define DLL_NAME "LAgents"
    #define DLL_CLASS CModule
    #include "..\Common\Utilities\Module.c"


IAgent* CreateAgent ( int i)
    {
        const char* str = LA::AServer->GetAgentCreatorName ( i);
        if ( str)
            return new CLinkedAgent (str);
        else
            return 0;
    }

void RegisterAgentCreator ( IConsole* c, int, char* data)
    {
        char* name = new char [ strlen ( data) + 1]; //!!!!!! undeleted new
        strcpy ( name, data);
        LA::AServer->RegisterAgentCreator ( CreateAgent, name, "Unknown", 0);
        int ai = LA::AServer->FindAgentCreatorByName ( name);
        LA::Console->printf ( "New ainame = %d.\n", ai);
    }

DLL_INIT (loader)
    {
        Assert (loader);

        LinkerDataTypesLibInit ( loader);

        loader->GetInterface ( IID_IAgentsServer, (void**)&LA::AServer);
        Assert ( LA::AServer);
        LA::Server = LA::AServer->GetAgentsInterface();
        Assert ( LA::Server);
        loader->GetInterface ( IID_ILinker, (void**)&LA::Linker);
        Assert ( LA::Linker);
        loader->GetInterface ( IID_ICPPVM, (void**)&LA::CPPVM);
        Assert ( LA::CPPVM);
        loader->GetInterface ( IID_ISmallVM, (void**)&LA::SmVM);
        Assert ( LA::SmVM);
        loader->GetInterface ( IID_IConsole, (void**)&LA::Console);
        Assert ( LA::Console);

        Assert ( CLinkedAgentsServerAgents::selfReg() && CAgentData::selfReg());

        LA::Console->AddCommand ( "RegisterAgentCreator", (CommandProcP)RegisterAgentCreator, false);

        LA::Linker->getGlobalScope().addVar ( CVar ( CObjRef ( new CLinkedAgentsServerAgents()),
"Server"));
    }

DLL_ONLOAD (loader)
{
    IConsole* con = NULL;
    loader->GetInterface (IID_IConsole, (void**)&con);
    Assert (con);
    con->printf ("LAGENTS: Linker Agents Service by Se version 1.0\n");
}
```

Файл LAgents\system.cpp

```
#include "system.h"

ISmallVM*               LA::SmVM = 0;
ICPPVM*                 LA::CPPVM = 0;
ILinker*                LA::Linker = 0;
IAgentsServer*          LA::AServer = 0;
IAgentsServerAgents*    LA::Server = 0;
IConsole*               LA::Console = 0;
```

Файл LAgents\LAgentCore.cpp

```
#include "LAgentCore.h"
#include "system.h"

SClassRef CAgentData::sClass = SClassRef ( 0);

CAgentData::CAgentData()
    {
```

```
        myClass = sClass;
    }

bool CAgentData::selfReg()
    {
        sClass = LA::CPPVM->registerClass ("LinkedAgent", "LinkedAgent");

        LA::CPPVM->registerFunc ( sClass, "=", (pObjFunc) operator==, ( TYPES SParam::eAny));

        LA::CPPVM->registerFunc ( sClass, "rX", (pObjFunc) CAgentData::rX, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rY", (pObjFunc) rY, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rZ", (pObjFunc) rZ, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rIndex", (pObjFunc) rIndex, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "rSHBC", (pObjFunc) rSHBP, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "setSHBP", (pObjFunc) setSHBP, (TYPES SParam::eBool));

        return 1;
    }

SParam/*eBool*/    CAgentData::operator== ( const CParams& pars) //1: 0 - eAny
    {
        switch ( pars[0].type) {
            case SParam::ePObj : {
                return false;
                }
            case SParam::eBool :
            default:
                return false;
            }
    }

SParam CAgentData::rX(const CParams&)
    { return m_x; }
SParam CAgentData::rY(const CParams&)
    { return m_y; }
SParam CAgentData::rZ(const CParams&)
    { return m_z; }

SParam CAgentData::rIndex(const CParams&)
    {
        ;
        return m_index;
    }
SParam CAgentData::rSHBP(const CParams&)
    { return m_shouldBeProcessed;}
SParam CAgentData::setSHBP(const CParams& par)
    { return m_shouldBeProcessed = par[0];}

//---------------------------------------------------------------------------------------
CLinkedAgent::CLinkedAgent()
    {
        myAgent = LA::Linker->getGlobalScope()("Pimp") ("new")();
        myAgent ("InitData:") (PARAMS CObjRef ( this));
    }
CLinkedAgent::CLinkedAgent ( const char* str)
    {
        myAgent = LA::Linker->getGlobalScope()(str) ("new")();
        myAgent ("InitData:") (PARAMS CObjRef ( this));
    }

int CLinkedAgent::Go()
    {
        return (int) myAgent ("Go") ();
    }

int CLinkedAgent::Ok()
    {
        return (int) myAgent ("Ok") ();
    }

void CLinkedAgent::Init()
    {
        myAgent ("Init") ();
    }

int CLinkedAgent::MessageProc(AgentMessage* msg)
    {
        return (int) myAgent ("MessageProc:") (PARAMS (void*) msg);
    }
void CLinkedAgent::CollisionDetected (void* reserved)
    {
        myAgent ("CollisionDetected:") (PARAMS (void*) reserved);
    }
```

Файл LAgents\LServer.cpp

```
#include "LServer.h"
#include "LAgentCore.h"

SClassRef CLinkedAgentsServerAgents::sClass = SClassRef ( 0);
```

```
CLinkedAgentsServerAgents::CLinkedAgentsServerAgents()
    {
        myClass = sClass;
    }

bool CLinkedAgentsServerAgents::selfReg()
    {
        sClass = LA::CPPVM->registerClass ("LinkedServer", "LinkedServer");

        LA::CPPVM->registerFunc ( sClass, "CreateAgentByName:", (pObjFunc) CreateAgentByName, ( TYPES
SParam::eStr));
        LA::CPPVM->registerFunc ( sClass, "CreateAgentByAI:", (pObjFunc) CreateAgentByAI, ( TYPES
SParam::eInt));
        LA::CPPVM->registerFunc ( sClass, "RemoveAgent:", (pObjFunc) RemoveAgent, ( TYPES SParam::eInt));
        LA::CPPVM->registerFunc ( sClass, "GetAgent:", (pObjFunc) GetAgent, ( TYPES SParam::eInt));
        LA::CPPVM->registerFunc ( sClass, "FindAgent:", (pObjFunc) FindAgent, ( TYPES SParam::ePObj));
        LA::CPPVM->registerFunc ( sClass, "GetAgentOf:Number:", (pObjFunc) GetAgentOfNumber, ( TYPES
SParam::eStr, SParam::eInt));
        LA::CPPVM->registerFunc ( sClass, "ReHash", (pObjFunc) ReHash, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "GetNumAgents", (pObjFunc) GetNumAgents, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "GetNumAgentsOf:", (pObjFunc) GetNumAgentsOf, ( TYPES
SParam::eStr));

        LA::CPPVM->registerFunc ( sClass, "RegisterAgentCreator:", (pObjFunc) RegisterAgentCreator, (
TYPES SParam::eStr));
        LA::CPPVM->registerFunc ( sClass, "FindAgentCreator:", (pObjFunc) FindAgentCreator, ( TYPES
SParam::eStr));
        LA::CPPVM->registerFunc ( sClass, "GetMaxCreators", (pObjFunc) GetMaxCreators, CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "GetAgentCreatorName:", (pObjFunc) GetAgentCreatorName, ( TYPES
SParam::eInt));

        LA::CPPVM->registerFunc ( sClass, "SayToUserFrom:What:", (pObjFunc) lSayToUser, ( TYPES
SParam::ePObj, SParam::eStr));
        LA::CPPVM->registerFunc ( sClass, "SetSpeedFor:X:Y:Z:", (pObjFunc) lSetSpeed, ( TYPES
SParam::ePObj, SParam::eFloat, SParam::eFloat, SParam::eFloat));
        LA::CPPVM->registerFunc ( sClass, "SetAccelerationFor:X:Y:Z:", (pObjFunc) lSetAcceleration, (
TYPES SParam::ePObj, SParam::eFloat, SParam::eFloat, SParam::eFloat));
        LA::CPPVM->registerFunc ( sClass, "GetSpeedXFor:", (pObjFunc) lGetSpeedX, ( TYPES SParam::ePObj));
        LA::CPPVM->registerFunc ( sClass, "GetSpeedYFor:", (pObjFunc) lGetSpeedY, ( TYPES SParam::ePObj));
        LA::CPPVM->registerFunc ( sClass, "GetSpeedZFor:", (pObjFunc) lGetSpeedZ, ( TYPES SParam::ePObj));
        LA::CPPVM->registerFunc ( sClass, "GetAccelerationXFor:", (pObjFunc) lGetAccelerationX, ( TYPES
SParam::ePObj));
        LA::CPPVM->registerFunc ( sClass, "GetAccelerationYFor:", (pObjFunc) lGetAccelerationY, ( TYPES
SParam::ePObj));
        LA::CPPVM->registerFunc ( sClass, "GetAccelerationZFor:", (pObjFunc) lGetAccelerationZ, ( TYPES
SParam::ePObj));
        LA::CPPVM->registerFunc ( sClass, "GetFallingAcceleration", (pObjFunc) lGetFallingAcceleration,
CParamsTypes());
        LA::CPPVM->registerFunc ( sClass, "GetLandscapeLevelAtX:Y:", (pObjFunc) GetLandscapeLevelAtXY, (
TYPES SParam::eFloat, SParam::eFloat));
        LA::CPPVM->registerFunc ( sClass, "SetAgentPositionFor:X:Y:Z:", (pObjFunc) lSetAgentPosition, (
TYPES SParam::ePObj, SParam::eFloat, SParam::eFloat, SParam::eFloat));

        return 1;
    }

SParam/*self*/ CLinkedAgentsServerAgents::CreateAgentByName( const CParams& pars) //1: 0 - eStr.
    {
        CString command ( "addagent ");
        command << pars[0].rString();
        LA::Console->RaiseCommand ( command.c_str());
        return CObjRef ( this);
    }

SParam/*self*/ CLinkedAgentsServerAgents::CreateAgentByAI( const CParams& pars) //1: 0 - eInt.
    {
        const char* name = LA::AServer->GetAgentCreatorName ( pars[0].rInt());
        CString command ( "addagent ");
        command << name;
        LA::Console->RaiseCommand ( command.c_str());
        return CObjRef ( this);
    }

SParam/*eInt*/ CLinkedAgentsServerAgents::RemoveAgent( const CParams& pars) //1: 0 - eInt.
    {
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;
        return as->RemoveAgent ( pars[0].rInt());

    }

SParam/*LAgent**/ CLinkedAgentsServerAgents::GetAgent( const CParams& pars) //1: 0 - eInt.
    {
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;
        IAgent* a = s->GetAgent ( pars[0].rInt());

        CAgentData* ag = dynamic_cast<CLinkedAgent*> ( a);
        if ( ag)
            return CObjRef ( ag);
        else
            return LA::Linker->getGlobalScope() ("nil");
    }

SParam/*eInt*/ CLinkedAgentsServerAgents::FindAgent( const CParams& pars) //1: 0 - LAgent*.
    {
```

```
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;

        CLinkedAgent* ag = dynamic_cast<CLinkedAgent*> ( pars[0].rObj().rObj());
        return ag? as->FindAgent ( ag) : 0;
    }

SParam/*LAgent*/   CLinkedAgentsServerAgents::GetAgentOfNumber       ( const CParams& pars) //2: 0 - eStr
1 - eInt
    {
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;

        int ai = as->FindAgentCreatorByName ( const_cast<char*> ( pars[0].rString()));

        int n = 0;
        for ( int i = 0; i < as->GetNumAgents(); i++) {
            IAgent* cur = as->GetAgent ( i);
            if ( cur && cur->m_ainame == ai) {
                if ( n == pars[1].rInt()) {
                    CLinkedAgent* ag = dynamic_cast<CLinkedAgent*> ( cur);
                    return ag? ag->rLinkedAgent() : false;
                    }
                n++;
                }
            }
        return false;
    }

SParam/*eInt*/ CLinkedAgentsServerAgents::GetNumAgentsOf     ( const CParams& pars) //1: 0 - eStr
    {
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;

        int ai = as->FindAgentCreatorByName ( const_cast<char*> ( pars[0].rString()));

        int n = 0;
        for ( int i = 0; i < as->GetNumAgents(); i++) {
            IAgent* cur = as->GetAgent ( i);
            if ( cur && cur->m_ainame == ai)
                n++;
            }
        return n;
    }

SParam/*self*/ CLinkedAgentsServerAgents::ReHash( const CParams& pars) //0.
    {
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;

        as->ReHash();

        return CObjRef ( this);
    }

SParam/*eInt*/ CLinkedAgentsServerAgents::GetNumAgents( const CParams& pars)//0.
    {
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;

        return as->GetNumAgents();
    }

SParam/*eInt*/ CLinkedAgentsServerAgents::RegisterAgentCreator( const CParams& pars)//1: 0 - eStr
    {
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;

        as->RegisterAgentCreator ( CreateAgent, const_cast<char*> ( pars[0].rString()), "Unknown", 0);
        return as->FindAgentCreatorByName ( const_cast<char*> ( pars[0].rString()));
    }

SParam/*eInt*/ CLinkedAgentsServerAgents::FindAgentCreator( const CParams& pars)//1: 0 - eStr
    {
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;

        return as->FindAgentCreatorByName ( const_cast<char*> ( pars[0].rString()));
    }

SParam/*eInt*/ CLinkedAgentsServerAgents::GetMaxCreators( const CParams& pars)//0.
    {
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;

        return as->GetMaxCreators();
    }

SParam/*eStr*/ CLinkedAgentsServerAgents::GetAgentCreatorName( const CParams& pars) //1: 0 - eInt
    {
        IAgentsServerAgents* s = LA::Server;
        IAgentsServer* as = LA::AServer;

        return as->GetAgentCreatorName ( pars[0].rInt());
    }

SParam CLinkedAgentsServerAgents::lSayToUser ( const CParams& par) //1 - LinkedAgent, 2 - string
```

```
    {
        LA::Server->SayToUser ( (int) ( par[0].rObj() ("rIndex")()), (char*) par[1].rString());
        return SParam();
    }
SParam CLinkedAgentsServerAgents::lSetSpeed ( const CParams& par) //1 - LinkedAgent, 2 - float x, 3 -
float y, 4 - float z
    {
        LA::Server->SetSpeed ( (int) ( par[0].rObj() ("rIndex")()), par[1].rFloat(), par[2].rFloat(),
par[3].rFloat());
        return SParam();
    }
SParam CLinkedAgentsServerAgents::lSetAcceleration ( const CParams& par) //1 - LinkedAgent, 2 - float x, 3
- float y, 4 - float z
    {
        LA::Server->SetAcceleration ( (int) ( par[0].rObj() ("rIndex")()), par[1].rFloat(),
par[2].rFloat(), par[3].rFloat());
        return SParam();
    }
SParam CLinkedAgentsServerAgents::lGetSpeedX ( const CParams& par) //1 - LinkedAgent
    {
        float x;
        float y;
        float z;
        LA::Server->GetSpeed ( (int) ( par[0].rObj() ("rIndex")()), &x, &y, &z);
        return x;
    }
SParam CLinkedAgentsServerAgents::lGetSpeedY ( const CParams& par) //1 - LinkedAgent
    {
        float x;
        float y;
        float z;
        LA::Server->GetSpeed ( (int) ( par[0].rObj() ("rIndex")()), &x, &y, &z);
        return y;
    }
SParam CLinkedAgentsServerAgents::lGetSpeedZ ( const CParams& par) //1 - LinkedAgent
    {
        float x;
        float y;
        float z;
        LA::Server->GetSpeed ( (int) ( par[0].rObj() ("rIndex")()), &x, &y, &z);
        return z;
    }
SParam CLinkedAgentsServerAgents::lGetAccelerationX ( const CParams& par) //1 - LinkedAgent
    {
        float x;
        float y;
        float z;
        LA::Server->GetAcceleration ( (int) ( par[0].rObj() ("rIndex")()), &x, &y, &z);
        return x;
    }
SParam CLinkedAgentsServerAgents::lGetAccelerationY ( const CParams& par) //1 - LinkedAgent
    {
        float x;
        float y;
        float z;
        LA::Server->GetAcceleration ( (int) ( par[0].rObj() ("rIndex")()), &x, &y, &z);
        return x;
    }
SParam CLinkedAgentsServerAgents::lGetAccelerationZ ( const CParams& par) //1 - LinkedAgent
    {
        float x;
        float y;
        float z;
        LA::Server->GetAcceleration ( (int) ( par[0].rObj() ("rIndex")()), &x, &y, &z);
        return x;
    }
SParam CLinkedAgentsServerAgents::lGetFallingAcceleration ( const CParams&)
    {
        return LA::Server->GetFallingAcceleration();
    }
SParam /*float*/ CLinkedAgentsServerAgents::GetLandscapeLevelAtXY ( const CParams& par) //1 - float x 2 -
float y
    {
        Assert ( par.num == 2 && "CLinkedAgentsServerAgents::GetLandscapeLevelAtX:Y:");
        return LA::Server->GetLandscapeLevel ( (float)par[0], (float)par[1]);
    }

SParam /*float*/ CLinkedAgentsServerAgents::lSetAgentPosition ( const CParams& par)
    {
        Assert ( par.num == 4);
        LA::Server->SetAgentPosition ( (int) ( par[0].rObj() ("rIndex")()), par[1].rFloat(),
par[2].rFloat(), par[3].rFloat());
        return 1; //!!!
    }
```

Файл Landscape\landscap.cpp

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** LANDSCAP.CPP                                                      ***/
```

```
/*** ╠╧╚╚╘╦╤╠═╚╧┤ ╠╪╤╦╠═╦╠╤ ╠╚╤╘╚═╦╠╤ ═╦▀ ╨╘╚╦╠▄▌ ╤ ╤╘═╪╘╚╦╘╠╚, ╠╚╘┌▀╚╤┤═▄┌ ***/
/*** ┬ ╘╚╔╔╤┼ LANDSCAP.H                                                  ***/
/***                                                                      ***/
/*** ╘╚╔╔ ═╘╚╚╦╘═ ╤╚╦╨╨╚═╦╤▄╠ ╘═╨╪╪╠ ╤┼╨┼┼┼┬╚╪┼╠ ┬ 2002 ┤╠═╨          ***/
/***                                                                      ***/
/***                                                                      ***/
/*************************************************************************/
/*************************************************************************/

#include "landscap.h"

#include <windows.h>

#define __PROGRAMMER "Xroft"
#define __MODULE_NAME "Landscape"

#include <exeption.h>

BOOL APIENTRY DllMain(HANDLE hModule,
                      DWORD  ul_reason_for_call,
                      LPVOID lpReserved)
{
return 1;
}

/*************************************************************************/
/*************************************************************************/
/********************* CLASS CFLOATMATRIX ********************************/
/*************************************************************************/
/*************************************************************************/

/*************************************************************************/

CFloatMatrix::CFloatMatrix ()
    {
    myMatrix = NULL;
    }

/*************************************************************************/

CFloatMatrix::CFloatMatrix (int x, int y) :
    mySizeX (x),
    mySizeY (y)
    {
    myMatrix = new float [x * y];
    }

/*************************************************************************/

CFloatMatrix::~CFloatMatrix ()
    {
    delete [] myMatrix;
    }

/*************************************************************************/

void CFloatMatrix::Init (int x, int y)
    {
    mySizeX = x;
    mySizeY = y;
    myMatrix = new float [x * y];
    }

/*************************************************************************/

int CFloatMatrix::RetX ()
    {
    return mySizeX;
    }

/*************************************************************************/

int CFloatMatrix::RetY ()
    {
    return mySizeY;
    }

/*************************************************************************/

int CFloatMatrix::ok (int x, int y)
    {
    return x < mySizeX && y < mySizeY;
    }

/*************************************************************************/

float &CFloatMatrix::RetElement (int x, int y)
    {
    AssertT (ok (x, y), "Element is out of range");
    return myMatrix [y * mySizeX + x];
    }

/*************************************************************************/

float &CFloatMatrix::operator () (int x, int y)
    {
    return RetElement (x, y);
```

```
    }
/*************************************************************************/

void CFloatMatrix::Fill (float n)
    {
    for (int y = 0; y < mySizeY; y ++)
        {
        for (int x = 0; x < mySizeX; x ++)
            {
            RetElement (x, y) = n;
            }
        }
    }
/*************************************************************************/

/*************************************************************************/
/*************************************************************************/
/********************* CLASS CPOINTMATRIX ********************************/
/*************************************************************************/
/*************************************************************************/

/*************************************************************************/

CPointMatrix::CPointMatrix ()
    {
    myMatrix = NULL;
    }

/*************************************************************************/

CPointMatrix::CPointMatrix (int x, int y)
    {
    Init (x, y);
    }

/*************************************************************************/

CPointMatrix::~CPointMatrix ()
    {
    delete [] myMatrix;
    }

/*************************************************************************/

void CPointMatrix::Init (int x, int y)
    {
    mySizeX = x;
    mySizeY = y;
    myMatrix = new float [x * y * FLOATSINPOINT];
    }

/*************************************************************************/

int CPointMatrix::RetX ()
    {
    return mySizeX;
    }

/*************************************************************************/

int CPointMatrix::RetY ()
    {
    return mySizeY;
    }

/*************************************************************************/

int CPointMatrix::ok (int x, int y, int z)
    {
    return x < mySizeX && y < mySizeY && z < FLOATSINPOINT;
    }

/*************************************************************************/

float &CPointMatrix::RetElement (int x, int y, int z)
    {
    AssertT (ok (x, y, z), "Element is out of range");
    return myMatrix [(y * mySizeX + x) * FLOATSINPOINT + z];
    }

/*************************************************************************/

float &CPointMatrix::operator () (int x, int y, int z)
    {
    return RetElement (x, y, z);
    }

/*************************************************************************/

void CPointMatrix::Fill (float x, float y, float z)
    {
    for (int y1 = 0; y1 < mySizeY; y1 ++)
        {
        for (int x1 = 0; x1 < mySizeX; x1 ++)
            {
```

```
            RetElement (x1, y1, 0) = x;
            RetElement (x1, y1, 1) = y;
            RetElement (x1, y1, 2) = z;
            }
        }
    }

/*****************************************************************************/

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CHORIZONT **************************************/
/*****************************************************************************/
/*****************************************************************************/

/*****************************************************************************/

CHorizont::CHorizont ()
    {
    myMatrix = NULL;
    }

/*****************************************************************************/

CHorizont::CHorizont (int x, int y)
    {
    Init (x, y);
    }

/*****************************************************************************/

CHorizont::CHorizont (int x, int y, float sqrsize) :
    mySquareSize (sqrsize)
    {
    Init (x, y);
    }

/*****************************************************************************/

CHorizont::~CHorizont ()
    {
    }

/*****************************************************************************/

void CHorizont::Set (int x, int y, float sqrsize)
    {
    Init (x, y);
    mySquareSize = sqrsize;
    }

/*****************************************************************************/

void CHorizont::SetColor (float r, float g, float b, float a)
    {
    myColor [0] = r;
    myColor [1] = g;
    myColor [2] = b;
    myColor [3] = a;
    }

/*****************************************************************************/

float * CHorizont::RetColor ()
    {
    return myColor;
    }

/*****************************************************************************/

void CHorizont::SetTexParameters (float x, float y)
    {
    myTexRepX = x;
    myTexRepY = y;
    }

/*****************************************************************************/

float CHorizont::RetTexX ()
    {
    return myTexRepX;
    }

/*****************************************************************************/

float CHorizont::RetTexY ()
    {
    return myTexRepY;
    }

/*****************************************************************************/

void CHorizont::SetDetailTexID (int id)
    {
    myDetailTexID = id;
    }
```

```
/*************************************************************************/

int CHorizont::RetDetailTexID ()
    {
    return myDetailTexID;
    }

/*************************************************************************/

void CHorizont::SetGlobalTexID (int id)
    {
    myGlobalTexID = id;
    }

/*************************************************************************/

int CHorizont::RetGlobalTexID ()
    {
    return myGlobalTexID;
    }

/*************************************************************************/

void CHorizont::SetSquareSize (float sqrsize)
    {
    mySquareSize = sqrsize;
    }

/*************************************************************************/

float CHorizont::RetSquareSize ()
    {
    return mySquareSize;
    }

/*************************************************************************/

/*************************************************************************/
/*************************************************************************/
/********************* CLASS CLANDSCAPE **********************************/
/*************************************************************************/
/*************************************************************************/

/*************************************************************************/

CLandscape::CLandscape ()
    {
    myHorizonts = NULL;
    }

/*************************************************************************/

CLandscape::CLandscape (int n)
    {
    Init (n);
    }

/*************************************************************************/

CLandscape::~CLandscape ()
    {
    delete [] myHorizonts;
    }

/*************************************************************************/

void CLandscape::Init (int n)
    {
    myHorNum = n;
    myCurHor = 0;
    myHorizonts = new CHorizont [n];
    myNormalMaps = new CPointMatrix [n];
    }

/*************************************************************************/

int CLandscape::ok ()
    {
    return myCurHor < myHorNum;
    }

/*************************************************************************/

int CLandscape::ok (int n)
    {
    return n < myCurHor;
    }

/*************************************************************************/

int CLandscape::InitNewHor ()
    {
    AssertT (ok (), "Can't init new horizont - no place");
    myCurHor ++;
    return myCurHor - 1;
    }
```

```
/*****************************************************************************/

int CLandscape::InitNewHor (int x, int y, float sqrsize)
    {
    AssertT (ok (), "Can't init new horizont - no place");
    myHorizonts [myCurHor].Set (x, y, sqrsize);
    myNormalMaps [myCurHor].Init (x, y);
    myCurHor ++;
    return myCurHor - 1;
    }

/*****************************************************************************/

int CLandscape::RetSize ()
    {
    return myCurHor;
    }

/*****************************************************************************/

void CLandscape::SetColor (int handle, float r, float g, float b, float a)
    {
    AssertT (ok (handle), "Handle is out of range");
    myHorizonts [handle].SetColor (r, g, b, a);
    }

/*****************************************************************************/

float * CLandscape::RetColor (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myHorizonts [handle].RetColor ();
    }

/*****************************************************************************/

void CLandscape::SetDetailTexID (int handle, int id)
    {
    AssertT (ok (handle), "Handle is out of range");
    myHorizonts [handle].SetDetailTexID (id);
    }

/*****************************************************************************/

int CLandscape::RetDetailTexID (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myHorizonts [handle].RetDetailTexID ();
    }

/*****************************************************************************/

void CLandscape::SetGlobalTexID (int handle, int id)
    {
    AssertT (ok (handle), "Handle is out of range");
    myHorizonts [handle].SetGlobalTexID (id);
    }

/*****************************************************************************/

int CLandscape::RetGlobalTexID (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myHorizonts [handle].RetGlobalTexID ();
    }

/*****************************************************************************/

void CLandscape::SetTexParameters (int handle, float x, float y)
    {
    AssertT (ok (handle), "Handle is out of range");
    myHorizonts [handle].SetTexParameters (x, y);
    }

/*****************************************************************************/

float CLandscape::RetTexX (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myHorizonts [handle].RetTexX ();
    }

/*****************************************************************************/

float CLandscape::RetTexY (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myHorizonts [handle].RetTexY ();
    }

/*****************************************************************************/

void CLandscape::SetSquareSize (int handle, float sqrsize)
    {
    AssertT (ok (handle), "Handle is out of range");
    myHorizonts [handle].SetSquareSize (sqrsize);
    }
```

```
/*************************************************************************/

float CLandscape::RetSquareSize (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myHorizonts [handle].RetSquareSize ();
    }

/*************************************************************************/

void CLandscape::Fill (int handle, float n)
    {
    AssertT (ok (handle), "Handle is out of range");
    myHorizonts [handle].Fill (n);
    }

/*************************************************************************/

CHorizont & CLandscape::RetHorizont (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myHorizonts [handle];
    }

/*************************************************************************/

CHorizont * CLandscape::RetPHorizont (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return &myHorizonts [handle];
    }

/*************************************************************************/

CPointMatrix & CLandscape::RetNormalMap (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myNormalMaps [handle];
    }

/*************************************************************************/

CPointMatrix * CLandscape::RetPNormalMap (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return &myNormalMaps [handle];
    }

/*************************************************************************/
```

Файл Linker\linker.cpp

```
#include "linker.h"
```

Файл Linker\linkercore.cpp

```
#include "linkercore.h"

CLinker::CLinker() :
    myClasses (1),
    myStrings (10)
    { myClasses.add (NULL); myStrings.add ("big big error!!! ~!@#$%^&*() !!!"); }

CLinker::~CLinker()
    { }

bool CLinker::ok() const
    { return myClasses.ok() && myStrings.ok(); }

int CLinker::registerName (const CString& str)
    {
    if ( !str.rLength()) return 0;
    int n = myStrings.search (str);
    if (n == -1) n = myStrings.add (str);
    return n;
    }

const CString& CLinker::getName (int name)
    {
    Assert (0 < name && name < CLinker::myStrings.rNumber());
    return myStrings[name];
    }
```

```
int CLinker::checkName (const CString& name)
    {
    for (int i = 0; i < myStrings.rNumber(); i++)
        if (myStrings[i] == name) return i;
    return 0;
    }

SParam CLinker::call (CObj* obj, int classSelector, int funcSelector, const CParams& par)
    { Assert (ok());
      Assert (classSelector && funcSelector);
      return myClasses[classSelector]->myVTable.functions[funcSelector] (obj, par); }

SParam CLinker::call (CObj* obj, int name, const CParams& par)
    { Assert ( ok());
    Assert ( obj && name > 0);
    Assert ( obj->rClass());


    Func* funcToCall = obj->rClass()->findFunc (name);
    //Assert ( funcToCall && "tryin' to call unexistin' func");
    if (!funcToCall)
        {
        Trace ("LINKER: Trying to call unexisting func '%s' for '%s'", getName (name).c_str(), getName
(obj->rClass()->myClassName).c_str());
//      Console->printf ("LINKER: Trying to call unexisting func '%s' for '%s'\n", getName (name).c_str(),
getName (obj->rClass()->myClassName).c_str());
        return 0;
        }
    return funcToCall->operator () (obj, par);
    }

SParam CLinker::staticCall (int className, int name, const CParams& par)
    { Assert (ok());
      Assert (name > 0);

    StaticFunc* statFuncToCall = getClassPtr(className)->findStaticFunc (name);
    Assert ( statFuncToCall);
    return statFuncToCall->operator () (par);
    }

CScope& CLinker::getGlobalScope()
    { return myGlobalScope; }

Func* CLinker::rFunc (int name, int className)
    { Assert (ok());

    CCont<Func>* pFunc = NULL;
    for (int i = 1; i < myClasses.rNumber(); i++)
        if (myClasses[i]->myClassName == className) pFunc = &myClasses[i]->myVTable.functions;
    if (!pFunc) return NULL;
    for ( i = 0; i < pFunc->rNumber(); i++)
        if ((*pFunc)[i].myName == name) return &(*pFunc)[i];
    return NULL;
    }

SClassRef CLinker::registerClass ( int clName, int varName)
    { Assert ( ok());

    if ( int i = getClassRef (clName)) return SClassRef (i);

    if ( varName) Assert ( !Linker->getGlobalScope().check ( varName));

    for (int i = 1; i < myClasses.rNumber(); i++)
        if (myClasses[i]->myClassName == clName)
            return SClassRef (i);

    myClasses.add ( new SClass (clName));  //will be deleted by garbage collector

    i = myClasses.rNumber() - 1;

    if ( varName) Linker->getGlobalScope().addVar ( CVar ( CObjRef ( myClasses[i]), varName));

    Console->printf ("Class \"%s\" is registered as global \"%s\"\n",
                    Linker->getName (clName).rString(),
                    varName? Linker->getName (varName).rString() : "");
    return SClassRef (i);
    }

CCont<SClass*>& CLinker::getClasses()
    { return myClasses; }

SClass* CLinker::getClassPtr (int className)
    { SClassRef cl = getClassRef (className);
      if (cl) return myClasses[cl];
      else return 0; }

SClassRef CLinker::getClassRef (const SClass* clPtr)
    { if ( !clPtr) return SClassRef (0);
      for ( int i = 1; i < myClasses.rNumber(); i++)
        if ( myClasses[i] == clPtr) return SClassRef (i);
      return 0; }

SClassRef CLinker::getClassRef (int className)
    { return SClassRef (getClassSelector (className)); }

int CLinker::getClassSelector (const CString& className)
    { return getClassSelector ( registerName (className)); }
```

```
int CLinker::getClassSelector (int className)
    {
    Assert (ok());
    Assert (className);

    for (int i = 1; i < myClasses.rNumber(); i++)
        if (myClasses[i]->myClassName == className) return i;
    return 0;
    }
int CLinker::getClassName (int sel)
    {
    return myClasses[sel]->myClassName;
    }

Func* CLinker::registerFunc ( const Func& func)
    {
    Assert ( func.ok());

    int classSelector = getClassSelector (func.myClassName);

    return myClasses[classSelector]->registerFunc (func);
    }

StaticFunc* CLinker::registerStaticFunc ( const StaticFunc& func)
    {
    Assert ( func.ok());

    int classSelector = getClassSelector (func.myClassName);

    return myClasses[classSelector]->registerStaticFunc (func);
    }

IMPLEMENT_IUNKNOWN (CLinker, ILinker, IID_ILinker);
```

Файл Linker\module.cpp

```
#define __PROGRAMMER   "Se"
#define __MODULE_NAME  "Console.dll initialization"

//#include <exeption.h>
#include "System.h"
#include "linkercore.h"

    #define DLL_NAME "Linker"
    #define DLL_CLASS CLinker
    #include "..\Common\Utilities\Module.c"


DLL_INIT(loader)
{
    Assert (loader);
    Loader = loader;

    LinkerDataTypesLibInit ( loader);

    Linker = (ILinker*)Loader->GetInterface (IID_ILinker);
    Loader->GetInterface (IID_IConsole, (void**)&Console);
    Assert (Console);
    Assert (Linker);
    Console->printf ("Linker initialized\n");
}

DLL_ONLOAD (loader)
{
    IConsole* con = NULL;
    loader->GetInterface (IID_IConsole, (void**)&con);
    Assert (con);
    con->printf ("LINKER: Linker by Se version 1.0\n");
}
```

Файл Linker\system.cpp

```
#define __PROGRAMMER   "Se"
#define __MODULE_NAME  "System"

#include "System.h"

//#include <Exeption.h>      // use ExeptionEngine

ILoader* Loader = 0;
ILinker* Linker = 0;
IConsole* Console = 0;
```

Файл LinkerDataTypes\all.cpp

```
#define __PROGRAMMER "Se"
#define __MODULE_NAME "Linker DataTypes static-link"

#include "stdafx.h"

#include "..\Linker\linker.h"

//-----------------------------------------------------------------------------------------------

UParam::UParam() : l (0) { }
UParam::UParam ( bool bo) : b (bo) {}
UParam::UParam ( char ch) : c (ch) { }
UParam::UParam ( int  in) : i (in) { }
UParam::UParam ( long lo) : l (lo) { }
UParam::UParam ( float fl) : f (fl) { }
UParam::UParam ( double db) : d (db) { }
UParam::UParam ( const char* stri) : str (stri) { }
UParam::UParam ( void* p) : pVoid (p) { };
UParam::UParam ( CObj* p) : pObj (p) { };

//-----------------------------------------------------------------------------------------------

SParam::SParam() : type (eAny) { }
SParam::SParam (bool bo) : param (bo), type (eBool) { }
SParam::SParam (char ch) : param (ch), type (eChar) { }
SParam::SParam (int i) : param (i), type (eInt) { }
SParam::SParam (long l) : param (l), type (eLong) { }
SParam::SParam (float f) : param (f), type (eFloat) { }
SParam::SParam (double d) : param (d), type (eDouble) { }
SParam::SParam (const char* str) : param (str), type (eStr) { }
SParam::SParam (void * ptr) : param (ptr), type (ePVoid) { }

SParam::SParam (CObjRef& Obj) :
    param (Obj.rObj()),
    type (ePObj)
    {
    if ( param.pObj) param.pObj->incRefs();
    }

SParam::SParam (CObj* Obj) :
    param (Obj),
    type (ePObj)
    {
    if ( param.pObj) param.pObj->incRefs();
    }

SParam::~SParam()
    {
    if ( type == ePObj && param.pObj) param.pObj->decRefs();
    }

SParam::SParam ( const SParam& par) :
    param ( par.param),
    type ( par.type)
    {
    if ( type == ePObj && param.pObj) param.pObj->incRefs();
    }

SParam& SParam::operator= ( const SParam& par)
    {
    param = par.param;
    type = par.type;
    if ( type == ePObj && param.pObj) param.pObj->incRefs();
    return *this;
    }

bool SParam::rBool() const
    {
    if (type == eBool) return param.b;
    switch (type)
        {
//      case eBool   : return param.b;
        case eChar   : return param.c? true : false;
        case eInt    : return param.i? true : false;
        case eLong   : return param.l? true : false;
        case eFloat  : return param.f? true : false;
        case eDouble : return param.d? true : false;
        case eStr    : return param.str? true : false;
        case ePVoid  : return param.pVoid? true : false;
        case ePObj   : return param.pObj? true : false;
//      case eAny   : return -1;
        default :      return 0;
        }
    }

char SParam::rChar() const
    {
    if (type == eChar) return param.c;
    switch (type)
        {
//      case eChar   : return param.c;
```

```
//      case eBool   : return param.b;
        case eInt    : return (char) param.i;
        case eLong   : return (char) param.l;
        case eFloat  : return (char) param.f;
        case eDouble : return (char) param.d;
        case eStr    : return param.str[0];
//      case ePVoid  : return -1;
//      case ePObj   : return -1;
//      case eAny    : return -1;
        default :      return -1;
        }
    }

int SParam::rInt() const
    {
    if (type == eInt) return param.i;
    switch (type)
        {
        case eBool   : return param.b;
        case eChar   : return param.c;
//      case eInt    : return param.i;
        case eLong   : return param.l;
        case eFloat  : return (int) param.f;
        case eDouble : return (int) param.d;
//      case eStr    : return -1;
//      case ePVoid  : return -1;
//      case ePObj   : return -1;
//      case eAny    : return -1;
        default      : return -1;
        }
    }

float SParam::rFloat() const
    {
    if (type == eFloat) return param.f;
    switch (type)
        {
        case eBool   : return param.b;
        case eChar   : return param.c;
        case eInt    : return (float) param.i;
        case eLong   : return (float) param.l;
//      case eFloat  : return param.f;
        case eDouble : return (float) param.d;
//      case eStr    : return -1;
//      case ePVoid  : return -1;
//      case ePObj   : return -1;
//      case eAny    : return -1;
        default      : return -1;
        }
    }

double SParam::rDouble() const
    {
    if (type == eDouble) return param.d;
    switch (type)
        {
        case eBool   : return param.b;
        case eChar   : return param.c;
        case eInt    : return param.i;
        case eLong   : return param.l;
        case eFloat  : return param.f;
//      case eDouble : return param.d;
//      case eStr    : return -1;
//      case ePVoid  : return -1;
//      case ePObj   : return -1;
//      case eAny    : return -1;
        default      : return -1;
        }
    }

long SParam::rLong() const
    {
    if (type == eLong) return param.l;
    switch (type)
        {
        case eBool   : return param.b;
        case eChar   : return param.c;
        case eInt    : return param.i;
//      case eLong   : return param.l;
        case eFloat  : return (long) param.f;
        case eDouble : return (long) param.d;
//      case eStr    : return -1;
//      case ePVoid  : return -1;
//      case ePObj   : return -1;
//      case eAny    : return -1;
        default      : return -1;
        }
    }

const char* SParam::rString() const
    {
    if (type == eStr) return param.str;

/*
    switch (type)
        {
//      case eBool   : return param.b;
```

```
//      case eChar   : return NULL;
//      case eInt    : return NULL;
//      case eLong   : return NULL;
//      case eFloat  : return NULL;
//      case eDouble : return NULL;
//      case eStr    : return param.str;
//      case ePVoid  : return NULL;
//      case ePObj   : return NULL;
//      case eAny    : return NULL;
        default      : return NULL;
        }
  */
    return NULL;
    }

void* SParam::rPVoid() const
    {
    if (type == ePVoid) return param.pVoid;
    switch (type)
        {
//      case eBool   : return param.b;
//      case eChar   : return NULL;
//      case eInt    : return NULL;
//      case eLong   : return NULL;
//      case eFloat  : return NULL;
//      case eDouble : return NULL;
        case eStr    : return (void*) param.str;
        case ePVoid  : return param.pVoid;
        case ePObj   : return param.pObj;
//      case eAny    : return NULL;
        default      : return NULL;
        }
    }

CObjRef SParam::rObj() const
    {
    if (type == ePObj) return param.pObj;
/*
    switch (type)
        {
//      case eBool   : return param.b;
//      case eChar   : return NULL;
//      case eInt    : return NULL;
//      case eLong   : return NULL;
//      case eFloat  : return NULL;
//      case eDouble : return NULL;
//      case eStr    : return NULL;
//      case ePVoid  : return (CObj*) param.pVoid;
//      case ePObj   : return param.pObj;
//      case eAny    : return NULL;
        default      : return NULL;
        }
*/
    return NULL;
    }

SParam::operator bool() const { return rBool(); }
SParam::operator char() const { return rChar(); }
SParam::operator int() const { return rInt(); }
SParam::operator long() const { return rLong(); }
SParam::operator float() const { return rFloat(); }
SParam::operator double() const { return rDouble(); }
SParam::operator const char*() const { return rString(); }
SParam::operator void*() const { return rPVoid(); }
SParam::operator CObj*() const { return rObj().rObj(); }
SParam::operator CObjRef() const { return rObj(); }

SObjFunc SParam::operator() ( const CString& fName) const
    {
    Assert (SParam::type == ePObj && "tryin' to call function of return non-object value"); //!!! redo
    return SObjFunc ( param.pObj, fName);
    }

//-----------------------------------------------------------------------------------------------

CParams::CParams ( bool ok) : num ( ok? 0 : -1) { }

CParams& CParams::operator, (bool b)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = b; return *this; }

CParams& CParams::operator, (char ch)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = ch; return *this; }

CParams& CParams::operator, (int i)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = i; return *this; }

CParams& CParams::operator, (long l)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = l; return *this; }

CParams& CParams::operator, (const char* str)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = str; return *this; }
```

```
CParams& CParams::operator, (void* ptr)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = ptr; return *this; }

CParams& CParams::operator, (CObjRef& ptr)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = ptr; return *this; }

CParams& CParams::operator, ( SParam& p)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = p; return *this; }

const SParam& CParams::operator[] (int i) const
    { Assert (0 <= i && i < num && "CParams::[] (i)"); return params[i]; }

SParam& CParams::operator[] (int i)
    { Assert ( 0 <= i && i < MaxParams);
    if ( i >= num) num = i + 1;
    return params[i];
    }

CParams& CParams::operator<< (bool b)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = b; return *this; }

CParams& CParams::operator<< (char ch)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = ch; return *this; }

CParams& CParams::operator<< (int i)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = i; return *this; }

CParams& CParams::operator<< (long l)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = l; return *this; }

CParams& CParams::operator<< (const char* str)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = str; return *this; }

CParams& CParams::operator<< (void* ptr)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = ptr; return *this; }

CParams& CParams::operator<< (CObjRef& ptr)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = ptr; return *this; }

CParams& CParams::operator<< ( SParam& p)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = p; return *this; }

CParams& CParams::add ( const SParam& par)
    { Assert (0 <= num && num - 1 < MaxParams);
      params[num++] = par; return *this; }

//-----------------------------------------------------------------------------------------------------

SParamTypeInfo::SParamTypeInfo ( SParam::Types t, const SClassRef& cl) :
    myType ( t),
    myClassRef ( cl)
    { }

//-----------------------------------------------------------------------------------------------------

CParamsTypes::CParamsTypes() :
    num (0)
    { }

CParamsTypes::CParamsTypes ( const CParams& params) :
    num (0)
    {
    for ( int i = 0; i < params.num; i++)
        if ( params[i].type == SParam::ePObj)
            {
            CObj* o = params[i].param.pObj;
            SClassRef cl = o? o->rClass() : SClassRef();
            (*this) << SParamTypeInfo ( SParam::ePObj, cl);
            continue;
            }
        else
            (*this) << params[i].type;
    }

CParamsTypes& CParamsTypes::operator<< ( SParamTypeInfo t)
    {
    Assert (0 <= num && num - 1 < MaxParams);
    types[num++] = t;
    return *this;
    }

CParamsTypes& CParamsTypes::operator , ( SParamTypeInfo t)
    {
    Assert (0 <= num && num - 1 < MaxParams);
    types[num++] = t;
    return *this;
```

```
    }
int CParamsTypes::conformsTo ( const CParamsTypes& params)
    {
    if ( params.num != num) return 100;

    int difference = 0;
    for ( int i = 0; i < num; i++)
        if ( params.types[i].myType == types[i].myType)
            {
            if ( types[i].myType == SParam::ePObj && (bool) params.types[i].myClassRef)
                {
                if ( ! types[i].myClassRef)
                    difference++;
                else
                    difference += types[i].myClassRef->conformsTo ( params.types[i].myClassRef)? 0 : 1;
                }
            continue;
            }
        else
            difference += 2; // !!! TODO; check possible convertions
    return difference;
    }

//-----------------------------------------------------------------------------------------------------

CObj::CObj() :
    myRefsNum (0)
    { }

CObj::~CObj()
    { }

bool CObj::ok() const
    { return myRefsNum >= 0; }

void CObj::incRefs ()
    {
    myRefsNum++;
    }

void CObj::decRefs ()
    {
    myRefsNum--;
    Assert (myRefsNum >= 0);
    if (!myRefsNum) delete this;
    }

SParam CObj::call (int expression, const CParams& par)
    { Assert (ok());
    return LinkerDataTypes::Linker->call (this, expression, par);
    }
SParam CObj::call (const CString& expression, const CParams& par)
    { return call ( LinkerDataTypes::Linker->registerName (expression)); }

//-----------------------------------------------------------------------------------------------

SObjFunc::SObjFunc() :
    fName (NULL)
    { }

SObjFunc::SObjFunc (CObjRef n, int name) :
    myObj  (n),
    fName (name)
    { }

SParam SObjFunc::operator() (const CParams& par)
    { Assert ( myObj.rObj() && "tryin' to call func of NULL-obj");
      return myObj.rObj()->call (fName, par); }

//-----------------------------------------------------------------------------------------------

CObjRef::CObjRef() :
    myObj (NULL)
    { }

CObjRef::CObjRef ( CObj* obj) :
    myObj ( obj)
    { if ( myObj) myObj->incRefs(); }

CObjRef::CObjRef ( const CObjRef& ref) :
    myObj ( ref.rObj())
    { if ( myObj) myObj->incRefs(); }

CObjRef& CObjRef::operator= (const CObjRef& ref)
    { if (myObj) myObj->decRefs();
      myObj = ref.rObj();
      if (myObj) myObj->incRefs();
      return *this; }

CObjRef::~CObjRef()
    { if (myObj) myObj->decRefs(); }

SObjFunc CObjRef::operator() ( int fName) const
    { return SObjFunc ( myObj, fName); }
SObjFunc CObjRef::operator() ( const CString& fName) const
    { return SObjFunc ( myObj, LinkerDataTypes::Linker->registerName (fName)); }
```

```
SObjFunc CObjRef::operator>> ( int fName) const
    { return SObjFunc ( myObj, fName); }
SObjFunc CObjRef::operator>> ( const CString& fName) const
    { return SObjFunc ( myObj, LinkerDataTypes::Linker->registerName (fName)); }
//CObjRef& CObjRef::operator-> ()
//  { return *this; }

bool CObjRef::operator== (const CObjRef& ref) const
    { return myObj == ref.rObj(); }

//-------------------------------------------------------------------------------------------------

Func::Func() :
    myName (0),
    mySelectorInVM (0),
    myClassName (0),
    myClassSelectorInVM (0),
    myVM (NULL)
    { }

Func::Func ( IVM* vm, int name, int className, const CParamsTypes& types) :
    myName (name),
    mySelectorInVM (0),
    myClassName (className),
    myClassSelectorInVM (0),
    myTypes (types),
    myVM (vm)
    { if (!vm) return;
      myClassSelectorInVM = vm->getClassSelector (className);
      mySelectorInVM = vm->getFuncSelector(className, name); }

bool Func::ok() const
    { return (myName || mySelectorInVM) && (myClassName || myClassSelectorInVM) && myVM; }

SParam Func::operator() (CObj* obj, const CParams& params) const
    { Assert ( ok() && "Errorness function call!\n");
      return myVM->call (obj, myClassName, myClassSelectorInVM, myName, mySelectorInVM, params); }

bool Func::operator== (const Func& func) const
    { return myName == func.myName && mySelectorInVM == func.mySelectorInVM &&
             myClassName == func.myClassName && myClassSelectorInVM == func.myClassSelectorInVM && myVM ==
func.myVM; }

//-------------------------------------------------------------------------------------------------

StaticFunc::StaticFunc() :
    myName (0),
    mySelectorInVM (0),
    myClassName (0),
    myClassSelectorInVM (0),
    myVM (NULL)
    { }

StaticFunc::StaticFunc ( IVM* vm, int name, int className, const CParamsTypes& types) :
    myName (name),
    mySelectorInVM (0),
    myClassName (className),
    myClassSelectorInVM (0),
    myTypes (types),
    myVM (vm)
    { Assert (vm);
      myClassSelectorInVM = vm->getClassSelector (className);
      mySelectorInVM = vm->getStaticFuncSelector(className, name); }

bool StaticFunc::ok() const
    { return (myName || mySelectorInVM) && (myClassName || myClassSelectorInVM) && myVM; }

SParam StaticFunc::operator() (const CParams& params) const
    { Assert (ok() && "Errorness function call!\n");
      return myVM->staticCall (myClassName, myClassSelectorInVM, myName, mySelectorInVM, params); }

bool StaticFunc::operator== (const StaticFunc& func) const
    { return myName == func.myName && mySelectorInVM == func.mySelectorInVM &&
             myClassName == func.myClassName && myClassSelectorInVM == func.myClassSelectorInVM && myVM ==
func.myVM; }

//-------------------------------------------------------------------------------------------------
/*
ImportFunc::ImportFunc (int clName, int fName) :
    myFunc ( LinkerDataTypes::Linker->rFunc(fName, clName))
    { }

int ImportFunc::call (CObj* Obj, ...)
    {
    Assert (this);
    Assert (myFunc);
    Assert (Obj);
    Assert (Obj->rClass().operator !());
    Assert (Obj->rClass()->myClassName == myFunc->myClassName);
    if (myFunc->myVM == CPPVM)
        {
        pObjFunc funcPtr = CPPVM->rFuncPtr (myFunc);
        //vasfed
        return 0;
        }
```

```
    return (*myFunc) (Obj, CParams()); //*** vasfed must translate parameters to CParams. CParamsTypes are
in myFunc;
    }
*/
//-----------------------------------------------------------------------------------------------------

SVTable::SVTable() :
    functions (1)
    { functions.add (Func()); }

SVTable& SVTable::operator= (const SVTable& t)
    { functions = t.functions; return *this; }

bool SVTable::operator== (const SVTable& table) const
    { return functions == table.functions; }

//-----------------------------------------------------------------------------------------------------

SClassRef::SClassRef ( SClass* clPtr) :
    mySelectorInLinker ((int) LinkerDataTypes::Linker->getClassRef(clPtr))
    { }
SClassRef::SClassRef (int selInL) :
    mySelectorInLinker (selInL)
    { }
SClassRef::operator SClass*()
//  { return LinkerDataTypes::Linker->getClassPtr (mySelectorInLinker); }
    { if (! mySelectorInLinker) return NULL;
      Assert (0 < mySelectorInLinker && mySelectorInLinker < LinkerDataTypes::Linker-
>getClasses().rNumber());
      return LinkerDataTypes::Linker->getClasses()[mySelectorInLinker];  }

SClassRef::operator bool() const
    { return mySelectorInLinker > 0; }

SClassRef::operator int() const
    { return mySelectorInLinker; }

bool SClassRef::operator!() const
    { return !mySelectorInLinker; }

SClass* SClassRef::operator->() const
    { return LinkerDataTypes::Linker->getClasses()[mySelectorInLinker]; }

bool SClassRef::operator== (const SClassRef& cl) const
    { return mySelectorInLinker == cl.mySelectorInLinker; }

//-----------------------------------------------------------------------------------------------------

SClass::SClass (int clName) :
    myClassName (clName),
    myStaticFuncs (1),
    myParents (1)
    { myStaticFuncs.add (StaticFunc());
      myParents.add (0); }

SClass::~SClass()
    { }

bool SClass::operator== (const SClass& cl) const
    { return myClassName == cl.myClassName; }

SParam SClass::call (int expression, const CParams& par)
    {
    Assert (expression);

    StaticFunc* fPtr = findStaticFunc (expression);
    if ( !fPtr) {
        Trace ( "Linker: Tryin' to call unexistin' static func '%s' of class '%s'",
                LinkerDataTypes::Linker->getName ( expression),
                LinkerDataTypes::Linker->getName ( myClassName));
        return false;
        }

    return (*fPtr) (par);
    }

int SClass::getFuncSelector ( int name)
    {
    Assert (0 < name);

    for (int i = 1; i < myVTable.functions.rNumber(); i++)
        if (myVTable.functions[i].myName == name)
            return i;
    return 0;
    }

int SClass::getStaticFuncSelector ( int name)
    {
    Assert (0 < name);

    for (int i = 1; i < myStaticFuncs.rNumber(); i++)
        if (myStaticFuncs[i].myName == name)
            return i;
    return 0;
    }

Func* SClass::findFunc ( int name, const CParams& params)
```

```
    {
    Assert ( 0 < name);

    int funcs [MaxOnenamedFuncsNumber];
    int num = 0;

    for ( int i = 1; i < myVTable.functions.rNumber(); i++)
        if ( myVTable.functions[i].myName == name)
            { Assert ( num < MaxOnenamedFuncsNumber);
            funcs[num++] = i;
            }

    if ( num == 1)
        return &myVTable.functions[funcs[0]];

    if ( num > 1)
        {
        CParamsTypes p (params);
        int minDif = 1000;
        int bestIndex = -1;
        int bestIndex2 = -1;
        for ( i = 0; i < num; i++)
            {
            int dif = p.conformsTo ( myVTable.functions[funcs[i]].myTypes);
            if ( dif < minDif)
                bestIndex = i;
            else if ( dif == minDif)
                bestIndex2 = i;
            }
        if ( bestIndex == bestIndex2)
            throw ( AmbiguousFunctions ( SClassRef (this), funcs[bestIndex], funcs[bestIndex2]));
        return & myVTable.functions[funcs[bestIndex]];
        }

    for ( i = 1; i < myParents.rNumber(); i++)
        {
        Func* func = myParents[i]->findFunc (name);
        if ( func) return func;
        }

    return NULL;
    }
StaticFunc* SClass::findStaticFunc ( int name, const CParams& params)
    {
    Assert (0 < name);

    int funcs [MaxOnenamedFuncsNumber];
    int num = 0;

    for ( int i = 1; i < myStaticFuncs.rNumber(); i++)
        if ( myStaticFuncs[i].myName == name)
            { Assert ( num < MaxOnenamedFuncsNumber);
            funcs[num++] = i;
            }

    if ( num == 1)
        return &myStaticFuncs[funcs[0]];

    if ( num > 1)
        {
        CParamsTypes p (params);
        int minDif = 1000;
        int bestIndex = -1;
        int bestIndex2 = -1;
        for ( i = 0; i < num; i++)
            {
            int dif = p.conformsTo ( myStaticFuncs[funcs[i]].myTypes);
            if ( dif < minDif)
                bestIndex = i;
            else if ( dif == minDif)
                bestIndex2 = i;
            }
        if ( bestIndex == bestIndex2)
            throw ( AmbiguousStaticFunctions ( SClassRef (this), funcs[bestIndex], funcs[bestIndex2]));
        return & myStaticFuncs[funcs[bestIndex]];
        }

    for (i = 1; i < myParents.rNumber(); i++)
        {
        StaticFunc* func = myParents[i]->findStaticFunc (name);
        if (func) return func;
        }

    return NULL;
    }
SClass& SClass::registerParent (SClassRef cl)
    {
    Assert (cl); //!!! redo
    if (myParents.search (cl) != -1) return *this;
    Assert ( !cl->checkParent (this));

    myParents.add (cl);
    return *this;
    }
```

```
bool SClass::checkParent (SClassRef cl)
    {
    Assert (cl);
    if ((SClass*)cl == this) return 1;

    for (int i = 1; i < myParents.rNumber(); i++)
        {
        bool r = myParents[i]->checkParent (cl);
        if (r) return 1;
        }
    return 0;
    }

bool SClass::conformsTo ( SClassRef cl)
    {
    return checkParent ( cl);
    }

Func* SClass::registerFunc (const Func& func)
    {
    Assert (func.myClassName == myClassName);

    for (int i = 1; i < myVTable.functions.rNumber(); i++)
        if (myVTable.functions[i].myName == func.myName)
            {
            myVTable.functions[i] = func;
            return & myVTable.functions[i];
            }

    SHOW ( printf ("Function \"%s\" is registered to class \"%s\"\n",
                   LinkerDataTypes::Linker->getName ( func.myName).rString(),
                   LinkerDataTypes::Linker->getName ( myClassName).rString()),
           4);

    myVTable.functions.add (func);
    return & myVTable.functions[ myVTable.functions.rNumber() - 1];
    }

StaticFunc* SClass::registerStaticFunc (const StaticFunc& func)
    {
    Assert (func.myClassName == myClassName);

    for (int i = 1; i < myStaticFuncs.rNumber(); i++)
        if (myStaticFuncs[i].myName == func.myName)
            {
            myStaticFuncs[i] = func;
            return & myStaticFuncs[i];
            }

    myStaticFuncs.add (func);
    SHOW ( printf ("Static function \"%s\" is registered to class \"%s\"\n",
                   LinkerDataTypes::Linker->getName ( func.myName).rString(),
                   LinkerDataTypes::Linker->getName ( myClassName).rString()),
           4);

    return & myStaticFuncs[ myStaticFuncs.rNumber() - 1];
    }

//-------------------------------------------------------------------------------------------------------

CVariable::CVariable (SParam p, const CString& str) :
    myObj ( CObjRef ((CObj*) p) )  ,
    myName ( LinkerDataTypes::Linker->registerName (str))
    { }

CVariable::CVariable (const CObjRef& Obj, const CString& str) :
    myObj (Obj),
    myName ( LinkerDataTypes::Linker->registerName ( str))
    { }

CVariable::CVariable (const CObjRef& Obj, int name) :
    myObj (Obj),
    myName (name)
    { }

CVariable::CVariable (int name) :
    myName (name)
    { }

const CVariable& CVariable::operator= (const CVariable& var)
    { myObj = var.rObj();
      myName = var.rName();
      return *this; }

CVariable::operator CObjRef&()
    { return myObj; }

const CObjRef& CVariable::rObj() const
    { return myObj; }
CObjRef& CVariable::gObj()
    { return myObj; }

int CVariable::rName() const
    { return myName; }

SObjFunc CVariable::operator() ( const CString& fName) const
    { return myObj (fName); }
```

```
bool CVariable::operator== (const CVariable& var) const
    {
        return myObj == var.rObj();
    }
bool CVariable::operator== (CObjRef& obj) const
    {
        return myObj == obj;
    }
bool CVariable::operator== (const CString& str) const
    {
        return myName? myName == LinkerDataTypes::Linker->checkName (str) : 0;
    }
bool CVariable::operator== (int str) const
    {
        return myName? myName == str : 0;
    }

CVariable& CVariable::assign (CVariable& obj)
    { myObj = obj.rObj(); return *this; }

CVariable& CVariable::operator= (CObjRef& Obj)
    { myObj = Obj; return *this; }

CVariable& CVariable::operator= (const CString& str)
    { myName = LinkerDataTypes::Linker->registerName (str); return *this; }

CVariable& CVariable::operator= (int str)
    { myName = str; return *this; }

//-----------------------------------------------------------------------------------------------------

CScope::CScope ( CScope* parent, int num) :
    myVars ( num + 1),
    myParent ( parent)
    { myVars.add ( CVar()); }

CVar& CScope::gVar ( int name)
    { if ( !name) return myVars[0];
      for ( int i = 1; i < myVars.rNumber(); i++)
        if ( myVars[i] == name) return myVars[i];
      if ( myParent) return myParent->gVar ( name);
      Trace ( "tryin' to get an unexisting variable of scope '%s'", LinkerDataTypes::Linker->getName (
name) .c_str());
      return myVars[0]; }

CVar& CScope::operator() ( const CString& str)
    { return gVar ( LinkerDataTypes::Linker->registerName ( str)); }

CVar& CScope::operator() ( int name)
    { return gVar ( name); }

bool CScope::check ( const CString& name)
    { return check ( LinkerDataTypes::Linker->registerName ( name)); }

bool CScope::check ( int name)
    {
    for ( int i = 1; i < myVars.rNumber(); i++)
        if ( myVars[i].rName() == name) return 1;
    if ( myParent) return myParent->check ( name);
    return 0;
    }


CScope& CScope::addVar (const CVar& var)
    {
    for (int i = 1; i < myVars.rNumber(); i++)
        if (myVars[i].rName() == var.rName())
            { myVars[i] = var; return *this; }
    myVars.add (var);
    return *this;
    }

CScope& CScope::delVar (int name)
    {
    for (int i = 1; i < myVars.rNumber(); i++)
        if (myVars[i] == name)
            {
            myVars.del(i);
            break;
            }
    return *this;
    }

CScope& CScope::setParent ( CScope* scope)
    {
    myParent = scope;
    return *this;
    }

//-----------------------------------------------------------------------------------------------------
```

Файл LinkerDataTypes\defs.cpp

```
#define __PROGRAMMER "Se"
#define __MODULE_NAME "Linker DataTypes static-link"

//#include "defs.h"
#include "stdafx.h"

#include "..\Linker\linker.h"

int TabSize = 4;
```

Файл LinkerDataTypes\small.cpp

```
#include "stdafx.h"

CPosition::CPosition (CString fname, int pos, int strnum, int symbnum) :
    myFileName     (fname),
    myPosition     (pos),
    myStringNumber (strnum),
    mySymbolNumber (symbnum)
    { }

CPosition::CPosition() :
    myFileName     (0),
    myPosition     (-1),
    myStringNumber (-1),
    mySymbolNumber (-1)
    { }

bool CPosition::operator == ( const CPosition& position)
    {
        return
            myFileName == position.rFileName() &&
            myPosition == position.rPos () &&
            myStringNumber == position.rStr() &&
            mySymbolNumber == position.rSymb();
    }

//---------------------------------------------------------------------------------------------------------
```

Файл LinkerDataTypes\system.cpp

```
#define __PROGRAMMER "Se"
#define __MODULE_NAME "Linker DataTypes static-link"

#include "stdafx.h"
#include "system.h"
#include "..\Linker\linker.h"

#include "..\Console\Console.h"

CScope* LinkerDataTypes::GlobalScope = 0;
ILinker* LinkerDataTypes::Linker = 0;
ILoader* LinkerDataTypes::Loader = 0;

void SetShowDepth (IConsole* c, int ln, Lexem1* data)
{
    if (ln == 1)
        if (data[0].type == Lexem1::LT_Number)
            {
            int d = data[0].idata;
            if ( d >= 0)
                {
                c->printf ("Show depth was set from %d to %d.\n", ShowDepth, d);
                ShowDepth = d;
                }
            else
                c->printf ("Depth can't be less then zero. ignored.\n");
            return;
            }
    c->printf ( "Usage: SetShowDepth <level>\n");
}

void LinkerDataTypesLibInit (ILoader* loader)
{
    LinkerDataTypes::Loader = loader;
    Assert (loader);

    LinkerDataTypes::Linker = (ILinker*)LinkerDataTypes::Loader->GetInterface (IID_ILinker);
    Assert ( LinkerDataTypes::Linker);
```

```
    LinkerDataTypes::GlobalScope = &( LinkerDataTypes::Linker->getGlobalScope());
    Assert ( LinkerDataTypes::GlobalScope);

    IConsole* con = 0;
    loader->GetInterface (IID_IConsole, (void**)&con);
    //if (con)
    //  try {
    //  con->AddCommand ( "SetShowDepth", (CommandProcP)SetShowDepth, true);
    //  } Catch()}
}
```

Файл LinkerDataTypes\utilities.cpp

```
#define __PROGRAMMER "Se"
#define __MODULE_NAME "Linker DataTypes static-link"

#include "stdafx.h"

#include "..\Linker\linker.h"

int ShowDepth = 10;

ExpressionPosition::ExpressionPosition ( const char* fileName, int lineNumber) throw() :
    myFileName ( fileName),
    myLineNumber ( lineNumber)
    { }

CException::CException (const char* fileName, int line, const char* comments) throw() :
    myPos ( fileName, line),
    myComments ( comments)
    { }

CException::CException ( const char* comments, const ExpressionPosition& pos) throw() :
    myPos ( pos),
    myComments ( comments)
    { }

void CException::printf (FILE* file) const throw()
    {
    fprintf (file, "Exception: ");
    if ( rFileName()) fprintf (file, "file: \"%s\" ", rFileName());
    if ( rLineNumber() > 0) fprintf (file, "line: %d\n", rLineNumber());
    if (myComments) fprintf (file, "comments: \"%s\" is false.\n", myComments);
    }

CLetters::CLetters() :
    myChars (NULL),
    myLetters (NULL),
    myLettersNumber (0)
    { }
CLetters::CLetters (const char* chars) :
    myChars (chars),
    myLetters (NULL),
    myLettersNumber (0)
    { }
CLetters::CLetters (const char* chars, const CLetters& letters1) :
    myChars (chars),
    myLetters (0),
    myLettersNumber (0)
    { add (letters1); }
CLetters::CLetters (const char* chars, const CLetters& letters1, const CLetters& letters2) :
    myChars (chars),
    myLetters (0),
    myLettersNumber (0)
    { add (letters1); add (letters2); }
CLetters::CLetters (const char* chars, const CLetters& letters1, const CLetters& letters2, const CLetters&
letters3) :
    myChars (chars),
    myLetters (0),
    myLettersNumber (0)
    { add (letters1); add (letters2); add (letters3); }
CLetters::CLetters (const CLetters& letters) :
    myChars (letters.rChars()),
    myLetters (new CLetters [letters.rLettersNumber()]),
    myLettersNumber (letters.rLettersNumber())
    { for (int i = 0; i < myLettersNumber; i++)
        myLetters[i] = letters.rLetters()[i]; }
CLetters& CLetters::operator= (const CLetters& let)
    { myChars = let.rChars();
      myLetters = new CLetters [ let.rLettersNumber()];
      for (int i = 0; i < let.rLettersNumber(); i++)
        myLetters[i] = let.rLetters()[i];
      myLettersNumber = let.rLettersNumber();
      return *this; }
CLetters::~CLetters()
    { if (myLetters) delete[] myLetters; }

const char*&  CLetters::getChars()       { return myChars; }
const char*     CLetters::rChars() const { return myChars; }
CLetters*     CLetters::getLetters()     { return myLetters; }
const CLetters* CLetters::rLetters() const { return myLetters; }
```

```
int& CLetters::getLettersNumber()       { return myLettersNumber; }
int    CLetters::rLettersNumber() const { return myLettersNumber; }

bool CLetters::ok() const { return 0 <= myLettersNumber && myLettersNumber? !!myLetters : 1; }

bool CLetters::includes (char c)
    {
    Assert (CLetters::ok());

    if (!c) return 0;
    if (myChars && strchr (myChars, c)) return 1;
    for (int i = 0; i < myLettersNumber; i++)
        if (myLetters[i].includes (c)) return 1;
    return 0;
    }

CLetters& CLetters::add (const CLetters& letters)
    {
    ASSERTPARAM (&letters != this && "CLetters::add is called to add is own object!");
    Assert (CLetters::ok());
    incLetters();
    myLetters [myLettersNumber - 1] = letters;
    return *this;
    }

int CLetters::incLetters (int n)
    {
    ASSERTPARAM (n >= 0);
    Assert (CLetters::ok());

    CLetters* t = new CLetters [myLettersNumber + n];
    for (int i = 0; i < myLettersNumber; i++)
        t[i] = myLetters[i];
    delete[] myLetters; //!!! !!! here is a gluk
    myLetters = t;
    myLettersNumber += n;
    return myLettersNumber;
    }

inline bool Includes (const char* str, char x)
    {
    ASSERTPARAM (str);
    for (int i = 0; str[i] != '\0'; i++)
        if (str[i] == x) return 1;
    return 0;
    }

CString::CString() : myString (NULL) { }
CString::CString (const char* str) :
    myString ( new char [ strlen (str) + 1])
    {
//  Assert (myString);  //no need; new throws...
    strcpy (myString, str);
    }
    CString::CString ( std::string str) :
        myString ( 0)
        { *this = CString ( str.c_str()); }
/*
CString::CString (char ch) :
    myString (new char [2])
    { myString[0] = ch; myString[1] = '\0'; }
*/
CString::CString ( int integer) :
    myString ( new char [1])
    {
    ASSERTPARAM ( integer == 0 && "CString ( integer)");
    myString[0] = '\0';
    }

CString::~CString()
    {
    if ( myString) delete[] myString;
    }

CString::CString ( const CString& str) :
    myString ( new char [ str.rLength() + 1])
    { strcpy ( myString, str.rString()); }

CString& CString::operator= (const CString& str)
    { if (myString) delete[] myString;
      myString = new char [ str.rLength() + 1];
      strcpy ( myString, str.rString());
      return *this; }

CString& CString::operator << ( const CString& str)
    {
        if ( !str) return *this;
        if ( !myString)
            return (*this) = str;
        char* tmp = new char [ rLength() + str.rLength() + 1 ];
        strcpy ( tmp, myString);
        strcpy ( tmp + strlen ( myString), str.rString());
        delete[] myString;
        myString = tmp;
        return *this;
    }
```

```
CString::operator int() const
    { return LinkerDataTypes::Linker->registerName (*this); }

bool CString::operator== (const CString& str) const
    { return !strcmp ( myString, str.rString()); }

const char* CString::rString() const { return myString; }
char*& CString::getString() { return myString; }
size_t CString::rLength() const { return strlen (myString); }
```

Файл Loader\Commands.cpp

```
// Commands.cpp - implementation of Commands.h, of module Loader
// created 05.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Loader"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Commands.h"

#include <Exeption.h>      // use ExeptionEngine

int LoaderInitModules   (IConsole*c, int, void*)
{
    c->printf ("Initializing uninitialized modules...\n");
    try {
    Loader->InitModules ();
    c->printf ("Completed.\n");
    }
    Catch ()}catch (...){c->printf ("Failture.\n");}
    return true;
}

int LoaderReInitModules   (IConsole*c, int, void*)
{
    c->printf ("Reinitializing modules...\n");
    try {
    Loader->InitModules ();
    c->printf ("Completed.\n");
    }
    Catch ()}catch (...){c->printf ("Failture.\n");}
    return true;
}

int LoaderLoadModule   (IConsole* c, int lc, Lexem1* data)
{
    try {
    AssertT (lc && data, "Dll name not specified");
    Loader->LoadComponent (data[0].data);
    //c->printf ("Module '%s' sucessfully loaded\n", data[0].data);
    }
    Catch()
        c->printf ("LOADER: Cannot load '%s': \n\t %s\n", lc? data[0].data: "(nothing)", exeption.m_text);
    }
    catch (...)
        {
        c->printf ("LOADER: Cannot load '%s': unknown error\n", data[0].data);
        }
    return true;
}

void ListComponents (IConsole* c, int, char*)
{
    c->printf ("Listing Loader Modules:\n");

    int n = 0;
    for (int i = 0; i < Loader->m_size; i++)
        if (Loader->m_data[i].info)
            {
            n++;
            c->printf ("\t[%d]: %s by %s (library '%s')\n",
                i,
                Loader->m_data[i].info->name,
                Loader->m_data[i].info->programmers,
                Loader->m_data[i].name);
            }
    c->printf ("%d modules total.\n", n);
}

void InitConsole (IConsole* c)
{
    //Assert (c);

    try {c->AddCommand ("load", (CommandProcP) LoaderLoadModule, true, "Loader");}
        catch (...){}
    try {c->AddCommand ("init", (CommandProcP) LoaderInitModules, false, "Loader");}
        catch (...){}
    try {c->AddCommand ("reinit", (CommandProcP) LoaderReInitModules, false, "Loader");}
```

```
        catch (...){}
    try {c->AddCommand ("list", (CommandProcP) ListComponents, false, "Loader");}
        catch (...){}

}
```

Файл Loader\LoaderCore.cpp

```
// LoaderCore.cpp - implementation of LoaderCore.h
// created 09.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Loader"

#include "StdAfx.h"      // use precompiled header
#include "LoaderCore.h"
#include <windows.h>
#include <Exeption.h>      // use ExeptionEngine
#include "Commands.h"

#pragma warning (disable:4005)
//#define Message()
#pragma warning (disable:4002)

#include "..\Console\Console.h"

IConsole* Console = NULL;
CLoader* Loader = NULL;

//
//TODO:
//  [_]DllUpdateReferences(ILoader) - Modules alredy contain it
// UnLoad ("DllName");

//#define CONMSG
const int   VersionH = 1,
            VersionL = 5;

const bool AutoloadConsole = true;
#define Con if (Console) Console

CLoader::CLoader   (bool)
    {
    Loader = this;
    m_initialized = false;
    m_data = NULL;
    m_size = NULL;
    m_refs = NULL;
    Init (false);
    Trace ("LOADER: Loader version %d.%d by Vasfed. Created successfully.", VersionH, VersionL);
    }
//---------------------------------------------------------------------------
CLoader::~CLoader  ()
    {
    if (m_initialized)
        {
        Trace ("LOADER: Was not terminated by main program. \n\tPossible crash. Now trying to
terminate.");
        #ifdef CLAIMERRORS
        MessageBox (NULL, "Loader was not terminated by main programm and now is being destructed by
Windows.\n\r"
                          "Now Loader will try to unload modules, but there may fatal errors.\n\r"
                          "Click OK and not be afraid of messges like \"Abnormal program termination\"",
                          "Loader module by Vasfed version 1.5", MB_OK | MB_ICONSTOP);
        #endif
        Terminate ();
        }
    }
//---------------------------------------------------------------------------
void CLoader::Init (bool fullInit)
{
    if (m_initialized)
        {
        CoInitialize (NULL);
        InitModules ();
        GetInterface (IID_IConsole, (void**)&Console);

        if (Console)
            InitConsole (Console);

        Con->printf ("LOADER: Loader version %d.%d by Vasfed: init ok.\n", VersionH, VersionL);
        return;
        }

    Assert (!m_initialized);

    GrowTo (10);
    m_initialized = true;
}
//---------------------------------------------------------------------------
void CLoader::InitModules  ()
```

```
{
    for (int i = 0; i < m_size; i++)
        if (m_data [i].module)
            if (!m_data[i].initialized)
                {
                DLLINIT dllInit = (DLLINIT) GetProcAddress (m_data[i].module, "DllInit");
                if (!dllInit) dllInit = (DLLINIT) GetProcAddress (m_data[i].module, "_DllInit");

                if (dllInit)
                    try
                    {
                    dllInit (static_cast<ILoader*>(this));
                    }
                    catch (...)
                        {
                        Con->printf (10, "LOADER: Module %d (%s) error: Error in DllInit\n", i,
m_data[i].name);
                        Trace ("LOADER: Module %d (%s) error: Error in DllInit\n", i, m_data[i].name);
                        }
                m_data [i].initialized  = true;
                }
}
//---------------------------------------------------------------------------
void  __cdecl CLoader::ReInitModules ()
{
    for (int i = 0; i < m_size; i++)
        if (m_data [i].module)
            //if (!m_data[i].initialized)
                {
                DLLTERMINATE dllTerm = (DLLTERMINATE) GetProcAddress (m_data[i].module, "DllTerminate");
                    if (!dllTerm) dllTerm = (DLLTERMINATE) GetProcAddress (m_data[i].module,
"_DllTerminate");

                    if (dllTerm)
                        try
                            {
                            dllTerm ();
                            }
                        catch (...)
                            {
                            try {
                                Trace ("LOADER: Module %d (%s) error: Error in DllTerminate\n", i,
m_data[i].name);
                                printf ("LOADER: Module %d (%s) error: Error in DllTerminate\n", i,
m_data[i].name);
                                }
                            catch (...){}
                            }
                DLLINIT dllInit = (DLLINIT) GetProcAddress (m_data[i].module, "DllInit");
                if (!dllInit) dllInit = (DLLINIT) GetProcAddress (m_data[i].module, "_DllInit");

                if (dllInit)
                    try
                    {
                    dllInit (static_cast<ILoader*>(this));
                    }
                    catch (...)
                        {
                        Con->printf (10, "LOADER: Module %d (%s) error: Error in DllInit\n", i,
m_data[i].name);
                        Trace ("LOADER: Module %d (%s) error: Error in DllInit\n", i, m_data[i].name);
                        }
                m_data [i].initialized  = true;
                }
}
//---------------------------------------------------------------------------
void CLoader::LoadComponent                (HMODULE dll)
{
    Assert (this);
    AssertT (m_initialized, "Using uninitialized Loader");
    AssertT (dll, "Invalid dll handler\n");

    DLLGETINFO dllGetInfo = (DLLGETINFO) GetProcAddress (dll, "DllGetInfo");
    //if (!dllGetInfo) Throw (runtime, "Cannot load '%s', invalid dll", dllname));

    DllInfo* dllInfo = NULL;
    if (dllGetInfo) dllGetInfo ((void**)&dllInfo);

    DLLGETINSTANCE dllGetInstance = (DLLGETINSTANCE) GetProcAddress (dll, "DllGetInstance");
    if (!dllGetInfo) Throw (runtime, "Cannot load module by handle, invalid module"));
    IUnknown* iunknown = NULL;
    dllGetInstance ((void**)&iunknown);
    if (!iunknown) Throw (runtime, "Cannot load module by handle, cannot get dll interface"));

    int i = 0;
    m_data [i = SearchBlank()].iunknown = iunknown;
    m_data [i].info = dllInfo;
    m_data [i].module = dll;
    m_data [i].initialized = false;
}
//---------------------------------------------------------------------------
void CLoader::LoadComponent                (char* dllname)
{
    Assert (this);
    AssertT (m_initialized, "Using uninitialized Loader");
    AssertC (dllname, badParams);
```

```
    Con->printf ("LOADER: Loading component '%s'\n", dllname);
    HMODULE dll = LoadLibrary (dllname);

    if (!dll)
        {
        char msg[256] = "Nothing";
        int e = GetLastError();
        if (e)
        FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS, NULL,
            e, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), (LPTSTR) msg, 256, NULL);

        Throw (runtime, "Cannot load '%s', Windows said: %s", dllname, msg));
        }

    DLLONLOAD dllOnLoad = (DLLONLOAD) GetProcAddress (dll, "DllOnLoad");
    if (!dllOnLoad) dllOnLoad = (DLLONLOAD) GetProcAddress (dll, "_DllOnLoad");

    if (dllOnLoad)
        {
        //Con->printf ("LOADER: Preinitializing module...\n");
        try
            {
            dllOnLoad (static_cast <ILoader*> (this));
            }
        catch (...)
            {
            Throw (runtime, "Cannot load '%s': Errors inside module (in DllOnLoad)", dllname));
            }
        }

    DLLGETINFO dllGetInfo = (DLLGETINFO) GetProcAddress (dll, "DllGetInfo");
    if (!dllGetInfo) dllGetInfo = (DLLGETINFO) GetProcAddress (dll, "_DllGetInfo");
    DllInfo* dllInfo = NULL;
    if (dllGetInfo)
        try {dllGetInfo ((void**)&dllInfo);
        }catch (...)
            {
            Throw (runtime, "Cannot load '%s': Errors inside module (in GetDllInfo)", dllname));
            }


    DLLGETINSTANCE dllGetInstance = (DLLGETINSTANCE) GetProcAddress (dll, "DllGetInstance");
    if (!dllGetInstance) dllGetInstance = (DLLGETINSTANCE) GetProcAddress (dll, "_DllGetInstance");
    if (!dllGetInstance) Throw ("Invalid Dll"));

    IUnknown* iunknown = NULL;
    try {
        dllGetInstance ((void**)&iunknown);
        }
    catch (...)
            {
            Throw (runtime, "Cannot load '%s': Errors inside module (in GetInstance)", dllname));
            }

    //if (!iunknown) Throw (runtime, "Cannot load '%s', cannot get dll interface", dllname));


    int i = 0;
    m_data [i = SearchBlank()].iunknown = iunknown;
    m_data [i].info = dllInfo;
    m_data [i].module = dll;
    m_data [i].initialized = false;
    strcpy (m_data[i].name, dllname); //make shortname!

    Con->printf ("LOADER: Module loaded successfully.\n");
    Trace ("LOADER: Loaded module: '%s': %s by %s", dllname, dllInfo->dllname, dllInfo->programmers);
}
//-----------------------------------------------------------------------------
void CLoader::LoadForeignComponent    (const GUID* clsid, DllInfo alias){Throw ("Not implemented yet"));}
//-----------------------------------------------------------------------------
const DllInfo* CLoader::GetDllInfo    (int component)
{
    Assert (component >= 0 && component < m_size);
    return m_data[component].info;
}
//-----------------------------------------------------------------------------
int CLoader::GetMaxComponentNum       (){return m_size - 1;}
//-----------------------------------------------------------------------------
IUnknown* CLoader::GetInterface (const IID& iid)
{
    Assert (m_data && m_initialized);

    #ifdef CONMSG
    Con->printf ("LOADER: GetInterface (IID) called\n");
    #endif

    try {
    IUnknown* inter = NULL;
    for (int i = 0; i < m_size; i++)
        if (m_data [i].iunknown)
            {
            m_data [i].iunknown->QueryInterface (iid, (void**)&inter);
            if (inter) throw (inter);
            }
        }
    catch (IUnknown* inter)
        {
```

```
        return inter;
        }
    return NULL;
}
//-----------------------------------------------------------------------------
IUnknown* CLoader::GetInterface (const IID& iid, int module)
{
    #ifdef CONMSG
    Con->printf ("LOADER: GetInterface (IID, int) called\n");
    #endif

    Trace ("LOADER: Call to half-implemented and untested procedure " __FUNCDNAME__ ".");

    Assert (m_data && m_initialized);
    AssertC (module > 0 && module < m_size, badParams);

    IUnknown* inter = NULL;
    if (m_data [module].iunknown)
        m_data [module].iunknown->QueryInterface (iid, (void**)&inter);
    return inter;
}
//-----------------------------------------------------------------------------
void CLoader::GetInterface (const IID& iid, void** ppr)
{
    Assert (m_data && m_initialized && ppr);

    #ifdef CONMSG
    Con->printf ("LOADER: GetInterface (IID, void**) called\n");
    #endif

    try {
        IUnknown* inter = NULL;
        for (int i = 0; i < m_size; i++)
            if (m_data [i].iunknown)
                {
                m_data [i].iunknown->QueryInterface (iid, (void**)&inter);
                if (inter) throw (inter);
                }
        }
    catch (IUnknown* inter)
        {
        *ppr = inter;
        return;
        }
    catch (...)
        {
        *ppr = 0;
        return;
        }
    *ppr = NULL;
    return;
}
//-----------------------------------------------------------------------------
void CLoader::GetInterface (const IID& iid, int module, void** ppr)
{

    //#ifdef CONMSG
    Con->printf ("LOADER: GetInterface (IID, int, void**) called\n");
    //#endif

    Trace ("LOADER: Call to unimplemented procedure " __FUNCDNAME__ ".");

    if (ppr)
        *ppr = NULL;
}
//-----------------------------------------------------------------------------
int CLoader::SearchComponent   (char* reserved, int reserved2){Throw("Not implemented"));}
//-----------------------------------------------------------------------------
void CLoader::UnloadComponent  (int reserved){Throw ("Not implemented"));}
//-----------------------------------------------------------------------------
void CLoader::UnloadUnused     (){Throw ("Not implemented"));}
//-----------------------------------------------------------------------------
void CLoader::Terminate        ()
    {
    try {Console->printf ("LOADER: Terminating system..\n");}catch (...){}


    Trace ("LOADER: Unloading modules");

    for (int i = 0; i < m_size; i++)
        if (m_data [i].module)
                {
                try {
                    DLLTERMINATE dllTerm = (DLLTERMINATE) GetProcAddress (m_data[i].module,
"DllTerminate");
                    if (!dllTerm) dllTerm = (DLLTERMINATE) GetProcAddress (m_data[i].module,
"_DllTerminate");

                    if (dllTerm)
                        try
                            {
                            dllTerm ();
                            }
                        catch (...)
                            {
                            try {
```

```
                                Trace ("LOADER: Module %d (%s) error: Error in DllTerminate\n", i,
m_data[i].name);
                                printf ("LOADER: Module %d (%s) error: Error in DllTerminate\n", i,
m_data[i].name);
                                }
                            catch (...){}
                            }

                    FreeLibrary (m_data[i].module);
                    m_data[i].module = NULL;
                    m_data[i].info = NULL;
                    m_data[i].initialized = false;
                    m_data[i].iunknown = NULL;
                    m_data[i].name[0] = '\0';
                    }
                    catch (...)
                        {
                        char mn[128] = "<Cannot get info>";
                        try {strcpy (mn, m_data[i].info->name);}
                        catch (...){}
                        Trace ("LOADER: Cannot terminate module '%s'", mn);
                        }
                }
    CoUninitialize ();
    printf ("LOADER: Terminated.\n");
    Trace ("LOADER: Terminated correctly");
    m_initialized = false;
    }
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

ULONG STDMETHODCALLTYPE CLoader::AddRef ()
    {
    return InterlockedIncrement (&m_refs);
    }
//---------------------------------------------------------------------------

ULONG STDMETHODCALLTYPE CLoader::Release ()
    {
    if (InterlockedDecrement (&m_refs) == 0)
        {
        delete this;
        return 0;
        }
    return m_refs;
    }
//---------------------------------------------------------------------------
HRESULT STDMETHODCALLTYPE CLoader::QueryInterface (const IID &iid, void ** ptr)
    {
    if (iid == IID_IUnknown)
        {
        *ptr = static_cast <IUnknown*> (this);
        }
    else
        {
        if (iid == IID_ILoader)
            {
            *ptr = static_cast <ILoader*> (this);
            }
        else
            {
            *ptr = NULL;
            return E_NOINTERFACE;
            }
        }
    reinterpret_cast <IUnknown *> (* ptr) -> AddRef ();
    return S_OK;
    }
//---------------------------------------------------------------------------
void CLoader::Grow     ()
{
    GrowFor (m_size / 2 > 1? m_size /2: 2);
}
//---------------------------------------------------------------------------
void CLoader::GrowFor  (int size)
{
    GrowTo (m_size + size);
}
//---------------------------------------------------------------------------
void CLoader::GrowTo   (int size)
{
    DllTechInfo* temp = new DllTechInfo[size];
    ZeroMemory (temp, sizeof (DllTechInfo)*size);

    if (m_data && m_size)
        {
        for (int i = 0; i < m_size; i++)
            temp [i] = m_data [i];
        delete m_data;
        }

    m_data = temp;
    m_size = size;

}
//---------------------------------------------------------------------------
int CLoader::SearchBlank ()
```

```
{
    for (int i = 0; i < m_size; i++)
        if (!m_data[i].module)
            return i;
    int s = m_size;
    Grow ();
    return s;
}
//---------------------------------------------------------------------------
void CLoader::Reset ()
    {Throw ("Not implemented"));}
```

Файл Loader\loader.cpp

```
// LoaderCore.cpp - implementation of Dll functions
// created 09.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "LoaderCore"

#include "StdAfx.h"      // use precompiled header
#include "LoaderCore.h"
#include <Exeption.h>      // use ExeptionEngine

/*  #define DLL_CLASS CLoader
    #define DLL_NAME "Loader"
    #include "..\Common\Utilities\Module.c"*/

DllInfo DLL_INFO =
    {
    "Loader",
    "Loader.dll",
    "Vasfed",
    false,
    false,
    false,
    false,
    false
    };

BOOL APIENTRY DllMain(HANDLE, DWORD, LPVOID lpReserved)
{
    return TRUE;
}

CLoader LoaderObject;

extern "C" __declspec (dllexport) void __cdecl DllGetInfo     (void** ppr)
    {if (ppr) *ppr = &DLL_INFO;}

extern "C" __declspec (dllexport) void __cdecl DllGetInstance (void** ppr)
    {if (ppr) {LoaderObject.AddRef(); *ppr = &LoaderObject;}}
```

Файл LRenederer\LRendererCore.cpp

```
#include "LRendererCore.h"
#include "system.h"

SClassRef CLinkedRenderer::sClass = SClassRef ( 0);

CLinkedRenderer::CLinkedRenderer()
    { myClass = sClass; }

bool CLinkedRenderer::selfReg()
    {
        sClass = LR::CPPVM->registerClass ("LinkedAgent", "LinkedAgent");

        return 1;
    }

IMPLEMENT_IUNKNOWN (CLinkedRenderer, ILRenderer, IID_ILRenderer);
```

Файл LRenederer\Module.cpp

```
// Module.cpp - implementation of Module.h, of module LAgents
// created 03.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "LAgents"
```

```
#include "StdAfx.h"      // use precompiled header
#include "System.h"
//#include "Module.h"

#include "LRendererCore.h"

#include <Exeption.h>      // use ExeptionEngine

#include "..\Common\Utilities\Module.h"

    #define DLL_NAME "LRenderer"
    #define DLL_CLASS CLinkedRenderer
    #include "..\Common\Utilities\Module.c"

DLL_INIT (loader)
    {
        Assert (loader);

        LinkerDataTypesLibInit ( loader);

        loader->GetInterface ( IID_IRenderer, (void**)&LR::Renderer);
        Assert ( LR::Renderer);
        loader->GetInterface ( IID_ILinker, (void**)&LR::Linker);
        Assert ( LR::Linker);
        loader->GetInterface ( IID_ICPPVM, (void**)&LR::CPPVM);
        Assert ( LR::CPPVM);

        Assert ( CLinkedRenderer::selfReg());

        LR::Linker->getGlobalScope().addVar ( CVar ( CObjRef ( LR::Self), "Renderer"));
    }

DLL_ONLOAD (loader)
{
    IConsole* con = NULL;
    loader->GetInterface (IID_IConsole, (void**)&con);
    Assert (con);
    con->printf ("LASERVER: Agents Server Linker interface by Se version 1.0\n");
}
```

Файл LRenederer\system.cpp

```
#include "system.h"

ICPPVM*                LR::CPPVM = 0;
ILinker*               LR::Linker = 0;
IRenderer*             LR::Renderer = 0;
CLinkedRenderer*       LR::Self = 0;
```

Файл Map\Console.cpp

```
// Console.cpp - implementation of Console.h
// created 22.09.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Console"

#include "StdAfx.h"      // use precompiled header
#include "Console.h"
#include "System.h"
#include <Exeption.h>      // use ExeptionEngine

int LoadMap (IConsole* c, int nl, Lexem1* data)
{
    Assert (Map);
    try {
        if (!nl || !data)
            {
            return c->printf ("Usage: loadmap <filename.bmp>");
            }
        Map->LoadBmp (data[0].data);
        }
    catch (...)
        {
        c->printf ("Loading Failed.\n");
        }
    return true;
}

//-----------------------------------------------------------------------------
int Command1 (IConsole* c, int nl, Lexem1* data)
{

    return true;
}

//-----------------------------------------------------------------------------
/*int Command (IConsole* c, int nl, Lexem* data)
```

```
{

    return true;
}
*/
//-------------------------------------------------------------------------------------

//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------

void ConfigureConsole ()
{
    Console->AddCommand ("loadmap", (CommandProcP)LoadMap, true);
    //Console->AddCommand ("", (CommandProcP), true);
//Console->AddCommand ("", (CommandProcP), true);
}
```

Файл Map\DataTypes.cpp

```
// DataTypes.cpp - implementation of DataTypes.h
// created 29.10.2002 by header creator

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "AgentsMapDataTypes"

#include "StdAfx.h"      // use precompiled header
#include "DataTypes.h"

SMapUnit::SMapUnit (int c, int ds, int* d)
{
    code = c;
    datasize = ds;
    data = d;
}

//-------------------------------------------------------------------------------------

SMapUnit::~SMapUnit()
{
    if (data) delete data;
}
```

Файл Map\MapCore.cpp

```
// MapCore.cpp - implementation of MapCore.h, of module Map
// created 29.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Map"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "MapCore.h"
#include <stdio.h>
#include <io.h>
#include <fcntl.h>
#include <stdarg.h>
#include <Exeption.h>      // use ExeptionEngine

#pragma warning (disable:4244)
#pragma warning (disable:4309)
#pragma warning (disable:4267)

IMPLEMENT_IUNKNOWN (CAgentsMap, IAgentsMap, IID_IAgentsMap);

CAgentsMap::CAgentsMap (void)
{
    m_map = NULL;
    m_sizex = 0;
    m_sizey = 0;
    m_ssize = 0;
    m_sizexh = 0;
    m_sizeyh = 0;
    m_errorflag = 0;
}
//-------------------------------------------------------------------------------------
CAgentsMap::~CAgentsMap(void)
    { Reset (true);}
//-------------------------------------------------------------------------------------
void    CAgentsMap::Init   (void)
{
    if (m_map) return;
    m_map = NULL;
    m_sizex = 0;
    m_sizey = 0;
```

```
    m_ssize = 0;
    m_sizexh = 0;
    m_sizeyh = 0;
    m_errorflag = 0;
}
//---------------------------------------------------------------------------
void    CAgentsMap::Reset  (bool confirm)
{
    if (!confirm) return;
    if (m_map) delete[] m_map;
    m_map = NULL;
    m_sizex = 0;
    m_sizey = 0;
    m_ssize = 0;
    m_sizexh = 0;
    m_sizeyh = 0;
    m_errorflag = 0;
}
//---------------------------------------------------------------------------
int     CAgentsMap::Load   (FILE* file)
{
    AssertTC (file, badParams, "No file");
    AssertTC (!ferror (file), badParams, "File with an error");

    setmode (fileno (file), O_BINARY);

    Reset (true);
    try
        {
        SMapFileHeader header;
        ZeroMemory (&header, sizeof (header));
        fread (&header, sizeof (header), 1, file);

        char* sig = "MAPF";
        AssertT (header.signature == *((int*)sig), "File is not Map file.");

        m_sizex = header.sizeX;
        m_sizey = header.sizeY;
        m_sizexh = m_sizex / 2;
        m_sizeyh = m_sizey / 2;
        m_ssize = header.sizeS > 0? header.sizeS: 1.0F;

        fseek (file, header.offset, SEEK_CUR);

        m_map = new SMapUnit [m_sizex * m_sizey + 1];
        AssertT (m_map, "Cannot allocate memory");

        for (int y = 0; y < m_sizey; y++)
            for (int x = 0; x < m_sizex; x++)
                {
                SMapUnitInFile unit;
                fread (&unit, sizeof (unit), 1, file);
                Assert (!ferror (file));

                m_map[x + m_sizex*y].code = unit.code;
                m_map[x + m_sizex*y].datasize = unit.datasize;
                m_map[x + m_sizex*y].data = NULL;

                if (unit.datasize)
                    {
                    m_map[x + m_sizex*y].data = new int [unit.datasize];
                    fread (m_map[x + m_sizex*y].data, sizeof (int), unit.datasize, file);
                    Assert (!ferror (file));
                    }
                }
            fprintf (file, "\nEnd of Map\n");
        }
    catch(...)
        {
        //loading failed, cleanup
        Reset (true);
        return false;
        }
    return true;
}
//---------------------------------------------------------------------------
int     CAgentsMap::Load   (char* filename)
{
    FILE* file = fopen (filename, "rb");
    if (!file) return false;
    int rv = Load (file);
    fclose (file);
    return rv;
}
//---------------------------------------------------------------------------
int     CAgentsMap::LoadBmp(FILE* file)
{
    AssertTC (file, badParams, "No file");
    AssertTC (!ferror (file), badParams, "File with an error");

    setmode (fileno (file), O_BINARY);

    BITMAPFILEHEADER bfh;
    fread (&bfh, sizeof (bfh), 1, file);
    AssertT (!ferror (file), "Failed to read Header of BMP");

    BITMAPINFOHEADER bih;
```

```
    fread (&bih, sizeof (bih), 1, file);
    AssertT (!ferror (file), "Failed to read Info header of BMP")

    if (bih.biBitCount != 24)
        //if (bih.biBitCount != 8)
            Throw ("LoadBMP: File must be 24 bits RGB"));

    fseek (file, bfh.bfOffBits + sizeof (RGBQUAD) * bih.biClrUsed/*+ sizeof (BITMAPFILEHEADER)*/,
SEEK_SET);
    //54
    if (bih.biBitCount == 24) //loading RGB
        {
        if (m_map) delete[] m_map;
        m_sizex = bih.biWidth;
        m_sizey = bih.biHeight;
        m_sizexh = m_sizex / 2;
        m_sizeyh = m_sizey / 2;
        m_ssize = 1.0;
        int m_sizemap = m_sizex * m_sizey + 1;

        m_map = new SMapUnit [m_sizemap];
        Assert (m_map);
        ZeroMemory (m_map, m_sizemap*sizeof(SMapUnit));


        for (int y = bih.biHeight - 1; y >= 0; y--)
            for (int x = 0; x < bih.biWidth; x++)
            {
            char bu[3] = {0};
            if (!fread (bu, 3, 1, file))
                Message ("!fread");
            AssertT (!ferror (file), "Failed to read 3 bytes from BMP stream");

            int buffer = 0;
            char r = bu [0], g = bu[1], b = bu[2];
            buffer = (b&0xff) << 16 | (g&0xff) << 8 | r&0xff;
            //now rgb value is in 'buffer'
            switch (buffer)
                {
                case 0xffffff: m_map[y * m_sizex + x].code = EMC_Empty;       break;
                case 0x006496: m_map[y * m_sizex + x].code = EMC_Road;        break;
                case 0x646464: m_map[y * m_sizex + x].code = EMC_Rock;        break;
                case 0x00ff00: m_map[y * m_sizex + x].code = EMC_Field;       break;
                case 0x0000ff: m_map[y * m_sizex + x].code = EMC_Water;       break;
                case 0xff80ff: m_map[y * m_sizex + x].code = EMC_House;       break;
                case 0xff0000: m_map[y * m_sizex + x].code = EMC_Castle;  break;
                case 0x008000: m_map[y * m_sizex + x].code = EMC_Forest;  break;
                default:       m_map[y * m_sizex + x].code = EMC_Empty;       break;
                }
            m_map[y * m_sizex + x].datasize = 1;
            m_map[y * m_sizex + x].data = new int[m_map[y * m_sizex + x].datasize];
            m_map[y * m_sizex + x].data [0] = 0;
            }
        }
    return true;
}
//---------------------------------------------------------------------------------------
int     CAgentsMap::LoadBmp (char* filename)
{
    FILE* file = fopen (filename, "rbS");
    if (!file) return false;
    int rv = LoadBmp (file);
    fclose (file);
    return rv;
}
//---------------------------------------------------------------------------------------
int     CAgentsMap::Save   (FILE* file)
{
    try
        {
        SMapFileHeader header;
        ZeroMemory (&header, sizeof (header));

        char* sig = "MAPF";

        header.signature = *((int*)sig);
        header.sizeX = m_sizex;
        header.sizeY = m_sizey;
        header.sizeS = m_ssize;

        char* garbage = "Map file";
        header.offset = strlen (garbage);
        fwrite (&header, sizeof (header), 1, file);
        fwrite (garbage, header.offset, 1, file);

        for (int y = 0; y < m_sizey; y++)
            for (int x = 0; x < m_sizex; x++)
                {
                SMapUnitInFile unit;

                unit.code = m_map[x + m_sizex*y].code;
                unit.datasize = m_map[x + m_sizex*y].data? m_map[x + m_sizex*y].datasize: NULL;

                fwrite (&unit, sizeof (unit), 1, file);

                if (unit.datasize)
                    fwrite (m_map[x + m_sizex*y].data, sizeof (int), unit.datasize, file);
```

```
                }
            fprintf (file, "\nEnd of Map\n");
        }
    catch(...)
        {
        return false;
        }
    return true;
}
//-------------------------------------------------------------------------------
int     CAgentsMap::SaveBmp (FILE* file)
{

    try {
        AssertTC (file, badParams, "No file");
        AssertTC (!ferror (file), badParams, "File with an error");

        setmode (fileno (file), O_BINARY);

        BITMAPFILEHEADER bfh;
        ZeroMemory (&bfh, sizeof (bfh));
        bfh.bfType = 'B' | 'M' << 8;
        bfh.bfSize = sizeof (bfh);
        fwrite (&bfh, sizeof (bfh), 1, file);
        AssertT (!ferror (file), "Failed to write Header of BMP");

        BITMAPINFOHEADER bih;
        ZeroMemory (&bih, sizeof (bih));
        bih.biSize = sizeof (bih);
        bih.biBitCount = 24;
        bih.biHeight = m_sizey;
        bih.biWidth = m_sizex;
        bih.biPlanes = 1;
        fwrite (&bih, sizeof (bih), 1, file);
        AssertT (!ferror (file), "Failed to write Info header of BMP")

        //m_ssize = 1.0;

        for (int y = bih.biHeight - 1; y >= 0; y--)
            for (int x = 0; x < bih.biWidth; x++)
                {
                int c = 0;
                char rgb[3];

                switch (m_map[y * m_sizex + x].code)
                    {
                    case EMC_Empty:     rgb [0] = 0xff; rgb [1] = 0xff; rgb [2] = 0xff; break;
                    case EMC_Road:      rgb [0] = 0x00; rgb [1] = 0x64; rgb [2] = 0x96; break;
                    case EMC_Rock:      rgb [0] = 0x64; rgb [1] = 0x64; rgb [2] = 0x64; break;
                    case EMC_Field:     rgb [0] = 0x00; rgb [1] = 0xff; rgb [2] = 0x00;   break;
                    case EMC_Water:     rgb [0] = 0x00; rgb [1] = 0x00; rgb [2] = 0xff; break;
                    case EMC_House:     rgb [0] = 0xff; rgb [1] = 0x80; rgb [2] = 0xff; break;
                    case EMC_Castle:    rgb [0] = 0xff; rgb [1] = 0x00; rgb [2] = 0x00;   break;
                    case EMC_Forest:    rgb [0] = 0x00; rgb [1] = 0x80; rgb [2] = 0x00;   break;
                    default:            rgb [0] = 0x00; rgb [1] = 0x00; rgb [2] = 0x00;   break;
                    }

                fwrite (rgb, 1, 3, file);

                AssertT (!ferror (file), "Failed to write BMP data");
                }
            }
    catch (...)
        {return false;}
    return true;
}
//-------------------------------------------------------------------------------
int     CAgentsMap::Save   (char* filename)
{
    FILE* file = fopen (filename, "wb");
    if (!file) return false;
    int rv = Save (file);
    fclose (file);
    return rv;
}
//-------------------------------------------------------------------------------
int     CAgentsMap::SaveBmp (char* filename)
{
    FILE* file = fopen (filename, "wb");
    if (!file) return false;
    int rv = SaveBmp (file);
    fclose (file);
    return rv;
}
//-------------------------------------------------------------------------------
int     CAgentsMap::Ok     (void)
{
    return (m_map != NULL && m_sizex && m_sizey && (m_ssize > 0));
}
//-------------------------------------------------------------------------------
int     CAgentsMap::GetUnitCode    (float xx, float yy) const
{
    if (!m_map) return NULL;

    int x = (int)((xx / m_ssize) + m_sizexh);
    int y = (int)((yy / m_ssize) + m_sizeyh);
```

```
    if (x  >= m_sizex) return NULL;
    if (y  >= m_sizey) return NULL;
    if (x  <= 0) return NULL;
    if (y  <= 0) return NULL;

    return m_map [x + m_sizex * y].code;
}
//-------------------------------------------------------------------------------------
int*    CAgentsMap::GetPUnitData   (float xx, float yy) const
{
    if (!m_map) return NULL;

    int x = (int)((xx / m_ssize) + m_sizexh);
    int y = (int)((yy / m_ssize) + m_sizeyh);

    if (x  >= m_sizex) return NULL;
    if (y  >= m_sizey) return NULL;
    if (x  <= 0) return NULL;
    if (y  <= 0) return NULL;

    return m_map [x + m_sizex * y].data;
}
//-------------------------------------------------------------------------------------
int     CAgentsMap::GetUnitData    (float xx, float yy, int index) const
{
    if (!m_map) return NULL;

    int x = (int)((xx / m_ssize) + m_sizexh);
    int y = (int)((yy / m_ssize) + m_sizeyh);

    if (x  >= m_sizex) return NULL;
    if (y  >= m_sizey) return NULL;
    if (x  <= 0) return NULL;
    if (y  <= 0) return NULL;

    if (index >= m_map [x + m_sizex * y].datasize || index < 0) return NULL;
    return  m_map [x + m_sizex * y].data[index];
}
//-------------------------------------------------------------------------------------
int     CAgentsMap::GetUnitDataSize    (float xx, float yy) const
{
    if (!m_map) return NULL;

    int x = (int)((xx / m_ssize) + m_sizexh);
    int y = (int)((yy / m_ssize) + m_sizeyh);

    if (x  >= m_sizex) return NULL;
    if (y  >= m_sizey) return NULL;
    if (x  <= 0) return NULL;
    if (y  <= 0) return NULL;

    return m_map [x + m_sizex * y].datasize;
}
//-------------------------------------------------------------------------------------
void    CAgentsMap::ModifyUnitData (float xx, float yy, int index, int newvalue)
{
    if (!m_map) return;

    int x = (int)((xx / m_ssize) + m_sizexh);
    int y = (int)((yy / m_ssize) + m_sizeyh);

    if (x  >= m_sizex) return;
    if (y  >= m_sizey) return;
    if (x  <= 0) return;
    if (y  <= 0) return;

    if (index < 0 || index >= m_map [x + m_sizex * y].datasize)
        return;
    m_map [x + m_sizex * y].data[index] = newvalue;
}
//-------------------------------------------------------------------------------------
int     CAgentsMap::GetSizeX       (void)  const
    { return m_sizex;}
//-------------------------------------------------------------------------------------
int     CAgentsMap::GetSizeY       (void)  const
    { return m_sizey;}
//-------------------------------------------------------------------------------------
float   CAgentsMap::GetSquareSize  (void)  const
    { return m_ssize;}
//-------------------------------------------------------------------------------------
void    CAgentsMap::SetSquareSize  (float newsize)
    { m_ssize = newsize;}

//-------------------------------------------------------------------------------------
int     CAgentsMap::GetUnitCode    (int x, int y) const
{
    if (!m_map) return NULL;

    if (x  >= m_sizex) return NULL;
    if (y  >= m_sizey) return NULL;
    if (x  <= 0) return NULL;
    if (y  <= 0) return NULL;

    return m_map [x + m_sizex * y].code;
}
//-------------------------------------------------------------------------------------
int*    CAgentsMap::GetPUnitData   (int x, int y) const
```

```
{
    if (!m_map) return NULL;

    if (x  >= m_sizex) return NULL;
    if (y  >= m_sizey) return NULL;
    if (x  <= 0) return NULL;
    if (y  <= 0) return NULL;

    return m_map [x + m_sizex * y].data;
}
//-----------------------------------------------------------------------------
int     CAgentsMap::GetUnitData    (int x, int y, int index) const
{
    if (!m_map) return NULL;

    if (x  >= m_sizex) return NULL;
    if (y  >= m_sizey) return NULL;
    if (x  <= 0) return NULL;
    if (y  <= 0) return NULL;

    if (index >= m_map [x + m_sizex * y].datasize || index < 0) return NULL;
    return  m_map [x + m_sizex * y].data[index];
}
//-----------------------------------------------------------------------------
int     CAgentsMap::GetUnitDataSize    (int x, int y) const
{
    if (!m_map) return NULL;

    if (x  >= m_sizex) return NULL;
    if (y  >= m_sizey) return NULL;
    if (x  <= 0) return NULL;
    if (y  <= 0) return NULL;

    return m_map [x + m_sizex * y].datasize;
}
//-----------------------------------------------------------------------------
void    CAgentsMap::ModifyUnitData (int x, int y, int index, int newvalue)
{
    if (!m_map) return;

    if (x  >= m_sizex) return;
    if (y  >= m_sizey) return;
    if (x  <= 0) return;
    if (y  <= 0) return;

    if (index < 0 || index >= m_map [x + m_sizex * y].datasize)
        return;
    m_map [x + m_sizex * y].data[index] = newvalue;
}
//-----------------------------------------------------------------------------
```

Файл Map\Module.cpp

```
// Module.cpp - implementation of module Map
// created 29.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Map"

#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine

#include "MapCore.h"
#include "..\Console\Console.h"

ILoader*       Loader = NULL;
IAgentsServer*  Server = NULL;

    #define DLL_NAME "AgentsMap"
    #define DLL_CLASS CAgentsMap
    #include "..\Common\Utilities\Module.c"

DLL_INIT (loader)
{
    Assert (loader);
    Loader = loader;

    //Loader->GetInterface (IID_IAgentsServer, (void**)&Server);
    //Assert (Server);

}

DLL_ONLOAD (loader)
{
    IConsole* con = NULL;
    loader->GetInterface (IID_IConsole, (void**)&con);
    Assert (con);
    con->printf ("MAP: Map by Vasfed version 1.0\n");
}
```

Файл MapConverter\MapConverter.cpp

```
// MapConverter.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <conio.h>
#include "..\Loader\Loader.h"
#include "..\Map\Map.h"

ILoader* Loader = NULL;
IAgentsMap* Map = NULL;

int CreateLoader (char* dllname);

int _tmain(int argc, _TCHAR* argv[])
{
    if (!CreateLoader ("Loader.dll")) return printf ("Cannot create Loader\n");
    try
        {
        Loader->LoadComponent ("Map.dll");
        Loader->GetInterface (IID_IAgentsMap, (void**)&Map);
        if (!Map) return printf ("Cannot create agents Map\n");
        Loader->InitModules ();

        if (argc != 3)
            return printf ("Usage: MapConverter <input bmp file> <output map file>\n");

        Map->LoadBmp (argv[1]);
        if (!Map->Save (argv[2])) printf ("Failed!\n");

        printf ("Ok.\n");
        //getch ();
        }
    catch (...)
        {
        printf ("Error!\n");
        }
    Loader->Terminate ();
    return 0;
}

int CreateLoader (char* dllname)
{
    try
        {
        HMODULE LoaderDll = LoadLibrary (dllname);
        Assert (LoaderDll);
        typedef void (*DLLGETINSTANCE)(void**);
        DLLGETINSTANCE LoaderGetInstance = (DLLGETINSTANCE)GetProcAddress (LoaderDll, "DllGetInstance");
        Assert (LoaderGetInstance);
        LoaderGetInstance ((void**)&Loader);
        //Loader->Init (true);
        }
    catch(...)
        {
        return false;
        }
    return true;
}
```

Файл Memory\memory.cpp

```
#include "memory.h"

#define __PROGRAMMER "Xroft"
#define __MODULE_NAME "Memory"

#include <exeption.h>

BOOL APIENTRY DllMain(HANDLE hModule,
                      DWORD  ul_reason_for_call,
                      LPVOID lpReserved)
{
return 1;
}

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CMEMORYBLOCK ***********************************/
/*****************************************************************************/
/*****************************************************************************/

/*****************************************************************************/

CMemoryBlock::CMemoryBlock ()
    {
    Init ();
```

```
    }
/*************************************************************************/

CMemoryBlock::~CMemoryBlock ()
    {
    delete myNext;
    if (myHandle != NULL)
        {
        LocalFree (myHandle);
        Trace ("Memory block with size %d deleted\n", mySize);
        }
    }
/*************************************************************************/

void CMemoryBlock::Init ()
    {
    myPointer = NULL;
    myHandle = NULL;
    mySize = 0;
    myObjSize = 0;
    myObjNum = 0;
    myCurrentObjNum = 0;
    myNext = NULL;
    }
/*************************************************************************/

long CMemoryBlock::RetObjNum ()
    {
    if (myNext != NULL) return myObjNum + myNext -> RetObjNum ();
    return myObjNum;
    }
/*************************************************************************/

long CMemoryBlock::RetCurObjNum ()
    {
    return myCurrentObjNum;
    }
/*************************************************************************/

int CMemoryBlock::Allocate (int objsize, int objnum)
    {
    if (myHandle == NULL)
        {
        myHandle = LocalAlloc (GMEM_MOVEABLE, objsize * objnum);
        AssertT (myHandle != NULL, "No memory allocated!");
        mySize = objnum * objsize;
        myObjNum = objnum;
        myObjSize = objsize;
        myCurrentObjNum = 0;
        myObjMask = new int [myObjNum];
        for (int i = 0; i < myObjNum; i ++) myObjMask [i] = 0;
        Trace ("Memory block with size %d allocated\n", mySize);
        }
    else
        {
        AssertT (objsize == myObjSize, "Illegal object size");
        if (myNext == NULL) myNext = new CMemoryBlock;
        return myNext -> Allocate (objsize, objnum);
        }

    return mySize;
    }
/*************************************************************************/

void CMemoryBlock::Lock ()
    {
    AssertT (myHandle != NULL, "No memory allocated!");
    if (myPointer == NULL) myPointer = reinterpret_cast <char *> (LocalLock (myHandle));
    if (myNext != NULL) myNext -> Lock ();
    }
/*************************************************************************/

void CMemoryBlock::Unlock ()
    {
    AssertT (myHandle != NULL, "No memory allocated!");
    if (myPointer != NULL) LocalUnlock (myHandle);
    if (myNext != NULL) myNext -> Unlock ();
    myPointer = NULL;
    }
/*************************************************************************/

void CMemoryBlock::Free ()
    {
    AssertT (myHandle != NULL, "No memory allocated!");
    LocalFree (myHandle);
    Init ();
    Trace ("Memory block with size %d freed\n", mySize);
    }
```

```
/*****************************************************************************/

char * CMemoryBlock::RetObject (int n)
    {
    AssertT (myPointer != NULL, "Memory is not locked");
    if (n >= myObjNum)
        {
        if (myNext != NULL) return myNext -> RetObject (n - myObjNum);
        AssertT (0, "Can't return unexisting object");
        }
    else
        {
        if (myObjMask [n] == 1) return myPointer + (n * myObjSize);
        return NULL;
        }
    }

/*****************************************************************************/

int CMemoryBlock::AddObject (void * obj)
    {
    AssertT (myPointer != NULL, "Memory is not locked");
    for (int n = 0; n < myObjNum; n ++)
        {
        if (myObjMask [n] == 0)
            {
            myObjMask [n] = 1;
            memcpy (myPointer + (n * myObjSize),
                    reinterpret_cast <char *> (obj), myObjSize);
            myCurrentObjNum ++;
            return 1;
            }
        }
    if (myNext != NULL)
        {
        myCurrentObjNum ++;
        return myNext -> AddObject (obj);
        }
    return 0;
    }

/*****************************************************************************/

void CMemoryBlock::RemoveObject (int n)
    {
    if (n >= myObjNum)
        {
        if (myNext != NULL) myNext -> RemoveObject (n - myObjNum);
        else AssertT (0, "Can't remove unexisting object");
        }
    else
        {
        myObjMask [n] = 0;
        }
    myCurrentObjNum --;
    }

/*****************************************************************************/

char * CMemoryBlock::CreateObject ()
    {
    AssertT (myPointer != NULL, "Memory is not locked");
    for (int n = 0; n < myObjNum; n ++)
        {
        if (myObjMask [n] == 0)
            {
            myObjMask [n] = 1;
            myCurrentObjNum ++;
            return myPointer + (n * myObjSize);
            }
        }
    if (myNext != NULL)
        {
        myCurrentObjNum ++;
        return myNext -> CreateObject ();
        }
    return NULL;
    }

/*****************************************************************************/

char * CMemoryBlock::CreateObjects (int n)
    {
    char * objects = CreateObject ();
    for (int i = 1; i < n; i ++)
        {
        if (CreateObject () == NULL) break;
        }
    return objects;
    }

/*****************************************************************************/

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CSMARTPOINTER **********************************/
/*****************************************************************************/
```

```
/*************************************************************************/

/*************************************************************************/

CSmartPointer::CSmartPointer ()
    {
    myMemoryBlock = NULL;
    myObjectOffset = 0;
    }

/*************************************************************************/

CSmartPointer::CSmartPointer (CMemoryBlock * block, int offset)
    {
    myMemoryBlock = block;
    myObjectOffset = offset;
    }

/*************************************************************************/

CSmartPointer::~CSmartPointer ()
    {
    }

/*************************************************************************/

int CSmartPointer::New (int objsize, int objnum)
    {
    myObjectOffset = 0;
    myMemoryBlock = new CMemoryBlock;
    return myMemoryBlock -> Allocate (objsize, objnum);
    }

/*************************************************************************/

void CSmartPointer::Delete ()
    {
    myObjectOffset = 0;
    if (myMemoryBlock != NULL) myMemoryBlock -> Free ();
    myMemoryBlock = NULL;
    }

/*************************************************************************/

int CSmartPointer::Grow (int objsize, int objnum)
    {
    if (myMemoryBlock != NULL) return myMemoryBlock -> Allocate (objsize, objnum);
    return 0;
    }

/*************************************************************************/

int CSmartPointer::RetObjNum ()
    {
    if (myMemoryBlock != NULL) return myMemoryBlock -> RetObjNum () - myObjectOffset;
    }

/*************************************************************************/

int CSmartPointer::RetCurObjNum ()
    {
    if (myMemoryBlock != NULL) return myMemoryBlock -> RetCurObjNum ();
    }

/*************************************************************************/

void CSmartPointer::operator = (CMemoryBlock &memblock)
    {
    myMemoryBlock = &memblock;
    }

/*************************************************************************/

void CSmartPointer::operator = (CSmartPointer &ptr)
    {
    myMemoryBlock = ptr.myMemoryBlock;
    myObjectOffset = ptr.myObjectOffset;
    }

/*************************************************************************/

void CSmartPointer::operator += (int offset)
    {
    if (myObjectOffset + offset <= myMemoryBlock -> RetObjNum ())
        myObjectOffset += offset;
    }

/*************************************************************************/

void CSmartPointer::operator -= (int offset)
    {
    if (myObjectOffset - offset >= 0)
        myObjectOffset -= offset;
    }

/*************************************************************************/
```

```
void CSmartPointer::operator ++ ()
    {
    if (myObjectOffset + 1 <= myMemoryBlock -> RetObjNum ())
        myObjectOffset ++;
    }

/*************************************************************************/

void CSmartPointer::operator -- ()
    {
    if (myObjectOffset - 1 >= 0)
        myObjectOffset --;
    }

/*************************************************************************/

char * CSmartPointer::RetObject (int n)
    {
    if (myMemoryBlock != NULL) return myMemoryBlock -> RetObject (n + myObjectOffset);
    return NULL;
    }

/*************************************************************************/

int CSmartPointer::AddObject (void * obj)
    {
    if (myMemoryBlock != NULL) return myMemoryBlock -> AddObject (obj);
    return 0;
    }

/*************************************************************************/

void CSmartPointer::RemoveObject (int n)
    {
    if (myMemoryBlock != NULL) myMemoryBlock -> RemoveObject (n + myObjectOffset);
    }

/*************************************************************************/

char * CSmartPointer::CreateObject ()
    {
    if (myMemoryBlock != NULL) return myMemoryBlock -> CreateObject ();
    return NULL;
    }

/*************************************************************************/

char * CSmartPointer::CreateObjects (int n)
    {
    if (myMemoryBlock != NULL) return myMemoryBlock -> CreateObjects (n);
    return NULL;
    }

/*************************************************************************/

void CSmartPointer::Lock ()
    {
    if (myMemoryBlock != NULL) myMemoryBlock -> Lock ();
    }

/*************************************************************************/

void CSmartPointer::Unlock ()
    {
    if (myMemoryBlock != NULL) myMemoryBlock -> Unlock ();
    }

/*************************************************************************/
```

Файл Physics\physics.cpp

```
/*************************************************************************/
/*************************************************************************/
/***                                                                   ***/
/*** PHYSICS.CPP                                                       ***/
/*** ╬╨╚╤╦╗╠╬╨╤╦╗╠╬╨╤╦╗ CENVIRONMENT ╠╨╤╦╗╠╬╨╤╦╗╠╬╨╤╦╗╠╬╨╤╦╗╠╬╨╤╦╗   ***/
/***                                                                   ***/
/*** ╠╠╗╗ ═╠╨╚╤╦╗═ ╗╤╦╗╨╨╚╤╦╗╠╬╨╤╦╗ ═╬╨╤╦╗ ╗╬╨╤╦╗╤╦╗╠╬╨╤╦╗ ╤ 2002 ╠╬╨╗  ***/
/***                                                                   ***/
/***                                                                   ***/
/*************************************************************************/
/*************************************************************************/

#include "physics.h"

#define __PROGRAMMER "Xroft"
#define __MODULE_NAME "Physics"

#include <exeption.h>
```

```
    #define DLL_NAME "Physics"
    #define DLL_CLASS CEnvironment
    #include "..\Common\Utilities\Module.c"

IConsole * Console = NULL;

/*****************************************************************************/

DLL_ONLOAD (loader)
    {
    Assert (loader);
    loader -> GetInterface (IID_IConsole, (void **) &Console);

    if (Console)
        {
        Console -> printf ("PHYSICS: Physics version 0.8 by Xroft.\n");
        }
    }

/*****************************************************************************/

DLL_INIT (loader)
    {
    }

/*****************************************************************************/

//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\// CLASS CPHYSTABLE //\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//

/*****************************************************************************/

CPhysTable::CPhysTable ()
    {
    myMeshes              = NULL;
    mySpeeds              = NULL;
    myAccelerations       = NULL;
    myIntersectionSpheres = NULL;
    myIntersectionSpaces  = NULL;
    myBoundingBoxes       = NULL;
    myDeltas              = NULL;
    myPositions           = NULL;
    myOldPositions        = NULL;
    myMaterials           = NULL;
    myLand                = NULL;
    }

/*****************************************************************************/

CPhysTable::CPhysTable (int size)
    {
    Init (size);
    }

/*****************************************************************************/

CPhysTable::~CPhysTable ()
    {
    delete [] myMeshes;
    delete [] mySpeeds;
    delete [] myAccelerations;
    delete [] myIntersectionSpheres;
    delete [] myIntersectionSpaces;
    delete [] myBoundingBoxes;
    delete [] myDeltas;
    delete [] myPositions;
    delete [] myOldPositions;
    delete [] myMaterials;
    }

/*****************************************************************************/

void CPhysTable::Init (int size)
    {
    mySize = size;
    myCurPos = 0;
    myMeshes              = new CMesh *               [size];
    mySpeeds              = new CSpeed *              [size];
    myAccelerations       = new CAcceleration **      [size];
    myIntersectionSpheres = new CIntersectionSphere * [size];
    myIntersectionSpaces  = new CIntersectionSpace *  [size];
    myBoundingBoxes       = new CBoundingBox *        [size];
    myDeltas              = new CTimeDelta *          [size];
    myPositions           = new CPosition *           [size];
    myOldPositions        = new CPosition *           [size];
    myMaterials           = new CPhysMaterial *       [size];
    NullArray ();
    }

/*****************************************************************************/

void CPhysTable::NullArray ()
    {
    for (int n = 0; n < mySize; n ++)
```

```
        {
        myMeshes             [n] = NULL;
        mySpeeds             [n] = NULL;
        myAccelerations      [n] = NULL;
        myIntersectionSpaces [n] = NULL;
        myBoundingBoxes      [n] = NULL;
        myPositions          [n] = NULL;
        myPositions          [n] = NULL;
        myDeltas             [n] = NULL;
        myMaterials          [n] = NULL;
        }
    myCurPos = 0;
    }

/*************************************************************************/

int CPhysTable::ok ()
    {
    return myCurPos < mySize;
    }

/*************************************************************************/

int CPhysTable::ok (int n)
    {
    return n < myCurPos;
    }

/*************************************************************************/

int CPhysTable::RetCurSize ()
    {
    return myCurPos;
    }

/*************************************************************************/

int CPhysTable::RetMaxSize ()
    {
    return mySize;
    }

/*************************************************************************/

int CPhysTable::SetElement (CMesh * mesh, CSpeed * speed, CAcceleration * acceleration,
                            CIntersectionSphere * sphere, CIntersectionSpace * box,
                            CBoundingBox * box2, CPhysMaterial * mat)
    {
    AssertT (ok (), "No place for new element in Physic Table");
    myMeshes [myCurPos] = mesh;
    myMeshes [myCurPos] -> SetID (myCurPos);
    mySpeeds [myCurPos] = speed;
    myAccelerations [myCurPos] = new CAcceleration * [MAXACCELERATIONS];
    myAccelerations [myCurPos][0] = acceleration;
    for (int i = 1; i < MAXACCELERATIONS; i ++)
        {
        myAccelerations [myCurPos][i] = NULL;
        }
    myIntersectionSpheres [myCurPos] = sphere;
    myIntersectionSpaces [myCurPos] = box;
    myIntersectionSpaces [myCurPos] -> Add (mesh -> RetCenter ());
    box2 -> AddToCorner (mesh -> RetCenter ());
    myBoundingBoxes [myCurPos] = box2;
    myPositions [myCurPos] = new CPosition;
    myPositions [myCurPos] -> Init (mesh->RetCenter ());
    myOldPositions [myCurPos] = new CPosition;
    myOldPositions [myCurPos] -> Init (mesh->RetCenter ());
    myDeltas [myCurPos] = new CTimeDelta;
    myDeltas [myCurPos] -> NullTime ();
    myMaterials [myCurPos] = mat;
    myCurPos ++;
    return myCurPos - 1;
    }

/*************************************************************************/

void CPhysTable::NewAcceleration (int handle, int n)
    {
    AssertT (ok (handle), "Handle is out of range");
    AssertT (myAccelerations [handle] [n] == NULL, "Can't renew already existing acceleration");
    myAccelerations [handle] [n] = new CAcceleration;
    }

/*************************************************************************/

void CPhysTable::SetMesh (int handle, CMesh * mesh)
    {
    AssertT (ok (handle), "Handle is out of range");
    *(myMeshes [handle]) = *(mesh);
    }

/*************************************************************************/

void CPhysTable::SetSpeed (int handle, CSpeed * speed)
    {
    AssertT (ok (handle), "Handle is out of range");
    *(mySpeeds [handle]) = *(speed);
```

```
    }

/*****************************************************************************/

void CPhysTable::SetAcceleration (int handle, CAcceleration * acceleration, int n)
    {
    AssertT (ok (handle), "Handle is out of range");
    if (myAccelerations [handle][n] == NULL) NewAcceleration (handle, n);
    *(myAccelerations [handle][n]) = *(acceleration);
    }

/*****************************************************************************/

void CPhysTable::SetIntersectionSphere (int handle, CIntersectionSphere * sphere)
    {
    AssertT (ok (handle), "Handle is out of range");
    myIntersectionSpheres [handle] = sphere;
    }

/*****************************************************************************/

void CPhysTable::SetIntersectionSpace (int handle, CIntersectionSpace * box)
    {
    AssertT (ok (handle), "Handle is out of range");
    myIntersectionSpaces [handle] = box;
    }

/*****************************************************************************/

void CPhysTable::SetBoundingBox (int handle, CBoundingBox * box)
    {
    AssertT (ok (handle), "Handle is out of range");
    myBoundingBoxes [handle] = box;
    }

/*****************************************************************************/

void CPhysTable::SetTimeDelta (int handle, CTimeDelta * delta)
    {
    AssertT (ok (handle), "Handle is out of range");
    myDeltas [handle] = delta;
    }

/*****************************************************************************/

void CPhysTable::SetPosition (int handle, CPosition * pos)
    {
    AssertT (ok (handle), "Handle is out of range");
    myPositions [handle] = pos;
    }

/*****************************************************************************/

void CPhysTable::SetOldPosition (int handle, CPosition * pos)
    {
    AssertT (ok (handle), "Handle is out of range");
    myOldPositions [handle] = pos;
    }

/*****************************************************************************/

void CPhysTable::SetMaterial (int handle, CPhysMaterial * mat)
    {
    AssertT (ok (handle), "Handle is out of range");
    myMaterials [handle] = mat;
    }

/*****************************************************************************/

void CPhysTable::SetLandscape (CLandscape * land)
    {
    myLand = land;
    }

/*****************************************************************************/

CMesh * CPhysTable::RetMesh (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myMeshes [handle];
    }

/*****************************************************************************/

CSpeed * CPhysTable::RetSpeed (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return mySpeeds [handle];
    }

/*****************************************************************************/

CAcceleration * CPhysTable::RetAcceleration (int handle, int n)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myAccelerations [handle][n];
    }
```

```
/*****************************************************************************/

CIntersectionSphere * CPhysTable::RetIntersectionSphere (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myIntersectionSpheres [handle];
    }

/*****************************************************************************/

CIntersectionSpace * CPhysTable::RetIntersectionSpace (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myIntersectionSpaces [handle];
    }

/*****************************************************************************/

CBoundingBox * CPhysTable::RetBoundingBox (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myBoundingBoxes [handle];
    }

/*****************************************************************************/

CTimeDelta * CPhysTable::RetTimeDelta (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myDeltas [handle];
    }

/*****************************************************************************/

CPosition * CPhysTable::RetPosition (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myPositions [handle];
    }

/*****************************************************************************/

CPosition * CPhysTable::RetOldPosition (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myOldPositions [handle];
    }

/*****************************************************************************/

CPhysMaterial * CPhysTable::RetMaterial (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myMaterials [handle];
    }

/*****************************************************************************/

CLandscape * CPhysTable::RetLandscape ()
    {
    return myLand;
    }

/*****************************************************************************/

void CPhysTable::RetAccelSum (int handle, CAcceleration * sum)
    {
    sum -> Init (0.0F, 0.0F, 0.0F);
    for (int n = 0; n < MAXACCELERATIONS && myAccelerations [handle][n] != NULL; n ++)
        {
        sum -> RetPoint () += myAccelerations [handle][n] -> RetPoint ();
        }
    }

/*****************************************************************************/

void CPhysTable::AddToTime (int handle, float time)
    {
    AssertT (ok (handle), "Handle is out of range");
    myDeltas [handle] -> AddTime (time);
    }

/*****************************************************************************/

void CPhysTable::ChangeMovingStep (int handle)
    {
    RetTimeDelta (handle) -> GetNewTime ();
    RetTimeDelta (handle) -> NullTime ();
    RetSpeed (handle) -> RetPoint (2) = RetSpeed (handle) -> RetPoint (1);
    RetPosition (handle) -> RetPoint () = RetMesh (handle) -> RetCenter ();
    }

/*****************************************************************************/

void CPhysTable::ChangeAcceleration (int handle, int n, float x, float y, float z)
    {
```

```
    if (RetAcceleration (handle, n) == NULL) NewAcceleration (handle, n);
    RetAcceleration (handle, n) -> Init (x, y, z);
    ChangeMovingStep (handle);
    }

/*************************************************************************/

void CPhysTable::AddToAcceleration (int handle, int n, float x, float y, float z)
    {
    if (RetAcceleration (handle, n) == NULL) NewAcceleration (handle, n);
    RetAcceleration (handle, n) -> Add (x, y, z);
    ChangeMovingStep (handle);
    }

/*************************************************************************/

void CPhysTable::ChangeSpeed (int handle, int n, float x, float y, float z)
    {
    RetSpeed (handle) -> Init (n, x, y, z);
    if (n == SIMPLE_SPEED)
        {
        ChangeMovingStep (handle);
        }
    }

/*************************************************************************/

void CPhysTable::AddToSpeed (int handle, int n, float x, float y, float z)
    {
    RetSpeed (handle) -> Add (n, x, y, z);
    if (n == SIMPLE_SPEED)
        {
        ChangeMovingStep (handle);
        }
    }

/*************************************************************************/

//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\// CLASS CENVIRONMENT \\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//

/*************************************************************************/

CEnvironment::CEnvironment () :
    myRefs (0)
    {
    }

/*************************************************************************/

CEnvironment::CEnvironment (int size) :
    myTable (size),
    myRefs (0)
    {
    }

/*************************************************************************/

CEnvironment::~CEnvironment ()
    {
    }

/*************************************************************************/

HRESULT __stdcall CEnvironment::QueryInterface (const IID &iid, void ** ptr)
    {
    if (iid == IID_IUnknown)
        {
        *ptr = static_cast <IEnvironment *> (this);
        }
    else
        {
        if (iid == IID_IEnvironment)
            {
            *ptr = static_cast <IEnvironment *> (this);
            }
        else
            {
            *ptr = NULL;
            return E_NOINTERFACE;
            }
        }
    reinterpret_cast <IUnknown *> (* ptr)->AddRef ();
    return S_OK;
    }

/*************************************************************************/

ULONG __stdcall CEnvironment::AddRef ()
    {
    return InterlockedIncrement (&myRefs);
    }

/*************************************************************************/
```

```
ULONG __stdcall CEnvironment::Release ()
    {
    if (InterlockedDecrement (&myRefs) == 0)
        {
        delete this;
        return 0;
        }
    return myRefs;
    }

/*************************************************************************/

void CEnvironment::SetGravity (float x, float y, float z)
    {
    myGravity.Init (x, y, z);
    }

/*************************************************************************/

CPoint3D & CEnvironment::RetGravity ()
    {
    return myGravity;
    }

/*************************************************************************/

void CEnvironment::SetPhysicLevel (int pl)
    {
    myPhysicLevel = pl;
    if (myPhysicLevel == PL_NO_MOVE) myTimePassed.ZeroTime ();
    if (myPhysicLevel >= PL_MOVE_ONLY)
        {
        for (int n = 0; n < myTable.RetCurSize (); n ++)
            myTable.RetTimeDelta (n) -> ReduceTime (myTimePassed.RetTime ());
        myTimePassed.ZeroTime ();
        }
    }

/*************************************************************************/

void CEnvironment::InitTable (int n)
    {
    myTable.Init (n);
    }

/*************************************************************************/

void CEnvironment::FillTable (CMesh * meshes, CSpeed * speeds, CAcceleration * accels,
                              CIntersectionSphere * spheres, CIntersectionSpace * boxes,
                              CBoundingBox * boxes2, CPhysMaterial * mats,
                              int num, CLandscape * land)
    {
    if (num <= myTable.RetMaxSize ())
        {
        for (int n = 0; n < num; n ++)
            {
            myTable.SetElement (&meshes [n], &speeds [n], &accels [n], &spheres [n], &boxes [n], &boxes2
[n], &mats [n]);
            myTable.SetLandscape (land);
            }
        }
    }

/*************************************************************************/

void CEnvironment::FillTable (CSmartPointer & meshes, CSmartPointer & speeds, CSmartPointer & accels,
                              CSmartPointer & spheres, CSmartPointer & ispaces,
                              CSmartPointer & boxes, CSmartPointer & mats,
                              CSmartPointer & land)
    {
    if (meshes.RetObjNum () <= myTable.RetMaxSize ())
        {
        for (int n = 0; n < meshes.RetObjNum (); n ++)
            {
            myTable.SetElement ((CMesh *) meshes.RetObject (n), (CSpeed *) speeds.RetObject (n),
                                (CAcceleration *) accels.RetObject (n),
                                (CIntersectionSphere *) spheres.RetObject (n),
                                (CIntersectionSpace *) ispaces.RetObject (n),
                                (CBoundingBox *) boxes.RetObject (n), (CPhysMaterial *) mats.RetObject
(n));
            }
        myTable.SetLandscape ((CLandscape *) land.RetObject (0));
        }
    }

/*************************************************************************/

void CEnvironment::MakeTable (CMesh * meshes, CSpeed * speeds, CAcceleration * accels,
                              CIntersectionSphere * spheres, CIntersectionSpace * boxes,
                              CBoundingBox * boxes2, CPhysMaterial * mats,
                              int num, CLandscape * land)
    {
    myTable.Init (num);
    for (int n = 0; n < num; n ++)
        {
```

```
        myTable.SetElement (&meshes [n], &speeds [n], &accels [n], &spheres [n], &boxes [n], &boxes2 [n],
&mats [n]);
        }
    myTable.SetLandscape (land);
    }

/*****************************************************************************/

void CEnvironment::MakeTable (CSmartPointer & meshes, CSmartPointer & speeds, CSmartPointer & accels,
                              CSmartPointer & spheres, CSmartPointer & ispaces,
                              CSmartPointer & boxes, CSmartPointer & mats,
                              CSmartPointer & land)
    {
    myTable.Init (meshes.RetObjNum ());
    for (int n = 0; n < meshes.RetObjNum (); n ++)
        {
        myTable.SetElement ((CMesh *) meshes.RetObject (n), (CSpeed *) speeds.RetObject (n),
                            (CAcceleration *) accels.RetObject (n),
                            (CIntersectionSphere *) spheres.RetObject (n),
                            (CIntersectionSpace *) ispaces.RetObject (n),
                            (CBoundingBox *) boxes.RetObject (n), (CPhysMaterial *) mats.RetObject (n));
        }
    myTable.SetLandscape ((CLandscape *) land.RetObject (0));
    }

/*****************************************************************************/

void CEnvironment::AddElement (CMesh * mesh, CSpeed * speed, CAcceleration * accel,
                               CIntersectionSphere * sphere, CIntersectionSpace * box,
                               CBoundingBox * box2, CPhysMaterial * mat)
    {
    myTable.SetElement (mesh, speed, accel, sphere, box, box2, mat);
    }

/*****************************************************************************/

void * CEnvironment::RetPhysTable ()
    {
    return (void *) &myTable;
    }

/*****************************************************************************/

void CEnvironment::SetMesh (int handle, CMesh * mesh)
    {
    myTable.SetMesh (handle, mesh);
    }

/*****************************************************************************/

void CEnvironment::SetSpeed (int handle, CSpeed * speed)
    {
    myTable.SetSpeed (handle, speed);
    }

/*****************************************************************************/

void CEnvironment::SetAcceleration (int handle, CAcceleration * acceleration, int n)
    {
    myTable.SetAcceleration (handle, acceleration, n);
    }

/*****************************************************************************/

void CEnvironment::SetIntersectionSpace (int handle, CIntersectionSpace * box)
    {
    myTable.SetIntersectionSpace (handle, box);
    }

/*****************************************************************************/

void CEnvironment::SetBoundingBox (int handle, CBoundingBox * box)
    {
    myTable.SetBoundingBox (handle, box);
    }

/*****************************************************************************/

void CEnvironment::SetTimeDelta (int handle, CTimeDelta * delta)
    {
    myTable.SetTimeDelta (handle, delta);
    }

/*****************************************************************************/

void CEnvironment::SetPosition (int handle, CPosition * pos)
    {
    myTable.SetPosition (handle, pos);
    }

/*****************************************************************************/

void CEnvironment::SetLandscape (CLandscape * land)
    {
    myTable.SetLandscape (land);
    }
```

```
/*************************************************************************/

CMesh * CEnvironment::RetMesh (int handle)
    {
    return myTable.RetMesh (handle);
    }

/*************************************************************************/

CSpeed * CEnvironment::RetSpeed (int handle)
    {
    return myTable.RetSpeed (handle);
    }

/*************************************************************************/

CAcceleration * CEnvironment::RetAcceleration (int handle, int n)
    {
    return myTable.RetAcceleration (handle, n);
    }

/*************************************************************************/

CIntersectionSpace * CEnvironment::RetIntersectionSpace (int handle)
    {
    return myTable.RetIntersectionSpace (handle);
    }

/*************************************************************************/

CBoundingBox * CEnvironment::RetBoundingBox (int handle)
    {
    return myTable.RetBoundingBox (handle);
    }

/*************************************************************************/

CTimeDelta * CEnvironment::RetTimeDelta (int handle)
    {
    return myTable.RetTimeDelta (handle);
    }

/*************************************************************************/

CPosition * CEnvironment::RetPosition (int handle)
    {
    return myTable.RetPosition (handle);
    }

/*************************************************************************/

void CEnvironment::TransformBox (int n, CMatrix4x4 &m)
    {
    CBoundingBox * box = myTable.RetBoundingBox (n);
    box -> RetCorner ()  -= myTable.RetMesh (n) -> RetCenter ();
    box -> RetCorner ()  = m.DotProduct (box -> RetCorner ());
    box -> RetCorner ()  += myTable.RetMesh (n) -> RetCenter ();
    box -> RetVector (0) = m.DotProduct (box -> RetVector (0));
    box -> RetVector (1) = m.DotProduct (box -> RetVector (1));
    box -> RetVector (2) = m.DotProduct (box -> RetVector (2));
    }

/*************************************************************************/

int CEnvironment::SphereCheck (int n1, int n2)
    {
    return ((myTable.RetIntersectionSphere (n1) -> RetRadius () + myTable.RetIntersectionSphere (n2) ->
RetRadius ()) >=
            (myTable.RetMesh (n1) -> RetCenter () - myTable.RetMesh (n2) -> RetCenter ()).RetLength ());
    }

/*************************************************************************/

int CEnvironment::IntersectionCheckX (CIntersectionSpace * box1, CIntersectionSpace * box2)
    {
    if (box2->RetPoint () (0) > box2->RetPoint () (0) + box2->RetX ())
        {
        if (box1->RetPoint () (0) <= box2->RetPoint () (0) &&
            box1->RetPoint () (0) >= box2->RetPoint () (0) + box2->RetX ())
            return 1;
        if (box1->RetPoint () (0) + box1->RetX () <= box2->RetPoint () (0) &&
            box1->RetPoint () (0) + box1->RetX () >= box2->RetPoint () (0) + box2->RetX ())
            return 1;
        }
    if (box2->RetPoint () (0) < box2->RetPoint () (0) + box2->RetX ())
        {
        if (box1->RetPoint () (0) >= box2->RetPoint () (0) &&
            box1->RetPoint () (0) <= box2->RetPoint () (0) + box2->RetX ())
            return 1;
        if (box1->RetPoint () (0) + box1->RetX () >= box2->RetPoint () (0) &&
            box1->RetPoint () (0) + box1->RetX () <= box2->RetPoint () (0) + box2->RetX ())
            return 1;
        }
    return 0;
    }

/*************************************************************************/
```

```
int CEnvironment::IntersectionCheckY (CIntersectionSpace * box1, CIntersectionSpace * box2)
    {
    if (box2->RetPoint () (1) > box2->RetPoint () (1) + box2->RetY ())
        {
        if (box1->RetPoint () (1) <= box2->RetPoint () (1) &&
            box1->RetPoint () (1) >= box2->RetPoint () (1) + box2->RetY ())
            return 1;
        if (box1->RetPoint () (1) + box1->RetY () <= box2->RetPoint () (1) &&
            box1->RetPoint () (1) + box1->RetY () >= box2->RetPoint () (1) + box2->RetY ())
            return 1;
        }
    if (box2->RetPoint () (1) < box2->RetPoint () (1) + box2->RetY ())
        {
        if (box1->RetPoint () (1) >= box2->RetPoint () (1) &&
            box1->RetPoint () (1) <= box2->RetPoint () (1) + box2->RetY ())
            return 1;
        if (box1->RetPoint () (1) + box1->RetY () >= box2->RetPoint () (1) &&
            box1->RetPoint () (1) + box1->RetY () <= box2->RetPoint () (1) + box2->RetY ())
            return 1;
        }
    return 0;
    }

/*************************************************************************/

int CEnvironment::IntersectionCheckZ (CIntersectionSpace * box1, CIntersectionSpace * box2)
    {
    if (box2->RetPoint () (2) > box2->RetPoint () (2) + box2->RetZ ())
        {
        if (box1->RetPoint () (2) <= box2->RetPoint () (2) &&
            box1->RetPoint () (2) >= box2->RetPoint () (2) + box2->RetZ ())
            return 1;
        if (box1->RetPoint () (2) + box1->RetZ () <= box2->RetPoint () (2) &&
            box1->RetPoint () (2) + box1->RetZ () >= box2->RetPoint () (2) + box2->RetZ ())
            return 1;
        }
    if (box2->RetPoint () (2) < box2->RetPoint () (2) + box2->RetZ ())
        {
        if (box1->RetPoint () (2) >= box2->RetPoint () (2) &&
            box1->RetPoint () (2) <= box2->RetPoint () (2) + box2->RetZ ())
            return 1;
        if (box1->RetPoint () (2) + box1->RetZ () >= box2->RetPoint () (2) &&
            box1->RetPoint () (2) + box1->RetZ () <= box2->RetPoint () (2) + box2->RetZ ())
            return 1;
        }
    return 0;
    }

/*************************************************************************/

int CEnvironment::IntersectionCheck (int nobj1, int nobj2, CPoint3D &norm)
    {
    CIntersectionSpace * box1 = myTable.RetIntersectionSpace (nobj1),
                       * box2 = myTable.RetIntersectionSpace (nobj2);
    if (IntersectionCheckX (box1, box2) &&
        IntersectionCheckY (box1, box2) &&
        IntersectionCheckZ (box1, box2))
        return 1;
    return 0;
    }

/*************************************************************************/

int CEnvironment::LineAndQuadCheck (CPoint3D & line, CPoint3D & dest,
                                    CPoint3D & quad1, CPoint3D & quad2, CPoint3D & quad3, CPoint3D & quad4)
    {
    float T = 0.0F, D = 0.0F;
    CPoint3D norm = (quad2 - quad1).CrossProduct (quad3 - quad1);
    norm.Normalize ();
    D = - quad1.DotProduct (norm);
    if (norm.DotProduct (dest) == 0.0F) return 0;
    T = - ((line.DotProduct (norm) + D) / norm.DotProduct (dest));
    CPoint3D result = line + dest * T;
    if (T <= 0.0F || T >= 1.0F) return 0;

    float S, S1, S2, S3, S4;
    S = (quad2 - quad1).RetLength () * (quad3 - quad1).RetLength ();

    S1 = 0.5F * sqrt ((quad1 - result).RetLength () * (quad2 - result).RetLength () *
                      (quad1 - result).RetLength () * (quad2 - result).RetLength () -
                      (quad1 - result).DotProduct (quad2 - result) * (quad1 - result).DotProduct (quad2 -
result));
    S2 = 0.5F * sqrt ((quad2 - result).RetLength () * (quad3 - result).RetLength () *
                      (quad2 - result).RetLength () * (quad3 - result).RetLength () -
                      (quad2 - result).DotProduct (quad3 - result) * (quad2 - result).DotProduct (quad3 -
result));
    S3 = 0.5F * sqrt ((quad3 - result).RetLength () * (quad4 - result).RetLength () *
                      (quad3 - result).RetLength () * (quad4 - result).RetLength () -
                      (quad3 - result).DotProduct (quad4 - result) * (quad3 - result).DotProduct (quad4 -
result));
    S4 = 0.5F * sqrt ((quad1 - result).RetLength () * (quad4 - result).RetLength () *
                      (quad1 - result).RetLength () * (quad4 - result).RetLength () -
                      (quad1 - result).DotProduct (quad4 - result) * (quad1 - result).DotProduct (quad4 -
result));
    if ((S1 + S2 + S3 + S4) > S - 0.001F && (S1 + S2 + S3 + S4) < S + 0.001F) return 1;
    else return 0;
```

```
    }
/*************************************************************************/

int CEnvironment::LineAndBoxCheck (CPoint3D & line, CPoint3D & dest, CBoundingBox * bbox)
    {
    CPoint3D corner = bbox -> RetCorner (),
             vec1   = bbox -> RetVector (0),
             vec2   = bbox -> RetVector (1),
             vec3   = bbox -> RetVector (2);

    if (LineAndQuadCheck (line, dest, corner, corner + vec1, corner + vec3, corner + vec1 + vec3) == 1)
        return 1;
    if (LineAndQuadCheck (line, dest, corner, corner + vec1, corner + vec2, corner + vec1 + vec2) == 1)
        return 1;
    if (LineAndQuadCheck (line, dest, corner, corner + vec2, corner + vec3, corner + vec2 + vec3) == 1)
        return 1;
    if (LineAndQuadCheck (line, dest, corner + vec2, corner + vec1 + vec2, corner + vec1 + vec2 + vec3,
corner + vec2 + vec3) == 1)
        return 1;
    if (LineAndQuadCheck (line, dest, corner + vec1, corner + vec1 + vec2, corner + vec1 + vec2, corner +
vec1 + vec2 + vec3) == 1)
        return 1;
    if (LineAndQuadCheck (line, dest, corner + vec3, corner + vec1 + vec3, corner + vec3 + vec2, corner +
vec1 + vec2 + vec3) == 1)
        return 1;
    return 0;
    }

/*************************************************************************/

int CEnvironment::BoundCheck (int nobj1, int nobj2, CPoint3D &norm)
    {
    CBoundingBox * box1   = myTable.RetBoundingBox (nobj1),
                 * box2   = myTable.RetBoundingBox (nobj2);
    CPoint3D       corner = box1 -> RetCorner (),
                   vec1   = box1 -> RetVector (0),
                   vec2   = box1 -> RetVector (1),
                   vec3   = box1 -> RetVector (2);

    if (LineAndBoxCheck (corner,               vec1, box2)) return 1;
    if (LineAndBoxCheck (corner,               vec2, box2)) return 1;
    if (LineAndBoxCheck (corner,               vec3, box2)) return 1;
    if (LineAndBoxCheck (corner + vec1,        vec2, box2)) return 1;
    if (LineAndBoxCheck (corner + vec1,        vec3, box2)) return 1;
    if (LineAndBoxCheck (corner + vec2,        vec1, box2)) return 1;
    if (LineAndBoxCheck (corner + vec2,        vec3, box2)) return 1;
    if (LineAndBoxCheck (corner + vec3,        vec1, box2)) return 1;
    if (LineAndBoxCheck (corner + vec3,        vec2, box2)) return 1;
    if (LineAndBoxCheck (corner + vec1 + vec2, vec3, box2)) return 1;
    if (LineAndBoxCheck (corner + vec1 + vec3, vec2, box2)) return 1;
    if (LineAndBoxCheck (corner + vec2 + vec3, vec1, box2)) return 1;
    return 0;
    }

/*************************************************************************/

int CEnvironment::DetectCollisionBetween (int nobj1, int nobj2, CPoint3D &norm)
    {
    int res = SphereCheck (nobj1, nobj2);
    if (res == 1)
        {
        if (myPhysicLevel >= PL_ISPACE_TEST)
            {
            res = IntersectionCheck (nobj1, nobj2, norm) || IntersectionCheck (nobj2, nobj1, norm);
            if (res && myPhysicLevel == PL_BBOX_TEST)
                {
                return (BoundCheck (nobj1, nobj2, norm) || BoundCheck (nobj2, nobj1, norm));
                }
            else return res;
            }
        else return res;
        }
    else return 0;
    }

/*************************************************************************/

float CEnvironment::RetHorizontHeight (int hornum, float x, float y, CPoint3D &normal)
    {
    CHorizont * hor = myTable.RetLandscape () -> RetPHorizont (hornum);
    int x1 = (int) ((x + (hor -> RetX () / 2) * hor -> RetSquareSize ()) / hor -> RetSquareSize ()),
        y1 = (int) ((y + (hor -> RetY () / 2) * hor -> RetSquareSize ()) / hor -> RetSquareSize ()),
        x2, y2;
    x2 = x1 + 1; y2 = y1 + 1;
    if (x1 < 0 || y1 < 0 || x1 >= hor -> RetX () || y1 >= hor -> RetY () ||
        x2 < 0 || y2 < 0 || x2 >= hor -> RetX () || y2 >= hor -> RetY ()) return -1000.0F;

    CPoint3D p1 (0.0F, 0.0f, 0.0F), p2 (0.0F, 0.0F, 0.0F), p3 (0.0F, 0.0F, 0.0F),
             norm (0.0F, 0.0F, 0.0F), start (0.0F, 0.0F, 0.0F), dest (0.0F, 0.0F, 1.0F),
             result (0.0F, 0.0F, 0.0F);
    float T = 0.0F, D = 0.0F;
    x -= (float) x1 * hor -> RetSquareSize () - (hor -> RetX () / 2) * hor -> RetSquareSize ();
    y -= (float) y1 * hor -> RetSquareSize () - (hor -> RetY () / 2) * hor -> RetSquareSize ();
    start.Init (x, y, 0.0F);
    if (y >= x)
        {
```

```
        p1.Init (0.0F,                    0.0F,                    hor -> RetElement (x1, y1));
        p2.Init (0.0F,                    hor -> RetSquareSize (), hor -> RetElement (x1, y2));
        p3.Init (hor -> RetSquareSize (), hor -> RetSquareSize (), hor -> RetElement (x2, y2));
        }
    if (y < x)
        {
        p1.Init (0.0F,                    0.0F,                    hor -> RetElement (x1, y1));
        p2.Init (hor -> RetSquareSize (), 0.0F,                    hor -> RetElement (x2, y1));
        p3.Init (hor -> RetSquareSize (), hor -> RetSquareSize (), hor -> RetElement (x2, y2));
        }
    norm = (p1 - p2).CrossProduct (p3 - p2);
    norm.Normalize ();
    D = - (p1.DotProduct (norm));
    T = - ((start.DotProduct (norm) + D) / norm.DotProduct (dest));
    result = start + dest * T;
    normal = norm;
    return result (2);
    }

/*************************************************************************/

int CEnvironment::DetectLandCollision (int nobj, CPoint3D &norm)
    {
    CLandscape * land = myTable.RetLandscape ();
    if (land != NULL)
        {
        CBoundingBox * box = myTable.RetBoundingBox (nobj);
        CPoint3D point (0.0F, 0.0F, 0.0F);
        for (int n = 0; n < land -> RetSize (); n ++)
            {
            point = box -> RetCorner ();
            if (RetHorizontHeight (n, point (0), point (1), norm) >= point (2)) return 1;
            point = box -> RetCorner () + box -> RetVector (0);
            if (RetHorizontHeight (n, point (0), point (1), norm) >= point (2)) return 1;
            point = box -> RetCorner () + box -> RetVector (1);
            if (RetHorizontHeight (n, point (0), point (1), norm) >= point (2)) return 1;
            point = box -> RetCorner () + box -> RetVector (2);
            if (RetHorizontHeight (n, point (0), point (1), norm) >= point (2)) return 1;
            point = box -> RetCorner () + box -> RetVector (0) + box -> RetVector (1);
            if (RetHorizontHeight (n, point (0), point (1), norm) >= point (2)) return 1;
            point = box -> RetCorner () + box -> RetVector (0) + box -> RetVector (2);
            if (RetHorizontHeight (n, point (0), point (1), norm) >= point (2)) return 1;
            point = box -> RetCorner () + box -> RetVector (1) + box -> RetVector (2);
            if (RetHorizontHeight (n, point (0), point (1), norm) >= point (2)) return 1;
            point = box -> RetCorner () + box -> RetVector (0) + box -> RetVector (1) + box -> RetVector
(2);
            if (RetHorizontHeight (n, point (0), point (1), norm) >= point (2)) return 1;
            }
        return 0;
        }
    return 0;
    }

/*************************************************************************/

SDCResult &CEnvironment::DetectCollision (int nobj, CPoint3D &norm)
    {
    norm.Init (0.0F, 1.0F, 1.0F);
    norm.Normalize ();
    myLastDCResult.myIsCollisionDetected = false;
    for (int n = nobj + 1; n < myTable.RetCurSize(); n ++)
        {
        if (DetectCollisionBetween (nobj, n, norm) == 1)
            {
            myLastDCResult.myIsCollisionDetected = true;
            myLastDCResult.myObjectNum = n;
            }
        }
    if (DetectLandCollision (nobj, norm) == 1)
        {
        myLastDCResult.myIsCollisionDetected = true;
        myLastDCResult.myObjectNum = LANDSCAPE_COLLISIONED;
        }
    return myLastDCResult;
    }

/*************************************************************************/

void CEnvironment::RotateChildCenters (CMesh * mesh, CMatrix4x4 &m)
    {
    for (int n = 0; n < mesh -> RetCurChildren (); n ++)
        {
        CPoint3D buf = mesh -> RetChild (n) -> RetCenter ();
        mesh -> RetChild (n) -> RetCenter () = m.DotProduct (mesh -> RetChild (n) -> RetCenter ());
        buf = mesh -> RetChild (n) -> RetCenter () - buf;
        myTable.RetPosition (mesh -> RetChild (n) -> RetID ()) -> RetPoint () += buf;
        myTable.RetIntersectionSpace (mesh -> RetChild (n) -> RetID ()) -> RetPoint () += buf;
        myTable.RetBoundingBox (mesh -> RetChild (n) -> RetID ()) -> RetCorner () += buf;
        RotateChildCenters (mesh -> RetChild (n), m);
        }
    }

/*************************************************************************/

void CEnvironment::ReflectSpeed (int n, CPoint3D &norm)
    {
    CPoint3D speed (- myTable.RetSpeed (n) -> RetPoint (SIMPLE_SPEED) (0),
```

```
                    - myTable.RetSpeed (n) -> RetPoint (SIMPLE_SPEED) (1),
                    - myTable.RetSpeed (n) -> RetPoint (SIMPLE_SPEED) (2)),
            rspeed (0.0F, 0.0F, 0.0F);
    float scale = speed.Normalize ();
    if (scale != 0.0F)
        {
        rspeed = (((norm * speed.DotProduct (norm) * 2.0F) - speed) / scale) * myTable.RetMaterial (n) ->
RetImpulceCoeff ();
        ChangeSpeed (n, SIMPLE_SPEED, rspeed (0), rspeed (1), rspeed (2));
        }
    }

/*************************************************************************/

void CEnvironment::ChangeMovingStep (int handle)
    {
    myTable.ChangeMovingStep (handle);
    }

/*************************************************************************/

void CEnvironment::ChangeAcceleration (int handle, int n, float x, float y, float z)
    {
    myTable.ChangeAcceleration (handle, n, x, y, z);
    }

/*************************************************************************/

void CEnvironment::AddToAcceleration (int handle, int n, float x, float y, float z)
    {
    myTable.AddToAcceleration (handle, n, x, y, z);
    }

/*************************************************************************/

void CEnvironment::ChangeSpeed (int handle, int n, float x, float y, float z)
    {
    myTable.ChangeSpeed (handle, n, x, y, z);
    }

/*************************************************************************/

void CEnvironment::AddToSpeed (int handle, int n, float x, float y, float z)
    {
    myTable.AddToSpeed (handle, n, x, y, z);
    }

/*************************************************************************/

void CEnvironment::MoveAll ()
    {
    if (myPhysicLevel == PL_NO_MOVE)
        {
        myTimePassed.GetNewTime ();
        }

    if (myPhysicLevel >= PL_MOVE_ONLY)
        {
        for (int n = 0; n < myTable.RetCurSize(); n ++)
            {
            if (myTable.RetMesh (n) -> RetParent () == NULL)
                {
                myTable.RetTimeDelta (n) -> GetNewTime ();
                Move (n);
                Rotate (n);
                }
            }
        }

    if (myPhysicLevel >= PL_SPHERE_TEST)
        {
        for (int n = 0; n < myTable.RetCurSize(); n ++)
            {
            if (myTable.RetMesh (n) -> RetParent () == NULL)
                {
                ProcessCollision (n);
                }
            }
        }
    }

/*************************************************************************/

void CEnvironment::Move (int n)
    {
    float prevtime = myTable.RetTimeDelta (n) -> RetPrevTime ();
    CPoint3D OldPos, Norm;
    OldPos.Init (myTable.RetMesh (n)->RetCenter ().RetPointer ());
    myTable.RetOldPosition (n) -> RetPoint () = OldPos;
    CAcceleration accel;
    myTable.RetAccelSum (n, &accel);
    myTable.RetSpeed (n)->RetPoint (1) = myTable.RetSpeed (n)->RetPoint (2) + ((accel.RetPoint () +
myGravity) * myTable.RetTimeDelta (n)->RetTime ());
    myTable.RetMesh (n)->RetCenter () = myTable.RetPosition (n)->RetPoint () + myTable.RetSpeed (n)-
>RetPoint (2) * myTable.RetTimeDelta (n)->RetTime () +
        ((accel.RetPoint () + myGravity) * myTable.RetTimeDelta (n)->RetTime () * myTable.RetTimeDelta
(n)->RetTime ()) / 2;
```

```
    myTable.RetIntersectionSpace (n)->Add (myTable.RetMesh (n)->RetCenter () - OldPos);
    myTable.RetBoundingBox (n)->AddToCorner (myTable.RetMesh (n)->RetCenter () - OldPos);

    for (int i = 0; i < myTable.RetMesh (n) -> RetCurChildren (); i ++)
        Move (myTable.RetMesh (n) -> RetChild (i) -> RetID ());
    }

/*************************************************************************/

void CEnvironment::Rotate (int n)
    {
    float delta = myTable.RetTimeDelta (n) -> RetTime () - myTable.RetTimeDelta (n) -> RetPrevTime ();
    CPoint3D sp = myTable.RetSpeed (n) -> RetPoint (CIRCULAR_SPEED) * delta;
    if (sp (0) != 0.0F || sp (1) != 0.0F || sp (2) != 0.0F)
        {
        CMatrix4x4 mx, my, mz;
        mx.MakeRotXMatrix (sp (0));
        my.MakeRotYMatrix (sp (1));
        mz.MakeRotZMatrix (sp (2));
        mx.DotProduct (my);
        mx.DotProduct (mz);
        myTable.RetMesh (n) -> Rotate (mx);
        RotateChildCenters (myTable.RetMesh (n), mx);
        TransformBox (n, mx);
        }
    for (int i = 0; i < myTable.RetMesh (n) -> RetCurChildren (); i ++)
        Rotate (myTable.RetMesh (n) -> RetChild (i) -> RetID ());
    }

/*************************************************************************/

void CEnvironment::ProcessCollision (int n)
    {
    CPoint3D Norm;
    DetectCollision (n, Norm);
    if (myLastDCResult.myIsCollisionDetected == true)
        {
        int nobj = myLastDCResult.myObjectNum;
        CAcceleration accel1, accel2;
        myTable.RetAccelSum (n, &accel1);
        float prevtime1 = myTable.RetTimeDelta (n) -> RetPrevTime (), prevtime2;
        CPoint3D OldPos1 = myTable.RetOldPosition (n) -> RetPoint (), OldPos2;
        if (nobj != LANDSCAPE_COLLISIONED)
            {
            myTable.RetAccelSum (nobj, &accel2);
            prevtime2 = myTable.RetTimeDelta (nobj) -> RetPrevTime ();
            OldPos2 = myTable.RetOldPosition (nobj) -> RetPoint ();
            myTable.RetSpeed (n) -> RetPoint (1) = myTable.RetSpeed (n) -> RetPoint (2) +
((accel1.RetPoint () + myGravity) * prevtime1);
            myTable.RetIntersectionSpace (n) -> Add (OldPos1 - (myTable.RetMesh (n) -> RetCenter ()));
            myTable.RetBoundingBox (n) -> AddToCorner (OldPos1 - (myTable.RetMesh (n) -> RetCenter ()));
            myTable.RetMesh (n) -> RetCenter () = OldPos1;
            ChangeSpeed (n, CIRCULAR_SPEED, 0.0F, 0.0F, 0.0F);
            myTable.RetSpeed (nobj) -> RetPoint (1) = myTable.RetSpeed (nobj) -> RetPoint (2) +
((accel2.RetPoint () + myGravity) * prevtime2);
            myTable.RetIntersectionSpace (nobj) -> Add (OldPos2 - (myTable.RetMesh (nobj) -> RetCenter
()));
            myTable.RetBoundingBox (nobj) -> AddToCorner (OldPos2 - (myTable.RetMesh (nobj) -> RetCenter
()));
            myTable.RetMesh (nobj) -> RetCenter () = OldPos2;
            ChangeSpeed (nobj, CIRCULAR_SPEED, 0.0F, 0.0F, 0.0F);
            CPoint3D speed1 = (myTable.RetSpeed (nobj) -> RetPoint (1) * (2 * myTable.RetMaterial (nobj) -
> RetMass ()) +
                               myTable.RetSpeed (n) -> RetPoint (1) * (myTable.RetMaterial (n) -> RetMass
() - myTable.RetMaterial (nobj) -> RetMass ())) /
                              (myTable.RetMaterial (n) -> RetMass () + myTable.RetMaterial (nobj) ->
RetMass ());
            CPoint3D speed2 = (myTable.RetSpeed (n) -> RetPoint (1) * (2 * myTable.RetMaterial (n) ->
RetMass ()) +
                               myTable.RetSpeed (nobj) -> RetPoint (1) * (myTable.RetMaterial (nobj) ->
RetMass () - myTable.RetMaterial (n) -> RetMass ())) /
                              (myTable.RetMaterial (n) -> RetMass () + myTable.RetMaterial (nobj) ->
RetMass ());
            ChangeSpeed (n, SIMPLE_SPEED, speed1 (0), speed1 (1), speed1 (2));
            ChangeSpeed (nobj, SIMPLE_SPEED, speed2 (0), speed2 (1), speed2 (2));
            }
        else
            {
            myTable.RetSpeed (n) -> RetPoint (1) = myTable.RetSpeed (n) -> RetPoint (2) +
((accel1.RetPoint () + myGravity) * prevtime1);
            myTable.RetIntersectionSpace (n) -> Add (OldPos1 - (myTable.RetMesh (n) -> RetCenter ()));
            myTable.RetBoundingBox (n) -> AddToCorner (OldPos1 - (myTable.RetMesh (n) -> RetCenter ()));
            myTable.RetMesh (n) -> RetCenter () = OldPos1;
            ReflectSpeed (n, Norm);
            ChangeSpeed (n, CIRCULAR_SPEED, 0.0F, 0.0F, 0.0F);
            }
        }
    for (int i = 0; i < myTable.RetMesh (n) -> RetCurChildren (); i ++)
        ProcessCollision (myTable.RetMesh (n) -> RetChild (i) -> RetID (), myTable.RetMesh (n) ->
RetCenter ());
    }

/*************************************************************************/

void CEnvironment::ProcessCollision (int n, CPoint3D &point)
    {
    CPoint3D Norm;
```

```
    myTable.RetMesh (n) -> RetCenter () += point;
    myTable.RetIntersectionSpace (n) -> RetPoint () += point;
    myTable.RetBoundingBox (n) -> RetCorner () += point;
    DetectCollision (n, Norm);
    myTable.RetMesh (n) -> RetCenter () -= point;
    myTable.RetIntersectionSpace (n) -> RetPoint () -= point;
    myTable.RetBoundingBox (n) -> RetCorner () -= point;

    if (myLastDCResult.myIsCollisionDetected == true)
        {
        int nobj = myLastDCResult.myObjectNum;
        CAcceleration accel1, accel2;
        myTable.RetAccelSum (n, &accel1);
        float prevtime1 = myTable.RetTimeDelta (n) -> RetPrevTime (), prevtime2;
        CPoint3D OldPos1 = myTable.RetOldPosition (n) -> RetPoint (), OldPos2;
        if (nobj != LANDSCAPE_COLLISIONED)
            {
            myTable.RetAccelSum (nobj, &accel2);
            prevtime2 = myTable.RetTimeDelta (nobj) -> RetPrevTime ();
            OldPos2 = myTable.RetOldPosition (nobj) -> RetPoint ();
            myTable.RetSpeed (n) -> RetPoint (1) = myTable.RetSpeed (n) -> RetPoint (2) +
((accel1.RetPoint () + myGravity) * prevtime1);
            myTable.RetIntersectionSpace (n) -> Add (OldPos1 - (myTable.RetMesh (n) -> RetCenter ()));
            myTable.RetBoundingBox (n) -> AddToCorner (OldPos1 - (myTable.RetMesh (n) -> RetCenter ()));
            myTable.RetMesh (n) -> RetCenter () = OldPos1;
            ChangeSpeed (n, CIRCULAR_SPEED, 0.0F, 0.0F, 0.0F);
            myTable.RetSpeed (nobj) -> RetPoint (1) = myTable.RetSpeed (nobj) -> RetPoint (2) +
((accel2.RetPoint () + myGravity) * prevtime2);
            myTable.RetIntersectionSpace (nobj) -> Add (OldPos2 - (myTable.RetMesh (nobj) -> RetCenter
()));
            myTable.RetBoundingBox (nobj) -> AddToCorner (OldPos2 - (myTable.RetMesh (nobj) -> RetCenter
()));
            myTable.RetMesh (nobj) -> RetCenter () = OldPos2;
            ChangeSpeed (nobj, CIRCULAR_SPEED, 0.0F, 0.0F, 0.0F);
            CPoint3D speed1 = (myTable.RetSpeed (nobj) -> RetPoint (1) * (2 * myTable.RetMaterial (nobj) -
> RetMass ()) +
                              myTable.RetSpeed (n) -> RetPoint (1) * (myTable.RetMaterial (n) -> RetMass
() - myTable.RetMaterial (nobj) -> RetMass ())) /
                             (myTable.RetMaterial (n) -> RetMass () + myTable.RetMaterial (nobj) ->
RetMass ());
            CPoint3D speed2 = (myTable.RetSpeed (n) -> RetPoint (1) * (2 * myTable.RetMaterial (n) ->
RetMass ()) +
                              myTable.RetSpeed (nobj) -> RetPoint (1) * (myTable.RetMaterial (nobj) ->
RetMass () - myTable.RetMaterial (n) -> RetMass ())) /
                             (myTable.RetMaterial (n) -> RetMass () + myTable.RetMaterial (nobj) ->
RetMass ());
            ChangeSpeed (n, SIMPLE_SPEED, speed1 (0), speed1 (1), speed1 (2));
            ChangeSpeed (nobj, SIMPLE_SPEED, speed2 (0), speed2 (1), speed2 (2));
            }
        else
            {
            myTable.RetSpeed (n) -> RetPoint (1) = myTable.RetSpeed (n) -> RetPoint (2) +
((accel1.RetPoint () + myGravity) * prevtime1);
            myTable.RetIntersectionSpace (n) -> Add (OldPos1 - (myTable.RetMesh (n) -> RetCenter ()));
            myTable.RetBoundingBox (n) -> AddToCorner (OldPos1 - (myTable.RetMesh (n) -> RetCenter ()));
            myTable.RetMesh (n) -> RetCenter () = OldPos1;
            ReflectSpeed (n, Norm);
            ChangeSpeed (n, CIRCULAR_SPEED, 0.0F, 0.0F, 0.0F);
            }
        }
    for (int i = 0; i < myTable.RetMesh (n) -> RetCurChildren (); i ++)
        ProcessCollision (myTable.RetMesh (n) -> RetChild (i) -> RetID (), myTable.RetMesh (n) ->
RetCenter ());
    }

/*********************************************************************/

/*********************************************************************/
/*=юыєўш€№ ¤ъчхьяы Ё юс·хъ€р CEnvironment*/

void GetInstance (void ** ptr)
    {
    IUnknown * pUnknown = static_cast <IEnvironment *> (new CEnvironment);
    pUnknown -> AddRef ();
    *ptr = pUnknown;
    }

/*********************************************************************/
```

Файл Physics\utils.cpp

Файл PhysicsBase\physbase.cpp

```
/*****************************************************************************/
/*****************************************************************************/
/***                                                                       ***/
/*** PHYSBASE.CPP                                                          ***/
/*** ╬╬╚╦┬╠═╚╬┼ ╠┼┬╬┬╬┬ ┴└╠╬┬▀┌ ╬╦└═╬┬. ╬└┌▀┬╦┼═▀┌ ┬ PHYSBASE.H      ***/
/***                                                                       ***/
/*** └└╔╔ =╚╩╦└═ ╥└╦└╙╚═┬╬▀ └═╙┼┼╠ ┬┼╙┼┼┼╦╠╬╠ ┬ 2002 ├╬═╙               ***/
/***                                                                       ***/
/***                                                                       ***/
/*****************************************************************************/
/*****************************************************************************/

#include "physbase.h"

#define __PROGRAMMER "Xroft"
#define __MODULE_NAME "PhysicsBase"

#include <exeption.h>

BOOL APIENTRY DllMain(HANDLE hModule,
                      DWORD  ul_reason_for_call,
                      LPVOID lpReserved)
{
return 1;
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CTIMER  * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*****************************************************************************/

CTimer::CTimer () :
    myTime (0.0F),
    myStartTime (timeGetTime () / 1000.0F)
    {
    }

/*****************************************************************************/

CTimer::~CTimer ()
    {
    }

/*****************************************************************************/

void CTimer::Init ()
    {
    myTime = 0.0F;
    myStartTime = timeGetTime () / 1000.0F;
    }

/*****************************************************************************/

void CTimer::GetTime ()
    {
    myTime = (timeGetTime () / 1000.0F) - myStartTime;
    }

/*****************************************************************************/

float CTimer::RetTime ()
    {
    return myTime;
    }

/*****************************************************************************/

void CTimer::SetTime (float time)
    {
    myTime = time;
    }

/*****************************************************************************/

void CTimer::AddTime (float time)
    {
    myStartTime -= time;
    }

/*****************************************************************************/

void CTimer::ReduceTime (float time)
    {
    myStartTime += time;
    }

/*****************************************************************************/
```

```
void CTimer::operator = (CTimer & timer)
    {
    myStartTime = timer.myStartTime;
    myTime = timer.myTime;
    }

/*****************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * CLASS CTIMEDELTA  * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*****************************************************************************/

CTimeDelta::CTimeDelta () :
    myPrevTime (0.0F)
    {
    }

/*****************************************************************************/

CTimeDelta::~CTimeDelta ()
    {
    }

/*****************************************************************************/

void CTimeDelta::NullTime ()
    {
    myTime -= myPrevTime;
    myStartTime += myPrevTime;
    myPrevTime = 0.0F;
    }

/*****************************************************************************/

void CTimeDelta::ZeroTime ()
    {
    Init ();
    myPrevTime = 0.0F;
    myTime = 0.0F;
    }

/*****************************************************************************/

void CTimeDelta::GetNewTime ()
    {
    myPrevTime = myTime;
    GetTime ();
    }

/*****************************************************************************/

float CTimeDelta::RetPrevTime ()
    {
    return myPrevTime;
    }

/*****************************************************************************/

void CTimeDelta::operator = (CTimeDelta & timer)
    {
    myStartTime = timer.myStartTime;
    myTime = timer.myTime;
    myPrevTime = timer.myPrevTime;
    }

/*****************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * CLASS CPOSITION * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*****************************************************************************/

CPosition::CPosition ()
    {
    }

/*****************************************************************************/

CPosition::CPosition (float x, float y, float z) :
    myStartPosition (x, y, z)
    {
    }

/*****************************************************************************/

CPosition::CPosition (float xyz [3]) :
    myStartPosition (xyz)
    {
    }
```

```
/*****************************************************************************/

CPosition::CPosition (CPoint3D &speed) :
    myStartPosition (speed)
    {
    }

/*****************************************************************************/

CPosition::~CPosition ()
    {
    }

/*****************************************************************************/

void CPosition::Init (float x, float y, float z)
    {
    myStartPosition.Init (x, y, z);
    }

/*****************************************************************************/

void CPosition::Init (float xyz [3])
    {
    myStartPosition.Init (xyz);
    }

/*****************************************************************************/

void CPosition::Init (CPoint3D &speed)
    {
    myStartPosition.Init (speed);
    }

/*****************************************************************************/

void CPosition::Add (float x, float y, float z)
    {
    myStartPosition.AddToFloats (x, y, z);
    }

/*****************************************************************************/

void CPosition::Add (float xyz [3])
    {
    myStartPosition.AddToFloats (xyz [0], xyz [1], xyz [2]);
    }

/*****************************************************************************/

void CPosition::Add (CPoint3D &speed)
    {
    myStartPosition += speed;
    }

/*****************************************************************************/

float &CPosition::operator () (int n)
    {
    return myStartPosition (n);
    }

/*****************************************************************************/

CPoint3D &CPosition::RetPoint ()
    {
    return myStartPosition;
    }

/*****************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * CLASS CSPEED  * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*****************************************************************************/

CSpeed::CSpeed ()
    {
    }

/*****************************************************************************/

CSpeed::CSpeed (float x, float y, float z, float cx, float cy, float cz) :
    mySpeed (x, y, z),
    myStartSpeed (x, y, z),
    myCircularSpeed (cx, cy, cz)
    {
    }

/*****************************************************************************/

CSpeed::CSpeed (float xyz [3], float cxyz [3]) :
    mySpeed (xyz),
    myStartSpeed (xyz),
```

```
    myCircularSpeed (cxyz)
    {
    }

/*****************************************************************************/

CSpeed::CSpeed (CPoint3D &speed, CPoint3D &cspeed) :
    mySpeed (speed),
    myStartSpeed (speed),
    myCircularSpeed (cspeed)
    {
    }

/*****************************************************************************/

CSpeed::~CSpeed ()
    {
    }

/*****************************************************************************/

void CSpeed::Init (int n, float x, float y, float z)
    {
    if (n == SIMPLE_SPEED)
        {
        mySpeed.Init (x, y, z);
        myStartSpeed.Init (x, y, z);
        }
    if (n == CIRCULAR_SPEED)
        {
        myCircularSpeed.Init (x, y, z);
        }
    }

/*****************************************************************************/

void CSpeed::Init (int n, float xyz [3])
    {
    if (n == SIMPLE_SPEED)
        {
        mySpeed.Init (xyz);
        myStartSpeed.Init (xyz);
        }
    if (n == CIRCULAR_SPEED)
        {
        myCircularSpeed.Init (xyz);
        }
    }

/*****************************************************************************/

void CSpeed::Init (int n, CPoint3D &speed)
    {
    if (n == SIMPLE_SPEED)
        {
        mySpeed.Init (speed);
        myStartSpeed.Init (speed);
        }
    if (n == CIRCULAR_SPEED)
        {
        myCircularSpeed.Init (speed);
        }
    }

/*****************************************************************************/

void CSpeed::Change (int n, float x, float y, float z)
    {
    if (n == SIMPLE_SPEED)
        {
        mySpeed.Init (x, y, z);
        }
    if (n == CIRCULAR_SPEED)
        {
        myCircularSpeed.Init (x, y, z);
        }
    }

/*****************************************************************************/

void CSpeed::Change (int n, float xyz [3])
    {
    if (n == SIMPLE_SPEED)
        {
        mySpeed.Init (xyz);
        }
    if (n == CIRCULAR_SPEED)
        {
        myCircularSpeed.Init (xyz);
        }
    }

/*****************************************************************************/

void CSpeed::Change (int n, CPoint3D &speed)
    {
    if (n == SIMPLE_SPEED)
```

```
        {
        mySpeed.Init (speed);
        }
    if (n == CIRCULAR_SPEED)
        {
        myCircularSpeed.Init (speed);
        }
    }

/*****************************************************************************/

void CSpeed::Add (int n, float x, float y, float z)
    {
    if (n == SIMPLE_SPEED)
        {
        mySpeed.AddToFloats (x, y, z);
        }
    if (n == CIRCULAR_SPEED)
        {
        myCircularSpeed.AddToFloats (x, y, z);
        }
    }

/*****************************************************************************/

void CSpeed::Add (int n, float xyz [3])
    {
    if (n == SIMPLE_SPEED)
        {
        mySpeed.AddToFloats (xyz [0], xyz [1], xyz [2]);
        }
    if (n == CIRCULAR_SPEED)
        {
        myCircularSpeed.AddToFloats (xyz [0], xyz [1], xyz [2]);
        }
    }

/*****************************************************************************/

void CSpeed::Add (int n, CPoint3D &speed)
    {
    if (n == SIMPLE_SPEED)
        {
        mySpeed += speed;
        }
    if (n == CIRCULAR_SPEED)
        {
        myCircularSpeed += speed;
        }
    }

/*****************************************************************************/

void CSpeed::operator = (CSpeed & that)
    {
    mySpeed         = that.mySpeed;
    myStartSpeed    = that.myStartSpeed;
    myCircularSpeed = that.myCircularSpeed;
    }

/*****************************************************************************/

CPoint3D &CSpeed::operator () (int n)
    {
    return RetPoint (n);
    }

/*****************************************************************************/

CPoint3D &CSpeed::RetPoint (int n)
    {
    switch (n)
        {
        case SIMPLE_SPEED:   return mySpeed;
            break;
        case START_SPEED:    return myStartSpeed;
            break;
        case CIRCULAR_SPEED: return myCircularSpeed;
            break;
        default: return mySpeed;
        }
    }

/*****************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CACCELERATION * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*****************************************************************************/

CAcceleration::CAcceleration ()
    {
    }
```

```
/*****************************************************************************/

CAcceleration::CAcceleration (float x, float y, float z) :
    myAcceleration (x, y, z)
    {
    }

/*****************************************************************************/

CAcceleration::CAcceleration (float xyz [3]) :
    myAcceleration (xyz)
    {
    }

/*****************************************************************************/

CAcceleration::CAcceleration (CPoint3D &acceleration) :
    myAcceleration (acceleration)
    {
    }

/*****************************************************************************/

CAcceleration::~CAcceleration ()
    {
    }

/*****************************************************************************/

void CAcceleration::Init (float x, float y, float z)
    {
    myAcceleration.Init (x, y, z);
    }

/*****************************************************************************/

void CAcceleration::Init (float xyz [3])
    {
    myAcceleration.Init (xyz);
    }

/*****************************************************************************/

void CAcceleration::Init (CPoint3D &acceleration)
    {
    myAcceleration.Init (acceleration);
    }

/*****************************************************************************/

void CAcceleration::Add (float x, float y, float z)
    {
    myAcceleration.AddToFloats (x, y, z);
    }

/*****************************************************************************/

void CAcceleration::Add (float xyz [3])
    {
    myAcceleration.AddToFloats (xyz [0], xyz [1], xyz [2]);
    }

/*****************************************************************************/

void CAcceleration::Add (CPoint3D &acceleration)
    {
    myAcceleration += acceleration;
    }

/*****************************************************************************/

void CAcceleration::operator = (CAcceleration & that)
    {
    myAcceleration = that.myAcceleration;
    }

/*****************************************************************************/

float &CAcceleration::operator () (int n)
    {
    return myAcceleration (n);
    }

/*****************************************************************************/

CPoint3D &CAcceleration::RetPoint ()
    {
    return myAcceleration;
    }

/*****************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CINTERSECTIONSPHERE * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
/*****************************************************************************/

CIntersectionSphere::CIntersectionSphere ()
    {
    }

/*****************************************************************************/

CIntersectionSphere::CIntersectionSphere (float radius)
    {
    myRadius = radius;
    }

/*****************************************************************************/

CIntersectionSphere::~CIntersectionSphere ()
    {
    }

/*****************************************************************************/

void CIntersectionSphere::Init (float radius)
    {
    myRadius = radius;
    }

/*****************************************************************************/

float & CIntersectionSphere::RetRadius ()
    {
    return myRadius;
    }

/*****************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CINTERSECTIONSPACE  * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*****************************************************************************/

CIntersectionSpace::CIntersectionSpace ()
    {
    }

/*****************************************************************************/

CIntersectionSpace::CIntersectionSpace (float x, float y, float z, float x1, float y1, float z1) :
    myPosition (x, y, z),
    myX (x1),
    myY (y1),
    myZ (z1)
    {
    }

/*****************************************************************************/

CIntersectionSpace::CIntersectionSpace (float pos [3], float xyz [3]) :
    myPosition (pos),
    myX (xyz [0]),
    myY (xyz [1]),
    myZ (xyz [2])
    {
    }

/*****************************************************************************/

CIntersectionSpace::CIntersectionSpace (CPoint3D &point, float x, float y, float z) :
    myX (x),
    myY (y),
    myZ (z)
    {
    myPosition = point;
    }

/*****************************************************************************/

CIntersectionSpace::~CIntersectionSpace ()
    {
    }

/*****************************************************************************/

void CIntersectionSpace::Init (float x, float y, float z, float x1, float y1, float z1)
    {
    myPosition.Init (x, y, z);
    myX = x1;
    myY = y1;
    myZ = z1;
    }

/*****************************************************************************/

void CIntersectionSpace::Init (float pos [3], float xyz [3])
```

```
    {
    myPosition.Init (pos);
    myX = xyz [0];
    myY = xyz [1];
    myZ = xyz [2];
    }

/*****************************************************************************/

void CIntersectionSpace::Init (CPoint3D &point, float x, float y, float z)
    {
    myPosition = point;
    myX = x;
    myY = y;
    myZ = z;
    }

/*****************************************************************************/

void CIntersectionSpace::Change (float x, float y, float z)
    {
    myPosition.Init (x, y, z);
    }

/*****************************************************************************/

void CIntersectionSpace::Change (float xyz [3])
    {
    myPosition.Init (xyz);
    }

/*****************************************************************************/

void CIntersectionSpace::Change (CPoint3D &point)
    {
    myPosition.Init (point);
    }

/*****************************************************************************/

void CIntersectionSpace::Add (float x, float y, float z)
    {
    myPosition.AddToFloats (x, y, z);
    }

/*****************************************************************************/

void CIntersectionSpace::Add (float xyz [3])
    {
    myPosition += xyz;
    }

/*****************************************************************************/

void CIntersectionSpace::Add (CPoint3D &point)
    {
    myPosition += point;
    }

/*****************************************************************************/

CPoint3D &CIntersectionSpace::RetPoint ()
    {
    return myPosition;
    }

/*****************************************************************************/

float CIntersectionSpace::RetX ()
    {
    return myX;
    }

/*****************************************************************************/

float CIntersectionSpace::RetY ()
    {
    return myY;
    }

/*****************************************************************************/

float CIntersectionSpace::RetZ ()
    {
    return myZ;
    }

/*****************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CBOUNDINGBOX  * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*****************************************************************************/
```

```
CBoundingBox::CBoundingBox ()
    {
    }

/*****************************************************************************/

CBoundingBox::~CBoundingBox ()
    {
    }

/*****************************************************************************/

void CBoundingBox::SetCorner (float x, float y, float z)
    {
    myCorner.Init (x, y, z);
    }

/*****************************************************************************/

CPoint3D &CBoundingBox::RetCorner ()
    {
    return myCorner;
    }

/*****************************************************************************/

void CBoundingBox::SetVector (int n, float x, float y, float z)
    {
    AssertT (n < 3 && n >= 0, "Vector doesn't exist");
    myVector [n].Init (x, y, z);
    }

/*****************************************************************************/

CPoint3D &CBoundingBox::RetVector (int n)
    {
    AssertT (n < 3 && n >= 0, "Vector doesn't exist");
    return myVector [n];
    }

/*****************************************************************************/

CPoint3D &CBoundingBox::operator () (int n)
    {
    return RetVector (n);
    }

/*****************************************************************************/

void CBoundingBox::AddToCorner (float x, float y, float z)
    {
    myCorner.AddToFloats (x, y, z);
    }

/*****************************************************************************/

void CBoundingBox::AddToCorner (float xyz [3])
    {
    myCorner += xyz;
    }

/*****************************************************************************/

void CBoundingBox::AddToCorner (CPoint3D &point)
    {
    myCorner += point;
    }

/*****************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * CLASS CPHYSMATERIAL * * * * * * * * * * * * * * */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*****************************************************************************/

CPhysMaterial::CPhysMaterial ()
    {
    }

/*****************************************************************************/

CPhysMaterial::CPhysMaterial (float mass, float vol, float imp)
    {
    Init (mass, vol, imp);
    }

/*****************************************************************************/

CPhysMaterial::~CPhysMaterial ()
    {
    }

/*****************************************************************************/
```

```
void CPhysMaterial::Init (float mass, float vol, float imp)
    {
    myMass = mass;
    myVolume = vol;
    myImpulceCoeff = imp;
    }

/*****************************************************************************/

float &CPhysMaterial::RetMass ()
    {
    return myMass;
    }

/*****************************************************************************/

float &CPhysMaterial::RetVolume ()
    {
    return myVolume;
    }

/*****************************************************************************/

float &CPhysMaterial::RetImpulceCoeff ()
    {
    return myImpulceCoeff;
    }

/*****************************************************************************/
```

Файл Renderer\renderer.cpp

```
/*****************************************************************************/
/*****************************************************************************/
/***                                                                       ***/
/*** RENDERER.CPP                                                          ***/
/*** ╬╩╚╦╘═╚╬┼ ╠╫╫╬╬╫┬ ╬╦└╤═└ CRENDERER ╚ ┬┬═╩╬╠┼└╫┼╦▄═▀┌ ╬╦└╤═╬┬      ***/
/***                                                                       ***/
/*** └└╔═ ═└═╚╦└═ ╥└╦╙╙╚═╬┬▀╠ └═╙╫┼┼╠ ╤┼╙┼┼┼┬╚╫┼╠ ┬ 2002 ┼╬═╙          ***/
/***                                                                       ***/
/***                                                                       ***/
/*****************************************************************************/
/*****************************************************************************/

#include "renderer.h"

#include <exeption.h>

// ╬Єюс√ ъюьяюэхэЄ ёЄры ьюфєыхь ╤шёЄхь√:
#ifndef RENDERER_MODULE
#define RENDERER_MODULE

#define __PROGRAMMER "Xroft"
#define __MODULE_NAME "Renderer"

#endif

    #define DLL_NAME "Renderer"
    #define DLL_CLASS CRenderer
    #include "..\Common\Utilities\Module.c"


IStorageMaster * STM     = NULL;
IConsole *       Console = NULL;

/*****************************************************************************/

DLL_ONLOAD (loader)
    {
    Assert (loader);
    loader -> GetInterface (IID_IConsole, (void **) &Console);

    if (Console)
        {
        Console -> printf ("RENDERER: Renderer version 0.9 by Xroft.\n");
        }
    }

/*****************************************************************************/

DLL_INIT (loader)
    {
    loader -> GetInterface (IID_IStorageMaster,   (void **) &STM);
    Assert (STM);
    }

/*****************************************************************************/

/*****************************************************************************/
```

```
LRESULT APIENTRY CRendererWndProc (HWND hWnd, UINT messg,
                                   WPARAM wParam, LPARAM lParam)
{
return DllPointerOnRenderer->WindowProc (hWnd, messg, wParam, lParam);
}

/*************************************************************************/

void SetCRendererPointer (CRenderer * renderer)
{
DllPointerOnRenderer = renderer;
}

/*************************************************************************/

/*************************************************************************/
/*************************************************************************/
/********************* CLASS CCAMERA *************************************/
/*************************************************************************/
/*************************************************************************/

/*************************************************************************/

void CCamera::CountSpeed ()
    {
    mySpeed.Init (myDirection - myPosition);
    mySpeed.Normalize ();
    }

/*************************************************************************/

void CCamera::CountPoints ()
    {
    float leftX  = myLeftX  * (myFarZ / myNearZ),
          rightX = myRightX * (myFarZ / myNearZ),
          leftY  = myLeftY  * (myFarZ / myNearZ),
          rightY = myRightY * (myFarZ / myNearZ);

    CPoint3D ox (0.0F, 0.0F, 0.0F), oy (0.0F, 0.0F, 0.0F);
    ox = (myDirection - myPosition).CrossProduct (myUpPoint);
    ox.Normalize ();
    oy = ox.CrossProduct (myDirection - myPosition);
    oy.Normalize ();

    myUpperLeft   = ox * leftX  + oy * rightY + mySpeed * myFarZ + myPosition;
    myUpperRight  = ox * rightX + oy * rightY + mySpeed * myFarZ + myPosition;
    myBottomLeft  = ox * leftX  + oy * leftY  + mySpeed * myFarZ + myPosition;
    myBottomRight = ox * rightX + oy * leftY  + mySpeed * myFarZ + myPosition;
    }

/*************************************************************************/

CCamera::CCamera ()
    {
    }

/*************************************************************************/

CCamera::CCamera (float ex, float ey, float ez, float lx, float ly, float lz, float ux, float uy, float
uz) :
    myPosition (ex, ey, ez),
    myDirection (lx, ly, lz),
    myUpPoint (ux, uy, uz)
    {
    MakeCurrent ();
    CountSpeed ();
    CountPoints ();
    }

/*************************************************************************/

CCamera::~CCamera ()
    {
    glLoadIdentity ();
    }

/*************************************************************************/

void CCamera::Set (float ex, float ey, float ez, float lx, float ly, float lz, float ux, float uy, float
uz)
    {
    myPosition.Init (ex, ey, ez);
    myDirection.Init (lx, ly, lz);
    myUpPoint.Init (ux, uy, uz);
    MakeCurrent ();
    CountSpeed ();
    CountPoints ();
    }

/*************************************************************************/

void CCamera::MakeCurrent ()
    {
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glFrustum (myLeftX, myRightX, myLeftY, myRightY, myNearZ, myFarZ);
```

```
    glMatrixMode (GL_MODELVIEW);
    glLoadIdentity ();
    gluLookAt (myPosition (0) - myDirection (0),
               myPosition (1) - myDirection (1),
               myPosition (2) - myDirection (2),
               0.0F, 0.0F, 0.0F,
               myUpPoint (0), myUpPoint (1), myUpPoint (2));
    glTranslatef (-myDirection (0), -myDirection (1), -myDirection (2));
    }

/*****************************************************************************/

void CCamera::SetFrustum (float leftx, float rightx, float lefty, float righty, float nearz, float farz)
    {
    myLeftX  = leftx;
    myRightX = rightx;
    myLeftY  = lefty;
    myRightY = righty;
    myNearZ  = nearz;
    myFarZ   = farz;
    }

/*****************************************************************************/

void CCamera::MoveAt (float x, float y, float z)
    {
    myPosition.AddToFloats    (x, y, z);
    myDirection.AddToFloats   (x, y, z);
    CountPoints ();
    MakeCurrent ();
    }

/*****************************************************************************/

void CCamera::LookAt (float x, float y, float z)
    {
    myDirection.AddToFloats   (x, y, z);
    CountPoints ();
    MakeCurrent ();
    }

/*****************************************************************************/

void CCamera::Follow (CPoint3D &point)
    {
    CPoint3D dir  = myPosition    - myDirection,
             ul   = myUpperLeft   - myDirection,
             ur   = myUpperRight  - myDirection,
             bl   = myBottomLeft  - myDirection,
             br   = myBottomRight - myDirection;
    myDirection   = point;
    myPosition    = myDirection + dir;
    myUpperLeft   = myDirection + ul;
    myUpperRight  = myDirection + ur;
    myBottomLeft  = myDirection + bl;
    myBottomRight = myDirection + br;
    MakeCurrent ();
    }

/*****************************************************************************/

void CCamera::Seek (CPoint3D &point)
    {
    myDirection = point;
    CountSpeed  ();
    CountPoints ();
    MakeCurrent ();
    }

/*****************************************************************************/

void CCamera::SetPosition (float x, float y, float z)
    {
    myPosition.Init (x, y, z);
    CountSpeed  ();
    CountPoints ();
    MakeCurrent ();
    }

/*****************************************************************************/

void CCamera::Remove ()
    {
    glLoadIdentity ();
    }

/*****************************************************************************/

void CCamera::Slide (float k)
    {
    if (sqrt ((myPosition - myDirection) (0) * (myPosition - myDirection) (0) +
              (myPosition - myDirection) (1) * (myPosition - myDirection) (1) +
              (myPosition - myDirection) (2) * (myPosition - myDirection) (2)) > 0.1F || k < 0)
        {
        myPosition    += (mySpeed * k);
        myUpperLeft   += (mySpeed * k);
        myUpperRight  += (mySpeed * k);
```

```
        myBottomLeft  += (mySpeed * k);
        myBottomRight += (mySpeed * k);
        MakeCurrent ();
        }
    }

/*************************************************************************/

void CCamera::Rotate (CPoint3D & point, float a, CPoint3D & axis)
    {
    CPoint3D newPoint (0.0F, 0.0F, 0.0F);
    float c = (float) cos (a / 57.295779F), s = (float) sin (a / 57.295779F);

    newPoint (0)  = (c + (1 - c) * axis (0) * axis (0))             * point (0);
    newPoint (0) += ((1 - c) * axis (0) * axis (1) - axis (2) * s)  * point (1);
    newPoint (0) += ((1 - c) * axis (0) * axis (2) + axis (1) * s)  * point (2);

    newPoint (1)  = ((1 - c) * axis (0) * axis (1) + axis (2) * s)  * point (0);
    newPoint (1) += (c + (1 - c) * axis (1) * axis (1))             * point (1);
    newPoint (1) += ((1 - c) * axis (1) * axis (2) - axis (0) * s)  * point (2);

    newPoint (2)  = ((1 - c) * axis (0) * axis (2) - axis (1) * s)  * point (0);
    newPoint (2) += ((1 - c) * axis (1) * axis (2) + axis (0) * s)  * point (1);
    newPoint (2) += (c + (1 - c) * axis (2) * axis (2))             * point (2);

    point = newPoint;
    }

/*************************************************************************/

void CCamera::RotateUp (float a)
    {
    CPoint3D point = myPosition    - myDirection,
             ul    = myUpperLeft   - myDirection,
             ur    = myUpperRight  - myDirection,
             bl    = myBottomLeft  - myDirection,
             br    = myBottomRight - myDirection,
             normalizedpoint (0.0F, 0.0F, 0.0F),
             axis = (myDirection - myPosition).CrossProduct (myUpPoint);

    axis.Normalize ();
    Rotate (point, a, axis);
    if (!RotationCheck (point)) return;

    Rotate (ul, a, axis);
    Rotate (ur, a, axis);
    Rotate (bl, a, axis);
    Rotate (br, a, axis);
    Rotate (mySpeed, a, axis);
    myPosition    = point + myDirection;
    myUpperLeft   = ul    + myDirection;
    myUpperRight  = ur    + myDirection;
    myBottomLeft  = bl    + myDirection;
    myBottomRight = br    + myDirection;

    MakeCurrent ();
    }

/*************************************************************************/

void CCamera::RotateZ (float a)
    {
    CPoint3D point = myPosition    - myDirection,
             ul    = myUpperLeft   - myDirection,
             ur    = myUpperRight  - myDirection,
             bl    = myBottomLeft  - myDirection,
             br    = myBottomRight - myDirection,
             normalizedpoint (0.0F, 0.0F, 0.0F),
             axis (0.0F, 0.0F, 1.0F);

    Rotate (point, a, axis);
    if (!RotationCheck (point)) return;

    Rotate (ul, a, axis);
    Rotate (ur, a, axis);
    Rotate (bl, a, axis);
    Rotate (br, a, axis);
    Rotate (mySpeed, a, axis);
    myPosition    = point + myDirection;
    myUpperLeft   = ul    + myDirection;
    myUpperRight  = ur    + myDirection;
    myBottomLeft  = bl    + myDirection;
    myBottomRight = br    + myDirection;

    MakeCurrent ();
    }

/*************************************************************************/

int CCamera::RotationCheck (CPoint3D & point)
    {
    CPoint3D normalizedpoint = point;
    normalizedpoint.Normalize ();

    if (normalizedpoint (0) <= myUpPoint (0) + 0.1F &&
        normalizedpoint (0) >= myUpPoint (0) - 0.1F &&
        normalizedpoint (1) <= myUpPoint (1) + 0.1F &&
```

```
        normalizedpoint (1) >= myUpPoint (1) - 0.1F &&
        normalizedpoint (2) <= myUpPoint (2) + 0.1F &&
        normalizedpoint (2) >= myUpPoint (2) - 0.1F) return 0;
    if (normalizedpoint (0) >= (myUpPoint (0) + 0.1F) * (-1.0F) &&
        normalizedpoint (0) <= (myUpPoint (0) - 0.1F) * (-1.0F) &&
        normalizedpoint (1) >= (myUpPoint (1) + 0.1F) * (-1.0F) &&
        normalizedpoint (1) <= (myUpPoint (1) - 0.1F) * (-1.0F) &&
        normalizedpoint (2) >= (myUpPoint (2) + 0.1F) * (-1.0F) &&
        normalizedpoint (2) <= (myUpPoint (2) - 0.1F) * (-1.0F)) return 0;
    return 1;
    }

/*************************************************************************/

void CCamera::SetViewFromMouse (int centerX, int centerY)
    {
    if (myIsRotateFromMouse)
        {
        POINT curPos;
        GetCursorPos (&curPos);
        SetCursorPos (centerX, centerY);
        float angleUp = - (float) (curPos.y - centerY) / 10.0F,
              angleZ  = - (float) (curPos.x - centerX) / 10.0F;
        RotateUp (angleUp);
        RotateZ  (angleZ);
        }
    }

/*************************************************************************/

float CCamera::RetMinX ()
    {
    float x = myPosition (0);
    if (myUpperLeft   (0) < x) x = myUpperLeft   (0);
    if (myUpperRight  (0) < x) x = myUpperRight  (0);
    if (myBottomLeft  (0) < x) x = myBottomLeft  (0);
    if (myBottomRight (0) < x) x = myBottomRight (0);
    return x;
    }

/*************************************************************************/

float CCamera::RetMaxX ()
    {
    float x = myPosition (0);
    if (myUpperLeft   (0) > x) x = myUpperLeft   (0);
    if (myUpperRight  (0) > x) x = myUpperRight  (0);
    if (myBottomLeft  (0) > x) x = myBottomLeft  (0);
    if (myBottomRight (0) > x) x = myBottomRight (0);
    return x;
    }

/*************************************************************************/

float CCamera::RetMinY ()
    {
    float y = myPosition (1);
    if (myUpperLeft   (1) < y) y = myUpperLeft   (1);
    if (myUpperRight  (1) < y) y = myUpperRight  (1);
    if (myBottomLeft  (1) < y) y = myBottomLeft  (1);
    if (myBottomRight (1) < y) y = myBottomRight (1);
    return y;
    }

/*************************************************************************/

float CCamera::RetMaxY ()
    {
    float y = myPosition (1);
    if (myUpperLeft   (1) > y) y = myUpperLeft   (1);
    if (myUpperRight  (1) > y) y = myUpperRight  (1);
    if (myBottomLeft  (1) > y) y = myBottomLeft  (1);
    if (myBottomRight (1) > y) y = myBottomRight (1);
    return y;
    }

/*************************************************************************/

CPoint3D & CCamera::RetSpeed ()
    {
    return mySpeed;
    }

/*************************************************************************/

/*************************************************************************/
/*************************************************************************/
/******************** CLASS CFONT ****************************************/
/*************************************************************************/
/*************************************************************************/

/*************************************************************************/

CFont::CFont ()
    {
    myItalic = false;
    myUnderLined = false;
```

```
    myStrikeOut = false;
    myWeight = FW_BOLD;
    }

/*****************************************************************************/

CFont::~CFont ()
    {
    }

/*****************************************************************************/

void CFont::Init (char * name, int height, int startLetter, int lastLetter)
    {
    myHeight = height;
    memcpy (myName, name, strlen (name) + 1);
    myFirstLetter = startLetter, myLastLetter = lastLetter;

    myList = glGenLists (256);
    myHDCFont = CreateFont (myHeight,
                            0,
                            0,
                            0,
                            myWeight,
                            myItalic,
                            myUnderLined,
                            myStrikeOut,
                            ANSI_CHARSET,
                            OUT_TT_PRECIS,
                            CLIP_DEFAULT_PRECIS,
                            ANTIALIASED_QUALITY,
                            FF_DONTCARE|DEFAULT_PITCH,
                            myName);
    }

/*****************************************************************************/

void CFont::SetParameters (int weight, bool italic, bool underlined, bool strikeout)
    {
    myWeight = weight;
    myItalic = italic;
    myUnderLined = underlined;
    myStrikeOut = strikeout;
    }

/*****************************************************************************/

/*****************************************************************************/
/*****************************************************************************/
/********************* CLASS CTRANSPFACEBUFFER ***************************/
/*****************************************************************************/
/*****************************************************************************/

/*****************************************************************************/

CTranspFaceBuffer::CTranspFaceBuffer ()
    {
    myBoards        = NULL;
    myCenters       = NULL;
    myPolys         = NULL;
    myTranspTexList = NULL;
    }

/*****************************************************************************/

CTranspFaceBuffer::CTranspFaceBuffer (int bsize, int psize, int texnum)
    {
    Init (bsize, psize, texnum);
    }

/*****************************************************************************/

CTranspFaceBuffer::~CTranspFaceBuffer ()
    {
    delete [] myBoards;
    delete [] myCenters;
    delete [] myPolys;
    delete [] myTranspTexList;
    }

/*****************************************************************************/

void CTranspFaceBuffer::Init (int bsize, int psize, int texnum)
    {
    myBillBoardsNum = bsize;
    myCurBoardNum   = 0;
    myPolygonsNum   = psize;
    myCurPolyNum    = 0;
    myBoards        = new CBillBoard * [bsize];
    myCenters       = new CPoint3D [psize];
    myPolys         = new CPolygon * [psize];
    myTexNum        = texnum;
    myCurTexNum     = 0;
    myTranspTexList = new int [texnum];
    }

/*****************************************************************************/
```

```
void CTranspFaceBuffer::AddTranspTex (int texid)
    {
    AssertT (myCurTexNum < myTexNum, "List of transparent textures is full");
    myTranspTexList [myCurTexNum] = texid;
    myCurTexNum ++;
    Trace ("Texture %d contains transparent texels", texid);
    }

/*************************************************************************/

bool CTranspFaceBuffer::IfTransparent (int texid)
    {
    for (int n = 0; n < myCurTexNum; n ++)
        if (myTranspTexList [n] == texid) return true;
    return false;
    }

/*************************************************************************/

void CTranspFaceBuffer::Clear ()
    {
    myCurBoardNum = 0;
    myCurPolyNum = 0;
    }

/*************************************************************************/

int CTranspFaceBuffer::RetCurBoardNum ()
    {
    return myCurBoardNum;
    }

/*************************************************************************/

int CTranspFaceBuffer::AddBillBoard (CBillBoard * board)
    {
    if (myCurBoardNum < myBillBoardsNum)
        {
        myBoards [myCurBoardNum] = board;
        myCurBoardNum ++;
        return 1;
        }
    return 0;
    }

/*************************************************************************/

CBillBoard * CTranspFaceBuffer::RetBillBoard (int n)
    {
    AssertT (n < myCurBoardNum, "Can't return billboard");
    return myBoards [n];
    }

/*************************************************************************/

int CTranspFaceBuffer::RetCurPolyNum ()
    {
    return myCurPolyNum;
    }

/*************************************************************************/

int CTranspFaceBuffer::AddPolygon (CPoint3D & center, CPolygon * poly)
    {
    if (myCurPolyNum < myPolygonsNum)
        {
        myCenters [myCurPolyNum] = center;
        myPolys   [myCurPolyNum] = poly;
        myCurPolyNum ++;
        return 1;
        }
    return 1;
    }

/*************************************************************************/

CPoint3D & CTranspFaceBuffer::RetPolyCenter (int n)
    {
    AssertT (n < myCurPolyNum, "Can't return center");
    return myCenters [n];
    }

/*************************************************************************/

CPolygon * CTranspFaceBuffer::RetPolygon (int n)
    {
    AssertT (n < myCurPolyNum, "Can't return polygon");
    return myPolys [n];
    }

/*************************************************************************/

/*************************************************************************/
/*************************************************************************/
/********************** CLASS CATMOSPHERE ********************************/
/*************************************************************************/
```

```
/*****************************************************************************/

/*****************************************************************************/

CAtmosphere::CAtmosphere ()
    {
    myStartTexX = 0.0F;
    myStartTexY = 0.0F;
    }

/*****************************************************************************/

CAtmosphere::~CAtmosphere ()
    {
    }

/*****************************************************************************/

void CAtmosphere::SetSkyColor (float r, float g, float b, float a)
    {
    mySkyColor.Init (r, g, b, a);
    }

/*****************************************************************************/

void CAtmosphere::SetSkyPlane (float x1, float y1, float x2, float y2, float height,
                               float r, float g, float b, float a, int tx, int ty, int texId)
    {
    myX1 = x1; myY1 = y1; myX2 = x2; myY2 = y2;
    myHeight = height;
    mySkyColor.Init (r, g, b, a);
    myTexX = tx; myTexY = ty;
    myStartTexX = 0.0F;
    myStartTexY = 0.0F;
    myTexID = texId;
    }

/*****************************************************************************/

void CAtmosphere::SetSkySpeed (float x, float y)
    {
    mySkySpeed.Init (x, y, 0.0F);
    }

/*****************************************************************************/

void CAtmosphere::SetFogColor (float r, float g, float b, float a)
    {
    myFogColor.Init (r, g, b, a);
    }

/*****************************************************************************/

void CAtmosphere::SetFogParams (float density, float start, float end)
    {
    myDensity = density;
    myStart   = start;
    myEnd     = end;
    }

/*****************************************************************************/

void CAtmosphere::SetVolumeFog (bool isVolume, float depth)
    {
    myIsVolumeFogEnable = isVolume;
    myVolumeFogDepth    = depth;
    }

/*****************************************************************************/

/*****************************************************************************/
/*****************************************************************************/
/********************** CLASS CWATER *****************************************/
/*****************************************************************************/
/*****************************************************************************/

/*****************************************************************************/

CWater::CWater ()
    {
    myTexPosX = 0.0F; myTexPosY = 0.0F;
    myStartTime = timeGetTime () * 0.001F;
    }

/*****************************************************************************/

CWater::~CWater ()
    {
    }

/*****************************************************************************/

void CWater::Init (int sizex, int sizey, float height, float tileSize)
    {
    mySizeX = sizex;
    mySizeY = sizey;
    myHeight = height;
```

```
    myTileSize = tileSize;
    }

/*****************************************************************************/

void CWater::SetColor (float r, float g, float b, float a)
    {
    myColor.Init (r, g, b, a);
    }

/*****************************************************************************/

void CWater::InitTexture (int globalTex, int detailTex, int tx, int ty)
    {
    myGlobalTextureID = globalTex;
    myDetailTextureID = detailTex;
    myTexCoordX = tx;
    myTexCoordY = ty;
    }

/*****************************************************************************/

void CWater::SetSpeed (float x, float y)
    {
    mySpeed.Init (x, y, 0.0F);
    }

/*****************************************************************************/

void CWater::Move ()
    {
    myCurrentTime = timeGetTime () * 0.001F;
    myTexPosX += mySpeed (0) * (myCurrentTime - myStartTime);
    myTexPosY += mySpeed (1) * (myCurrentTime - myStartTime);
    myStartTime = myCurrentTime;
    }

/*****************************************************************************/

/*****************************************************************************/
/*****************************************************************************/
/********************* CLASS CRENDERER ***************************************/
/*****************************************************************************/
/*****************************************************************************/

#define LOD_AXIS_X 1
#define LOD_AXIS_Y 2
#define LOD_GLOBAL 3

/*****************************************************************************/

CRenderer::CRenderer () :
    myRefs (0)
    {
    myWindow             = NULL;
    myCurLight           = 0;
    myCameraNum          = 0;
    myCurrentCamera      = 0;
    myRenderLevel        = RL_WIREFRAME;
    myRotationWay        = RW_NO_ROTATING;
    KeyboardFunc         = NULL;
    RedrawFunc           = NULL;
    glMultiTexCoord2fARB = NULL;
    glActiveTextureARB   = NULL;
    glFogCoordfEXT       = NULL;
    myBufferWriteMode    = 1;
    myIsMouseCaptured    = false;
    myStartTime          = (float) timeGetTime () * 0.001F;
    myFPS                = 0;
    myCurrentFrames      = 0;
    myPreviousCameraPosition.Init (0.0F, 0.0F, 0.0F);

    myBufPoint.Init (0.0F, 0.0F, 0.0F);
    myBufMaterial.SetColorSource (COLOR_PER_VERTEX);
    myBufMaterial.Init (1, 0, 0);

    SetCRendererPointer (this);
    }

/*****************************************************************************/

CRenderer::CRenderer (int x1, int y1, int x2, int y2) :
    myRefs (0)
    {
    myWindow             = NULL;
    myX1                 = x1;
    myY1                 = y1;
    myX2                 = x2;
    myY2                 = y2;
    KeyboardFunc         = NULL;
    RedrawFunc           = NULL;
    myCurLight           = 0;
    myCameraNum          = 0;
    myCurrentCamera      = 0;
    myRenderLevel        = RL_WIREFRAME;
    myRotationWay        = RW_NO_ROTATING;
    glMultiTexCoord2fARB = NULL;
```

```
    glActiveTextureARB   = NULL;
    glFogCoordfEXT       = NULL;
    myBufferWriteMode    = 1;
    myIsMouseCaptured    = false;
    myStartTime          = (float) timeGetTime () * 0.001F;
    myFPS                = 0;
    myCurrentFrames      = 0;
    myPreviousCameraPosition.Init (0.0F, 0.0F, 0.0F);

    myBufPoint.Init (0.0F, 0.0F, 0.0F);
    myBufMaterial.SetColorSource (COLOR_PER_VERTEX);
    myBufMaterial.Init (1, 0, 0);

    SetCRendererPointer (this);
    }

/*************************************************************************/

CRenderer::~CRenderer ()
    {
    }

/*************************************************************************/

HRESULT __stdcall CRenderer::QueryInterface (const IID &iid, void ** ptr)
    {
    if (iid == IID_IUnknown)
        {
        *ptr = static_cast <IRenderer *> (this);
        }
    else
        {
        if (iid == IID_IRenderer)
            {
            *ptr = static_cast <IRenderer *> (this);
            }
        else
            {
            *ptr = NULL;
            return E_NOINTERFACE;
            }
        }
    reinterpret_cast <IUnknown *> (* ptr) -> AddRef ();
    return S_OK;
    }

/*************************************************************************/

ULONG __stdcall CRenderer::AddRef ()
    {
    return InterlockedIncrement (&myRefs);
    }

/*************************************************************************/

ULONG __stdcall CRenderer::Release ()
    {
    if (InterlockedDecrement (&myRefs) == 0)
        {
        delete this;
        return 0;
        }
    return myRefs;
    }

/*************************************************************************/

LRESULT APIENTRY CRenderer::WindowProc (HWND hWnd, UINT messg,
                                        WPARAM wParam, LPARAM lParam)
{
PAINTSTRUCT ps;

switch (messg)
    {
    case WM_CREATE:
        myDC = GetDC (hWnd);
        SetupPixelFormat ();
         SetupPalette ();
        myGLRC = wglCreateContext (myDC);
         wglMakeCurrent (myDC, myGLRC);
        InitGL ();
        break;

    case WM_PALETTECHANGED:
         if (myGLRC && myPalette && (HWND) wParam != hWnd)
            {
             UnrealizeObject(myPalette);
             SelectPalette(myDC, myPalette, FALSE);
             RealizePalette(myDC);
             break;
            }
         break;

    case WM_QUERYNEWPALETTE:
         if (myGLRC && myPalette)
            {
             UnrealizeObject(myPalette);
```

```
            SelectPalette(myDC, myPalette, FALSE);
            RealizePalette(myDC);
            return TRUE;
            }
        break;

    case WM_PAINT:
        BeginPaint (hWnd, &ps);
        Redraw ();
        EndPaint (hWnd, &ps);
        break;

    case WM_DESTROY:
        if (myGLRC)
            {
            wglMakeCurrent(NULL, NULL);
            wglDeleteContext(myGLRC);
            }
        if (myPalette)
            {
            DeleteObject(myPalette);
            }
        return(DefWindowProc(hWnd, messg, wParam, lParam));
        break;

    case WM_SIZE:
        if (!myIsFullScreen)
            {
            myX2 = LOWORD (lParam);
            myY2 = HIWORD (lParam);
            Resize ();
            }
        break;

    case WM_MOVE:
        myX1 = LOWORD (lParam);
        myY1 = HIWORD (lParam);
        break;

    case WM_CHAR:
        if (KeyboardFunc != NULL) (* KeyboardFunc) (hWnd, messg, wParam, lParam);
        break;

    case WM_LBUTTONUP:
        if (myIsMouseCaptured == false)
            {
            for ( ; ; ) if (ShowCursor (false) < 0) break;
            SetCursorPos (myX1 + myX2 / 2, myY1 + myY2 / 2);
            myIsMouseCaptured = true;
            break;
            }
        else
            {
            ShowCursor (true);
            myIsMouseCaptured = false;
            break;
            }

    case WM_KILLFOCUS:
        if (myIsMouseCaptured == true)
            {
            ShowCursor (true);
            myIsMouseCaptured = false;
            }
        break;

    default:
        return(DefWindowProc(hWnd, messg, wParam, lParam));

    }
return(0);
}

/*************************************************************************/

int CRenderer::MakeGLWindow (HINSTANCE hInst, int nCmdShow,
                             char * progname, char * wndname)
    {
    DWORD style;

    myWndClass.lpszClassName = progname;
    myWndClass.hInstance     = hInst;
    myWndClass.lpfnWndProc   = CRendererWndProc;
    myWndClass.hCursor       = LoadCursor(NULL, IDC_ARROW);
    myWndClass.hIcon         = 0;
    myWndClass.lpszMenuName  = 0;
    myWndClass.hbrBackground = NULL;
    myWndClass.style         = CS_HREDRAW | CS_VREDRAW;
    myWndClass.cbClsExtra    = 0;
    myWndClass.cbWndExtra    = 0;

    if (!RegisterClass (&myWndClass))
        return 0;

    if (myIsFullScreen)
        {
        style = WS_POPUP;
```

```
        SetFullScreen ();
        }
    else
        {
        style = WS_OVERLAPPEDWINDOW;
        }
    RECT rect;
    rect.left   = 0;
    rect.right  = myX2;
    rect.top    = 0;
    rect.bottom = myY2;
    AdjustWindowRect(&rect, style, false);

    myWindow = CreateWindow (progname, wndname, style,
                             myX1, myY1, myX2, myY2, (HWND)NULL, (HMENU)NULL,
                             (HINSTANCE)hInst, (LPSTR)NULL);
    ShowWindow (myWindow, nCmdShow);
    UpdateWindow (myWindow);
    Trace ("OpenGL window initialized successfully");
    return 1;
    }

/*****************************************************************************/

void CRenderer::ShowGLWindow ()
    {
    ShowWindow (myWindow, SW_SHOWNORMAL);
    }

/*****************************************************************************/

void CRenderer::HideGLWindow ()
    {
    ShowWindow (myWindow, SW_HIDE);
    }

/*****************************************************************************/

int CRenderer::RetGLWindowX1 ()
    {
    return myX1;
    }

/*****************************************************************************/

int CRenderer::RetGLWindowY1 ()
    {
    return myY1;
    }

/*****************************************************************************/

int CRenderer::RetGLWindowX2 ()
    {
    return myX2;
    }

/*****************************************************************************/

int CRenderer::RetGLWindowY2 ()
    {
    return myY2;
    }

/*****************************************************************************/

void CRenderer::SetFullScreen ()
    {
    DEVMODE dmSettings = {0};

    AssertT (EnumDisplaySettings (NULL, ENUM_CURRENT_SETTINGS, &dmSettings), "Could Not Enum Display
Settings");

    dmSettings.dmPelsWidth = myX2;
    dmSettings.dmPelsHeight= myY2;
    dmSettings.dmFields     = DM_BITSPERPEL | DM_PELSWIDTH | DM_PELSHEIGHT | DM_DISPLAYFREQUENCY;

    int result = ChangeDisplaySettings (&dmSettings, CDS_FULLSCREEN);
    AssertT (result == DISP_CHANGE_SUCCESSFUL, "Display mode is not compatible");
    }

/*****************************************************************************/

void CRenderer::SetRenderLevel (int rl)
    {
    myRenderLevel = rl;
    if (rl >= RL_EXTENSIONS_USED) SetVolumeFogDepth (myAtmosphere.myIsVolumeFogEnable,
myAtmosphere.myVolumeFogDepth);
    if (rl < RL_LOW_TEXTURED) glDisable (GL_TEXTURE_2D);
    else glEnable (GL_TEXTURE_2D);
    if (rl < RL_LIGHTENED) DisableLighting ();
    else EnableLighting ();
    }

/*****************************************************************************/

void CRenderer::InitGL (void)
```

```
    {
    Resize ();
    glMatrixMode (GL_MODELVIEW);
    glTranslatef (0.0F, 0.0F, 0.0F);

    glShadeModel (GL_SMOOTH);

    glEnable (GL_DEPTH_TEST);
    glEnable (GL_BLEND);
    glEnable (GL_POLYGON_OFFSET_FILL);
    glPolygonOffset (1.0F, 1.0F);
    if (myRenderLevel >= RL_LIGHTENED)     EnableLighting ();
    if (myRenderLevel >= RL_LOW_TEXTURED)  glEnable (GL_TEXTURE_2D);
    glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glEnable (GL_NORMALIZE);
    glCullFace (GL_BACK);

    SetClearColor (0.0F, 0.0F, 0.0F, 1.0F);
    glFogi  (GL_FOG_MODE,    GL_EXP2);
    glFogfv (GL_FOG_COLOR,   myClearColor.RetPointer ());
    glFogf  (GL_FOG_DENSITY, 0.01F);
    glHint  (GL_FOG_HINT,    GL_NICEST);
    glFogf  (GL_FOG_START,   0.0F);
    glFogf  (GL_FOG_END,     5.0F);
    glEnable (GL_FOG);

    glLightModeli (GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
    glLightModeli (GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
    myDefaultAmbient [0] = 0.1F; myDefaultAmbient [1] = 0.1F;
    myDefaultAmbient [2] = 0.1F; myDefaultAmbient [3] = 1.0F;
    glMaterialfv (GL_FRONT_AND_BACK, GL_AMBIENT, myDefaultAmbient);

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);
    LoadExtFuncs ();
    }

/*****************************************************************************/

void CRenderer::LoadExtFuncs ()
    {
    glActiveTextureARB   = (PFNGLACTIVETEXTUREARBPROC)   wglGetProcAddress("glActiveTextureARB");
    glMultiTexCoord2fARB = (PFNGLMULTITEXCOORD2FARBPROC) wglGetProcAddress("glMultiTexCoord2fARB");
    glFogCoordfEXT       = (PFNGLFOGCOORDFEXTPROC)       wglGetProcAddress("glFogCoordfEXT");
    if(!glActiveTextureARB || !glMultiTexCoord2fARB || !glFogCoordfEXT)
        Trace ("Warning: some OpenGL extension functions couldn't be loaded");
    }

/*****************************************************************************/

void CRenderer::Resize ()
    {
    glViewport (0, 0, myX2, myY2);
    }

/*****************************************************************************/

void CRenderer::Resize (int x1, int y1, int x2, int y2, bool fullscr)
    {
    myX1 = x1;
    myY1 = y1;
    myX2 = x2;
    myY2 = y2;
    Resize ();
    if (myWindow != NULL)
        {
        if (fullscr)
            {
            SetFullScreen ();
            SetWindowLong (myWindow, GWL_STYLE, WS_POPUP);
            }
        else
            {
            ChangeDisplaySettings (NULL, 0);
            SetWindowLong (myWindow, GWL_STYLE, WS_OVERLAPPEDWINDOW);
            InvalidateRect (NULL, NULL, true);
            }
        SetWindowPos (myWindow, HWND_TOP, myX1, myY1, myX2, myY2, SWP_SHOWWINDOW | SWP_NOZORDER);
        }
    myIsFullScreen = fullscr;
    }

/*****************************************************************************/

void CRenderer::SetupPixelFormat ()
    {
    PIXELFORMATDESCRIPTOR pfd = {
        sizeof(PIXELFORMATDESCRIPTOR),
        1,
        PFD_SUPPORT_OPENGL | PFD_DRAW_TO_WINDOW |
        PFD_DOUBLEBUFFER,
        PFD_TYPE_RGBA,
        16,
        0, 0, 0, 0, 0, 0,
        0,
        0,
        0,
        0, 0, 0, 0,
```

```
        16,
        0,
        0,
        PFD_MAIN_PLANE,
        0,
        0, 0, 0,
        };
    int pixelFormat;
    pixelFormat = ChoosePixelFormat(myDC, &pfd);

    if (pixelFormat == 0)
        {
        MessageBox (WindowFromDC(myDC), "ChoosePixelFormat failed.", "Error",
                MB_ICONERROR | MB_OK);        exit(1);
        }

    if (SetPixelFormat(myDC, pixelFormat, &pfd) != TRUE)
        {
        MessageBox(WindowFromDC(myDC), "SetPixelFormat failed.", "Error",
                MB_ICONERROR | MB_OK);        exit(1);
        }
    }

/*****************************************************************************/

void CRenderer::SetupPalette ()
    {
    int pixelFormat = GetPixelFormat(myDC);

    PIXELFORMATDESCRIPTOR pfd;
    LOGPALETTE * pPal;
    int paletteSize;

    DescribePixelFormat(myDC, pixelFormat, sizeof(PIXELFORMATDESCRIPTOR), &pfd);
    if (pfd.dwFlags & PFD_NEED_PALETTE)
        {
        paletteSize = 1 << pfd.cColorBits;
        }
    else
        {
        return;
        }

    pPal = (LOGPALETTE*) malloc(sizeof(LOGPALETTE) + paletteSize * sizeof(PALETTEENTRY));

    pPal->palVersion = 0x300;
    pPal->palNumEntries = paletteSize;
    /* build a simple RGB color palette */
        {
        int redMask = (1 << pfd.cRedBits) - 1;
        int greenMask = (1 << pfd.cGreenBits) - 1;
        int blueMask = (1 << pfd.cBlueBits) - 1;        int i;
        for (i=0; i<paletteSize; ++i)
            {
            pPal->palPalEntry[i].peRed =
                    (((i >> pfd.cRedShift) & redMask) * 255) / redMask;
            pPal->palPalEntry[i].peGreen =
                    (((i >> pfd.cGreenShift) & greenMask) * 255) / greenMask;
            pPal->palPalEntry[i].peBlue =
                    (((i >> pfd.cBlueShift) & blueMask) * 255) / blueMask;
            pPal->palPalEntry[i].peFlags = 0;
            }
        }

    myPalette = CreatePalette(pPal);
    free(pPal);

    if (myPalette)
        {
        SelectPalette(myDC, myPalette, FALSE);
        RealizePalette(myDC);
        }
    }

/*****************************************************************************/

void CRenderer::MakeCurrent ()
    {
    wglMakeCurrent (myDC, myGLRC);
    Resize ();
    }

/*****************************************************************************/

void CRenderer::Clear ()
    {
    glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    }

/*****************************************************************************/

void CRenderer::SetClearColor (float r, float g, float b, float a)
    {
    myClearColor.Init (r, g, b, a);
    myAtmosphere.SetFogColor (r, g, b, a);
    glClearColor (myClearColor (0), myClearColor (1), myClearColor (2), myClearColor (3));
    glFogfv (GL_FOG_COLOR, myAtmosphere.myFogColor.myRGBA);
```

```
    }
/*****************************************************************************/
void CRenderer::SetFogParameters (float density, float start, float end)
    {
    myAtmosphere.SetFogParams (density, start, end);
    glFogf (GL_FOG_DENSITY, density);
    glFogf (GL_FOG_START,   start);
    glFogf (GL_FOG_END,     end);
    }
/*****************************************************************************/
void CRenderer::SetVolumeFogDepth (bool isenable, float depth)
    {
    myAtmosphere.SetVolumeFog (isenable, depth);

    if (isenable && myRenderLevel >= RL_EXTENSIONS_USED)
        {
        glFogi (GL_FOG_MODE, GL_LINEAR);
        glFogi (GL_FOG_COORDINATE_SOURCE_EXT, GL_FOG_COORDINATE_EXT);
        }
    else
        {
        glFogi (GL_FOG_MODE, GL_EXP2);
        glFogi (GL_FOG_COORDINATE_SOURCE_EXT, GL_FRAGMENT_DEPTH_EXT);
        }
    }
/*****************************************************************************/
void CRenderer::SetSkyPlane (float x1, float y1, float x2, float y2, float height,
                             float r, float g, float b, float a, int tx, int ty, int texId)
    {
    myAtmosphere.SetSkyPlane (x1, y1, x2, y2, height, r, g, b, a, tx, ty, texId);
    }
/*****************************************************************************/
void CRenderer::SetSkyColor (float r, float g, float b, float a)
    {
    myAtmosphere.SetSkyColor (r, g, b, a);
    }
/*****************************************************************************/
void CRenderer::SetSkySpeed (float x, float y)
    {
    myAtmosphere.SetSkySpeed (x, y);
    }
/*****************************************************************************/
void CRenderer::InitTransparentBuffer (int bsize, int psize, int texnum)
    {
    myBuffer.Init (bsize, psize, texnum);
    }
/*****************************************************************************/
void CRenderer::SetRotationWay (int rotationway)
    {
    myRotationWay = rotationway;
    }
/*****************************************************************************/
void CRenderer::SetLoDLevel (int lod)
    {
    myLoDLevel = lod;
    }
/*****************************************************************************/
void CRenderer::Redraw ()
    {
    SwapBuffers(myDC);
    }
/*****************************************************************************/
int CRenderer::RetFPS ()
    {
    return myFPS;
    }
/*****************************************************************************/
void CRenderer::CountFPS ()
    {
    myCurrentFrames ++;
    myCurrentTime = (float) timeGetTime () * 0.001F;

    if (myCurrentTime - myStartTime > 1.0F)
        {
        myFPS = myCurrentFrames;
```

```
        myCurrentFrames = 0;
        myStartTime = myCurrentTime;
        }
    }

/*************************************************************************/

void CRenderer::SetKeyboardFunc (void (* kbf )(HWND hWnd,UINT messg, WPARAM wParam,LPARAM lParam))
    {
    KeyboardFunc = kbf;
    }

/*************************************************************************/

void CRenderer::PrepareMeshes (CSmartPointer &meshes, CSmartPointer &polys,
                               CSmartPointer &meshes2, CSmartPointer &polys2, CSmartPointer &lights)
    {
    int pnum = 0, lnum = 0;
    for (int n = 0; n < meshes.RetObjNum (); n ++)
        {
        ((CMesh *) meshes.RetObject (n)) -> New ();
        int polynum = ((CMesh *) meshes.RetObject (n)) -> RetMaxPolys (),
            lightnum = ((CMesh *) meshes.RetObject (n)) -> RetMaxLights ();
        for (int p = pnum; p < polynum + pnum; p ++)
            ((CMesh *) meshes.RetObject (n)) -> AddPoly ((CPolygon *) polys.CreateObject ());
        for (int l = 0; l < lightnum + lnum; l ++)
            {
            ((CMesh *) meshes.RetObject (n)) -> AddLight ((CLightSource *) lights.CreateObject ());
            glEnable (((CLightSource *) lights.RetObject (l)) -> myLightNum);
            }
        pnum += polynum; lnum += lightnum;
        }

    pnum = 0;
    for (n = 0 ; n < meshes.RetObjNum (); n++)
        {
        CMesh * mesh = ((CMesh *) meshes.CreateObject ());
        int polynum = mesh -> RetMaxPolys ();
        mesh -> Clone (((CMesh *) meshes.RetObject (n)),
                       ((CPolygon *) polys2.CreateObjects (pnum)));
        pnum += polynum;
        }
    }

/*************************************************************************/

void CRenderer::PrepareMeshes (CSmartPointer &meshes, CSmartPointer &polys,
                               CSmartPointer &meshes2, CSmartPointer &polys2, CSmartPointer &lights, int
start, int stop)
    {
    int pnum = 0, lnum = 0;
    for (int n = start; n < stop; n ++)
        {
        ((CMesh *) meshes.RetObject (n)) -> New ();
        int polynum = ((CMesh *) meshes.RetObject (n)) -> RetMaxPolys (),
            lightnum = ((CMesh *) meshes.RetObject (n)) -> RetMaxLights ();
        for (int p = pnum; p < polynum + pnum; p ++)
            ((CMesh *) meshes.RetObject (n)) -> AddPoly ((CPolygon *) polys.RetObject (p));
        for (int l = 0; l < lightnum + lnum; l ++)
            {
            ((CMesh *) meshes.RetObject (n)) -> AddLight ((CLightSource *) lights.CreateObject ());
            glEnable (((CLightSource *) lights.RetObject (l)) -> myLightNum);
            }
        pnum += polynum; lnum += lightnum;
        }

    pnum = 0;
    for (n = start; n < stop; n++)
        {
        CMesh * mesh = ((CMesh *) meshes2.CreateObject ());
        int polynum = ((CMesh *) meshes.RetObject (n)) -> RetMaxPolys ();
        mesh -> Clone (((CMesh *) meshes.RetObject (n)),
                       ((CPolygon *) polys2.CreateObjects (polynum)));
        pnum += polynum;
        }
    }

/*************************************************************************/

void CRenderer::PrepareMeshes (void * meshes, void * meshes2, int nmeshes, void * polys, void * polys2,
int npolys, void * lights, int nlights)
    {
    PrepareMeshes (reinterpret_cast <CMesh *> (meshes), reinterpret_cast <CMesh *> (meshes2), nmeshes,
                   reinterpret_cast <CPolygon *> (polys), reinterpret_cast <CPolygon *> (polys2), npolys,
                   reinterpret_cast <CLightSource *> (lights), nlights);
    }

/*************************************************************************/

void CRenderer::PrepareMeshes (CMesh * meshes, CMesh * meshes2, int nmeshes, CPolygon * polys, CPolygon *
polys2, int npolys, CLightSource * lights, int nlights)
    {
    int pnum = 0, lnum = 0;
    for (int n = 0; n < nmeshes; n ++)
        {
        meshes [n].New ();
        int polynum = meshes [n].RetMaxPolys (), lightnum = meshes [n].RetMaxLights ();
```

```
        meshes [n].AddPolys (&polys [pnum], polynum);
        meshes [n].AddLights (&lights [lnum], lightnum);
        for (int i = lnum; i < lnum + lightnum; i ++)
            glEnable (lights [i].myLightNum);
        pnum += polynum; lnum += lightnum;
        }

    pnum = 0;
    for (n = 0 ; n < nmeshes; n++)
        {
        int polynum = meshes [n].RetMaxPolys ();
        meshes2 [n].Clone (&meshes [n], &polys2 [pnum]);
        pnum += polynum;
        }
    }

/*****************************************************************************/

void CRenderer::InitFont (char * name, int height, int startLetter, int lastLetter,
                          int weight, bool italic, bool underlined, bool strikeout)
    {
    myFont.SetParameters (weight, italic, underlined, strikeout);
    myFont.Init (name, height, startLetter, lastLetter);
    SelectObject (myDC, myFont.myHDCFont);
    wglUseFontBitmaps (myDC, startLetter, lastLetter, myFont.myList);
    }

/*****************************************************************************/

void CRenderer::SetTextPosition (int x, int y)
    {
    glPushAttrib (GL_TRANSFORM_BIT | GL_VIEWPORT_BIT);
    glMatrixMode (GL_PROJECTION);
    glPushMatrix ();
    glLoadIdentity ();
    glMatrixMode (GL_MODELVIEW);
    glPushMatrix ();
    glLoadIdentity ();

    y = myY2 - myFont.myHeight - y;
    glViewport (x - 1, y - 1, 0, 0);
    glRasterPos4f (0, 0, 0, 1);

    glPopMatrix ();
    glMatrixMode (GL_PROJECTION);
    glPopMatrix ();
    glPopAttrib ();
    glMatrixMode (GL_MODELVIEW);
    }

/*****************************************************************************/

void CRenderer::OutText (int x, int y, float r, float g, float b, float a, char * format, ...)
    {
    if (myRenderLevel >= RL_LIGHTENED) glDisable (GL_LIGHTING);
    glColor4f (r, g, b, a);
    SetTextPosition (x, y);

    char text [256];
    va_list arguments;
    va_start (arguments, format);
    vsprintf (text, format, arguments);
    va_end (arguments);
    glListBase (myFont.myList);
    glCallLists (strlen (text), GL_UNSIGNED_BYTE, text);

    glPopAttrib ();
    glPopMatrix ();
    if (myRenderLevel >= RL_LIGHTENED) glEnable (GL_LIGHTING);
    }

/*****************************************************************************/

void CRenderer::SetRedrawFunc (void (* redraw) ())
    {
    RedrawFunc = redraw;
    }

/*****************************************************************************/

void CRenderer::Render ()
    {
    MakeCurrent ();
    myBuffer.Clear ();
    if (myIsMouseCaptured == true) myCameras [myCurrentCamera - 1].SetViewFromMouse (myX1 + myX2 / 2, myY1
+ myY2 / 2);
    MakeCameraCurrent (myCurrentCamera);
    ResetLights (myGlobalLights, myCurLight);
    if (RedrawFunc != NULL) (* RedrawFunc) ();
    RenderAtmosphere ();
    RenderTransparent ();
    RenderWater ();
    Redraw ();
    glLightModelfv (GL_LIGHT_MODEL_AMBIENT, myDefaultAmbient);
    myPreviousCameraPosition = myCameras [myCurrentCamera - 1].myPosition;
    CountFPS ();
    }
```

```
/*************************************************************************/

int CRenderer::LoadTexture (CColorMatrix * tex, char * filename)
    {
    AUX_RGBImageRec * image = new AUX_RGBImageRec;
    image = auxDIBImageLoad (filename);
    tex -> Init (image -> sizeX, image -> sizeY);
    for (int y = 0; y < tex -> RetY (); y ++)
        {
        for (int x = 0; x < tex -> RetX (); x ++)
            {
            char r = image -> data [(y * image -> sizeX + x) * 3 + 0];
            char g = image -> data [(y * image -> sizeX + x) * 3 + 1];
            char b = image -> data [(y * image -> sizeX + x) * 3 + 2];
            tex -> AddTexel (x, y, r / 255.0F, g / 255.0F, b / 255.0F, 1.0F);
            }
        }
    delete [] image -> data;
    return 1;
    }

/*************************************************************************/

int CRenderer::LoadTexture (CColorMatrix * tex, char * filename, char ar, char ag, char ab)
    {
    AUX_RGBImageRec * image = new AUX_RGBImageRec;
    image = auxDIBImageLoad (filename);
    tex -> Init (image -> sizeX, image -> sizeY);
    for (int y = 0; y < tex -> RetY (); y ++)
        {
        for (int x = 0; x < tex -> RetX (); x ++)
            {
            char r = image -> data [(y * image -> sizeX + x) * 3 + 0];
            char g = image -> data [(y * image -> sizeX + x) * 3 + 1];
            char b = image -> data [(y * image -> sizeX + x) * 3 + 2];
            tex -> AddTexel (x, y, r / 255.0F, g / 255.0F, b / 255.0F, 1.0F);
            if (r == ar && g == ag && b == ab) tex -> RetElement (x, y, 3) = 0.0F;
            }
        }
    delete [] image -> data;
    return 1;
    }

/*************************************************************************/

int CRenderer::LoadTexture (CColorMatrix * tex, char * filename, char ar, char ag, char ab, float alpha)
    {
    AUX_RGBImageRec * image = new AUX_RGBImageRec;
    image = auxDIBImageLoad (filename);
    tex -> Init (image -> sizeX, image -> sizeY);
    for (int y = 0; y < tex -> RetY (); y ++)
        {
        for (int x = 0; x < tex -> RetX (); x ++)
            {
            char r = image -> data [(y * image -> sizeX + x) * 3 + 0];
            char g = image -> data [(y * image -> sizeX + x) * 3 + 1];
            char b = image -> data [(y * image -> sizeX + x) * 3 + 2];
            tex -> AddTexel (x, y, r / 255.0F, g / 255.0F, b / 255.0F, 1.0F);
            if (r == ar && g == ag && b == ab) tex -> RetElement (x, y, 3) = alpha;
            }
        }
    delete [] image -> data;
    return 1;
    }

/*************************************************************************/

int CRenderer::LoadHorizont (CLandscape * land, int nhor, char * filename, float n)
    {
    AUX_RGBImageRec * image = new AUX_RGBImageRec;
    image = auxDIBImageLoad (filename);
    land -> RetHorizont (nhor).Init (image -> sizeX, image -> sizeY);
    land -> RetNormalMap (nhor).Init (image -> sizeX, image -> sizeY);
    for (int y = 0; y < land -> RetHorizont (nhor).RetY (); y ++)
        {
        for (int x = 0; x < land -> RetHorizont (nhor).RetX (); x ++)
            {
            float r = (float) image -> data [(y * image -> sizeX + x) * 3 + 0];
            float g = (float) image -> data [(y * image -> sizeX + x) * 3 + 1];
            float b = (float) image -> data [(y * image -> sizeX + x) * 3 + 2];
            land -> RetHorizont (nhor).myMatrix [y * image -> sizeX + x] = (r - 128.0F) / n;
            }
        }
    return 1;
    }

/*************************************************************************/

int CRenderer::LoadHorizont (CHorizont * hor, CPointMatrix * norms, char * filename, float n)
    {
    AUX_RGBImageRec * image = new AUX_RGBImageRec;
    image = auxDIBImageLoad (filename);
    hor -> Init (image -> sizeX, image -> sizeY);
    norms -> Init (image -> sizeX, image -> sizeY);
    for (int y = 0; y < hor -> RetY (); y ++)
        {
```

```
        for (int x = 0; x < hor -> RetX (); x ++)
            {
            float r = (float) image -> data [(y * image -> sizeX + x) * 3 + 0];
            float g = (float) image -> data [(y * image -> sizeX + x) * 3 + 1];
            float b = (float) image -> data [(y * image -> sizeX + x) * 3 + 2];
            hor -> RetElement (x, y) = (r - 128.0F) / n;
            }
        }
    return 1;
    }

/*************************************************************************/

int CRenderer::LoadGeneralINI (FILE * file)
    {
    char string [256], command [256], scrmode [20];
    int iparams [8];
    for ( ; ; )
        {
        if (feof (file) || fscanf (file, " %[-a-zA-Z0-9 ;,.=]s\n", string) == 0)
            {
            if (strcmp (scrmode, "window") == 0)
                Resize (0, 0, iparams [0], iparams [1], false);
            else if (strcmp (scrmode, "fullscreen") == 0)
                    Resize (0, 0, iparams [0], iparams [1], true);
            MakeGLWindow (GetModuleHandle (NULL), SW_HIDE, "prog1", "OpenGL window");
            SetRenderLevel (iparams [2]);
            STM -> SetNewDataType (sizeof (CColorMatrix), myMaxTexturesNum, DT_TEXTURE);
            STM -> SetNewDataType (sizeof (CLightSource), MAXLIGHTS, DT_LIGHTSOURCE);
            return 1;
            }
        if (strlen (string) == 0) continue;
        else if (string [0] == ';') continue;
            else
                {
                sscanf (string, " %[a-zA-Z]s", command);
                if (strcmp (command, "Resolution") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d, %d",
                            &iparams [0], &iparams [1]);
                    continue;
                    }
                if (strcmp (command, "ScreenMode") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %[a-z]s",
                            scrmode);
                    continue;
                    }
                if (strcmp (command, "RenderLevel") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [2]);
                    continue;
                    }
                if (strcmp (command, "RotationWay") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [3]);
                    SetRotationWay (iparams [3]);
                    continue;
                    }
                if (strcmp (command, "TransparentBufferSize") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d, %d, %d",
                            &iparams [4], &iparams [5], &iparams [6]);
                    InitTransparentBuffer (iparams [4], iparams [5], iparams [6]);
                    continue;
                    }
                if (strcmp (command, "TexturesNumber") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [7]);
                    myMaxTexturesNum = iparams [7];
                    continue;
                    }
                }
        }
    return 1;
    }

/*************************************************************************/

int CRenderer::LoadFontINI (FILE * file)
    {
    char string [256], command [256];
    char fontName [256], weight [10];
    int iparams [6];
    for ( ; ; )
        {
        if (feof (file) || fscanf (file, " %[-a-zA-Z0-9 ;,.=]s\n", string) == 0)
            {
            if (strcmp (weight, "bold") == 0)
                InitFont (fontName, iparams [0], iparams [1], iparams [2],
                          FW_BOLD, !!iparams [3], !!iparams [4], !!iparams [5]);
            else if (strcmp (weight, "normal") == 0)
                    InitFont (fontName, iparams [0], iparams [1], iparams [2],
                              FW_NORMAL, !!iparams [3], !!iparams [4], !!iparams [5]);
```

```
            return 1;
            }
        if (strlen (string) == 0) continue;
        else if (string [0] == ';') continue;
            else
                {
                sscanf (string, " %[a-zA-Z]s", command);
                if (strcmp (command, "Name") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %[a-zA-Z ]s",
                            fontName);
                    continue;
                    }
                if (strcmp (command, "Size") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [0]);
                    continue;
                    }
                if (strcmp (command, "LettersRange") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d, %d",
                            &iparams [1], &iparams [2]);
                    continue;
                    }
                if (strcmp (command, "Weight") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %s",
                            weight);
                    continue;
                    }
                if (strcmp (command, "Italic") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [3]);
                    continue;
                    }
                if (strcmp (command, "Underlined") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [4]);
                    continue;
                    }
                if (strcmp (command, "StrikeOut") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [5]);
                    continue;
                    }
                }
            }
    return 1;
    }

/*****************************************************************************/

int CRenderer::LoadAtmosphereINI (FILE * file)
    {
    char string [256], command [256];
    int iparams [4];
    float fparams [20];
    for ( ; ; )
        {
        if (feof (file) || fscanf (file, " %[-a-zA-Z0-9 ;,.=]s\n", string) == 0)
            {
            myAtmosphere.SetSkyPlane (fparams [4], fparams [5], fparams [6], fparams [7], fparams [8],
                                      fparams [0], fparams [1], fparams [2], fparams [3],
                                      iparams [0], iparams [1], iparams [2]);
            return 1;
            }
        if (strlen (string) == 0) continue;
        else if (string [0] == ';') continue;
            else
                {
                sscanf (string, " %[a-zA-Z]s", command);
                if (strcmp (command, "SkyColor") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f, %f",
                            &fparams [0], &fparams [1], &fparams [2], &fparams [3]);
                    continue;
                    }
                if (strcmp (command, "SkyPlane") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f, %f, %f",
                            &fparams [4], &fparams [5], &fparams [6], &fparams [7], &fparams [8]);
                    continue;
                    }
                if (strcmp (command, "SkyTexCoords") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d, %d",
                            &iparams [0], &iparams [1]);
                    continue;
                    }
                if (strcmp (command, "SkyTexID") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [2]);
```

```
                    continue;
                    }
                if (strcmp (command, "SkyMoveSpeed") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f",
                            &fparams [9], &fparams [10]);
                    myAtmosphere.SetSkySpeed (fparams [9], fparams [10]);
                    continue;
                    }
                if (strcmp (command, "FogColor") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f, %f",
                            &fparams [11], &fparams [12], &fparams [13], &fparams [14]);
                    SetClearColor (fparams [11], fparams [12], fparams [13], fparams [14]);
                    continue;
                    }
                if (strcmp (command, "FogParameters") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f",
                            &fparams [15], &fparams [16], &fparams [17]);
                    SetFogParameters (fparams [15], fparams [16], fparams [17]);
                    continue;
                    }
                if (strcmp (command, "VolumeFogParameters") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d, %f",
                            &iparams [3], &fparams [18]);
                    SetVolumeFogDepth (!!iparams [3], fparams [18]);
                    continue;
                    }
                }
            }
        return 1;
        }

/*****************************************************************************/

int CRenderer::LoadWaterINI (FILE * file)
    {
    char string [256], command [256];
    int iparams [6];
    float fparams [8];
    for ( ; ; )
        {
        if (feof (file) || fscanf (file, " %[-a-zA-Z0-9 ;,.=_]s\n", string) == 0)
            {
            myWater.SetColor (fparams [0], fparams [1], fparams [2], fparams [3]);
            myWater.Init (iparams [4], iparams [5], fparams [5], fparams [4]);
            myWater.InitTexture (iparams [0], iparams [1], iparams [2], iparams [3]);
            myWater.SetSpeed (fparams [6], fparams [7]);
            return 1;
            }
        if (strlen (string) == 0) continue;
        else if (string [0] == ';') continue;
            else
                {
                sscanf (string, " %[a-zA-Z]s", command);
                if (strcmp (command, "Color") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f, %f",
                            &fparams [0], &fparams [1], &fparams [2], &fparams [3]);
                    continue;
                    }
                if (strcmp (command, "TileSize") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f",
                            &fparams [4]);
                    continue;
                    }
                if (strcmp (command, "GlobalTextureID") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [0]);
                    continue;
                    }
                if (strcmp (command, "DetailTextureID") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [1]);
                    continue;
                    }
                if (strcmp (command, "TexCoords") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d, %d",
                            &iparams [2], &iparams [3]);
                    continue;
                    }
                if (strcmp (command, "VertexNumber") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d, %d",
                            &iparams [4], &iparams [5]);
                    continue;
                    }
                if (strcmp (command, "Height") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f",
                            &fparams [5]);
```

```
                    continue;
                    }
                if (strcmp (command, "Speed") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f",
                            &fparams [6], &fparams [7]);
                    continue;
                    }
                }
            }
        return 1;
        }

/*****************************************************************************/

int CRenderer::LoadCameraINI (FILE * file)
    {
    char string [256], command [256];
    int iparams [1];
    float fparams [15];
    for ( ; ; )
        {
        if (feof (file) || fscanf (file, " %[-a-zA-Z0-9 ;,.=]s\n", string) == 0)
            {
            SetCamera (fparams [0], fparams [1], fparams [2],
                       fparams [3], fparams [4], fparams [5],
                       fparams [6], fparams [7], fparams [8],
                       fparams [9], fparams [10], fparams [11], fparams [12], fparams [13], fparams [14],
!! iparams [0]);
            return 1;
            }
        if (strlen (string) == 0) continue;
        else if (string [0] == ';') continue;
            else
                {
                sscanf (string, " %[a-zA-Z]s", command);
                if (strcmp (command, "Position") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f",
                            &fparams [0], &fparams [1], &fparams [2]);
                    continue;
                    }
                if (strcmp (command, "Direction") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f",
                            &fparams [3], &fparams [4], &fparams [5]);
                    continue;
                    }
                if (strcmp (command, "UpVector") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f",
                            &fparams [6], &fparams [7], &fparams [8]);
                    continue;
                    }
                if (strcmp (command, "Frustum") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f, %f, %f, %f",
                            &fparams [9], &fparams [10], &fparams [11],
                            &fparams [12], &fparams [13], &fparams [14]);
                    continue;
                    }
                if (strcmp (command, "MouseControl") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [0]);
                    continue;
                    }
                }
            }
        return 1;
        }

/*****************************************************************************/

int CRenderer::LoadLightSourceINI (FILE * file)
    {
    char string [256], command [256];
    int iparams [2];
    float fparams [12];
    char id [20];
    for ( ; ; )
        {
        if (feof (file) || fscanf (file, " %[-a-zA-Z0-9 ;,.=_]s\n", string) == 0)
            {
            STM -> Lock (DT_LIGHTSOURCE);
            CLightSource * light = (CLightSource *) STM -> CreateObject (DT_LIGHTSOURCE);
            light -> Init (iparams [1], iparams [0]);
            light -> SetColor (fparams [0], fparams [1], fparams [2], fparams [3]);
            light -> SetPosition (fparams [4], fparams [5], fparams [6], fparams [7]);
            light -> SetDirection (fparams [8], fparams [9], fparams [10]);
            light -> SetCutoff (fparams [11]);
            SetLight (light);
            STM -> Unlock (DT_LIGHTSOURCE);
            return 1;
            }
        if (strlen (string) == 0) continue;
        else if (string [0] == ';') continue;
```

```
            else
                {
                sscanf (string, " %[a-zA-Z]s", command);
                if (strcmp (command, "Color") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f, %f",
                            &fparams [0], &fparams [1], &fparams [2], &fparams [3]);
                    continue;
                    }
                if (strcmp (command, "Position") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f, %f",
                            &fparams [4], &fparams [5], &fparams [6], &fparams [7]);
                    continue;
                    }
                if (strcmp (command, "Direction") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f",
                            &fparams [8], &fparams [9], &fparams [10]);
                    continue;
                    }
                if (strcmp (command, "Identificator") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %s", id);
                    if (strcmp (id, "GL_LIGHT0") == 0) iparams [0] = GL_LIGHT0;
                    if (strcmp (id, "GL_LIGHT1") == 0) iparams [0] = GL_LIGHT1;
                    if (strcmp (id, "GL_LIGHT2") == 0) iparams [0] = GL_LIGHT2;
                    if (strcmp (id, "GL_LIGHT3") == 0) iparams [0] = GL_LIGHT3;
                    if (strcmp (id, "GL_LIGHT4") == 0) iparams [0] = GL_LIGHT4;
                    if (strcmp (id, "GL_LIGHT5") == 0) iparams [0] = GL_LIGHT5;
                    if (strcmp (id, "GL_LIGHT6") == 0) iparams [0] = GL_LIGHT6;
                    if (strcmp (id, "GL_LIGHT7") == 0) iparams [0] = GL_LIGHT7;
                    continue;
                    }
                if (strcmp (command, "Type") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [1]);
                    continue;
                    }
                if (strcmp (command, "Cutoff") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f",
                            &fparams [11]);
                    continue;
                    }
                }
            }
        return 1;
        }

/*****************************************************************************/

int CRenderer::LoadTextureINI (FILE * file)
    {
    char string [256], command [256];
    int iparams [4];
    char filename [20];
    for ( ; ; )
        {
        if (feof (file) || fscanf (file, " %[-a-zA-Z0-9 ;,.=_]s\n", string) == 0)
            {
            AssertT (STM -> RetHandle (DT_TEXTURE) != -1, "Textures aren't initialized");
            STM -> Lock (DT_TEXTURE);
            CColorMatrix * tex = (CColorMatrix *) STM -> CreateObject (DT_TEXTURE);
            tex -> SetNewID (iparams [0]);
            if (iparams [1] == -1 || iparams [2] == -1 || iparams [3] == -1)
                LoadTexture (tex, filename);
            else
                {
                LoadTexture (tex, filename, (char) iparams [1], (char) iparams [2], (char) iparams [3]);
                if (iparams [0] != 0) myBuffer.AddTranspTex (iparams [0]);
                }
            if (iparams [0] != 0) InitTex (*tex);
            STM -> Unlock (DT_TEXTURE);
            return 1;
            }
        if (strlen (string) == 0) continue;
        else if (string [0] == ';') continue;
            else
                {
                sscanf (string, " %[a-zA-Z]s", command);
                if (strcmp (command, "File") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %s", filename);
                    continue;
                    }
                if (strcmp (command, "ID") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [0]);
                    continue;
                    }
                if (strcmp (command, "TransparentColor") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d, %d, %d",
                            &iparams [1], &iparams [2], &iparams [3]);
```

```
                    continue;
                    }
                }
        }
    return 1;
    }

/*****************************************************************************/

int CRenderer::LoadLandscapeINI (FILE * file)
    {
    char string [256], command [256];
    int iparams [2];
    for ( ; ; )
        {
        if (feof (file) || fscanf (file, " %[-a-zA-Z0-9 ;,.=_]s\n", string) == 0)
            {
            STM -> SetNewDataType (sizeof (CLandscape), 1, DT_LANDSCAPE);
            STM -> Lock (DT_LANDSCAPE);
            CLandscape * land = (CLandscape *) STM -> CreateObject (DT_LANDSCAPE);
            land -> Init (iparams [0]);
            STM -> Unlock (DT_LANDSCAPE);
            return 1;
            }
        if (strlen (string) == 0) continue;
        else if (string [0] == ';') continue;
            else
                {
                sscanf (string, " %[a-zA-Z]s", command);
                if (strcmp (command, "Horizonts") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [0]);
                    continue;
                    }
                if (strcmp (command, "LoDLevel") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [1]);
                    SetLoDLevel (iparams [1]);
                    continue;
                    }
                }
        }
    return 1;
    }

/*****************************************************************************/

int CRenderer::LoadHorizontINI (FILE * file)
    {
    char string [256], command [256];
    int iparams [4];
    float fparams [6];
    char filename [20];
    for ( ; ; )
        {
        if (feof (file) || fscanf (file, " %[-a-zA-Z0-9 ;,.=_]s\n", string) == 0)
            {
            AssertT (STM -> RetHandle (DT_LANDSCAPE) != -1, "Landscape is not initialized");
            STM -> Lock (DT_LANDSCAPE);
            CLandscape * land = (CLandscape *) STM -> RetObject (DT_LANDSCAPE, 0);

            int index = land -> InitNewHor ();
            land -> SetColor (index, fparams [0], fparams [1], fparams [2], fparams [3]);
            land -> SetGlobalTexID (index, iparams [0]);
            land -> SetDetailTexID (index, iparams [1]);
            land -> SetSquareSize (index, fparams [4]);
            land -> SetTexParameters (index, (float) iparams [2], (float) iparams [3]);
            LoadHorizont (land, index, filename, fparams [5]);
            ComputeNormals (land -> RetHorizont (index), land -> RetNormalMap (index));

            STM -> Lock (DT_LANDSCAPE);
            return 1;
            }
        if (strlen (string) == 0) continue;
        else if (string [0] == ';') continue;
            else
                {
                sscanf (string, " %[a-zA-Z]s", command);
                if (strcmp (command, "HeightMap") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %s", filename);
                    continue;
                    }
                if (strcmp (command, "Color") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f, %f, %f, %f",
                            &fparams [0], &fparams [1], &fparams [2], &fparams [3]);
                    continue;
                    }
                if (strcmp (command, "TileSize") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f",
                            &fparams [4]);
                    continue;
                    }
```

```
                if (strcmp (command, "HeightDelta") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %f",
                            &fparams [5]);
                    continue;
                    }
                if (strcmp (command, "GlobalTextureID") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [0]);
                    continue;
                    }
                if (strcmp (command, "DetailTextureID") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d",
                            &iparams [1]);
                    continue;
                    }
                if (strcmp (command, "TexCoords") == 0)
                    {
                    sscanf (&string [strlen (command)], " = %d, %d",
                            &iparams [2], &iparams [3]);
                    continue;
                    }
                }
        }
    return 1;
    }

/*****************************************************************************/

int CRenderer::LoadINI (char * filename)
    {
    FILE * file = fopen (filename, "rt");
    char initype [50], string [256];
    for ( ; !feof (file) ; )
        {
        if (fgets (string, 256, file) == 0) break;
        if (string [0] == 13) continue;
        else if (string [0] == ';') continue;
            else if (string [0] == '[')
                    {
                    sscanf (string, "[%[a-zA-Z]s\]", initype);
                    if (strcmp (initype, "General") == 0)
                        {
                        LoadGeneralINI (file);
                        continue;
                        }
                    if (strcmp (initype, "Font") == 0)
                        {
                        LoadFontINI (file);
                        continue;
                        }
                    if (strcmp (initype, "Atmosphere") == 0)
                        {
                        LoadAtmosphereINI (file);
                        continue;
                        }
                    if (strcmp (initype, "Water") == 0)
                        {
                        LoadWaterINI (file);
                        continue;
                        }
                    if (strcmp (initype, "Camera") == 0)
                        {
                        LoadCameraINI (file);
                        continue;
                        }
                    if (strcmp (initype, "LightSource") == 0)
                        {
                        LoadLightSourceINI (file);
                        continue;
                        }
                    if (strcmp (initype, "Texture") == 0)
                        {
                        LoadTextureINI (file);
                        continue;
                        }
                    if (strcmp (initype, "Landscape") == 0)
                        {
                        LoadLandscapeINI (file);
                        continue;
                        }
                    if (strcmp (initype, "Horizont") == 0)
                        {
                        LoadHorizontINI (file);
                        continue;
                        }
                    }
        }
    fclose (file);
    return 1;
    }

/*****************************************************************************/
```

```
int CRenderer::SetCamera (float ex, float ey, float ez, float lx, float ly, float lz, float ux, float uy,
float uz,
                          float x1, float x2, float y1, float y2, float nz, float fz, bool
isrotatefrommouse)
    {
    AssertT (myCameraNum < MAXCAMERAS, "Too many cameras");
    myCameras [myCameraNum].SetFrustum (x1, x2, y1, y2, nz, fz);
    myCameras [myCameraNum].Set (ex, ey, ez, lx, ly, lz, ux, uy, uz);
    myCameras [myCameraNum].myIsRotateFromMouse = isrotatefrommouse;
    myCameraNum ++;
    return myCameraNum;
    }

/*****************************************************************************/

void CRenderer::MakeCameraCurrent (int n)
    {
    AssertT (n - 1 < myCameraNum, "Can't make camera current");
    myCurrentCamera = n;
    myCameras [myCurrentCamera - 1].MakeCurrent ();
    }

/*****************************************************************************/

int CRenderer::RetCurCameraNum ()
    {
    return myCurrentCamera;
    }

/*****************************************************************************/

void CRenderer::CameraSlide (float k)
    {
    myCameras [myCurrentCamera - 1].Slide (k);
    }

/*****************************************************************************/

void CRenderer::CameraRotate (float up, float z)
    {
    myCameras [myCurrentCamera - 1].RotateUp (up);
    myCameras [myCurrentCamera - 1].RotateZ  (z);
    }

/*****************************************************************************/

void CRenderer::CameraSeek (CPoint3D &point)
    {
    myCameras [myCurrentCamera - 1].Seek (point);
    }

/*****************************************************************************/

void CRenderer::CameraFollow (CPoint3D &point)
    {
    myCameras [myCurrentCamera - 1].Follow (point);
    }

/*****************************************************************************/

void CRenderer::CameraMove (float k)
    {
    CPoint3D target = myCameras [myCurrentCamera - 1].myDirection +
                      myCameras [myCurrentCamera - 1].mySpeed * k;
    myCameras [myCurrentCamera - 1].Follow (target);
    }

/*****************************************************************************/

void CRenderer::CameraSetPosition (float x, float y, float z)
    {
    myCameras [myCurrentCamera - 1].SetPosition (x, y, z);
    }

/*****************************************************************************/

void CRenderer::CameraSlide (int n, float k)
    {
    AssertT (n - 1 < myCameraNum, "Can't slide camera");
    myCameras [n - 1].Slide (k);
    myCameras [myCurrentCamera - 1].MakeCurrent ();
    }

/*****************************************************************************/

void CRenderer::CameraRotate (int n, float up, float z)
    {
    AssertT (n - 1 < myCameraNum, "Can't slide camera");
    myCameras [n - 1].RotateUp (up);
    myCameras [n - 1].RotateZ  (z);
    myCameras [myCurrentCamera - 1].MakeCurrent ();
    }

/*****************************************************************************/

void CRenderer::CameraSeek (int n, CPoint3D &point)
    {
```

```
    AssertT (n - 1 < myCameraNum, "Can't slide camera");
    myCameras [n - 1].Seek (point);
    myCameras [myCurrentCamera - 1].MakeCurrent ();
    }

/*************************************************************************/

void CRenderer::CameraFollow (int n, CPoint3D &point)
    {
    AssertT (n - 1 < myCameraNum, "Can't slide camera");
    myCameras [n - 1].Follow (point);
    myCameras [myCurrentCamera - 1].MakeCurrent ();
    }

/*************************************************************************/

void CRenderer::CameraMove (int n, float k)
    {
    AssertT (n - 1 < myCameraNum, "Can't move camera");
    CPoint3D target = myCameras [n - 1].myDirection +
                      myCameras [n - 1].mySpeed * k;
    myCameras [n - 1].Follow (target);
    myCameras [myCurrentCamera - 1].MakeCurrent ();
    }

/*************************************************************************/

void CRenderer::CameraSetPosition (int n, float x, float y, float z)
    {
    AssertT (n - 1 < myCameraNum, "Can't set camera position");
    myCameras [n - 1].SetPosition (x, y, z);
    myCameras [myCurrentCamera - 1].MakeCurrent ();
    }

/*************************************************************************/

void CRenderer::CameraOnLand (int n, CHorizont * hor)
    {
    AssertT (n - 1 < myCameraNum, "Can't slide camera");
    float height   = RetHorizontHeight (*hor, myCameras [n - 1].myPosition (0), myCameras [n -
1].myPosition (1)) + 1.0F;
    CPoint3D delta = myCameras [n - 1].myDirection - myCameras [n - 1].myPosition;
    if (myCameras [n - 1].myPosition (2) < height)
        {
        myCameras [n - 1].SetPosition (myCameras [n - 1].myPosition (0), myCameras [n - 1].myPosition (1),
height);
        myCameras [n - 1].myDirection = myCameras [n - 1].myPosition + delta;
        myCameras [n - 1].CountSpeed ();
        myCameras [n - 1].CountPoints ();
        }
    myCameras [myCurrentCamera - 1].MakeCurrent ();
    }

/*************************************************************************/

void CRenderer::SetMaterial (CGraphMaterial &material)
    {
    myBufMaterial = material;
    glMaterialf (GL_FRONT_AND_BACK, GL_SHININESS, material.RetShininess ());
    if (material.RetColorSource () == COLOR_FROM_MATERIAL) SetColorCurrent (material.myRGBA);
    }

/*************************************************************************/

void CRenderer::SetColorCurrent (float * color)
    {
    if (myRenderLevel >= RL_LIGHTENED)
        {
        glMaterialfv (GL_FRONT_AND_BACK, GL_AMBIENT, myDefaultAmbient);
        if (myBufMaterial.RetEmission () == 1.0F) glMaterialfv (GL_FRONT_AND_BACK, GL_EMISSION, color);
        glMaterialfv (GL_FRONT_AND_BACK, GL_DIFFUSE, color);
        }
    else
        {
        glColor4fv (color);
        }
    }

/*************************************************************************/

void CRenderer::SetColorCurrent (CColor &color)
    {
    SetColorCurrent (color.RetPointer ());
    }

/*************************************************************************/

void CRenderer::SetColorOfLightCurrent (CColor &color, GLenum nlight)
    {
    glLightfv (nlight, GL_AMBIENT,  color.RetPointer ());
    glLightfv (nlight, GL_DIFFUSE,  color.RetPointer ());
    glLightfv (nlight, GL_SPECULAR, color.RetPointer ());
    }

/*************************************************************************/

void CRenderer::SetPosOfLightCurrent (CColor &point, GLenum nlight)
```

```
    {
    glLightfv (nlight, GL_POSITION, point.RetPointer ());
    }

/*************************************************************************/

void CRenderer::SetDirOfLightCurrent (CPoint3D &point, GLenum nlight)
    {
    glLightfv (nlight, GL_SPOT_DIRECTION, point.RetPointer ());
    }

/*************************************************************************/

void CRenderer::SetCutoffOfLightCurrent (float cutoff, GLenum nlight)
    {
    glLightf (nlight, GL_SPOT_CUTOFF, cutoff);
    }

/*************************************************************************/

void CRenderer::ResetLight (CLightSource * light)
    {
    SetPosOfLightCurrent (light -> myPosition, light -> myLightNum);
    SetColorOfLightCurrent (light -> myColor, light -> myLightNum);
    if (light -> myLightType == LIGHT_SPOT)
        {
        SetDirOfLightCurrent (light -> myDirection, light -> myLightNum);
        SetCutoffOfLightCurrent (light -> myParam, light -> myLightNum);
        }
    }

/*************************************************************************/

void CRenderer::SetLight (CLightSource * light)
    {
    AssertT (myCurLight < MAXLIGHTS, "Can't init light source");
    MakeCameraCurrent (myCurrentCamera);
    glEnable (light -> myLightNum);
    ResetLight (light);
    myGlobalLights [myCurLight] = light;
    myCurLight ++;
    }

/*************************************************************************/

void CRenderer::TurnOffLight (CLightSource * light)
    {
    glDisable (light -> myLightNum);
    }

/*************************************************************************/

void CRenderer::SetLights (CSmartPointer &lights)
    {
    MakeCurrent ();
    for (int n = 0; n < lights.RetObjNum (); n ++)
        if (lights.RetObject (n) != NULL )
            SetLight ((CLightSource *) lights.RetObject (n));
    }

/*************************************************************************/

void CRenderer::SetLights (CLightSource * lights, int nlights)
    {
    MakeCurrent ();
    for (int n = 0; n < nlights; n ++)
        SetLight (&lights [n]);
    }

/*************************************************************************/

void CRenderer::ResetLights (CLightSource * lights, int nlights)
    {
    for (int n = 0; n < nlights; n ++)
        ResetLight (&lights [n]);
    }

/*************************************************************************/

void CRenderer::ResetLights (CLightSource ** lights, int nlights)
    {
    for (int n = 0; n < nlights; n ++)
        ResetLight (lights [n]);
    }

/*************************************************************************/

void CRenderer::ResetLights ()
    {
    ResetLights (myGlobalLights, myCurLight);
    }

/*************************************************************************/

void CRenderer::TurnOffLights (CLightSource * lights, int nlights)
    {
    for (int n = 0; n < nlights; n ++)
```

```
        TurnOffLight (&lights [n]);
    }

/*****************************************************************************/

void CRenderer::EnableLighting ()
    {
    glEnable (GL_LIGHTING);
    }

/*****************************************************************************/

void CRenderer::DisableLighting ()
    {
    glDisable (GL_LIGHTING);
    }

/*****************************************************************************/

void CRenderer::TransformMesh (CMesh * mesh, CMesh * mesh2, CMatrix4x4 &m)
    {
    for (int n = 0; n < mesh -> RetCurPolys (); n ++)
        {
        for (int i = 0; i < 3; i ++)
            {
            mesh2 -> RetPoly (n) -> retvert (i).RetPoint () =
                m.DotProduct (mesh -> RetPoly (n) -> retvert (i).RetPoint ());
            mesh2 -> RetPoly (n) -> retvert (i).RetNormal () =
                m.DotProduct (mesh -> RetPoly (n) -> retvert (i).RetNormal ());
            }
        }
    mesh -> RetTransformMatrix () = mesh2 -> RetTransformMatrix ();
    }

/*****************************************************************************/

void CRenderer::TransformMeshes (CMesh * mesh, CMesh * mesh2, int num)
    {
    for (int n = 0; n < num; n ++)
        {
        TransformMesh (&mesh [n], &mesh2 [n], mesh2 [n].RetTransformMatrix ());
        }
    }

/*****************************************************************************/

void CRenderer::TransformMeshes (CSmartPointer &meshes, CSmartPointer &meshes2)
    {
    for (int n = 0; n < meshes.RetObjNum (); n ++)
        {
        TransformMesh ((CMesh *) meshes.RetObject (n), (CMesh *) meshes2.RetObject (n),
                       ((CMesh *) meshes2.RetObject (n)) -> RetTransformMatrix ());
        }
    }

/*****************************************************************************/

void CRenderer::RenderSprites (CSmartPointer &sprites)
    {
    for (int n = 0; n < sprites.RetObjNum (); n ++)
        {
        if (sprites.RetObject (n) != NULL)
            RenderSprite ((CColorMatrix *) sprites.RetObject (n));
        }
    }

/*****************************************************************************/

void CRenderer::RenderSprites (void * sprites, int nsprites)
    {
    RenderSprites (reinterpret_cast <CColorMatrix *> (sprites), nsprites);
    }

/*****************************************************************************/

void CRenderer::RenderSprites (CColorMatrix * sprites, int nsprites)
    {
    for (int n = 0; n < nsprites; n ++)
        {
        RenderSprite (&sprites [n]);
        }
    }

/*****************************************************************************/

void CRenderer::RenderSprite (CColorMatrix * sprite)
    {
    glRasterPos3f (sprite -> RetPosition () (0),
                   sprite -> RetPosition () (1),
                   sprite -> RetPosition () (2));
    glDrawPixels (sprite -> RetX (), sprite -> RetY (), GL_RGBA,
                  GL_FLOAT, sprite -> RetPointer (0));
    }

/*****************************************************************************/

void CRenderer::RenderPictures (CSmartPointer &pictures)
```

```
    {
    for (int i = 0; i < pictures.RetObjNum (); i ++)
        if (pictures.RetObject (i) != NULL)
            RenderPicture ((CPicture *) pictures.RetObject (i));
    }

/*************************************************************************/

void CRenderer::RenderPictures (void * pictures, int n)
    {
    RenderPictures (reinterpret_cast <CPicture *> (pictures), n);
    }

/*************************************************************************/

void CRenderer::RenderPictures (CPicture * pictures, int n)
    {
    for (int i = 0; i < n; i ++)
        RenderPicture (&pictures [i]);
    }

/*************************************************************************/

void CRenderer::RenderPicture (CPicture * pic)
    {
    SetColorCurrent (pic -> RetColor ());
    glRasterPos3f (pic -> RetPos () (0), pic -> RetPos () (1), pic -> RetPos () (2));
    glBitmap (pic -> RetSizeX (), pic -> RetSizeY (), 0.0F, 0.0F, 1.0F, 0.0F,
              (unsigned char *) pic -> RetPointer ());
    }

/*************************************************************************/

void CRenderer::RenderBillBoards (CSmartPointer &boards)
    {
    for (int i = 0; i < boards.RetObjNum (); i ++)
        if (boards.RetObject (i) != NULL)
            RenderBillBoard ((CBillBoard *) boards.RetObject (i));
    }

/*************************************************************************/

void CRenderer::RenderBillBoards (void * boards, int n)
    {
    RenderBillBoards (reinterpret_cast <CBillBoard *> (boards), n);
    }

/*************************************************************************/

void CRenderer::RenderBillBoards (CBillBoard * boards, int n)
    {
    for (int i = 0; i < n; i ++)
        RenderBillBoard (&boards [i]);
    }

/*************************************************************************/

void CRenderer::RenderBillBoard (CBillBoard * board)
    {
    if (myBuffer.IfTransparent (board -> myTexID) || board -> myColor.myRGBA [3] != 1.0F)
        {
        if (myBufferWriteMode)
            if (myBuffer.AddBillBoard (board)) return;
        }

    CPoint3D up (0.0F, 0.0F, 1.0F), dir (0.0F, 0.0F, 0.0F),
             ox (0.0F, 0.0F, 0.0F), oy (0.0F, 0.0F, 0.0F);
    dir = (board -> RetCenter () - myCameras [myCurrentCamera - 1].myPosition) * board -> myFreedomVector;
    dir.Normalize ();
    ox = (dir).CrossProduct (up);
    ox.Normalize ();
    oy = ox.CrossProduct (dir);
    oy.Normalize ();
    CPoint3D ul = ox * board -> RetMinX () + oy * board -> RetMaxY (),
             ur = ox * board -> RetMaxX () + oy * board -> RetMaxY (),
             bl = ox * board -> RetMinX () + oy * board -> RetMinY (),
             br = ox * board -> RetMaxX () + oy * board -> RetMinY ();

    glPushMatrix ();
    glTranslatef (board -> RetCenter () (0), board -> RetCenter () (1), board -> RetCenter () (2));
    if (myRenderLevel >= RL_LIGHTENED) DisableLighting ();
    glColor4fv (board -> myColor.myRGBA);
    ExtSetMultiTex (board -> RetTexID (), 0, 0, 0);
    if (myRenderLevel >= RL_SOLID) RenderSolidBoard (ul, ur, bl, br);
    else RenderWireBoard (ul, ur, bl, br);
    if (myRenderLevel >= RL_LIGHTENED) EnableLighting ();
    glPopMatrix ();
    ExtDisableMultiTex ();
    }

/*************************************************************************/

void CRenderer::RenderSolidBoard (CPoint3D &ul, CPoint3D &ur, CPoint3D &bl, CPoint3D &br)
    {
    glBegin (GL_QUADS);

    glTexCoord2f (0.0F, 0.0F); glVertex3f (ul (0), ul (1), ul (2));
```

```
    glTexCoord2f (1.0F, 0.0F); glVertex3f (ur (0), ur (1), ur (2));
    glTexCoord2f (1.0F, 1.0F); glVertex3f (br (0), br (1), br (2));
    glTexCoord2f (0.0F, 1.0F); glVertex3f (bl (0), bl (1), bl (2));

    glEnd ();
    }

/*************************************************************************/

void CRenderer::RenderWireBoard (CPoint3D &ul, CPoint3D &ur, CPoint3D &bl, CPoint3D &br)
    {
    glBegin (GL_LINES);

    glVertex3f (ul (0), ul (1), ul (2)); glVertex3f (ur (0), ur (1), ur (2));
    glVertex3f (ur (0), ur (1), ur (2)); glVertex3f (br (0), br (1), br (2));
    glVertex3f (br (0), br (1), br (2)); glVertex3f (bl (0), bl (1), bl (2));
    glVertex3f (bl (0), bl (1), bl (2)); glVertex3f (ul (0), ul (1), ul (2));

    glEnd ();
    }

/*************************************************************************/

void CRenderer::RenderLandscape (CSmartPointer &landscape)
    {
    RenderLandscape (landscape.RetObject (0));
    }

/*************************************************************************/

void CRenderer::RenderLandscape (void * landscape)
    {
    RenderLandscape (*(reinterpret_cast <CLandscape *> (landscape)));
    }

/*************************************************************************/

void CRenderer::RenderLandscape (CLandscape &landscape)
    {
    glEnable (GL_CULL_FACE);
    for (int n = 0; n < landscape.RetSize (); n ++)
        {
        RenderHorizont (landscape.RetHorizont (n), landscape.RetNormalMap (n));
        }
    glDisable (GL_CULL_FACE);
    }

/*************************************************************************/

void CRenderer::RenderHorizont (CHorizont &horizont, CPointMatrix &norm)
    {
    if (myRenderLevel >= RL_SOLID) RenderSolidHor (horizont, norm);
    else RenderWireHor (horizont, norm);
    }

/*************************************************************************/

void CRenderer::RenderSolidHor (CHorizont &horizont, CPointMatrix &norm)
    {
    float sqrsize = horizont.RetSquareSize (),
          rx = horizont.RetTexX (),
          ry = horizont.RetTexY ();
    if (horizont.RetDetailTexID () != 0)
        {
        rx = 1.0F; ry = 1.0F;
        }
    int sizex = horizont.RetX () - 1, sizey = horizont.RetY () - 1,
        startx = 0, starty = 0, endx = 0, endy = 0, nx = 0, ny = 0, stripSize = 0, allSize = 0;
    int lodAxis = SetLoDParams (horizont, sizex, sizey, sqrsize, startx, starty, endx, endy, nx, ny,
stripSize, allSize);
    glMaterialf (GL_FRONT_AND_BACK, GL_SHININESS, 0.0F);
    SetColorCurrent (horizont.RetColor ());
    if (horizont.myColor [3] != 1.0F) glDepthMask (GL_FALSE);
    ExtSetMultiTex (horizont.RetGlobalTexID (), horizont.RetDetailTexID (),
                    horizont.RetTexX (), horizont.RetTexY ());

    if (lodAxis == LOD_AXIS_Y) RenderSolidLoDAlongY (horizont, norm, sizex, sizey, sqrsize, rx, ry,
startx, starty, endx, endy, nx, ny, stripSize, allSize);
    else if (lodAxis == LOD_AXIS_X) RenderSolidLoDAlongX (horizont, norm, sizex, sizey, sqrsize, rx, ry,
startx, starty, endx, endy, nx, ny, stripSize, allSize);
    else if (lodAxis == LOD_GLOBAL) RenderSolidLoDGlobal (horizont, norm, sizex, sizey, sqrsize, rx, ry,
startx, starty, endx, endy, nx, ny, stripSize, allSize);

    glFlush ();
    glBindTexture (GL_TEXTURE_2D, 0);
    ExtDisableMultiTex ();
    if (horizont.myColor [3] != 1.0F) glDepthMask (GL_TRUE);
    }

/*************************************************************************/

void CRenderer::RenderSolidLoDAlongX (CHorizont &horizont, CPointMatrix &norm, int sizex, int sizey, float
sqrsize, float rx, float ry,
                                      int startx, int starty, int endx, int endy, int nx, int ny, int
stripSize, int allSize)
    {
    int inverseRendering = 0, div = 0, startLod = 0, endLod = 0;
```

```
    div = CountLoDDiv () / myLoDLevel;

    glFrontFace (GL_CW);
    for (int n = 0; n <= div; n ++)
        {
        if (startx < endx)
            {
            if (n < div)
                {
                startLod = (startx + n * stripSize) / nx * nx;
                if (startLod > endx) continue;
                endLod = (startx + (n + 1) * stripSize) / (nx * 2) * (nx * 2);
                if (endLod > endx)
                    {
                    endLod = endx / nx * nx;
                    }
                }
            else
                {
                nx /= 2; ny /= 2;
                startLod = endx;
                endLod = startx + allSize;
                }
            }
        else
            {
            if (n < div)
                {
                startLod = (startx - n * stripSize) / nx * nx;
                if (startLod < endx) continue;
                endLod = (startx - (n + 1) * stripSize) / (nx * 2) * (nx * 2);
                if (endLod < endx)
                    {
                    endLod = endx / nx * nx;
                    }
                }
            else
                {
                nx /= 2; ny /= 2;
                startLod = endx;
                endLod = startx - allSize;
                }
            }
        if (startLod != startx && n != div && abs (startLod - endLod) >= abs (nx))
            {
            RenderSolidStepStripX (horizont, norm, sizex, sizey, sqrsize, rx, ry, startLod, starty, endy,
nx / 2, ny / 2);
            startLod += nx;
            }
        for (int x = startLod; ; x += nx)
            {
            if (nx > 0 && x > endLod - nx) break;
            else if (nx < 0 && x < endLod - nx) break;
            if (nx < 0 || ny < 0)
                if (!(nx < 0 && ny < 0))
                    {
                    x += nx; nx = -nx;
                    inverseRendering = 1;
                    }

            glBegin (GL_TRIANGLE_STRIP);
            for (int y = starty; ; y += ny)
                {
                if (ny > 0 && y > endy) break;
                else if (ny < 0 && y < endy) break;
                ExtSetFogCoord (horizont.myMatrix [y * (sizex + 1) + x + nx]);
                glNormal3fv (&(norm.myMatrix [(y * (sizex + 1) + x + nx) * FLOATSINPOINT]));
                ExtMultiTexCoords ((1.0F / sizex) * (x + ny) * rx, (1.0F / sizey) * y * ry);
                glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                            (y - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [y * (sizex + 1) + x + nx]);
                ExtSetFogCoord (horizont.myMatrix [y * (sizex + 1) + x]);
                glNormal3fv (&(norm.myMatrix [(y * (sizex + 1) + x) * FLOATSINPOINT]));
                ExtMultiTexCoords ((1.0F / sizex) * x * rx, (1.0F / sizey) * y * ry);
                glVertex3f ((x - (sizex / 2)) * sqrsize,
                            (y - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [y * (sizex + 1) + x]);
                }

            if (inverseRendering)
                {
                x += nx; nx = -nx;
                inverseRendering = 0;
                }
            glEnd ();
            }
        nx *= 2; ny *= 2;
        }
    glFrontFace (GL_CCW);
    }

/*****************************************************************************/

void CRenderer::RenderSolidStepStripX (CHorizont &horizont, CPointMatrix &norm, int sizex, int sizey,
float sqrsize,
                                       float rx, float ry, int startx, int starty, int endy, int nx, int
ny)
```

```
    {
    int x = startx;
    int index = 0;
    glLightModeli (GL_LIGHT_MODEL_TWO_SIDE, GL_FALSE);
    glDisable (GL_CULL_FACE);

    for (int y = starty; y <= endy - ny * 2; y += ny * 2)
        {
        glBegin (GL_TRIANGLE_STRIP);

        index = (y + ny * 2) * (sizex + 1) + x + nx * 2;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x + nx * 2) * rx, (1.0F / sizey) * (y + ny * 2) * ry);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y + ny * 2) * (sizex + 1) + x;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x) * rx, (1.0F / sizey) * (y + ny * 2) * ry);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y + ny) * (sizex + 1) + x + nx;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x + nx) * rx, (1.0F / sizey) * (y + ny) * ry);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y + ny) * (sizex + 1) + x;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x) * rx, (1.0F / sizey) * (y + ny) * ry);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y) * (sizex + 1) + x;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x) * rx, (1.0F / sizey) * (y) * ry);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y + ny) * (sizex + 1) + x + nx;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x + nx) * rx, (1.0F / sizey) * (y + ny) * ry);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y) * (sizex + 1) + x + nx * 2;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x + nx * 2) * rx, (1.0F / sizey) * (y) * ry);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y + ny * 2) * (sizex + 1) + x + nx * 2;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x + nx * 2) * rx, (1.0F / sizey) * (y + ny * 2) * ry);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);
        glEnd ();
        }

    glEnable (GL_CULL_FACE);
    glLightModeli (GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
    }

/*****************************************************************************/

void CRenderer::RenderSolidLoDAlongY (CHorizont &horizont, CPointMatrix &norm, int sizex, int sizey, float
sqrsize, float rx, float ry,
                                      int startx, int starty, int endx, int endy, int nx, int ny, int
stripSize, int allSize)
    {
    int inverseRendering = 0, div = 0, startLod = 0, endLod = 0;
    div = CountLoDDiv () / myLoDLevel;

    for (int n = 0; n <= div; n ++)
        {
        if (starty < endy)
            {
            if (n < div)
                {
                startLod = (starty + n * stripSize) / ny * ny;
```

```
                if (startLod > endy) continue;
                endLod = (starty + (n + 1) * stripSize) / (ny * 2) * (ny * 2);
                if (endLod > endy)
                    {
                    endLod = endy / ny * ny;
                    }
                }
            else
                {
                nx /= 2; ny /= 2;
                startLod = endy;
                endLod = starty + allSize;
                }
            }
        else
            {
            if (n < div)
                {
                startLod = (starty - n * stripSize) / ny * ny;
                if (startLod < endy) continue;
                endLod = (starty - (n + 1) * stripSize) / (ny * 2) * (ny * 2);
                if (endLod < endy)
                    {
                    endLod = endy / ny * ny;
                    }
                }
            else
                {
                nx /= 2; ny /= 2;
                startLod = endy;
                endLod = starty - allSize;
                }
            }
        if (startLod != starty && n != div && abs (startLod - endLod) >= abs (ny))
            {
            RenderSolidStepStripY (horizont, norm, sizex, sizey, sqrsize, rx, ry, startx, startLod, endx,
nx / 2, ny / 2);
            startLod += ny;
            }
        for (int y = startLod; ; y += ny)
            {
            if (ny > 0 && y > endLod - ny) break;
            else if (ny < 0 && y < endLod - ny) break;
            if (nx < 0 || ny < 0)
                if (!(nx < 0 && ny < 0))
                    {
                    y += ny; ny = -ny;
                    inverseRendering = 1;
                    }

            glBegin (GL_TRIANGLE_STRIP);
            for (int x = startx; ; x += nx)
                {
                if (nx > 0 && x > endx) break;
                else if (nx < 0 && x < endx) break;
                ExtSetFogCoord (horizont.myMatrix [(y + ny) * (sizex + 1) + x]);
                glNormal3fv (&(norm.myMatrix [((y + ny) * (sizex + 1) + x) * FLOATSINPOINT]));
                ExtMultiTexCoords ((1.0F / sizex) * x * rx, (1.0F / sizey) * (y + ny) * ry);
                glVertex3f ((x - (sizex / 2)) * sqrsize,
                            (y + ny - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [(y + ny) * (sizex + 1) + x]);

                ExtSetFogCoord (horizont.myMatrix [y * (sizex + 1) + x]);
                glNormal3fv (&(norm.myMatrix [(y * (sizex + 1) + x) * FLOATSINPOINT]));
                ExtMultiTexCoords ((1.0F / sizex) * x * rx, (1.0F / sizey) * y * ry);
                glVertex3f ((x - (sizex / 2)) * sqrsize,
                            (y - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [y * (sizex + 1) + x]);
                }

            if (inverseRendering)
                {
                y += ny; ny = -ny;
                inverseRendering = 0;
                }
            glEnd ();
            }
        nx *= 2; ny *= 2;
        }
    }

/*****************************************************************************/

void CRenderer::RenderSolidStepStripY (CHorizont &horizont, CPointMatrix &norm, int sizex, int sizey,
float sqrsize,
                                       float rx, float ry, int startx, int starty, int endx, int nx, int
ny)
    {
    int y = starty;
    int index = 0;
    glLightModeli (GL_LIGHT_MODEL_TWO_SIDE, GL_FALSE);
    glDisable (GL_CULL_FACE);

    for (int x = startx; x <= endx - nx * 2; x += nx * 2)
        {
        glBegin (GL_TRIANGLE_STRIP);
```

```
        index = (y + ny * 2) * (sizex + 1) + x;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x) * rx, (1.0F / sizey) * (y + ny * 2) * ry);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y) * (sizex + 1) + x;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x) * rx, (1.0F / sizey) * (y) * ry);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y + ny) * (sizex + 1) + x + nx;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x + nx) * rx, (1.0F / sizey) * (y + ny) * ry);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y) * (sizex + 1) + x + nx;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x + nx) * rx, (1.0F / sizey) * (y) * ry);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y) * (sizex + 1) + x + nx * 2;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x + nx * 2) * rx, (1.0F / sizey) * (y) * ry);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y + ny) * (sizex + 1) + x + nx;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x + nx) * rx, (1.0F / sizey) * (y + ny) * ry);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y + ny * 2) * (sizex + 1) + x + nx * 2;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x + nx * 2) * rx, (1.0F / sizey) * (y + ny * 2) * ry);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        index = (y + ny * 2) * (sizex + 1) + x;
        ExtSetFogCoord (horizont.myMatrix [index]);
        glNormal3fv (&(norm.myMatrix [index * FLOATSINPOINT]));
        ExtMultiTexCoords ((1.0F / sizex) * (x) * rx, (1.0F / sizey) * (y + ny * 2) * ry);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [index]);

        glEnd ();
        }

    glEnable (GL_CULL_FACE);
    glLightModeli (GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
    }

/*****************************************************************************/

void CRenderer::RenderSolidLoDGlobal (CHorizont &horizont, CPointMatrix &norm, int sizex, int sizey, float
sqrsize,
                                      float rx, float ry, int startx, int starty, int endx, int endy,
                                      int nx, int ny, int stripSize, int allSize)
    {
    for (int y = starty; y <= endy - ny; y += ny)
        {
        glBegin (GL_TRIANGLE_STRIP);
        for (int x = startx; x <= endx; x += nx)
            {
            ExtSetFogCoord (horizont.myMatrix [(y + ny) * (sizex + 1) + x]);
            glNormal3fv (&(norm.myMatrix [((y + ny) * (sizex + 1) + x) * FLOATSINPOINT]));
            ExtMultiTexCoords ((1.0F / sizex) * x * rx, (1.0F / sizey) * (y + ny) * ry);
            glVertex3f ((x - (sizex / 2)) * sqrsize,
                        (y + ny - (sizey / 2)) * sqrsize,
                        horizont.myMatrix [(y + ny) * (sizex + 1) + x]);

            ExtSetFogCoord (horizont.myMatrix [y * (sizex + 1) + x]);
            glNormal3fv (&(norm.myMatrix [(y * (sizex + 1) + x) * FLOATSINPOINT]));
            ExtMultiTexCoords ((1.0F / sizex) * x * rx, (1.0F / sizey) * y * ry);
            glVertex3f ((x - (sizex / 2)) * sqrsize,
                        (y - (sizey / 2)) * sqrsize,
                        horizont.myMatrix [y * (sizex + 1) + x]);
            }
```

```
        glEnd ();
        }
    }

/*****************************************************************************/

void CRenderer::RenderWireHor (CHorizont &horizont, CPointMatrix &norm)
    {
    float sqrsize = horizont.RetSquareSize ();
    int sizex = horizont.RetX () - 1, sizey = horizont.RetY () - 1,
        startx = 0, starty = 0, endx = 0, endy = 0, nx = 0, ny = 0;
    int stripSize = 0, allSize = 0;
    int lodAxis = SetLoDParams (horizont, sizex, sizey, sqrsize, startx, starty, endx, endy, nx, ny,
stripSize, allSize);
    SetColorCurrent (horizont.RetColor ());

    if (lodAxis == LOD_AXIS_Y) RenderWireLoDAlongY (horizont, sizex, sizey, sqrsize, startx, starty, endx,
endy, nx, ny, stripSize, allSize);
    else if (lodAxis == LOD_AXIS_X) RenderWireLoDAlongX (horizont, sizex, sizey, sqrsize, startx, starty,
endx, endy, nx, ny, stripSize, allSize);
    else if (lodAxis == LOD_GLOBAL) RenderWireLoDGlobal (horizont, sizex, sizey, sqrsize, startx, starty,
endx, endy, nx, ny, stripSize, allSize);

    glFlush ();
    }

/*****************************************************************************/

void CRenderer::RenderWireLoDAlongX (CHorizont &horizont, int sizex, int sizey, float sqrsize, int startx,
                                     int starty, int endx, int endy, int nx, int ny, int stripSize, int
allSize)
    {
    int inverseRendering = 0, div = 0, startLod = 0, endLod = 0;
    div = CountLoDDiv () / myLoDLevel;

    for (int n = 0; n <= div; n ++)
        {
        if (startx < endx)
            {
            if (n < div)
                {
                startLod = (startx + n * stripSize) / nx * nx;
                if (startLod > endx) continue;
                endLod = (startx + (n + 1) * stripSize) / (nx * 2) * (nx * 2);
                if (endLod > endx)
                    {
                    endLod = endx / nx * nx;
                    }
                }
            else
                {
                nx /= 2; ny /= 2;
                startLod = endx;
                endLod = startx + allSize;
                }
            }
        else
            {
            if (n < div)
                {
                startLod = (startx - n * stripSize) / nx * nx;
                if (startLod < endx) continue;
                endLod = (startx - (n + 1) * stripSize) / (nx * 2) * (nx * 2);
                if (endLod < endx)
                    {
                    endLod = endx / nx * nx;
                    }
                }
            else
                {
                nx /= 2; ny /= 2;
                startLod = endx;
                endLod = startx - allSize;
                }
            }
        if (startLod != startx && n != div && abs (startLod - endLod) >= abs (nx))
            {
            RenderWireStepStripX (horizont, sizex, sizey, sqrsize, startLod, starty, endy, nx / 2, ny /
2);
            startLod += nx;
            }
        for (int x = startLod; ; x += nx)
            {
            if (nx > 0 && x > endLod - nx) break;
            else if (nx < 0 && x < endLod - nx) break;
            if (nx < 0 || ny < 0)
                if (!(nx < 0 && ny < 0))
                    {
                    x += nx; nx = -nx;
                    inverseRendering = 1;
                    }
            for (int y = starty; ; y += ny)
                {
                if (ny > 0 && y > endy - ny) break;
                else if (ny < 0 && y < endy - ny) break;

                glBegin (GL_LINE_STRIP);
```

```
                glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                            (y - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [y * (sizex + 1) + x + nx]);
                glVertex3f ((x - (sizex / 2)) * sqrsize,
                            (y - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [y * (sizex + 1) + x]);
                glVertex3f ((x - (sizex / 2)) * sqrsize,
                            (y + ny - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [(y + ny) * (sizex + 1) + x]);
                glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                            (y - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [y * (sizex + 1) + x + nx]);
                glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                            (y + ny - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
                glVertex3f ((x - (sizex / 2)) * sqrsize,
                            (y + ny - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [(y + ny) * (sizex + 1) + x]);
                glEnd ();
                }
            if (inverseRendering)
                {
                x += nx; nx = -nx;
                inverseRendering = 0;
                }
            }
        nx *= 2; ny *= 2;
        }
    }

/*****************************************************************************/

void CRenderer::RenderWireStepStripX (CHorizont &horizont, int sizex, int sizey, float sqrsize,
                                      int startx, int starty, int endy, int nx, int ny)
    {
    int x = startx;
    for (int y = starty; y <= endy - ny * 2; y += ny * 2)
        {
        glBegin (GL_LINE_LOOP);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y) * (sizex + 1) + x]);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y) * (sizex + 1) + x + nx * 2]);
        glEnd ();

        glBegin (GL_LINE_LOOP);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y) * (sizex + 1) + x + nx * 2]);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny * 2) * (sizex + 1) + x + nx * 2]);
        glEnd ();

        glBegin (GL_LINE_LOOP);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny * 2) * (sizex + 1) + x + nx * 2]);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny * 2) * (sizex + 1) + x]);
        glEnd ();

        glBegin (GL_LINE_LOOP);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny * 2) * (sizex + 1) + x]);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x]);
        glEnd ();

        glBegin (GL_LINE_LOOP);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x]);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y) * (sizex + 1) + x]);
        glEnd ();
```

```
        }
    }
/*************************************************************************/
void CRenderer::RenderWireLoDAlongY (CHorizont &horizont, int sizex, int sizey, float sqrsize, int startx,
                                     int starty, int endx, int endy, int nx, int ny, int stripSize, int
allSize)
    {
    int inverseRendering = 0, div = 0, startLod = 0, endLod = 0;
    div = CountLoDDiv () / myLoDLevel;

    for (int n = 0; n <= div; n ++)
        {
        if (starty < endy)
            {
            if (n < div)
                {
                startLod = (starty + n * stripSize) / ny * ny;
                if (startLod > endy) continue;
                endLod = (starty + (n + 1) * stripSize) / (ny * 2) * (ny * 2);
                if (endLod > endy)
                    {
                    endLod = endy / ny * ny;
                    }
                }
            else
                {
                nx /= 2; ny /= 2;
                startLod = endy;
                endLod = starty + allSize;
                }
            }
        else
            {
            if (n < div)
                {
                startLod = (starty - n * stripSize) / ny * ny;
                if (startLod < endy) continue;
                endLod = (starty - (n + 1) * stripSize) / (ny * 2) * (ny * 2);
                if (endLod < endy)
                    {
                    endLod = endy / ny * ny;
                    }
                }
            else
                {
                nx /= 2; ny /= 2;
                startLod = endy;
                endLod = starty - allSize;
                }
            }
        if (startLod != starty && abs (startLod - endLod) >= abs (ny))
            {
            RenderWireStepStripY (horizont, sizex, sizey, sqrsize, startx, startLod, endx, nx / 2, ny /
2);
            startLod += ny;
            }
        for (int y = startLod; ; y += ny)
            {
            if (ny > 0 && y > endLod - ny) break;
            else if (ny < 0 && y < endLod - ny) break;
            if (nx < 0 || ny < 0)
                if (!(nx < 0 && ny < 0))
                    {
                    y += ny; ny = -ny;
                    inverseRendering = 1;
                    }
            for (int x = startx; ; x += nx)
                {
                if (nx > 0 && x > endx - nx) break;
                else if (nx < 0 && x < endx - nx) break;

                glBegin (GL_LINE_STRIP);
                glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                            (y - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [y * (sizex + 1) + x + nx]);
                glVertex3f ((x - (sizex / 2)) * sqrsize,
                            (y - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [y * (sizex + 1) + x]);
                glVertex3f ((x - (sizex / 2)) * sqrsize,
                            (y + ny - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [(y + ny) * (sizex + 1) + x]);
                glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                            (y - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [y * (sizex + 1) + x + nx]);
                glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                            (y + ny - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
                glVertex3f ((x - (sizex / 2)) * sqrsize,
                            (y + ny - (sizey / 2)) * sqrsize,
                            horizont.myMatrix [(y + ny) * (sizex + 1) + x]);
                glEnd ();
                }
            if (inverseRendering)
                {
                y += ny; ny = -ny;
```

```
                inverseRendering = 0;
                }
            }
        nx *= 2; ny *= 2;
        }
    }

/*****************************************************************************/

void CRenderer::RenderWireStepStripY (CHorizont &horizont, int sizex, int sizey, float sqrsize,
                                      int startx, int starty, int endx, int nx, int ny)
    {
    int y = starty;
    for (int x = startx; x <= endx - nx * 2; x += nx * 2)
        {
        glBegin (GL_LINE_LOOP);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y) * (sizex + 1) + x]);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y) * (sizex + 1) + x + nx]);
        glEnd ();

        glBegin (GL_LINE_LOOP);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y) * (sizex + 1) + x + nx]);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y) * (sizex + 1) + x + nx * 2]);
        glEnd ();

        glBegin (GL_LINE_LOOP);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y) * (sizex + 1) + x + nx * 2]);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny * 2) * (sizex + 1) + x + nx * 2]);
        glEnd ();

        glBegin (GL_LINE_LOOP);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
        glVertex3f ((x + nx * 2 - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny * 2) * (sizex + 1) + x + nx * 2]);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny * 2) * (sizex + 1) + x]);
        glEnd ();

        glBegin (GL_LINE_LOOP);
        glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                    (y + ny - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y + ny * 2 - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y + ny * 2) * (sizex + 1) + x]);
        glVertex3f ((x - (sizex / 2)) * sqrsize,
                    (y - (sizey / 2)) * sqrsize,
                    horizont.myMatrix [(y) * (sizex + 1) + x]);
        glEnd ();
        }
    }

/*****************************************************************************/

void CRenderer::RenderWireLoDGlobal (CHorizont &horizont, int sizex, int sizey, float sqrsize, int startx,
                                     int starty, int endx, int endy, int nx, int ny, int stripSize, int
allSize)
    {
    for (int y = starty; y <= endy - ny; y += ny)
        {
        for (int x = startx; x <= endx - nx; x += nx)
            {
            glBegin (GL_LINE_STRIP);
            glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                        (y - (sizey / 2)) * sqrsize,
                        horizont.myMatrix [y * (sizex + 1) + x + nx]);
            glVertex3f ((x - (sizex / 2)) * sqrsize,
                        (y - (sizey / 2)) * sqrsize,
                        horizont.myMatrix [y * (sizex + 1) + x]);
            glVertex3f ((x - (sizex / 2)) * sqrsize,
                        (y + ny - (sizey / 2)) * sqrsize,
                        horizont.myMatrix [(y + ny) * (sizex + 1) + x]);
```

```
            glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                        (y - (sizey / 2)) * sqrsize,
                        horizont.myMatrix [y * (sizex + 1) + x + nx]);
            glVertex3f ((x + nx - (sizex / 2)) * sqrsize,
                        (y + ny - (sizey / 2)) * sqrsize,
                        horizont.myMatrix [(y + ny) * (sizex + 1) + x + nx]);
            glVertex3f ((x - (sizex / 2)) * sqrsize,
                        (y + ny - (sizey / 2)) * sqrsize,
                        horizont.myMatrix [(y + ny) * (sizex + 1) + x]);
            glEnd ();
            }
        }
    }

/*****************************************************************************/

int CRenderer::SetLoDParams (CHorizont & hor, int sizex, int sizey, float sqrsize, int & startx, int &
starty,
                             int & endx, int & endy, int & nx, int & ny, int & stripSize, int & allSize)
    {
    float maxx = myCameras [myCurrentCamera - 1].RetMaxX (),
          maxy = myCameras [myCurrentCamera - 1].RetMaxY (),
          minx = myCameras [myCurrentCamera - 1].RetMinX (),
          miny = myCameras [myCurrentCamera - 1].RetMinY ();
    startx = ((int)(minx / sqrsize) + (sizex / 2)) - 1;
    starty = ((int)(miny / sqrsize) + (sizey / 2)) - 1;
    endx   = ((int)(maxx / sqrsize) + (sizex / 2)) + 1;
    endy   = ((int)(maxy / sqrsize) + (sizey / 2)) + 1;
    stripSize = ((int) (myCameras [myCurrentCamera - 1].myFarZ / sqrsize) / (CountLoDDiv () / myLoDLevel))
/ myLoDLevel * myLoDLevel;
    nx = 1; ny = 1;
    int lodAxis = LOD_AXIS_X, divx = 0, divy = 0;

    if (myCameras [myCurrentCamera - 1].myPosition (0) == maxx)
        {
        int buf = startx; startx = endx; endx = buf; nx = -1;
        lodAxis = LOD_AXIS_X;
        }
    else if (myCameras [myCurrentCamera - 1].myPosition (1) == maxy)
        {
        int buf = starty; starty = endy; endy = buf; ny = -1;
        lodAxis = LOD_AXIS_Y;
        }
    else if (myCameras [myCurrentCamera - 1].myPosition (0) == minx)
        {
        lodAxis = LOD_AXIS_X;
        }
    else if (myCameras [myCurrentCamera - 1].myPosition (1) == miny)
        {
        lodAxis = LOD_AXIS_Y;
        }
    else
        {
        float height = abs (RetHorizontHeight (hor, myCameras [myCurrentCamera - 1].myPosition (0),
                           myCameras [myCurrentCamera - 1].myPosition (1)) - myCameras [myCurrentCamera -
1].myPosition (2));
        int count = height / (myCameras [myCurrentCamera - 1].myFarZ) * (CountLoDDiv () / myLoDLevel);
        startx = startx / myLoDLevel * myLoDLevel - myLoDLevel;
        starty = starty / myLoDLevel * myLoDLevel - myLoDLevel;
        for (int n = 0; n < count; n ++)
            {
            nx *= 2; ny *= 2;
            }
        lodAxis = LOD_GLOBAL;
        }

    if (nx < 0)
        endx = endx / myLoDLevel * myLoDLevel - myLoDLevel;
    else
        endx = endx / myLoDLevel * myLoDLevel + myLoDLevel;

    if (ny < 0)
        endy = endy / myLoDLevel * myLoDLevel - myLoDLevel;
    else
        endy = endy / myLoDLevel * myLoDLevel + myLoDLevel;

    if (lodAxis == LOD_AXIS_X) starty = starty / myLoDLevel * myLoDLevel - myLoDLevel;
    else if (lodAxis == LOD_AXIS_Y) startx = startx / myLoDLevel * myLoDLevel - myLoDLevel;

    if (startx < 0) startx = 0; if (startx > sizex) startx = sizex;
    if (starty < 0) starty = 0; if (starty > sizey) starty = sizey;
    if (endx < 0) endx = 0;     if (endx > sizex) endx = sizex;
    if (endy < 0) endy = 0;     if (endy > sizey) endy = sizey;

    switch (lodAxis)
        {
        case LOD_AXIS_X:
            divy = myLoDLevel; divx = 1;
            allSize = abs (startx - endx);
            break;
        case LOD_AXIS_Y:
            divx = myLoDLevel; divy = 1;
            allSize = abs (starty - endy);
            break;
        case LOD_GLOBAL:
            divx = myLoDLevel; divy = myLoDLevel;
            break;
```

```
        default:
            lodAxis = LOD_AXIS_Y;
            divx = myLoDLevel; divy = 1;
            allSize = abs (starty - endy);
            break;
        }

    if (abs (endx - startx) > divx)
        {
        int deltax = abs (((endx - startx) / divx) * divx);
        if (endx >= startx) endx = startx + deltax;
        else endx = startx - deltax;
        }
    if (abs (endy - starty) > divy)
        {
        int deltay = abs (((endy - starty) / divy) * divy);
        if (endy >= starty) endy = starty + deltay;
        else endy = starty - deltay;
        }
    return lodAxis;
    }

/*************************************************************************/

int CRenderer::CountLoDDiv ()
    {
    int div = 0, x = 1;
    for ( ; div < 10; div ++)
        {
        x = 1;
        for (int n = 0; n < div; n ++) x *= 2;
        if (x == myLoDLevel)
            {
            return (div + 1) * myLoDLevel;
            }
        }
    return 1;
    }

/*************************************************************************/

void CRenderer::ComputeNormals (CHorizont &hor, CPointMatrix & norms)
    {
    for (int y = 0; y < hor.RetY (); y ++)
        for (int x = 0; x < hor.RetX (); x ++)
            ComputeNormal (hor, norms, x, y);
    }

/*************************************************************************/

void CRenderer::ComputeNormal (CHorizont &hor, CPointMatrix &norms, int x, int y)
    {
    CPoint3D normals [6] = {CPoint3D (0.0F, 0.0F, 1.0F), CPoint3D (0.0F, 0.0F, 1.0F), CPoint3D (0.0F,
0.0F, 1.0F),
                            CPoint3D (0.0F, 0.0F, 1.0F), CPoint3D (0.0F, 0.0F, 1.0F), CPoint3D (0.0F, 0.0F,
1.0F)},
            result (0.0F, 0.0F, 0.0F);
    CPoint3D vectors [2];
    if (y > 0 && x < hor.RetX () - 1)
        {
        vectors [0] (0) = hor.RetSquareSize (); vectors [0] (1) = 0; vectors [0] (2) = hor (x + 1, y - 1)
- hor (x, y - 1);
        vectors [1] (0) = 0; vectors [1] (1) = hor.RetSquareSize (); vectors [1] (2) = hor (x, y) - hor
(x, y - 1);
        normals [0] = vectors [0].CrossProduct (vectors [1]);
        vectors [0] (0) = - hor.RetSquareSize (); vectors [0] (1) = 0; vectors [0] (2) = hor (x, y) - hor
(x + 1, y);
        vectors [1] (0) = 0; vectors [1] (1) = - hor.RetSquareSize (); vectors [1] (2) = hor (x + 1, y -
1) - hor (x + 1, y);
        normals [1] = vectors [0].CrossProduct (vectors [1]);
        }
    if (x > 0 && y > 0)
        {
        vectors [0] (0) = - hor.RetSquareSize (); vectors [0] (1) = 0; vectors [0] (2) = hor (x - 1, y) -
hor (x, y);
        vectors [1] (0) = 0; vectors [1] (1) = - hor.RetSquareSize (); vectors [1] (2) = hor (x, y - 1) -
hor (x, y);
        normals [2] = vectors [0].CrossProduct (vectors [1]);
        }
    if (x > 0 && y < hor.RetY () - 1)
        {
        vectors [0] (0) = hor.RetSquareSize (); vectors [0] (1) = 0; vectors [0] (2) = hor (x, y) - hor (x
- 1, y);
        vectors [1] (0) = 0; vectors [1] (1) = hor.RetSquareSize (); vectors [1] (2) = hor (x - 1, y + 1)
- hor (x - 1, y);
        normals [3] = vectors [0].CrossProduct (vectors [1]);
        vectors [0] (0) = - hor.RetSquareSize (); vectors [0] (1) = 0; vectors [0] (2) = hor (x - 1, y +
1) - hor (x, y + 1);
        vectors [1] (0) = 0; vectors [1] (1) = - hor.RetSquareSize (); vectors [1] (2) = hor (x, y) - hor
(x, y + 1);
        normals [4] = vectors [0].CrossProduct (vectors [1]);
        }
    if (x < hor.RetX () - 1 && y < hor.RetY () - 1)
        {
        vectors [0] (0) = hor.RetSquareSize (); vectors [0] (1) = 0; vectors [0] (2) = hor (x + 1, y) -
hor (x, y);
```

```
        vectors [1] (0) = 0; vectors [1] (1) = hor.RetSquareSize (); vectors [1] (2) = hor (x, y + 1) -
hor (x, y);
        normals [5] = vectors [0].CrossProduct (vectors [1]);
        }

    for (int n = 0; n < 6; n ++)
        {
        normals [n].Normalize ();
        result += normals [n];
        }
    result /= 6.0F;
    result.Normalize ();
    norms (x, y, 0) = result (0);
    norms (x, y, 1) = result (1);
    norms (x, y, 2) = result (2);
    }

/*************************************************************************/

float CRenderer::RetHorizontHeight (CHorizont & hor, float x, float y)
    {
    int x1 = (int) ((x + (hor.RetX () / 2) * hor.RetSquareSize ()) / hor.RetSquareSize ()),
        y1 = (int) ((y + (hor.RetY () / 2) * hor.RetSquareSize ()) / hor.RetSquareSize ()),
        x2, y2;
    x2 = x1 + 1; y2 = y1 + 1;
    if (x1 < 0 || y1 < 0 || x1 >= hor.RetX () || y1 >= hor.RetY () ||
        x2 < 0 || y2 < 0 || x2 >= hor.RetX () || y2 >= hor.RetY ()) return -1000.0F;

    CPoint3D p1 (0.0F, 0.0f, 0.0F), p2 (0.0F, 0.0F, 0.0F), p3 (0.0F, 0.0F, 0.0F),
             norm (0.0F, 0.0F, 0.0F), start (0.0F, 0.0F, 0.0F), dest (0.0F, 0.0F, 1.0F),
             result (0.0F, 0.0F, 0.0F);
    float T = 0.0F, D = 0.0F;
    x -= (float) x1 * hor.RetSquareSize () - (hor.RetX () / 2) * hor.RetSquareSize ();
    y -= (float) y1 * hor.RetSquareSize () - (hor.RetY () / 2) * hor.RetSquareSize ();
    start.Init (x, y, 0.0F);
    if (y >= x)
        {
        p1.Init (0.0F,                 0.0F,                 hor (x1, y1));
        p2.Init (0.0F,                 hor.RetSquareSize (), hor (x1, y2));
        p3.Init (hor.RetSquareSize (), hor.RetSquareSize (), hor (x2, y2));
        }
    if (y < x)
        {
        p1.Init (0.0F,                 0.0F,                 hor (x1, y1));
        p2.Init (hor.RetSquareSize (), 0.0F,                 hor (x2, y1));
        p3.Init (hor.RetSquareSize (), hor.RetSquareSize (), hor (x2, y2));
        }
    norm = (p1 - p2).CrossProduct (p3 - p2);
    norm.Normalize ();
    D = - (p1.DotProduct (norm));
    T = - ((start.DotProduct (norm) + D) / norm.DotProduct (dest));
    result = start + dest * T;
    return result (2);
    }

/*************************************************************************/

void CRenderer::RenderMeshes (CSmartPointer &meshes, CSmartPointer &meshes2)
    {
    wglMakeCurrent (myDC, myGLRC);
    if (myRotationWay == RW_MATRIX_ROTATING) TransformMeshes (meshes, meshes2);
    for (int n = 0; n < meshes.RetCurObjNum (); n ++)
        if (((CMesh *) meshes2.RetObject (n)) -> RetParent () == NULL) SetMeshLights ((CMesh *)
meshes2.RetObject (n));
    for (n = 0; n < meshes.RetCurObjNum (); n ++)
        if (((CMesh *) meshes2.RetObject (n)) -> RetParent () == NULL) RenderMesh ((CMesh *)
meshes2.RetObject (n));
    }

/*************************************************************************/

void CRenderer::RenderMeshes (void * meshes, void * meshes2, int nmeshes)
    {
    RenderMeshes (reinterpret_cast <CMesh *> (meshes), reinterpret_cast <CMesh *> (meshes2), nmeshes);
    }

/*************************************************************************/

void CRenderer::RenderMeshes (CMesh * meshes, CMesh * meshes2, int nmeshes)
    {
    wglMakeCurrent (myDC, myGLRC);
    if (myRotationWay == RW_MATRIX_ROTATING) TransformMeshes (meshes, meshes2, nmeshes);
    for (int n = 0; n < nmeshes; n ++)
        if (meshes2 [n].RetParent () == NULL) SetMeshLights (&meshes2 [n]);
    for (n = 0; n < nmeshes; n ++)
        if (meshes2 [n].RetParent () == NULL) RenderMesh (&meshes2 [n]);
    }

/*************************************************************************/

void CRenderer::RenderMesh (CMesh * mesh)
    {
    glPushMatrix ();
    SetMaterial (mesh -> RetMaterial ());
    if (myRotationWay == RW_OGL_ROTATING) glMultMatrixf (mesh -> RetTransformMatrix ().RetPointer ());
    glTranslatef (mesh -> RetCenter () (0), mesh -> RetCenter () (1), mesh -> RetCenter () (2));
    myBufPoint = mesh -> RetCenter ();
```

```
    glBindTexture (GL_TEXTURE_2D, 0);
    for (int n = 0; n < mesh->RetCurPolys (); n ++)
        RenderPoly (mesh->RetPoly (n));

    for (int i = 0; i < mesh -> RetCurChildren (); i ++)
        RenderMesh (mesh -> RetChild (i));

    glPopMatrix ();
    myBufPoint.Init (0.0F, 0.0F, 0.0F);
    myBufMaterial.SetColorSource (COLOR_PER_VERTEX);
    myBufMaterial.Init (1, 0, 0);
    }

/*************************************************************************/

void CRenderer::SetMeshLights (CMesh * mesh)
    {
    glPushMatrix ();
    glTranslatef (mesh -> RetCenter()(0), mesh -> RetCenter()(1), mesh -> RetCenter()(2));
    for (int n = 0; n < mesh -> RetCurLights (); n ++)
        ResetLight (mesh -> RetLight (n));

    for (int i = 0; i < mesh -> RetCurChildren (); i ++)
        SetMeshLights (mesh -> RetChild (i));

    glPopMatrix ();
    }

/*************************************************************************/

void CRenderer::RenderPrimitives (CSmartPointer &prims)
    {
    wglMakeCurrent (myDC, myGLRC);
    for (int n = 0; n < prims.RetObjNum (); n ++)
        {
        if (prims.RetObject (n) != NULL)
            RenderPrimitive ((CStandardPrimitive *) prims.RetObject (n));
        }
    }

/*************************************************************************/

void CRenderer::RenderPrimitives (void * prims, int nprims)
    {
    RenderPrimitives (reinterpret_cast <CStandardPrimitive *> (prims), nprims);
    }

/*************************************************************************/

void CRenderer::RenderPrimitives (CStandardPrimitive * prims, int nprims)
    {
    wglMakeCurrent (myDC, myGLRC);
    for (int n = 0; n < nprims; n ++)
        {
        RenderPrimitive (&prims [n]);
        }
    }

/*************************************************************************/

void CRenderer::RenderPrimitive (CStandardPrimitive * prim)
    {
    glPushMatrix ();
    glTranslatef (prim -> RetCenter()(0), prim -> RetCenter()(1), prim -> RetCenter()(2));
    glBindTexture (GL_TEXTURE_2D, 0);
    SetColorCurrent (prim -> RetColor ());
    switch (prim -> RetType ())
        {
        case STDPR_TEAPOT:
            //glutSolidTeapot (prim -> RetParam (0));
            break;
        case STDPR_CUBE:
            //glutSolidCube (prim -> RetParam (0));
            break;
        case STDPR_SPHERE:
            //glutSolidSphere (prim -> RetParam (0), prim -> RetParam (1), prim -> RetParam (2));
            break;
        case STDPR_CONE:
            //glutSolidCone (prim -> RetParam (0), prim -> RetParam (1), prim -> RetParam (2), prim ->
RetParam (3));
            break;
        case STDPR_TORUS:
            //glutSolidTorus (prim -> RetParam (0), prim -> RetParam (1), prim -> RetParam (2), prim ->
RetParam (3));
            break;
        default:
            break;
        }
    glPopMatrix ();
    }

/*************************************************************************/

void CRenderer::RenderPolys (CSmartPointer &polys)
    {
    wglMakeCurrent (myDC, myGLRC);
    for (int n = 0; n < polys.RetObjNum (); n ++)
```

```
        if (polys.RetObject (n) != NULL)
            RenderPoly ((CPolygon *) polys.RetObject (n));
    }

/*************************************************************************/

void CRenderer::RenderPolys (void * polys, int npolys)
    {
    RenderPolys (reinterpret_cast <CPolygon *> (polys), npolys);
    }

/*************************************************************************/

void CRenderer::RenderPolys (CPolygon * polys, int npolys)
    {
    wglMakeCurrent (myDC, myGLRC);
    for (int n = 0; n < npolys; n ++)
        RenderPoly (&polys [n]);
    }

/*************************************************************************/

void CRenderer::RenderPoly (CPolygon * poly)
    {
    if (myBuffer.IfTransparent (poly -> myTexPointer) ||
        poly -> myVertexArray [0].myColor.myRGBA [3] != 1.0F ||
        poly -> myVertexArray [1].myColor.myRGBA [3] != 1.0F ||
        poly -> myVertexArray [2].myColor.myRGBA [3] != 1.0F)
        {
        if (myBufferWriteMode)
            if (myBuffer.AddPolygon (myBufPoint, poly)) return;
        }

    if (myRenderLevel >= RL_LOW_TEXTURED)
        {
        glBindTexture (GL_TEXTURE_2D, 0);
        if (poly->RetTexPointer () == 0)
            RenderSolidPoly (poly);
        else
            RenderTexPoly (poly);
        }
    else
        {
        if (myRenderLevel >= RL_SOLID)
            RenderSolidPoly (poly);
        }
    if (myRenderLevel == RL_WIREFRAME)
        {
        RenderWirePoly (poly);
        }
    }

/*************************************************************************/

void CRenderer::RenderTransparent ()
    {
    myBufferWriteMode = 0;
    glEnable (GL_ALPHA_TEST);
    glAlphaFunc (GL_GEQUAL, 0.1F);

    CBillBoard * board = NULL;
    for (int i = 0; i < myBuffer.myCurBoardNum; i ++)
        {
        board = myBuffer.myBoards [i];
        if (board -> myColor.myRGBA [3] != 1.0F)
            glDepthMask (GL_FALSE);
        RenderBillBoard (myBuffer.myBoards [i]);
        if (board -> myColor.myRGBA [3] != 1.0F)
            glDepthMask (GL_TRUE);
        }

    CPolygon * poly = NULL;
    for (int n = 0; n < myBuffer.myCurPolyNum; n ++)
        {
        poly = myBuffer.myPolys [n];
        myBufPoint = myBuffer.myCenters [n];
        if (poly -> myVertexArray [0].myColor.myRGBA [3] != 1.0F ||
            poly -> myVertexArray [1].myColor.myRGBA [3] != 1.0F ||
            poly -> myVertexArray [2].myColor.myRGBA [3] != 1.0F)
            glDepthMask (GL_FALSE);
        glTranslatef (myBuffer.myCenters [n].myXYZ [0], myBuffer.myCenters [n].myXYZ [1],
myBuffer.myCenters [n].myXYZ [2]);
        RenderPoly (poly);
        glTranslatef (- myBuffer.myCenters [n].myXYZ [0], - myBuffer.myCenters [n].myXYZ [1], -
myBuffer.myCenters [n].myXYZ [2]);
        if (poly -> myVertexArray [0].myColor.myRGBA [3] != 1.0F ||
            poly -> myVertexArray [1].myColor.myRGBA [3] != 1.0F ||
            poly -> myVertexArray [2].myColor.myRGBA [3] != 1.0F)
            glDepthMask (GL_TRUE);
        }

    glDisable (GL_ALPHA_TEST);
    myBufferWriteMode = 1;
    }

/*************************************************************************/
```

```
void CRenderer::RenderAtmosphere ()
    {
    if (myRenderLevel >= RL_LIGHTENED) DisableLighting ();
    glDepthMask (GL_FALSE);
    myBufferWriteMode = 0;
    ExtSetMultiTex (myAtmosphere.myTexID, 0, 0, 0);

    CPoint3D dir = (myCameras [myCurrentCamera - 1].myPosition - myPreviousCameraPosition);
    myAtmosphere.myStartTexX += (dir (0) / ((myAtmosphere.myX2 - myAtmosphere.myX1) / myAtmosphere.myTexX)
+ myAtmosphere.mySkySpeed (0));
    myAtmosphere.myStartTexY += (dir (1) / ((myAtmosphere.myY2 - myAtmosphere.myY1) / myAtmosphere.myTexY)
+ myAtmosphere.mySkySpeed (1));

    glPushMatrix ();
    glTranslatef (myCameras [myCurrentCamera - 1].myPosition (0),
                  myCameras [myCurrentCamera - 1].myPosition (1),
                  myCameras [myCurrentCamera - 1].myPosition (2));

    glBegin (GL_QUADS);
    glColor4fv (myAtmosphere.mySkyColor.myRGBA);

    glTexCoord2f (myAtmosphere.myStartTexX,
                  myAtmosphere.myStartTexY);
    glVertex3f (myAtmosphere.myX1, myAtmosphere.myY1, myAtmosphere.myHeight);

    glTexCoord2f (myAtmosphere.myStartTexX + (float) myAtmosphere.myTexX,
                  myAtmosphere.myStartTexY);
    glVertex3f (myAtmosphere.myX2, myAtmosphere.myY1, myAtmosphere.myHeight);

    glTexCoord2f (myAtmosphere.myStartTexX + (float) myAtmosphere.myTexX,
                  myAtmosphere.myStartTexY + (float) myAtmosphere.myTexY);
    glVertex3f (myAtmosphere.myX2, myAtmosphere.myY2, myAtmosphere.myHeight);

    glTexCoord2f (myAtmosphere.myStartTexX,
                  myAtmosphere.myStartTexY + (float) myAtmosphere.myTexY);
    glVertex3f (myAtmosphere.myX1, myAtmosphere.myY2, myAtmosphere.myHeight);

    glEnd ();

    glPopMatrix ();
    glBindTexture (GL_TEXTURE_2D, 0);
    ExtDisableMultiTex ();

    myBufferWriteMode = 1;
    glDepthMask (GL_TRUE);
    if (myRenderLevel >= RL_LIGHTENED) EnableLighting ();
    glFlush ();
    }

/*************************************************************************/

void CRenderer::RenderWater ()
    {
    if (myRenderLevel >= RL_SOLID) RenderSolidWater ();
    else RenderWireWater ();
    }

/*************************************************************************/

void CRenderer::RenderWireWater ()
    {
    float sqrsize = myWater.myTileSize;
    int sizex = myWater.mySizeX, sizey = myWater.mySizeY;
    SetColorCurrent (myWater.myColor.myRGBA);
    myWater.Move ();

    for (int y = 0; y < sizey - 1; y += 1)
        for (int x = 0; x < sizex - 1; x += 1)
            {
            glBegin (GL_LINE_STRIP);
            glVertex3f ((x + 1 - (sizex / 2)) * sqrsize,
                        (y - (sizey / 2)) * sqrsize,
                        myWater.myHeight);
            glVertex3f ((x - (sizex / 2)) * sqrsize,
                        (y - (sizey / 2)) * sqrsize,
                        myWater.myHeight);
            glVertex3f ((x - (sizex / 2)) * sqrsize,
                        (y + 1 - (sizey / 2)) * sqrsize,
                        myWater.myHeight);
            glVertex3f ((x + 1 - (sizex / 2)) * sqrsize,
                        (y - (sizey / 2)) * sqrsize,
                        myWater.myHeight);
            glVertex3f ((x + 1 - (sizex / 2)) * sqrsize,
                        (y + 1 - (sizey / 2)) * sqrsize,
                        myWater.myHeight);
            glVertex3f ((x - (sizex / 2)) * sqrsize,
                        (y + 1 - (sizey / 2)) * sqrsize,
                        myWater.myHeight);
            glEnd ();
            }
    glFlush ();
    }

/*************************************************************************/

void CRenderer::RenderSolidWater ()
    {
```

```
    float sqrsize = myWater.myTileSize,
          rx = myWater.myTexCoordX,
          ry = myWater.myTexCoordY;
    if (myWater.myDetailTextureID != 0)
        {
        rx = 1.0F; ry = 1.0F;
        }
    int sizex = myWater.mySizeX, sizey = myWater.mySizeY;
    glMaterialf (GL_FRONT_AND_BACK, GL_SHININESS, 0.0F);
    SetColorCurrent (myWater.myColor.myRGBA);
    if (myWater.myColor.myRGBA [3] != 1.0F) glDepthMask (GL_FALSE);
    myWater.Move ();
    ExtSetMultiTex (myWater.myGlobalTextureID, myWater.myDetailTextureID,
                    myWater.myTexCoordX, myWater.myTexCoordY);

    for (int y = 0; y < sizey - 1; y += 1)
        {
        glBegin (GL_TRIANGLE_STRIP);
        for (int x = 0; x < sizex; x += 1)
            {
            ExtSetFogCoord (myWater.myHeight);
            glNormal3f (1.0F, 1.0F, 1.0F);
            ExtMultiTexCoords ((1.0F / sizex) * x * rx + myWater.myTexPosX, (1.0F / sizey) * (y + 1) * ry
+ myWater.myTexPosY);
            glVertex3f ((x - (sizex / 2)) * sqrsize,
                        (y + 1 - (sizey / 2)) * sqrsize,
                        myWater.myHeight);
            ExtSetFogCoord (myWater.myHeight);
            glNormal3f (1.0F, 1.0F, 1.0F);
            ExtMultiTexCoords ((1.0F / sizex) * x * rx + myWater.myTexPosX, (1.0F / sizey) * y * ry +
myWater.myTexPosY);
            glVertex3f ((x - (sizex / 2)) * sqrsize,
                        (y - (sizey / 2)) * sqrsize,
                        myWater.myHeight);
            }
        glEnd ();
        }

    glFlush ();
    glBindTexture (GL_TEXTURE_2D, 0);
    ExtDisableMultiTex ();
    if (myWater.myColor.myRGBA [3] != 1.0F) glDepthMask (GL_TRUE);
    }

/*****************************************************************************/

void CRenderer::MakeMipmaps (CColorMatrix& tex)
    {
    glBindTexture (GL_TEXTURE_2D, tex.RetID ());
    gluBuild2DMipmaps (GL_TEXTURE_2D, 4, tex.RetX (), tex.RetY (),
                  GL_RGBA, GL_FLOAT, tex.RetPointer (0));
    }

/*****************************************************************************/

void CRenderer::GetTexID (CColorMatrix& tex)
    {
    unsigned int id = 0;
    glGenTextures (1, &id);
    tex.SetNewID (id);
    }

/*****************************************************************************/

void CRenderer::BindTexture (int id)
    {
    glBindTexture (GL_TEXTURE_2D, id);
    if (myRenderLevel == RL_LOW_TEXTURED)
        {
        glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
        glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
        }
    if (myRenderLevel >= RL_HIGH_TEXTURED)
        {
        glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
        glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
        }
    }

/*****************************************************************************/

void CRenderer::InitTex (CColorMatrix& tex)
    {
    glPixelStorei (GL_UNPACK_ALIGNMENT, 1);

    glBindTexture (GL_TEXTURE_2D, tex.RetID ());

    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);

    glTexEnvf (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

    glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA, tex.RetX (), tex.RetY (), 0,
                  GL_RGBA, GL_FLOAT, tex.RetPointer (0));
    MakeMipmaps (tex);
```

```
    }
/*************************************************************************/

void CRenderer::ExtSetMultiTex (int tex1, int tex2, int scalex, int scaley)
    {
    if (myRenderLevel >= RL_EXTENSIONS_USED)
        {
        glActiveTextureARB (GL_TEXTURE0_ARB);
        glEnable (GL_TEXTURE_2D);
        BindTexture (tex1);

        if (tex2 != 0)
            {
            glActiveTextureARB (GL_TEXTURE1_ARB);
            glEnable (GL_TEXTURE_2D);

            glTexEnvi (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_COMBINE_ARB);
            glTexEnvi (GL_TEXTURE_ENV, GL_RGB_SCALE_ARB, 2);

            BindTexture (tex2);

            glMatrixMode (GL_TEXTURE);
            glLoadIdentity ();
            glScalef ((float) scalex, (float) scaley, 1);
            glMatrixMode (GL_MODELVIEW);
            }
        }
    else
        {
        if (myRenderLevel >= RL_LOW_TEXTURED) BindTexture (tex1);
        }
    }
/*************************************************************************/

void CRenderer::ExtDisableMultiTex ()
    {
    if (myRenderLevel >= RL_EXTENSIONS_USED)
        {
        glActiveTextureARB(GL_TEXTURE1_ARB);
        glDisable(GL_TEXTURE_2D);

        glActiveTextureARB(GL_TEXTURE0_ARB);
        glDisable(GL_TEXTURE_2D);
        }
    }
/*************************************************************************/

void CRenderer::ExtMultiTexCoords (float x, float y)
    {
    if (myRenderLevel >= RL_EXTENSIONS_USED)
        {
        glMultiTexCoord2fARB(GL_TEXTURE0_ARB, x, y);
        glMultiTexCoord2fARB(GL_TEXTURE1_ARB, x, y);
        }
    else
        {
        glTexCoord2f (x, y);
        }
    }
/*************************************************************************/

void CRenderer::ExtSetFogCoord (float height)
    {
    if (myRenderLevel >= RL_EXTENSIONS_USED && myAtmosphere.myIsVolumeFogEnable)
        {
        float fogH = 0;
        if (height > myAtmosphere.myVolumeFogDepth) fogH = 0;
        else fogH = myAtmosphere.myVolumeFogDepth - height;
        glFogCoordfEXT (fogH);
        }
    }
/*************************************************************************/

void CRenderer::RenderTexPoly (CPolygon * poly)
    {
    ExtSetMultiTex (poly->RetTexPointer (), 0, 0, 0);

    glBegin (GL_TRIANGLES);

    glTexCoord2f (poly -> myVertexArray [0].myCoord.myX, poly -> myVertexArray [0].myCoord.myY);
    PutVertex (poly, 0);
    glTexCoord2f (poly -> myVertexArray [1].myCoord.myX, poly -> myVertexArray [1].myCoord.myY);
    PutVertex (poly, 1);
    glTexCoord2f (poly -> myVertexArray [2].myCoord.myX, poly -> myVertexArray [2].myCoord.myY);
    PutVertex (poly, 2);

    glEnd ();
    glBindTexture (GL_TEXTURE_2D, 0);
    }
/*************************************************************************/
```

```
void CRenderer::RenderSolidPoly (CPolygon * poly)
    {
    switch (poly -> myCurrVert)
        {
        case 2: glBegin (GL_LINES);
                break;
        case 3: glBegin (GL_TRIANGLES);
                break;
        }

    for (int n = 0; n < poly -> myCurrVert; n ++)
        PutVertex (poly, n);

    glEnd ();
    }

/*************************************************************************/

void CRenderer::RenderWirePoly (CPolygon * poly)
    {
    glBegin (GL_LINES);

    PutVertex (poly, 0); PutVertex (poly, 1);
    PutVertex (poly, 0); PutVertex (poly, 2);
    PutVertex (poly, 1); PutVertex (poly, 2);

    glEnd ();
    }

/*************************************************************************/

void CRenderer::PutVertex (CPolygon * poly, int n)
    {
    ExtSetFogCoord (poly -> myVertexArray [n].myPoint.myXYZ [2] + myBufPoint.myXYZ [2]);
    if (myBufMaterial.myColorSource == COLOR_PER_VERTEX) SetColorCurrent (poly -> myVertexArray
[n].myColor.myRGBA);
    glNormal3fv (poly -> myVertexArray [n].myNormal.myXYZ);
    glVertex3fv (poly -> myVertexArray [n].myPoint.myXYZ);
    glFlush ();
    }

/*************************************************************************/

/*************************************************************************/
/*≟ююыєўш€№ ¤ъчхьяы Ё юс·хъЄр CRenderer*/

void GetInstance (void ** ptr)
    {
    IUnknown * pUnknown = static_cast <IRenderer *> (new CRenderer);
    pUnknown -> AddRef ();
    *ptr = pUnknown;
    }

/*************************************************************************/
```

Файл SmallCompiler\Console.cpp

```
// Console.cpp - implementation of Console.h, of module SmallCompiler
// created 14.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Se"
#define __MODULE_NAME  "SmallCompiler"

#include "System.h"
#include "Console.h"
#include "..\Console\Console.h"
#include "..\ExeptionEngine\Exeption.h"

void CommandCompileFromFile (IConsole* c, int, char* data)
    {
    if (! ( data && strlen ( data) && "SmallCompiler : CommandCompileFromFile bad params. file name
expected.\n"))
        {
        c->printf ("Filename expected.\n");
        return;
        }
    c->printf ( "Compiling smalltalk file \"%s\"...", data);
    if ( !SmCompiler::Compiler->compileFromFile ( data))
        { c->printf ("Failed!\n"); }
    else c->printf ("Success.\n");
    }
/*
void CommandInterpreter (IConsole* c, int ln, Lexem1* data)
{
c->printf ( "Smt interpreting started...\n");
CScope& gs = SmCompiler::Linker->getGlobalScope();
int cur = 0;
bool in = 0;
CVar* inVar;
if ( CString(":=") == CString ( data[1].data))
```

```
    {
    c->printf ( "input var found.\n");
    in = 1;
    if ( !gs.check ( CString ( data[0].data)))
        {
        c->printf ( "no input var. creating...\n");
        gs.addVar ( CVar ( CString ( data[0].data)));
        }
    inVar = & gs ( CString ( data[0].data));
    cur = 2;
    }

c->printf ( "trying to find receipient\n");
CObjRef receipient = gs ( CString ( data[cur++].data));

c->printf ( "trying to call func\n");
SParam res = receipient ( CString ( data[cur].data))();

if ( res.type == SParam::ePObj)
    c->printf ( "object returned\n");
else if ( res.type == SParam::eInt)
    c->printf ( "integer %d returned\n", res.rInt());
else if ( res.type == SParam::eLong)
    c->printf ( "long %ld returned\n", res.rLong());
else
    c->printf ( "somthing is returned\n", res.rInt());


if ( in && res.type == SParam::ePObj) *inVar = CVar ( res.rObj(), inVar->rName());

}
*/
//void CommandInterpreter (IConsole* c, int ln, Lexem1* data)
void CommandInterpreter22 (IConsole* c, int, char* data)
{
c->printf ( "Smt interpreting started...\n");

bool r = SmCompiler::Compiler->executeString ( data);

c->printf ( "Executed %s", r? "successfully" : "failure");
}

void AddCommands ()
{
    //Assert (Console);
    SmCompiler::Console->AddCommand ( "smt", (CommandProcP)CommandInterpreter22, false);
    SmCompiler::Console->AddCommand ("com", (CommandProcP) CommandCompileFromFile, false);
}
```

Файл SmallCompiler\Module.cpp

```
#define __PROGRAMMER    "Se"
#define __MODULE_NAME   "SmallSyntax"

#include "System.h"
#include "SmallCompilerCore.h"
#include "..\ExeptionEngine\exeption.h"
#include "console.h"

    #define DLL_NAME "SmallSyntax"
    #define DLL_CLASS CSmallCompiler
    #include "..\Common\Utilities\Module.c"

#pragma warning (disable: 4275) // non dll-interface class used as base for dll-interface class

DLL_INIT(loader)
{
    Assert (loader);
    SmCompiler::Loader = loader;

    LinkerDataTypesLibInit ( loader);

    SmCompiler::Loader->GetInterface (IID_ISmallCompiler, (void**)&SmCompiler::Compiler);
    Assert (SmCompiler::Compiler);

    SmCompiler::Loader->GetInterface (IID_ILinker, (void**)&SmCompiler::Linker);
    Assert (SmCompiler::Linker);
    SmCompiler::GS = & SmCompiler::Linker->getGlobalScope();

    SmCompiler::Loader->GetInterface (IID_ISmallVM, (void**)&SmCompiler::SmVM);
    Assert (SmCompiler::SmVM);

    SmCompiler::Loader->GetInterface (IID_IConsole, (void**)&SmCompiler::Console);
    Assert (SmCompiler::Console);

    SmCompiler::Console->printf ("SmallCompiler initialized.\n");
    AddCommands ();
}

DLL_ONLOAD (loader)
{
    IConsole* con = NULL;
```

```
    loader->GetInterface (IID_IConsole, (void**)&con);
    Assert (con);
    con->printf ("COMPILER: Smalltalk Compiler by Se version 1.0\n");
}

#pragma warning (default: 4275) // non dll-interface class used as base for dll-interface class
```

Файл SmallCompiler\SmallCompilerCore.cpp

```
// SmallCompilerCore.cpp - implementation of SmallCompilerCore.h, of module SmallCompiler
// created 15.10.2002 by Se in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Se"
#define __MODULE_NAME  "SmallCompiler"

#include "System.h"
#include "SmallCompilerCore.h"
#include "TokenCore.h"
#include "SyntaxCore.h"

#include "..\ExeptionEngine\exeption.h"

IMPLEMENT_IUNKNOWN ( CSmallCompiler, ISmallCompiler, IID_ISmallCompiler);

CSmallCompiler::CSmallCompiler()
    {
    compileAsDebug = true;
    }

CSmallCompiler::~CSmallCompiler()
    { }

BlockPtr CSmallCompiler::compileString ( const char* str)
{
    std::string s ( str);
    int d = s.find_first_not_of ( std::string ( " \t\n\r"));
    if ( s[d] != '^')
        s.insert ( 0, '^');
    s += " ";
    //!!! bad bad bad
    FILE* temp = fopen ("smt.tmp", "wt");
    fputs ( s.c_str(), temp);
    fclose (temp);

    temp = fopen ("smt.tmp", "rt");
    Assert (temp && "temp file in smt string execution\n");
    myTokens = CTokenContainer ( 10);
    myMessages = CMessageContainer ( 10);

    CTokenAnalyzer tanalyzer( &myTokens, temp, "smt.tmp", &myMessages);
    bool res = tanalyzer.analyze();

    //myTokens.print();

    CSyntaxAnalyzer sanalyzer ( &myTokens, SmCompiler::SmVM, &myMessages);

    BlockPtr pBl = SmCompiler::SmVM->regBlock ( std::auto_ptr<Block> ( new Block()));

    sanalyzer.eatStatementTo ( pBl);

    myMessages.print();
    return pBl;
}

bool CSmallCompiler::executeString ( const char *str)
    {
//  SmCompiler::Console->printf ("smt executing...\n");

    BlockPtr pBl = compileString ( str);

    SParam r = pBl->call ( &SmCompiler::Linker->getGlobalScope(), CParams());

    if ( r.type == SParam::ePObj)
        SmCompiler::Console->printf ( "object returned\n");
    else if ( r.type == SParam::eInt)
        SmCompiler::Console->printf ( "integer %d returned\n", r.rInt());
    else if ( r.type == SParam::eFloat)
        SmCompiler::Console->printf ( "float %g returned\n", r.rDouble());
    else if ( r.type == SParam::eDouble)
        SmCompiler::Console->printf ( "double %lg returned\n", r.rDouble());
    else if ( r.type == SParam::eLong)
        SmCompiler::Console->printf ( "long %ld returned\n", r.rLong());
    else
        SmCompiler::Console->printf ( "something maybe returned\n");

    return 1;
    };

bool CSmallCompiler::compileFromFile ( const char* fileName, void* smVM)
    {
    if (!smVM)
```

```
        smVM = SmCompiler::SmVM;
    Assert (smVM);

    FILE* fileIn = fopen ( fileName, "rt");
    if (!fileIn)
        {
        SmCompiler::Console->printf ( "SmallCompiler::compileFromFile can't open input file \"%s\".\n",
fileName);
        Throw ( general, "SmallCompiler::compileFromFile can't open input file \"%s\".\n", fileName));
        }

    myTokens = CTokenContainer ( 1000);
    myMessages = CMessageContainer ( 100);

    CTokenAnalyzer tanalyzer( &myTokens, fileIn, fileName, &myMessages);

    bool res = tanalyzer.analyze();

    //myTokens.print();

    CSyntaxAnalyzer sanalyzer ( &myTokens, (CSmallVM*)smVM, &myMessages);

    SmCompiler::Console->RaiseCommand ( "pause3d");

    res = res && sanalyzer.analyze();

    SmCompiler::Console->RaiseCommand ( "pause3d");

    myMessages.print();

    return res;
    }
```

Файл SmallCompiler\system.cpp

```
#include "system.h"

ILoader*        SmCompiler::Loader = 0;
ILinker*        SmCompiler::Linker = 0;
CSmallVM*       SmCompiler::SmVM = 0;
ISmallCompiler* SmCompiler::Compiler = 0;
IConsole*       SmCompiler::Console = 0;
CScope*         SmCompiler::GS = 0;
```

Файл SmallCompiler\tokencore.cpp

```
#include "TokenCore.h"
#include "system.h"

#include <string>

//-----------------------------------------------------------------------------------------------------------

CPosition::CPosition (CString fname, int pos, int strnum, int symbnum) :
    myFileName     (fname),
    myPosition     (pos),
    myStringNumber (strnum),
    mySymbolNumber (symbnum)
    { }

CPosition::CPosition() :
    myFileName     (0),
    myPosition     (-1),
    myStringNumber (-1),
    mySymbolNumber (-1)
    { }

bool CPosition::operator == ( const CPosition& position)
    {
        return
            myFileName == position.rFileName() &&
            myPosition == position.rPos () &&
            myStringNumber == position.rStr() &&
            mySymbolNumber == position.rSymb();
    }

//-----------------------------------------------------------------------------------------------------------

CMessage::CMessage() :
    myType (0),
    myText (NULL),
    myPos (NULL)
    { }

CMessage::CMessage (int type, const CPosition* pos, const char* text) :
    myType (type),
    myText (text? new char [ strlen (text) + 1] : NULL),
    myPos (pos? new CPosition (*pos) : NULL)
```

```
    {
    if (text) strcpy (myText, text);
    }

CMessage::CMessage (const CMessage& m) :
    myType (m.rType()),
    myText (m.rText()? new char [ strlen (m.rText()) + 1] : NULL),
    myPos (m.rPos()? new CPosition (*m.rPos()) : NULL)
    {
    if (m.rText()) strcpy (myText, m.rText());
    }

CMessage& CMessage::operator= (const CMessage& m)
    {
    myType = m.rType();
    myText = m.rText()? new char [ strlen (m.rText()) + 1] : NULL;
    myPos = m.rPos()? new CPosition (*m.rPos()) : NULL;
    if (m.rText()) strcpy (myText, m.rText());
    return *this;
    }
CMessage::CMessage (int type, const CPosition* pos, int length, const char* format, ...) :
    myType (type),
    myText (length && format? new char [length+1] : NULL),
    myPos (pos? new CPosition (*pos) : NULL)
    {
    va_list args;
    va_start (args, format);
    if (!length || !format)
        { ASSERTPARAM (length && format);
          return; }
    Assert ( _vsnprintf (myText, length, format, args) >= 0);
    }

CMessage::~CMessage()
    {
    if (myText) delete[] myText; if (myPos) delete myPos;
    }

void CMessage::print (FILE*)
    {
    if (!myPos)
        { SmCompiler::Console->printf ( "%d: \"%s\"\n", myType, myText? myText : "No message"); return; }
    else SmCompiler::Console->printf ( "%s(%d) : message %d: %s",
        myPos->rFileName()? myPos->rFileNameStr() : "No file", myPos->rStr(),
        myType, myText? myText : "No message");
    }

//-------------------------------------------------------------------------------------------------------

CMessageContainer::CMessageContainer (int n) :
    CObjectsContainer<CMessage> (n)
    { }

void CMessageContainer::print (FILE* file)
    {
    for (int i = 0; i < myNumber; i++)
        myObjects[i].print (file);
    }

//-------------------------------------------------------------------------------------------------------

CSourceCode::CSourceCode (FILE* File, CString fname) :
    myFile (File),
    myCurrentPos (fname, 0, 0, 0),
    myLastGivenPos (fname, 0, 0, 0),
    mySavedPos ( 10)
    { }
/*  CSourceCode (const CSourceCode &str) :
        myFile (str.rFile()),
        myCurrentPos (str.rCurrentPos()),
        myLastGivenPos (str.rLastGivenPos())
        { } */
CSourceCode::~CSourceCode()
    { }

bool CSourceCode::ok() const
    {
    return myFile != NULL;
    }

bool CSourceCode::isEndOfFile() const
    {
    return const_cast<CSourceCode*> ( this) -> showNextNonSpaceSymbol()? 0 : 1;
    }

bool CSourceCode::isEOF() const
    {
    return isEndOfFile();
    }

char CSourceCode::eatSymbol()
    {
    char res = fgetc (myFile);
    getPos()++;
    getSymbolNum()++;
    return res;
    }
```

```
CTokenPosition CSourceCode::updateLastGivenPos()
    {
    return myLastGivenPos = myCurrentPos;
    }

int CSourceCode::skipSpaces()
    {
    int i = 0;
    while (1)
        {
        char ch = showNextSymbol();
        switch (ch)
            {
            case ' '  : eatSymbol(); break;
            case '\t' :
                getSymbolNum() += TabSize;
                fgetc(myFile);
                getPos()++;
                break;
            case '\n' :
                getFileNum()++;
                getSymbolNum() = 0;
                fgetc(myFile);
                getPos()++;
                break;
            case '\r' :
                getSymbolNum() = 0;
                fgetc(myFile);
                getPos()++;
                break;
            case '"'  :
                i += skipComments() - 1 /*because "i++" later*/;
                break;
            default  : return i;
            }
        i++;
        }
    }

int CSourceCode::skipComments()
    {
    Assert (getNextSymbol() == '"');

    char symbol = 0;
    for (int i = 0;  (symbol = getNextSymbol()) != '"' && symbol != 0; i++)
        ;
    return i;
    }

char CSourceCode::getNextSymbol()
    {
    char res = showNextSymbol();
    if (res == '\0')
        return '\0';
    switch (res)
        {
        case '\t' :
            getSymbolNum() += TabSize;
            fgetc (myFile);
            getPos()++;
            break;
        case '\n' :
            getFileNum()++;
            getSymbolNum() = 0;
            fgetc (myFile);
            getPos()++;
            break;
        case '\r' :
            getSymbolNum() = 0;
            fgetc (myFile);
            getPos()++;
            break;
        default   : eatSymbol();
        };
    return res;
    }

char CSourceCode::getNextNonSpaceSymbol()
    {
    skipSpaces();
    return getNextSymbol();
    }

char CSourceCode::showNextSymbol()
    {
    savePos();
    char t = fgetc (myFile);
    loadPos();
    return t == EOF? 0 : t;
    }

char CSourceCode::showNextNonSpaceSymbol()
    {
    savePos();
    char t = getNextNonSpaceSymbol(); // I'll be back!
    loadPos();
```

```
    return t;
    }

void CSourceCode::savePos()
    {
    BinFilePos p;
    p.mySavedFilePos = ftell ( myFile);
    p.mySavedPos = myCurrentPos;
    mySavedPos.push ( p);
    }
void CSourceCode::loadPos()
    {
    BinFilePos p = mySavedPos.pop();
    fseek ( myFile, p.mySavedFilePos, SEEK_SET);
    myCurrentPos = p.mySavedPos;
    }

//---------------------------------------------------------------------------------------------------

CToken::CToken (types type, const CTokenPosition &pos, const Atom& data) :
    myType (type),
    myPos  (pos),
    myData (data)
    { }

CToken::CToken() :
    myType (eUnknown),
    myData (0L)
    { }

bool CToken::operator== (const CToken& token) const
    { return 0; } //!!! - TODO

void CToken::print() const
    {
    if (myPos.rFileName()) SmCompiler::Console->printf ( "\"%s\", ", myPos.rFileName()?
myPos.rFileNameStr() : "<<no file>>");
    SmCompiler::Console->printf ( "pos %04d, str: #%03d, symb: #%03d - ", myPos.rPos(), myPos.rStr(),
myPos.rSymb());

    switch (myType)
        {
        case eOpenBlockBracket :
            SmCompiler::Console->printf ( "open block bracket: \"[\"\n"); break;
        case eCloseBlockBracket :
            SmCompiler::Console->printf ( "close block bracket: \"]\"\n"); break;
        case eDoubleDot :
            SmCompiler::Console->printf ( "double dot: \":\"\n"); break;
        case eDot :
            SmCompiler::Console->printf ( "dot: \".\"\n"); break;
        case eDotComa :
            SmCompiler::Console->printf ( "dot: \";\"\n"); break;
        case eOpenBracket :
            SmCompiler::Console->printf ( "open bracket: \"(\"\n"); break;
        case eCloseBracket :
            SmCompiler::Console->printf ( "close bracket: \")\"\n"); break;
        case eAssignmentOperator :
            SmCompiler::Console->printf ( "assignmentOperator: \":=\"\n"); break;
        case eReturnOperator :
            SmCompiler::Console->printf ( "returnOperator: \"^\"\n"); break;
        case eQuotedCharacter :
            SmCompiler::Console->printf ( "quotedCharacter: \"%c\"\n", myData.Symbol); break;
        case eIdentifier :
            SmCompiler::Console->printf ( "identifier: \"%s\"\n", rString().rString()); break;
        case eKeyword :
            SmCompiler::Console->printf ( "keyword: \"%s\"\n", rString().rString()); break;
        case eBinarySelector :
            SmCompiler::Console->printf ( "binarySelector: \"%s\"\n", rString().rString()); break;
        case eQuotedString :
            SmCompiler::Console->printf ( "quotedString: \"%s\"\n", rString().rString()); break;
        case eQuotedSelector :
            SmCompiler::Console->printf ( "quotedSelector: \"%s\"\n", rString().rString()); break;
        case eHashedString :
            SmCompiler::Console->printf ( "hashedString: \"%s\"\n", rString().rString()); break;
        case eKeywordSelector :
            SmCompiler::Console->printf ( "keywordSelector: \"%s\"\n", rString().rString()); break;
        case eFloat :
            SmCompiler::Console->printf ( "float:%lg\n", myData.Double); break; //!!! fprintf rounds to
4rd positions after coma
        case eInteger :
            SmCompiler::Console->printf ( "integer:%d\n", myData.Integer); break;
        default :
            SmCompiler::Console->printf ( "type is %d...\n", myType); break;
        }
    }
//  enum { eUnknown, eIdentifier, eKeyword, eBinarySelector, eReturnOperator, eAssignmentOperator,
//         eInteger, eFloat, eQuotedCharacter, eQuotedSring, eQuotedSelector, eHashedString,
eKeywordSelector }; //to be continued

//---------------------------------------------------------------------------------------------------

CTokenContainer::CTokenContainer (int lecsQuant) :
    CObjectsContainer<CToken> (lecsQuant),
    myTexts (lecsQuant)
    { myTexts.add (NULL); }

bool CTokenContainer::ok() const
```

```
    {
    return CObjectsContainer<CToken>::ok() && myTexts.ok();
    }

void CTokenContainer::print ( int n) const
    {
        Assert (CTokenContainer::ok());
        if ( n != -1)
            {
                myObjects[n].print();
                return;
            }

        for ( int i = 0; i < myNumber; i++)
            {
                SmCompiler::Console->printf ( "l#%04d: ", (int)i); myObjects[i].print();
            }
    }

//---------------------------------------------------------------------------------------------------------

CTokenAnalyzer::CTokenAnalyzer (CTokenContainer* containerPtr, const CSourceCode& srcStr,
                                                                CMessageContainer* mess) :
    mySrc (srcStr),
    myTokens (containerPtr),
    myMessages (mess)
    { }

CTokenAnalyzer::CTokenAnalyzer (CTokenContainer* containerPtr, FILE* file, CString fname,
                                                                CMessageContainer* mess) :
    mySrc (file, fname),
    myTokens (containerPtr),
    myMessages (mess)
    { }

bool CTokenAnalyzer::ok()
    {
    return mySrc.ok() && myTokens && myTokens->ok() && myMessages && myMessages->ok();
    }

bool CTokenAnalyzer::genToken()
    { Assert (ok());
    mySrc.skipSpaces();
    mySrc.updateLastGivenPos();

    char currentSymbol = mySrc.showNextSymbol();
    switch (currentSymbol)
        {
        case '[' :
            mySrc.getNextSymbol();
            addToken ( CToken::eOpenBlockBracket, Atom());
            return 1;
        case ']' :
            mySrc.getNextSymbol();
            addToken ( CToken::eCloseBlockBracket, Atom());
            return 1;
        case '.' :
            mySrc.getNextSymbol();
            addToken ( CToken::eDot, Atom());
            return 1;
        case ';' :
            mySrc.getNextSymbol();
            addToken ( CToken::eDotComa, Atom());
            return 1;
        case '(' :
            mySrc.getNextSymbol();
            addToken ( CToken::eOpenBracket, Atom());
            return 1;
        case ')' :
            mySrc.getNextSymbol();
            addToken ( CToken::eCloseBracket, Atom());
            return 1;
        case '^' :
            addToken ( CToken::eReturnOperator, Atom());
            mySrc.getNextSymbol();
            return 1;
        case '$' :
            mySrc.getNextSymbol();
            addToken ( CToken::eQuotedCharacter, Atom ( mySrc.getNextSymbol()));
            if ( IdentifierLetter.includes ( mySrc.showNextSymbol()))
                addMessage ( CMessage ( CMessage::eLexicalError, &mySrc.rLastGivenPos(), 50,
                "Symbol $ can be followed by one symbol only\n"));
            return 1;
        case ':' :
            mySrc.getNextSymbol();
            if (mySrc.showNextSymbol() == '=') {
                mySrc.getNextSymbol();
                addToken (CToken::eAssignmentOperator, Atom());
                }
            else
                addToken ( CToken::eDoubleDot, Atom());
            return 1;
        case '#' :
            mySrc.getNextSymbol();
            if ( mySrc.showNextSymbol() == '\'')
                return getQuotedString ( 1);
            return getQuotedSelector();
```

```
        case '\'' : return getQuotedString();
        case '\0' : return 0;
        }

    if (Letter.includes (currentSymbol)) return getIdentifier();
    if (Number.includes (currentSymbol)) return getNumber();
    if (BinaryLetter.includes (currentSymbol)) return getBinarySelector();

    mySrc.getNextNonSpaceSymbol();
    addMessage
        ( CMessage (CMessage::eLexicalError, &mySrc.rLastGivenPos(), 50,
                "not possible symbol '%c' to start a token\n", currentSymbol));
    return 1;
    }

CTokenAnalyzer& CTokenAnalyzer::savePos()
    {
    myTempFilePos = ftell ( mySrc.rFile());
    myTempTokenPos = mySrc.rCurrentPos();
    return *this;
    }

CTokenAnalyzer& CTokenAnalyzer::loadPos()
    {
    fseek (mySrc.rFile(), myTempFilePos, SEEK_SET);
    mySrc.getCurrentPos() = myTempTokenPos;
    return *this;
    }

bool CTokenAnalyzer::getQuotedString ( bool hashed)
    { Assert (CTokenAnalyzer::ok());

    char c = mySrc.getNextSymbol();
    Assert ( c == '\'' && "CTokenAnalyzer::getQuotedString() : first symbol isn't quote");

    savePos();

    CCont<char> str (10);

    c = mySrc.getNextSymbol();
    while (1)
        {
        if ( c == '\n' || c == '\r')
            {
            addMessage ( CMessage (CMessage::eLexicalError, &mySrc.rLastGivenPos(),
                30, "unterminated quoted string\n"));
            break;
            }
        if ( c == '\'')
            {
            if ( mySrc.showNextSymbol() == '\'')
                mySrc.getNextSymbol();
            else
                break;
            }
        str.add ( c);
        c = mySrc.getNextSymbol();
        }
    str.add ( '\0');

    loadPos();
    for ( int i = 0; i < str.rNumber(); i++)
        mySrc.getNextSymbol();

    myTokens->myTexts.add ( str.rObjects());
    StringPtr s;
    s.contPtr = & myTokens->myTexts;
    s.strNum = myTokens->myTexts.rNumber() - 1;
    addToken ( hashed? CToken::eHashedString : CToken::eQuotedString, Atom ( s));

    return 1;
    }

bool CTokenAnalyzer::getNumber()
    {
    Assert (CTokenAnalyzer::ok());

    int type = CToken::eFloat;

    savePos();

    char next = mySrc.getNextSymbol();
    for ( int s = 1; Number.includes ( next); s++)
        next = mySrc.getNextSymbol();
    if ( next != '.' && next != 'e' && next != 'd' && next != 'q')
        type = CToken::eInteger;
    if ( next == '.')
        if ( ! Number.includes ( next = mySrc.getNextSymbol()))
            type = CToken::eInteger;
        else
            for ( s++; Number.includes ( next); s++)
                next = mySrc.getNextSymbol();

    if ( next == 'e' || next == 'd' || next == 'q')
        for ( next = mySrc.getNextSymbol(), s++; Number.includes ( next); s++)
            next = mySrc.getNextSymbol();
    if ( IdentifierLetter.includes ( next))
```

```
        addMessage ( CMessage (CMessage::eLexicalError, &mySrc.rLastGivenPos(),
            50, "numbers can't be followed by letters\n"));

    loadPos();

    char* startString = new char[s];
    for ( int i = 0; i < s - 1; i++)
        startString[i] = mySrc.getNextSymbol();
    startString[i] = '\0';
    char* continueString = NULL;

    if ( type == CToken::eInteger)
        {
        int l = strtol (startString, &continueString, 10);
        addToken ( CToken::eInteger, Atom ( l));
        }
    else
        {
        double d = strtod (startString, &continueString);
        addToken ( CToken::eFloat, Atom ( d));
        }

    Assert ( startString + s - 1 == continueString);
    return 1;
    }

bool CTokenAnalyzer::getIdentifier()
    { Assert (ok());

    savePos();

    int l = 0; char c = mySrc.getNextSymbol();
    CToken::types type = CToken::eIdentifier;
    while (1)
        {
        if (c == ':') { type = CToken::eKeyword; l++; break; }
        if ( !IdentifierLetter.includes (c)) break;
        c = mySrc.getNextSymbol();
        l++;
        }

    loadPos();

    char* str = new char [l+1];
    for (int i = 0; i < l; i++) str [i] = mySrc.getNextSymbol();
    str[i] = '\0';

    myTokens->myTexts.add (str);
    StringPtr s;
    s.contPtr = & myTokens->myTexts;
    s.strNum = myTokens->myTexts.rNumber() - 1;
    addToken ( type, Atom ( s));
    return 1;
    }

bool CTokenAnalyzer::getBinarySelector ( bool quoted)
    { Assert (CTokenAnalyzer::ok());

    savePos();

    for (int l = 0; BinaryLetter.includes  (mySrc.getNextSymbol()); l++) ;

    loadPos();

    char* str = new char [l + 1];
    for (int i = 0; i < l; i++) str[i] = mySrc.getNextSymbol();
    str[i] = '\0';

    myTokens->myTexts.add (str);
    StringPtr s;
    s.contPtr = & myTokens->myTexts;
    s.strNum = myTokens->myTexts.rNumber() - 1;
    addToken ( quoted? CToken::eQuotedSelector : CToken::eBinarySelector, Atom ( s));
    return 1;
    }

bool CTokenAnalyzer::getQuotedSelector()
    { Assert (ok());

    if ( BinaryLetter.includes ( mySrc.showNextSymbol())) return getBinarySelector ( 1);

    CCont<char> str (10);

    while (1)
        {
        char c = mySrc.showNextSymbol();
        if ( c != ':' && ! IdentifierLetter.includes ( c))
                break;
        c = mySrc.getNextSymbol();
        str.add ( c);
        }
    str.add ( '\0');

    myTokens->myTexts.add ( str.rObjects());
    StringPtr s;
    s.contPtr = & myTokens->myTexts;
    s.strNum = myTokens->myTexts.rNumber() - 1;
```

```
    addToken ( CToken::eQuotedSelector, Atom ( s));
    return 1;
    }

void CTokenAnalyzer::addMessage (const CMessage& mess)
    { Assert (CTokenAnalyzer::ok());
      myMessages->add (mess); }

int CTokenAnalyzer::addToken ( CToken::types type, const Atom &data)
    { Assert ( CTokenAnalyzer::ok());
    return myTokens->add ( CToken ( type, mySrc.rLastGivenPos(), data));
    }

bool CTokenAnalyzer::analyze()
    { Assert (CTokenAnalyzer::ok());
    while ( !mySrc.isEndOfFile())
        if ( !genToken()) return 0;
    return 1;
    }

//-----------------------------------------------------------------------------------------------------------

std::pair < std::vector<int>, bool> VariableList ( const CToken& tok)
    { Assert ( tok.rType() == CToken::eQuotedString);
    std::vector<int> v = std::vector<int>();
    std::string str;

    std::string in = tok.rString().rString();

    bool b = 1;
    for ( std::string::const_iterator i = in.begin(); i != in.end(); ++i)
        if ( CTokenAnalyzer::IdentifierLetter.includes ( *i)) {
            str += *i;
            continue;
            }
        else {
            if ( ! str.empty()) {
                v.push_back ( SmCompiler::Linker->registerName ( str));
                str = std::string();
                }
            if ( b && ! CTokenAnalyzer::WhiteSpace.includes ( *i))
                b = 0;
            }
    if ( ! str.empty())
        v.push_back ( SmCompiler::Linker->registerName ( str));

    return std::pair < std::vector<int>, bool> ( std::vector<int> ( v), b);
    }

//-----------------------------------------------------------------------------------------------------------

CLetters CTokenAnalyzer::UppercaseEnglishAlphabetic
    ("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
CLetters CTokenAnalyzer::LowercaseEnglishAlphabetic
    ("abcdefghijklmnopqrstuvwxyz");
CLetters CTokenAnalyzer::UppercaseRussianAlphabetic
    ("└┴┬├─┼╞╟╚╔╩╦╠═╬╧╨╤╥╙╘╒╓╫╪┘┌■▄ ▀");
CLetters CTokenAnalyzer::LowercaseRussianAlphabetic
    ("рстуфхцчшщъыьэюяЁёЄєЇїЎў°•·√№¤■ ");
CLetters CTokenAnalyzer::NonCaseLetter
    ("_");
CLetters CTokenAnalyzer::EnglishAlphabetic
    (NULL,
     CTokenAnalyzer::UppercaseEnglishAlphabetic,
     CTokenAnalyzer::LowercaseEnglishAlphabetic);
CLetters CTokenAnalyzer::RussianAlphabetic
    (NULL,
     CTokenAnalyzer::UppercaseRussianAlphabetic,
     CTokenAnalyzer::LowercaseRussianAlphabetic);
CLetters CTokenAnalyzer::UppercaseAlphabetic
    (NULL,
     CTokenAnalyzer::UppercaseEnglishAlphabetic,
     CTokenAnalyzer::UppercaseRussianAlphabetic);
CLetters CTokenAnalyzer::LowercaseAlphabetic
    (NULL,
     CTokenAnalyzer::LowercaseEnglishAlphabetic,
     CTokenAnalyzer::LowercaseRussianAlphabetic);

//CLetters CTokenAnalyzer::Letter (NULL);

CLetters CTokenAnalyzer::Letter
    (NULL,
     CTokenAnalyzer::UppercaseAlphabetic,
     CTokenAnalyzer::LowercaseAlphabetic,
     CTokenAnalyzer::NonCaseLetter);

CLetters CTokenAnalyzer::Number ("0123456789");
CLetters CTokenAnalyzer::IdentifierLetter
    (NULL, CTokenAnalyzer::Number, CTokenAnalyzer::Letter);

CLetters CTokenAnalyzer::BinaryLetter
    ("!%&*+,/<=>?@~|-\\");
CLetters CTokenAnalyzer::WhiteSpace
    (" \t\r\n");
```

Файл SmallCompiler\syntaxcore.cpp

```
#include "SyntaxCore.h"

CSyntaxAnalyzer::CSyntaxAnalyzer ( CTokenContainer* cont, CSmallVM* vm, CMessageContainer* mesCont) :
    myTokenCont ( cont),
    myMesCont ( mesCont),
    myPointer ( cont),
    myVM ( vm),
    myLastMethodName ( 0)
    { }

bool CSyntaxAnalyzer::ok() const
    {
    return myTokenCont && myMesCont && myVM ;
    }

const CPosition* CSyntaxAnalyzer::rTokenPos() const throw()
    {
    return myPointer? myPointer->rPos() : NULL;
    }

void CSyntaxAnalyzer::addMessage ( const CString& str, CMessage::MessageTypes t)
    {
    addMessage ( t, str.rLength()+1, str.rString());
    }

void CSyntaxAnalyzer::addMessage (int type, int length, const char* format, ...)
    { Assert (ok());
    va_list args;
    va_start (args, format);
    myMesCont->add ( CMessage ( type, rTokenPos(), length, format, args));
    }

void CSyntaxAnalyzer::addMessage (const CMessage& mess)
    { Assert (CSyntaxAnalyzer::ok());
    myMesCont->add (mess);
    }

bool CSyntaxAnalyzer::analyze()
    {
    try
        { Assert (ok());
        bool contin = 1;

        if ( contin && myPointer->rType() == CToken::eIdentifier && myPointer->rString() == "Smalltalk")
myPointer++;
        else contin = 0, addMessage ( "\"Smalltalk\" expected. Enter a correct Interchange Version
Identifier\n");
        if ( contin && myPointer->rType() == CToken::eKeyword && myPointer->rString() ==
"interchangeVersion:") myPointer++;
        else contin = 0, addMessage ( "\"interchangeVersion:\" expected. Enter a correct Interchange
Version Identifier\n");
        if ( contin && myPointer->rType() == CToken::eQuotedString && myPointer->rString() == "1.0")
myPointer++;
        else addMessage ( "This implementation is developed to support '1.0' standart.\n"), myPointer++;
        if ( contin && checkExclamationPoint()) myPointer++;
        else contin = 0, addMessage ( "Exclamation point \"!\" is expected.\n");

        if ( !contin)
            gotoNextInterchangeUnit();

        while (myPointer)
            {

            if ( myPointer->rType() != CToken::eIdentifier)
                {
                addMessage ("Interchange unit can start only with identifier. Searching next unit.");
                gotoNextInterchangeUnit();
                continue;
                }

            if ( myPointer->rString() == "Class")
                {
                if ( !eatClassDefinition()) {
                    addMessage ( "Class definition is wrong. Searching for next interchange unit.\n");
                    gotoNextInterchangeUnit();
                    }
                continue;
                }

            int identifier = SmCompiler::Linker->registerName ( myPointer->rString());
            SmallClass* pSmCl = SmCompiler::SmVM->getClassPtr ( identifier);
            if ( pSmCl) {
                myPointer++;
                if ( myPointer->rType() != CToken::eIdentifier) {
                    addMessage ( "Identifier expected.\n");
                    continue;
                    }

                if ( myPointer->rString() == "method") {
                    myPointer++;
```

```
                    if ( !checkExclamationPoint()) {
                        addMessage ( "Exclamation point expected.");
                        continue;
                        }
                    myPointer++; // $!

                    if ( ! eatMethodDefinition ( pSmCl)) {
                        CString str = "ClassMethod definition ( possibly '";
                               str << SmCompiler::Linker->getName ( myLastMethodName);
                               str << "') is wrong. Searching for next interchange unit.\n";
                        addMessage ( str);
                        gotoNextInterchangeUnit();
                        }
                    continue;
                    }

                if ( myPointer->rString() == "classInstanceMethod") {
                    myPointer++;
                    if ( !checkExclamationPoint()) {
                        addMessage ( "Exclamation point expected.");
                        continue;
                        }
                    myPointer++; // $!

                    if ( ! eatClassMethodDefinition ( pSmCl)) {
                        addMessage ( "ClassInstanceMethod definition is wrong. Searching for next
interchange unit.\n");
                        gotoNextInterchangeUnit();
                        }
                    continue;
                    }
                }
            else
                {
                    CString str = "Undefind class '";
                        str << SmCompiler::Linker->getName ( myLastMethodName);
                        str << myPointer->rString();
                        str << "'.";
                    addMessage ( str);
                    gotoNextInterchangeUnit();
                    continue;
                }
            }
        return 1;
        }
    catch ( unexpectedEOF)
        {
        addMessage ( "Fatal error. Unexpected end of file. Syntax analyzer is stopped.\n");
        return 0;
        }
    }

bool CSyntaxAnalyzer::eatClassDefinition()
    {
    myPointer++; //\"Class\"

    SmallClass* clS = NULL;

    if ( myPointer->rType() == CToken::eKeyword && myPointer->rString() == "named:") myPointer++;
    else addMessage ( "\"named:\" expected. No comments.\n");
    if ( myPointer->rType() == CToken::eQuotedString)
        {
        int name = SmCompiler::Linker->registerName ( myPointer->rString());
        SClassRef clL = myVM->registerClass ( name, name);
        if ( !clL)
            return addMessage ( "Global name for this class is already used", CMessage::eSystemError), 0;
        clS = SmCompiler::SmVM->getClassPtr ( name);
        myPointer++;
        }
    else return addMessage ( "Quoted string with name is expected\n"), 0;

    if ( myPointer->rType() == CToken::eKeyword && myPointer->rString() == "superclass:")
        {
        myPointer++;
        if ( !myPointer || myPointer->rType() != CToken::eQuotedString)
            addMessage ( "quoted string with name of superclass is expected");
        int name = SmCompiler::Linker->registerName ( myPointer->rString());
        myPointer++;
        if ( name)
            clS->regBaseClass ( name);
        }

    if ( myPointer->rType() == CToken::eKeyword && myPointer->rString() == "indexedInstanceVariables:")
        {
        myPointer++;
        addMessage ( "\"indexedInstanceVariables\" isn't realized yet. Ignored.\n",
CMessage::eInternalWarning);
        myPointer++; //eHashedString
        }

    if ( myPointer->rType() == CToken::eKeyword && myPointer->rString() == "instanceVariableNames:")
        {
        myPointer++;
        if ( !myPointer || myPointer->rType() != CToken::eQuotedString)
            addMessage ( "Quoted string as identifier list is expected after
'instanceVariableNames:'.\n");
        else {
```

```
            std::pair < std::vector<int>, bool> res = VariableList ( *myPointer);
            myPointer++;
            if ( !res.second)
                addMessage ( "Wrong symbols in identifier list of instance variable names\n");
            for ( std::vector<int>::iterator i = res.first.begin(); i != res.first.end(); ++i)
                {
                    clS->myObjPro.addVar ( *i);
                }
            }
        }

    if ( myPointer->rType() == CToken::eKeyword && myPointer->rString() == "classVariableNames:")
        {
        myPointer++;
        if ( !myPointer || myPointer->rType() != CToken::eQuotedString)
            addMessage ( "Quoted string as identifier list is expected after 'classVariableNames:'.\n");
        else {
            std::pair < std::vector<int>, bool> res = VariableList ( *myPointer);
            myPointer++;
            if ( !res.second)
                addMessage ( "Wrong symbols in identifier list of class variable names\n");
            for ( std::vector<int>::iterator i = res.first.begin(); i != res.first.end(); ++i)
                clS->addVar ( *i);
            }
        }

    if ( myPointer->rType() == CToken::eKeyword && myPointer->rString() == "sharedPools:")
        {
        myPointer++;
        if ( !myPointer || myPointer->rType() != CToken::eQuotedString)
            addMessage ( "Quoted string as identifier list is expected after 'sharedPools:'.\n");
        else {
            std::pair < std::vector<int>, bool> res = VariableList ( *myPointer);
            myPointer++;
            if ( !res.second)
                addMessage ( "Wrong symbols in identifier list of sharedPools\n");

            addMessage ( "\"sharedPools\" isn't realized yet. Ignored.\n", CMessage::eInternalWarning);
//          for ( std::vector<int>::iterator i = res.first.begin(); i != res.first.end(); ++i)
//              ...
            }
        }

    if ( myPointer->rType() == CToken::eKeyword && myPointer->rString() == "classInstanceVariableNames:")
        {
        myPointer++;
        if ( !myPointer || myPointer->rType() != CToken::eQuotedString)
            addMessage ( "Quoted string as identifier list is expected after
'classInstanceVariableNames:'.\n");
        else {
            std::pair < std::vector<int>, bool> res = VariableList ( *myPointer);
            myPointer++;
            if ( !res.second)
                addMessage ( "Wrong symbols in identifier list of class-instance variable names\n");
            for ( std::vector<int>::iterator i = res.first.begin(); i != res.first.end(); ++i)
                clS->addVar ( *i); //!!!
            }
        }
    if ( !myPointer)
        return 0;
    if ( ! checkExclamationPoint())
        return addMessage ( "Declaration of class must be finished by exclamation point\n"), 0;
    myPointer++; // $!
    return 1;
    }

bool CSyntaxAnalyzer::eatMethodDefinition ( SmallClass* pSmCl)
    {
    Assert ( "CSyntaxAnalyzer::eatMethodDefinition ( SmallClass* pSmCl)" && pSmCl);
    bool res = 1;

    SmallFunc func ( pSmCl->myClassName);
    func.myBlock = SmCompiler::SmVM->regBlock ( std::auto_ptr<Block> ( new Block()));
    BlockPtr pBlock = func.myBlock;

    if ( SmCompiler::Compiler->compileAsDebug) pBlock->addCommand ( CCommand ( CCommand::eSetWorkScope));
    updateWorkPos ( pBlock);

    res &= eatMethodHeaderTo ( &func);

    if ( myPointer->rType() == CToken::eBinarySelector && myPointer->rString() == "|")
        {
        myPointer++;
        res &= eatTemplatesTo ( pBlock);
        }

    res &= eatStatementTo ( pBlock);
    while ( *myPointer == CToken::eDot)
        {
        if ( *myPointer == CToken::eBinarySelector && myPointer->rString() == "!")
            break;
        pBlock->addCommand ( CCommand::ePop, 0);
        myPointer++;
        res &= eatStatementTo ( pBlock);
        }
    pBlock->addCommand ( CCommand::ePop, 0);
```

```
    if ( ! checkExclamationPoint()) return 0;
    myPointer++; //"!"
    pSmCl->addFunc ( func);

    return 1;
    }

bool CSyntaxAnalyzer::eatClassMethodDefinition ( SmallClass* pSmCl)
    {
    Assert ( "CSyntaxAnalyzer::eatClassMethodDefinition ( SmallClass* pSmCl)" && pSmCl);
    bool res = 1;

    SmallStaticFunc func ( pSmCl->myClassName);
    func.myBlock = SmCompiler::SmVM->regBlock ( std::auto_ptr<Block> ( new Block()));
    BlockPtr pBlock = func.myBlock;

    if ( SmCompiler::Compiler->compileAsDebug) pBlock->addCommand ( CCommand ( CCommand::eSetWorkScope));
    updateWorkPos ( pBlock);

    res &= eatMethodHeaderTo ( &func);

    if ( myPointer->rType() == CToken::eBinarySelector && myPointer->rString() == "|")
        {
        myPointer++;
        res &= eatTemplatesTo ( pBlock);
        }

    res &= eatStatementTo ( pBlock);
    while ( *myPointer == CToken::eDot)
        {
        if ( *myPointer == CToken::eBinarySelector && myPointer->rString() == "!")
            break;
        pBlock->addCommand ( CCommand::ePop, 0);
        myPointer++;
        res &= eatStatementTo ( pBlock);
        }
    pBlock->addCommand ( CCommand::ePop, 0);

    if ( ! checkExclamationPoint()) return 0;
    myPointer++; //"!"
    pSmCl->addStaticFunc ( func);

    return 1;
    }

bool CSyntaxAnalyzer::eatMethodHeaderTo ( ISmallFunc* pSmFunc)
    {
    Assert ( pSmFunc);

    switch ( myPointer->rType()) {
        case CToken::eIdentifier : {
            pSmFunc->myName = myPointer->rString();
            myPointer++;
            break;
            }
        case CToken::eBinarySelector : {
            pSmFunc->myName = myPointer->rString();
            myPointer++;
            if ( !myPointer || myPointer->rType() != CToken::eIdentifier) {
                addMessage ( "Argument name is expected.\n");
                return 0;
                }
            pSmFunc->myBlock->addLocalVar ( SmCompiler::Linker->registerName ( myPointer->rString()));
            myPointer++;
            pSmFunc->myTypes << SParam::eAny;
            break;
            }
        case CToken::eKeyword : {
            std::string fName = "";
            do {
                fName.append ( myPointer->rString().rString());
                myPointer++;
                if ( !myPointer || myPointer->rType() != CToken::eIdentifier) {
                    addMessage ( "Argument name is expected.\n");
                    return 0;
                    }
                pSmFunc->myBlock->addLocalVar ( SmCompiler::Linker->registerName ( myPointer->rString()));
                pSmFunc->myTypes << SParam::eAny;
                myPointer++;
                } while ( myPointer->rType() == CToken::eKeyword);
            pSmFunc->myName = SmCompiler::Linker->registerName ( fName);
            break;
            }
        default :
            addMessage ( "Method header is expected.\n");
            return 0;
        }
    myLastMethodName = pSmFunc->myName;
    return 1;
    }

bool CSyntaxAnalyzer::eatStatementTo ( BlockPtr pBlock)
    {
    if ( *myPointer == CToken::eReturnOperator) {
        pointer p = myPointer;
        myPointer++;
        bool res = eatExpressionTo( pBlock);
```

```
        updateWorkPos( pBlock, p);
        pBlock->addCommand ( CCommand::eReturn, 0);
        return res;
        }
    return eatExpressionTo ( pBlock);
    }

bool CSyntaxAnalyzer::eatTemplatesTo ( BlockPtr pBlock)
    {
    while ( myPointer->rType() != CToken::eBinarySelector && myPointer->rString() != "|")
        if ( myPointer->rType() != CToken::eIdentifier) {
            addMessage ( "Identifier is expected in templates block.\n");
            myPointer++;
            continue;
            }
        else
            pBlock->addLocalVar ( SmCompiler::Linker->registerName ( myPointer->rString())), myPointer++;

    myPointer++;
    return 1;
    }

bool CSyntaxAnalyzer::eatExpressionTo ( BlockPtr pBlock) //leaves at the top of stack its result
    {
    if ( myPointer[1] == CToken::eAssignmentOperator)
        return eatAssignmentTo ( pBlock);
    else return eatBasicExpressionTo ( pBlock);
    }

bool CSyntaxAnalyzer::eatAssignmentTo ( BlockPtr pBlock) //leaves at the top of stack its result
    {
    Assert ( myPointer[1] == CToken::eAssignmentOperator);

    if ( myPointer != CToken::eIdentifier)
        return addMessage ( "Variable name is expected before assignment operator \":=\".\n"), 0;
    const CString& lValueName = myPointer->rString();
    myPointer ++;
    pointer p = myPointer;
    myPointer++;

    bool res = eatExpressionTo ( pBlock);

    updateWorkPos( pBlock, p);
    pBlock->addCommand ( generateCommandTo ( pBlock, CCommand::ePop, lValueName));
    pBlock->addCommand ( generateCommandTo ( pBlock, CCommand::ePush, lValueName));

    return res;
    }

bool CSyntaxAnalyzer::eatBasicExpressionTo ( BlockPtr pBlock) //leaves at the top of stack its result
    {
    bool res = eatPrimaryTo ( pBlock);

    int localPrimaryIndex = saveTopTo ( pBlock);

    //if ( !eatMessageTo ( pBlock)) return 0;
    eatMessageTo ( pBlock);
    while ( *myPointer == CToken::eDotComa) {
        myPointer++;
        pBlock->addCommand ( CCommand ( CCommand::ePop, 0)); //forget result of message
        pBlock->addCommand ( CCommand ( CCommand::ePushL, localPrimaryIndex));
        if ( !eatMessageTo ( pBlock)) return 0;
        }

/*  Assert ( localPrimaryIndex == pSmFunc->myLocals.rNumber() - 1);
    pSmFunc->myLocals.del ( localPrimaryIndex);*/ // add command to control myLocals size at run-time

    return res;
    }

bool CSyntaxAnalyzer::eatPrimaryTo ( BlockPtr pBlock) //leaves at the top of the stack an object-receiver
    {
    if ( myPointer->rType() == CToken::eIdentifier) {
        pBlock->addCommand ( generateCommandTo ( pBlock, CCommand::ePush, myPointer->rString()));
        myPointer++;
        return 1;
        }

    if ( *myPointer == CToken::eOpenBracket) {
        myPointer++;
        bool res = eatExpressionTo ( pBlock);
        return *myPointer == CToken::eCloseBracket? myPointer++, res : 0;
        }

    if ( *myPointer == CToken::eInteger) {
        int i = myPointer->rInteger();
        myPointer++;
        pBlock->addCommand ( CCommand ( CCommand::ePushI, i));
        pBlock->addCommand ( CCommand ( CCommand::eNew, CCommand::stdTypes::eInteger));
        return 1;
        }

    if ( *myPointer == CToken::eFloat) {
        float d = myPointer->rDouble();
        myPointer++;
        pBlock->addCommand ( CCommand ( CCommand::ePushI, *reinterpret_cast<long*> ( &d)));
        pBlock->addCommand ( CCommand ( CCommand::eNew, CCommand::stdTypes::eFloat));
```

```
        return 1;
        }

    if ( *myPointer == CToken::eQuotedCharacter) {
        char c = myPointer->rSymbol();
        myPointer++;
        pBlock->addCommand ( CCommand ( CCommand::ePushI, c));
        pBlock->addCommand ( CCommand ( CCommand::eNew, CCommand::stdTypes::eChar));
        return 1;
        }
    if ( *myPointer == CToken::eQuotedString) {
        CString str = myPointer->rString();
        myPointer++;

        ConstStringPtr pStr = SmCompiler::SmVM->regConstString ( str.rString());

        pBlock->addCommand ( CCommand ( CCommand::ePushI, pStr.rIndex()));
        pBlock->addCommand ( CCommand ( CCommand::eNew, CCommand::stdTypes::eStr));
        return 1;
        }
    if ( *myPointer == CToken::eOpenBlockBracket) {
        myPointer++;

        BlockPtr bl = SmCompiler::SmVM->regBlock ( std::auto_ptr<Block> ( new Block()));
        if ( SmCompiler::Compiler->compileAsDebug) bl->addCommand ( CCommand ( CCommand::eSetWorkScope));

        bool res = 1;
        if ( myPointer->rType() == CToken::eDoubleDot)
            {
            myPointer++;
            res &= eatTemplatesTo ( bl);
            }
        res &= eatStatementTo ( bl);
        while ( *myPointer == CToken::eDot)
            {
            if ( *myPointer == CToken::eCloseBlockBracket)
                break;
            bl->addCommand ( CCommand::ePop, 0);
            myPointer++;
            res &= eatStatementTo ( bl);
            }
        bl->addCommand ( CCommand::ePop, 0);

        pBlock->addCommand ( CCommand ( CCommand::ePushI, bl.gIndex()));
        pBlock->addCommand ( CCommand ( CCommand::eNew, CCommand::stdTypes::eBlock));

        return *myPointer == CToken::eCloseBlockBracket? myPointer++, res : 0;
        }

    return 0;
    }

bool CSyntaxAnalyzer::eatMessageTo ( BlockPtr pBlock) //leaves at the top of the stack its result
    {
    bool res = 1;

    res &= eatUnaryMessagesTo ( pBlock);
    res &= eatBinaryMessagesTo ( pBlock);
    res &= eatKeywordMessageTo ( pBlock);

    return res;
    }

bool CSyntaxAnalyzer::eatUnaryMessagesTo ( BlockPtr pBlock) //leaves at the top of the stack its result
    {
    while ( *myPointer == CToken::eIdentifier) {
        updateWorkPos( pBlock);
        pBlock->addCommand ( CCommand::ePushI, 0); // zero parameters
        pBlock->addCommand ( CCommand::eCall, SmCompiler::Linker->registerName ( myPointer->rString()));
        myPointer++;
        }

    return 1;
    }

bool CSyntaxAnalyzer::eatBinaryMessagesTo ( BlockPtr pBlock) //leaves at the top of the stack its result
    {
    bool res = 1;

    while ( *myPointer == CToken::eBinarySelector && myPointer->rString() != "!") {
        pointer p = myPointer;
        int methodName = SmCompiler::Linker->registerName ( myPointer->rString());
        myPointer++;
        res &= eatPrimaryTo ( pBlock);
        res &= eatUnaryMessagesTo ( pBlock);
        updateWorkPos( pBlock, p);
        pBlock->addCommand ( CCommand::ePushI, 1); //one parameter
        pBlock->addCommand ( CCommand::eCall, methodName);
        }

    return res;
    }

bool CSyntaxAnalyzer::eatKeywordMessageTo ( BlockPtr pBlock) //leaves at the top of the stack its result
    {
    bool res = 1;
```

```
    std::string messageName = "";
    int argumentsN = 0;

    pointer p = myPointer;

    while ( *myPointer == CToken::eKeyword) {
        messageName.append ( myPointer->rString().rString());
        myPointer++;
        argumentsN++;
        res &= eatPrimaryTo ( pBlock);
        res &= eatUnaryMessagesTo ( pBlock);
        res &= eatBinaryMessagesTo ( pBlock);
        }

    if ( !argumentsN) return 1;

    updateWorkPos( pBlock, p);

    pBlock->addCommand ( CCommand::ePushI, argumentsN);
    pBlock->addCommand ( CCommand::eCall, SmCompiler::Linker->registerName ( messageName));

    return res;
    }

CCommand CSyntaxAnalyzer::generateCommandTo ( BlockPtr pBlock, CCommand::action action, const CString&
param) throw ( BadParam)
    {
    int name = SmCompiler::Linker->registerName ( param);
    bool local = 0;

    for ( int i = 1; i < pBlock->gScope().rVars().rNumber(); i++)
        if ( pBlock->gScope()[i] == name) {
            local = 1;
            break;
            }

    switch ( action)
        {
        case CCommand::ePush : return CCommand ( local? CCommand::ePushL : CCommand::ePushG, local? i :
name);
        case CCommand::ePop  : return CCommand ( local? CCommand::ePopL  : CCommand::ePopG,  local? i :
name);
        default: throw BadParam ( "Syntax analyzer: generateCommand : impossible action as param");
        }
    }

int CSyntaxAnalyzer::saveTopTo ( BlockPtr pBlock)
    {
    pBlock->addLocalVar ( 0);
    int res = pBlock->gScope().rVars().rNumber()-1;
    pBlock->addCommand ( CCommand ( CCommand::ePopL, res));
    pBlock->addCommand ( CCommand ( CCommand::ePushL, res));
    return res;
    }

void CSyntaxAnalyzer::eatToken ( CToken::types tokType, const CString& val, const CString& err,
CMessage::MessageTypes mesType)
    {
    if ( !myPointer || myPointer->rType() != tokType || myPointer->rString() != val)
        addMessage ( err, mesType);
    }

bool CSyntaxAnalyzer::checkToken (CString comments)
    {
    if (myPointer) return 1;
    if (!comments)
        addMessage ( CMessage::eUnexpectedEOF, 30, "unexpected end of file.");
    else
        addMessage ( CMessage::eUnexpectedEOF, 35 + comments.rLength(),
                                "unexpected end of file, %s", comments.rString());
    return 0;
    }

bool CSyntaxAnalyzer::checkExclamationPoint()
    {
    return myPointer->rType() == CToken::eBinarySelector && myPointer->rString() == "!";
    }

void CSyntaxAnalyzer::gotoNextInterchangeUnit()
    {
    while ( !checkExclamationPoint())
        myPointer++;
    myPointer++;
    }

void CSyntaxAnalyzer::updateWorkPos ( BlockPtr pBlock, pointer p)
    {
        if ( ! SmCompiler::Compiler->compileAsDebug)
            return;
        if ( !p)
            p = myPointer;
        pBlock->addCommand ( CCommand::ePushI, p->rPos()->rFileName());
        pBlock->addCommand ( CCommand::eUpdateWorkPosition, p->rPos()->rPos());
    }

//---------------------------------------------------------------------------------------------------
```

```
/*
implementation features:
    none yet
*/

/*
question:
    exampleClass method !
        key: arg
        arg := 0.
        ^arg.
        !
    exampleClass method !
        func1
        | t |
        t := 1.
        self key: t.
        " ўхьє ёртхэ t "
        ...
        !

*/
```

Файл SmallDebugger\DebuggerCore.cpp

```
#define __MODULE_NAME "Debugger"
#define __PROGRAMMER "Tankisto"

#include "DebuggerCore.h"
#include "system.h"
#include "..\SmalltalkEditor\Editor.h"
#include <exeption.h>
//#define __FULLDEBUG

Breakpoint::Breakpoint()
    {
        myPosition.myFileName = 0;
        myPosition.myFirstPos = -1;
        myPosition.myLastPos = -2;
    }

Breakpoint::Breakpoint ( const BreakpointPosition &position) :
    myPosition ( position)
    {
    }

bool Breakpoint::covers ( const BreakpointPosition& position)
    {
        return myPosition.myFileName == position.myFileName &&
               myPosition.myFirstPos <= myPosition.myLastPos &&
               myPosition.myFirstPos <= position.myFirstPos &&
               myPosition.myLastPos >= position.myLastPos;
    }

bool Breakpoint::covers ( const SymbolPosition& position)
    {
        return myPosition.myFileName == position.myFileName &&
               myPosition.myFirstPos <= position.myPosition &&
               myPosition.myLastPos >= position.myPosition;
    }

bool Breakpoint::operator == ( const Breakpoint& breakpoint)
    {
        return myPosition.myFileName == breakpoint.rPosition().myFileName &&
               myPosition.myFirstPos == breakpoint.rPosition().myFirstPos &&
               myPosition.myLastPos  == breakpoint.rPosition().myLastPos;
    }

//---------------------------------------------------------------------------------------------

CSmallDebugger::CSmallDebugger() :
    myPause ( 0),
    myBreakpoints ( 20)
    {
        myLastPosition.myFileName = 0;
        myLastPosition.myPosition = -1;
        myBreakpoints.add ( Breakpoint ());
    }

void CSmallDebugger::updateWorkPosition ( SymbolPosition position)
    {
    myLastPosition = position;
    int b = 0;

    #ifdef __FULLDEBUG
    SmDebugger::Console->printf ("DEBUGGER: Current position is %d\n", position.myPosition);
    #endif

    while ( 1)
        {
        if ( myCommand == eRun) {
```

```
            if ( ( myCurBreakpointIndex = checkBreakpoint ( position)))
                {
                myCommand = ePause;

                SmDebugger::Console->printf ("DEBUGGER: Stopped at breakpoint no %d at position %d\n",
myCurBreakpointIndex, position.myPosition);

                SmDebugger::Editor->stoppedAt ( position.myFileName, position.myPosition, b);
                }
            break;
            }
        else if ( myCommand == eStop) {
            myCommand = ePause;
            SmDebugger::Editor->stoppedAt ( myLastPosition.myFileName, myLastPosition.myPosition, 0);
            SmDebugger::Console->printf ("DEBUGGER: Stopped at position %d\n", position.myPosition);
            }
        else if ( myCommand == ePause) {
            continue;
            }
        else if ( myCommand == eStepIn) {
            static bool stop = 0;
            if ( stop == false) { //here pressed stepIn
                stop = true;
                break;
                }
            else { //here we stop
                stop = false;
                myCommand = ePause;
                SmDebugger::Editor->stoppedAt ( myLastPosition.myFileName, myLastPosition.myPosition, 0);
                }
            }
        else if ( myCommand == eStep) {
            static int depth = -1;
            if ( depth == -1) { //here pressed step
                depth = SmDebugger::SmVM->workScopeDepth();
                break;
                }
            else { //here we stop
                if ( depth <= SmDebugger::SmVM->workScopeDepth()) {
                    depth = -1;
                    myCommand = ePause;
                    SmDebugger::Editor->stoppedAt ( myLastPosition.myFileName, myLastPosition.myPosition,
0);
                    }
                }
            }
        else if ( myCommand == eStepOut) {
            static int depthOut = 0;
            if ( depthOut == 0) { //here pressed step
                depthOut = SmDebugger::SmVM->workScopeDepth();
                break;
                }
            else { //here we stop
                if ( depthOut <= SmDebugger::SmVM->workScopeDepth() - 1) {
                    depthOut = 0;
                    myCommand = ePause;
                    SmDebugger::Editor->stoppedAt ( position.myFileName, position.myPosition, 0);
                    }
                }
            }
        }
    }

int CSmallDebugger::setBreakpoint ( BreakpointPosition pos)
    {
        myBreakpoints.add ( pos);
        #ifdef __FULLDEBUG
        SmDebugger::Console->printf ("DEBUGGER: Breakpoint added.\n");
        #endif

        return myBreakpoints.rNumber() - 1;
    }

int CSmallDebugger::getBreakpointsNumber()
    {
        return myBreakpoints.rNumber();
    }

void CSmallDebugger::delBreakpoint ( int index)
    {
        if ( !index) return ;
        myBreakpoints.del ( index);
        myCurBreakpointIndex = 0;
    }

int CSmallDebugger::getCurBreakpointIndex()
    {
        return myCurBreakpointIndex;
    }

BreakpointPosition CSmallDebugger::getBreakpoint ( int index)
    {
        return myBreakpoints [ index ];
    }

int CSmallDebugger::checkBreakpoint ( SymbolPosition position)
    {
```

```
        for ( int i = 1; i < myBreakpoints.rNumber(); i++)
            if ( myBreakpoints[i].covers ( position))
                return i;
        return 0;
    }

void CSmallDebugger::pauseOn()
    {
        #ifdef __FULLDEBUG
        SmDebugger::Console->printf ( "DEBUGGER: CSmallDebugger::pauseOn() called.\n");
        #endif
        SmDebugger::Console -> RaiseCommand ( "pause3d on");
        myCommand = ePause;
    }
void CSmallDebugger::pauseOff()
    {
        #ifdef __FULLDEBUG
        SmDebugger::Console->printf ( "DEBUGGER: CSmallDebugger::pauseOff() called.\n");
        #endif

        SmDebugger::Console -> RaiseCommand ( "pause3d off");
        myCommand = eRun;
    }

IMPLEMENT_IUNKNOWN ( CSmallDebugger, ISmallDebugger, IID_ISmallDebugger);
```

Файл SmallDebugger\Module.cpp

```
#define __PROGRAMMER   "Se"
#define __MODULE_NAME  "SmallDebugger"

#include "System.h"
#include "DebuggerCore.h"

#include "..\ExeptionEngine\exeption.h"

const int Version1 = 0,
          Version2 = 9;

    #define DLL_NAME "SmallDebugger"
    #define DLL_PROGRAMMER "Tankisto"
    #define DLL_CLASS CSmallDebugger
    #include "..\Common\Utilities\Module.c"

#pragma warning (disable: 4275) // non dll-interface class used as base for dll-interface class

DLL_ONLOAD (loader)
{
    SmDebugger::Loader = loader;
    SmDebugger::Loader->GetInterface ( IID_IConsole, (void**)&SmDebugger::Console);
    Assert ( SmDebugger::Console);
    SmDebugger::Console->printf ("DEBUGGER: Debugger version %d.%d by Tankisto.\n", Version1, Version2);
}

DLL_INIT(loader)
{
    Assert (loader);
    SmDebugger::Loader = loader;

    LinkerDataTypesLibInit ( loader);

    SmDebugger::Loader->GetInterface ( IID_ISmallCompiler, (void**)&SmDebugger::Compiler);
    Assert ( SmDebugger::Compiler);

    SmDebugger::Loader->GetInterface ( IID_ILinker, (void**)&SmDebugger::Linker);
    Assert ( SmDebugger::Linker);
    SmDebugger::GS = & SmDebugger::Linker->getGlobalScope();

    SmDebugger::Loader->GetInterface ( IID_ISmallVM, (void**)&SmDebugger::SmVM);
    Assert ( SmDebugger::SmVM);

    SmDebugger::Loader->GetInterface ( IID_IEditor, (void**)&SmDebugger::Editor);
    Assert ( SmDebugger::Editor);

    SmDebugger::Loader->GetInterface ( IID_ISmallDebugger, (void**) &SmDebugger::Debugger);

    SmDebugger::Console->printf ( "SmallDebugger initialized.\n");

}

#pragma warning (default: 4275) // non dll-interface class used as base for dll-interface class
```

Файл SmallDebugger\system.cpp

```
#include "system.h"
```

```
ILoader*        SmDebugger::Loader = 0;
ILinker*        SmDebugger::Linker = 0;
ISmallVM*       SmDebugger::SmVM = 0;
ISmallCompiler* SmDebugger::Compiler = 0;
IConsole*       SmDebugger::Console = 0;
ISmallDebugger* SmDebugger::Debugger = 0;
IEditor*        SmDebugger::Editor = 0;
CScope*         SmDebugger::GS = 0;
```

Файл SmalltalkEditor\Commands.cpp

```
// Commands.cpp - implementation of Commands.h, of module SmalltalkEditor
// created 29.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "SmalltalkEditor"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Commands.h"
#include "ThreadGate.h"
#include <Exeption.h>      // use ExeptionEngine

void ShowEditor (IConsole* c, int, void*)
{
    c->printf ("Starting Smalltalk Editor..\n");
    EG::Editor->ShowEditorWindow ();
}
//-----------------------------------------------------------------------------
void TestThreadGate (IConsole*c, int, void*)
{
    //(EG::Editor->GetThreadGate())->StoreCall(MessageBox, )
    IDelayedExecutor* ex = EG::Editor->GetThreadGate ();

    ex->StoreCall ((void*)MessageBox, 0, 1, 0, 0, 4, 0, "This is worrking", "Some title", 0);
    c->printf ("Call stored\n");
}

//-----------------------------------------------------------------------------

void AddCommands (IConsole* console)
{
    console->AddCommand ("editor", (CommandProcP)ShowEditor, false, "Editor");

    console->AddCommand ("testgate", (CommandProcP)TestThreadGate, false, "Editor");
}
```

Файл SmalltalkEditor\EditorCore.cpp

```
// EditorCore.cpp - implementation of EditorCore.h, of module SmalltalkEditor
// created 29.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "SmalltalkEditor"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "..\SmallDebugger\Debugger.h"
#include "EditorCore.h"
#include "ThreadGateCore.h"
#include <Exeption.h>      // use ExeptionEngine

IMPLEMENT_IUNKNOWN (CEditorInterface, IEditor, IID_IEditor);

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------

CEditorInterface::CEditorInterface () :
    myLocalsList ( 10)
        {}
//-----------------------------------------------------------------------------
CEditorInterface::~CEditorInterface ()
{
    Terminate ();
//  delete m_ThreadGate;
//  m_ThreadGate = 0;
}
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//IEditor:

void CEditorInterface::ShowEditorWindow ()
{
    m_Editor->ShowWindow ();
}
```

```
//-----------------------------------------------------------------------------

void __cdecl SetCursorCallback (IBCEditor* ed, char*c, int i)
{
    ed->SetCursor (c, i, 0);
}

void CEditorInterface::setCursorTo ( SymbolPosition position)
    {
        //EG::Console->printf ( "CEditorInterface::setCursorTo called");
        //m_Editor->SetCursor ( EG::Linker->getName ( position.myFileName) .c_str(), position.myPosition);
        m_ThreadGate.StoreCall ( m_ThreadGate.ConvertToVoidPtr (0, m_Editor->SetCursor), 0, 0, 0,
m_Editor, 4, m_Editor, EG::Linker->getName ( position.myFileName) .c_str(), position.myPosition, 0);
    }

//-----------------------------------------------------------------------------

void CEditorInterface::insertTextTo ( const char* text, SymbolPosition position)
    {
        EG::Console->printf ( "CEditorInterface::insertTextTo called");
    }

//-----------------------------------------------------------------------------

void CEditorInterface::output ( const char* text)
    {
        EG::Console->printf ( "CEditorInterface::output : '%s'\n", text);
    }

//-----------------------------------------------------------------------------

//void CEditorInterface::task ( MessageData message)

//-----------------------------------------------------------------------------

void CEditorInterface::systemStarted()
    {
        EG::Console->printf ( "CEditorInterface::systemStarted called\n");
    }

//-----------------------------------------------------------------------------

void CEditorInterface::stoppedAt ( int fileName, int position, int breakpointIndex)
    {
        #ifdef __FULLDEBUG
        EG::Console->printf ( "CEditorInterface::stoppedAt called ( file = \"%s\", pos = %d, breakpoint# =
%d.\n",
                              fileName, position, breakpointIndex);
        #endif
        SymbolPosition p;
        p.myFileName = fileName;
        p.myPosition = position;
        setCursorTo ( p);
    }

//-----------------------------------------------------------------------------

void CEditorInterface::systemStopped()
    {
        EG::Console->printf ( "CEditorInterface::systemStopped called");
    }

//-----------------------------------------------------------------------------

void CEditorInterface::breakpointIsSet ( int breakpointIndex)
    {
        EG::Console->printf ( "CEditorInterface::breakpointIsSet %d called", breakpointIndex);
    }

//-----------------------------------------------------------------------------

void CEditorInterface::breakpointIsRemoved ( const BreakpointPosition& breakpoint)
    {
        EG::Console->printf ( "CEditorInterface::breakpointIsRemoved");
    }

//-----------------------------------------------------------------------------

void CEditorInterface::clearLocals()
    {
        myLocalsList = CCont < APDebugTree > ( 10);
        //clearLocalsScreen();
    }

//-----------------------------------------------------------------------------

void CEditorInterface::updateLocals()
    {
        clearLocals();
        myLocalsList.add ( APDebugTree ( EG::SmVM->getWorkTree(), 0));
        clickLocals ( 0);
        redrawLocals();
    }

//-----------------------------------------------------------------------------

void CEditorInterface::clickLocals ( int position)
```

```
    {
        if ( position < 0 || position >= myLocalsList.rNumber())
            return;

        if ( myLocalsList[position]->number() == 0 || position == 0) {
            if ( position != 0) {
                myLocalsList[position]->expand();
                }
            myLocalsList.incTo ( myLocalsList.rNumber() + myLocalsList[position]->number() + 1);

            APDebugTree& tree = myLocalsList[position];
            for ( int i = 0 ; i < tree->number(); i++)
                myLocalsList.insert (
                    position + 1 + i,
                    APDebugTree ( tree->operator [] ( i), tree.myDepth + 1)
                    );
            }
        else //myLocalsList[position].tree->number() == 0
            {
                int n = myLocalsList[position]->number();
                myLocalsList[position]->unexpand();
                for ( int i = n; i > 0; i--) {
                    //delete ( myLocalsList[position + i].tree.release() );
                    myLocalsList[position + i].tree.release();//!!! bad bad bad
                    myLocalsList.del ( position + i);
                    }

            }
        return;
    }

//-----------------------------------------------------------------------------

void CEditorInterface::redrawLocals()
    {
        m_Editor->RedrawLocals();
    }

//-----------------------------------------------------------------------------

void CEditorInterface::Init ()
{
    m_EditorModule = LoadLibrary ("Editor.dll");
    Assert (m_EditorModule);
    GetInstance proc = (GetInstance)GetProcAddress (m_EditorModule, "_GetInstance");
    Assert (proc);
    proc ((void**)&m_Editor);
    m_Editor->Init (static_cast <ISystem*>(this));
}
//-----------------------------------------------------------------------------

void CEditorInterface::Terminate ()
{

    if (m_Editor)
        m_Editor->Terminate ();
    if (m_EditorModule)
        FreeLibrary (m_EditorModule);

    m_EditorModule = NULL;
    m_Editor = NULL;
}


//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------

void __cdecl CEditorInterface::run()
    {
        EG::Debugger->myCommand = ISmallDebugger::eCommands::eRun;
    }

void __cdecl CEditorInterface::stop()
    {
        EG::Debugger->myCommand = ISmallDebugger::eCommands::eStop;
    }

void __cdecl CEditorInterface::pause()
    {
        EG::Debugger->myCommand = ISmallDebugger::eCommands::ePause;
    }

void __cdecl CEditorInterface::step()
    {
        EG::Debugger->myCommand = ISmallDebugger::eCommands::eStep;
    }

void __cdecl CEditorInterface::stepIn()
    {
        EG::Debugger->myCommand = ISmallDebugger::eCommands::eStepIn;
    }

void __cdecl CEditorInterface::stepOut()
    {
        EG::Debugger->myCommand = ISmallDebugger::eCommands::eStepOut;
    }
```

```
//---------------------------------------------------------------------------

int __cdecl CEditorInterface::printf(char* f, ...)
{
    va_list list;
    va_start (list, f);
    int r = EG::Console->vprintf (f, list);
    va_end (list);
    return r;
}
//---------------------------------------------------------------------------

void __cdecl CEditorInterface::delCurBreakpoint()
{
    EG::Debugger->delBreakpoint ( EG::Debugger->getCurBreakpointIndex());
}

//---------------------------------------------------------------------------

void __cdecl CEditorInterface::setBreakpoint ( const char* fileName, int stPos, int endPos)
    {
        BreakpointPosition bp;
        bp.myFileName = EG::Linker->registerName ( fileName);
        bp.myFirstPos = stPos;
        bp.myLastPos = endPos;
        EG::Debugger->setBreakpoint (bp);
    }
//---------------------------------------------------------------------------

int  __cdecl CEditorInterface::getLocalsListNumber()
    {
        return myLocalsList.rNumber();
    }
//---------------------------------------------------------------------------

void __cdecl CEditorInterface::clickLocalsAt ( int index)
    {
        clickLocals ( index);
    }
//---------------------------------------------------------------------------

void __cdecl CEditorInterface::getLocalsAt ( int index, int* depth, const char** name, const char** value)
    {
        if ( depth)
            *depth = myLocalsList[index].myDepth;
        if ( name)
            *name = myLocalsList[index].tree->rName();
        if ( value)
            *value = myLocalsList[index].tree->asString();
    }
//---------------------------------------------------------------------------
bool ThreadGateErrorShown = false;
IDelayedExecutor* __cdecl CEditorInterface::GetThreadGate (void)
{
    /*if (!m_ThreadGate)
        if (!ThreadGateErrorShown)
            {
            ThreadGateErrorShown = true;
            TraceEx ("Warning: Function \"" __FUNCDNAME__ "\": Returning NULL, there's no thread gate!");
            }*/
    return &m_ThreadGate;
}

void __cdecl CEditorInterface::CompileFile (char* filename, char* shortfilename)
{
    try {EG::Compiler->compileFromFile (filename, NULL);}
        Catch ()
            try {EG::Compiler->compileFromFile (shortfilename, NULL);}
            Catch ()
            MessageBox (NULL, "Compiler cannot compile file", "Error", MB_OK | MB_ICONSTOP);
            }
            catch (...){}
        }
    catch (...){}
}

//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

#include <string>

Breakpoint::Breakpoint()
    {
        myPosition.myFileName = 0;
        myPosition.myFirstPos = -1;
        myPosition.myLastPos = -2;
    }

Breakpoint::Breakpoint ( const BreakpointPosition &position) :
    myPosition ( position)
    {
    }

bool Breakpoint::covers ( const BreakpointPosition& position)
    {
        return myPosition.myFileName == position.myFileName &&
```

```
                myPosition.myFirstPos <= myPosition.myLastPos &&
                myPosition.myFirstPos <= position.myFirstPos &&
                myPosition.myLastPos >= position.myLastPos;
    }

bool Breakpoint::covers ( const SymbolPosition& position)
    {
        return myPosition.myFileName == position.myFileName &&
                myPosition.myFirstPos <= position.myPosition &&
                myPosition.myLastPos >= position.myPosition;
    }

bool Breakpoint::operator == ( const Breakpoint& breakpoint)
    {
        return myPosition.myFileName == breakpoint.rPosition().myFileName &&
                myPosition.myFirstPos == breakpoint.rPosition().myFirstPos &&
                myPosition.myLastPos  == breakpoint.rPosition().myLastPos;
    }

//-------------------------------------------------------------------------------------------------

APDebugTree::APDebugTree ( std::auto_ptr < IDebugObjectsTree > other, int depth) :
    tree ( other),
    myDepth ( depth)
    {
    }

APDebugTree& APDebugTree::operator = ( const APDebugTree& apTree)
    {
        //tree = std::auto_ptr < IDebugObjectsTree > ( * const_cast < std::auto_ptr < IDebugObjectsTree >
* > ( & apTree.tree));
        tree = * ( const_cast < std::auto_ptr < IDebugObjectsTree > * > ( & ( apTree.tree)));

        myDepth = apTree.myDepth;
        return *this;
    }

bool APDebugTree::operator== ( const APDebugTree& apTree)
    {
        return tree.get() == apTree.tree.get();
    }

//-------------------------------------------------------------------------------------------------
```

Файл SmalltalkEditor\Module.cpp

```
// Module.cpp - implementation of Module.h, of module SmalltalkEditor
// created 29.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "SmalltalkEditor"

#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine
#include "EditorCore.h"
#include "commands.h"

    #define DLL_CLASS CEditorInterface
    #define DLL_NAME "SmallTalk Editor"
    #define DLL_PROGRAMMER "Vasfed"
    #include "..\Common\Utilities\Module.c"

#pragma warning (disable:4297)

DLL_ONLOAD (loader)
{
    Assert (loader);
    EG::Loader = loader;
    EG::Loader->GetInterface (IID_IConsole, (void**)&EG::Console);
    Assert (EG::Console);
    EG::Console->printf ("EDITOR: SmallTalk IDE by Vasfed&Se version 1.0\n");
}

DLL_INIT (loader)
{

    Assert (loader);
    EG::Loader = loader;


    AddCommands (EG::Console);
    EG::Loader->GetInterface (IID_IEditor, (void**)&EG::Editor);
    Assert (EG::Editor);
    EG::Editor->Init ();

    LinkerDataTypesLibInit ( loader);

    EG::Loader->GetInterface (IID_ISmallCompiler, (void**)&EG::Compiler);
    Assert (EG::Compiler);
```

```
    EG::Loader->GetInterface (IID_ILinker, (void**)&EG::Linker);
    Assert (EG::Linker);
    EG::GS = & EG::Linker->getGlobalScope();

    EG::Loader->GetInterface (IID_ISmallVM, (void**)&EG::SmVM);
    Assert (EG::SmVM);

//  EG::Loader->GetInterface (IID_IConsole, (void**)&EG::Console);
    Assert (EG::Console);


    EG::Loader->GetInterface (IID_ISmallDebugger, (void**) &EG::Debugger);


}

DLL_ONTERMINATE()
{
    DLL_OBJECT.Terminate ();
    Trace ("EDITOR: Terminated successfully");
}
```

Файл SmalltalkEditor\System.cpp

```
#define __PROGRAMMER "Vasfed&Se"
#define __MODULE_NAME "Smalltalk Editor"

#include "system.h"

ILoader*        EG::Loader = 0;
ILinker*        EG::Linker = 0;
ISmallVM*       EG::SmVM = 0;
ISmallCompiler* EG::Compiler = 0;
IConsole*       EG::Console = 0;
ISmallDebugger* EG::Debugger = 0;
IEditor*        EG::Editor = 0;
CScope*         EG::GS = 0;
```

Файл SmalltalkEditor\Test.cpp

```
// Test.cpp - implementation of Test.h, of module SmalltalkEditor
// created 01.12.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "SmalltalkEditor"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include <Exeption.h>      // use ExeptionEngine
#include <conio.h>
#include <stdio.h>

//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------
//ThreadGate Test
#include "ThreadGateCore.h"
#include <process.h>

CThreadGate Gate;

void Thread (void*)
{
    char str[128];
    sprintf (str, "Thread2: started. Setting up messge listener..\n");
    MessageBox (NULL, str, "Thread2", MB_OK);

    while (true)
        {
        //Gate.ExecuteAll ();
        if (Gate.PerformCall (NULL, NULL))
            MessageBox (NULL, "Somethig was executed", "Thread2", MB_OK);
        Sleep (500);
        }
}

void Func (int p1, int p2)
{
    char str[128];
    sprintf (str, "called Func (%d, %d)\n", p1, p2);
    MessageBox (NULL, str, "Func called", MB_OK);
}

class CClass
{
    public:
```

```
    int a;
    CClass(){a = 2;}

    void Func ()
    {
    char str[128];
    sprintf (str, "called Class::Func (): a =%d\n", a);
    MessageBox (NULL, str, "Class::Func called", MB_OK);
    }
};

void Func2 (int p1, char* p2, double p3)
{
    char str[128];
    sprintf (str, "called Func2 (%d, %s, %f)\n", p1, p2, p3);
    MessageBox (NULL, str, "Func2 called", MB_OK);
}

extern "C" void __declspec (dllexport) ThreadGateTest (void) throw ()
{
    char str[128];
    sprintf (str, "\nThreadGate Test Started.\n");
    MessageBox (NULL, str, "Main Thread", MB_OK);

    _beginthread (Thread, NULL, NULL);

    sprintf (str, "Sending commands: Func (2, 3), Func2 (1, \"String\", (double)1.2), a->Func (void)\n");
    MessageBox (NULL, str, "Main Thread", MB_OK);

    CClass a;

    Gate.StoreCall ((void*)Func, 0, 0, 0, 0, 2, 2, 3);
    Gate.StoreCall ((void*)Func2, 0, 0, 0, 0, 4, 1, "String", 1.2);
    Gate.StoreCall (Gate.ConvertToVoidPtr (0, a.Func), 0, 0, 0, &a, 1, &a);
    sprintf (str, "Command stored.\n");
    MessageBox (NULL, str, "Main Thread", MB_OK);

    sprintf (str, "Good bye!");
    MessageBox (NULL, str, "Main Thread", MB_OK);
    exit (0);
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
```

Файл SmalltalkEditor\ThreadGateCore.cpp

```
// ThreadGateCore.cpp - implementation of ThreadGateCore.h, of module SmalltalkEditor
// created 01.12.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "SmalltalkEditor::ThreadGate"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include <Exeption.h>      // use ExeptionEngine
#include <mmsystem.h>
#include <stdarg.h>
#include "ThreadGateCore.h"

CThreadGate::CThreadGate (int startsize, int maxSize)
{
    m_data = NULL;
    m_dataGetP = 0;
    m_dataPutP = 0;
    m_dataSize = 0;
    m_maxSize = maxSize;
    m_blockInput = false;
    m_blockOutput = false;
    m_numCalls = 0;
    GrowTo (startsize);
}
//---------------------------------------------------------------------------

CThreadGate::~CThreadGate ()
{
    //TODO: Normal cleanup with notifications.
    if (m_data)
        delete m_data;
}
//---------------------------------------------------------------------------

bool __cdecl CThreadGate::ReserveMemory    (int numCalls)
{
    return GrowFor (numCalls);
}
//---------------------------------------------------------------------------

bool __cdecl CThreadGate::StoreCallEx  (SDelayedCommand)
{
```

```
    throw;
}
//---------------------------------------------------------------------------

bool __cdecl CThreadGate::StoreCallS   (void* proc, int numParam, ...)
{
    throw;
}
//---------------------------------------------------------------------------

bool __cdecl CThreadGate::StoreSimpleCall(IDelayedExecutor::VOIDPROC proc)
{
    return StoreCall (proc, 0, 0, 0, 0, 0);
}
//---------------------------------------------------------------------------

bool  __cdecl CThreadGate::StoreCall      (void* proc, int rt, int procName, int timeToLive, void*
EcxReg,
                                    int numParam, ...)
{
    SDelayedCommand cmd;
    ZeroMemory (&cmd, sizeof (cmd));

    cmd.callingConversion = procName;//cmd.E_CDECL;
    cmd.numparams = numParam;
    cmd.params = new int[numParam];
    if (!cmd.params) return false;
    cmd.proc = proc;
    cmd.EcxReg = EcxReg;
    cmd.returnType = rt;
    cmd.timeCreated = timeGetTime ();
    cmd.timeToLive = timeToLive;

    va_list params;
    va_start (params, numParam);

    try
        {
        //Argument passing:
        //cdecl - upside down order, stack cleaned by caller
        //stdcall - cdecl, but callee cleans stack
        //pascal - <cesored>, forward order (fixed number of args), callee cleans

        if (cmd.callingConversion == cmd.E_CDECL || cmd.callingConversion == cmd.E_STDCALL)
            for (int i = numParam - 1; i >= 0; i--)
                cmd.params [i] = va_arg (params, int);
        else if (cmd.callingConversion == cmd.E_PASCAL)
            for (int i = 0; i < numParam; i++)
                cmd.params [i] = va_arg (params, int);
            else Throw ("Unknown calling convertion"));
        }
    catch (...)
        {
        Throw ("Wrong number of arguments"));
        }

    va_end (params);

    Push (cmd);
    return true;
}
//---------------------------------------------------------------------------

int __cdecl CThreadGate::PerformCall   (/*out*/ int* procName, SResult* returnedValue)
{
    //if (m_blockOutput) return false;

    m_blockOutput = true;
    //m_blockInput = true;
    if (reinterpret_cast<void*> (this) == (void*)procName) procName = NULL;

    SDelayedCommand cmd;
    if (!Pop (&cmd))
        {
        m_blockOutput = false;
        m_blockInput = false;

        return false;
        }

    SResult rv;
    ZeroMemory (&rv, sizeof (rv));

    try
        {
        void* proc = cmd.proc;
        int numParam = cmd.numparams;
        int* params = cmd.params;

        if (!params && numParam) return false;
        int i;
        int value;
        int res;
        int stacksize;
        void* EcxReg = cmd.EcxReg;
        stacksize = cmd.callingConversion == cmd.E_CDECL? numParam * 4: 0;
```

```
        //Return value:
        //less or equal 32 bits - EAX
        //64 bits               - EDX:EAX
        //more than 64 bits     - *EAX (hidden pointer) How to cleanup?

        void* end = params + numParam;

        __asm {
            mov ecx, EcxReg
            mov ebx, params
            mov eax, end
            }
        PushCircle:
        __asm {
            mov edx, [ebx]
            push edx
            add ebx, 4
            cmp ebx, eax
            jl PushCircle

            call proc
            mov res, eax
            mov eax, stacksize
            add esp, eax
            }

        rv.data.i = res;
        rv.type = cmd.returnType;
        }
    catch (...)
        {
        m_blockOutput = false;
        m_blockInput = false;

        return false;
        }


    if (cmd.params) delete cmd.params;

    if (procName)
        *procName = cmd.procName;
    if (returnedValue)
        *returnedValue = rv;

    m_blockOutput = false;
    m_blockInput = false;

    return true;
}
//---------------------------------------------------------------------------

int __cdecl CThreadGate::ExecuteAll    (bool stopOnError)
{
    if (m_numCalls) return 0;
    while (m_numCalls)
        PerformCall (NULL, NULL);

    return true;
}
//---------------------------------------------------------------------------

int   __cdecl CThreadGate::GetNumCalls (void)
{
    return m_numCalls;
}
//---------------------------------------------------------------------------

SDelayedCommand*  __cdecl CThreadGate::Command (int i)
{
    throw;
}
//---------------------------------------------------------------------------

int   __cdecl CThreadGate::DeleteCommand   (int i)
{
    throw;
}
//---------------------------------------------------------------------------

bool  CThreadGate::Push    (SDelayedCommand c) //FIFO
{
    if (m_numCalls >= m_dataSize - 2)
        return false;

    m_numCalls++;

    m_data[m_dataPutP] = c;
    m_dataPutP++;

    if (m_dataPutP >= m_dataSize)
        m_dataPutP = 0;

    return true;
}
//---------------------------------------------------------------------------
```

```
bool  CThreadGate::Pop     (SDelayedCommand* r)
{
    if (!m_numCalls)
        return false;
    if (!r)
        return false;

    m_numCalls--;
    *r = m_data [m_dataGetP];

    m_dataGetP++;
    if (m_dataGetP >= m_dataSize)
        m_dataGetP = 0;
    return true;
}
//---------------------------------------------------------------------------

bool   CThreadGate::Grow       (int type)
{
    return GrowFor (m_dataSize? m_dataSize/2: 10);
}
//---------------------------------------------------------------------------

bool   CThreadGate::GrowFor    (int n)
{
    return GrowTo (m_dataSize + n);
}
//---------------------------------------------------------------------------

bool   CThreadGate::GrowTo (int n)
{
    if (n < m_dataSize)
        return true; //it is not critical error, we may show off that we've allocated memory

    SDelayedCommand* temp = new SDelayedCommand [n];

    if (!temp) return false;

    if (m_data) //we've to copy old data
        {
        int i = 0;
        for (int p = m_dataGetP; i < m_numCalls; p++, i++)
            {
            if (p >= m_dataSize)
                p = 0;

            temp[i] = m_data [p];
            }

        m_dataGetP = 0;
        m_dataPutP = i;
        delete m_data;
        }

    m_data = temp;
    m_dataSize = n;
    return true;
}
//---------------------------------------------------------------------------
void  __cdecl CThreadGate::BlockInput (bool value)
    {m_blockInput = value;}

void  __cdecl CThreadGate::BlockOutput (bool value)
    {m_blockOutput = value;}

void*  __cdecl CThreadGate::ConvertToVoidPtr (int f, ...)
{
    va_list l;
    va_start (l, f);
    return va_arg (l, void*);
    va_end (l);
}
```

Файл SmalltalkEditor\Builder\About.cpp

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "About.h"
//---------------------------------------------------------------------
#pragma resource "*.dfm"
TAboutBox *AboutBox;
//---------------------------------------------------------------------
__fastcall TAboutBox::TAboutBox(TComponent* AOwner)
    : TForm(AOwner)
{
}
//---------------------------------------------------------------------
```

Файл SmalltalkEditor\Builder\DebugForm.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "DebugForm.h"
#include "EditorInt.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TDebugFrm *DebugFrm;
//---------------------------------------------------------------------------
__fastcall TDebugFrm::TDebugFrm(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TDebugFrm::ListDblClick(TObject *Sender)
{
    int index = List->ItemIndex;
    if (index < 0) return;

    Syst->clickLocalsAt ( ++index);

    Redraw();

}
//---------------------------------------------------------------------------

void __fastcall TDebugFrm::FormShow(TObject *Sender)
{
    if (Syst)
        Syst->updateLocals();
    else
        List->AddItem ("No debugger found in system", NULL);
    Redraw();
    FormResize (this);
}

void __fastcall TDebugFrm::Redraw()
{
    int n = Syst->getLocalsListNumber();
    List->Clear();
    if (!n)
        {
        List->AddItem ("No variables", NULL);
        return;
        }

    for ( int i = 1; i < n; i++) {
        const char* name = 0;
        const char* value = 0;
        int tabs = 0;
        Syst->getLocalsAt ( i, & tabs, &name, &value);
        AnsiString s = "";
        for ( int i = 0; i < tabs; i++)
            s += "    ";
        s += name;
        s += " = ";
        s += value;
        List->AddItem(s, NULL);
        }

}
//---------------------------------------------------------------------------

void __fastcall TDebugFrm::FormClose(TObject *Sender, TCloseAction &Action)
{
    //
    delete this;
}
//---------------------------------------------------------------------------

void __fastcall TDebugFrm::FormResize(TObject *Sender)
{
    List->Height = ClientHeight - 10;
    List->Width = ClientWidth - 10;
}
//---------------------------------------------------------------------------
```

Файл SmalltalkEditor\Builder\DocumentForm.cpp

```
//---------------------------------------------------------------------------
```

```
#include <vcl.h>
#pragma hdrstop

#include "DocumentForm.h"
#include "Globals.h"
#include "stdio.h"
#include "MainFrm.h"
#include "SearchReplace.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TDocument *Document;
//---------------------------------------------------------------------------
__fastcall TDocument::TDocument(TComponent* Owner)
        : TForm(Owner)
{
}

__fastcall TDocument::~TDocument ()
{
}
//---------------------------------------------------------------------------
void __fastcall TDocument::FormResize(TObject *Sender)
{
        Edit->Width = this->ClientWidth;
        Edit->Height = this->ClientHeight;
}
//---------------------------------------------------------------------------
void __fastcall TDocument::FormCreate(TObject *Sender)
{
        Edit->Width = this->ClientWidth;
        Edit->Height = this->ClientHeight;
        Edit->Text = "Smalltalk interchangeVersion: '1.0' !\n\n";
        Edit->SelStart = Edit->Text.Length();
        Edit->SelLength = 0;
}
//---------------------------------------------------------------------------

void TDocument::SaveClick()
{
    if (m_Filename == "")
        return SaveAsClick ();

    FILE* file = fopen (m_Filename.c_str(), "wt");
    fwrite (Edit->Text.c_str(), Edit->Text.Length(), 1, file);
    fclose (file);

    if (ColorizeKeysEnabled) ColorizeText ();
}

void TDocument::SaveAsClick()
{
    if (m_Filename != "")
        {
        SaveDialog->FileName = m_Filename;
        SaveDialog->Title = "Save " + m_Filename + " as..";
        }
    else
        {
        SaveDialog->FileName = "";
        SaveDialog->Title = "Save new SmallTalk File...";
        }
    if (SaveDialog->Execute ())
        {
        m_Filename = SaveDialog->FileName;
        FILE* file = fopen (m_Filename.c_str(), "wt");
        fwrite (Edit->Text.c_str(), Edit->Text.Length(), 1, file);
        fclose (file);
        }
    else return;

    int i;
    for (i = m_Filename.Length(); i > 0; i--)
        if (m_Filename[i] == '\\')
            break;
    Caption = (m_Filename.c_str() + i);

    if (ColorizeKeysEnabled) ColorizeText ();
}

void TDocument::CloseClick()
{
    Close ();
    //delete this;
}

void TDocument::OpenFile(AnsiString filename)
{
    FILE* file = fopen (filename.c_str(), "rt");
    if (!file) return;

    Edit->Visible = false;
    Edit->Text = "";
    Edit->SelLength = 0;
    Edit->SelStart = 1;
    Edit->SelAttributes->Color = MakeColor (Default);
```

```
    char str[2048];
    while (fgets (str, 2048, file))
        {
        if (str[strlen (str) - 1] < 13)
            str[strlen (str) - 1] = '\0';

        Edit->Lines->Append (str);
        }

    fclose (file);

    m_Filename = filename;
    int i;
    for (i = m_Filename.Length(); i > 0; i--)
        if (m_Filename[i] == '\\')
            break;
    Caption = (filename.c_str() + i);

    if (ColorizeKeysEnabled) ColorizeText ();
    else Edit->Visible = true;

    Edit->SelStart = strlen ("Smalltalk interchangeVersion: '1.0' !") + 2;
    Edit->SelLength = 0;
    if (!Edit->Visible)
        Edit->Visible = true;
}
//---------------------------------------------------------
//#region Text Colorizer

bool EatLexem (TRichEdit* Edit, bool eat, int* ii, int n)
{
    bool s = false;
    for (int i = *ii; i < Edit->Text.Length () - 1; i++)
        {
        switch (Edit->Text[i])
            {
            case '!': return false;
            case ':':
                Edit->SelStart = *ii;
                Edit->SelLength = i - *ii + 1;
                Edit->SelAttributes->Color = MakeColor (MethodDef);
               *ii = i + 1;

                return true;
                break;
            case ' ': case '\n': case '\t': case '\r':
                if (!s) continue;
                if (eat)
                    {
                     Edit->SelStart = *ii;
                     Edit->SelLength = i - *ii + 1;
                    Edit->SelAttributes->Color = n? MakeColor (MethodDefParam): MakeColor (MethodDef);
                     *ii = i;
                     }
                return false;
               break;
            default: s = true; break;
            }
        }
    return false;
}

int EatMethodDef (TRichEdit* Edit, int* ii)
{
    if (EatLexem (Edit, true, ii, 0))
        {
        EatLexem (Edit, true, ii, 1);

        while (EatLexem (Edit, false, ii, 1))
            EatLexem (Edit, true, ii, 1);
        }
}

int EatVariable (TRichEdit* Edit, int* ii)
{
    bool s = false;
    for (int i = *ii; i < Edit->Text.Length (); i++)
        {
        switch (Edit->Text[i])
            {
            case '|':
                Edit->SelStart = *ii;
                Edit->SelLength = i - *ii + 1;
                Edit->SelAttributes->Color = MakeColor (TemporaryVariablesSymbol);
               *ii = i + 1;

                return false;
                break;
            case ' ': case '\n': case '\t': case '\r':
                if (!s) continue;

                 Edit->SelStart = *ii;
                 Edit->SelLength = i - *ii + 1;
                 Edit->SelAttributes->Color = MakeColor (TemporaryVariablesSymbol);
                 *ii = i + 1;
                 return true;
               break;
```

```
            default: s = true; break;
            }
        }

    return false;
}

int EatVariableList (TRichEdit* Edit, int* ii)
{
    while (EatVariable (Edit, ii));
}

void TDocument::ColorizeText()
{
    int oldss = Edit->SelStart;
    int oldsl = Edit->SelLength;
    ColorNames color = Default;
    int blocksCounter = 0;
    int bracketsCounter = 0;
    bool vzt = true;
    bool tvs = true;
    bool symbol = false;
    int expecting = Object;

    int tl = Edit->Text.Length();
    int l = 1;
    int v1, v2;
    if (sscanf (Edit->Text.c_str(), "Smalltalk interchangeVersion: '%d.%d' !%n", &v1, &v2, &l) < 2)
        if (MessageBox (NULL, "Text inside editor is not smalltalk source code.\nDo you want to try to
colorize it?\nThis may cause system freeze.", "?", MB_YESNO | MB_ICONQUESTION) == IDNO)
            return;

    Edit->HideSelection = true;
    this->Color = MakeColor (BackColor);
    Edit->Visible = false;
    Edit->Color = MakeColor (BackColor);

    Edit->SelStart = 0;
    Edit->SelLength = l;
    Edit->SelAttributes->Color = MakeColor (FileHeader);

    for (int i = l + 1; i < tl; i++)
        {
        switch (Edit->Text[i])
            {
            case '"': //we have Comment
                {
                int start = i;
                i++;
                for (; i < tl; i++)
                    if (Edit->Text[i] == '"') break;
                    if (i >= tl)
                        {
/*                      Edit.SelectionStart = i - 1;
                        Edit.SelectionLength = 0;
                        Edit.SelectionColor = MakeColor (Error);
                        Edit.SelectedText = "Expected: end of comment";*/
                        }

                    Edit->SelStart = start - 1;
                    Edit->SelLength = i - start + 1;
                    Edit->SelAttributes->Color = MakeColor (Comment);
                    //i++;
                    //error
                    }
                     continue;
                    break;
                case '\'': //we have String
                {
                    int start = i;
                    i++;
                    for (; i < tl; i++)
                        if (Edit->Text[i] == '\'') break;
                                if (i >= tl)
                        {/*Unfinished string*/}
                     Edit->SelStart = start - 1;
                    Edit->SelLength = i - start + 1;
                    //i++;
                    Edit->SelAttributes->Color = MakeColor (Str);
                }
                 expecting = Method;
                 continue;
                break;
                case ' ': case '\t': case '\n': case '\r':
                    if (expecting == Object && symbol == true)
                        {
                        symbol = false;
                        expecting = Method;
                        }

                    break;

                case ':':
                    if (expecting == Method)
                        {
                        color = symbol? Method: Operator;
                        expecting = Object;
```

```
                    symbol = false;
                    }
                else //we have list of block parameters
                    {
                    color = TemporaryVariablesSymbol;
                     {
                     Edit->SelStart = i - 1;
                     Edit->SelLength = 1;
                     Edit->SelAttributes->Color = MakeColor (TemporaryVariablesSymbol);
                     i++;
                    EatVariableList (Edit, &i);
                     }
                     tvs = false;
                    }
                break;

            case '|': //we have temporary variables list
                 {
                 bool cont = tvs;
                 color = TemporaryVariablesSymbol;

                 if (tvs)
                     {
                     Edit->SelStart = i - 1;
                     Edit->SelLength = 1;
                     Edit->SelAttributes->Color = MakeColor (TemporaryVariablesSymbol);
                     i++;
                    EatVariableList (Edit, &i);
                     }

                 tvs = true;
                 expecting = Object;
                 symbol = false;
                 if (cont) continue;
                 }
                break;
             case '#':
                 color = ArraySymbol;
                 expecting = Object;
                 symbol = false;
                 break;
            case '0': case '1': case '2': case '3': case '4': case '5': case '6': case '7': case '8':
case '9':
                color = Number;
                expecting = Method;
                symbol = false;
                break;

            case '%': case '&': case '*': case '+': case ',': case '/': case '<': case '>':
            case '=': case '?': case '@': case '~': case '-': case '\\': case '^':
                //we have operator
                expecting = Object;
                color = Operator;
                symbol = false;
                break;

            case '.': //dot
             if (!blocksCounter)
                 if (bracketsCounter)
                     {
                     Edit->SelLength = 0;
                     Edit->SelStart = i - 1;
                     Edit->SelAttributes->Color = MakeColor (Error);
                     if (bracketsCounter > 0)
                         while (bracketsCounter--) Edit->SelText = ")";
                     }

                expecting = Object;
                symbol = false;
                color = Dot;
                break;

            case '!':
                if (vzt == false)
                    {
                     Edit->SelStart = i - 1;
                     Edit->SelLength = 1;
                     Edit->SelAttributes->Color = MakeColor (VZ);

                    i++;
                     EatMethodDef (Edit, &i);

                     vzt = true;
                     expecting = Object;
                     symbol = false;
                     continue;
                    }
                 else {
                     if (bracketsCounter)
                         {
                         Edit->SelStart = i - 1;
                         Edit->SelLength = 0;
                         Edit->SelAttributes->Color = MakeColor (Error);

                         if (bracketsCounter > 0)
                              {
                                 do
```

```
                                    {
                                    Edit->SelText = ")";
                                    i++;
                                    }
                                    while ((--bracketsCounter));

                                Edit->SelText = "\"Bracket added\"";
                                i += strlen ("\"Bracket added\"");
                                }
                            }
                         if (blocksCounter)
                            {
                            Edit->SelStart = i - 1;
                            Edit->SelLength = 0;
                            Edit->SelAttributes->Color = MakeColor (Error);
                            if (blocksCounter > 0)
                                {
                                do
                                    {
                                    Edit->SelText = "]."; i+=2;
                                    } while ((--blocksCounter));

                                Edit->SelText = "\"Block bracket added\"";
                                i += strlen ("\"Block bracket added\"");
                                }
                            }

                        expecting = Object;
                        vzt = false;
                        symbol = false;
                        color = VZ;
                         }
                    break;

                case '(':
                     expecting = Object; symbol = false;
                    color = Brackets;
                    bracketsCounter++;
                    break;

                case ')': expecting = Method; symbol = false;
                    color = Brackets;
                    if (bracketsCounter-- < 0) color = Error;
                    break;

                case '[': case '{':
                    expecting = Object; symbol = false;
                    color = Block;
                    blocksCounter++;
                    break;
                case ']': case '}':
                    expecting = Method; symbol = false;
                    color = Block;
                    if (blocksCounter-- < 0)
                        color = Error;
                    break;

                case ';': expecting = Method; symbol = false;
                    color = MethodCascadeSymbol;
                    break;

                default:
                    symbol = true;
                    color = expecting;
                    break;
            }

        Edit->SelStart = i - 1;
        Edit->SelLength = 1;
        if (color == None)
            color = expecting; //? Object: Method;
        Edit->SelAttributes->Color = MakeColor (color);
        color = None;
        }

    Edit->SelStart = oldss;
    Edit->SelLength = oldsl;
    Edit->Visible = true;
    Edit->HideSelection = false;
    Edit->SetFocus();
}

//#endregion
//---------------------------------------------------------------
void __fastcall TDocument::FormCloseQuery(TObject *Sender, bool &CanClose)
{
    int r = IDYES;
    if (r = MessageBox (NULL, ("Do you want to save document \"" + Caption + "\"?").c_str(), "Exit",
MB_YESNOCANCEL) == IDYES)
        SaveClick ();
    if (r == IDCANCEL)
        CanClose = false;
}
//---------------------------------------------------------------------------

void __fastcall TDocument::ColorizeText1Click(TObject *Sender)
{
```

```
    ColorizeText();
}
//---------------------------------------------------------------------------

void __fastcall TDocument::FormShortCut(TWMKey &Msg, bool &Handled)
{
    Handled = true;
    if (!MainForm->MainMenu->IsShortCut(Msg))
        if (!(MainMenu->IsShortCut(Msg)))
            Handled = false;

}
//---------------------------------------------------------------------------

void __fastcall TDocument::FormKeyPress(TObject *Sender, char &Key)
{
    if (ColorizeKeysEnabled)
        {
        if (Key == VK_BACK)
             {
               ColorizeText ();
               return;
             }

        for (int i = 0; i < NumColorizeKeys; i++)
            if (ColorizeKeysCh[i] == (char)(Key))
                {
                Edit->SelText = Key;
                Key = NULL;
                ColorizeText ();
                return;
                }
        }
}
//---------------------------------------------------------------------------

void __fastcall TDocument::Addclassdeclaration1Click(TObject *Sender)
{
    Edit->SelText = "\n\"------------------------------------------------------------\"\nClass named: '";
    Edit->SelStart = Edit->SelLength + Edit->SelStart;
    Edit->SelLength = 0;
    int Start = Edit->SelStart;
    Edit->SelText = "ClassName'\n";
    Edit->SelText = "instanceVariableNames: '\"Instance variable names go here\"'\n";
    Edit->SelText = "classVariableNames: '\"Class variable names go here\"'\n";
    Edit->SelText = "\"sharedPools: ''\"\n";
    Edit->SelText = "\" classInstanceVariableNames: <classInstVariableList>\"\n";
    Edit->SelText = "\n\"------------------------------------------------------------\"!\n";
    Edit->SelStart = Start;
    Edit->SelLength = strlen ("ClassName");
}
//---------------------------------------------------------------------------
/*Class named: 'Pimp'
    instanceVariableNames: 'Me m_DestX m_DestY m_ovx m_ovy m_BestH tvx tvy tvz ready'
    classVariableNames: ''
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"*/

void __fastcall TDocument::Addmethoddeclaration1Click(TObject *Sender)
{
    Edit->SelText = "\n\"------------------------------------------------------------\"\n<ClassName> method!\n";
    int Start = Edit->SelStart;
    Edit->SelText = "MethodDeclaration\n";
    Edit->SelText = "\n\n\n";
    Edit->SelText = "\n\"------------------------------------------------------------\"!\n";
    Edit->SelStart = Start;
    Edit->SelLength = strlen ("MethodDeclaration");
}
//---------------------------------------------------------------------------

void __fastcall TDocument::FormClose(TObject *Sender, TCloseAction &Action)
{
    //
    delete this;
}
//---------------------------------------------------------------------------

void __fastcall TDocument::SetFocus()
{
    //TODO: Add your source code here
    this->Edit->SetFocus();
}
void __fastcall TDocument::Undo1Click(TObject *Sender)
{
    Edit->Undo();
}
//---------------------------------------------------------------------------

void __fastcall TDocument::SearchReplace1Click(TObject *Sender)
{
    SearchReplaceForm->Show();
}
//---------------------------------------------------------------------------

void __fastcall TDocument::SearchAgain1Click(TObject *Sender)
{
    SearchReplaceForm->SearchClick (this);
```

```
}
//---------------------------------------------------------------------------

void __fastcall TDocument::Decreaseindent1Click(TObject *Sender)
{
    //
    int start = Edit->SelStart;
    if (start < 1) start = 1;
    int end = start + Edit->SelLength;

    for (int i = start - 1; i < end && i < Edit->Text.Length(); i++)
        {
        if (Edit->Text[i] == '\n')
            if (Edit->Text[++i] == '\t')
                {
                Edit->SelStart = i;
                Edit->SelLength = 2;
                Edit->SelText = Edit->Text[i+1];
                }
        }
}
//---------------------------------------------------------------------------

void __fastcall TDocument::Increaseindent1Click(TObject *Sender)
{
    //
    int start = Edit->SelStart;
    int end = start + Edit->SelLength;

}
//---------------------------------------------------------------------------
```

Файл SmalltalkEditor\Builder\EditorInt.cpp

```
//---------------------------------------------------------------------------
#pragma hdrstop

#include "EditorInt.h"
#include <windows.h>
#include <vcl.h>
#include <process.h>
#include "MainFrm.h"
#include "Options.h"
#include "DocumentForm.h"
#include "DebugForm.h"
#include "Main.h"
#include "SearchReplace.h"
//---------------------------------------------------------------------------

//#pragma package(smart_init)

CEditor::CEditor (){}
CEditor::~CEditor (){}

ISystem* Syst = NULL;

//---------------------------------------------------------------------------

void __cdecl CEditor::Init (ISystem* syst)
{
    Syst = syst;
}

//---------------------------------------------------------------------------

void __cdecl CEditor::OpenFile (char* filename)
{
    //
    if (!MainForm)
        ShowWindow();

    TDocument* d = new TDocument (MainForm);
    d->OpenFile(filename);
}

//---------------------------------------------------------------------------

void __cdecl CEditor::SetCursor ( const char* file, int position, int length)
{
    if (!MainForm)
        {
        ShowWindow ();
        }

    MainForm->SetFocus ();
    MainForm->BringToFront();
    MainForm->GotoPosition(file, position, length, true);
}

//---------------------------------------------------------------------------
```

```
void __cdecl CEditor::RedrawLocals()
{
    if (DebugFrm)
        if (DebugFrm->Visible)
            DebugFrm->OnShow (DebugFrm);
}

//---------------------------------------------------------------------------
bool RunnedFirstTime = true;

void MainLoop (void* s)
{
    try {
        if (!RunnedFirstTime)
            {
            if (Application)
                delete Application;
            Application = new TApplication (NULL);
            }
        RunnedFirstTime = false;
        Application->Initialize ();
        Application->Title = "Smalltalk Development Environment";
        Application->CreateForm (__classid (TMainForm), &MainForm);
        //OptionsDlg = new TOptionsDlg    (MainForm);
        //SearchReplaceForm = new TSearchReplaceForm  (MainForm);
        Application->CreateForm (__classid (TOptionsDlg), &OptionsDlg);
        Application->CreateForm (__classid (TSearchReplaceForm), &SearchReplaceForm);

        Editor.GetCommands();
        MainForm->Show();
        MainForm->BringToFront();
        Application->Run ();
        }
    catch (...)
        {
        MessageBox (NULL, "Error", "Error", MB_OK);
        }

    if (MainForm) delete MainForm;
    MainForm = NULL;

    if (Application)
        delete Application;

    Application = NULL;
}
//---------------------------------------------------------------------------
void __cdecl CEditor::ShowWindow ()
{
    if (!MainForm)
        {
       if (_beginthread (MainLoop, NULL, NULL) <= 0)
            MessageBox (NULL, "Error: Cannot create thread for IDE", "Error", NULL);
        Syst->printf ("EDITOR: Window created.\n");
        Sleep (1000);
        }
    else
        {
        Syst->printf ("EDITOR: Window is already created.\n");
        Application->Restore();
        Application->BringToFront();
        }
}
//---------------------------------------------------------------------------
void __cdecl CEditor::Terminate ()
{
   if (Application)
        {
            Application->Terminate ();
            MainForm = NULL;
            Application = NULL;
            Sleep (100);
        }
}

//---------------------------------------------------------------------------
bool Var = true;

void __cdecl CEditor::GetCommands ()
{
    IDelayedExecutor* cl = Syst->GetThreadGate();
    if (!cl) return;

    //cl->ExecuteAll();
    try {
       cl->PerformCall (NULL, NULL);
 //       cl->ExecuteAll ();
        }
    catch (...)
        {
        if (Var)
            {
            Var = false;
            MessageBox (NULL, "Exeption thrown while executing delayed command", "Error", NULL);
            }
        }
```

```
}
```

Файл SmalltalkEditor\Builder\ErrorMessagesForm.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "ErrorMessagesForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TErrorMessageForm *ErrorMessageForm;
//---------------------------------------------------------------------------
__fastcall TErrorMessageForm::TErrorMessageForm(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TErrorMessageForm::FormShow(TObject *Sender)
{
    List->Height = this->ClientHeight;
    List->Width = this->ClientWidth;

}
//---------------------------------------------------------------------------
void __fastcall TErrorMessageForm::FormResize(TObject *Sender)
{
    List->Height = this->ClientHeight;
    List->Width = this->ClientWidth;
}
//---------------------------------------------------------------------------

void __fastcall TErrorMessageForm::ListDblClick(TObject *Sender)
{
    //
    MessageBox (NULL, "Going to place of message...", "..", MB_OK);
}
//---------------------------------------------------------------------------
```

Файл SmalltalkEditor\Builder\Form.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Form.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

Файл SmalltalkEditor\Builder\Globals.cpp

```
//---------------------------------------------------------------------------

#pragma hdrstop

#include "Globals.h"
#include <windows.h>
//---------------------------------------------------------------------------

#pragma package(smart_init)

int ColorValues[MaxColorName] = {
        0, //None,
        0, //Default,
        RGB (0, 0, 200), //Object,
        RGB (100, 0, 0), //Method,
        RGB (0, 200, 200), //MethodDef,
        RGB (0, 100, 100), //MethodDefParam,
        RGB (100, 100, 100), //FileHeader,
```

```
        RGB (200, 0, 0), //SystemChar,
        RGB (0, 0, 100), //Dot,
        RGB (100, 0, 0), //MethodCascadeSymbol,
        RGB (0, 100, 200), //Operator,
        RGB (200, 0, 0), //VZ,
        RGB (0, 0, 200), //String,
        RGB (0, 0, 200), //Number,
        RGB (0, 255, 0), //Comment,
        RGB (0, 0, 200), //ArraySymbol,
        RGB (0, 150, 0), //Brackets,
        RGB (0, 150, 0), //Block,
        RGB (255, 255, 255), //BackColor,
        RGB (100, 0, 0), //TemporaryVariablesSymbol,
        RGB (255, 0, 0) //Error,
        };

char ColorizeKeysCh[255] = {'.', ';', '(', ')',
                            '[', ']', '\n',
                            '\'', '"', ' ', '\0'};
//".;()[] \t\n'\"";
int NumColorizeKeys = strlen (ColorizeKeysCh);
bool ColorizeKeysEnabled = false;

int MakeColor (ColorNames color)
{
    if (color > 0 && color < MaxColorName)
        return ColorValues [color];
    else return ColorValues [Default];
};

int* MakeColorPointer (ColorNames color)
{
    if (color > 0 && color < MaxColorName)
        return &ColorValues [color];
    else return NULL;
};
```

Файл SmalltalkEditor\Builder\Main.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#include <windows.h>
#include "EditorInt.h"
#pragma hdrstop
//---------------------------------------------------------------------------
#pragma argsused
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fwdreason, LPVOID lpvReserved)
{
    return 1;
}
//---------------------------------------------------------------------------

CEditor Editor;

extern "C" void __declspec (dllexport) GetInstance (void** ppr)
{
    if (ppr)
        *ppr = &Editor;
}
```

Файл SmalltalkEditor\Builder\MainFrm.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "MainFrm.h"
#include "DocumentForm.h"
#include "Options.h"
#include "About.h"
#include "DebugForm.h"
#include "EditorInt.h"
#include "ErrorMessagesForm.h"
#include <typeinfo>
#include "Main.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TMainForm *MainForm = NULL;
//---------------------------------------------------------------------------
__fastcall TMainForm::TMainForm(TComponent* Owner)
    : TForm(Owner)
{
}

__fastcall TMainForm::~TMainForm()
```

```
{
    return;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    Application->OnHint = ShowHint;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::ShowHint(TObject *Sender)
{
    StatusLine->Panels->Items[0]->Text = Application->Hint;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FileOpen(TObject *Sender)
{
    if (OpenDialog->Execute())
    {
        TDocument* d = new TDocument(this);
        d->OpenFile (OpenDialog->FileName);
        d->Show();
        //---- Add code to open OpenDialog->FileName ----
    }
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FileSave(TObject *Sender)
{
    //---- Add code to save current file under current name ----
    ((TDocument*)ActiveMDIChild)->SaveClick ();
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
void __fastcall TMainForm::FileExit(TObject *Sender)
{
    Close();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::EditUndo(TObject *Sender)
{
    //---- Add code to perform Edit Undo ----
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::EditCut(TObject *Sender)
{
    //---- Add code to perform Edit Cut ----
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::EditCopy(TObject *Sender)
{
    //--- Add code to perform Edit Copy ----
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::EditPaste(TObject *Sender)
{
    //---- Add code to perform Edit Paste ----
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::WindowTile(TObject *Sender)
{
    Tile();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::WindowCascade(TObject *Sender)
{
    Cascade();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::WindowArrange(TObject *Sender)
{
    ArrangeIcons();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::HelpContents(TObject *Sender)
{
    Application->HelpCommand(HELP_CONTENTS, 0);
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::HelpSearch(TObject *Sender)
{
    Application->HelpCommand(HELP_PARTIALKEY, Longint(""));
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::HelpHowToUse(TObject *Sender)
{
    Application->HelpCommand(HELP_HELPONHELP, 0);
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::HelpAbout(TObject *Sender)
{
    TAboutBox* d = new TAboutBox(this);
    d->ShowModal();
    delete d;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FileNewItemClick(TObject *Sender)
{
        TDocument* d = new TDocument (this);
```

```
        d->Show ();
        this->MainMenu->AutoHotkeys = false;
        d->MainMenu->AutoHotkeys = true;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Save1Click(TObject *Sender)
{
        ((TDocument*)this->ActiveMDIChild)->SaveClick();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::SaveAs1Click(TObject *Sender)
{
        ((TDocument*)this->ActiveMDIChild)->SaveAsClick();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Close1Click(TObject *Sender)
{
      ActiveMDIChild->Close ();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Preferences1Click(TObject *Sender)
{
    //
    OptionsDlg->Visible = true;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Showlocals1Click(TObject *Sender)
{
    //
//    if (!m_debugFrm)
        m_debugFrm = new TDebugFrm(this);

    m_debugFrm->Show ();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Setbreakpointcall1Click(TObject *Sender)
{
    if ( !ActiveMDIChild) return;
    int st = ((TDocument*)ActiveMDIChild)->Edit->SelStart - 1; //!!!
    int l = ((TDocument*)ActiveMDIChild)->Edit->SelLength + 1;

    if (!ActiveMDIChild->InheritsFrom(__classid (TDocument)))
        return;

    AnsiString f = ((TDocument*)ActiveMDIChild)->m_Filename;
    Syst->setBreakpoint ( f.c_str(), st, st + l);
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Run1Click(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->run();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Compile1Click(TObject *Sender)
{
    if (!ActiveMDIChild)
        return;
    if (!ActiveMDIChild->InheritsFrom(__classid (TDocument)))
        {
        return;
        }

    ((TDocument*)ActiveMDIChild)->SaveClick();
    if (((TDocument*)ActiveMDIChild)->m_Filename == "")
        {
        MessageBox (NULL, "You must save file before compiling it", "Error", MB_OK);
        return;
        }

    AnsiString f = ActiveMDIChild->Caption;
    Syst->CompileFile (((TDocument*)ActiveMDIChild)->m_Filename.c_str(), f.c_str());
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn1Click(TObject *Sender)
{
    //
    if (!m_ErrorsForm) m_ErrorsForm = new TErrorMessageForm (this);
    m_ErrorsForm->Show();
}
//---------------------------------------------------------------------------
TDocument* TMainForm::ActivateDocument (AnsiString filename)
{
    int n = this->MDIChildCount;
    for (int i = 0; i < n; i++)
        if (MDIChildren[i]->InheritsFrom (__classid (TDocument)))
```

```
            {
            if (((TDocument*)MDIChildren[i])->m_Filename == filename
                || ((TDocument*)MDIChildren[i])->Caption == filename)
                {
                //we've found it
                MDIChildren[i]->Show ();
                MDIChildren[i]->BringToFront();
                MDIChildren[i]->SetFocus();
                return (TDocument*)MDIChildren[i];
                }
            }
    return NULL;
}
//---------------------------------------------------------------------------
TDocument* TMainForm::GotoPosition (AnsiString file, int position, int len, bool open, bool thread)
{
//
    TDocument* doc = NULL;

    if (doc = ActivateDocument (file))
        {
        doc->Show();
        doc->SetFocus();
        doc->BringToFront();
        doc->Edit->HideSelection = false;
        doc->Edit->SelStart = position;
        //doc->Edit->SelLength = abs (len) /*+ 1*/;
        doc->Edit->SelLength = 1;
        }
    else
        if (open)
            if (FileExists(file))
              {
              //doc = new TDocument (this);
              Application->CreateForm(__classid (TDocument), &doc);

              doc->Edit->SelAttributes->Color =0;
              doc->Edit->Text = "Opening file..";
              doc->OpenFile (file);
              doc->Show();
              doc->SetFocus();
              doc->BringToFront();
              doc->Edit->HideSelection = false;
              doc->Edit->SelStart = position;
//              doc->Edit->SelLength = abs (len)/* + 1*/;
            doc->Edit->SelLength = 1;
            }
    return doc;
}

void __fastcall TMainForm::FindWindowBtnClick(TObject *Sender)
{
    if (!ActivateDocument (FindWindowNames->Text))
        {
        FindWindowNames->Items->Append(FindWindowNames->Text);
        MessageBox (0, "Window not found", "Window finder", 0);
        }
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::ToolButton1Click(TObject *Sender)
{
    FileNewItemClick (this);
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::ToolButton2Click(TObject *Sender)
{
    this->FileOpen(this);
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::ToolButton3Click(TObject *Sender)
{
    this->FileSave(this);
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::FindWindowNamesKeyPress(TObject *Sender,
      char &Key)
{
    if (Key == 13)
        FindWindowBtnClick (this);
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::GetSystemMessagesTimerTimer(TObject *Sender)
{
    Editor.GetCommands();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::DebugRunClick(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
```

```
    Syst->run();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::DebugPauseClick(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->pause();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::DebugStopClick(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->stop();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::DebugStepIntoClick(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->stepIn();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::StepOverClick(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->step();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::DebugStepOutClick(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->stepOut();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::DelCurBreakpoint(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->delCurBreakpoint();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::Button1Click(TObject *Sender)
{
    this->Showlocals1Click (this);
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::StatusBarTimerTimer(TObject *Sender)
{
    StatusLine->Panels[0].Items[0]->Text = "Ready";
    StatusBarTimer->Enabled = false;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Say (AnsiString s)
{
    StatusLine->Panels[0].Items[0]->Text = s;
    StatusBarTimer->Enabled = true;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::CompileAll1Click(TObject *Sender)
{
    for (int i = 0; i < MDIChildCount; i++)
        if (MDIChildren [i])
            if (MDIChildren [i]->InheritsFrom(__classid (TDocument)))
                {
                ((TDocument*)ActiveMDIChild)->SaveClick();
                if (((TDocument*)ActiveMDIChild)->m_Filename == "")
                    {
                    MessageBox (NULL, "You must save file before compiling it.\nCompilation will now
stop.", "Error", MB_OK);
                    return;
                    }

                AnsiString f = ActiveMDIChild->Caption;
                Syst->CompileFile (((TDocument*)ActiveMDIChild)->m_Filename.c_str(), f.c_str());
                }
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::ShowMessagesWindow1Click(TObject *Sender)
{
```

```
    MessageBox (NULL, "Not implemented yet", "Error", MB_OK);
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Stepinto1Click(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->stepIn();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::StepOver1Click(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->step();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::StepOut1Click(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->stepOut();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Pauseatexecutionpoint1Click(TObject *Sender)
{
    if (ActiveMDIChild)
        if (ActiveMDIChild->InheritsFrom(__classid (TDocument)))
            ((TDocument*)ActiveMDIChild)->Edit->SelLength = 0;
    Syst->pause();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::FileSaveAllClick(TObject *Sender)
{
    //
    for (int i = 0; i < MDIChildCount; i++)
        if (MDIChildren [i])
            if (MDIChildren [i]->InheritsFrom(__classid (TDocument)))
                {
                ((TDocument*)MDIChildren[i])->SaveClick();
                }
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::FormShow(TObject *Sender)
{
    //
/*    if (Syst->Ok())
        {
        DebugToolBar->Enabled = true;
        }
    else
        {
        DebugToolBar->Enabled = false;
        Compiler1->Enabled = false;
        Compile1->Enabled = false;
        CompileAll1->Enabled = false;
        Debug1->Enabled = false;
        }*/

}
//---------------------------------------------------------------------------
```

Файл SmalltalkEditor\Builder\Options.cpp

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Options.h"
#include "globals.h"
//---------------------------------------------------------------------
#pragma link "CSPIN"
#pragma resource "*.dfm"
TOptionsDlg *OptionsDlg;
//---------------------------------------------------------------------
__fastcall TOptionsDlg::TOptionsDlg(TComponent* AOwner)
    : TForm(AOwner)
{
}
//---------------------------------------------------------------------
```

```
void __fastcall TOptionsDlg::ColorsSheetShow(TObject *Sender)
{
    //
    TextExample->Color = MakeColor (BackColor);
    TextExample->Font->Color = MakeColor (ColorsList->ItemIndex + 1);
}
//---------------------------------------------------------------------------

void __fastcall TOptionsDlg::ChangeColorBtnClick(TObject *Sender)
{
    //
    ColorDialog->Color = MakeColor (BackColor);

    if (ColorDialog->Execute())
        {
        int* p = MakeColorPointer (ColorsList->ItemIndex + 1);
        if (p)
            *p = ColorDialog->Color;

        TextExample->Color = MakeColor (BackColor);
        TextExample->Font->Color = MakeColor (ColorsList->ItemIndex + 1);
        }
}
//---------------------------------------------------------------------------

void __fastcall TOptionsDlg::ColorsListClick(TObject *Sender)
{
    TextExample->Color = MakeColor (BackColor);
    TextExample->Font->Color = MakeColor (ColorsList->ItemIndex + 1);
}
//---------------------------------------------------------------------------

void __fastcall TOptionsDlg::ColorizerEnablerClick(TObject *Sender)
{
    //
    ColorizeKeysEnabled = ColorizerEnabler->Checked;
}
//---------------------------------------------------------------------------
```

Файл SmalltalkEditor\Builder\SearchReplace.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "SearchReplace.h"
#include "Main.h"
#include "MainFrm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSearchReplaceForm *SearchReplaceForm;
//---------------------------------------------------------------------------
__fastcall TSearchReplaceForm::TSearchReplaceForm(TComponent* Owner)
    : TForm(Owner)
{
    m_LastSearchSucceded = false;
    m_PromptError = true;
}
//---------------------------------------------------------------------------
void __fastcall TSearchReplaceForm::CancelClick(TObject *Sender)
{
    Hide ();
}
//---------------------------------------------------------------------------
void __fastcall TSearchReplaceForm::SearchClick(TObject *Sender)
{
    doc = NULL;

    if (MainForm->ActiveMDIChild)
        if (MainForm->ActiveMDIChild->InheritsFrom (__classid (TDocument)))
            doc = (TDocument*)MainForm->ActiveMDIChild;

    if (!doc) {Hide(); return;}

    TSearchTypes searchoptions = TSearchTypes();
    if (MatchCase->Checked)
        searchoptions << stMatchCase;
    if (MatchWholeWord->Checked)
        searchoptions << stWholeWord;

    switch (SearchWhere->ItemIndex)
        {
        case 0: //Forward
            m_SearchStart = doc->Edit->SelStart + doc->Edit->SelLength;
            m_SearchEnd = doc->Edit->Text.Length();
            break;
        case 1://Backward
            m_SearchStart = 0;
```

```
            m_SearchEnd = doc->Edit->SelStart + doc->Edit->SelLength;
            break;
        case 2://Entire CurrentDoc
            m_SearchStart = 0;
            m_SearchEnd = doc->Edit->Text.Length();
            SearchWhere->ItemIndex = 0;
            break;
        case 3://All Docs
            SearchWhere->ItemIndex = 0;
            MessageBox (NULL, "Search in all open docs is not implemented now", "", MB_OK);
            return;
        case 4://CurrentSel
            m_SearchStart = doc->Edit->SelStart;
            m_SearchEnd = doc->Edit->SelLength + m_SearchStart;
            SearchWhere->ItemIndex = 0;
            break;
        default: return;
        }

    m_SearchStart = doc->Edit->FindTextA(FindWhat->Text, m_SearchStart, abs (m_SearchStart - m_SearchEnd),
searchoptions);
    if (m_SearchStart == -1)
        {
        if (m_PromptError) MessageBox (NULL, ("Search text\n\"" + FindWhat->Text + "\"\n not found in
specified region with specified options\nTry search with Case and WholeWord settings turned
off.").c_str(), "Search", MB_OK);
        m_LastSearchSucceded = false;
        m_SearchStart = 0;
        return;
        }
    doc->Edit->SelStart = m_SearchStart;
    doc->Edit->SelLength = FindWhat->Text.Length();
    doc->Edit->HideSelection = false;
    m_LastSearchSucceded = true;
}
//---------------------------------------------------------------------------
void __fastcall TSearchReplaceForm::ReplaceClick(TObject *Sender)
{
    SearchClick (this);
    if (m_LastSearchSucceded)
        if (doc)
            doc->Edit->SelText = ReplaceWith->Text;
}
//---------------------------------------------------------------------------
void __fastcall TSearchReplaceForm::ReplaceAllClick(TObject *Sender)
{
    if (MessageBox (NULL, "Replace all?", "?", MB_YESNO) == IDYES)
        do
            {
            SearchClick (this);
            if (m_LastSearchSucceded)
                {
                m_PromptError = false;
                if (doc)
                    doc->Edit->SelText = ReplaceWith->Text;
                }
            else break;
            }
        while (m_LastSearchSucceded);
    m_PromptError = true;
}
//---------------------------------------------------------------------------
void __fastcall TSearchReplaceForm::FormKeyPress(TObject *Sender, char &Key)
{
    if (Key == 27) Hide ();
}
//---------------------------------------------------------------------------
void __fastcall TSearchReplaceForm::FormShow(TObject *Sender)
{
    FindWhat->SetFocus();
}
//---------------------------------------------------------------------------
```

Файл SmallVM\DebugObjectsTree.cpp

```
#include "DebugObjectsTree.h"
#include "Standart Smalltalk Library\fundamental protocols.h"
#include "Standart Smalltalk Library\numeric protocols.h"
#include "Standart Smalltalk Library\stream protocols.h"
#include "Standart Smalltalk Library\collection protocols.h"
#include "SmallVMCore.h"

CDebugObjectsTree::CDebugObjectsTree ( const CString& str, CObj* obj) :
    myObjects ( 5),
    myMe ( obj),
    myName ( str)
    {
    }

CDebugObjectsTree::~CDebugObjectsTree()
```

```
    {
        for ( int i = 0; i < myObjects.rNumber(); i++)
            if ( myObjects[i])
                delete myObjects[i];
    }

bool CDebugObjectsTree::ok()
    {
        return myMe && myMe->rRefsNum() > 0;
    }

const char* CDebugObjectsTree::asString()
    {
        if ( !ok()) {
            if ( !myMe)
                myString = "<<NULL>>";
            else //myMe->rRefsNum <= 0
                myString = "<<deleted>>";
            }
        else {
            if ( !myMe->rClass())
                myString = "<<Bad Class>>";
            else if ( myMe->rClass() == Character::rClass()) {
                char c = myMe->call ( "asCharacter");
                char str[] = "$c";
                str[1] = c;
                myString = str;
                }
            else if ( myMe->rClass() == SString::rClass()) {
                const char* str = myMe->call ( "asString");
                myString = str;
                }
            else if ( myMe->rClass() == Integer::rClass()) {
                int c = myMe->call ( "asInteger");
                char str[256];
                sprintf ( str, "%d", c);
                myString = str;
                }
            else if ( myMe->rClass() == Float::rClass()) {
                double c = myMe->call ( "asFloat");
                char str[256];
                sprintf ( str, "%lg", c);
                myString = str;
                }
            else
                myString = SmallVM::Linker->getName ( myMe->rClass()->myClassName);
        }
    return myString.rString();
    }

const char* CDebugObjectsTree::rName()
    {
        return myName.rString();
    }

int CDebugObjectsTree::number()
    {
        return myObjects.rNumber();
    }

std::auto_ptr<IDebugObjectsTree> CDebugObjectsTree::operator[] ( int index)
    {
        return std::auto_ptr<IDebugObjectsTree> ( new CDebugObjectsTree ( myObjects[index]->first,
myObjects[index]->second));
    }

IDebugObjectsTree* CDebugObjectsTree::expand()
    {
        CSmObj* sm = 0;
        if ( sm = dynamic_cast<CSmObj*> ( myMe)) {
            for ( int i = myObjects.rNumber() - 1; i >= 0; i--) {
                if ( myObjects[i]) delete myObjects[i];
                myObjects.del ( i);
                }
            const CCont < CVar > * vars = & sm->getScope().rVars();
            for ( int i = 0; i < vars->rNumber(); i++)
                if ( vars->at ( i) .rName())
                    myObjects.add ( new std::pair<CString, CObj*> ( SmallVM::Linker->getName ( vars->at (
i) .rName()), vars->at ( i) .rObj().rObj()));
            }
        return this;
    }

IDebugObjectsTree* CDebugObjectsTree::unexpand()
    {
        myObjects = CCont < std::pair< CString, CObj* > * > ( 5);
        return this;
    }

CDebugObjectsTree* CDebugObjectsTree::addObject ( const CString& name, CObj* obj)
    {
        myObjects.add ( new std::pair<CString, CObj*> ( name, obj));
        return this;
    }

/*
class CDebugObjectsTree : public IDebugObjectsTree
```

```
    {
    private:
    CObj* myMe; //ёюсёЄтхээ√щ ■
    CString myString;
    CCont < CObj* > myObjects;

    public:
        CDebugObjectsTree ( CObj* obj);
        ~CDebugObjectsTree();

        bool ok();

        const char* asString();

        int number();
        IDebugObjectsTree* operator[] ( int index);
        IDebugObjectsTree* expand();

        CDebugObjectsTree* addObject ( CObj* obj);
    };
*/
```

Файл SmallVM\SmallVMCore.cpp

```
#include "system.h"
#include "SmallVMCore.h"
//#include "hardware.h"
//#include "token.h"

#include "Standart Smalltalk Library\Fundamental Protocols.h"
#include "Standart Smalltalk Library\Numeric Protocols.h"
#include "Standart Smalltalk Library\Collection Protocols.h"

#include "DebugObjectsTree.h"

SClassRef BlockObj::sClass = SClassRef ( 0);

//-----------------------------------------------------------------------------------------

CCommand::CCommand (action code, int param) :
    myCode (code),
    myParam (param)
    { }
CCommand::~CCommand() { }

bool CCommand::operator== ( const CCommand& com)
    { return myCode == com.rCode() && myParam == com.rParam(); }

//-----------------------------------------------------------------------------------------------------

Block::Block()
{ }

Block::Block ( int i) //Block(0)
{
    Assert ( i == 0 && "in Block ( i)");
}

Block::Block ( const Block& block) :
    myCommands ( block.rCommands()),
    myScope ( block.rScope())
    { }

bool Block::operator== ( const Block& block) const
{
    return myCommands == block.rCommands() /*&&
            myScope == block.rScope()*/;
}

Block* Block::addCommand ( CCommand::action action, int parameter)
{
    return addCommand ( CCommand ( action, parameter));
}

Block* Block::addCommand ( const CCommand& com)
{
    myCommands.add ( com);
    return this;
}

Block* Block::addLocalVar ( int name)
{
    myScope.addVar ( name);
    return this;
}

SParam Block::call ( CScope* upper, const CParams& params)
    {
    bool workScopeIsSet = 0;
```

```
    CScope locals = myScope;
    locals.setParent ( upper);

    for ( int i = 1 ; i < params.num + 1; i++)
        locals[i].gObj() = CSmallVM::Par2Obj ( params[i-1]); //initialize local variables-params

    for ( int comIndex = 0; comIndex < myCommands.rNumber(); comIndex++) {
        CCommand com = myCommands[comIndex];
        switch (com.rCode())
        {
            case CCommand::eNew : {
                switch ( com.rParam()) {
                    case CCommand::stdTypes::eInteger : {
                        SmallVM::SmallVM->getStack().pushObj ( new Integer ( SmallVM::SmallVM-
>getStack().pop()));
                        break;
                        }
                    case CCommand::stdTypes::eFloat : {
                        long f = SmallVM::SmallVM->getStack().pop();
                        SmallVM::SmallVM->getStack().pushObj ( new Float ( *reinterpret_cast <float*> (
&f)));
                        break;
                        }
                    case CCommand::stdTypes::eChar : {
                        long c = SmallVM::SmallVM->getStack().pop();
                        Assert ( c < 256);
                        SmallVM::SmallVM->getStack().pushObj ( new Character ( c));
                        break;
                        }
                    case CCommand::stdTypes::eBlock : { //top of stack - is a BlockPtr.rIndex()
                        SmallVM::SmallVM->getStack().pushObj ( new BlockObj ( SmallVM::SmallVM-
>getStack().pop(), &locals));
                        break;
                        }
                    case CCommand::stdTypes::eStr : { //top of stack - is a ConstStringPtr.rIndex()
                        ConstStringPtr pStr = SmallVM::SmallVM->getStack().pop();
                        SmallVM::SmallVM->getStack().pushObj ( new SString ( pStr->rString()));
                        break;
                        }
                    default :
                        Assert ( ! "Smalltalk virtual machine: command 'new' has wrong parameter. possible
type is not suuported.");
                        break;
                    }
                }
                break;
            case CCommand::eCall : {
                //Command parameter is a myCommands name. Stack: int paramsNumber, obj param[n...1], obj
receiver
                CParams params = CParams();
                int paramNum = SmallVM::SmallVM->getStack().pop();
                for ( int i = paramNum-1; i >= 0; i--)
                    params[i] = CSmallVM::Obj2Par ( SmallVM::SmallVM->getStack().popObj());
                CObjRef receiver = SmallVM::SmallVM->getStack().popObj();
                int messageName = com.rParam();
                SParam res = receiver->call ( messageName, params);
                SmallVM::SmallVM->getStack().pushObj ( CSmallVM::Par2Obj ( res));
                break;
                }
            case CCommand::eSetWorkScope :
                SmallVM::SmallVM->setWorkScope ( &locals);
                workScopeIsSet = 1;
                break;
            case CCommand::eUpdateWorkPosition :
                SymbolPosition pos;
                pos.myFileName = SmallVM::SmallVM->getStack().pop();
                pos.myPosition = com.rParam();
                if ( SmallVM::Debugger)
                    SmallVM::Debugger->updateWorkPosition ( pos);
                break;
            case CCommand::ePushI :
                SmallVM::SmallVM->getStack().push ( com.rParam());
                break;
            case CCommand::ePop : {
                SmallVM::SmallVM->getStack().popObj();
                break;
                }
            case CCommand::ePushG :
                {
                CObjRef& o = locals.gVar ( com.rParam()) . gObj();
                Assert ( o.rObj() && "tryin' to push a 0 object");
                SmallVM::SmallVM->getStack().pushObj ( o);
                }
                break;
            case CCommand::ePushL :
                {
                int p = com.rParam();
                Assert ( 0 <= p && p < locals.rVars().rNumber());
                SmallVM::SmallVM->getStack().pushObj ( locals[p]);
                }
                break;
            case CCommand::ePopL :
                {
                int p = com.rParam();
                Assert ( 0 <= p && p < locals.rVars().rNumber());
                locals[p] = SmallVM::SmallVM->getStack().popObj();
                }
```

```
                break;
            case CCommand::ePopG :
                {
                locals.gVar ( com.rParam()) . gObj() = SmallVM::SmallVM->getStack().popObj();
                }
                break;
            case CCommand::eReturn :
                if ( workScopeIsSet) SmallVM::SmallVM->releaseWorkScope();
                return CSmallVM::Obj2Par ( SmallVM::SmallVM->getStack().popObj());

            default : Assert ( ! "Smalltalk VM: Unknown command in function!");
            }
        }

    Assert ( ! "smalltalk funcs are not working yet");
    return 0;
    }

//-------------------------------------------------------------------------------------------------

BlockPtr::BlockPtr() :
    myIndex ( 0)
    { }

BlockPtr::BlockPtr ( int i) :
    myIndex ( i)
    { }

int BlockPtr::gIndex()
    {
        return myIndex;
    }

Block* BlockPtr::operator-> ()
    { return gBlock(); }

Block* BlockPtr::gBlock()
{
    return SmallVM::SmallVM->getBlock ( myIndex);
}

//-------------------------------------------------------------------------------------------------

BlockObj::BlockObj ( BlockPtr block, CScope* parent) :
    myBlock ( block),
    myParent ( parent)
    { myClass = sClass; }

bool BlockObj::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ("BlockObj", "");

    SmallVM::CPPVM->registerFunc ( sClass, "call", (pObjFunc) call, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "call:", (pObjFunc) callWith, (PARAMS SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "callWith:", (pObjFunc) callWith, (PARAMS SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "callWith:With:", (pObjFunc) callWithWith, (PARAMS
SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "callWith:With:With:", (pObjFunc) callWithWithWith, (PARAMS
SParam::eAny));

    return 1;
    }

SParam BlockObj::call ( const CParams& params)
{
    return myBlock->call ( myParent, params);
}
SParam BlockObj::callWith ( const CParams& params)
{
    return myBlock->call ( myParent, params);
}
SParam BlockObj::callWithWith ( const CParams& params)
{
    return myBlock->call ( myParent, params);
}
SParam BlockObj::callWithWithWith ( const CParams& params)
{
    return myBlock->call ( myParent, params);
}
/*
SParam BlockObj::while ( const CParams& params) 1 - bool
{
    return myBlock->call ( myParent, params);
}*/

//-------------------------------------------------------------------------------------------------

ConstString::ConstString()
    {
    }

ConstString::ConstString ( const CString& str) :
    myString ( str)
    {
    }

ConstString::ConstString ( const ConstString& str) :
```

```
    myString ( str.rStdString())
    {
    }

ConstString& ConstString::operator = ( const ConstString& str)
    {
    myString = str.rStdString();
    return *this;
    }

const char* ConstString::rString() const
    {
        return myString.rString();
    }

bool ConstString::operator== ( const ConstString& str) const
    {
        return myString == str.rStdString();
    }

ConstStringPtr::ConstStringPtr ( int i) :
    myIndex ( i)
    {
    }

ConstString* ConstStringPtr::operator-> () { return gString(); }
ConstString* ConstStringPtr::gString()
    {
        return SmallVM::SmallVM->getConstString ( myIndex);
    }

//---------------------------------------------------------------------------------------------------

ISmallFunc::~ISmallFunc()
    {
    }

ISmallFunc::ISmallFunc ( int className, int funcName, const BlockPtr& bl) :
    mySelInL ( 0),
    myClassName ( className),
    myName ( funcName),
    myBlock ( bl)
    { }

bool ISmallFunc::operator== ( const ISmallFunc& func) const
    { return myName == func.myName && myClassName == func.myClassName; }

ISmallFunc& ISmallFunc::addParamType ( SParamTypeInfo t)
    {
    myTypes << t;
    return *this;
    }

//---------------------------------------------------------------------------------------------------

SmallFunc::SmallFunc ( int className, int funcName, const BlockPtr& bl) :
    ISmallFunc ( className, funcName, bl)
    { }

SmallFunc::~SmallFunc()
    {
    //SHOW ( SmallVM::Console->printf ( "SmallFunc is being destructed. press any key.\n"), 15);
    }

//---------------------------------------------------------------------------------------------------

SmallStaticFunc::SmallStaticFunc (int className, int funcName, const BlockPtr& bl) :
    ISmallFunc ( className, funcName, bl)
    { }

SmallStaticFunc::~SmallStaticFunc()
    {
    //SHOW ( SmallVM::Console->printf ( "SmallStaticFunc is being destructed. press any key.\n"), 15);
    }

//---------------------------------------------------------------------------------------------------

CSmObj::CSmObj ( SmallClass* cl) :
    myScope ( cl? &cl->myScope : 0),
    myClassObject ( cl)
    { }

CSmObj::CSmObj ( const CSmObjPrototype& objPro) :
    myClassObject ( objPro.rClassObject()),
    myScope ( objPro.rScope())
    {
    Assert ( CSmObj::myClass = objPro.rClass());
    }

bool CSmObj::operator== ( const CSmObj& obj)
    { return myClassObject == obj.rClassObject() && myClass == obj.rClass(); }

//---------------------------------------------------------------------------------------------------

CSmObjPrototype::CSmObjPrototype ( SmallClass* cl) :
    CSmObj ( cl)
    { Assert ( cl); }
```

```
bool CSmObjPrototype::ok() const
    { return myClassObject? 1 : 0; }

CSmObjPrototype& CSmObjPrototype::addVar ( const CVar& var)
    { Assert ( CSmObjPrototype::ok());
      myScope.addVar ( var);
      return *this; }

//-----------------------------------------------------------------------------------------------------------

SmallClass::SmallClass ( int className) :
    myClassName ( className),
    myClassRef ( ),
//  myStaticFuncs (1),
    myFuncs (1),
    myBaseClass ( 0),
    myScope ( SmallVM::Linker->getGlobalScope()),
#pragma warning (disable: 4355) //'this' used in base member initializer list is right here
    myObjPro ( this)
#pragma warning (default: 4355)
    {
    myFuncs.add ( SmallFunc());
    myStaticFuncs.add ( SmallStaticFunc());
    }

int SmallClass::getFuncSelector ( int funcName)
    {
    for ( int i = 1; i < myFuncs.rNumber(); i++)
        if ( myFuncs[i].myName == funcName &&
             myFuncs[i].myClassName == myClassName/*check*/) return i;
    return 0;
    }

int SmallClass::getStaticFuncSelector ( int funcName)
    {
        for ( int i = 1; i < myStaticFuncs.rNumber(); i++)
        if ( myStaticFuncs[i].myName == funcName &&
             myStaticFuncs[i].myClassName == myClassName/*check*/) return i;
    return 0;
    }

bool SmallClass::operator== ( const SmallClass& cl) const
    { return myClassName == cl.myClassName; }

SmallClass& SmallClass::addFunc ( const SmallFunc& func)
    {
    for ( int i = 1; i < myFuncs.rNumber(); i++)
        if ( myFuncs[i].myName == func.myName) {
            SmallVM::Console->printf ( "Smalltalk compiler: redefining method '%s' of class '%s'\n",
                SmallVM::Linker->getName ( func.myName).rString(), SmallVM::Linker->getName (
func.myClassName).rString());
            myFuncs[i].myBlock = func.myBlock;
            myFuncs[i].myTypes = func.myTypes;
            myClassRef->myVTable.functions[myFuncs[i].mySelInL].myTypes = func.myTypes;
            return *this;
            }
    myFuncs.add ( func);
    Func f ( SmallVM::SmallVM, func.myName, myClassName, func.myTypes);
    myClassRef->registerFunc ( f);
    myFuncs.back().mySelInL = myClassRef->getFuncSelector ( func.myName);
    return *this;
    }

SmallClass& SmallClass::addStaticFunc ( const SmallStaticFunc& func)
    {
    for ( int i = 1; i < myStaticFuncs.rNumber(); i++)
        if ( myStaticFuncs[i].myName == func.myName) {
            SmallVM::Console->printf ( "Smalltalk compiler: redefining static method '%s' of class '%s'",
                SmallVM::Linker->getName ( func.myName).rString(), SmallVM::Linker->getName (
func.myClassName).rString());
            myStaticFuncs[i].myBlock = func.myBlock;
            myStaticFuncs[i].myTypes = func.myTypes;
            myClassRef->myVTable.functions[myStaticFuncs[i].mySelInL].myTypes = func.myTypes;
            return *this;
            }
    myStaticFuncs.add ( func);
    StaticFunc f ( SmallVM::SmallVM, func.myName, myClassName, func.myTypes);
    myClassRef->registerStaticFunc ( f);
    myStaticFuncs.back().mySelInL = myClassRef->getStaticFuncSelector ( func.myName);
    return *this;
    }

SmallClass& SmallClass::setBaseClass ( SmallClass* base)
    {
    Assert ( base && "tring to set a 0 base class to Smalltalk class");
    myBaseClass = base;
    myScope.setParent ( & base->myScope);
    return *this;
    }

SmallClass& SmallClass::regBaseClass ( int baseClassName)
    {
    Assert ( baseClassName);
    SmallVM::SmallVM->registerBaseClass ( myClassName, baseClassName);
    SmallClass* basePtr = SmallVM::SmallVM->getClassPtr ( baseClassName);
    Assert ( basePtr && "smalltalkVM, registering base class: no class was found"); //!!! redo
```

```
    setBaseClass ( basePtr);
    return *this;
    }

bool SmallClass::selfReg()
    {
    Assert ( myClassName);
    myClassRef = SmallVM::CPPVM->registerClass ( myClassName, myClassName); //Linker->registerClass...
included
    myObjPro.setClassRef ( myClassRef);


    RegisterStaticFuncForSmallClass ( myClassRef, "new", (pSmallClassFunc) sNew, CParamsTypes());

    return 1;
    }

SParam SmallClass::call ( CSmObj* obj, int funcName, int funcSel, const CParams& params)
    {
    if (funcSel <= 0 || funcSel > myFuncs.rNumber()) funcSel = getFuncSelector (funcName);
    Assert ( funcSel && "SmallVM & Linker internal error. tryin' to call not existing SMALLTALK function
");
    Assert ( myFuncs[funcSel].mySelInL);

    int paramsExpected = myClassRef->myVTable.functions [ myFuncs[funcSel].mySelInL].myTypes.num;
    Assert ( paramsExpected == params.num && "wrong number of params to called smalltalk function");

    CScope s ( & obj->getScope(), 1);
    s.addVar ( CVar ( CObjRef ( obj), "self"));

    return myFuncs[funcSel].myBlock->call ( &s, params);
    }

SParam SmallClass::staticCall ( int funcName, int funcSel, const CParams& params)
    {
    if (funcSel <= 0 || funcSel > myStaticFuncs.rNumber()) funcSel = getStaticFuncSelector (funcName);
    Assert ( funcSel && "SmallVM & Linker internal error. tryin' to call not existing SMALLTALK static
function ");
    Assert ( myStaticFuncs[funcSel].mySelInL);

    int paramsExpected = myClassRef->myStaticFuncs [ myStaticFuncs[funcSel].mySelInL].myTypes.num;
    Assert ( paramsExpected == params.num && "wrong number of params to called smalltalk function");

    return myStaticFuncs[funcSel].myBlock->call ( & myScope, params);
    }

SParam SmallClass::sNew ( const CParams& )
    {
    return new CSmObj ( myObjPro);
    }

//-------------------------------------------------------------------------------------------------------

CStack::CStack() :
    myObjs ( 100),
    myRefs ( 100)
    { }
CStack::~CStack() { }

bool CStack::ok() { return myObjs.ok() && myRefs.ok(); }

CStack& CStack::push ( int obj)
    { Assert ( ok());
      myObjs.add ( obj);
      return *this; }

CStack& CStack::pushObj ( const CObjRef& obj)
    { Assert (ok());
      myRefs.add ( obj);
      return *this; }

int CStack::pop()
    { Assert ( ok());
      Assert ( myObjs.rNumber() > 0);
      int r = myObjs [ myObjs.rNumber() - 1];
      myObjs.del ( myObjs.rNumber() - 1);
      return r; }

CObjRef CStack::popObj()
    { Assert (ok());
      Assert ( myRefs.rNumber() > 0);
      CObjRef r = myRefs [ myRefs.rNumber() - 1];
      myRefs.del ( myRefs.rNumber() - 1);
      return r; }

//-------------------------------------------------------------------------------------------------------

APBlock::APBlock() :
      myBlock ( std::auto_ptr<Block>())
      { }

APBlock::APBlock ( std::auto_ptr<Block> bl) :
      myBlock ( bl)
      { }

APBlock& APBlock::operator= ( const APBlock& bl)
      {
```

```
          myBlock = *const_cast <std::auto_ptr<Block>*> ( &bl.myBlock);
          return *this;
      }

bool APBlock::operator== ( const APBlock& bl)
    { return 0; } //фєсышЁє■•шїё  с√Є№ эх ьюцхЄ, шёърЄ№ эх яюЄЁхсєхЄё

std::auto_ptr<Block>* APBlock::operator-> ()
    { return &myBlock; }

//-----------------------------------------------------------------------------------------------------------

IMPLEMENT_IUNKNOWN ( CSmallVM, ISmallVM, IID_ISmallVM);

CSmallVM::CSmallVM() :
    myClasses (1),
    myStrings ( 2)
    {
        myClasses.add ( 0);
        myBlocks.add ( std::auto_ptr<Block> ());
        myStrings.add ( CString ( "Error-string in Small VM ~!@#$%^&*()"));
    }

SParam CSmallVM::call ( CObj* obj, int classNam e, int classSel, int funcName, int funcSel, const CParams&
params)
    {
    CSmObj* smalltalkObjectPtr = static_cast<CSmObj*> (obj);
    Assert ( smalltalkObjectPtr && "trying to call smalltalk function form non-smalltalk object");

    if ( classSel <= 0 || classSel >= myClasses.rNumber()) classSel = getClassSelector ( funcName);
    Assert ( classSel && "trying to call function of not existing class");

    return myClasses[classSel]->call ( smalltalkObjectPtr, funcName, funcSel, params);
    }

SParam CSmallVM::staticCall ( int classNam e, int classSel, int funcName, int funcSel, const CParams&
params)
    {
    if ( classSel <= 0 || classSel >= myClasses.rNumber()) classSel = getClassSelector ( funcName);
    Assert ( classSel && "trying to static call function of not existing class");

    return myClasses[classSel]->staticCall ( funcName, funcSel, params);
    }

std::auto_ptr<IDebugObjectsTree> CSmallVM::getWorkTree()
    {
        CDebugObjectsTree* res = new CDebugObjectsTree ( "some scope", 0);
        CScope * scope = getWorkScope();
        while ( scope) {
            //if ( scope->getParent())
            //  res->addObject ( "deeper scope", ??? scope->getParent());
            for ( int i = 0; i < scope->rVars().rNumber(); i++)
                if ( scope->rVars().at ( i) .rName())
                    res->addObject ( SmallVM::Linker->getName ( scope->rVars().at ( i) .rName()), scope-
>rVars().at ( i) .rObj().rObj());
            scope = scope->getParent();
            }
        return std::auto_ptr<IDebugObjectsTree> ( res);
    }

SParam CSmallVM::Obj2Par ( CObjRef& obj)
    {
    Assert ( obj);
    if ( obj->rClass() == Integer::rClass())
        {
        Integer* i = dynamic_cast<Integer*> ( obj.rObj());
        Assert ( i);
        return i->rInteger();
        }
    if ( obj->rClass() == Float::rClass())
        {
        Float* f = dynamic_cast<Float*> ( obj.rObj());
        Assert ( f);
        return f->rFloat();
        }
    if ( obj->rClass() == Character::rClass())
        {
        Character* c = dynamic_cast<Character*> ( obj.rObj());
        Assert ( c);
        return c->rChar();
        }
    if ( obj->rClass() == SString::rClass())
        {
        SString* s = dynamic_cast<SString*> ( obj.rObj());
        Assert ( s);
        return s->rString().c_str();
        }
    if ( obj->rClass() == Boolean::rClass())
        {
        Boolean* b = dynamic_cast<Boolean*> ( obj.rObj());
        Assert ( b);
        return b->rBool();
        }

    return SParam ( obj);
    }
```

```
CObjRef CSmallVM::Par2Obj ( const SParam& par)
    {
    switch (par.type)
        {
        case SParam::eBool   : return new Boolean   ( par.param.b);
        case SParam::eChar   : return new Character ( par.param.c);
        case SParam::eInt    : return new Integer   ( par.param.i);
        case SParam::eLong   : return new Integer   ( par.param.l);
        case SParam::eFloat  : return new Float     ( par.param.f);
        case SParam::eDouble : return new Float     ( par.param.d);
        case SParam::eStr    : return new SString   ( par.param.str);
        case SParam::ePVoid  : return 0;
        case SParam::ePObj   : return par.rObj();
        case SParam::eAny    : return 0;
        default              : return 0;
        }
    }

SmallClass* CSmallVM::getClassPtr ( int className)
    {
    return myClasses [ getClassSelector ( className)];
    }

int CSmallVM::getClassSelector ( int className)
    {
    for ( int i = 1; i < myClasses.rNumber(); i++) {
        Assert ( myClasses[i]);
        if ( myClasses[i]->myClassName == className) return i;
        }
    return 0;
    }

int CSmallVM::getFuncSelector ( int className, int funcName)
    {
    int classSel = getClassSelector (className);
    if (!classSel) return 0;
    return myClasses[classSel]->getFuncSelector(funcName);
    }

int CSmallVM::getStaticFuncSelector ( int className, int funcName)
    {
    int classSel = getClassSelector (className);
    if ( !classSel) return 0;
    return myClasses[classSel]->getStaticFuncSelector ( funcName);
    }

BlockPtr CSmallVM::rFuncPtr ( Func* func)
    { return rFuncPtr ( func->myClassName, func->myClassSelectorInVM, func->myName, func->mySelectorInVM);
}

BlockPtr CSmallVM::rFuncPtr ( int className, int classSel, int funcName, int funcSel)
    {
    if ( !classSel) classSel = getClassSelector (className);
    Assert ( 0 < classSel && classSel < myClasses.rNumber());

    if ( !funcSel) funcSel = myClasses[classSel]->getFuncSelector (funcName);
    Assert ( 0 < funcSel && funcSel < myClasses[classSel]->myFuncs.rNumber());

    return myClasses[classSel]->myFuncs[funcSel].myBlock;
    }

BlockPtr CSmallVM::rStatFuncPtr ( StaticFunc* func)
    {
    Assert ( !"Small VM can't call a static func now");;
//  return rStatFuncPtr (func->myClassName, func->myClassSelectorInVM, func->myName, func-
>mySelectorInVM);
#pragma warning (disable: 4172) //returning address of local variable or temporary
    return BlockPtr();
    }

BlockPtr CSmallVM::rStatFuncPtr (int className, int classSel, int funcName, int funcSel)
    {/*
    if (!classSel) classSel = getClassSelector (className);
    Assert (0 < classSel && classSel < myClasses.rNumber());

    if (!funcSel) funcSel = myClasses[classSel]->getStaticFuncSelector (funcName);
    Assert (0 < funcSel && funcSel < myClasses[classSel]->myStaticFuncs.rNumber());

    return myClasses[classSel]->myStaticFuncs[funcSel].myFunc;
    */
    Assert ( !"Small VM can't call a static func now");;
    return BlockPtr();
#pragma warning (default: 4172)
    }

BlockPtr CSmallVM::regBlock ( std::auto_ptr<Block> block)
{
    Assert ( block.get() && "CSmallVM::regBlock ( block)");
    myBlocks.add ( block);
    return myBlocks.rNumber()-1;
}

Block* CSmallVM::getBlock ( int index)
{
    Assert ( 0 < index && index < myBlocks.rNumber() && "CSmallVM::getBlock ( index)");
    return myBlocks[index]->get();
}
```

```
ConstStringPtr CSmallVM::regConstString ( const char* str)
{
    myStrings.add ( ConstString ( CString ( str)));
    return myStrings.rNumber()-1;
}

ConstStringPtr CSmallVM::regConstString ( const ConstString& str)
{
    myStrings.add ( str);
    return myStrings.rNumber()-1;
}

ConstString* CSmallVM::getConstString ( int index)
{
    Assert ( 0 < index && index < myStrings.rNumber() && "CSmallVM::getConstString ( index)");
    return &myStrings[index];
}

SClassRef CSmallVM::registerClass ( int className, int varName)
    {
    if ( getClassSelector ( className))
        return SmallVM::Linker->getClassRef ( className);

    if ( int i = SmallVM::Linker->getClassRef (className))
        {
        myClasses.add ( new SmallClass ( className)); //will be deleted automatically
        return SClassRef ( i);
        }

    myClasses.add ( new SmallClass ( className)); //the same

    myClasses.back()->selfReg();

    return SmallVM::Linker->getClassRef ( className);
    }

SClassRef CSmallVM::registerBaseClass ( int className, int baseClassName)
    {
    SClass* cl = SmallVM::Linker->getClassPtr ( className);
    SClassRef base  = SmallVM::Linker->getClassRef ( baseClassName);
    Assert ( cl && "SmallVM: can't find class for addin' a base class to it"); //!!! redo
    Assert ( (SClass*)base && "SmallVM: can't find class to add it as a base class"); //!!! redo
    return & cl->registerParent ( base);
    }

StaticFunc* CSmallVM::registerStaticFunc ( SClassRef cl, int funcName, const BlockPtr& func, const
CParamsTypes& types)
    {/*
    int classSel = getClassSelector (cl->myClassName);
    Assert ( 0 < classSel && classSel < myClasses.rNumber()); //!!! redo

    for ( int i = 1; i < myClasses[classSel]->myStaticFuncs.rNumber(); i++)
        Assert ( myClasses[classSel]->myStaticFuncs[i].myName != funcName) //!!! redo

    myClasses[classSel]->myStaticFuncs.add ( SmallStaticFunc ( cl->myClassName, funcName, func));

    StaticFunc* fPtr =  cl->registerStaticFunc ( StaticFunc ( this, funcName, cl->myClassName, types));

    if (fPtr) return fPtr;

    myClasses[classSel]->myStaticFuncs.del ( myClasses[classSel]->myStaticFuncs.rNumber() - 1);

    return 0;
    */
    Assert ( !"Small VM can't register static funcs");
    return 0;
    }

Func* CSmallVM::registerFunc ( SClassRef cl, int funcName, const BlockPtr& func, const CParamsTypes&
types)
    {
    int sel = getClassSelector ( SmallVM::Linker->getClasses()[cl]->myClassName);
    Assert ( 0 < sel && sel < myClasses.rNumber()); //!!! redo

    for ( int i = 1; i < myClasses[sel]->myFuncs.rNumber(); i++)
        Assert ( myClasses[sel]->myFuncs[i].myName != funcName); //!!! redo

    myClasses[sel]->myFuncs.add ( SmallFunc ( myClasses[sel]->myClassName, funcName, func));

    Func* fPtr = SmallVM::Linker->getClasses()[cl]->registerFunc ( Func ( this, funcName, cl->myClassName,
types));

    if ( fPtr)
        { myClasses[sel]->myFuncs[ myClasses[sel]->myFuncs.rNumber()-1].mySelInL = cl->getFuncSelector (
funcName);
        return fPtr; }

    myClasses[sel]->myFuncs.del ( myClasses[sel]->myFuncs.rNumber() - 1);

    return 0;
    }

SmallObjFunc::SmallObjFunc ( SmallClass* cl, pSmallClassFunc func) :
    myClass ( cl),
    myFunc ( func)
    { }
```

```
SParam SmallObjFunc::operator() ( const CParams& params)
{
    Assert ( myClass && myFunc);
    return (myClass->*myFunc) ( params);
}

void RegisterStaticFuncForSmallClass ( SClassRef cl, const CString& funcName, pSmallClassFunc func, const
CParamsTypes& types)
{
    SmallVM::CPPVM->registerObjectFuncAsStatic ( cl, funcName, std::auto_ptr <ObjFunc> ( new SmallObjFunc
( SmallVM::SmallVM->getClassPtr ( cl->myClassName), func)), types);
}
```

Файл SmallVM\System.cpp

```
#include "System.h"

//#include <Exeption.h>      // use ExeptionEngine

ILoader* SmallVM::Loader = 0;
ILinker* SmallVM::Linker = 0;
CSmallVM* SmallVM::SmallVM = 0;
ICPPVM* SmallVM::CPPVM = 0;
IConsole* SmallVM::Console = 0;
ISmallDebugger* SmallVM::Debugger = 0;
```

Файл SmallVM\module.cpp

```
#define __PROGRAMMER "Se"
#define __MODULE_NAME "SmallVM"
#include "..\ExeptionEngine\exeption.h"
#include "System.h"
#include "SmallVMCore.h"

#include "Standart Smalltalk Library\Fundamental Protocols.h"
#include "Standart Smalltalk Library\Numeric Protocols.h"
#include "Standart Smalltalk Library\Stream Protocols.h"
#include "Standart Smalltalk Library\Collection Protocols.h"

    #define DLL_NAME "SmallVM"
    #define DLL_CLASS CSmallVM
    #include "..\Common\Utilities\Module.c"

#pragma warning (disable: 4275) // non dll-interface class used as base for dll-interface class

DLL_INIT(loader)
{
    Assert ( loader);
    SmallVM::Loader = loader;

    LinkerDataTypesLibInit ( loader);

    loader->GetInterface (IID_ILinker, (void**)&SmallVM::Linker);
    Assert ( SmallVM::Linker);
    //SmallVM::SmallVM = dynamic_cast<CSmallVM*> ( (ISmallVM*) SmallVM::Loader->GetInterface
(IID_ISmallVM));
    loader->GetInterface (IID_ISmallVM, (void**)&SmallVM::SmallVM);
    Assert ( SmallVM::SmallVM);
    loader->GetInterface (IID_IConsole, (void**)&SmallVM::Console);
    Assert (SmallVM::Console);
    loader->GetInterface (IID_ICPPVM, (void**)&SmallVM::CPPVM);
    Assert (SmallVM::CPPVM);
    loader->GetInterface (IID_ISmallDebugger, (void**)&SmallVM::Debugger);
//  Assert (SmallVM::Debugger); //!!!

    Assert ( FundamentalProtocolsReg() && NumericProtocolsReg() && StreamProtocolsReg() &&
CollectionProtocolsReg() && BlockObj::selfReg());

    SmallVM::Console->printf ("SmallVM initialized.\n");
}

DLL_ONLOAD (loader)
{
    IConsole* con = NULL;
    loader->GetInterface (IID_IConsole, (void**)&con);
    Assert (con);
    con->printf ("SMALLVM: Smalltalk VirtualMashine by Se version 1.0\n");
}
#pragma warning (default: 4275) // non dll-interface class used as base for dll-interface class
```

Файл SmallVM\Standard Smalltalk Library\Collection Protocols.cpp

```
#include "Collection Protocols.h"

//-----------------------------------------------------------------------------------------------------------

SClassRef SString::sClass;

//-----------------------------------------------------------------------------------------------------------

bool CollectionProtocolsReg()
    {
    return SString::selfReg();
    }

//-----------------------------------------------------------------------------------------------------------

SString::SString()
    {
        myClass = sClass;
    }

SString::SString ( const char* str) :
    myString ( str)
    {
        myClass = sClass;
    }

SString::~SString()
    {
    }

bool SString::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "string", "");

    SmallVM::CPPVM->registerFunc ( sClass, "+", (pObjFunc) operator+, (TYPES SParam::eStr));
    SmallVM::CPPVM->registerFunc ( sClass, "+=", (pObjFunc) operator+=, (TYPES SParam::eStr));
    SmallVM::CPPVM->registerFunc ( sClass, "asString", (pObjFunc) asString, CParamsTypes());

    return 1;
    }

SParam /*SString*/ SString::operator + ( const CParams& params)
    {
        Assert ( params.num == 1 && "SString::operator+");
        return new SString ( ( myString + params[0].rString()).c_str());
    }

SParam /*SString*/ SString::operator += ( const CParams& params)
    {
        Assert ( params.num == 1 && "SString::operator+=");
        myString += params[0].rString();
        return this;
    }

SParam /*eString*/ SString::asString ( const CParams& params)
    {
        return myString.c_str();
    }

SParam SString::sDo ( const CParams& params)
    {
    Assert ( params.num == 1);

    for ( unsigned int i = 0; i < myString.length(); i++)
        params[0].rObj() ("call:") (PARAMS myString[i]);

    return this;
    }

//-----------------------------------------------------------------------------------------------------------
```

Файл SmallVM\Standard Smalltalk Library\Fundamental Protocols.cpp

```
#include "Fundamental Protocols.h"
#include "Collection Protocols.h"
#include "..\SmallVMCore.h"
//-----------------------------------------------------------------------------------------------------------

int SmallLibraryMethodShowDepth = 9;

//-----------------------------------------------------------------------------------------------------------
SClassRef Character::sClass = SClassRef ( 0);
SClassRef Boolean::sClass = SClassRef ( 0);
const Boolean* True = 0;
const Boolean* False = 0;
```

```
const Nil*     NUL = 0;
SClassRef Nil::sClass = SClassRef ( 0);
SClassRef Instantiator::sClass = SClassRef ( 0);

//-------------------------------------------------------------------------------------------------------------

bool FundamentalProtocolsReg()
    { return Character::selfReg() && Boolean::selfReg() && Instantiator::selfReg(); }

//-------------------------------------------------------------------------------------------------------------

bool Character::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "character", "");

    SmallVM::CPPVM->registerFunc ( sClass, "asCharacter", (pObjFunc) asCharacter, (TYPES SParam::eChar));
    SmallVM::CPPVM->registerFunc ( sClass, "asLowercase", (pObjFunc) asLowercase, (TYPES SParam::eChar));

    return 1;
    }

Character::Character ( char i) :
    myChar ( i)
    { myClass = sClass; }

SParam Character::asLowercase ( const CParams& params)
    { return (char) tolower ( myChar); }

SParam Character::asCharacter ( const CParams& params)
    { return myChar; }

//-------------------------------------------------------------------------------------------------------------

bool Boolean::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "Boolean", "");
    //RegisterBaseClass ( &sClass, "Object");

    const Boolean* True = new Boolean ( true);
    const Boolean* False = new Boolean ( false);;

    SmallVM::CPPVM->registerFunc ( sClass, "isTrue", (pObjFunc) isTrue, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "isFalse", (pObjFunc) isFalse, CParamsTypes());

    SmallVM::CPPVM->registerFunc ( sClass, "ifTrue:", (pObjFunc) ifTrue, (TYPES SParam::ePObj));
    SmallVM::CPPVM->registerFunc ( sClass, "ifFalse:", (pObjFunc) ifFalse, (TYPES SParam::ePObj));

    SmallVM::CPPVM->registerFunc ( sClass, "and:", (pObjFunc) and, (TYPES SParam::eBool));
    SmallVM::CPPVM->registerFunc ( sClass, "&", (pObjFunc) and, (TYPES SParam::eBool));

    SmallVM::CPPVM->registerFunc ( sClass, "asString", (pObjFunc) printString, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "printString", (pObjFunc) printString, CParamsTypes());

    SmallVM::Linker->getGlobalScope().addVar ( CVar ( CObjRef ( const_cast<CObj*> ( ( const CObj*) True)),
"true")); //syntax analyzer wont let to change it
    SmallVM::Linker->getGlobalScope().addVar ( CVar ( CObjRef ( const_cast<CObj*> ( ( const CObj*)
False)), "false"));

    return 1;
    }

Boolean::Boolean ( bool b) :
    myBool ( b)
    { myClass = sClass; }

SParam Boolean::ifTrue ( const CParams& par) // 1 - block
{
    Assert ( par.num == 1 && "Boolean::ifTrue:");
//  BlockObj* pBlock = dynamic_cast<BlockObj*> ( par[0].rObj().rObj());
//  Assert ( pBlock && "Boolean::ifTrue:");
//  return myBool? pBlock->call ( CParams()) : SParam();
    return myBool? par[0].rObj() ("call")() : SParam();
}

SParam Boolean::and ( const CParams& par) // 1 - bool
    {
    Assert ( par.num == 1 && "Boolean::and:");
    return myBool && par[0].rBool();
    }

SParam Boolean::isTrue ( const CParams& par) // 0
    {
    Assert ( par.num == 0 && "Boolean::isTrue");
    return myBool;
    }

SParam Boolean::isFalse ( const CParams& par) // 0
    {
    Assert ( par.num == 0 && "Boolean::isFalse");
    return !myBool;
    }

SParam Boolean::ifFalse ( const CParams& par) // 1 - block
{
    Assert ( par.num == 1 && "Boolean::ifTrue:");
//  BlockObj* pBlock = dynamic_cast<BlockObj*> ( par[0].rObj().rObj());
//  Assert ( pBlock && "Boolean::ifTrue:");
```

```
//  return !myBool? pBlock->call ( CParams()) : SParam();
    return !myBool? par[0].rObj() ("call")() : SParam();
}

SParam Boolean::printString ( const CParams& par)
    {
        return new SString ( myBool? "true" : "false");
    }

//-----------------------------------------------------------------------------------------------------

bool Nil::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "Nil", "");
    //RegisterBaseClass ( &sClass, "Object");

    Nil* NUL = new Nil();
    NUL->incRefs();

    SmallVM::CPPVM->registerFunc ( sClass, "isEqual:", (pObjFunc) isEqual, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "==", (pObjFunc) isEqual, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "=", (pObjFunc) isEqual, CParamsTypes());

    SmallVM::Linker->getGlobalScope().addVar ( CVar ( CObjRef ( const_cast<CObj*> ( ( const CObj*) NUL)),
"nil")); //syntax analyzer wont let change it

    return 1;
    }

Nil::Nil()
    { myClass = sClass; }

SParam Nil::isEqual ( const CParams& par)
    {
        return ! par[0].rBool();
    }

SParam Nil::printString ( const CParams& par)
    {
        return new SString ( "nil");
    }

//-----------------------------------------------------------------------------------------------------

Instantiator::Instantiator()
    { Assert ( Instantiator::sClass);
      myClass = sClass; }

bool Instantiator::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "Instantiator", "");

    SmallVM::CPPVM->registerFunc ( sClass, "new", (pObjFunc) sNew, CParamsTypes());

    return 1;
    }

//-----------------------------------------------------------------------------------------------------
```

Файл SmallVM\Standard Smalltalk Library\Numeric Protocols.cpp

```
#include "Numeric Protocols.h"
#include "Collection Protocols.h"
#include "..\SmallVMCore.h"

#include <math.h>

SClassRef Magnitude::sClass = SClassRef ( 0);
SClassRef Number::sClass = SClassRef ( 0);
SClassRef Rational::sClass = SClassRef ( 0);
SClassRef Integer::sClass = SClassRef ( 0);
SClassRef Float::sClass = SClassRef ( 0);

//-----------------------------------------------------------------------------------------------------

bool NumericProtocolsReg()
    {
        return Magnitude::selfReg() && Number::selfReg() && Rational::selfReg() && Integer::selfReg() &&
Float::selfReg();
    }

//-----------------------------------------------------------------------------------------------------

Magnitude::Magnitude()
    { Assert ( Magnitude::sClass);
      myClass = Magnitude::sClass; } //!!! may be cleared, because no objects of this class must be
created

bool Magnitude::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ("Magnitude", "");
```

```
    SmallVM::CPPVM->registerFunc ( sClass, "<", (pObjFunc) operator<, (TYPES SParam::ePObj));
    SmallVM::CPPVM->registerFunc ( sClass, "<=", (pObjFunc) operator<=, (TYPES SParam::ePObj));
    SmallVM::CPPVM->registerFunc ( sClass, ">", (pObjFunc) operator>, (TYPES SParam::ePObj));
    SmallVM::CPPVM->registerFunc ( sClass, ">=", (pObjFunc) operator>=, (TYPES SParam::ePObj));
    SmallVM::CPPVM->registerFunc ( sClass, "=", (pObjFunc) operator==, (TYPES SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "isEqual:", (pObjFunc) operator==, (TYPES SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "between:and:", (pObjFunc) betweenAnd, (TYPES SParam::ePObj,
SParam::ePObj));
    SmallVM::CPPVM->registerFunc ( sClass, "max:", (pObjFunc) sMax, (TYPES SParam::ePObj));
    SmallVM::CPPVM->registerFunc ( sClass, "min:", (pObjFunc) sMin, (TYPES SParam::ePObj));

    return 1;
    }

//---------------------------------------------------------------------------------------------------

Number::Number()
    { Assert ( Number::sClass);
      myClass = Number::sClass; } //!!! may be cleared, because no objects of this class must be created

bool Number::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "Number", "");
    SmallVM::CPPVM->registerBaseClass ( "Number", "Magnitude");

    SmallVM::CPPVM->registerFunc ( sClass, "+", (pObjFunc) operator+, (TYPES SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "-", (pObjFunc) operator-, (TYPES SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "*", (pObjFunc) operator*, (TYPES SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "/", (pObjFunc) operator/, (TYPES SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "abs", (pObjFunc) abs, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "asInteger", (pObjFunc) asInteger, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "asFloat", (pObjFunc) asFloat, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "asString", (pObjFunc) asString, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "printString", (pObjFunc) asString, CParamsTypes());

    return 1;
    }

//---------------------------------------------------------------------------------------------------

Rational::Rational()
    { Assert ( Rational::sClass);
      myClass = Rational::sClass; } //!!! may be cleared, because no objects of this class must be created

bool Rational::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "Rational", "");
    SmallVM::CPPVM->registerBaseClass ( "Rational", "Number");

//  RegisterFunc ( &sClass, "denominator", (pObjFunc), denominator, CParamsTypes());
//  RegisterFunc ( &sClass, "numerator", (pObjFunc), numerator, CParamsTypes());

    return 1;
    }

SParam Rational::denominator ( const CParams& params) //0 params, return new
    {
    throw ( CSmallVM::NotRealizedMethod ( "protocol<Rational>: #denominator"));
    }

SParam Rational::numerator ( const CParams& params) //0 params, return new
    {
    throw ( CSmallVM::NotRealizedMethod ( "protocol<Rational>: #numerator"));
    }

//---------------------------------------------------------------------------------------------------

Integer::Integer() :
    myInteger ( 0L)
    { Assert ( Integer::sClass);
      myClass = Integer::sClass; } //!!! may be cleared, because no objects of this class must be created

Integer::Integer ( long i) :
    myInteger ( i)
    { Assert ( Integer::sClass);
      myClass = Integer::sClass; } //!!! may be cleared, because no objects of this class must be created

bool Integer::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "Integer", "Integer"); //!!! varname is used, not std
    SmallVM::CPPVM->registerBaseClass ( "Integer", "Rational");

    SmallVM::CPPVM->registerFunc ( sClass, "repeat:", (pObjFunc) repeat, (TYPES SParam::ePObj));
    SmallVM::CPPVM->registerStaticFunc ( sClass, "construct:", (pStatFunc) sConstruct, ( TYPES
SParam::eInt));

    return 1;
    }

SParam Integer::operator+ ( const CParams& params)
    {
    Assert ( params.num == 1);

    if ( params[0].type == SParam::ePObj)
        {
        CObjRef o = params[0].rObj();
```

```
        Assert ( o.rObj());  // redo
        if ( o->rClass() == Integer::rClass())
            {
            Integer* ip = (Integer*) ( o.rObj()); // !!! RTTI is very important
            Assert ( ip); //if rClass() is Integer, then object is integer
            return myInteger + ip->rInteger();
            }
        //if ( o->rClass() == Float::rClass())
        //...
        throw
            ( CObj::WrongParameter ("protocol<Integer>: #+ : "
                                    "param int or Integer-object (Float... isn't realized yet) is
expected"));
            }
    else
        return myInteger + params[0].rInt();
    }

SParam Integer::operator- ( const CParams& params)
    {
    Assert ( params.num == 1);

    if ( params[0].type == SParam::ePObj)
        {
        CObjRef o = params[0].rObj();
        Assert ( o.rObj());
        if ( o->rClass() == Integer::rClass())
            {
            Integer* ip = dynamic_cast<Integer*> ( o.rObj());
            Assert (ip); //if rClass() is Integer, then object is integer
            return myInteger - ip->rInteger();
            }
        //if ( o->rClass() == Float::rClass())
        //...
        throw
            ( CObj::WrongParameter ("protocol<Integer>: #- : "
                                    "param int or Integer-object (Float... isn't realized yet) is
expected"));
            }
    else
        return myInteger - params[0].rInt();
    }

SParam Integer::operator* ( const CParams& params)
    {
    Assert ( params.num == 1);

    if ( params[0].type == SParam::ePObj)
        {
        CObjRef o = params[0].rObj();
        Assert ( o.rObj());
        if ( o->rClass() == Integer::rClass())
            {
            Integer* ip = dynamic_cast<Integer*> ( o.rObj());
            Assert (ip); //if rClass() is Integer, then object is integer
            return myInteger * ip->rInteger();
            }
        //if ( o->rClass() == Float::rClass())
        //...
        throw
            ( CObj::WrongParameter ("protocol<Integer>: #* : "
                                    "param int or Integer-object (Float... isn't realized yet) is
expected"));
            }
    else
        return myInteger * params[0].rInt();
    }

SParam Integer::operator/ ( const CParams& params)
    {
    Assert ( params.num == 1);

    if ( params[0].type == SParam::ePObj)
        {
        CObjRef o = params[0].rObj();
        Assert ( o.rObj());
        if ( o->rClass() == Integer::rClass())
            {
            Integer* ip = dynamic_cast<Integer*> ( o.rObj());
            Assert (ip); //if rClass() is Integer, then object is integer
            return myInteger / ip->rInteger();
            }
        //if ( o->rClass() == Float::rClass())
        //...
        throw
            ( CObj::WrongParameter ("protocol<Integer>: #/ : "
                                    "param int or Integer-object (Float... isn't realized yet) is
expected"));
            }
    else
        return myInteger / params[0].rInt();
    }

SParam Integer::operator< ( const CParams& params)
    {
    Assert ( params.num == 1);
```

```
    if ( params[0].type == SParam::ePObj)
        {
        CObjRef o = params[0].rObj();
        Assert ( o.rObj());
        if ( o->rClass() == Integer::rClass())
            {
            Integer* ip = dynamic_cast<Integer*> ( o.rObj());
            Assert (ip); //if rClass() is Integer, then object is integer
            return myInteger < ip->rInteger();
            }
        //if ( o->rClass() == Float::rClass())
        //...
        throw
            ( CObj::WrongParameter ("protocol<Integer>: #< : "
                                    "param int or Integer-object (Float... isn't realized yet) is
expected"));
        }
    else
        return myInteger < params[0].rInt();
    }

SParam Integer::operator<= ( const CParams& params)
    {
    Assert ( params.num == 1);

    if ( params[0].type == SParam::ePObj)
        {
        CObjRef o = params[0].rObj();
        Assert ( o.rObj());
        if ( o->rClass() == Integer::rClass())
            {
            Integer* ip = dynamic_cast<Integer*> ( o.rObj());
            Assert (ip); //if rClass() is Integer, then object is integer
            return myInteger <= ip->rInteger();
            }
        //if ( o->rClass() == Float::rClass())
        //...
        throw
            ( CObj::WrongParameter ("protocol<Integer>: #<= : "
                                    "param int or Integer-object (Float... isn't realized yet) is
expected"));
        }
    else
        return myInteger <= params[0].rInt();
    }

SParam Integer::operator> ( const CParams& params)
    {
    Assert ( params.num == 1);

    if ( params[0].type == SParam::ePObj)
        {
        CObjRef o = params[0].rObj();
        Assert ( o.rObj());
        if ( o->rClass() == Integer::rClass())
            {
            Integer* ip = dynamic_cast<Integer*> ( o.rObj());
            Assert (ip); //if rClass() is Integer, then object is integer
            return myInteger > ip->rInteger();
            }
        //if ( o->rClass() == Float::rClass())
        //...
        throw
            ( CObj::WrongParameter ("protocol<Integer>: #> : "
                                    "param int or Integer-object (Float... isn't realized yet) is
expected"));
        }
    else
        return myInteger > params[0].rInt();
    }

SParam Integer::operator>= ( const CParams& params)
    {
    Assert ( params.num == 1);

        return myInteger >= params[0].rInt();
    }

/*eBool*/SParam Integer::operator== ( const CParams& params) // 1: integer
    {
    Assert ( params.num == 1);
        return myInteger == params[0].rInt();
    }

SParam Integer::betweenAnd ( const CParams& params)
    {
    SParam r1 = this->operator >= ( CParams().add ( params[0]));
    SParam r2 = this->operator <= ( CParams().add ( params[1]));
    bool b1 = r1.type==SParam::ePObj? CSmallVM::Obj2Par ( r1.rObj()).rBool() : r1.rBool();
    bool b2 = r2.type==SParam::ePObj? CSmallVM::Obj2Par ( r2.rObj()).rBool() : r2.rBool();
    return b1 && b2;
    }

SParam Integer::abs ( const CParams& params)
    { return ::abs ( myInteger); }

SParam Integer::asInteger ( const CParams& params)
```

```
    { return myInteger; }
SParam Integer::asFloat ( const CParams& params)
    { return (double) myInteger; }

SParam Integer::asString ( const CParams&)
    {
    char str[256];
    sprintf ( str, "%d", myInteger);
    return new SString ( str);
    }

SParam Integer::repeat ( const CParams& par) // 1 - block
{
    Assert ( par.num == 1 && "Integer::repeat");
//  BlockObj* pBlock = dynamic_cast<BlockObj*> ( par[0].rObj().rObj());
//  Assert ( pBlock && "Integer::repeat");
//  for ( int i = 0; i < myInteger; i++)
//      pBlock->call ( CParams());
    for ( int i = 0; i < myInteger; i++)
        par[0].rObj() ("call")();
    return this;
}

//-------------------------FLOAT-------------------------------------------------------------------------
------------------------

Float::Float() :
    myFloat ( 0L)
    { Assert ( Float::sClass);
      myClass = Float::sClass; }

Float::Float ( double d) :
    myFloat ( d)
    { Assert ( Float::sClass);
      myClass = Float::sClass; }

bool Float::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "Float", "");
    SmallVM::CPPVM->registerBaseClass ( "Float", "Rational");

    SmallVM::CPPVM->registerStaticFunc ( sClass, "construct:", (pStatFunc) sConstruct, ( TYPES
SParam::eFloat));
    SmallVM::CPPVM->registerFunc ( sClass, "+", (pObjFunc) operator+, (TYPES SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "-", (pObjFunc) operator-, (TYPES SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "*", (pObjFunc) operator*, (TYPES SParam::eAny));
    SmallVM::CPPVM->registerFunc ( sClass, "/", (pObjFunc) operator/, (TYPES SParam::eAny));

    return 1;
    }

SParam Float::operator+ ( const CParams& params)
    {
    Assert ( params.num == 1 && "Float::operator+");
    return myFloat + params[0].rFloat();
    }

SParam Float::operator- ( const CParams& params)
    {
    Assert ( params.num == 1 && "Float::operator-");
    return myFloat - params[0].rFloat();
    }

SParam Float::operator* ( const CParams& params)
    {
    Assert ( params.num == 1 && "Float::operator*");
    return myFloat * params[0].rFloat();
    }

SParam Float::operator/ ( const CParams& params)
    {
    Assert ( params.num == 1 && "Float::operator/");
    return myFloat / params[0].rFloat();
    }

SParam Float::operator< ( const CParams& params)
    {
    Assert ( params.num == 1 && "Float::operator<");
    return myFloat < params[0].rFloat();
    }

SParam Float::operator<= ( const CParams& params)
    {
    Assert ( params.num == 1 && "Float::operator<=");
    return myFloat <= params[0].rFloat();
    }

SParam Float::operator> ( const CParams& params)
    {
    Assert ( params.num == 1 && "Float::operator>");
    return myFloat > params[0].rFloat();
    }

SParam Float::operator>= ( const CParams& params)
    {
    Assert ( params.num == 1 && "Float::operator<");
    return myFloat >= params[0].rFloat();
```

```
    }

/*eBool*/SParam Float::operator== ( const CParams& params) // 1: float
    {
    Assert ( params.num == 1);
        return myFloat == params[0].rFloat();
    }

SParam Float::betweenAnd ( const CParams& params)
    {
    Assert ( params.num == 2 && "Float::between:and:");
    SParam r1 = this->operator >= ( CParams().add ( params[0]));
    SParam r2 = this->operator <= ( CParams().add ( params[1]));
    bool b1 = r1.type==SParam::ePObj? CSmallVM::Obj2Par ( r1.rObj()).rBool() : r1.rBool();
    bool b2 = r2.type==SParam::ePObj? CSmallVM::Obj2Par ( r2.rObj()).rBool() : r2.rBool();
    return b1 && b2;
    }

SParam Float::abs ( const CParams& params)
    { return ::fabs ( myFloat); }

SParam Float::asInteger ( const CParams& params)
    { return (long) myFloat; }
SParam Float::asFloat ( const CParams& params)
    { return myFloat; }

SParam Float::asString ( const CParams&)
    {
    char str[256];
    sprintf ( str, "%lg", myFloat);
    return new SString ( str);
    }
```

Файл SmallVM\Standard Smalltalk Library\Stream Protocols.cpp

```
#include "Stream Protocols.h"
#include "Collection Protocols.h"

//---------------------------------------------------------------------------------------------------------

SClassRef PuttableStream::sClass;
SClassRef Transcript::sClass;

//---------------------------------------------------------------------------------------------------------

bool StreamProtocolsReg()
    {
    return PuttableStream::selfReg() && Transcript::selfReg();
    }

//---------------------------------------------------------------------------------------------------------

PuttableStream::PuttableStream()
    { myClass = sClass; }

PuttableStream::~PuttableStream()
    { }

bool PuttableStream::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "puttableStream", "");

    SmallVM::CPPVM->registerFunc ( sClass, "cr", (pObjFunc) cr, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "flush", (pObjFunc) flush, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "space", (pObjFunc) space, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "tab", (pObjFunc) tab, CParamsTypes());
    SmallVM::CPPVM->registerFunc ( sClass, "nextPut:", (pObjFunc) nextPut, (TYPES SParam::ePObj));
    SmallVM::CPPVM->registerFunc ( sClass, "nextPutAll:", (pObjFunc) nextPutAll, (TYPES SParam::ePObj));

    return 1;
    }

//---------------------------------------------------------------------------------------------------------

Transcript::Transcript ()
    { myClass = sClass; }

Transcript::~Transcript()
    { }

bool Transcript::selfReg()
    {
    sClass = SmallVM::CPPVM->registerClass ( "transcript", "");
    SmallVM::CPPVM->registerBaseClass ( "transcript", "puttableStream");

    SmallVM::Linker->getGlobalScope().addVar ( CVar ( CObjRef ( new Transcript()), "Transcript"));

    return 1;
    }

SParam Transcript::cr ( const CParams&)
```

```
    {
    SmallVM::Console->printf ("\n");
    return (CObj*) this;
    }

SParam Transcript::flush ( const CParams&)
    {
    return (CObj*) this;
    }

SParam Transcript::space ( const CParams&)
    {
    printf (" ");
    return (CObj*) this;
    }

SParam Transcript::tab ( const CParams&)
    {
    printf ("\t");
    return (CObj*) this;
    }

SParam Transcript::nextPut ( const CParams& param) //object
    {
    printf ( "%c", param[0].rChar());
    return (CObj*) this;
    }

SParam Transcript::nextPutAll ( const CParams& par) //collection
    {
    if ( par[0].type == SParam::eStr)
        SmallVM::Console->printf ( "%s", par[0].rString());
    else
        SmallVM::Console->printf ( "Transcript #nextPutAll: can't out this collection\n");
    return this;
    }

//-------------------------------------------------------------------------------------------------
```

Файл Sound\Commands.cpp

```
// Commands.cpp - implementation of Commands.h, of module Sound
// created 06.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Sound"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "Commands.h"

#include <Exeption.h>      // use ExeptionEngine

void LoadSoundResource (IConsole* c, int ln, Lexem1* data)
{
    if (ln != 1)
        {
        c->printf ("USAGE: loadsoundresource \"<filename>\"");
        return;
        }
    if (data[0].type != Lexem1::LT_ConstString)
        {
        c->printf ("USAGE: loadsoundresource \"<filename>\"");
        return;
        }

    Sound->LoadResource (data [0].data);
}

void LoadSound (IConsole* c, int ln, Lexem1* data)
{
    if (ln != 2)
        {
        c->printf ("USAGE: loadsound \"<filename>\" \"<soundname>\"");
        return;
        }
    if (data[0].type != Lexem1::LT_ConstString ||
        data[1].type != Lexem1::LT_ConstString)
        {
        c->printf ("USAGE: loadsound \"<filename>\" \"<soundname>\"\n");
        return;
        }

    int a = Sound->AddSound (data [0].data, data [1].data);
    if (a > 0)
        c->printf ("Sound id is %d\n", a);
    else c->printf ("Failed\n");
}

void PlaySound1 (IConsole* c, int ln, Lexem1* data)
{
```

```
    if (ln >= 1)
        {
        if (data [0].type == data[0].LT_ConstString)
            Sound->Play (data[0].data);
        else Sound->Play (data[0].idata);
        }
    else c->printf ("USAGE: playsound <soundname or soundID>\n");
}

void ListSounds (IConsole* c, int, char*)
{
    c->printf ("Listing sound loaded to SoundEngine:\n");
    for (int i = 0; i < Sound->m_soundsSize; i++)
        if (Sound->m_sounds [i])
            printf ("\tSound %d is \"%s\"\n", i, Sound->m_sounds [i]->name);
    printf ("There's total %d sounds.\n", Sound->m_numSounds);
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

void AddCommands (IConsole* console)
{
    Assert (Console);

    console->AddCommand ("loadsound", (CommandProcP) LoadSound, true, "SoundEngine");
    console->AddCommand ("loadsoundresource", (CommandProcP) LoadSoundResource, true, "SoundEngine");
    console->AddCommand ("playsound", (CommandProcP) PlaySound1, true, "SoundEngine");
    console->AddCommand ("listsounds", (CommandProcP) ListSounds, false, "SoundEngine");
}
```

Файл Sound\Module.cpp

```
// Module.cpp - implementation of Module.h, of module Sound
// created 06.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Sound"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "commands.h"
#include <Exeption.h>      // use ExeptionEngine

    #define DLL_NAME "SoundEngine"
    #define DLL_CLASS CSoundEngine
    #include "..\Common\Utilities\Module.c"

DLL_ONLOAD (loader)
{
    Assert (loader);
    Loader = loader;

    Loader->GetInterface (IID_IConsole, (void**)&Console);
    //Loader->GetInterface (IID_ISoundEngine, (void**)&Sound);
    Sound = &DLL_OBJECT;
    if (Console)
        {
        Console->printf ("SOUND: SoundEngine version 1.0 by Vasfed.\n");
        }

        Sound->Init (2);
}

DLL_INIT (loader)
{
    if (!Console)
        Loader->GetInterface (IID_IConsole, (void**)&Console);
    Assert (Console);

    AddCommands (Console);
}
```

Файл Sound\SoundCore.cpp

```
// SoundCore.cpp - implementation of SoundCore.h, of module Sound
// created 06.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Sound"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
#include "SoundCore.h"
#include <stdio.h>
#include <Exeption.h>      // use ExeptionEngine

/*
CSoundEngine::CSoundEngine ()
```

```
{
m_sound = NULL;
m_pMusicManager = NULL;
}

CSoundEngine::~CSoundEngine ()
{

}

IMPLEMENT_IUNKNOWN (CSoundEngine, ISoundEngine, IID_ISoundEngine);

void CSoundEngine::Play (int)
{
//if( pSound ) pSound->Play( DMUS_SEGF_SECONDARY );
    if (m_sound) m_sound->Play (DMUS_SEGF_SECONDARY);
    else Console->printf ("Cannot play\n");
}

void CSoundEngine::Init ()
{
    if (Console)
        Console->printf ("Sound init\n");
    m_pMusicManager = new CMusicManager();
    m_pMusicManager->Initialize (NULL);
    //m_pMusicManager->CreateSegmentFromResource( &m_sound,     (TCHAR*)("BEGINLEVEL"), (TCHAR*)("Wav") );
    m_pMusicManager->CreateSegmentFromFile (&m_sound, "1.wav");
}

void CSoundEngine::Terminate ()
{}
*/
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
IMPLEMENT_IUNKNOWN (CSoundEngine, ISoundEngine, IID_ISoundEngine);

CSoundEngine::CSoundEngine ()
{
    m_sounds = NULL;
    m_soundsSize = 0;
    m_musicManager = NULL;
    m_numSounds = 0;
    m_initialized = false;
}
//-----------------------------------------------------------------------------

CSoundEngine::~CSoundEngine ()
{
    Terminate ();
}
//-----------------------------------------------------------------------------

void CSoundEngine::Terminate ()
{
    m_initialized = false;
    if (m_sounds)
        for (int i = 0; i < m_soundsSize; i++)
            if (m_sounds [i])
                {
                if (m_sounds [i]->segment)
                    m_sounds [i]->segment->Stop ();
                delete m_sounds [i];
                m_sounds [i] = NULL;
                }
    m_sounds = NULL;
    if (m_musicManager)
        delete m_musicManager;

    m_soundsSize = 0;
    m_numSounds = 0;
}
//-----------------------------------------------------------------------------

void CSoundEngine::Reset   ()
{
    Terminate ();
    Init (10);
}
//-----------------------------------------------------------------------------

void CSoundEngine:: Init   (int startsize)
{
    if (m_initialized) return;

    m_numSounds = 0;
    m_musicManager = new CMusicManager ();
    m_musicManager->Initialize (NULL);
    GrowTo (startsize);
    m_initialized = true;
}
//-----------------------------------------------------------------------------

void CSoundEngine::LoadResource           (char* filename)
{
    if (!filename) return;
```

```
    FILE* file = fopen (filename, "rt");
    if (!file)
        {
        if (Console) Console->printf ("SOUND: Can't open resource scrpt file '%s'\n", filename);
        return;
        }

    char str [256];
    fgets (str, sizeof (str), file);
    if (strcmpi (str, "SOUNDENGINE SOUNDMAP\n") != 0)
        {
        if (Console) Console->printf ("SOUND: File %s if not sound resource script.\n", filename);
        return;
        }

    int num = 0;
    if (fscanf (file, "SOUNDNUM = %d\n", &num) != 1)
        return;
    char curdir[256];
    GetCurrentDirectory (256, curdir);
    char dir[256];
    if (fscanf (file, "SOUNDDIR = %s\n", dir))
        SetCurrentDirectory (dir);

    for (int i = 0; i < num; i++)
        {
        fscanf (file, "SOUND\n");
        char f [256];
        char soundname [256];
        int r1 =
        fscanf (file, "NAME = %s\n", soundname),
        r2 = fscanf (file, "FILENAME = %s\n", f);
        fscanf (file, "ENDSOUND\n");

        if (!r1) continue;
        if (!r2) strcpy (soundname, f);
        AddSound (f, soundname);
        }
    SetCurrentDirectory (curdir);
    fclose (file);
}
//---------------------------------------------------------------------------------------
void CSoundEngine::RemoveSound            (int index)
{
    if (index < 0 || index >= m_soundsSize)
        return;

    if (m_sounds [index])
        {
        if (m_sounds [index]->segment) delete m_sounds [index]->segment;
        delete m_sounds [index];
        m_sounds [index] = NULL;
        m_numSounds--;
        }
}
//---------------------------------------------------------------------------------------

int CSoundEngine:: FindSound              (char* name)
{
    if (!name) return -1;

    for (int i = 0; i < m_soundsSize; i++)
        if (m_sounds [i])
            if (!strcmpi (m_sounds [i]->name, name))
                return i;
    return -1;
}
//---------------------------------------------------------------------------------------

int CSoundEngine:: AddSound               (char* filename, char* name)
{
    if (!filename) return -1;

    int place = this->FindFreeSpace ();
    if (place  < 0) return place;

    m_sounds [place] = new SSound;
    if (!m_sounds [place]) return -1;
    try
        {
        strcpy (m_sounds [place]->name, name);
        m_sounds [place]->segment = NULL;
        m_musicManager->CreateSegmentFromFile (&m_sounds [place]->segment, filename);
        if (!m_sounds[place]->segment) throw 1;
        m_numSounds ++;
        }
    catch (...)
        {
        delete m_sounds [place];
        m_sounds [place] = NULL;
        }
    return place;
}
//---------------------------------------------------------------------------------------

int CSoundEngine:: AddSoundFromResource   (void* reserved)
{
```

```
    Throw ("Not implemented."));
    return NULL;
}
//-----------------------------------------------------------------------------

void CSoundEngine:: Play (int index)
{
    if (index < 0 || index >= m_soundsSize) return;

    if (m_sounds [index])
        if (m_sounds [index]->segment)
            m_sounds [index]->segment->Play (DMUS_SEGF_SECONDARY);
}
//-----------------------------------------------------------------------------
void CSoundEngine:: Play (char*name)
{
    return Play (FindSound (name));
}
//-----------------------------------------------------------------------------
void CSoundEngine:: Stop (int index)
{
    if (index < 0 || index >= m_soundsSize) return;

    if (m_sounds [index])
        if (m_sounds [index]->segment)
            m_sounds [index]->segment->Stop ();
}
//-----------------------------------------------------------------------------
void CSoundEngine:: Stop (char*name)
{
    return Stop (FindSound (name));
}
//-----------------------------------------------------------------------------
void CSoundEngine:: Grow       (int)
{
    return GrowFor (1);
}
//-----------------------------------------------------------------------------
void CSoundEngine:: GrowFor    (int n)
{
    return GrowTo (m_soundsSize + n);
}
//-----------------------------------------------------------------------------
void CSoundEngine:: GrowTo     (int n)
{
    if (n <= m_soundsSize) return;

    SSound** temp = new SSound*[n];
    ZeroMemory (temp, sizeof (SSound*) * n);

    if (m_sounds)
        for (int i = 0; i < m_soundsSize; i++)
            temp[i] = m_sounds [i];
    m_sounds = temp;
    m_soundsSize = n;
}
//-----------------------------------------------------------------------------
int CSoundEngine:: FindFreeSpace ()
{
    for (int i = 0; i < m_soundsSize - 1; i++)
        if (!m_sounds [i])
            return i;

    GrowFor (1);
    return i;
}
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
/*EOF*/
```

Файл Sound\System.cpp

```
// System.cpp - implementation of System.h, of module Sound
// created 06.11.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Vasfed"
#define __MODULE_NAME  "Sound"

#include "StdAfx.h"      // use precompiled header
#include "System.h"

#include <Exeption.h>      // use ExeptionEngine

ILoader* Loader = NULL;
CSoundEngine* Sound = NULL;
IConsole* Console = NULL;
```

Файл Sound\dmutil.cpp

```
//-----------------------------------------------------------------------------
// File: DMUtil.cpp
//
// Desc: DirectMusic framework classes for playing DirectMusic segments and
//       DirectMusic scripts. Feel free to use this class as a starting point
//       for adding extra functionality.
//
// Copyright (c) 1999-2001 Microsoft Corp. All rights reserved.
//-----------------------------------------------------------------------------
#define STRICT
#include <dmusicc.h>
#include <dmusici.h>
#include <dsound.h>
#include <dxerr8.h>
#include "DMUtil.h"
#include "DXUtil.h"

//-----------------------------------------------------------------------------
// Name: CMusicManager::CMusicManager()
// Desc: Constructs the class
//-----------------------------------------------------------------------------
CMusicManager::CMusicManager()
{
    m_pLoader       = NULL;
    m_pPerformance  = NULL;

    // Initialize COM
    HRESULT hr = CoInitialize(NULL);
    m_bCleanupCOM = SUCCEEDED(hr);
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::~CMusicManager()
// Desc: Destroys the class
//-----------------------------------------------------------------------------
CMusicManager::~CMusicManager()
{
    SAFE_RELEASE( m_pLoader );

    if( m_pPerformance )
    {
        // If there is any music playing, stop it.
        m_pPerformance->Stop( NULL, NULL, 0, 0 );
        m_pPerformance->CloseDown();

        SAFE_RELEASE( m_pPerformance );
    }

    if( m_bCleanupCOM )
        CoUninitialize();
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::Initialize()
// Desc: Inits DirectMusic using a standard audio path
//-----------------------------------------------------------------------------
HRESULT CMusicManager::Initialize( HWND hWnd, DWORD dwPChannels, DWORD dwDefaultPathType )
{
    HRESULT hr;

    // Create loader object
    if( FAILED( hr = CoCreateInstance( CLSID_DirectMusicLoader, NULL, CLSCTX_INPROC,
                                       IID_IDirectMusicLoader8, (void**)&m_pLoader ) ) )
        return 0;//DXTRACE_ERR( TEXT("CoCreateInstance"), hr );

    // Create performance object
    if( FAILED( hr = CoCreateInstance( CLSID_DirectMusicPerformance, NULL, CLSCTX_INPROC,
                                       IID_IDirectMusicPerformance8, (void**)&m_pPerformance ) ) )
        return 0;//DXTRACE_ERR( TEXT("CoCreateInstance"), hr );

    // Initialize the performance with the standard audio path.
    // This initializes both DirectMusic and DirectSound and
    // sets up the synthesizer. Typcially its easist to use an
    // audio path for playing music and sound effects.
    if( FAILED( hr = m_pPerformance->InitAudio( NULL, NULL, hWnd, dwDefaultPathType,
                                                dwPChannels, DMUS_AUDIOF_ALL, NULL ) ) )
    {
        if( hr == DSERR_NODRIVER )
        {
            //DXTRACE( "Warning: No sound card found\n" );
            return hr;
        }

        return 0;//DXTRACE_ERR( TEXT("InitAudio"), hr );
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::SetSearchDirectory()
```

```
// Desc: Sets the search directory.  If not called, the current working
//       directory is used to load content.
//-----------------------------------------------------------------------------
HRESULT CMusicManager::SetSearchDirectory( const TCHAR* strMediaPath )
{
    if( NULL == m_pLoader )
        return E_UNEXPECTED;

    // DMusic only takes wide strings
    WCHAR wstrMediaPath[MAX_PATH];
    DXUtil_ConvertGenericStringToWide( wstrMediaPath, strMediaPath );

    return m_pLoader->SetSearchDirectory( GUID_DirectMusicAllTypes,
                                          wstrMediaPath, FALSE );
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::GetDefaultAudioPath()
// Desc:
//-----------------------------------------------------------------------------
IDirectMusicAudioPath8* CMusicManager::GetDefaultAudioPath()
{
    IDirectMusicAudioPath8* pAudioPath = NULL;
    if( NULL == m_pPerformance )
        return NULL;

    m_pPerformance->GetDefaultAudioPath( &pAudioPath );
    return pAudioPath;
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::CollectGarbage()
// Desc: Tells the loader to cleanup any garbage from previously
//       released objects.
//-----------------------------------------------------------------------------
VOID CMusicManager::CollectGarbage()
{
    if( m_pLoader )
        m_pLoader->CollectGarbage();
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::CreateSegmentFromFile()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CMusicManager::CreateSegmentFromFile( CMusicSegment** ppSegment,
                                              TCHAR* strFileName,
                                              BOOL bDownloadNow,
                                              BOOL bIsMidiFile )
{
    HRESULT             hr;
    IDirectMusicSegment8* pSegment = NULL;

    // DMusic only takes wide strings
    WCHAR wstrFileName[MAX_PATH];
    DXUtil_ConvertGenericStringToWide( wstrFileName, strFileName );

    if ( FAILED( hr = m_pLoader->LoadObjectFromFile( CLSID_DirectMusicSegment,
                                                     IID_IDirectMusicSegment8,
                                                     wstrFileName,
                                                     (LPVOID*) &pSegment ) ) )
    {
        if( hr == DMUS_E_LOADER_FAILEDOPEN )
            return hr;
        return 0;//DXTRACE_ERR( TEXT("LoadObjectFromFile"), hr );
    }

    *ppSegment = new CMusicSegment( m_pPerformance, m_pLoader, pSegment );
    if (!*ppSegment)
        return E_OUTOFMEMORY;

    if( bIsMidiFile )
    {
        if( FAILED( hr = pSegment->SetParam( GUID_StandardMIDIFile,
                                             0xFFFFFFFF, 0, 0, NULL ) ) )
            return 0;//DXTRACE_ERR( TEXT("SetParam"), hr );
    }

    if( bDownloadNow )
    {
        if( FAILED( hr = (*ppSegment)->Download() ) )
            return 0;//DXTRACE_ERR( TEXT("Download"), hr );
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::CreateSegmentFromResource()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CMusicManager::CreateSegmentFromResource( CMusicSegment** ppSegment,
                                                  TCHAR* strResource,
                                                  TCHAR* strResourceType,
                                                  BOOL bDownloadNow,
                                                  BOOL bIsMidiFile )
{
```

```
    HRESULT              hr;
    IDirectMusicSegment8* pSegment      = NULL;
    HRSRC                hres           = NULL;
    void*                pMem           = NULL;
    DWORD                dwSize         = 0;
    DMUS_OBJECTDESC      objdesc;

    // Find the resource
    hres = FindResource( NULL,strResource,strResourceType );
    if( NULL == hres )
        return E_FAIL;

    // Load the resource
    pMem = (void*)LoadResource( NULL, hres );
    if( NULL == pMem )
        return E_FAIL;

    // Store the size of the resource
    dwSize = SizeofResource( NULL, hres );

    // Set up our object description
    ZeroMemory(&objdesc,sizeof(DMUS_OBJECTDESC));
    objdesc.dwSize = sizeof(DMUS_OBJECTDESC);
    objdesc.dwValidData = DMUS_OBJ_MEMORY | DMUS_OBJ_CLASS;
    objdesc.guidClass = CLSID_DirectMusicSegment;
    objdesc.llMemLength =(LONGLONG)dwSize;
    objdesc.pbMemData = (BYTE*)pMem;

    if (FAILED ( hr = m_pLoader->GetObject( &objdesc,
                                            IID_IDirectMusicSegment8,
                                            (void**)&pSegment ) ) )
    {
        if( hr == DMUS_E_LOADER_FAILEDOPEN )
            return hr;
        return 0;//DXTRACE_ERR( TEXT("LoadObjectFromFile"), hr );
    }

    *ppSegment = new CMusicSegment( m_pPerformance, m_pLoader, pSegment );
    if( NULL == *ppSegment )
        return E_OUTOFMEMORY;

    if( bIsMidiFile )
    {
        // Do this to make sure that the default General MIDI set
        // is connected appropriately to the MIDI file and
        // all instruments sound correct.
        if( FAILED( hr = pSegment->SetParam( GUID_StandardMIDIFile,
                                             0xFFFFFFFF, 0, 0, NULL ) ) )
            return 0;//DXTRACE_ERR( TEXT("SetParam"), hr );
    }

    if( bDownloadNow )
    {
        // The segment needs to be download first before playing.
        // However, some apps may want to wait before calling this
        // to because the download allocates memory for the
        // instruments. The more instruments currently downloaded,
        // the more memory is in use by the synthesizer.
        if( FAILED( hr = (*ppSegment)->Download() ) )
            return 0;//DXTRACE_ERR( TEXT("Download"), hr );
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::CreateScriptFromFile()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CMusicManager::CreateScriptFromFile( CMusicScript** ppScript,
                                             TCHAR* strFileName )
{
    HRESULT              hr;
    IDirectMusicScript* pScript = NULL;

    // DMusic only takes wide strings
    WCHAR wstrFileName[MAX_PATH];
    DXUtil_ConvertGenericStringToWide( wstrFileName, strFileName );

    if ( FAILED( hr = m_pLoader->LoadObjectFromFile( CLSID_DirectMusicScript,
                                                     IID_IDirectMusicScript8,
                                                     wstrFileName,
                                                     (LPVOID*) &pScript ) ) )
        return 0;//DXTRACE_ERR_NOMSGBOX( TEXT("LoadObjectFromFile"), hr );

    if ( FAILED( hr = pScript->Init( m_pPerformance, NULL ) ) )
        return 0;//DXTRACE_ERR( TEXT("Init"), hr );

    *ppScript = new CMusicScript( m_pPerformance, m_pLoader, pScript );
    if (!*ppScript)
        return E_OUTOFMEMORY;

    return hr;
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::CreateChordMapFromFile()
```

```
// Desc:
//-----------------------------------------------------------------------------
HRESULT CMusicManager::CreateChordMapFromFile( IDirectMusicChordMap8** ppChordMap,
                                               TCHAR* strFileName )
{
    // DMusic only takes wide strings
    WCHAR wstrFileName[MAX_PATH];
    DXUtil_ConvertGenericStringToWide( wstrFileName, strFileName );

    return m_pLoader->LoadObjectFromFile( CLSID_DirectMusicChordMap,
                                          IID_IDirectMusicChordMap8,
                                          wstrFileName, (LPVOID*) ppChordMap );
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::CreateChordMapFromFile()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CMusicManager::CreateStyleFromFile( IDirectMusicStyle8** ppStyle,
                                            TCHAR* strFileName )
{
    // DMusic only takes wide strings
    WCHAR wstrFileName[MAX_PATH];
    DXUtil_ConvertGenericStringToWide( wstrFileName, strFileName );

    return m_pLoader->LoadObjectFromFile( CLSID_DirectMusicStyle,
                                          IID_IDirectMusicStyle8,
                                          wstrFileName, (LPVOID*) ppStyle );
}

//-----------------------------------------------------------------------------
// Name: CMusicManager::GetMotifFromStyle()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CMusicManager::GetMotifFromStyle( IDirectMusicSegment8** ppMotif8,
                                          TCHAR* strStyle, TCHAR* strMotif )
{
    HRESULT             hr;
    IDirectMusicStyle8*  pStyle = NULL;
    IDirectMusicSegment* pMotif = NULL;

    if( FAILED( hr = CreateStyleFromFile( &pStyle, strStyle ) ) )
        return 0;//DXTRACE_ERR( TEXT("CreateStyleFromFile"), hr );

    if( pStyle )
    {
        // DMusic only takes wide strings
        WCHAR wstrMotif[MAX_PATH];
        DXUtil_ConvertGenericStringToWide( wstrMotif, strMotif );

        hr = pStyle->GetMotif( wstrMotif, &pMotif );
        SAFE_RELEASE( pStyle );

        if( FAILED( hr ) )
            return 0;//DXTRACE_ERR( TEXT("GetMotif"), hr );

        pMotif->QueryInterface( IID_IDirectMusicSegment8, (LPVOID*) ppMotif8 );
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CMusicSegment::CMusicSegment()
// Desc: Constructs the class
//-----------------------------------------------------------------------------
CMusicSegment::CMusicSegment( IDirectMusicPerformance8* pPerformance,
                              IDirectMusicLoader8*      pLoader,
                              IDirectMusicSegment8*     pSegment )
{
    m_pPerformance          = pPerformance;
    m_pLoader               = pLoader;
    m_pSegment              = pSegment;
    m_pEmbeddedAudioPath    = NULL;
    m_bDownloaded           = FALSE;

    // Try to pull out an audio path from the segment itself if there is one.
    // This embedded audio path will be used instead of the default
    // audio path if the app doesn't wish to use an overriding audio path.
    IUnknown* pConfig = NULL;
    if( SUCCEEDED( m_pSegment->GetAudioPathConfig( &pConfig ) ) )
    {
        m_pPerformance->CreateAudioPath( pConfig, TRUE, &m_pEmbeddedAudioPath );
        SAFE_RELEASE( pConfig );
    }

}

//-----------------------------------------------------------------------------
// Name: CMusicSegment::~CMusicSegment()
// Desc: Destroys the class
//-----------------------------------------------------------------------------
CMusicSegment::~CMusicSegment()
{
    if( m_pSegment )
    {
        // Tell the loader that this object should now be released
```

```
        if( m_pLoader )
            m_pLoader->ReleaseObjectByUnknown( m_pSegment );

        if( m_bDownloaded )
        {
            if( m_pEmbeddedAudioPath )
                m_pSegment->Unload( m_pEmbeddedAudioPath );
            else
                m_pSegment->Unload( m_pPerformance );
        }

        SAFE_RELEASE( m_pEmbeddedAudioPath );
        SAFE_RELEASE( m_pSegment );
    }

    m_pPerformance = NULL;
}

//-----------------------------------------------------------------------------
// Name: CMusicSegment::Play()
// Desc: Plays the sound using voice management flags.  Pass in DSBPLAY_LOOPING
//       in the dwFlags to loop the sound
//-----------------------------------------------------------------------------
HRESULT CMusicSegment::Play( DWORD dwFlags, IDirectMusicAudioPath8* pAudioPath )
{
    if( m_pSegment == NULL || m_pPerformance == NULL )
        return CO_E_NOTINITIALIZED;

    if( !m_bDownloaded )
        return E_FAIL;

    // If an audio path was passed in then use it, otherwise
    // use the embedded audio path if there was one.
    if( pAudioPath == NULL && m_pEmbeddedAudioPath != NULL )
        pAudioPath = m_pEmbeddedAudioPath;

    // If pAudioPath is NULL then this plays on the default audio path.
    return m_pPerformance->PlaySegmentEx( m_pSegment, 0, NULL, dwFlags,
                                          0, 0, NULL, pAudioPath );
}

//-----------------------------------------------------------------------------
// Name: CMusicSegment::Download()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CMusicSegment::Download( IDirectMusicAudioPath8* pAudioPath )
{
    HRESULT hr;

    if( m_pSegment == NULL )
        return CO_E_NOTINITIALIZED;

    // If no audio path was passed in, then download
    // to the embedded audio path if it exists
    // else download to the performance
    if( pAudioPath == NULL )
    {
        if( m_pEmbeddedAudioPath )
            hr = m_pSegment->Download( m_pEmbeddedAudioPath );
        else
            hr = m_pSegment->Download( m_pPerformance );
    }
    else
    {
        hr = m_pSegment->Download( pAudioPath );
    }

    if ( SUCCEEDED( hr ) )
        m_bDownloaded = TRUE;

    return hr;
}

//-----------------------------------------------------------------------------
// Name: CMusicSegment::Unload()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CMusicSegment::Unload( IDirectMusicAudioPath8* pAudioPath )
{
    HRESULT hr;

    if( m_pSegment == NULL )
        return CO_E_NOTINITIALIZED;

    // If no audio path was passed in, then unload
    // from the embedded audio path if it exists
    // else unload from the performance
    if( pAudioPath == NULL )
    {
        if( m_pEmbeddedAudioPath )
            hr = m_pSegment->Unload( m_pEmbeddedAudioPath );
        else
            hr = m_pSegment->Unload( m_pPerformance );
    }
    else
    {
        hr = m_pSegment->Unload( pAudioPath );
```

```
    }

    if ( SUCCEEDED( hr ) )
        m_bDownloaded = FALSE;

    return hr;
}

//-----------------------------------------------------------------------------
// Name: CMusicSegment::IsPlaying()
// Desc:
//-----------------------------------------------------------------------------
BOOL CMusicSegment::IsPlaying()
{
    if( m_pSegment == NULL || m_pPerformance == NULL )
        return CO_E_NOTINITIALIZED;

    return ( m_pPerformance->IsPlaying( m_pSegment, NULL ) == S_OK );
}

//-----------------------------------------------------------------------------
// Name: CMusicSegment::Stop()
// Desc: Stops the sound from playing
//-----------------------------------------------------------------------------
HRESULT CMusicSegment::Stop( DWORD dwFlags )
{
    if( m_pSegment == NULL || m_pPerformance == NULL )
        return CO_E_NOTINITIALIZED;

    return m_pPerformance->Stop( m_pSegment, NULL, 0, dwFlags );;
}

//-----------------------------------------------------------------------------
// Name: CMusicSegment::SetRepeats()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CMusicSegment::SetRepeats( DWORD dwRepeats )
{
    if( m_pSegment == NULL )
        return CO_E_NOTINITIALIZED;

    return m_pSegment->SetRepeats( dwRepeats );
}

//-----------------------------------------------------------------------------
// Name: CMusicSegment::GetStyle()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CMusicSegment::GetStyle( IDirectMusicStyle8** ppStyle, DWORD dwStyleIndex )
{
    // Get the Style from the Segment by calling the Segment's GetData() with
    // the data type GUID_StyleTrackStyle. 0xffffffff indicates to look at
    // tracks in all TrackGroups in the segment. The first 0 indicates to
    // retrieve the Style from the first Track  in the indicated TrackGroup.
    // The second 0 indicates to retrieve the Style from the beginning of the
    // segment, i.e. time 0 in Segment time. If this Segment was loaded from a
    // section file, there is only one Style and it is at time 0.
    return m_pSegment->GetParam( GUID_IDirectMusicStyle, 0xffffffff, dwStyleIndex,
                                 0, NULL, (VOID*)ppStyle );
}

//-----------------------------------------------------------------------------
// Name: CMusicScript::CMusicScript()
// Desc: Constructs the class
//-----------------------------------------------------------------------------
CMusicScript::CMusicScript( IDirectMusicPerformance8* pPerformance,
                            IDirectMusicLoader8* pLoader,
                            IDirectMusicScript8* pScript )
{
    m_pPerformance = pPerformance;
    m_pLoader      = pLoader;
    m_pScript      = pScript;
}

//-----------------------------------------------------------------------------
// Name: CMusicScript::~CMusicScript()
// Desc: Destroys the class
//-----------------------------------------------------------------------------
CMusicScript::~CMusicScript()
{
    if( m_pLoader )
    {
        // Tell the loader that this object should now be released
        m_pLoader->ReleaseObjectByUnknown( m_pScript );
        m_pLoader = NULL;
    }

    SAFE_RELEASE( m_pScript );
    m_pPerformance = NULL;
}

//-----------------------------------------------------------------------------
// Name: CMusicScript::Play()
// Desc: Calls a routine in the script
//-----------------------------------------------------------------------------
HRESULT CMusicScript::CallRoutine( TCHAR* strRoutine )
{
```

```
    // DMusic only takes wide strings
    WCHAR wstrRoutine[MAX_PATH];
    DXUtil_ConvertGenericStringToWide( wstrRoutine, strRoutine );

    return m_pScript->CallRoutine( wstrRoutine, NULL );
}

//-----------------------------------------------------------------------------
// Name: CMusicScript::SetVariableNumber()
// Desc: Sets the value of a variable in the script
//-----------------------------------------------------------------------------
HRESULT CMusicScript::SetVariableNumber( TCHAR* strVariable, LONG lValue )
{
    // DMusic only takes wide strings
    WCHAR wstrVariable[MAX_PATH];
    DXUtil_ConvertGenericStringToWide( wstrVariable, strVariable );

    return m_pScript->SetVariableNumber( wstrVariable, lValue, NULL );
}

//-----------------------------------------------------------------------------
// Name: CMusicScript::GetVariableNumber()
// Desc: Gets the value of a variable in the script
//-----------------------------------------------------------------------------
HRESULT CMusicScript::GetVariableNumber( TCHAR* strVariable, LONG* plValue )
{
    // DMusic only takes wide strings
    WCHAR wstrVariable[MAX_PATH];
    DXUtil_ConvertGenericStringToWide( wstrVariable, strVariable );

    return m_pScript->GetVariableNumber( wstrVariable, plValue, NULL );
}
```

Файл Sound\dsutil.cpp

```
//-----------------------------------------------------------------------------
// File: DSUtil.cpp
//
// Desc: DirectSound framework classes for reading and writing wav files and
//       playing them in DirectSound buffers. Feel free to use this class
//       as a starting point for adding extra functionality.
//
// Copyright (c) 1999-2001 Microsoft Corp. All rights reserved.
//-----------------------------------------------------------------------------
#define STRICT
#include <windows.h>
#include <mmsystem.h>
#include <dxerr8.h>
#include <dsound.h>
#include "DSUtil.h"
#include "DXUtil.h"

//-----------------------------------------------------------------------------
// Name: CSoundManager::CSoundManager()
// Desc: Constructs the class
//-----------------------------------------------------------------------------
CSoundManager::CSoundManager()
{
    m_pDS = NULL;
}

//-----------------------------------------------------------------------------
// Name: CSoundManager::~CSoundManager()
// Desc: Destroys the class
//-----------------------------------------------------------------------------
CSoundManager::~CSoundManager()
{
    SAFE_RELEASE( m_pDS );
}

//-----------------------------------------------------------------------------
// Name: CSoundManager::Initialize()
// Desc: Initializes the IDirectSound object and also sets the primary buffer
//       format.  This function must be called before any others.
//-----------------------------------------------------------------------------
HRESULT CSoundManager::Initialize( HWND  hWnd,
                                   DWORD dwCoopLevel,
                                   DWORD dwPrimaryChannels,
                                   DWORD dwPrimaryFreq,
                                   DWORD dwPrimaryBitRate )
{
    HRESULT             hr;

    SAFE_RELEASE( m_pDS );

    // Create IDirectSound using the primary sound device
    if( FAILED( hr = DirectSoundCreate8( NULL, &m_pDS, NULL ) ) )
        return DXTRACE_ERR( TEXT("DirectSoundCreate8"), hr );

    // Set DirectSound coop level
```

```
    if( FAILED( hr = m_pDS->SetCooperativeLevel( hWnd, dwCoopLevel ) ) )
        return DXTRACE_ERR( TEXT("SetCooperativeLevel"), hr );

    // Set primary buffer format
    if( FAILED( hr = SetPrimaryBufferFormat( dwPrimaryChannels, dwPrimaryFreq, dwPrimaryBitRate ) ) )
        return DXTRACE_ERR( TEXT("SetPrimaryBufferFormat"), hr );

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CSoundManager::SetPrimaryBufferFormat()
// Desc: Set primary buffer to a specified format
//       For example, to set the primary buffer format to 22kHz stereo, 16-bit
//       then:   dwPrimaryChannels = 2
//               dwPrimaryFreq     = 22050,
//               dwPrimaryBitRate  = 16
//-----------------------------------------------------------------------------
HRESULT CSoundManager::SetPrimaryBufferFormat( DWORD dwPrimaryChannels,
                                               DWORD dwPrimaryFreq,
                                               DWORD dwPrimaryBitRate )
{
    HRESULT             hr;
    LPDIRECTSOUNDBUFFER pDSBPrimary = NULL;

    if( m_pDS == NULL )
        return CO_E_NOTINITIALIZED;

    // Get the primary buffer
    DSBUFFERDESC dsbd;
    ZeroMemory( &dsbd, sizeof(DSBUFFERDESC) );
    dsbd.dwSize        = sizeof(DSBUFFERDESC);
    dsbd.dwFlags       = DSBCAPS_PRIMARYBUFFER;
    dsbd.dwBufferBytes = 0;
    dsbd.lpwfxFormat   = NULL;

    if( FAILED( hr = m_pDS->CreateSoundBuffer( &dsbd, &pDSBPrimary, NULL ) ) )
        return DXTRACE_ERR( TEXT("CreateSoundBuffer"), hr );

    WAVEFORMATEX wfx;
    ZeroMemory( &wfx, sizeof(WAVEFORMATEX) );
    wfx.wFormatTag      = WAVE_FORMAT_PCM;
    wfx.nChannels       = (WORD) dwPrimaryChannels;
    wfx.nSamplesPerSec  = dwPrimaryFreq;
    wfx.wBitsPerSample  = (WORD) dwPrimaryBitRate;
    wfx.nBlockAlign     = (WORD) (wfx.wBitsPerSample / 8 * wfx.nChannels);
    wfx.nAvgBytesPerSec = wfx.nSamplesPerSec * wfx.nBlockAlign;

    if( FAILED( hr = pDSBPrimary->SetFormat(&wfx) ) )
        return DXTRACE_ERR( TEXT("SetFormat"), hr );

    SAFE_RELEASE( pDSBPrimary );

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CSoundManager::Get3DListenerInterface()
// Desc: Returns the 3D listener interface associated with primary buffer.
//-----------------------------------------------------------------------------
HRESULT CSoundManager::Get3DListenerInterface( LPDIRECTSOUND3DLISTENER* ppDSListener )
{
    HRESULT             hr;
    DSBUFFERDESC        dsbdesc;
    LPDIRECTSOUNDBUFFER pDSBPrimary = NULL;

    if( ppDSListener == NULL )
        return E_INVALIDARG;
    if( m_pDS == NULL )
        return CO_E_NOTINITIALIZED;

    *ppDSListener = NULL;

    // Obtain primary buffer, asking it for 3D control
    ZeroMemory( &dsbdesc, sizeof(DSBUFFERDESC) );
    dsbdesc.dwSize = sizeof(DSBUFFERDESC);
    dsbdesc.dwFlags = DSBCAPS_CTRL3D | DSBCAPS_PRIMARYBUFFER;
    if( FAILED( hr = m_pDS->CreateSoundBuffer( &dsbdesc, &pDSBPrimary, NULL ) ) )
        return DXTRACE_ERR( TEXT("CreateSoundBuffer"), hr );

    if( FAILED( hr = pDSBPrimary->QueryInterface( IID_IDirectSound3DListener,
                                                  (VOID**)ppDSListener ) ) )
    {
        SAFE_RELEASE( pDSBPrimary );
        return DXTRACE_ERR( TEXT("QueryInterface"), hr );
    }

    // Release the primary buffer, since it is not need anymore
    SAFE_RELEASE( pDSBPrimary );

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CSoundManager::Create()
// Desc:
//-----------------------------------------------------------------------------
```

```
HRESULT CSoundManager::Create( CSound** ppSound,
                               LPTSTR strWaveFileName,
                               DWORD dwCreationFlags,
                               GUID guid3DAlgorithm,
                               DWORD dwNumBuffers )
{
    HRESULT hr;
    HRESULT hrRet = S_OK;
    DWORD   i;
    LPDIRECTSOUNDBUFFER* apDSBuffer     = NULL;
    DWORD                dwDSBufferSize = NULL;
    CWaveFile*           pWaveFile      = NULL;

    if( m_pDS == NULL )
        return CO_E_NOTINITIALIZED;
    if( strWaveFileName == NULL || ppSound == NULL || dwNumBuffers < 1 )
        return E_INVALIDARG;

    apDSBuffer = new LPDIRECTSOUNDBUFFER[dwNumBuffers];
    if( apDSBuffer == NULL )
    {
        hr = E_OUTOFMEMORY;
        goto LFail;
    }

    pWaveFile = new CWaveFile();
    if( pWaveFile == NULL )
    {
        hr = E_OUTOFMEMORY;
        goto LFail;
    }

    pWaveFile->Open( strWaveFileName, NULL, WAVEFILE_READ );

    if( pWaveFile->GetSize() == 0 )
    {
        // Wave is blank, so don't create it.
        hr = E_FAIL;
        goto LFail;
    }

    // Make the DirectSound buffer the same size as the wav file
    dwDSBufferSize = pWaveFile->GetSize();

    // Create the direct sound buffer, and only request the flags needed
    // since each requires some overhead and limits if the buffer can
    // be hardware accelerated
    DSBUFFERDESC dsbd;
    ZeroMemory( &dsbd, sizeof(DSBUFFERDESC) );
    dsbd.dwSize          = sizeof(DSBUFFERDESC);
    dsbd.dwFlags         = dwCreationFlags;
    dsbd.dwBufferBytes   = dwDSBufferSize;
    dsbd.guid3DAlgorithm = guid3DAlgorithm;
    dsbd.lpwfxFormat     = pWaveFile->m_pwfx;

    // DirectSound is only guarenteed to play PCM data.  Other
    // formats may or may not work depending the sound card driver.
    hr = m_pDS->CreateSoundBuffer( &dsbd, &apDSBuffer[0], NULL );

    // Be sure to return this error code if it occurs so the
    // callers knows this happened.
    if( hr == DS_NO_VIRTUALIZATION )
        hrRet = DS_NO_VIRTUALIZATION;

    if( FAILED(hr) )
    {
        // DSERR_BUFFERTOOSMALL will be returned if the buffer is
        // less than DSBSIZE_FX_MIN (100ms) and the buffer is created
        // with DSBCAPS_CTRLFX.
        if( hr != DSERR_BUFFERTOOSMALL )
            DXTRACE_ERR( TEXT("CreateSoundBuffer"), hr );

        goto LFail;
    }

    for( i=1; i<dwNumBuffers; i++ )
    {
        if( FAILED( hr = m_pDS->DuplicateSoundBuffer( apDSBuffer[0], &apDSBuffer[i] ) ) )
        {
            DXTRACE_ERR( TEXT("DuplicateSoundBuffer"), hr );
            goto LFail;
        }
    }

    // Create the sound
    *ppSound = new CSound( apDSBuffer, dwDSBufferSize, dwNumBuffers, pWaveFile );

    SAFE_DELETE( apDSBuffer );
    return hrRet;

LFail:
    // Cleanup
    SAFE_DELETE( pWaveFile );
    SAFE_DELETE( apDSBuffer );
    return hr;
}
```

```
//-----------------------------------------------------------------------------
// Name: CSoundManager::CreateFromMemory()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CSoundManager::CreateFromMemory( CSound** ppSound,
                                         BYTE* pbData,
                                         ULONG  ulDataSize,
                                         LPWAVEFORMATEX pwfx,
                                         DWORD dwCreationFlags,
                                         GUID guid3DAlgorithm,
                                         DWORD dwNumBuffers )
{
    HRESULT hr;
    DWORD   i;
    LPDIRECTSOUNDBUFFER* apDSBuffer     = NULL;
    DWORD                dwDSBufferSize = NULL;
    CWaveFile*           pWaveFile      = NULL;

    if( m_pDS == NULL )
        return CO_E_NOTINITIALIZED;
    if( pbData == NULL || ppSound == NULL || dwNumBuffers < 1 )
        return E_INVALIDARG;

    apDSBuffer = new LPDIRECTSOUNDBUFFER[dwNumBuffers];
    if( apDSBuffer == NULL )
    {
        hr = E_OUTOFMEMORY;
        goto LFail;
    }

    pWaveFile = new CWaveFile();
    if( pWaveFile == NULL )
    {
        hr = E_OUTOFMEMORY;
        goto LFail;
    }

    pWaveFile->OpenFromMemory( pbData,ulDataSize, pwfx, WAVEFILE_READ );

    // Make the DirectSound buffer the same size as the wav file
    dwDSBufferSize = ulDataSize;

    // Create the direct sound buffer, and only request the flags needed
    // since each requires some overhead and limits if the buffer can
    // be hardware accelerated
    DSBUFFERDESC dsbd;
    ZeroMemory( &dsbd, sizeof(DSBUFFERDESC) );
    dsbd.dwSize          = sizeof(DSBUFFERDESC);
    dsbd.dwFlags         = dwCreationFlags;
    dsbd.dwBufferBytes   = dwDSBufferSize;
    dsbd.guid3DAlgorithm = guid3DAlgorithm;
    dsbd.lpwfxFormat     = pwfx;

    if( FAILED( hr = m_pDS->CreateSoundBuffer( &dsbd, &apDSBuffer[0], NULL ) ) )
    {
        DXTRACE_ERR( TEXT("CreateSoundBuffer"), hr );
        goto LFail;
    }

    for( i=1; i<dwNumBuffers; i++ )
    {
        if( FAILED( hr = m_pDS->DuplicateSoundBuffer( apDSBuffer[0], &apDSBuffer[i] ) ) )
        {
            DXTRACE_ERR( TEXT("DuplicateSoundBuffer"), hr );
            goto LFail;
        }
    }

    // Create the sound
    *ppSound = new CSound( apDSBuffer, dwDSBufferSize, dwNumBuffers, pWaveFile );

    SAFE_DELETE( apDSBuffer );
    return S_OK;

LFail:
    // Cleanup

    SAFE_DELETE( apDSBuffer );
    return hr;
}

//-----------------------------------------------------------------------------
// Name: CSoundManager::CreateStreaming()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CSoundManager::CreateStreaming( CStreamingSound** ppStreamingSound,
                                        LPTSTR strWaveFileName,
                                        DWORD dwCreationFlags,
                                        GUID guid3DAlgorithm,
                                        DWORD dwNotifyCount,
                                        DWORD dwNotifySize,
                                        HANDLE hNotifyEvent )
{
    HRESULT hr;

    if( m_pDS == NULL )
        return CO_E_NOTINITIALIZED;
```

```
    if( strWaveFileName == NULL || ppStreamingSound == NULL || hNotifyEvent == NULL )
        return E_INVALIDARG;

    LPDIRECTSOUNDBUFFER pDSBuffer      = NULL;
    DWORD               dwDSBufferSize = NULL;
    CWaveFile*          pWaveFile      = NULL;
    DSBPOSITIONNOTIFY*  aPosNotify     = NULL;
    LPDIRECTSOUNDNOTIFY pDSNotify      = NULL;

    pWaveFile = new CWaveFile();
    pWaveFile->Open( strWaveFileName, NULL, WAVEFILE_READ );

    // Figure out how big the DSound buffer should be
    dwDSBufferSize = dwNotifySize * dwNotifyCount;

    // Set up the direct sound buffer.  Request the NOTIFY flag, so
    // that we are notified as the sound buffer plays.  Note, that using this flag
    // may limit the amount of hardware acceleration that can occur.
    DSBUFFERDESC dsbd;
    ZeroMemory( &dsbd, sizeof(DSBUFFERDESC) );
    dsbd.dwSize          = sizeof(DSBUFFERDESC);
    dsbd.dwFlags         = dwCreationFlags |
                           DSBCAPS_CTRLPOSITIONNOTIFY |
                           DSBCAPS_GETCURRENTPOSITION2;
    dsbd.dwBufferBytes   = dwDSBufferSize;
    dsbd.guid3DAlgorithm = guid3DAlgorithm;
    dsbd.lpwfxFormat     = pWaveFile->m_pwfx;

    if( FAILED( hr = m_pDS->CreateSoundBuffer( &dsbd, &pDSBuffer, NULL ) ) )
    {
        // If wave format isn't then it will return
        // either DSERR_BADFORMAT or E_INVALIDARG
        if( hr == DSERR_BADFORMAT || hr == E_INVALIDARG )
            return DXTRACE_ERR_NOMSGBOX( TEXT("CreateSoundBuffer"), hr );

        return DXTRACE_ERR( TEXT("CreateSoundBuffer"), hr );
    }

    // Create the notification events, so that we know when to fill
    // the buffer as the sound plays.
    if( FAILED( hr = pDSBuffer->QueryInterface( IID_IDirectSoundNotify,
                                                (VOID**)&pDSNotify ) ) )
    {
        SAFE_DELETE( aPosNotify );
        return DXTRACE_ERR( TEXT("QueryInterface"), hr );
    }

    aPosNotify = new DSBPOSITIONNOTIFY[ dwNotifyCount ];
    if( aPosNotify == NULL )
        return E_OUTOFMEMORY;

    for( DWORD i = 0; i < dwNotifyCount; i++ )
    {
        aPosNotify[i].dwOffset     = (dwNotifySize * i) + dwNotifySize - 1;
        aPosNotify[i].hEventNotify = hNotifyEvent;
    }

    // Tell DirectSound when to notify us. The notification will come in the from
    // of signaled events that are handled in WinMain()
    if( FAILED( hr = pDSNotify->SetNotificationPositions( dwNotifyCount,
                                                          aPosNotify ) ) )
    {
        SAFE_RELEASE( pDSNotify );
        SAFE_DELETE( aPosNotify );
        return DXTRACE_ERR( TEXT("SetNotificationPositions"), hr );
    }

    SAFE_RELEASE( pDSNotify );
    SAFE_DELETE( aPosNotify );

    // Create the sound
    *ppStreamingSound = new CStreamingSound( pDSBuffer, dwDSBufferSize, pWaveFile, dwNotifySize );

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CSound::CSound()
// Desc: Constructs the class
//-----------------------------------------------------------------------------
CSound::CSound( LPDIRECTSOUNDBUFFER* apDSBuffer, DWORD dwDSBufferSize,
                DWORD dwNumBuffers, CWaveFile* pWaveFile )
{
    DWORD i;

    m_apDSBuffer = new LPDIRECTSOUNDBUFFER[dwNumBuffers];
    for( i=0; i<dwNumBuffers; i++ )
        m_apDSBuffer[i] = apDSBuffer[i];

    m_dwDSBufferSize = dwDSBufferSize;
    m_dwNumBuffers   = dwNumBuffers;
    m_pWaveFile      = pWaveFile;

    FillBufferWithSound( m_apDSBuffer[0], FALSE );

    // Make DirectSound do pre-processing on sound effects
    for( i=0; i<dwNumBuffers; i++ )
```

```
        m_apDSBuffer[i]->SetCurrentPosition(0);
}

//-----------------------------------------------------------------------------
// Name: CSound::~CSound()
// Desc: Destroys the class
//-----------------------------------------------------------------------------
CSound::~CSound()
{
    for( DWORD i=0; i<m_dwNumBuffers; i++ )
    {
        SAFE_RELEASE( m_apDSBuffer[i] );
    }

    SAFE_DELETE_ARRAY( m_apDSBuffer );
    SAFE_DELETE( m_pWaveFile );
}

//-----------------------------------------------------------------------------
// Name: CSound::FillBufferWithSound()
// Desc: Fills a DirectSound buffer with a sound file
//-----------------------------------------------------------------------------
HRESULT CSound::FillBufferWithSound( LPDIRECTSOUNDBUFFER pDSB, BOOL bRepeatWavIfBufferLarger )
{
    HRESULT hr;
    VOID*   pDSLockedBuffer      = NULL; // Pointer to locked buffer memory
    DWORD   dwDSLockedBufferSize = 0;    // Size of the locked DirectSound buffer
    DWORD   dwWavDataRead        = 0;    // Amount of data read from the wav file

    if( pDSB == NULL )
        return CO_E_NOTINITIALIZED;

    // Make sure we have focus, and we didn't just switch in from
    // an app which had a DirectSound device
    if( FAILED( hr = RestoreBuffer( pDSB, NULL ) ) )
        return DXTRACE_ERR( TEXT("RestoreBuffer"), hr );

    // Lock the buffer down
    if( FAILED( hr = pDSB->Lock( 0, m_dwDSBufferSize,
                                 &pDSLockedBuffer, &dwDSLockedBufferSize,
                                 NULL, NULL, 0L ) ) )
        return DXTRACE_ERR( TEXT("Lock"), hr );

    // Reset the wave file to the beginning
    m_pWaveFile->ResetFile();

    if( FAILED( hr = m_pWaveFile->Read( (BYTE*) pDSLockedBuffer,
                                        dwDSLockedBufferSize,
                                        &dwWavDataRead ) ) )
        return DXTRACE_ERR( TEXT("Read"), hr );

    if( dwWavDataRead == 0 )
    {
        // Wav is blank, so just fill with silence
        FillMemory( (BYTE*) pDSLockedBuffer,
                    dwDSLockedBufferSize,
                    (BYTE)(m_pWaveFile->m_pwfx->wBitsPerSample == 8 ? 128 : 0 ) );
    }
    else if( dwWavDataRead < dwDSLockedBufferSize )
    {
        // If the wav file was smaller than the DirectSound buffer,
        // we need to fill the remainder of the buffer with data
        if( bRepeatWavIfBufferLarger )
        {
            // Reset the file and fill the buffer with wav data
            DWORD dwReadSoFar = dwWavDataRead;    // From previous call above.
            while( dwReadSoFar < dwDSLockedBufferSize )
            {
                // This will keep reading in until the buffer is full
                // for very short files
                if( FAILED( hr = m_pWaveFile->ResetFile() ) )
                    return DXTRACE_ERR( TEXT("ResetFile"), hr );

                hr = m_pWaveFile->Read( (BYTE*)pDSLockedBuffer + dwReadSoFar,
                                        dwDSLockedBufferSize - dwReadSoFar,
                                        &dwWavDataRead );
                if( FAILED(hr) )
                    return DXTRACE_ERR( TEXT("Read"), hr );

                dwReadSoFar += dwWavDataRead;
            }
        }
        else
        {
            // Don't repeat the wav file, just fill in silence
            FillMemory( (BYTE*) pDSLockedBuffer + dwWavDataRead,
                        dwDSLockedBufferSize - dwWavDataRead,
                        (BYTE)(m_pWaveFile->m_pwfx->wBitsPerSample == 8 ? 128 : 0 ) );
        }
    }

    // Unlock the buffer, we don't need it anymore.
    pDSB->Unlock( pDSLockedBuffer, dwDSLockedBufferSize, NULL, 0 );

    return S_OK;
}
```

November 14, 2008 12:17 AM -- 500

```
//-----------------------------------------------------------------------------
// Name: CSound::RestoreBuffer()
// Desc: Restores the lost buffer. *pbWasRestored returns TRUE if the buffer was
//       restored.  It can also be NULL if the information is not needed.
//-----------------------------------------------------------------------------
HRESULT CSound::RestoreBuffer( LPDIRECTSOUNDBUFFER pDSB, BOOL* pbWasRestored )
{
    HRESULT hr;

    if( pDSB == NULL )
        return CO_E_NOTINITIALIZED;
    if( pbWasRestored )
        *pbWasRestored = FALSE;

    DWORD dwStatus;
    if( FAILED( hr = pDSB->GetStatus( &dwStatus ) ) )
        return DXTRACE_ERR( TEXT("GetStatus"), hr );

    if( dwStatus & DSBSTATUS_BUFFERLOST )
    {
        // Since the app could have just been activated, then DirectSound
        // may not be giving us control yet, so restoring the buffer may fail.
        // If it does, sleep until DirectSound gives us control.
        do
        {
            hr = pDSB->Restore();
            if( hr == DSERR_BUFFERLOST )
                Sleep( 10 );
        }
        while( hr != DS_OK );

        if( pbWasRestored != NULL )
            *pbWasRestored = TRUE;

        return S_OK;
    }
    else
    {
        return S_FALSE;
    }
}

//-----------------------------------------------------------------------------
// Name: CSound::GetFreeBuffer()
// Desc: Checks to see if a buffer is playing and returns TRUE if it is.
//-----------------------------------------------------------------------------
LPDIRECTSOUNDBUFFER CSound::GetFreeBuffer()
{
    if( m_apDSBuffer == NULL )
        return FALSE;

    for( DWORD i=0; i<m_dwNumBuffers; i++ )
    {
        if( m_apDSBuffer[i] )
        {
            DWORD dwStatus = 0;
            m_apDSBuffer[i]->GetStatus( &dwStatus );
            if ( ( dwStatus & DSBSTATUS_PLAYING ) == 0 )
                break;
        }
    }

    if( i != m_dwNumBuffers )
        return m_apDSBuffer[ i ];
    else
        return m_apDSBuffer[ rand() % m_dwNumBuffers ];
}

//-----------------------------------------------------------------------------
// Name: CSound::GetBuffer()
// Desc:
//-----------------------------------------------------------------------------
LPDIRECTSOUNDBUFFER CSound::GetBuffer( DWORD dwIndex )
{
    if( m_apDSBuffer == NULL )
        return NULL;
    if( dwIndex >= m_dwNumBuffers )
        return NULL;

    return m_apDSBuffer[dwIndex];
}

//-----------------------------------------------------------------------------
// Name: CSound::Get3DBufferInterface()
// Desc:
//-----------------------------------------------------------------------------
HRESULT CSound::Get3DBufferInterface( DWORD dwIndex, LPDIRECTSOUND3DBUFFER* ppDS3DBuffer )
{
    if( m_apDSBuffer == NULL )
        return CO_E_NOTINITIALIZED;
    if( dwIndex >= m_dwNumBuffers )
        return E_INVALIDARG;

    *ppDS3DBuffer = NULL;

    return m_apDSBuffer[dwIndex]->QueryInterface( IID_IDirectSound3DBuffer,
                                                  (VOID**)ppDS3DBuffer );
```

```
}

//-----------------------------------------------------------------------------
// Name: CSound::Play()
// Desc: Plays the sound using voice management flags.  Pass in DSBPLAY_LOOPING
//       in the dwFlags to loop the sound
//-----------------------------------------------------------------------------
HRESULT CSound::Play( DWORD dwPriority, DWORD dwFlags )
{
    HRESULT hr;
    BOOL    bRestored;

    if( m_apDSBuffer == NULL )
        return CO_E_NOTINITIALIZED;

    LPDIRECTSOUNDBUFFER pDSB = GetFreeBuffer();

    if( pDSB == NULL )
        return DXTRACE_ERR( TEXT("GetFreeBuffer"), E_FAIL );

    // Restore the buffer if it was lost
    if( FAILED( hr = RestoreBuffer( pDSB, &bRestored ) ) )
        return DXTRACE_ERR( TEXT("RestoreBuffer"), hr );

    if( bRestored )
    {
        // The buffer was restored, so we need to fill it with new data
        if( FAILED( hr = FillBufferWithSound( pDSB, FALSE ) ) )
            return DXTRACE_ERR( TEXT("FillBufferWithSound"), hr );

        // Make DirectSound do pre-processing on sound effects
        Reset();
    }

    return pDSB->Play( 0, dwPriority, dwFlags );
}

//-----------------------------------------------------------------------------
// Name: CSound::Stop()
// Desc: Stops the sound from playing
//-----------------------------------------------------------------------------
HRESULT CSound::Stop()
{
    if( m_apDSBuffer == NULL )
        return CO_E_NOTINITIALIZED;

    HRESULT hr = 0;

    for( DWORD i=0; i<m_dwNumBuffers; i++ )
        hr |= m_apDSBuffer[i]->Stop();

    return hr;
}

//-----------------------------------------------------------------------------
// Name: CSound::Reset()
// Desc: Reset all of the sound buffers
//-----------------------------------------------------------------------------
HRESULT CSound::Reset()
{
    if( m_apDSBuffer == NULL )
        return CO_E_NOTINITIALIZED;

    HRESULT hr = 0;

    for( DWORD i=0; i<m_dwNumBuffers; i++ )
        hr |= m_apDSBuffer[i]->SetCurrentPosition( 0 );

    return hr;
}

//-----------------------------------------------------------------------------
// Name: CSound::IsSoundPlaying()
// Desc: Checks to see if a buffer is playing and returns TRUE if it is.
//-----------------------------------------------------------------------------
BOOL CSound::IsSoundPlaying()
{
    BOOL bIsPlaying = FALSE;

    if( m_apDSBuffer == NULL )
        return FALSE;

    for( DWORD i=0; i<m_dwNumBuffers; i++ )
    {
        if( m_apDSBuffer[i] )
        {
            DWORD dwStatus = 0;
            m_apDSBuffer[i]->GetStatus( &dwStatus );
            bIsPlaying |= ( ( dwStatus & DSBSTATUS_PLAYING ) != 0 );
        }
    }

    return bIsPlaying;
}

//-----------------------------------------------------------------------------
// Name: CStreamingSound::CStreamingSound()
```

```
// Desc: Setups up a buffer so data can be streamed from the wave file into
//       buffer.  This is very useful for large wav files that would take a
//       while to load.  The buffer is initially filled with data, then
//       as sound is played the notification events are signaled and more data
//       is written into the buffer by calling HandleWaveStreamNotification()
//-----------------------------------------------------------------------------
CStreamingSound::CStreamingSound( LPDIRECTSOUNDBUFFER pDSBuffer, DWORD dwDSBufferSize,
                                  CWaveFile* pWaveFile, DWORD dwNotifySize )
                : CSound( &pDSBuffer, dwDSBufferSize, 1, pWaveFile )
{
    m_dwLastPlayPos     = 0;
    m_dwPlayProgress    = 0;
    m_dwNotifySize      = dwNotifySize;
    m_dwNextWriteOffset = 0;
    m_bFillNextNotificationWithSilence = FALSE;
}

//-----------------------------------------------------------------------------
// Name: CStreamingSound::~CStreamingSound()
// Desc: Destroys the class
//-----------------------------------------------------------------------------
CStreamingSound::~CStreamingSound()
{
}

//-----------------------------------------------------------------------------
// Name: CStreamingSound::HandleWaveStreamNotification()
// Desc: Handle the notification that tell us to put more wav data in the
//       circular buffer
//-----------------------------------------------------------------------------
HRESULT CStreamingSound::HandleWaveStreamNotification( BOOL bLoopedPlay )
{
    HRESULT hr;
    DWORD   dwCurrentPlayPos;
    DWORD   dwPlayDelta;
    DWORD   dwBytesWrittenToBuffer;
    VOID*   pDSLockedBuffer = NULL;
    VOID*   pDSLockedBuffer2 = NULL;
    DWORD   dwDSLockedBufferSize;
    DWORD   dwDSLockedBufferSize2;

    if( m_apDSBuffer == NULL || m_pWaveFile == NULL )
        return CO_E_NOTINITIALIZED;

    // Restore the buffer if it was lost
    BOOL bRestored;
    if( FAILED( hr = RestoreBuffer( m_apDSBuffer[0], &bRestored ) ) )
        return DXTRACE_ERR( TEXT("RestoreBuffer"), hr );

    if( bRestored )
    {
        // The buffer was restored, so we need to fill it with new data
        if( FAILED( hr = FillBufferWithSound( m_apDSBuffer[0], FALSE ) ) )
            return DXTRACE_ERR( TEXT("FillBufferWithSound"), hr );
        return S_OK;
    }

    // Lock the DirectSound buffer
    if( FAILED( hr = m_apDSBuffer[0]->Lock( m_dwNextWriteOffset, m_dwNotifySize,
                                            &pDSLockedBuffer, &dwDSLockedBufferSize,
                                            &pDSLockedBuffer2, &dwDSLockedBufferSize2, 0L ) ) )
        return DXTRACE_ERR( TEXT("Lock"), hr );

    // m_dwDSBufferSize and m_dwNextWriteOffset are both multiples of m_dwNotifySize,
    // it should the second buffer should never be valid
    if( pDSLockedBuffer2 != NULL )
        return E_UNEXPECTED;

    if( !m_bFillNextNotificationWithSilence )
    {
        // Fill the DirectSound buffer with wav data
        if( FAILED( hr = m_pWaveFile->Read( (BYTE*) pDSLockedBuffer,
                                            dwDSLockedBufferSize,
                                            &dwBytesWrittenToBuffer ) ) )
            return DXTRACE_ERR( TEXT("Read"), hr );
    }
    else
    {
        // Fill the DirectSound buffer with silence
        FillMemory( pDSLockedBuffer, dwDSLockedBufferSize,
                    (BYTE)( m_pWaveFile->m_pwfx->wBitsPerSample == 8 ? 128 : 0 ) );
        dwBytesWrittenToBuffer = dwDSLockedBufferSize;
    }

    // If the number of bytes written is less than the
    // amount we requested, we have a short file.
    if( dwBytesWrittenToBuffer < dwDSLockedBufferSize )
    {
        if( !bLoopedPlay )
        {
            // Fill in silence for the rest of the buffer.
            FillMemory( (BYTE*) pDSLockedBuffer + dwBytesWrittenToBuffer,
                        dwDSLockedBufferSize - dwBytesWrittenToBuffer,
                        (BYTE)(m_pWaveFile->m_pwfx->wBitsPerSample == 8 ? 128 : 0 ) );

            // Any future notifications should just fill the buffer with silence
            m_bFillNextNotificationWithSilence = TRUE;
```

```
        }
        else
        {
            // We are looping, so reset the file and fill the buffer with wav data
            DWORD dwReadSoFar = dwBytesWrittenToBuffer;    // From previous call above.
            while( dwReadSoFar < dwDSLockedBufferSize )
            {
                // This will keep reading in until the buffer is full (for very short files).
                if( FAILED( hr = m_pWaveFile->ResetFile() ) )
                    return DXTRACE_ERR( TEXT("ResetFile"), hr );

                if( FAILED( hr = m_pWaveFile->Read( (BYTE*)pDSLockedBuffer + dwReadSoFar,
                                                    dwDSLockedBufferSize - dwReadSoFar,
                                                    &dwBytesWrittenToBuffer ) ) )
                    return DXTRACE_ERR( TEXT("Read"), hr );

                dwReadSoFar += dwBytesWrittenToBuffer;
            }
        }
    }

    // Unlock the DirectSound buffer
    m_apDSBuffer[0]->Unlock( pDSLockedBuffer, dwDSLockedBufferSize, NULL, 0 );

    // Figure out how much data has been played so far.  When we have played
    // passed the end of the file, we will either need to start filling the
    // buffer with silence or starting reading from the beginning of the file,
    // depending if the user wants to loop the sound
    if( FAILED( hr = m_apDSBuffer[0]->GetCurrentPosition( &dwCurrentPlayPos, NULL ) ) )
        return DXTRACE_ERR( TEXT("GetCurrentPosition"), hr );

    // Check to see if the position counter looped
    if( dwCurrentPlayPos < m_dwLastPlayPos )
        dwPlayDelta = ( m_dwDSBufferSize - m_dwLastPlayPos ) + dwCurrentPlayPos;
    else
        dwPlayDelta = dwCurrentPlayPos - m_dwLastPlayPos;

    m_dwPlayProgress += dwPlayDelta;
    m_dwLastPlayPos = dwCurrentPlayPos;

    // If we are now filling the buffer with silence, then we have found the end so
    // check to see if the entire sound has played, if it has then stop the buffer.
    if( m_bFillNextNotificationWithSilence )
    {
        // We don't want to cut off the sound before it's done playing.
        if( m_dwPlayProgress >= m_pWaveFile->GetSize() )
        {
            m_apDSBuffer[0]->Stop();
        }
    }

    // Update where the buffer will lock (for next time)
    m_dwNextWriteOffset += dwDSLockedBufferSize;
    m_dwNextWriteOffset %= m_dwDSBufferSize; // Circular buffer

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CStreamingSound::Reset()
// Desc: Resets the sound so it will begin playing at the beginning
//-----------------------------------------------------------------------------
HRESULT CStreamingSound::Reset()
{
    HRESULT hr;

    if( m_apDSBuffer[0] == NULL || m_pWaveFile == NULL )
        return CO_E_NOTINITIALIZED;

    m_dwLastPlayPos     = 0;
    m_dwPlayProgress    = 0;
    m_dwNextWriteOffset = 0;
    m_bFillNextNotificationWithSilence = FALSE;

    // Restore the buffer if it was lost
    BOOL bRestored;
    if( FAILED( hr = RestoreBuffer( m_apDSBuffer[0], &bRestored ) ) )
        return DXTRACE_ERR( TEXT("RestoreBuffer"), hr );

    if( bRestored )
    {
        // The buffer was restored, so we need to fill it with new data
        if( FAILED( hr = FillBufferWithSound( m_apDSBuffer[0], FALSE ) ) )
            return DXTRACE_ERR( TEXT("FillBufferWithSound"), hr );
    }

    m_pWaveFile->ResetFile();

    return m_apDSBuffer[0]->SetCurrentPosition( 0L );
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::CWaveFile()
// Desc: Constructs the class.  Call Open() to open a wave file for reading.
//       Then call Read() as needed.  Calling the destructor or Close()
//       will close the file.
//-----------------------------------------------------------------------------
```

```
CWaveFile::CWaveFile()
{
    m_pwfx    = NULL;
    m_hmmio   = NULL;
    m_pResourceBuffer = NULL;
    m_dwSize  = 0;
    m_bIsReadingFromMemory = FALSE;
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::~CWaveFile()
// Desc: Destructs the class
//-----------------------------------------------------------------------------
CWaveFile::~CWaveFile()
{
    Close();

    if( !m_bIsReadingFromMemory )
        SAFE_DELETE_ARRAY( m_pwfx );
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::Open()
// Desc: Opens a wave file for reading
//-----------------------------------------------------------------------------
HRESULT CWaveFile::Open( LPTSTR strFileName, WAVEFORMATEX* pwfx, DWORD dwFlags )
{
    HRESULT hr;

    m_dwFlags = dwFlags;
    m_bIsReadingFromMemory = FALSE;

    if( m_dwFlags == WAVEFILE_READ )
    {
        if( strFileName == NULL )
            return E_INVALIDARG;
        SAFE_DELETE_ARRAY( m_pwfx );

        m_hmmio = mmioOpen( strFileName, NULL, MMIO_ALLOCBUF | MMIO_READ );

        if( NULL == m_hmmio )
        {
            HRSRC   hResInfo;
            HGLOBAL hResData;
            DWORD   dwSize;
            VOID*   pvRes;

            // Loading it as a file failed, so try it as a resource
            if( NULL == ( hResInfo = FindResource( NULL, strFileName, TEXT("WAVE") ) ) )
            {
                if( NULL == ( hResInfo = FindResource( NULL, strFileName, TEXT("WAV") ) ) )
                    return DXTRACE_ERR_NOMSGBOX( TEXT("FindResource"), E_FAIL );
            }

            if( NULL == ( hResData = LoadResource( NULL, hResInfo ) ) )
                return DXTRACE_ERR( TEXT("LoadResource"), E_FAIL );

            if( 0 == ( dwSize = SizeofResource( NULL, hResInfo ) ) )
                return DXTRACE_ERR( TEXT("SizeofResource"), E_FAIL );

            if( NULL == ( pvRes = LockResource( hResData ) ) )
                return DXTRACE_ERR( TEXT("LockResource"), E_FAIL );

            m_pResourceBuffer = new CHAR[ dwSize ];
            memcpy( m_pResourceBuffer, pvRes, dwSize );

            MMIOINFO mmioInfo;
            ZeroMemory( &mmioInfo, sizeof(mmioInfo) );
            mmioInfo.fccIOProc = FOURCC_MEM;
            mmioInfo.cchBuffer = dwSize;
            mmioInfo.pchBuffer = (CHAR*) m_pResourceBuffer;

            m_hmmio = mmioOpen( NULL, &mmioInfo, MMIO_ALLOCBUF | MMIO_READ );
        }

        if( FAILED( hr = ReadMMIO() ) )
        {
            // ReadMMIO will fail if its an not a wave file
            mmioClose( m_hmmio, 0 );
            return DXTRACE_ERR_NOMSGBOX( TEXT("ReadMMIO"), hr );
        }

        if( FAILED( hr = ResetFile() ) )
            return DXTRACE_ERR( TEXT("ResetFile"), hr );

        // After the reset, the size of the wav file is m_ck.cksize so store it now
        m_dwSize = m_ck.cksize;
    }
    else
    {
        m_hmmio = mmioOpen( strFileName, NULL, MMIO_ALLOCBUF  |
                                               MMIO_READWRITE |
                                               MMIO_CREATE );
        if( NULL == m_hmmio )
            return DXTRACE_ERR( TEXT("mmioOpen"), E_FAIL );

        if( FAILED( hr = WriteMMIO( pwfx ) ) )
```

```
        {
            mmioClose( m_hmmio, 0 );
            return DXTRACE_ERR( TEXT("WriteMMIO"), hr );
        }

        if( FAILED( hr = ResetFile() ) )
            return DXTRACE_ERR( TEXT("ResetFile"), hr );
    }

    return hr;
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::OpenFromMemory()
// Desc: copy data to CWaveFile member variable from memory
//-----------------------------------------------------------------------------
HRESULT CWaveFile::OpenFromMemory( BYTE* pbData, ULONG ulDataSize,
                                   WAVEFORMATEX* pwfx, DWORD dwFlags )
{
    m_pwfx       = pwfx;
    m_ulDataSize = ulDataSize;
    m_pbData     = pbData;
    m_pbDataCur  = m_pbData;
    m_bIsReadingFromMemory = TRUE;

    if( dwFlags != WAVEFILE_READ )
        return E_NOTIMPL;

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::ReadMMIO()
// Desc: Support function for reading from a multimedia I/O stream.
//       m_hmmio must be valid before calling.  This function uses it to
//       update m_ckRiff, and m_pwfx.
//-----------------------------------------------------------------------------
HRESULT CWaveFile::ReadMMIO()
{
    MMCKINFO        ckIn;           // chunk info. for general use.
    PCMWAVEFORMAT   pcmWaveFormat;  // Temp PCM structure to load in.

    m_pwfx = NULL;

    if( ( 0 != mmioDescend( m_hmmio, &m_ckRiff, NULL, 0 ) ) )
        return DXTRACE_ERR( TEXT("mmioDescend"), E_FAIL );

    // Check to make sure this is a valid wave file
    if( (m_ckRiff.ckid != FOURCC_RIFF) ||
        (m_ckRiff.fccType != mmioFOURCC('W', 'A', 'V', 'E') ) )
        return DXTRACE_ERR_NOMSGBOX( TEXT("mmioFOURCC"), E_FAIL );

    // Search the input file for for the 'fmt ' chunk.
    ckIn.ckid = mmioFOURCC('f', 'm', 't', ' ');
    if( 0 != mmioDescend( m_hmmio, &ckIn, &m_ckRiff, MMIO_FINDCHUNK ) )
        return DXTRACE_ERR( TEXT("mmioDescend"), E_FAIL );

    // Expect the 'fmt' chunk to be at least as large as <PCMWAVEFORMAT>;
    // if there are extra parameters at the end, we'll ignore them
       if( ckIn.cksize < (LONG) sizeof(PCMWAVEFORMAT) )
            return DXTRACE_ERR( TEXT("sizeof(PCMWAVEFORMAT)"), E_FAIL );

    // Read the 'fmt ' chunk into <pcmWaveFormat>.
    if( mmioRead( m_hmmio, (HPSTR) &pcmWaveFormat,
                  sizeof(pcmWaveFormat)) != sizeof(pcmWaveFormat) )
        return DXTRACE_ERR( TEXT("mmioRead"), E_FAIL );

    // Allocate the waveformatex, but if its not pcm format, read the next
    // word, and thats how many extra bytes to allocate.
    if( pcmWaveFormat.wf.wFormatTag == WAVE_FORMAT_PCM )
    {
        m_pwfx = (WAVEFORMATEX*)new CHAR[ sizeof(WAVEFORMATEX) ];
        if( NULL == m_pwfx )
            return DXTRACE_ERR( TEXT("m_pwfx"), E_FAIL );

        // Copy the bytes from the pcm structure to the waveformatex structure
        memcpy( m_pwfx, &pcmWaveFormat, sizeof(pcmWaveFormat) );
        m_pwfx->cbSize = 0;
    }
    else
    {
        // Read in length of extra bytes.
        WORD cbExtraBytes = 0L;
        if( mmioRead( m_hmmio, (CHAR*)&cbExtraBytes, sizeof(WORD)) != sizeof(WORD) )
            return DXTRACE_ERR( TEXT("mmioRead"), E_FAIL );

        m_pwfx = (WAVEFORMATEX*)new CHAR[ sizeof(WAVEFORMATEX) + cbExtraBytes ];
        if( NULL == m_pwfx )
            return DXTRACE_ERR( TEXT("new"), E_FAIL );

        // Copy the bytes from the pcm structure to the waveformatex structure
        memcpy( m_pwfx, &pcmWaveFormat, sizeof(pcmWaveFormat) );
        m_pwfx->cbSize = cbExtraBytes;

        // Now, read those extra bytes into the structure, if cbExtraAlloc != 0.
        if( mmioRead( m_hmmio, (CHAR*)(((BYTE*)&(m_pwfx->cbSize))+sizeof(WORD)),
                      cbExtraBytes ) != cbExtraBytes )
```

```
        {
            SAFE_DELETE( m_pwfx );
            return DXTRACE_ERR( TEXT("mmioRead"), E_FAIL );
        }
    }

    // Ascend the input file out of the 'fmt ' chunk.
    if( 0 != mmioAscend( m_hmmio, &ckIn, 0 ) )
    {
        SAFE_DELETE( m_pwfx );
        return DXTRACE_ERR( TEXT("mmioAscend"), E_FAIL );
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::GetSize()
// Desc: Retuns the size of the read access wave file
//-----------------------------------------------------------------------------
DWORD CWaveFile::GetSize()
{
    return m_dwSize;
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::ResetFile()
// Desc: Resets the internal m_ck pointer so reading starts from the
//       beginning of the file again
//-----------------------------------------------------------------------------
HRESULT CWaveFile::ResetFile()
{
    if( m_bIsReadingFromMemory )
    {
        m_pbDataCur = m_pbData;
    }
    else
    {
        if( m_hmmio == NULL )
            return CO_E_NOTINITIALIZED;

        if( m_dwFlags == WAVEFILE_READ )
        {
            // Seek to the data
            if( -1 == mmioSeek( m_hmmio, m_ckRiff.dwDataOffset + sizeof(FOURCC),
                                SEEK_SET ) )
                return DXTRACE_ERR( TEXT("mmioSeek"), E_FAIL );

            // Search the input file for the 'data' chunk.
            m_ck.ckid = mmioFOURCC('d', 'a', 't', 'a');
            if( 0 != mmioDescend( m_hmmio, &m_ck, &m_ckRiff, MMIO_FINDCHUNK ) )
              return DXTRACE_ERR( TEXT("mmioDescend"), E_FAIL );
        }
        else
        {
            // Create the 'data' chunk that holds the waveform samples.
            m_ck.ckid = mmioFOURCC('d', 'a', 't', 'a');
            m_ck.cksize = 0;

            if( 0 != mmioCreateChunk( m_hmmio, &m_ck, 0 ) )
                return DXTRACE_ERR( TEXT("mmioCreateChunk"), E_FAIL );

            if( 0 != mmioGetInfo( m_hmmio, &m_mmioinfoOut, 0 ) )
                return DXTRACE_ERR( TEXT("mmioGetInfo"), E_FAIL );
        }
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::Read()
// Desc: Reads section of data from a wave file into pBuffer and returns
//       how much read in pdwSizeRead, reading not more than dwSizeToRead.
//       This uses m_ck to determine where to start reading from.  So
//       subsequent calls will be continue where the last left off unless
//       Reset() is called.
//-----------------------------------------------------------------------------
HRESULT CWaveFile::Read( BYTE* pBuffer, DWORD dwSizeToRead, DWORD* pdwSizeRead )
{
    if( m_bIsReadingFromMemory )
    {
        if( m_pbDataCur == NULL )
            return CO_E_NOTINITIALIZED;
        if( pdwSizeRead != NULL )
            *pdwSizeRead = 0;

        if( (BYTE*)(m_pbDataCur + dwSizeToRead) >
            (BYTE*)(m_pbData + m_ulDataSize) )
        {
            dwSizeToRead = m_ulDataSize - (DWORD)(m_pbDataCur - m_pbData);
        }

        CopyMemory( pBuffer, m_pbDataCur, dwSizeToRead );

        if( pdwSizeRead != NULL )
            *pdwSizeRead = dwSizeToRead;
```

```
        return S_OK;
    }
    else
    {
        MMIOINFO mmioinfoIn; // current status of m_hmmio

        if( m_hmmio == NULL )
            return CO_E_NOTINITIALIZED;
        if( pBuffer == NULL || pdwSizeRead == NULL )
            return E_INVALIDARG;

        if( pdwSizeRead != NULL )
            *pdwSizeRead = 0;

        if( 0 != mmioGetInfo( m_hmmio, &mmioinfoIn, 0 ) )
            return DXTRACE_ERR( TEXT("mmioGetInfo"), E_FAIL );

        UINT cbDataIn = dwSizeToRead;
        if( cbDataIn > m_ck.cksize )
            cbDataIn = m_ck.cksize;

        m_ck.cksize -= cbDataIn;

        for( DWORD cT = 0; cT < cbDataIn; cT++ )
        {
            // Copy the bytes from the io to the buffer.
            if( mmioinfoIn.pchNext == mmioinfoIn.pchEndRead )
            {
                if( 0 != mmioAdvance( m_hmmio, &mmioinfoIn, MMIO_READ ) )
                    return DXTRACE_ERR( TEXT("mmioAdvance"), E_FAIL );

                if( mmioinfoIn.pchNext == mmioinfoIn.pchEndRead )
                    return DXTRACE_ERR( TEXT("mmioinfoIn.pchNext"), E_FAIL );
            }

            // Actual copy.
            *((BYTE*)pBuffer+cT) = *((BYTE*)mmioinfoIn.pchNext);
            mmioinfoIn.pchNext++;
        }

        if( 0 != mmioSetInfo( m_hmmio, &mmioinfoIn, 0 ) )
            return DXTRACE_ERR( TEXT("mmioSetInfo"), E_FAIL );

        if( pdwSizeRead != NULL )
            *pdwSizeRead = cbDataIn;

        return S_OK;
    }
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::Close()
// Desc: Closes the wave file
//-----------------------------------------------------------------------------
HRESULT CWaveFile::Close()
{
    if( m_dwFlags == WAVEFILE_READ )
    {
        mmioClose( m_hmmio, 0 );
        m_hmmio = NULL;
        SAFE_DELETE_ARRAY( m_pResourceBuffer );
    }
    else
    {
        m_mmioinfoOut.dwFlags |= MMIO_DIRTY;

        if( m_hmmio == NULL )
            return CO_E_NOTINITIALIZED;

        if( 0 != mmioSetInfo( m_hmmio, &m_mmioinfoOut, 0 ) )
            return DXTRACE_ERR( TEXT("mmioSetInfo"), E_FAIL );

        // Ascend the output file out of the 'data' chunk -- this will cause
        // the chunk size of the 'data' chunk to be written.
        if( 0 != mmioAscend( m_hmmio, &m_ck, 0 ) )
            return DXTRACE_ERR( TEXT("mmioAscend"), E_FAIL );

        // Do this here instead...
        if( 0 != mmioAscend( m_hmmio, &m_ckRiff, 0 ) )
            return DXTRACE_ERR( TEXT("mmioAscend"), E_FAIL );

        mmioSeek( m_hmmio, 0, SEEK_SET );

        if( 0 != (INT)mmioDescend( m_hmmio, &m_ckRiff, NULL, 0 ) )
            return DXTRACE_ERR( TEXT("mmioDescend"), E_FAIL );

        m_ck.ckid = mmioFOURCC('f', 'a', 'c', 't');

        if( 0 == mmioDescend( m_hmmio, &m_ck, &m_ckRiff, MMIO_FINDCHUNK ) )
        {
            DWORD dwSamples = 0;
            mmioWrite( m_hmmio, (HPSTR)&dwSamples, sizeof(DWORD) );
            mmioAscend( m_hmmio, &m_ck, 0 );
        }

        // Ascend the output file out of the 'RIFF' chunk -- this will cause
```

```
        // the chunk size of the 'RIFF' chunk to be written.
        if( 0 != mmioAscend( m_hmmio, &m_ckRiff, 0 ) )
            return DXTRACE_ERR( TEXT("mmioAscend"), E_FAIL );

        mmioClose( m_hmmio, 0 );
        m_hmmio = NULL;
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::WriteMMIO()
// Desc: Support function for reading from a multimedia I/O stream
//       pwfxDest is the WAVEFORMATEX for this new wave file.
//       m_hmmio must be valid before calling.  This function uses it to
//       update m_ckRiff, and m_ck.
//-----------------------------------------------------------------------------
HRESULT CWaveFile::WriteMMIO( WAVEFORMATEX *pwfxDest )
{
    DWORD    dwFactChunk; // Contains the actual fact chunk. Garbage until WaveCloseWriteFile.
    MMCKINFO ckOut1;

    dwFactChunk = (DWORD)-1;

    // Create the output file RIFF chunk of form type 'WAVE'.
    m_ckRiff.fccType = mmioFOURCC('W', 'A', 'V', 'E');
    m_ckRiff.cksize = 0;

    if( 0 != mmioCreateChunk( m_hmmio, &m_ckRiff, MMIO_CREATERIFF ) )
        return DXTRACE_ERR( TEXT("mmioCreateChunk"), E_FAIL );

    // We are now descended into the 'RIFF' chunk we just created.
    // Now create the 'fmt ' chunk. Since we know the size of this chunk,
    // specify it in the MMCKINFO structure so MMIO doesn't have to seek
    // back and set the chunk size after ascending from the chunk.
    m_ck.ckid = mmioFOURCC('f', 'm', 't', ' ');
    m_ck.cksize = sizeof(PCMWAVEFORMAT);

    if( 0 != mmioCreateChunk( m_hmmio, &m_ck, 0 ) )
        return DXTRACE_ERR( TEXT("mmioCreateChunk"), E_FAIL );

    // Write the PCMWAVEFORMAT structure to the 'fmt ' chunk if its that type.
    if( pwfxDest->wFormatTag == WAVE_FORMAT_PCM )
    {
        if( mmioWrite( m_hmmio, (HPSTR) pwfxDest,
                       sizeof(PCMWAVEFORMAT)) != sizeof(PCMWAVEFORMAT))
            return DXTRACE_ERR( TEXT("mmioWrite"), E_FAIL );
    }
    else
    {
        // Write the variable length size.
        if( (UINT)mmioWrite( m_hmmio, (HPSTR) pwfxDest,
                             sizeof(*pwfxDest) + pwfxDest->cbSize ) !=
                             ( sizeof(*pwfxDest) + pwfxDest->cbSize ) )
            return DXTRACE_ERR( TEXT("mmioWrite"), E_FAIL );
    }

    // Ascend out of the 'fmt ' chunk, back into the 'RIFF' chunk.
    if( 0 != mmioAscend( m_hmmio, &m_ck, 0 ) )
        return DXTRACE_ERR( TEXT("mmioAscend"), E_FAIL );

    // Now create the fact chunk, not required for PCM but nice to have.  This is filled
    // in when the close routine is called.
    ckOut1.ckid = mmioFOURCC('f', 'a', 'c', 't');
    ckOut1.cksize = 0;

    if( 0 != mmioCreateChunk( m_hmmio, &ckOut1, 0 ) )
        return DXTRACE_ERR( TEXT("mmioCreateChunk"), E_FAIL );

    if( mmioWrite( m_hmmio, (HPSTR)&dwFactChunk, sizeof(dwFactChunk)) !=
                   sizeof(dwFactChunk) )
         return DXTRACE_ERR( TEXT("mmioWrite"), E_FAIL );

    // Now ascend out of the fact chunk...
    if( 0 != mmioAscend( m_hmmio, &ckOut1, 0 ) )
        return DXTRACE_ERR( TEXT("mmioAscend"), E_FAIL );

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: CWaveFile::Write()
// Desc: Writes data to the open wave file
//-----------------------------------------------------------------------------
HRESULT CWaveFile::Write( UINT nSizeToWrite, BYTE* pbSrcData, UINT* pnSizeWrote )
{
    UINT cT;

    if( m_bIsReadingFromMemory )
        return E_NOTIMPL;
    if( m_hmmio == NULL )
        return CO_E_NOTINITIALIZED;
    if( pnSizeWrote == NULL || pbSrcData == NULL )
        return E_INVALIDARG;

    *pnSizeWrote = 0;
```

```
    for( cT = 0; cT < nSizeToWrite; cT++ )
    {
        if( m_mmioinfoOut.pchNext == m_mmioinfoOut.pchEndWrite )
        {
            m_mmioinfoOut.dwFlags |= MMIO_DIRTY;
            if( 0 != mmioAdvance( m_hmmio, &m_mmioinfoOut, MMIO_WRITE ) )
                return DXTRACE_ERR( TEXT("mmioAdvance"), E_FAIL );
        }

        *((BYTE*)m_mmioinfoOut.pchNext) = *((BYTE*)pbSrcData+cT);
        (BYTE*)m_mmioinfoOut.pchNext++;

        (*pnSizeWrote)++;
    }

    return S_OK;
}
```

Файл Sound\dxutil.cpp

```
//-----------------------------------------------------------------------------
// File: DXUtil.cpp
//
// Desc: Shortcut macros and functions for using DX objects
//
//
// Copyright (c) 1997-2001 Microsoft Corporation. All rights reserved
//-----------------------------------------------------------------------------
#define STRICT
#include <windows.h>
#include <mmsystem.h>
#include <tchar.h>
#include <stdio.h>
#include <stdarg.h>
#include "DXUtil.h"

//-----------------------------------------------------------------------------
// Name: DXUtil_GetDXSDKMediaPath()
// Desc: Returns the DirectX SDK media path
//-----------------------------------------------------------------------------
const TCHAR* DXUtil_GetDXSDKMediaPath()
{
    static TCHAR strNull[2] = _T("");
    static TCHAR strPath[MAX_PATH];
    DWORD dwType;
    DWORD dwSize = MAX_PATH;
    HKEY  hKey;

    // Open the appropriate registry key
    LONG lResult = RegOpenKeyEx( HKEY_LOCAL_MACHINE,
                                 _T("Software\\Microsoft\\DirectX SDK"),
                                 0, KEY_READ, &hKey );
    if( ERROR_SUCCESS != lResult )
        return strNull;

    lResult = RegQueryValueEx( hKey, _T("DX81SDK Samples Path"), NULL,
                               &dwType, (BYTE*)strPath, &dwSize );
    RegCloseKey( hKey );

    if( ERROR_SUCCESS != lResult )
        return strNull;

    _tcscat( strPath, _T("\\Media\\") );

    return strPath;
}

//-----------------------------------------------------------------------------
// Name: DXUtil_FindMediaFile()
// Desc: Returns a valid path to a DXSDK media file
//-----------------------------------------------------------------------------
HRESULT DXUtil_FindMediaFile( TCHAR* strPath, TCHAR* strFilename )
{
    HANDLE file;
    TCHAR strFullPath[1024];
    TCHAR *strShortName;
    DWORD cchPath;

    if( NULL==strFilename || NULL==strPath )
        return E_INVALIDARG;

    // Build full path name from strFileName (strShortName will be just the leaf filename)
    cchPath = GetFullPathName(strFilename, sizeof(strFullPath)/sizeof(TCHAR), strFullPath, &strShortName);
    if ((cchPath == 0) || (sizeof(strFullPath)/sizeof(TCHAR) <= cchPath))
        return E_FAIL;

    // first try to find the filename given a full path
    file = CreateFile( strFullPath, GENERIC_READ, FILE_SHARE_READ, NULL,
                       OPEN_EXISTING, 0, NULL );
```

```
    if( INVALID_HANDLE_VALUE != file )
    {
        _tcscpy( strPath, strFullPath );
        CloseHandle( file );
        return S_OK;
    }

    // next try to find the filename in the current working directory (path stripped)
    file = CreateFile( strShortName, GENERIC_READ, FILE_SHARE_READ, NULL,
                       OPEN_EXISTING, 0, NULL );
    if( INVALID_HANDLE_VALUE != file )
    {
        _tcscpy( strPath, strShortName );
        CloseHandle( file );
        return S_OK;
    }

    // last, check if the file exists in the media directory
    _stprintf( strPath, _T("%s%s"), DXUtil_GetDXSDKMediaPath(), strShortName );

    file = CreateFile( strPath, GENERIC_READ, FILE_SHARE_READ, NULL,
                       OPEN_EXISTING, 0, NULL );
    if( INVALID_HANDLE_VALUE != file )
    {
        CloseHandle( file );
        return S_OK;
    }

    // On failure, just return the file as the path
    _tcscpy( strPath, strFilename );
    return E_FAIL;
}

//-----------------------------------------------------------------------------
// Name: DXUtil_ReadStringRegKey()
// Desc: Helper function to read a registry key string
//-----------------------------------------------------------------------------
HRESULT DXUtil_ReadStringRegKey( HKEY hKey, TCHAR* strRegName, TCHAR* strValue,
                                 DWORD dwLength, TCHAR* strDefault )
{
    DWORD dwType;

    if( ERROR_SUCCESS != RegQueryValueEx( hKey, strRegName, 0, &dwType,
                                          (BYTE*)strValue, &dwLength ) )
    {
        _tcscpy( strValue, strDefault );
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: DXUtil_WriteStringRegKey()
// Desc: Helper function to write a registry key string
//-----------------------------------------------------------------------------
HRESULT DXUtil_WriteStringRegKey( HKEY hKey, TCHAR* strRegName,
                                  TCHAR* strValue )
{
    if( ERROR_SUCCESS != RegSetValueEx( hKey, strRegName, 0, REG_SZ,
                                        (BYTE*)strValue,
                                        (_tcslen(strValue)+1)*sizeof(TCHAR) ) )
        return E_FAIL;

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: DXUtil_ReadIntRegKey()
// Desc: Helper function to read a registry key int
//-----------------------------------------------------------------------------
HRESULT DXUtil_ReadIntRegKey( HKEY hKey, TCHAR* strRegName, DWORD* pdwValue,
                              DWORD dwDefault )
{
    DWORD dwType;
    DWORD dwLength = sizeof(DWORD);

    if( ERROR_SUCCESS != RegQueryValueEx( hKey, strRegName, 0, &dwType,
                                          (BYTE*)pdwValue, &dwLength ) )
    {
        *pdwValue = dwDefault;
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: DXUtil_WriteIntRegKey()
// Desc: Helper function to write a registry key int
//-----------------------------------------------------------------------------
HRESULT DXUtil_WriteIntRegKey( HKEY hKey, TCHAR* strRegName, DWORD dwValue )
{
    if( ERROR_SUCCESS != RegSetValueEx( hKey, strRegName, 0, REG_DWORD,
                                        (BYTE*)&dwValue, sizeof(DWORD) ) )
        return E_FAIL;

    return S_OK;
}
```

```
//-----------------------------------------------------------------------------
// Name: DXUtil_ReadBoolRegKey()
// Desc: Helper function to read a registry key BOOL
//-----------------------------------------------------------------------------
HRESULT DXUtil_ReadBoolRegKey( HKEY hKey, TCHAR* strRegName, BOOL* pbValue,
                               BOOL bDefault )
{
    DWORD dwType;
    DWORD dwLength = sizeof(BOOL);

    if( ERROR_SUCCESS != RegQueryValueEx( hKey, strRegName, 0, &dwType,
                                          (BYTE*)pbValue, &dwLength ) )
    {
        *pbValue = bDefault;
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: DXUtil_WriteBoolRegKey()
// Desc: Helper function to write a registry key BOOL
//-----------------------------------------------------------------------------
HRESULT DXUtil_WriteBoolRegKey( HKEY hKey, TCHAR* strRegName, BOOL bValue )
{
    if( ERROR_SUCCESS != RegSetValueEx( hKey, strRegName, 0, REG_DWORD,
                                        (BYTE*)&bValue, sizeof(BOOL) ) )
        return E_FAIL;

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: DXUtil_ReadGuidRegKey()
// Desc: Helper function to read a registry key guid
//-----------------------------------------------------------------------------
HRESULT DXUtil_ReadGuidRegKey( HKEY hKey, TCHAR* strRegName, GUID* pGuidValue,
                               GUID& guidDefault )
{
    DWORD dwType;
    DWORD dwLength = sizeof(GUID);

    if( ERROR_SUCCESS != RegQueryValueEx( hKey, strRegName, 0, &dwType,
                                          (LPBYTE) pGuidValue, &dwLength ) )
    {
        *pGuidValue = guidDefault;
    }

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: DXUtil_WriteGuidRegKey()
// Desc: Helper function to write a registry key guid
//-----------------------------------------------------------------------------
HRESULT DXUtil_WriteGuidRegKey( HKEY hKey, TCHAR* strRegName, GUID guidValue )
{
    if( ERROR_SUCCESS != RegSetValueEx( hKey, strRegName, 0, REG_BINARY,
                                        (BYTE*)&guidValue, sizeof(GUID) ) )
        return E_FAIL;

    return S_OK;
}

//-----------------------------------------------------------------------------
// Name: DXUtil_Timer()
// Desc: Performs timer opertations. Use the following commands:
//          TIMER_RESET           - to reset the timer
//          TIMER_START           - to start the timer
//          TIMER_STOP            - to stop (or pause) the timer
//          TIMER_ADVANCE         - to advance the timer by 0.1 seconds
//          TIMER_GETABSOLUTETIME - to get the absolute system time
//          TIMER_GETAPPTIME      - to get the current time
//          TIMER_GETELAPSEDTIME  - to get the time that elapsed between
//                                  TIMER_GETELAPSEDTIME calls
//-----------------------------------------------------------------------------
FLOAT __stdcall DXUtil_Timer( TIMER_COMMAND command )
{
    static BOOL     m_bTimerInitialized = FALSE;
    static BOOL     m_bUsingQPF         = FALSE;
    static BOOL     m_bTimerStopped     = TRUE;
    static LONGLONG m_llQPFTicksPerSec  = 0;

    // Initialize the timer
    if( FALSE == m_bTimerInitialized )
    {
        m_bTimerInitialized = TRUE;

        // Use QueryPerformanceFrequency() to get frequency of timer.  If QPF is
        // not supported, we will timeGetTime() which returns milliseconds.
        LARGE_INTEGER qwTicksPerSec;
        m_bUsingQPF = QueryPerformanceFrequency( &qwTicksPerSec );
        if( m_bUsingQPF )
            m_llQPFTicksPerSec = qwTicksPerSec.QuadPart;
    }
```

```
    if( m_bUsingQPF )
    {
        static LONGLONG m_llStopTime        = 0;
        static LONGLONG m_llLastElapsedTime = 0;
        static LONGLONG m_llBaseTime        = 0;
        double fTime;
        double fElapsedTime;
        LARGE_INTEGER qwTime;

        // Get either the current time or the stop time, depending
        // on whether we're stopped and what command was sent
        if( m_llStopTime != 0 && command != TIMER_START && command != TIMER_GETABSOLUTETIME)
            qwTime.QuadPart = m_llStopTime;
        else
            QueryPerformanceCounter( &qwTime );

        // Return the elapsed time
        if( command == TIMER_GETELAPSEDTIME )
        {
            fElapsedTime = (double) ( qwTime.QuadPart - m_llLastElapsedTime ) / (double)
m_llQPFTicksPerSec;
            m_llLastElapsedTime = qwTime.QuadPart;
            return (FLOAT) fElapsedTime;
        }

        // Return the current time
        if( command == TIMER_GETAPPTIME )
        {
            double fAppTime = (double) ( qwTime.QuadPart - m_llBaseTime ) / (double) m_llQPFTicksPerSec;
            return (FLOAT) fAppTime;
        }

        // Reset the timer
        if( command == TIMER_RESET )
        {
            m_llBaseTime        = qwTime.QuadPart;
            m_llLastElapsedTime = qwTime.QuadPart;
            m_llStopTime        = 0;
            m_bTimerStopped     = FALSE;
            return 0.0f;
        }

        // Start the timer
        if( command == TIMER_START )
        {
            if( m_bTimerStopped )
                m_llBaseTime += qwTime.QuadPart - m_llStopTime;
            m_llStopTime = 0;
            m_llLastElapsedTime = qwTime.QuadPart;
            m_bTimerStopped = FALSE;
            return 0.0f;
        }

        // Stop the timer
        if( command == TIMER_STOP )
        {
            m_llStopTime = qwTime.QuadPart;
            m_llLastElapsedTime = qwTime.QuadPart;
            m_bTimerStopped = TRUE;
            return 0.0f;
        }

        // Advance the timer by 1/10th second
        if( command == TIMER_ADVANCE )
        {
            m_llStopTime += m_llQPFTicksPerSec/10;
            return 0.0f;
        }

        if( command == TIMER_GETABSOLUTETIME )
        {
            fTime = qwTime.QuadPart / (double) m_llQPFTicksPerSec;
            return (FLOAT) fTime;
        }

        return -1.0f; // Invalid command specified
    }
    else
    {
        // Get the time using timeGetTime()
        static double m_fLastElapsedTime  = 0.0;
        static double m_fBaseTime         = 0.0;
        static double m_fStopTime         = 0.0;
        double fTime;
        double fElapsedTime;

        // Get either the current time or the stop time, depending
        // on whether we're stopped and what command was sent
        if( m_fStopTime != 0.0 && command != TIMER_START && command != TIMER_GETABSOLUTETIME)
            fTime = m_fStopTime;
        else
            fTime = timeGetTime() * 0.001;

        // Return the elapsed time
        if( command == TIMER_GETELAPSEDTIME )
        {
            fElapsedTime = (double) (fTime - m_fLastElapsedTime);
```

```
            m_fLastElapsedTime = fTime;
            return (FLOAT) fElapsedTime;
        }

        // Return the current time
        if( command == TIMER_GETAPPTIME )
        {
            return (FLOAT) (fTime - m_fBaseTime);
        }

        // Reset the timer
        if( command == TIMER_RESET )
        {
            m_fBaseTime         = fTime;
            m_fLastElapsedTime  = fTime;
            m_fStopTime         = 0;
            m_bTimerStopped     = FALSE;
            return 0.0f;
        }

        // Start the timer
        if( command == TIMER_START )
        {
            if( m_bTimerStopped )
                m_fBaseTime += fTime - m_fStopTime;
            m_fStopTime = 0.0f;
            m_fLastElapsedTime  = fTime;
            m_bTimerStopped = FALSE;
            return 0.0f;
        }

        // Stop the timer
        if( command == TIMER_STOP )
        {
            m_fStopTime = fTime;
            m_fLastElapsedTime  = fTime;
            m_bTimerStopped = TRUE;
            return 0.0f;
        }

        // Advance the timer by 1/10th second
        if( command == TIMER_ADVANCE )
        {
            m_fStopTime += 0.1f;
            return 0.0f;
        }

        if( command == TIMER_GETABSOLUTETIME )
        {
            return (FLOAT) fTime;
        }

        return -1.0f; // Invalid command specified
    }
}

//--------------------------------------------------------------------------
// Name: DXUtil_ConvertAnsiStringToWide()
// Desc: This is a UNICODE conversion utility to convert a CHAR string into a
//       WCHAR string. cchDestChar defaults -1 which means it
//       assumes strDest is large enough to store strSource
//--------------------------------------------------------------------------
VOID DXUtil_ConvertAnsiStringToWide( WCHAR* wstrDestination, const CHAR* strSource,
                                     int cchDestChar )
{
    if( wstrDestination==NULL || strSource==NULL )
        return;

    if( cchDestChar == -1 )
        cchDestChar = strlen(strSource)+1;

    MultiByteToWideChar( CP_ACP, 0, strSource, -1,
                         wstrDestination, cchDestChar-1 );

    wstrDestination[cchDestChar-1] = 0;
}

//--------------------------------------------------------------------------
// Name: DXUtil_ConvertWideStringToAnsi()
// Desc: This is a UNICODE conversion utility to convert a WCHAR string into a
//       CHAR string. cchDestChar defaults -1 which means it
//       assumes strDest is large enough to store strSource
//--------------------------------------------------------------------------
VOID DXUtil_ConvertWideStringToAnsi( CHAR* strDestination, const WCHAR* wstrSource,
                                     int cchDestChar )
{
    if( strDestination==NULL || wstrSource==NULL )
        return;

    if( cchDestChar == -1 )
        cchDestChar = wcslen(wstrSource)+1;

    WideCharToMultiByte( CP_ACP, 0, wstrSource, -1, strDestination,
                         cchDestChar-1, NULL, NULL );

    strDestination[cchDestChar-1] = 0;
}
```

```
//-----------------------------------------------------------------------------
// Name: DXUtil_ConvertGenericStringToAnsi()
// Desc: This is a UNICODE conversion utility to convert a TCHAR string into a
//       CHAR string. cchDestChar defaults -1 which means it
//       assumes strDest is large enough to store strSource
//-----------------------------------------------------------------------------
VOID DXUtil_ConvertGenericStringToAnsi( CHAR* strDestination, const TCHAR* tstrSource,
                                        int cchDestChar )
{
    if( strDestination==NULL || tstrSource==NULL || cchDestChar == 0 )
        return;

#ifdef _UNICODE
    DXUtil_ConvertWideStringToAnsi( strDestination, tstrSource, cchDestChar );
#else
    if( cchDestChar == -1 )
    {
        strcpy( strDestination, tstrSource );
    }
    else
    {
        strncpy( strDestination, tstrSource, cchDestChar );
        strDestination[cchDestChar-1] = '\0';
    }
#endif
}

//-----------------------------------------------------------------------------
// Name: DXUtil_ConvertGenericStringToWide()
// Desc: This is a UNICODE conversion utility to convert a TCHAR string into a
//       WCHAR string. cchDestChar defaults -1 which means it
//       assumes strDest is large enough to store strSource
//-----------------------------------------------------------------------------
VOID DXUtil_ConvertGenericStringToWide( WCHAR* wstrDestination, const TCHAR* tstrSource,
                                        int cchDestChar )
{
    if( wstrDestination==NULL || tstrSource==NULL || cchDestChar == 0 )
        return;

#ifdef _UNICODE
    if( cchDestChar == -1 )
    {
        wcscpy( wstrDestination, tstrSource );
    }
    else
    {
        wcsncpy( wstrDestination, tstrSource, cchDestChar );
        wstrDestination[cchDestChar-1] = L'\0';
    }
#else
    DXUtil_ConvertAnsiStringToWide( wstrDestination, tstrSource, cchDestChar );
#endif
}

//-----------------------------------------------------------------------------
// Name: DXUtil_ConvertAnsiStringToGeneric()
// Desc: This is a UNICODE conversion utility to convert a CHAR string into a
//       TCHAR string. cchDestChar defaults -1 which means it
//       assumes strDest is large enough to store strSource
//-----------------------------------------------------------------------------
VOID DXUtil_ConvertAnsiStringToGeneric( TCHAR* tstrDestination, const CHAR* strSource,
                                        int cchDestChar )
{
    if( tstrDestination==NULL || strSource==NULL || cchDestChar == 0 )
        return;

#ifdef _UNICODE
    DXUtil_ConvertAnsiStringToWide( tstrDestination, strSource, cchDestChar );
#else
    if( cchDestChar == -1 )
    {
        strcpy( tstrDestination, strSource );
    }
    else
    {
        strncpy( tstrDestination, strSource, cchDestChar );
        tstrDestination[cchDestChar-1] = '\0';
    }
#endif
}

//-----------------------------------------------------------------------------
// Name: DXUtil_ConvertAnsiStringToGeneric()
// Desc: This is a UNICODE conversion utility to convert a WCHAR string into a
//       TCHAR string. cchDestChar defaults -1 which means it
//       assumes strDest is large enough to store strSource
//-----------------------------------------------------------------------------
VOID DXUtil_ConvertWideStringToGeneric( TCHAR* tstrDestination, const WCHAR* wstrSource,
                                        int cchDestChar )
{
    if( tstrDestination==NULL || wstrSource==NULL || cchDestChar == 0 )
        return;

#ifdef _UNICODE
    if( cchDestChar == -1 )
    {
```

```
        wcscpy( tstrDestination, wstrSource );
    }
    else
    {
        wcsncpy( tstrDestination, wstrSource, cchDestChar );
        tstrDestination[cchDestChar-1] = L'\0';
    }
#else
    DXUtil_ConvertWideStringToAnsi( tstrDestination, wstrSource, cchDestChar );
#endif
}

//-----------------------------------------------------------------------------
// Name: _DbgOut()
// Desc: Outputs a message to the debug stream
//-----------------------------------------------------------------------------
HRESULT _DbgOut( TCHAR* strFile, DWORD dwLine, HRESULT hr, TCHAR* strMsg )
{
    TCHAR buffer[256];
    wsprintf( buffer, _T("%s(%ld): "), strFile, dwLine );
    OutputDebugString( buffer );
    OutputDebugString( strMsg );

    if( hr )
    {
        wsprintf( buffer, _T("(hr=%08lx)\n"), hr );
        OutputDebugString( buffer );
    }

    OutputDebugString( _T("\n") );

    return hr;
}

//-----------------------------------------------------------------------------
// Name: DXUtil_Trace()
// Desc: Outputs to the debug stream a formatted string with a variable-
//       argument list.
//-----------------------------------------------------------------------------
VOID DXUtil_Trace( TCHAR* strMsg, ... )
{
#if defined(DEBUG) | defined(_DEBUG)
    TCHAR strBuffer[512];

    va_list args;
    va_start(args, strMsg);
    _vsntprintf( strBuffer, 512, strMsg, args );
    va_end(args);

    OutputDebugString( strBuffer );
#else
    UNREFERENCED_PARAMETER(strMsg);
#endif
}

//-----------------------------------------------------------------------------
// Name: DXUtil_ConvertStringToGUID()
// Desc: Converts a string to a GUID
//-----------------------------------------------------------------------------
BOOL DXUtil_ConvertStringToGUID( const TCHAR* strIn, GUID* pGuidOut )
{
    UINT aiTmp[10];

    if( _stscanf( strIn, TEXT("{%8X-%4X-%4X-%2X%2X-%2X%2X%2X%2X%2X%2X}"),
                  &pGuidOut->Data1,
                  &aiTmp[0], &aiTmp[1],
                  &aiTmp[2], &aiTmp[3],
                  &aiTmp[4], &aiTmp[5],
                  &aiTmp[6], &aiTmp[7],
                  &aiTmp[8], &aiTmp[9] ) != 11 )
    {
        ZeroMemory( pGuidOut, sizeof(GUID) );
        return FALSE;
    }
    else
    {
        pGuidOut->Data2       = (USHORT) aiTmp[0];
        pGuidOut->Data3       = (USHORT) aiTmp[1];
        pGuidOut->Data4[0]    = (BYTE) aiTmp[2];
        pGuidOut->Data4[1]    = (BYTE) aiTmp[3];
        pGuidOut->Data4[2]    = (BYTE) aiTmp[4];
        pGuidOut->Data4[3]    = (BYTE) aiTmp[5];
        pGuidOut->Data4[4]    = (BYTE) aiTmp[6];
        pGuidOut->Data4[5]    = (BYTE) aiTmp[7];
        pGuidOut->Data4[6]    = (BYTE) aiTmp[8];
        pGuidOut->Data4[7]    = (BYTE) aiTmp[9];
        return TRUE;
    }
}

//-----------------------------------------------------------------------------
// Name: DXUtil_ConvertGUIDToString()
// Desc: Converts a GUID to a string
//-----------------------------------------------------------------------------
VOID DXUtil_ConvertGUIDToString( const GUID* pGuidIn, TCHAR* strOut )
{
    _stprintf( strOut, TEXT("{%0.8X-%0.4X-%0.4X-%0.2X%0.2X-%0.2X%0.2X%0.2X%0.2X%0.2X%0.2X}"),
```

```
                pGuidIn->Data1, pGuidIn->Data2, pGuidIn->Data3,
                pGuidIn->Data4[0], pGuidIn->Data4[1],
                pGuidIn->Data4[2], pGuidIn->Data4[3],
                pGuidIn->Data4[4], pGuidIn->Data4[5],
                pGuidIn->Data4[6], pGuidIn->Data4[7] );
}
```

Файл StorageMaster\objstock.cpp

```
/*****************************************************************************/
/*****************************************************************************/
/***                                                                       ***/
/*** OBJSTOCK.CPP                                                          ***/
/*** ╬╩╚═╚═╠ ╠╣╔╬╔═╬┬ ╬╗╚═╦└ CSTORAGEMASTER ╚ ┬┌═╬╠╠├└╗┌╗▄═▀F ╬╗└═╬┌┬╬┬ ***/
/***                                                                       ***/
/*** └└╔╦ ═└╩╚═└═ ╥└╗╙╙╚═╬┬▀╠ └══╙┼┼╠ ┌┼╙┼┼┼┬╚╬┼╠ ┬ 2002 ├╬═╙          ***/
/***                                                                       ***/
/***                                                                       ***/
/*****************************************************************************/
/*****************************************************************************/

#include "objstock.h"

#define __PROGRAMMER "Xroft"
#define __MODULE_NAME "StorageMaster"

#include <exeption.h>

    #define DLL_NAME "StorageMaster"
    #define DLL_CLASS CStorageMaster
    #include "..\Common\Utilities\Module.c"

IConsole * Console = NULL;

/*****************************************************************************/

DLL_ONLOAD (loader)
    {
    Assert (loader);
    loader -> GetInterface (IID_IConsole, (void **) &Console);

    if (Console)
        {
        Console -> printf ("STM32: StorageMaster version 0.9 by Xroft.\n");
        }
    }

/*****************************************************************************/

DLL_INIT (loader)
    {
    }

/*****************************************************************************/

    ////////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////////
  ////////////////////////// CLASS CDATABLOCK ////////////////////////////////
 ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

/*****************************************************************************/

CDataBlock::CDataBlock ()
    {
    }

/*****************************************************************************/

CDataBlock::~CDataBlock ()
    {
    }

/*****************************************************************************/

void CDataBlock::Init (int size, int num)
    {
    Allocate (size, num);
    Ok = NULL;
    Dump = NULL;
    Save = NULL;
    Load = NULL;
    }

/*****************************************************************************/

int CDataBlock::RetMax ()
    {
    return myObjNum;
    }
```

```
/*************************************************************************/

int CDataBlock::RetObjSize ()
    {
    return myObjSize;
    }

/*************************************************************************/

void CDataBlock::InitFuncs (int (* ok)(char *), int (* dump)(char *, FILE *),
                            int (* save)(char *, FILE *), int (* load)(char *, FILE *))
    {
    Ok = ok;
    Dump = dump;
    Save = save;
    Load = load;
    }

/*************************************************************************/

int CDataBlock::SaveData (FILE * file)
    {
    if (Save != NULL)
        {
        for (int n = 0; n < myObjNum; n ++)
            {
            if (Save (RetObject (n), file) == 0) return 0;
            }
        }
    else
        {
        AssertT (Save != NULL, "Save function pointer = NULL, just bytes will be moved in file");
        fwrite (myPointer, sizeof (char), myObjNum * myObjSize, file);
        }
    return 1;
    }

/*************************************************************************/

int CDataBlock::LoadData (FILE * file)
    {
    if (Load != NULL)
        {
        for (int n = 0; n < myObjNum; n ++)
            {
            if (Load (RetObject (n), file) == 0) return 0;
            }
        }
    else
        {
        AssertT (0, "Load function pointer = NULL, just bytes will be moved from file");
        fread (myPointer, sizeof (char), myObjNum * myObjSize, file);
        }
    return 1;
    }

/*************************************************************************/

int CDataBlock::DumpData (FILE * file)
    {
    AssertT (Dump != NULL, "Can't dump data: function pointer = NULL");
    for (int n = 0; n < myObjNum; n ++)
        {
        fprintf (file, "\nDUMP OF OBJECT %d:\n\n", n);
        if (Dump (RetObject (n), file) == 0) return 0;
        }
    return 1;
    }

/*************************************************************************/

int CDataBlock::TestData ()
    {
    AssertT (Ok != NULL, "Can't test data: function pointer = NULL");
    for (int n = 0; n < myObjNum; n ++)
        {
        if (Ok (RetObject (n)) == 0) return 0;
        }
    return 1;
    }

/*************************************************************************/

    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////////  CLASS CHANDLETABLE  //////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

/*************************************************************************/

CHandleTable::CHandleTable ()
    {
    myNames = NULL;
    myHandles = NULL;
    }
```

```
/*************************************************************************/

CHandleTable::CHandleTable (int num) :
    myMaxNum (num),
    myCurNum (0)
    {
    myNames = new char * [num];
    myHandles = new int [num];
    }

/*************************************************************************/

CHandleTable::~CHandleTable ()
    {
    delete [] myNames;
    delete [] myHandles;
    }

/*************************************************************************/

void CHandleTable::Init (int num)
    {
    myMaxNum = num;
    myCurNum = 0;
    myNames = new char * [num];
    myHandles = new int [num];
    }

/*************************************************************************/

int CHandleTable::ok ()
    {
    return myCurNum < myMaxNum;
    }

/*************************************************************************/

int CHandleTable::ok (int n)
    {
    return n < myCurNum;
    }

/*************************************************************************/

void CHandleTable::Add (int handle, char * name)
    {
    AssertT (ok (), "There is no place in handle table");
    myHandles [myCurNum] = handle;
    myNames [myCurNum] = new char [strlen (name) + 1];
    memcpy (myNames [myCurNum], name, strlen (name) + 1);
    myCurNum ++;
    }

/*************************************************************************/

int CHandleTable::RetHandle (char * name)
    {
    for (int n = 0; n < myCurNum; n ++)
        {
        for (int i = 0; ; i ++)
            {
            if (myNames [n] [i] != name [i]) break;
            else if (name [i] == 0) return n;
            }
        }
    return -1;
    }

/*************************************************************************/

char * CHandleTable::RetName (int handle)
    {
    for (int n = 0; n < myCurNum; n ++)
        {
        if (myHandles [n] == handle)
            return myNames [n];
        }
    return NULL;
    }

/*************************************************************************/
    ////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////
  ////////////////////// CLASS CSTORAGEMANAGER ///////////////////////////
 ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

/*************************************************************************/

CStorageMaster::CStorageMaster () :
    myRefs (0)
    {
    myData = NULL;
    myPointers = NULL;
    }
```

```
/*************************************************************************/

CStorageMaster::CStorageMaster (int maxblocks) :
    myMaxBlocks (maxblocks),
    myCurBlock (0),
    myRefs (0)
    {
    myTable.Init (maxblocks);
    myData = new CDataBlock [maxblocks];
    myPointers = new CSmartPointer [maxblocks];
    AssertT (!!myData, "Out of memory");
    }

/*************************************************************************/

CStorageMaster::~CStorageMaster ()
    {
    delete [] myData;
    delete [] myPointers;
    }

/*************************************************************************/

HRESULT __stdcall CStorageMaster::QueryInterface (const IID &iid, void ** ptr)
    {
    if (iid == IID_IUnknown)
        {
        *ptr = static_cast <IStorageMaster *> (this);
        }
    else
        {
        if (iid == IID_IStorageMaster)
            {
            *ptr = static_cast <IStorageMaster *> (this);
            }
        else
            {
            *ptr = NULL;
            return E_NOINTERFACE;
            }
        }
    reinterpret_cast <IUnknown *> (* ptr)->AddRef ();
    return S_OK;
    }

/*************************************************************************/

ULONG __stdcall CStorageMaster::AddRef ()
    {
    return InterlockedIncrement (&myRefs);
    }

/*************************************************************************/

ULONG __stdcall CStorageMaster::Release ()
    {
    if (InterlockedDecrement (&myRefs) == 0)
        {
        delete this;
        return 0;
        }
    return myRefs;
    }

/*************************************************************************/

void CStorageMaster::Init (int maxblocks)
    {
    myMaxBlocks = maxblocks;
    myCurBlock = 0;
    myData = new CDataBlock [maxblocks];
    myPointers = new CSmartPointer [maxblocks];
    myTable.Init (maxblocks);
    AssertT (!!myData, "Out of memory");
    }

/*************************************************************************/

int CStorageMaster::ok ()
    {
    return myCurBlock < myMaxBlocks;
    }

/*************************************************************************/

int CStorageMaster::ok (int n)
    {
    return n < myCurBlock;
    }

/*************************************************************************/

int CStorageMaster::SetNewDataType (int size, int num, char * name)
    {
    AssertT (ok (), "Can't initialize new data type: no place");
    AssertT (myTable.RetHandle (name) == -1, "Can't create data type with existing name");
    myData [myCurBlock].Init (size, num);
```

```
    myPointers [myCurBlock] = *((CMemoryBlock *) &myData [myCurBlock]);
    myTable.Add (myCurBlock, name);
    myCurBlock ++;
    return myCurBlock - 1;
    }

/*************************************************************************/

void CStorageMaster::InitFuncs (int handle, int (* test)(char *), int (* dump)(char *, FILE *),
                                int (* save)(char *, FILE *), int (* load)(char *, FILE *))
    {
    AssertT (ok (handle), "Handle is out of range");
    myData [handle].InitFuncs (test, dump, save, load);
    }

/*************************************************************************/

void CStorageMaster::RemoveDataType (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    for (int n = handle; n < myCurBlock - 1; n ++)
        myData [n] = myData [n + 1];
    myCurBlock --;
    }

/*************************************************************************/

int CStorageMaster::AddObject (int handle, void * obj)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myData [handle].AddObject (obj);
    }

/*************************************************************************/

CSmartPointer & CStorageMaster::RetData (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myPointers [handle];
    }

/*************************************************************************/

void CStorageMaster::RemoveObject (int handle, int n)
    {
    AssertT (ok (handle), "Handle is out of range");
    myData [handle].RemoveObject (n);
    }

/*************************************************************************/

char * CStorageMaster::RetObject (int handle, int nobj)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myData [handle].RetObject (nobj);
    }

/*************************************************************************/

char * CStorageMaster::CreateObject (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myData [handle].CreateObject ();
    }

/*************************************************************************/

char * CStorageMaster::CreateObjects (int handle, int num)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myData [handle].CreateObjects (num);
    }

/*************************************************************************/

int CStorageMaster::RetMaxObjects (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myData [handle].RetObjNum ();
    }

/*************************************************************************/

void CStorageMaster::Lock (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    myData [handle].Lock ();
    }

/*************************************************************************/

void CStorageMaster::Unlock (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    myData [handle].Unlock ();
    }
```

```
/*************************************************************************/

int CStorageMaster::DumpData (int handle, char * fname)
    {
    AssertT (ok (handle), "Handle is out of range");
    FILE * file = fopen (fname, "wb");
    AssertT (!!file, "Can't open file");
    int res = myData [handle].DumpData (file);
    fclose (file);
    return res;
    }

/*************************************************************************/

int CStorageMaster::SaveData (int handle, char * fname)
    {
    AssertT (ok (handle), "Handle is out of range");
    FILE * file = fopen (fname, "wb");
    AssertT (!!file, "Can't open file");
    int res = myData [handle].SaveData (file);
    fclose (file);
    return res;
    }

/*************************************************************************/

int CStorageMaster::LoadData (int handle, char * fname)
    {
    AssertT (ok (handle), "Handle is out of range");
    FILE * file = fopen (fname, "wb");
    AssertT (!!file, "Can't open file");
    int res = myData [handle].LoadData (file);
    fclose (file);
    return res;
    }

/*************************************************************************/

int CStorageMaster::TestData (int handle)
    {
    AssertT (ok (handle), "Handle is out of range");
    return myData [handle].TestData ();
    }

/*************************************************************************/

int CStorageMaster::SaveXXX (int handle, char * filename, char type)
    {
    AssertT (ok (handle), "Handle is out of range");
    FILE * file;
    if (type == 'n') file = fopen (filename, "wb");
    else {
         if (type == 'a') file = fopen (filename, "ab");
         else file = fopen (filename, "wb");
         }
    AssertT (file != NULL, "Cannot open file");
    putc ('[', file);
    putc ('@', file);
    fprintf (file, "%s", myTable.RetName (handle));
    putc ('#', file);
    putw ((short) RetMaxObjects (handle), file);
    putc ('$', file);
    putw ((short) myData [handle].RetObjSize (), file);
    putc (':', file);
    for (int n = 0; n < RetMaxObjects (handle); n ++)
        {
        for (int i = 0; i < myData [handle].RetObjSize (); i ++)
            {
            if (RetObject (handle, n) != NULL) putc (RetObject (handle, n) [i], file);
            }
        if (RetObject (handle, n) != NULL) putc (';', file);
        }
    putc (']', file);
    fclose (file);
    return 1;
    }

/*************************************************************************/

int CStorageMaster::LoadXXX (char * filename)
    {
    FILE * file = fopen (filename, "rb");
    AssertT (file != NULL, "Cannot open file");
    char dataname [50];
    short size, num;
    for (;;)
        {
        if (getc (file) != '[') return 1;
        getc (file);
        for (int j = 0; j < 50; j ++)
            {
            char ch = getc (file);
            if (ch != '#') dataname [j] = ch;
            else
                {
                dataname [j] = 0;
                break;
```

```
                }
            }
        num = getw (file);
        getc (file);
        size = getw (file);
        int handle = myTable.RetHandle (dataname);
        if (handle == -1)
            handle = SetNewDataType (size, num, dataname);
        Lock (handle);
        getc (file);
        for (int n = 0; n < num; n ++)
            {
            char * obj = CreateObject (handle);
            for (int i = 0; i < myData [handle].RetObjSize (); i ++)
                {
                obj [i] = getc (file);
                }
            getc (file);
            }
        getc (file);
        Unlock (handle);
        }
    fclose (file);
    }

/*************************************************************************/

int CStorageMaster::RetHandle (char * name)
    {
    return myTable.RetHandle (name);
    }

/*************************************************************************/

void CStorageMaster::InitFuncs (char * name, int (* test)(char *), int (* dump)(char *, FILE *),
                                int (* save)(char *, FILE *), int (* load)(char *, FILE *))
    {
    InitFuncs (myTable.RetHandle (name), test, dump, save, load);
    }

/*************************************************************************/

void CStorageMaster::RemoveDataType (char * name)
    {
    RemoveDataType (myTable.RetHandle (name));
    }

/*************************************************************************/

int CStorageMaster::AddObject (char * name, void * obj)
    {
    return AddObject (myTable.RetHandle (name), obj);
    }

/*************************************************************************/

CSmartPointer & CStorageMaster::RetData (char * name)
    {
    return RetData (myTable.RetHandle (name));
    }

/*************************************************************************/

void CStorageMaster::RemoveObject (char * name, int n)
    {
    RemoveObject (myTable.RetHandle (name), n);
    }

/*************************************************************************/

char * CStorageMaster::RetObject (char * name, int nobj)
    {
    return RetObject (myTable.RetHandle (name), nobj);
    }

/*************************************************************************/

char * CStorageMaster::CreateObject (char * name)
    {
    return CreateObject (myTable.RetHandle (name));
    }

/*************************************************************************/

char * CStorageMaster::CreateObjects (char * name, int num)
    {
    return CreateObjects (myTable.RetHandle (name), num);
    }

/*************************************************************************/

int CStorageMaster::RetMaxObjects (char * name)
    {
    return RetMaxObjects (myTable.RetHandle (name));
    }

/*************************************************************************/
```

```
void CStorageMaster::Lock (char * name)
    {
    Lock (myTable.RetHandle (name));
    }

/*************************************************************************/

void CStorageMaster::Unlock (char * name)
    {
    Unlock (myTable.RetHandle (name));
    }

/*************************************************************************/

int CStorageMaster::DumpData (char * name, char * fname)
    {
    return DumpData (myTable.RetHandle (name), fname);
    }

/*************************************************************************/

int CStorageMaster::SaveData (char * name, char * fname)
    {
    return SaveData (myTable.RetHandle (name), fname);
    }

/*************************************************************************/

int CStorageMaster::LoadData (char * name, char * fname)
    {
    return LoadData (myTable.RetHandle (name), fname);
    }

/*************************************************************************/

int CStorageMaster::TestData (char * name)
    {
    return TestData (myTable.RetHandle (name));
    }

/*************************************************************************/

int CStorageMaster::SaveXXX (char * name, char * filename, char type)
    {
    return SaveXXX (myTable.RetHandle (name), filename, type);
    }

/*************************************************************************/

/*************************************************************************/
/*╧ююыєўш№ ¤ъчхьяы Ё юс·хъЄр CStorageMaster*/

void GetInstance (void ** ptr)
    {
    IUnknown * pUnknown = static_cast <IStorageMaster *> (new CStorageMaster);
    pUnknown -> AddRef ();
    *ptr = pUnknown;
    }

/*************************************************************************/
```

Файл SystemExtentions1\Console.cpp

```
// Console.cpp - implementation of Console.h, of module SmallCompiler
// created 14.10.2002 by Vasily O. Fedoseyev in Microsoft Development Environment version 7.00

#define __PROGRAMMER   "Se"
#define __MODULE_NAME  "SmallCompiler"

#include "System.h"
#include "Console.h"
#include "..\Console\Console.h"
#include "..\ExeptionEngine\Exeption.h"

void CommandNewVar (IConsole* c, int ln, char* data)
{
    if ( Linker->getGlobalScope().check ( Linker->registerName ( data)) == 1) {
        Console->printf ( "variable with name \"%s\" already exists", data);
        }
    Linker->getGlobalScope().addVar ( CVar ( CObjRef (), CString ( data)));
}

void AddCommands ()
{
    //Assert (Console);
    Console->AddCommand ( "NewVar", (CommandProcP) CommandNewVar, false);
}
```

Файл SystemExtentions1\ExternCore.cpp

```
#include "ExternCore.h"
#include "system.h"
#include "console.h"

SClassRef CExtern::sClass = SClassRef ( 0);

CExtern::CExtern()
    {
        myClass = sClass;
    }

SParam CExtern::NewVar ( const CParams& par)
    {
        Assert ( par.num == 1);
        CommandNewVar ( Console, 1, (char*)par[0].rString());
        return Linker->getGlobalScope().gVar ( Linker->registerName ( par[0].rString()));
    }

SParam/*self*/ CExtern::cmd ( const CParams& par) // 0 - eStr
    {
        Assert ( par.num == 1);
        Console->RaiseCommand ( par[0].rString());
        return CObjRef ( Extern);
    }

/*eFloat*/SParam CExtern::Random ( const CParams& par) // 1 - eFloat
    {
        return ( (float)rand()) * par[0].rFloat() / RAND_MAX;
    }

/*eInt*/SParam CExtern::GetSystemTimer ( const CParams& par) // 0
    {
        return (int) timeGetTime();
    }

bool CExtern::selfReg()
    {
        sClass = CPPVM->registerClass ("Extern", "Extern");

        CPPVM->registerStaticFunc ( sClass, "NewVar", (pStatFunc) CExtern::NewVar, (TYPES SParam::eStr));
        CPPVM->registerStaticFunc ( sClass, "cmd:", (pStatFunc) CExtern::cmd, (TYPES SParam::eStr));
        CPPVM->registerStaticFunc ( sClass, "Random:", (pStatFunc) CExtern::Random, (TYPES
SParam::eFloat));
        CPPVM->registerStaticFunc ( sClass, "GetSystemTimer", (pStatFunc) CExtern::GetSystemTimer,
CParamsTypes());

        return 1;
    }

IMPLEMENT_IUNKNOWN (CExtern, IExtern, IID_IExtern);
```

Файл SystemExtentions1\Module.cpp

```
#define __PROGRAMMER   "Se"
#define __MODULE_NAME  "SystemExtentions"

#include "StdAfx.h"      // use precompiled header
#include "System.h"
//#include "Module.h"

#include "Console.h"
#include "Extern.h"
#include "ExternCore.h"

#include <Exeption.h>      // use ExeptionEngine

#include "..\Common\Utilities\Module.h"

    #define DLL_NAME "SystemExtentions1"
    #define DLL_CLASS CExtern
    #include "..\Common\Utilities\Module.c"

DLL_INIT (loader)
    {
        Assert (loader);

        LinkerDataTypesLibInit ( loader);

        loader->GetInterface ( IID_ILinker, (void**)&Linker);
        Assert ( Linker);
        loader->GetInterface ( IID_ICPPVM, (void**)&CPPVM);
        Assert ( CPPVM);
        loader->GetInterface ( IID_ISmallVM, (void**)&SmallVM);
        Assert ( SmallVM);
        loader->GetInterface ( IID_IConsole, (void**)&Console);
        Assert ( Console);
```

```
        Extern = &DLL_OBJECT;

        Extern->incRefs();

        Assert ( CExtern::selfReg());
        AddCommands();
    }

DLL_ONLOAD (loader)
{
    IConsole* con = NULL;
    loader->GetInterface (IID_IConsole, (void**)&con);
    Assert (con);
    con->printf ("SYSEX1: System Extensions Set 1 by Se&Vasfed version 1.0\n");
}
```

Файл SystemExtentions1\system.cpp

```
#include "system.h"
#include "externcore.h"

ILinker*  Linker  = 0;
ICPPVM*   CPPVM   = 0;
ISmallVM* SmallVM = 0;
IConsole* Console = 0;
CExtern*  Extern  = 0;
```

Файл Utilities\Utilities.cpp

```
// Utilities.cpp : Defines the entry point for the DLL application.
//

#define __PROGRAMMER "Vasfed"
#define __MODULE_NAME "Utilities"

#include "stdafx.h"
#include "Utilities.h"
#include <Exeption.h>

#include <stdlib.h>
#include <process.h>
#include <stdio.h>
#include <errno.h>
#include <mmsystem.h>
#define _USE_MATH_DEFINES
#include <math.h>

//Entry-point for DLL
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    return TRUE;
}

//Vector::
            Vector::Vector (float dx, float dy):x(dx), y (dy){}
Vector      Vector::operator +(Vector op2)
    {return Vector (x + op2.x, y + op2.y);}
Vector      Vector::operator -(Vector op2)
    {return Vector (x - op2.x, y - op2.y);}
Vector&     Vector::operator += (Vector op2)
    {x += op2.x; y += op2.y; return *this;}
Vector&     Vector::operator -= (Vector op2)
    {x -= op2.x; y -= op2.y; return *this;}
Vector      Vector::operator *(float number)
    {return Vector (x * number, y * number);}
Vector      Vector::operator /(float number)
    {AssertC (number != 0, badParams); return Vector (x / number, y / number);}
Vector& Vector::operator *=(float number)
    {x *= number; y *= number; return *this;}
Vector& Vector::operator /=(float number)
    {AssertC (number != 0, badParams); x /= number; y /= number; return *this;}

Vector::operator == (Vector v2)
{
    return ((v2.x == this->x) && (v2.y == this->y));
}

Vector&     Vector::Normalize()
    {
    if (x < 1 && y < 1 && x > -1 && y > -1) return *this;

    float x1 = x > 0? x:-x, y1 = y > 0? y:-y; //x1 = |x|, y1 = |y|
```

```
    if (x1 > y1)
        {y /= x1; x = x > 0? 1:-1;}
    else {x /= y1; y = y > 0? 1:-1;}
    return *this;
    }

Vector Vector::Rotare      (float angle, bool isDegrees)
{
    if (angle == 0) return *this;
    if (isDegrees)
        {
        angle *= M_PI;
        angle /= 180;
        }

    Vector temp = *this;


    float len = temp.Len ();
    if (temp.x != 0)
        angle += atan (temp.y / temp.x);
    temp.x = len * cos (angle);
    temp.y = len * sin (angle);
    return temp;
}

Vector& Vector::RotareIt   (float a, bool isDegrees) //Rotare that sets this vector to result
{
    if (a == 0) return *this;
    if (isDegrees)
        {
        a *= M_PI;
        a /= 180;
        }
    float len = Len();
    if (x!=0)
        a+= atan (y/x);
    x = cos (a) * len;
    y = sin (a) * len;
    return *this;
}

float Vector::Len ()
{
    return (float)sqrt (x*x + y*y);
}

void Vector::DrawTo (HDC dc, int xo, int yo, int zoom)
{
    POINT oldPos;
    MoveToEx (dc, xo, yo, &oldPos);
    int x2 = xo + x*zoom, y2 = yo + y*zoom;
    LineTo(dc, x2, y2);
    int dx = x > 0? 1:-1, dy = y > 0? 1:-1;
    LineTo(dc, x2 + dx, y2 + dy);
    MoveToEx (dc, x2, y2, NULL);
    LineTo(dc, x2 - dx, y2 + dy);
    MoveToEx (dc, oldPos.x, oldPos.y, NULL);
}
//end of Vector

//Matrix:

//CRectangle::
CRectangle::CRectangle (){}
CRectangle::CRectangle (int left, int top, int right, int bottom): x1(left), x2(right), y1(top),
y2(bottom){}
CRectangle::CRectangle (RECT* rect){AssertTC (rect, badParams, "CRectangle(NULL) called");}
void CRectangle::DrawTo (HDC dc, int x, int y, int zoom)
{
    AssertC (dc && zoom, badParams);
    Rectangle (dc, x1*zoom + x, y1*zoom + y, x2*zoom + x, y2*zoom + y);
}

//CThread
CThread::CThread (){StartThread();}
CThread::~CThread (){_endthread();}

void CThread::Main ()
{
    Message ("Pure Thread generated");
    _endthread ();
}

int CThread::StartThread()
{
    m_threadID = (int)_beginthread (this->ThreadEntryPoint, 0, (void*)this);
    if (m_threadID == -1L) //failed to create thread
        {
        if (errno == EAGAIN)
            Throw (runtime, "Failed to create thread agent: too many threads"));
        Throw (runtime, "Failed to create thread for agent: '%s'", sys_errlist[errno]));
        }
    return m_threadID;
}

void CThread::ThreadEntryPoint (void* This)
```

```
{
    ((CThread*)This)->Main();
}

void CThread::Init ()
{
    Message ("Init: pure vitrual call");
}
//---------------------------------------------------------------
void    randomize()
    {
    srand (timeGetTime ());
    }

float   random (float a)
{return (float) random ((double)a);}

double  random (double a)
{
    return ((((double)rand())*a)/RAND_MAX);
}

int     random (int a)
{
    AssertC (a != 0, badParams);
    return rand()%a;
}
//---------------------------------------------------------------

CBitMatrix::CBitMatrix (int cx, int cy)
{
    Assert (this);
    AssertC (cx > 2 && cy > 0, badParams);
    m_sizex = cx;
    m_sizey = cy;

    m_dataSize = ((cx*cy) / 8) + ((cx*cy)%8? 1:0);
    m_data = (unsigned char*)LocalAlloc (LMEM_FIXED, m_dataSize);
    AssertC (m_data, memoryLeak);
    ZeroMemory (m_data, m_dataSize);
}

bool CBitMatrix::FastGet   (unsigned int x, unsigned int y)
{
    void* data = m_data;
//  __asm {
//  mul

//  }
return false;
}

void CBitMatrix::FastSet   (unsigned int x, unsigned int y, bool val)
{

}

CBitMatrix::~CBitMatrix ()
{
    Assert (this);
    if (m_data)
        {
        LocalFree (m_data);
        m_data = NULL;
        }
}

bool CBitMatrix::operator ()(int x, int y)
{
    return this->Get (x, y);
}

bool CBitMatrix::Get (unsigned int x, unsigned int y)
{
    AssertC (x < m_sizex && y < m_sizey, badParams);
    Assert (m_data && this);
    int byte = (y*m_sizex + x);
    int bit = byte % 8;
    byte /= 8;
    return ((m_data[byte] & (1 << bit)) > 0);
}

void CBitMatrix::Set (unsigned int x, unsigned int y, bool value)
{
    AssertC (x < m_sizex && y < m_sizey, badParams);
    Assert (m_data && this);
    int byte = (y*m_sizex + x);
    int bit = byte % 8;
    byte /= 8;

    ((m_data[byte] & (1 << bit)) > 0);
    if (value) m_data [byte] |= (1 << bit);
    else m_data [byte] &= (0xff ^ (1 << bit));
}
```

Файл xxxcompiler\XXXCompiler.cpp

```
// XXXCompiler.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <exeption.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

int _tmain(int argc, _TCHAR* argv[])
{
CreateDumpFile ("xxxdump.txt");
try
    {
    AssertT (argc == 4, "Not equal number of parameters");
    FILE * infile  = fopen (argv [1], "rt");
    FILE * outfile;
    if (argv [3][0] == 'n')
        outfile = fopen (argv [2], "wb");
    else if (argv [3][0] == 'a')
          outfile = fopen (argv [2], "ab");
        else AssertT (0, "Illegal parameter of openinig output file");
    char name [50];
    int size = 0, num = 0;
    fscanf (infile, "datatype name: %s\n", name);
    fscanf (infile, "datatype number: %d\n", &num);
    fscanf (infile, "datatype size: %d\n", &size);
    putc ('[', outfile);
    putc ('@', outfile);
    fprintf (outfile, "%s", name);
    putc ('#', outfile);
    putw (num, outfile);
    putc ('$', outfile);
    putw (size, outfile);
    putc (':', outfile);
    fscanf (infile, "datatype data:");

    char type [10];
    int cursize = 0, totalsize = size * num;
    for (int n = 0; n < num; n ++)
        {
        fscanf (infile, "\n\t{\n");
        for ( ; ; )
            {
            fscanf (infile, "\t%s", type);
            if (type [0] == '}') break;
            if (strcmp (type, "float") == 0)
                {
                float f;
                char num [20];
                fscanf (infile, "%s\n", num);
                f = (float) strtod (num, NULL);
                fwrite (&f, sizeof (float), 1, outfile);
                cursize += 4;
                }
            else if (strcmp (type, "int") == 0)
                    {
                    int i;
                    fscanf (infile, "%d\n", &i);
                    fwrite (&i, sizeof (int), 1, outfile);
                    cursize += 4;
                    }
            }
        if (cursize > size)
            {
            printf ("Too many bytes in object number %d\n", n);
            getch ();
            }
        if (cursize < size)
            {
            printf ("Too little bytes in object number %d\n", n);
            getch ();
            }
        cursize = 0;
        putc (';', outfile);
        }
    putc (']', outfile);
    }
Catch (){}
return 0;
}
```

Приложение П2-а.
Описание интерфейсов компонентов

Файл IAgent.h - базовый интерфейс для Агентов

```
// IAgent.h - base interfaces for AI
// created 10.09.2001 by header creator

#pragma once
#ifndef __AGENTS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __AGENTS_HEADER_FILE_BY_VASFED_INCLUDED__


#include <objbase.h>
#include "MessageLanguages.h"

// {935FB5CA-D30C-4b5f-8236-7A0637D5173E}

static const GUID IID_IAgentsFactory =
    { 0x935fb5ca, 0xd30c, 0x4b5f, { 0x82, 0x36, 0x7a, 0x6, 0x37, 0xd5, 0x17, 0x3e } };

class agentPassTurn {}; // agent exception

//----------------------------
//additional structures and classes:

struct AgentMessage
{
   int    sender;       // Agent who send this message (handle in Server)
   int    receiver; // 0-> depends on type: broadcast, or local (to nearest agent...)
   int    type;     // Serving type = Broadcast, Private etc. see MessageCodes
   int    ainame;       // - who can understand it? (target AI name)
   int    code;     // user-defined message type

   float  x;
   float  y;
   float  z;

   void*     data;  // anything, deleted, if freedata is 1
   int    data1;    // anything
   int    datasize; // size of data pointed by data
   bool      freedata; // indictes, if *data should be freed by deleting it

   enum Types {zMT_NONE =0, zMT_BROADCAST =1, zMT_SOUND =2, zMT_TARGETED =3, zMT_TOUSER =10};
};


interface IAgent
{
   virtual int    Go()  PURE;
   virtual int    Ok()  PURE;
   virtual void   Init()   PURE;
   virtual int    MessageProc       (AgentMessage* msg)  PURE;
   virtual void   CollisionDetected (void* reserved)     {};

   float  m_x, m_y, m_z;

   bool      m_shouldBeProcessed;
   bool      m_ShouldExit;

   //Handles (often all the hashes are similar)

   int    m_index;      //in AgentsServer
```

```
    int    m_mesh;       //in Renderer
    int    m_physHash;   //in Physics module

    int    m_ainame; // to make different AI recognizeable
};

#endif //__AGENTS_HEADER_FILE_BY_VASFED_INCLUDED
```

Файл AgentsServer.h - интерфейс Сервера агентов

```
    // AgentsServer.h - header
    // created 10.09.2002  by header creator

    #pragma once
    #ifndef __AGENTSSERVER_HEADER_FILE_BY_VASFED_INCLUDED__
    #define __AGENTSSERVER_HEADER_FILE_BY_VASFED_INCLUDED__

    #include "Objbase.h"
    #include "..\AICore\Agents.h"
    #include <stdio.h>
    #include "Agents.h"

    // {E0F58370-14AC-4a17-9B6B-2B620B105F17}

    static const GUID IID_IAgentsServer =
    { 0xe0f5837O, 0x14ac, 0x4a17, { 0x9b, 0x6b, 0x2b, 0x62, 0xb, 0x10, 0x5f, 0x17 } };


    typedef IAgent* (*PAgentCreator)(int);


    interface IAgentsServer: public IUnknown
    {
       virtual int      ok         (void)    PURE;
       virtual void     Terminate  (void)    PURE;

       virtual int      AddAgent      (IAgent*, float x = 0, float y = 0, float z = 0)
    PURE;
       virtual int      RemoveAgent   (int)     PURE;
       virtual IAgent*  GetAgent      (int)     PURE;
       virtual int      FindAgent  (IAgent*)  PURE;
       virtual void     ReHash     (void)    PURE;
       virtual int      GetNumAgents  (void)    PURE;

       virtual int      RegisterAgentCreator (PAgentCreator func, char* ainame = "un-
    known",
                                    char* programmer = "unknown", int reserved=0)
                                    PURE;

       virtual int      RegisterAgentCreator (char* reserved, char* reserved1 = "unknown",
                                    char* reserved2 = "unknown")
                                    PURE;

       virtual int      UnRegisterAgentCreator (int index)    PURE;
       virtual int      FindAgentCreator       (PAgentCreator)PURE;
       virtual int      FindAgentCreator       (char* command)PURE;
       virtual int      FindAgentCreatorByName (char* ainame)    PURE;
       virtual int      GetMaxCreators         (void)            PURE;

       virtual int      CreateAgent            (int agentCreator = 0,
                                       float x = 0, float y = 0, float z = 0,
                                       int misc = 0) PURE;

       virtual void     PerformAgentsStep  (void)            PURE;
       virtual void     InitializeAgents       (void)            PURE;

       virtual void     SetAgentPosition       (int agent,
                                       float x = 10, float y = 10, float z = 10)
                                       PURE;

       virtual IAgentsServerAgents*GetAgentsInterface  (void)    PURE;

    //added by Se - 11.09.1002
```

```
      virtual const char*  GetAgentCreatorName     (int ai)      PURE;
   };

#endif //__AGENTSSERVER_HEADER_FILE_BY_VASFED_INCLUDED__
```

Файл AgentsServerAgents.h - взаимодействие между агентами

```
// AgentsServerAgents.h - header
// created 21.10.2001 Vasfed by header creator

#pragma once
#ifndef __AGENTSSERVERAGENTS_HEADER_FILE_BY_VASFED_INCLUDED__
#define __AGENTSSERVERAGENTS_HEADER_FILE_BY_VASFED_INCLUDED__

#include <objbase.h>
#include "..\AICore\Agents.h"

/*
This part AgentServer is visible to agents
and is the gate from multiagent system to other system.

This interface can be taken only from IAgentsServer::GetAgentsInterface
*/



// {C71528C6-E54C-4fba-9F59-4238EB7AFE80}

static const GUID IID_IAgentsServerAgents =
   { 0xc71528c6, 0xe54c, 0x4fba, { 0x9f, 0x59, 0x42, 0x38, 0xeb, 0x7a, 0xfe, 0x80 } };



interface IAgentsServerAgents
{
   //Communication between agents:
   virtual IAgent*      GetAgent            (int)                    PURE;
   virtual int          SendMessage         (AgentMessage)              PURE;

   //Map:
   virtual inline int   GetMapUnitCode      (float x, float y) const       PURE;
   virtual inline int*  GetPMapUnitData     (float x, float y) const       PURE;
   virtual inline int   GetMapUnitData      (float x, float y, int index = 0) const PURE;
   virtual inline int   GetMapUnitDataSize  (float x, float y) const       PURE;
   virtual inline void  ModifyMapUnitData(float x, float y, int index,
                                 int newvalue)             PURE;

   virtual inline int   GetMapSizeX         (void) const PURE;
   virtual inline int   GetMapSizeY         (void) const PURE;
   virtual inline float GetMapSquareSize    (void) const PURE;
   virtual float    GetLandscapeLevel    (float x, float y, int reserved = 0)  PURE;

   //Physics
   virtual void     SetSpeed            (int agent, float x, float y, float z = 0) PURE;
   virtual void     SetAcceleration     (int agent, float x, float y, float z = 0) PURE;
   virtual void     GetSpeed            (int agent, float*x, float*y, float*z = 0) PURE;
   virtual void     GetAcceleration     (int agent, float*x, float*y, float*z = 0) PURE;
   virtual float GetFallingAcceleration (void)                                PURE;
   virtual void     AddToAcceleration(int agent, float x, float y, float z = 0) PURE;
   virtual void     AddToSpeed       (int agent, float x, float y, float z = 0) PURE;
   virtual void     SetAgentPosition     (int agent,
                                 float x = 10, float y = 10, float z = 10) PURE;

   //User interface
   //(not all implemented yet)
   virtual void     PlaySound        (int agent, int soundcode)            PURE;
   virtual void     PlaySound        (int agent, char* soundname)             PURE;
   virtual void     PlayAnimation    (int agent, int animation, void* params) PURE;
   virtual void     SayToUser        (int agent, char* str)                PURE;
   virtual void     SayToUser        (char* format, ...)                   PURE;
};

#endif //__AGENTSSERVERAGENTS_HEADER_FILE_BY_VASFED_INCLUDED__
```

Приложение П2-б.
Исходные тексты на языке Smalltalk

Файл Scr\1.smt

```
Smalltalk interchangeVersion: '1.0' !

Class named: 'ExCl'
    " superclass: <superclassNameString>"
    " indexedInstanceVariables: <indexableInstVarType>"
    instanceVariableNames: 'instVar1 instVar2'
    classVariableNames: 'clVar1 clVar2'
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
    !

ExCl method !
    getX
    | t1 t2 t3 |

    "t1 := 1 + 2;
          +10. t1 = 11. "
    Transcript cr.
    Transcript nextPut: $c.
    Transcript cr.

    Transcript nextPutAll: 'First string is out as a whole string and not by single letters';cr;
          nextPutAll: 'Second string is out as a whole string and not by single letters';cr.

    t1 := 1.
    t2 := 2.
    t3 := 3.
    instVar1 := 10.
    instVar2 := 20.
clVar1 := 10.
Transcript nextPutAll: clVar1 asString.

    ^( t1 + t2 + t3 * instVar1 / instVar2 )
    !
```

Файл Scr\1Pimp.smt

```
Smalltalk interchangeVersion: '1.0' !

Class named: 'Pimp'
    instanceVariableNames: 'Me m_DestX m_DestY m_ovx m_ovy m_BestH tvx tvy tvz ready'
    classVariableNames: ''
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
    !

Pimp method !
    rData

    Transcript nextPutAll: 'Pimp rData called';cr.

    ^ Me
    !

Pimp method !
    InitData: Data

    Me := Data.

    ^1
    !

Pimp method !
    Go

"   Server Post: 'GoTo:' From: self To: 'All'.
```

```
    Server Post: 'Help:Against:Time:' From: self To: 'Pimp' With: Attacker With:10."

"   Transcript nextPutAll: (Me rIndex asString) += ' Position X: ' += Me rX asString.    "

    tvx := (m_DestX - (Me rX)).
    tvy := (m_DestY - (Me rY)).

        ((tvx abs < 1) & (tvy abs < 1)) ifTrue:
            [
            ready := true.
            Transcript nextPutAll: 'Agent reached (0,0)'; cr.
            ^1.
            ].


"fabs:"
        (((tvx > 0) ifTrue: [^tvx.];ifFalse:[^(0-tvx).]) > ((tvx > 0) ifTrue: [^tvy.];ifFalse:[^(0-
tvy).]))ifTrue:
            [
            tvy := tyv / ((tvx > 0) ifTrue: [^tvx.];ifFalse:[^(0-tvx).]).
            tvx := (tvx > 0 ifTrue: [^1.];ifFalse:[^(0-1).]).
            ^1.
            ].
            ifFalse:
            [
            tvx := tvx / (tvy > 0 ifTrue: [^tvy.];ifFalse:[^(0-tvy).]).
            tvy := (tvx > 0 ifTrue: [^1.0.];ifFalse:[^(0-1.0).]).
            ^1.
            ].

        tvx := tvx * 4.
        tvy := tvy * 4.
        tvz := Server getSpeedZFor: Me.

        (Me rZ > Server GetLandscapeLevelAtX: (Me rX) Y: (Me rY))
            ifTrue: [
                tvz := -3.
                ^1.
                ];
            ifFalse: [tvz := 3. ^1.].

        Server setSpeedFor:Me X: tvx Y: tvy Z:tvz.

    ^ 1
    !

Pimp method !
    Ok

"   Transcript nextPutAll: 'Pimp Ok called';cr."

    ^ 1
    !

Pimp method !
    Init

    "Transcript nextPutAll: 'Pimp Init called';cr."

    m_DestX := 0.0.
    m_DestY := 0.0.
    m_ovx := 0.
    m_ovy := 0.
    m_BestH := 10.0 .
    ready := false.

    Server SetAccelerationFor: Me X: 0 Y: 0 Z: ( Server GetFallingAcceleration).
    Server SetSpeedFor: Me X: 4 Y: 4 Z: 0 .
    ^ 1
    !

Pimp method !
    MessageProc: p

    "Transcript nextPutAll: 'Pimp MessageProc: called';cr."

    ^ 1
    !

Pimp method !
    CollisionDetected: reserved

    Transcript nextPutAll: 'Pimp CollisionDetected called';cr.

    ^ 1
    !
```

Файл Scr\Cactus-try.smt

```
Smalltalk interchangeVersion: '1.0' !
Class named: 'Cactus'
    instanceVariableNames: 'Me myEnergy timer myMaxEnergy myMaxTime mySpeed died ближайшаяВодаX
ближайшаяВодаY'
    classVariableNames: ''
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
    !
Cactus method !
    rData
    Transcript nextPutAll: 'Pimp rData called';cr.
    ^ Me
    !
Cactus method !
    InitData: Data

    Me := Data.

    ^1
    !
Cactus method !
ИскатьВоду

    ^true.
!
Cactus method !
    СпешитьКВоде
    self ИскатьВоду ifTrue:
        [

        ^1.
        ].


    ^1. !
Cactus method !
    Go
    | time |

    died ifFalse: [
        time := Server GetSystemTimer.

        (time - timer > myMaxTime)
            ifTrue:
                [
                died = true.
                Transcript nextPutAll: 'Cactus ' + (Me rIndex asString) + ' died'; cr.
                ^1.
                ];
            ifFalse:
                [
                myEnergy := myMaxEnergy * ((time - timer)/myMaxTime).
                ^1.].
        (myEnergy < (myMaxEnergy/2))
            ifTrue: [self СпешитьКВоде. ^1.].
        ^1.].
    ^ 1
    !
Cactus method !
    Ok
"   Transcript nextPutAll: 'Pimp Ok called';cr."
    ^ 1
    !
Cactus method !
    Init

    "Transcript nextPutAll: 'Pimp Init called';cr."
    timer := Server GetSystemTimer.
    died = false.
    myMaxEnergy := 100.
    myMaxTime := 5000.
    mySpeed := 4.
    ближайшаяВодаX := 0.
    ближайшаяВодаY := 0.
    ^ 1
    !
Cactus method !
    MessageProc: p
    "Transcript nextPutAll: 'Pimp MessageProc: called';cr."
    ^ 1
    !
Cactus method !
    CollisionDetected: reserved
    Transcript nextPutAll: 'Pimp CollisionDetected called';cr.
    ^ 1
    !!
```

Файл Scr\Test2.smt

```
Smalltalk interchangeVersion: '1.0' !

"-------------------------------------------------------"
Class named: 'TestClass'
instanceVariableNames: '"Instance variable names go here"'
classVariableNames: '"Class variable names go here"'
"sharedPools: ''"
" classInstanceVariableNames: <classInstVariableList>"

!"-------------------------------------------------------"
TestClass method!
Test
    Transcript nextPutAll: 'Test'; cr.
    Transcript nextPutAll: 'Test2'; cr.
^5.

"-------------------------------------------------------"!
```

Файл Scr\block.smt

```
Smalltalk interchangeVersion: '1.0' !

Class named: 'ExCl'
    " superclass: <superclassNameString>"
    " indexedInstanceVariables: <indexableInstVarType>"
    instanceVariableNames: 'instVar1 instVar2'
    classVariableNames: 'clVar1 clVar2'
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
    !

ExCl method !
    getX
    | t1 t2 t3 |

    "t1 := 1 + 2;
          +10. t1 = 11. "
    Transcript cr.
"   Transcript nextPut: $c.
    Transcript cr.

    (8 < 9) ifTrue:
        [ Transcript cr. Transcript nextPut: $t;cr. ^1. ].
    (8 < 9) ifFalse:
        [ Transcript cr. Transcript nextPut: $f;cr. ^2. ]."

    3 repeat:
        [
        [ : symb | Transcript nextPut:symb. ^1. ] call: $s.
        ^ 1.
        ].
    Transcript cr;cr.

    t1 := 5.
    (t1 + 1) repeat:
        [
        (t1 >= 0) ifTrue:
            [
            ^t1 repeat: [ ^Transcript nextPut: $X. ].
            ].
        Transcript cr.
        ^t1 := t1 - 1.
        ].

    t1 := 1.
    t2 := 2.
    t3 := 3.
    instVar1 := 10.
    instVar2 := 20.
    ^( t1 + t2 + t3 * instVar1 / instVar2 )
    !
```

Файл Scr\test.smt

```
Smalltalk interchangeVersion: '1.0' !
Class named: 'ExCl'
    " superclass: <superclassNameString>"
    " indexedInstanceVariables: <indexableInstVarType>"
    instanceVariableNames: 'instVar1 instVar2'
    classVariableNames: 'clVar1 clVar2'
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
```

```
    !
ExCl method !
    loadStr: str Tra: tra
    instVar1 := str.

    instVar2 := tra.
    ^ 1
    !
ExCl method !
    say: s
    Transcript cr; nextPutAll: s; cr.
    ^ 1
    !
ExCl method !
    go

    instVar2 nextPutAll: instVar1.
    ^ 0
    !
```

Файл Scr\x.smt

```
Smalltalk interchangeVersion: '1.0' !

Class named: 'ExCl'
    " superclass: <superclassNameString>"
    " indexedInstanceVariables: <indexableInstVarType>"
    instanceVariableNames: 'instVar1 instVar2'
    classVariableNames: 'clVar1 clVar2'
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
    !

ExCl method !
    getX
    | t1 t2 t3 |

    "t1 := 1 + 2;
          +10. t1 = 11. "
    Transcript cr.
    Transcript nextPut: $c.
    Transcript cr.

    Transcript nextPutAll: 'First string is out as a whole string and not by single letters';cr;
          nextPutAll: 'Second string is out as a whole string and not by single letters';cr.

    t1 := 1.
    t2 := 2.
    t3 := 3.
    instVar1 := 10.
    instVar2 := 20.
    ^( t1 + t2 + t3 * instVar1 / instVar2 )
    !
```

Файл Scr\Bump.smt

```
Smalltalk interchangeVersion: '1.0' !
Class named: 'Bump'
    instanceVariableNames: 'Me MinX MaxX MinY MaxY'
    classVariableNames: ''
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
    !
Bump method !
    rData
    Transcript nextPutAll: 'Bump rData called';cr.
    ^ Me
    !
Bump method !
    InitData: Data

    Me := Data.

    ^1
    !
Bump method !
    StayInBoxMinX: minx MinY: miny MaxX: maxx MaxY: maxy
    | vx vy vz |

    (Me rX > maxx) ifTrue:
        [
        vx := Server GetSpeedXFor: Me.
        vy := Server GetSpeedYFor: Me.
        vz := Server GetSpeedZFor: Me.
        vx := ( 0 - vx abs) - 1.
```

```
        Server SetSpeedFor: Me X: vx Y: vy Z: vz.
        ^1.
        ].
    (Me rY > maxy) ifTrue:
        [
        vx := Server GetSpeedXFor: Me.
        vy := Server GetSpeedYFor: Me.
        vz := Server GetSpeedZFor: Me.
        vy := ( 0 - vy abs) - 1.

        Server SetSpeedFor: Me X: vx Y: vy Z: vz.
        ^1.
        ].

    (Me rX < minx) ifTrue:
        [
        vx := Server GetSpeedXFor: Me.
        vy := Server GetSpeedYFor: Me.
        vz := Server GetSpeedZFor: Me.
        vx := 1 + vx abs.

        Server SetSpeedFor: Me X: vx Y: vy Z: vz.
        ^1.
        ].
    (Me rY < miny) ifTrue:
        [
        vx := Server GetSpeedXFor: Me.
        vy := Server GetSpeedYFor: Me.
        vz := Server GetSpeedZFor: Me.
        vy := 1 + vy abs.

        Server SetSpeedFor: Me X: vx Y: vy Z: vz.
        ^1.
        ].
    ^1
    !
Bump method !
    Go
    | tvz ax ay|

    self StayInBoxMinX: MinX MinY: MinY MaxX: MaxX MaxY: MaxY.
    ^ 1
    !
Bump method !
    Ok
"   Transcript nextPutAll: 'Bump Ok called';cr."
    ^ 1
    !
Bump method !
    Init

    MinX := 0-70.
    MinY := 0-70.
    MaxX := 70.
    MaxY := 70.
    ^ 1
    !
Bump method !
    MessageProc: p
    "Transcript nextPutAll: 'Bump MessageProc: called';cr."
    ^ 1
    !
Bump method !
    CollisionDetected: reserved
    Transcript nextPutAll: 'Bump CollisionDetected called';cr.
    ^ 1
    !
```

Файл Scr\Center.smt

```
Smalltalk interchangeVersion: '1.0' !
Class named: 'Center'
    instanceVariableNames: 'MinX MinY MaxX MaxY Me AX AY AZ Timer'
    classVariableNames: ''
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
    !
Center method !
    rData
    Transcript nextPutAll: 'Center rData called';cr.
    ^ Me
    !
Center method !
    InitData: Data

    Me := Data.

    ^1
    !
Center method !
```

```
    StayInBoxMinX: minx MinY: miny MaxX: maxx MaxY: maxy
    | vx vy vz |

    (Me rX > maxx) ifTrue:
        [
        vx := Server GetSpeedXFor: Me.
        vy := Server GetSpeedYFor: Me.
        vz := Server GetSpeedZFor: Me.
        vx := ( 0 - vx abs) - 1.

        Server SetSpeedFor: Me X: vx Y: vy Z: vz.
        ^1.
        ].
    (Me rY > maxy) ifTrue:
        [
        vx := Server GetSpeedXFor: Me.
        vy := Server GetSpeedYFor: Me.
        vz := Server GetSpeedZFor: Me.
        vy := ( 0 - vy abs) - 1.

        Server SetSpeedFor: Me X: vx Y: vy Z: vz.
        ^1.
        ].

    (Me rX < minx) ifTrue:
        [
        vx := Server GetSpeedXFor: Me.
        vy := Server GetSpeedYFor: Me.
        vz := Server GetSpeedZFor: Me.
        vx := 1 + vx abs.

        Server SetSpeedFor: Me X: vx Y: vy Z: vz.
        ^1.
        ].
    (Me rY < miny) ifTrue:
        [
        vx := Server GetSpeedXFor: Me.
        vy := Server GetSpeedYFor: Me.
        vz := Server GetSpeedZFor: Me.
        vy := 1 + vy abs.

        Server SetSpeedFor: Me X: vx Y: vy Z: vz.
        ^1.
        ].
    ^1
    !
Center method !
    Go

    "(Extern GetSystemTimer - Timer > 1000) ifTrue:
        [
        ( 5 < ( Extern Random: 10 )) ifTrue:
            [
            AX := 1 - ( Extern Random: 2).
            AY := 1 - ( Extern Random: 2).
            AZ := (Server GetFallingAcceleration) - ( Extern Random: 2).
            Server SetAccelerationFor: Me X: AX Y: AY Z: AZ.
            ^1.
            ].
        Timer := Extern GetSystemTimer.
        ^1.
        ].
    "
    ^ 1
    !
Center method !
    Ok
"   Transcript nextPutAll: 'Center Ok called';cr."
    ^ 1
    !
Center method !
    Init

    "Transcript nextPutAll: 'Center Init called';cr."
    MinX := 0 - 200.
    MinY := 0 - 200.
    MaxX := 200.
    MaxY := 200.
    AX := 0.
    AY := 0.
    AZ := Server GetFallingAcceleration.
    Timer := Extern GetSystemTimer.
    Server SetAccelerationFor: Me X: AX Y: AY Z: AZ.

    Server SetAgentPositionFor: Me X: 10 Y: 10 Z: 10.
    ^ 1
    !
Center method !
    MessageProc: p
    "Transcript nextPutAll: 'Center MessageProc: called';cr."
    ^ 1
    !
Center method !
    CollisionDetected: reserved
    Transcript nextPutAll: 'Center CollisionDetected called';cr.
    ^ 1
    !
```

Файл Scr\Pimp.smt

```
Smalltalk interchangeVersion: '1.0' !
Class named: 'Pimp'
    instanceVariableNames: 'Me BestH VX VY VZ Timer'
    classVariableNames: ''
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
    !
Pimp method !
    rData
    Transcript nextPutAll: 'Pimp rData called';cr.
    ^ Me
    !
Pimp method !
    InitData: Data

    Me := Data.

    ^1
    !
Pimp method !
    Go
    | tvz ax ay|
"   Server Post: 'GoTo:' From: self To: 'All'.

    Server Post: 'Help:Against:Time:' From: self To: 'Pimp' With: Attacker With:10."
"   Transcript nextPutAll: (Me rIndex asString) += ' Position X: ' += Me rX asString.    "

    VZ := Server GetSpeedZFor: Me.
    ax := Server GetAccelerationXFor: Me.
    ay := Server GetAccelerationYFor: Me.
    (Extern GetSystemTimer - Timer > 100) ifTrue:
        [
        ax := 15 - ( Extern Random: 30).
        ay := 15 - ( Extern Random: 30).
        Timer := Extern GetSystemTimer.
        ^1.
        ].

    tvz := VZ.
    ( Me rZ > ( ( Server GetLandscapeLevelAtX: Me rX Y: Me rY) + BestH))
        ifTrue: [
            tvz := 0 - 3.
            ^1.
            ];
        ifFalse: [tvz := 3. ^1.].
    VX := Server GetSpeedXFor: Me.
    VY := Server GetSpeedYFor: Me.
"   Server SetSpeedFor: Me X: VX Y: VY Z:tvz."
    Server SetAccelerationFor: Me X: ax Y: ay Z: (Server GetFallingAcceleration).
    ^ 1
    !
Pimp method !
    Ok
"   Transcript nextPutAll: 'Pimp Ok called';cr."
    ^ 1
    !
Pimp method !
    Init

    "Transcript nextPutAll: 'Pimp Init called';cr."

    VX := 4.
    VY := 4.
    VZ := 0.
    BestH := 5.0 .
    Timer := Extern GetSystemTimer.
    Server SetAccelerationFor: Me X: 0 Y: 0 Z: ( Server GetFallingAcceleration).
    Server SetSpeedFor: Me X: VX Y: VY Z: VZ .
    ^ 1
    !
Pimp method !
    MessageProc: p
    "Transcript nextPutAll: 'Pimp MessageProc: called';cr."
    ^ 1
    !
Pimp method !
    CollisionDetected: reserved
    Transcript nextPutAll: 'Pimp CollisionDetected called';cr.
    ^ 1
    !
```

Файл Scr\Pimp flyover.smt

```
Smalltalk interchangeVersion: '1.0' !
Class named: 'Pimp'
    instanceVariableNames: 'Me BestH VX VY VZ'
    classVariableNames: ''
```

```
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
    !
Pimp method !
    rData
    Transcript nextPutAll: 'Pimp rData called';cr.
    ^ Me
    !
Pimp method !
    InitData: Data

    Me := Data.

    ^1
    !
Pimp method !
    Go
    | tvz |
"   Server Post: 'GoTo:' From: self To: 'All'.

    Server Post: 'Help:Against:Time:' From: self To: 'Pimp' With: Attacker With:10."
"   Transcript nextPutAll: (Me rIndex asString) += ' Position X: ' += Me rX asString.    "

    VX := Server GetSpeedXFor: Me.
    VY := Server GetSpeedYFor: Me.
    VZ := Server GetSpeedZFor: Me.
    tvz := VZ.

    ( Me rZ > ( ( Server GetLandscapeLevelAtX: Me rX Y: Me rY) + BestH))
        ifTrue: [
            tvz := 0 - 3.
            ^1.
            ];
        ifFalse: [tvz := 3. ^1.].
    ( ( tvz <= VZ) & ( tvz >= VZ)) ifFalse:
        [
        Server SetSpeedFor:Me X: VX Y: VY Z:tvz.
        ^1.
        ].
    ^ 1
    !
Pimp method !
    Ok
"   Transcript nextPutAll: 'Pimp Ok called';cr."
    ^ 1
    !
Pimp method !
    Init

    "Transcript nextPutAll: 'Pimp Init called';cr."

    VX := 4.
    VY := 4.
    VZ := 0.
    BestH := 5.0 .
    Server SetAccelerationFor: Me X: 0 Y: 0 Z: ( Server GetFallingAcceleration).
    Server SetSpeedFor: Me X: VX Y: VY Z: VZ .
    ^ 1
    !
Pimp method !
    MessageProc: p
    "Transcript nextPutAll: 'Pimp MessageProc: called';cr."
    ^ 1
    !
Pimp method !
    CollisionDetected: reserved
    Transcript nextPutAll: 'Pimp CollisionDetected called';cr.
    ^ 1
    !
```

Файл Scr\Ufo.smt

```
Smalltalk interchangeVersion: '1.0' !
Class named: 'Ufo'
    instanceVariableNames: 'Me myCenter m_DestX m_DestY m_BestH myMovementState someTimer someTimerEmplace
myCurrentInterest myNextState myPreviousState'
    classVariableNames: ''
    " sharedPools: <poolList>"
    " classInstanceVariableNames: <classInstVariableList>"
!"-------------------------------------------------------------"
Ufo method !
    rData
    Transcript nextPutAll: 'Ufo rData called';cr.
    ^ Me    !
"-------------------------------------------------------------"
Ufo method !
    InitData: Data
        Me := Data.
    "   Transcript nextPutAll: 'Ufo init data called.'; cr."
        ^1
!"-------------------------------------------------------------"
Ufo method !
```

```
InterestingFoundAtX: x Y: y
myNextState := 2.
self GotoX: x Y: y.
^1.
!
"-------------------------------------------------------"
Ufo method !
LookForInteresting
    ( myCenter = false) ifFalse:
        [
        self InterestingFoundAtX: myCenter rX Y: myCenter rY.
        ^1.
        ].
    ^1.
!"-------------------------------------------------------------"
Ufo method !
PerformGotoMovement
    | tvx tvy tvz |
    tvx := (m_DestX - (Me rX)).
    tvy := (m_DestY - (Me rY)).
"   Transcript nextPutAll: ' goto '; cr."
        ((tvx abs < 1) & (tvy abs < 1)) ifTrue:
            [
            myMovementState := myNextState.
            someTimer := Extern GetSystemTimer.
"           Transcript nextPutAll: 'Agent reached (' + m_DestX asString + ', ' + m_DestY asString + ').';
cr."
"           Transcript nextPutAll: 'Agent reached some point';cr."
            ^1.
            ].
        (tvx abs > (tvy abs))
        ifTrue:
            [
            tvy := tyv / (tvx abs).
            (tvx > 0)
                ifTrue: [tvx := 1. ^1.];
                ifFalse:[tvx := (0-1). ^1.].
            ^1.
            ];
        ifFalse:
            [
            tvx := tvx / (tvy abs).
            (tvy > 0)
                ifTrue: [tvy := 1. ^1.];
                ifFalse:[tvy := (0-1). ^1.].
            ^1.
            ].
        tvx := tvx * 4.
        tvy := tvy * 4.
        tvz := self CheckHeight.
        Server SetSpeedFor: Me X: tvx Y: tvy Z:tvz.
^1.
!"-------------------------------------------------------------"
Ufo method!
CheckHeight "Returns corrected Z-speed"
    | temp |
    temp := 1.
    temp := Server GetSpeedZFor: Me.
    (Me rZ > (Server GetLandscapeLevelAtX: (Me rX) Y: (Me rY)))
        ifTrue: [
            temp := 0-3.
            ^1.
            ];
        ifFalse: [temp := 3. ^1.].
^temp.
!"-------------------------------------------------------------"
Ufo method !
    Go
    self LookForInteresting.
    (myMovementState = 0)
        ifTrue: [self PerformBrowsingMovement. ^1.].
    (myMovementState = 1)
        ifTrue: [self PerformGotoMovement. ^1.].
    (myMovementState = 2)
        ifTrue: [self PerformExploringMovement. ^1.].
    ^ 1
!"-------------------------------------------------------"
Ufo method !
PerformBrowsingMovement
| tvx tvy tvz |
    tvz := self CheckHeight.
    tvx := Server GetSpeedXFor: Me.
    tvy := Server GetSpeedYFor: Me.
    (Extern GetSystemTimer - someTimer > someTimerEmplace)
        ifTrue:
        [
        tvx := Extern Random: 5.
        tvy := Extern Random: 5.
        someTimer := Extern GetSystemTimer.
        ^1.
        ].
    (Me rX > 100 & (tvx > 0)) ifTrue:
        [
        tvx := 0 - tvx.
        ^1.
        ].
    (Me rY > 100 & (tvx > 0)) ifTrue:
```

```
        [
        tvy := 0 - tvy.
        ^1.
        ].
    (Me rX < (0 - 100) & (tvx < 0)) ifTrue:
        [
        tvx := 0 - tvx.
        ^1.
        ].
    (Me rY < (0 - 100) & (tvx < 0)) ifTrue:
        [
        tvy := 0 - tvy.
        ^1.
        ].
    Server SetSpeedFor: Me X: tvx Y: tvy Z: tvz.
        ^1.
"-------------------------------------------------------"!
Ufo method!
PerformExploringMovement
    ((Me rX - m_DestX) abs < 3)
         ifTrue:
            [
            Server SetSpeedFor: Me X: 0 Y: 0 Z: ((self CheckHeight) + (Extern Random: 1) - 0.5).
            ^1.
            ];
        ifFalse:
            [
            someTimerEmplace := 5000.
            myNextState := 2.
            myMovementState := 1.
            ^1.
            ].
    (Server GetSystemTime - someTimer > someTimerEmplace)
        ifTrue:
            [
            myMovementState := 0.
            someTimer := Extern GetSystemTimer.
            ].
    ^1.
"-------------------------------------------------------"!
Ufo method !
GotoX: x Y: y
    m_DestX := x.
    m_DestY := y.
    myMovementState := 1.^1.
!"-------------------------------------------------------------"
Ufo method !
    Ok
    ^ 1
!"-------------------------------------------------------------"
Ufo method !
    Init
    m_DestX := 0.
    m_DestY := 0.
    m_BestH := 10.
    myMovementState := 1.
    someTimer := 0.
    someTimerEmplace := 0.
    myCurrentInterest := 0.
    myNextState := 0.
    myPreviousState := 0.
    myCenter := Server GetAgentOf: 'Center' Number: 1.
    Transcript nextPutAll: 'Ufo Init: Variables initialized'; cr.
    Server SetAccelerationFor: Me X: 0 Y: 0 Z: ( Server GetFallingAcceleration).
"   Server SetSpeedFor: Me X: 4 Y: 4 Z: 0 ."
    Transcript nextPutAll: 'Ufo Init: ok.'; cr.
    ^ 1
!"-------------------------------------------------------------"
Ufo method !
    MessageProc: messageContainer
    Transcript nextPutAll: 'Ufo MessageProc: called';cr.
    self gotoX: messageContainer rX Y: messageContainer rY.
    ^ 1
!"-------------------------------------------------------------"
Ufo method !
    CollisionDetected: reserved
    Transcript nextPutAll: 'Ufo CollisionDetected called';cr.
    ^ 1
!"-------------------------------------------------------------"!!
```

Приложение П2-в.
Исходные тексты программ для консоли

Файл Scr\editor

```
@echo off
load "side.dll"
init
editor
```

Файл Scr\load3d

```
@echo off
load STM32.dll
load renderer.dll
load physics.dll
load Map.dll
load AgSrv32.dll
;load AiCore.dll
load con3d.dll
init
start3d
load lagents.dll

init
echo Graphics modules loaded.
```

Файл Scr\loadall

```
load linker
load cppvm
load smallvm
load smallcompiler
load smalldebugger
load systemextentions1
load side
init
editor

loadServAnd3d

RegisterAgentCreator First
RegisterAgentCreator Pimp
RegisterAgentCreator Ufo
```

Файл Scr\loadide

```
@echo off

load side
init
editor
echo IDE loaded.
```

Файл Scr\smalltalk

```
load "linker.dll"
load "cppvm.dll"
load "smallvm.dll"
load "smallcompiler.dll"
load "smalldebugger.dll"
load "systemextentions1.dll"
load "side.dll"
init
load "lagents.dll"
init
```

```
editor
```

Файл Scr\startup

```
@echo off
load "STM32.dll"
load "renderer.dll"
load "physics.dll"
load "Map.dll"
load "AgSrv32.dll"
load "AiCore.dll"
load "con3d.dll"
init
load "linker.dll"
load "cppvm.dll"
load "smallvm.dll"
load "smallcompiler.dll"
load "smalldebugger.dll"
load "systemextentions1.dll"
load "side.dll"
init
load "lagents.dll"
init

editor
rem start3d
rem pause3d
```

Файл Scr\LSA3D

```
@echo off
load STM32.dll
load renderer.dll
load physics.dll
load Map.dll
load AgSrv32.dll
load con3d.dll
init
start3d
load lagents.dll
init
RegisterAgentCreator Bump
RegisterAgentCreator Pimp
RegisterAgentCreator Center
RegisterAgentCreator Ufo
echo Graphics modules loaded.
```

Файл Scr\LSmt

```
@echo off
load linker
load cppvm
load smallvm
load smallcompiler
load smalldebugger
load systemextentions1
load side
echo Smalltalk modules loaded.
```

Файл Scr\LoadSmalltalk

```
@echo off
load linker
load cppvm
load smallvm
load smallcompiler
load smalldebugger
load systemextentions1
echo Smalltalk modules loaded.
```

Файл Scr\loadServAnd3d

```
@echo off
```

```
load STM32.dll
load renderer.dll
load physics.dll
load Map.dll
load AgSrv32.dll
;load AiCore.dll
load con3d.dll
init
start3d
load lagents.dll
init
```

Файл Common\Utilities\Module.txt

```
//Defines for Easy creating Dll-Module
//(including implementation of techincal functions)

// Module.h - header file
// created 10.09.2002 by header creator
#pragma once
#ifndef __MODULE_HEADER_FILE_BY_VASFED_INCLUDED__
#define __MODULE_HEADER_FILE_BY_VASFED_INCLUDED__

#include "..\..\Loader\Loader.h" //Loader interface

//DllGetInfo
//DllGetInstance
//DllInit

#define EXPORT __declspec (dllexport)
#define SELECTANY __declspec (selectany)

#define DECLARE_MODULE_INFO () DllInfo DLLInfo = {"Unknown", "Unknown", "Unknown", false, false, false,
false, false};
#define DECLARE_MODULE(classname) \
    \
    extern "C" void EXPORT DllGetInstance (void**ppr)\
    {if (!ppr) return; Object->AddRef(); *ppr = Object;}\
    extern "C" void EXPORT DllGetInfo (void** ppr){if (!ppr)return; *ppr = &DLLInfo;}

extern "C" EXPORT DllInit (ILoader* loader);

#endif //__MODULE_HEADER_FILE_BY_VASFED_INCLUDED__

ULONG STDMETHODCALLTYPE AddRef ();
ULONG STDMETHODCALLTYPE Release ();
HRESULT STDMETHODCALLTYPE QueryInterface (const IID &iid, void ** ptr);
volatile long m_refs;

//----------------
ULONG STDMETHODCALLTYPE CLoader::AddRef ()
    {
    return InterlockedIncrement (&m_refs);
    }
//-----------------------------------------------------------------------------

ULONG STDMETHODCALLTYPE CLoader::Release ()
    {
    if (InterlockedDecrement (&m_refs) == 0)
        {
        delete this;
        return 0;
        }
    return m_refs;
    }
//-----------------------------------------------------------------------------
HRESULT STDMETHODCALLTYPE CLoader::QueryInterface (const IID &iid, void ** ptr)
    {
    if (iid == IID_IUnknown)
        {
        *ptr = static_cast <IUnknown*> (this);
        }
    else
        {
        if (iid == IID_ILoader)
            {
            *ptr = static_cast <ILoader*> (this);
            }
        else
            {
            *ptr = NULL;
            return E_NOINTERFACE;
            }
        }
    reinterpret_cast <IUnknown *> (* ptr) -> AddRef ();
    return S_OK;
    }
//-----------------------------------------------------------------------------
```

Файл ConsoleRecursiveDescent\Zabor.txt

```
От: Илья Дединский [ded@concord.ru]
Кому: Василий Федосеев
Тема: RE: Рекурсивный спуск

Важность: Высокая

Привет.

Регулярные выражения знаешь?

Правила грамматики, например...

G0 ::= T                    ; G0 - начальный символ грамматики
T  ::= N ( [+-] N ) *       ; term
N  ::= [0123456789] +       ; number

пробелы для наглядности, вообще их для корректности надо убрать.

Метасимволы:
() - grouping
[] - set of chars
*  - when not in set, is repetition operator (repeat 0 or more times)
+  - when not in set, is repetition operator (repeat 1 or more times)
|  - OR operator
"c"- literal symbol when not in set

...описывающие выражения (для данных правил):

4
1652
25 + 1
16 - 6436
92+ 643 + 535
2 + 4 - 6
и т.д.

но не

54 +        only part of group exist
7 + -   must exist only one char of the set
62 *        * (in repeating sense) cannot be used in the target language
          or if * is multipication then this rule doesn't describe this
4 5     missing symbol denoted in the set
+ 46        wrong beginning
- 818   the same
{ 54 }  parentheses are not described in the rule above
( 64 )  see comment about "*"

...переводится в:

////////////////////////////////

const char* S = NULL;

int GetT();
int GetN();

////////////////////////////////
// Rule 1

int GetG0 (const char* s)
{
printf ("Postfix form: ");

S = s;
return GetT();
}

////////////////////////////////
// Rule 2

int GetT()
{
int val = GetN();
while (*S == '+' || *S == '-')
    {
    int op = *S++;
    int val2 = GetN();

    printf ("%c ", op);

    if (op == '+') val += val2;
    else           val -= val2;
    }

return val;
}

////////////////////////////////
// Rule 3
```

```
int GetN()
{
int val = 0;
while (isdigit (*S)) val = val*10 + (*S++ - '0');

printf ("%d ", val);
return val;
}

////////////////////////////////
// Test

void main()
{
const char ex[] = "43+5-7948-72";

printf ("\nGetN (%s) returned %d\n\n", ex, GetG0 (ex));
}
```

Если посмотреть внимательней, то видно, что это практически механический перевод. Можно даже написать транслятор. Но сейчас не пиши! Отнимет время. Задача простая и красивая, напишешь потом, в период релаксации после защиты.

Классическая грамматика для инфиксных выражений:

```
G0 ::= E
E  ::= T ( [+-] T ) *
T  ::= P ( [*/] P ) *
P  ::= N | "(" E ")" | F "(" E ")"
N  ::= [0123456789] +
F  ::= "sin" | "cos" | "tan" | ...
```

Здесь правило для обработки [+-] стоит ближе к G0 и, соответственно, выполняется позже, чем для [*/]. Так поддерживается приоритет. Постфиксная форма наиболее удобна для системы команд виртуальной машины: В ней

```
<число> => <push число>,
<op>    => <pop a>, <pop b>, <c = a op b>, <push c>.
```

Для эффективности лучше предварительно разобрать строку на лексемы (лексический разбор), т.е. с помощью большого свитча (это нормально, для данной задачи это - классика) превратить в

```
struct Token
    {
    enum {tkNone tkInt, tkDouble, tkOp, ...} type;
    union
        {
        int i;
        double d;
        char op;
        int keyword; // translated to number. Or merged with "op"
        int var;     // indexed in nametable
        int func;    // the same, or merged with "var" (differences see
                     // in the nametable)
        ...
        }
    }
    TokenArray [MAX];
```

И вообще см. Страуструпа, введение в C++, он там калькулятор в качестве примера пишет.

STL задуман хорошо, а реализован действительно не очень.

>Потом в консольке будет команда перехода на него.
На кого?

ded

```
Press RESET to continue...
```

-----Original Message-----
From: Василий Федосеев [mailto:vasfed@pochtamt.ru]
To: 'Илья Дединский'
Subject: Рекурсивный спуск
Importance: High

Привет Дед.
Калькулятор с рекурсивным спуском я написал (скобочки кушает). Только как приоритеты операций без гемора сделать? (нормального способа я сходу не вижу, не сортировать же их!)

В остальном проблем нету. Этот скриптовый язычок (пока без приоритетов операций) я дня за три легко сделаю.

В STL Стек - такое <даже слов нету>! В нем pop возвращает void (посмотреть что лежит - top(), а выпихнуть в переменную нельзя, только просто выкинуть)!

Потом в консольке будет команда перехода на него.
Я, наверное, там даже функции сделаю (возможность написать их на нем, зашитые команды - не в счет, там не нужно сильно напрягаться - вызов не настоящий).

Радостный Vasfed.

PS. Ты меня RecursiveDescent'ом не грузил. Без загруза писать коряво. Я, конечно, приблизительно представляю, что это такое, но специфических фишек не знаю.

Get free mailbox 20 Mb at http://www.hotbox.ru

Файл ConsoleUI2\ExeptionDump.txt

```
Exeption Engine version 1.4
Beginning dynamical synchronous dump of exeptions and messages
All the events are timed,
System clock is 11740181     (winmm timeGetTime() function used, see MSDN)

End of dump
Statistic:
there'as 0 total events:
    0   exeptions were thrown
    0   messages  were send
    0   exeptions were catched
    0   exeptions were handled
    0   exeptions were NOT handled

Note: If programmer was not using all the macroces, dump may be wrong
```

Файл meshes\ufo\3dsdump.txt

```
Exeption Engine version 1.5 by Vasfed
Beginning dynamical synchronous dump of exeptions and messages
Computer's clock is 37308196     (winmm timeGetTime())
System startup time Tue Dec 17 19:01:35 2002

------------------------------------------------------------
DUMP:

LOADER: Loader version 1.5 by Vasfed. Created successfully.
LOADER: Loaded module: 'STM32.dll': StorageMaster by Xroft
File ufo.3ds is found
New chunk is found, chunk id is 19789, chunk length is 6722
PRIMARY chunk is found
ProcessNextChunk started
New chunk is found, chunk id is 2, chunk length is 10
VERSION chunk is found
New chunk is found, chunk id is 15677, chunk length is 6482
OBJECTINFO chunk is found
New chunk is found, chunk id is 15678, chunk length is 10
ProcessNextChunk started
New chunk is found, chunk id is 45055, chunk length is 225
MATERIAL chunk is found
ProcessNextMaterialChunk started
New chunk is found, chunk id is 40960, chunk length is 18
MATNAME chunk is found
New chunk is found, chunk id is 40976, chunk length is 15
Unknown chunk is found
New chunk is found, chunk id is 40992, chunk length is 15
New chunk is found, chunk id is 41008, chunk length is 15
Unknown chunk is found
New chunk is found, chunk id is 41024, chunk length is 14
Unknown chunk is found
New chunk is found, chunk id is 41025, chunk length is 14
Unknown chunk is found
New chunk is found, chunk id is 41040, chunk length is 14
Unknown chunk is found
New chunk is found, chunk id is 41042, chunk length is 14
Unknown chunk is found
New chunk is found, chunk id is 41043, chunk length is 14
Unknown chunk is found
New chunk is found, chunk id is 41216, chunk length is 8
Unknown chunk is found
New chunk is found, chunk id is 41092, chunk length is 14
Unknown chunk is found
New chunk is found, chunk id is 41098, chunk length is 6
Unknown chunk is found
New chunk is found, chunk id is 41095, chunk length is 10
Unknown chunk is found
New chunk is found, chunk id is 41472, chunk length is 48
ProcessNextMaterialChunk started
New chunk is found, chunk id is 48, chunk length is 8
Unknown chunk is found
New chunk is found, chunk id is 41728, chunk length is 16
New chunk is found, chunk id is 41809, chunk length is 8
Unknown chunk is found
New chunk is found, chunk id is 41811, chunk length is 10
Unknown chunk is found
ProcessNextMaterialChunk exited
ProcessNextMaterialChunk exited
New chunk is found, chunk id is 256, chunk length is 10
Unknown chunk is found
0 bytes read
New chunk is found, chunk id is 16384, chunk length is 6231
OBJECT chunk is found
ProcessNextObjectChunk started
New chunk is found, chunk id is 16640, chunk length is 6214
```

```
OBJECT_MESH chunk is found
ProcessNextObjectChunk started
New chunk is found, chunk id is 16656, chunk length is 1856
OBJECT_VERTICES chunk is found
Vertices read: 154
Vertex 0 is 0, 0, 0
Vertex 1 is 0, 0, 0
Vertex 2 is 0, 0, 0
Vertex 3 is 0, 0, 0
Vertex 4 is 0, 0, 0
Vertex 5 is 0, 0, 0
Vertex 6 is 0, 0, 0
Vertex 7 is 0, 0, 0
Vertex 8 is 0, 0, 0
Vertex 9 is 0, 0, 0
Vertex 10 is 0, 0, 0
...
Vertex 142 is 0, 0, 0
Vertex 143 is 0, 0, 0
Vertex 144 is 0, 0, 0
Vertex 145 is 0, 0, 0
Vertex 146 is 0, 0, 0
Vertex 147 is 0, 0, 0
Vertex 148 is 0, 0, 0
Vertex 149 is 0, 0, 0
Vertex 150 is 0, 0, 0
Vertex 151 is 0, 0, 0
Vertex 152 is 0, 0, 0
Vertex 153 is 0, 0, 0
New chunk is found, chunk id is 16704, chunk length is 1240
OBJECT_UV chunk is found
texture coord read: 154
TexCoords 0 are 0, 0
TexCoords 1 are 0, 0
TexCoords 2 are 0, 0
TexCoords 3 are 0, 0
TexCoords 4 are 0, 0
TexCoords 5 are 0, 0
TexCoords 6 are 0, 0
...
TexCoords 145 are 0, 0
TexCoords 146 are 0, 0
TexCoords 147 are 0, 0
TexCoords 148 are 0, 0
TexCoords 149 are 0, 0
TexCoords 150 are 0, 0
TexCoords 151 are 0, 0
TexCoords 152 are 0, 0
TexCoords 153 are 0, 0
New chunk is found, chunk id is 16736, chunk length is 54
Unknown chunk is found
48 bytes read
New chunk is found, chunk id is 16672, chunk length is 3058
OBJECT_FACES chunk is found
Indices read: 216
Indices 0 are 0, 2, 1
Indices 1 are 0, 3, 2
Indices 2 are 0, 4, 3
Indices 3 are 0, 5, 4
...
Indices 208 are 146, 101, 102
Indices 209 are 147, 102, 103
Indices 210 are 148, 103, 104
Indices 211 are 149, 104, 105
Indices 212 are 150, 105, 106
Indices 213 are 151, 106, 107
Indices 214 are 152, 107, 108
Indices 215 are 153, 108, 91
New chunk is found, chunk id is 16688, chunk length is 452
Unknown chunk is found
446 bytes read
New chunk is found, chunk id is 16720, chunk length is 870
Unknown chunk is found
864 bytes read
ProcessNextObjectChunk exited
ProcessNextObjectChunk exited
Memory block with size 152 allocated

Memory block with size 36288 allocated

ProcessNextChunk exited
New chunk is found, chunk id is 45056, chunk length is 224
EDITKEYFRAME chunk is found
New chunk is found, chunk id is 45066, chunk length is 21
Unknown chunk is found
0 bytes read
New chunk is found, chunk id is 45064, chunk length is 14
Unknown chunk is found
0 bytes read
New chunk is found, chunk id is 45065, chunk length is 10
Unknown chunk is found
0 bytes read
New chunk is found, chunk id is 45058, chunk length is 173
Unknown chunk is found
0 bytes read
ProcessNextChunk exited
LOADER: Was not terminated by main program.
```

```
    Possible crash. Now trying to terminate.
LOADER: Unloading modules
Memory block with size 36288 deleted

Memory block with size 152 deleted

LOADER: Terminated correctly

End of dump
------------------------------------------------------------
Computer's clock is 37329256
System shutdown time is Tue Dec 17 19:01:56 2002
System was up for about 0 days 0 hours 0 minutes 21 seconds (if more, there's error)

Statistics:
there'as 627 total events:
    0   exeptions were thrown
    627 messages  were send
    0   exeptions were catched
    0   exeptions were handled
    0   exeptions were NOT handled

Note: If programmer was not using all the macroces, dump may be wrong
```

Файл meshes\ufo\xxxdump.txt

```
Exeption Engine version 1.5 by Vasfed
Beginning dynamical synchronous dump of exeptions and messages
Computer's clock is 37329426     (winmm timeGetTime())
System startup time Tue Dec 17 19:01:56 2002

------------------------------------------------------------
DUMP:

End of dump
------------------------------------------------------------
Computer's clock is 37329426
System shutdown time is Tue Dec 17 19:01:56 2002
System was up for about 0 days 0 hours 0 minutes 0 seconds (if more, there's error)

Statistics:
there'as 0 total events:
    0   exeptions were thrown
    0   messages  were send
    0   exeptions were catched
    0   exeptions were handled
    0   exeptions were NOT handled

Note: If programmer was not using all the macroces, dump may be wrong
```

Файл meshes\ball\3dsdump.txt

```
Exeption Engine version 1.4
Beginning dynamical synchronous dump of exeptions and messages
All the events are timed,
System clock is 10577820     (winmm timeGetTime() function used, see MSDN)

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line 48)
Trace message: File ball.3ds is found

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line 24)
Trace message: New chunk is found, chunk id is 19789, chunk length is 2200

Trace message: Indices 43 are 42, 21, 20

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line
277) Trace message: Indices 44 are 43, 22, 21

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line
277) Trace message: Indices 45 are 44, 23, 22

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line
277) Trace message: Indices 46 are 45, 24, 23

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line 24)
Trace message: New chunk is found, chunk id is 16720, chunk length is 198

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line
237) Trace message: Unknown chunk is found

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line
240) Trace message: 192 bytes read
```

```
Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line
248) Trace message: ProcessNextObjectChunk exited

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line
140) Trace message: Unknown chunk is found

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line
142) Trace message: 0 bytes read

Message: (module Unknown by UnknownLamer, file 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp' line
149) Trace message: ProcessNextChunk exited

End of dump
Statistic:
there'as 230 total events:
    0   exeptions were thrown
    230 messages  were send
    0   exeptions were catched
    0   exeptions were handled
    0   exeptions were NOT handled

Note: If programmer was not using all the macroces, dump may be wrong
```

Файл Scr\3dsdump.txt

```
Exeption Engine version 1.4
Beginning dynamical synchronous dump of exeptions and messages
All the events are timed,
System clock is 10600352     (winmm timeGetTime() function used, see MSDN)

Exeption thrown: (Unknown module by UnknownLamer, file:
'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp', line 359) Assertion Exeption: assertion failed: 'argc
== 3', (Not equal number of parameters in command line), generated at time 60

The Exeption '(Unknown module by UnknownLamer, file: 'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp',
line 359) Assertion Exeption: assertion failed: 'argc == 3', (Not equal number of parameters in command
line), generated at time 60
    Was Catched by UnknownLamer in module 'Unknown', file
'd:\!elendor\link\elendor\3dsloader\3dsloader.cpp', line 380, comments: 'NoComments'
```

Файл Scr\Agents.txt

```
@echo off
echo Creating Agents...
load "AICore"
init

rem Agent "AI name" "Mesh FIle"x y z misc
AGENT "Ufo" "Ufo.xxx" r r 10 0
AGENT "Ufo" "Ufo.xxx" r r 10 0
AGENT "Ufo" "Ufo.xxx" r r 10 0
AGENT "Ufo" "Ufo.xxx" r r 10 0
AGENT "Ship" "Ship.xxx" r r 1 0
```

Файл Scr\Modules.txt

```
LOAD MODULES:
conex32
stm32
map
agsrv32
aicore
physics
renderer
sound
```

Файл Scr\ScreenSettings.txt

```
SCREEN SETTINGS
Fullscreen = 0
WindowPosition = (100; 100)
WindowSize = 320x240
```

Файл Scr\Test.txt

```
@echo off
load linker
load cppvm
load smallvm
load smallcompiler
load smalldebugger
load SystemExtentions1
load side
init
echo Components loaded
editor
com pimp.smt
pause
NewVar temp
smt ^temp := Pimp new.
smt ^temp SetAmplitude: 10.
smt ^temp Go.
echo End.
```

Файл Scr\continue.txt

```
com pimp.smt
NewVar temp
smt temp := Pimp new
smt temp SetAmplitude: 10
smt temp Go
```

Файл Scr\testsmt.txt

```
load linker
load cppvm
load smallvm
load smallcompiler
init
com x.smt
```

Файл Scr\vw.txt

```
rem VIRTUALWORLD DEFINITION

STORAGEMASTER_SIZE 12
NUMAGENTS 5
AGENTSMAP map.map
SCREEN 320 240 0
FOG 0.0035 0.0 50.0
CAMERA 0.0 -5.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 -0.133 0.133 -0.1 0.1 0.1 500.0 0
CAMERA 5.0 5.0 20.0 0.0 0.0 0.0 0.0 0.0 1.0 -0.133 0.133 -0.1 0.1 0.1 500.0 1
FOGCOLOR 50 200 230 255
GRAVITY 0 0 -9.8
CD VirtualWorld
TEXTURES textures.rrc
LANDSCAPE landscap.rrc
LIGHTS lights.xxx
AGENTS ..\agents.txt
CD ..\
MAKEWINDOW
COMPILE
```

Файл Scr\all.txt

```
LSmt
init
smt 2 + 2
LSA3d
editor

rem Не работает ещё пауза физики по pause3d.
rem В pause3d on и off надо поменять местами!
rem rem LIDE не грузиться пока не загрузится графика!
```

Приложение П3. Диаграммы основных классов

П3.1. Диаграммы классов физического модуля

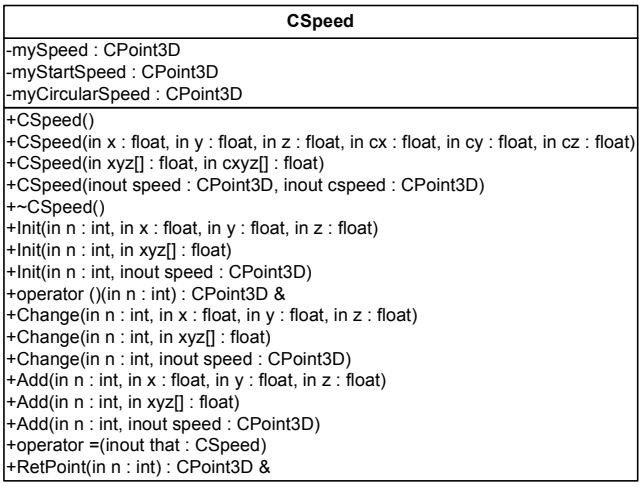


Диаграмма 3.1. Класс CSpeed.

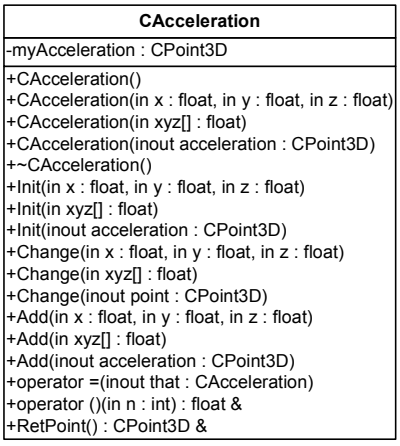


Диаграмма 3.2. Класс CAcceleration.

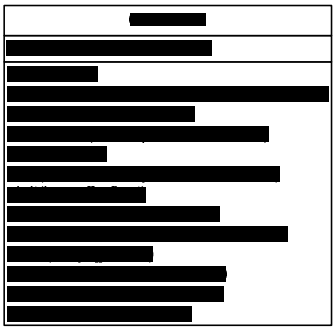


Диаграмма 3.3. Класс CPosition.

Диаграмма 3.4. Класс CPhysMaterial.

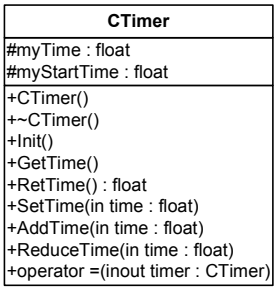


Диаграмма 3.5. Класс CTimer.

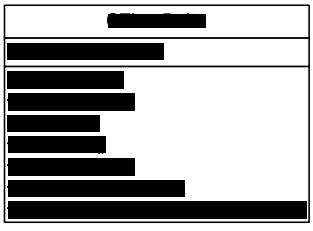


Диаграмма 3.6. Класс CTimeDelta.

November 14, 2008 12:17 AM -- 555

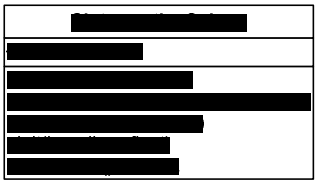


Диаграмма 3.7. Класс CIntersectionSphere.

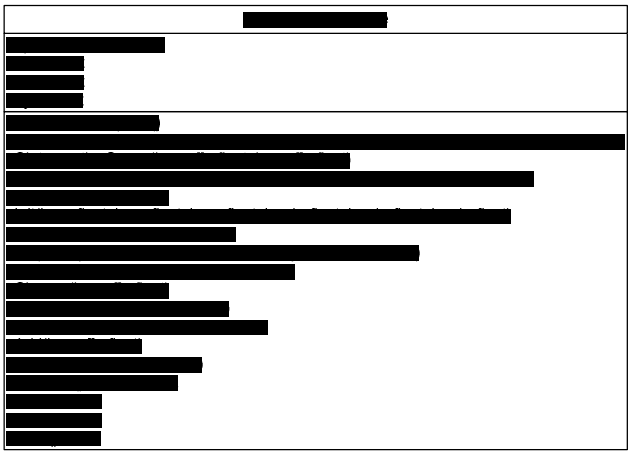


Диаграмма 3.8. Класс CIntersectionSpace.

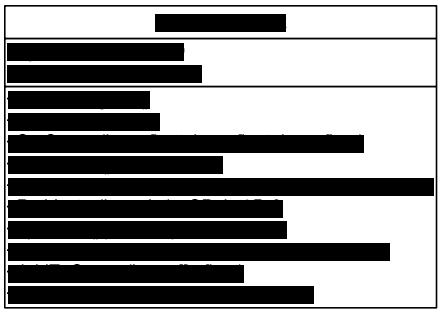


Диаграмма 3.9. Класс CBoundingBox.

П3.2. Диаграммы классов графического модуля

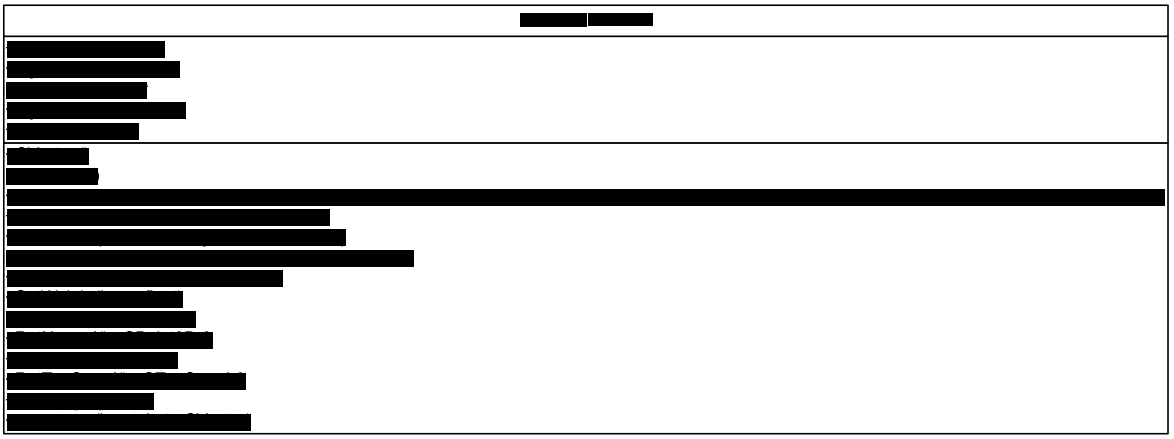


Диаграмма 4.1. Класс CVertex.

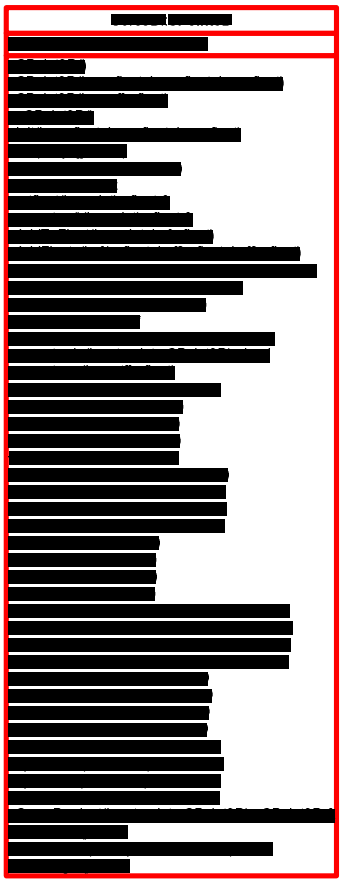


Диаграмма 4.2. Класс CPoint3D.

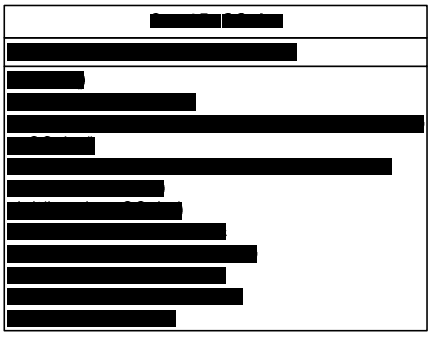


Диаграмма 4.3. Класс CColor.

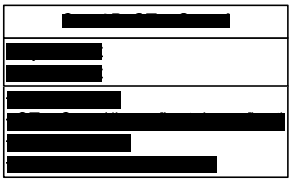


Диаграмма 4.4. Класс CTexCoord.

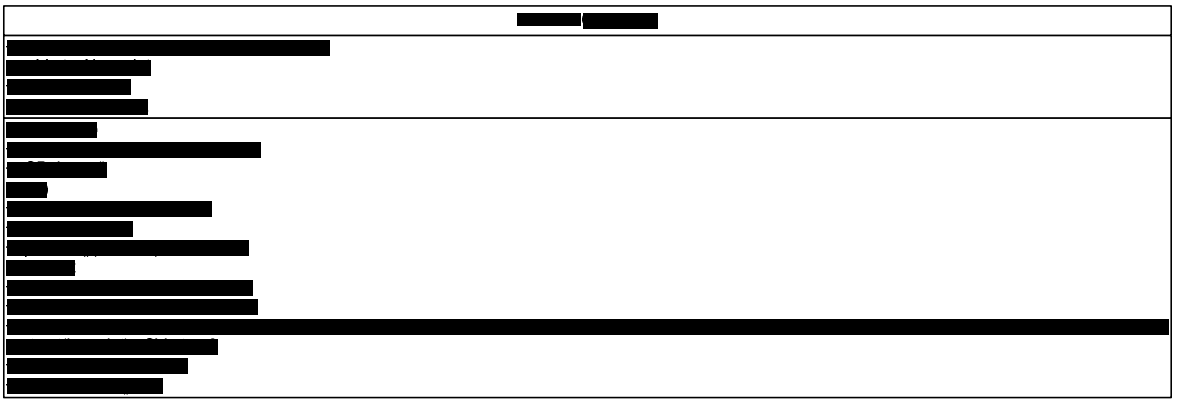


Диаграмма 4.5. Класс CPolygon.

Core3D::**CMesh**
-myLights : CLightSource ** -myPolyArray : CPolygon ** -myMaxPolys : int -myCurPoly : int -myMaxLights : int -myCurLight : int -myMaterial : CGraphMaterial -myBone : CBone -myParent : CMesh * -myChildren : CMesh ** -myMaxChildren : int -myCurChild : int -myID : int
+CMesh() +CMesh(in maxobj : int, in maxlight : int) +~CMesh() +Init(in maxobj : int, in maxlight : int) +New() +Delete() +RetMaxPolys() : int +RetMaxLights() : int +RetCurPolys() : int +RetCurLights() : int +RetID() : int +SetID(in id : int) +polyOk() : int +polyOk(in n : int) : int +lightOk() : int +lightOk(in n : int) : int +SetMaterial(in d : float, in s : float, in e : float) +SetMaterial(inout material : CGraphMaterial) +RetMaterial() : CGraphMaterial & +SetCenter(in x : float, in y : float, in z : float) +Move(in x : float, in y : float, in z : float) +RetCenter() : CPoint3D & +RetTransformMatrix() : CMatrix4x4 & +RetBone() : CBone & +Rotate(inout m : CMatrix4x4) +Rotate(in x : float, in y : float, in z : float) +SetAngle(in x : float, in y : float, in z : float) +SetMatrix(inout m : CMatrix4x4) +Clone(in that : CMesh*) +Clone(in that : CMesh*, in polys : CPolygon*) +operator =(inout that : CMesh) +AddPoly(in poly : CPolygon*) +AddPolys(in polys : CPolygon*, in n : int) +RetPoly(in n : int) : CPolygon * +AddLight(in light : CLightSource*) +AddLights(in lights : CLightSource*, in nlights : int) +RetLight(in n : int) : CLightSource * +SetParent(in parent : CMesh*) +InitChildren(in num : int) +SetChild(in child : CMesh*) +RetCurChildren() : int +RetMaxChildren() : int +RetParent() : CMesh * +RetChild(in num : int) : CMesh *

Диаграмма 4.6. Класс CMesh.

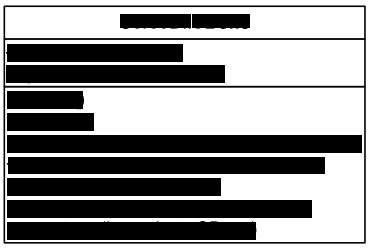


Диаграмма 4.7. Класс CBone.

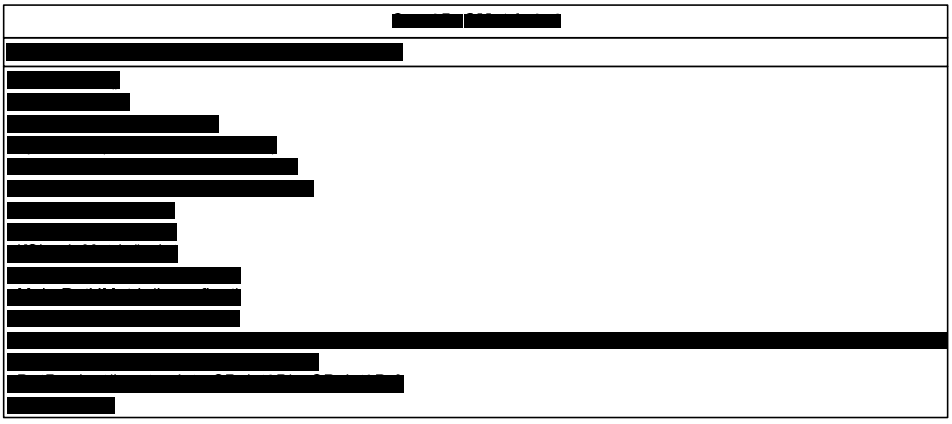


Диаграмма 4.8. Класс CMatrix4x4.

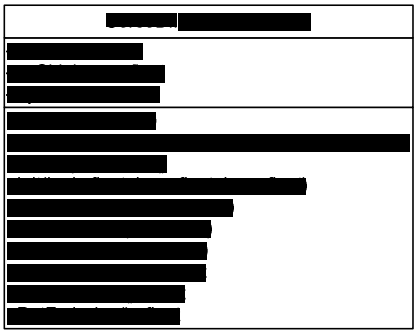


Диаграмма 4.9. Класс CGraphMaterial.

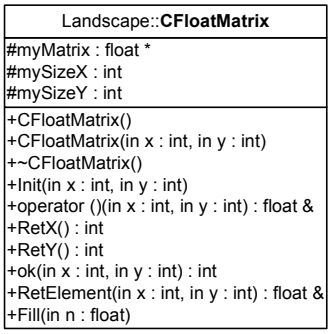


Диаграмма 4.10. Класс CFloatMatrix.

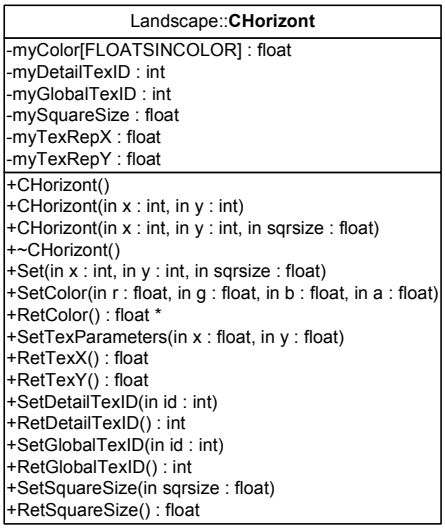


Диаграмма 4.11. Класс CHorizont.

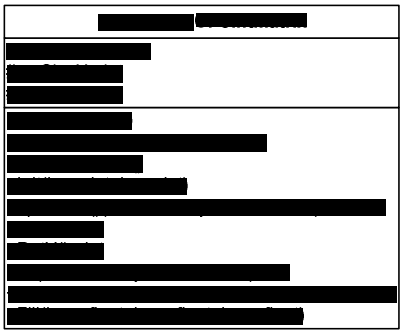


Диаграмма 4.12. Класс CPointMatrix.

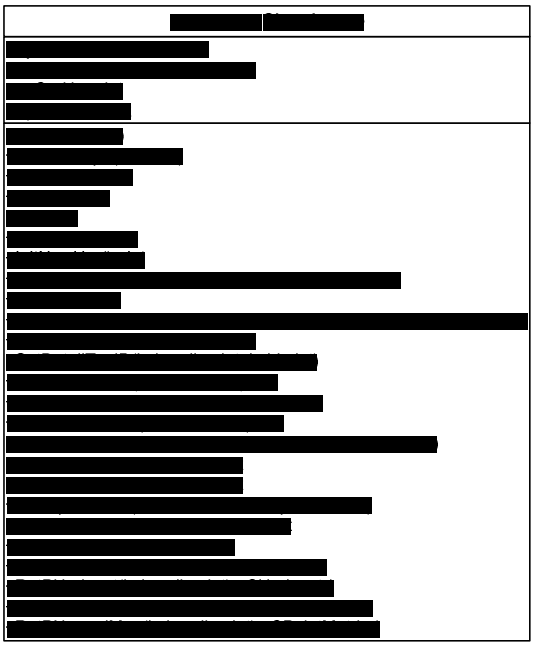


Диаграмма 4.13. Класс CLandscape.

Core3D::**CLightSource**
+myColor : CColor +myPosition : CColor +myDirection : CPoint3D +myLightNum : int +myLightType : int +myParam : float
+CLightSource() +CLightSource(in lighttype : int, in lightnum : int) +~CLightSource() +operator =(inout that : CLightSource) +Init(in lighttype : int, in lightnum : int) +ok(in type : int) : int +SetColor(in r : float, in g : float, in b : float, in a : float) +SetColor(in rgba : float*) +SetColor(inout color : CColor) +SetPosition(in x : float, in y : float, in z : float, in w : float) +SetPosition(in xyzw : float*) +SetPosition(inout pos : CColor) +SetDirection(in x : float, in y : float, in z : float) +SetDirection(in xyz : float*) +SetDirection(inout dir : CPoint3D) +SetCutoff(in cutoff : float)

Диаграмма 4.14. Класс CLightSource.

November 14, 2008 12:17 AM -- 562

CCamera
-myPosition : CPoint3D -myDirection : CPoint3D -myUpPoint : CPoint3D -mySpeed : CPoint3D -myLeftX : float -myRightX : float -myLeftY : float -myRightY : float -myNearZ : float -myFarZ : float -myUpperLeft : CPoint3D -myUpperRight : CPoint3D -myBottomLeft : CPoint3D -myBottomRight : CPoint3D -myIsRotateFromMouse : bool
-CountSpeed() -CountPoints() +CCamera() +CCamera(in ex : float, in ey : float, in ez : float, in lx : float, in ly : float, in lz : float, in ux : float, in uy : float, in uz : float) +~CCamera() +Set(in ex : float, in ey : float, in ez : float, in lx : float, in ly : float, in lz : float, in ux : float, in uy : float, in uz : float) +MakeCurrent() +SetFrustum(in leftx : float, in rightx : float, in lefty : float, in righty : float, in nearz : float, in farz : float) +MoveAt(in x : float, in y : float, in z : float) +LookAt(in x : float, in y : float, in z : float) +Follow(inout point : CPoint3D) +Seek(inout point : CPoint3D) +SetPosition(in x : float, in y : float, in z : float) +Remove() +Slide(in k : float) +Rotate(inout point : CPoint3D, in a : float, inout axis : CPoint3D) +RotateUp(in a : float) +RotateZ(in a : float) +RotationCheck(inout point : CPoint3D) : int +SetViewFromMouse(in centerX : int, in centerY : int) +RetMinX() : float +RetMaxX() : float +RetMinY() : float +RetMaxY() : float +RetSpeed() : CPoint3D &

Диаграмма 4.15. Класс CCamera.

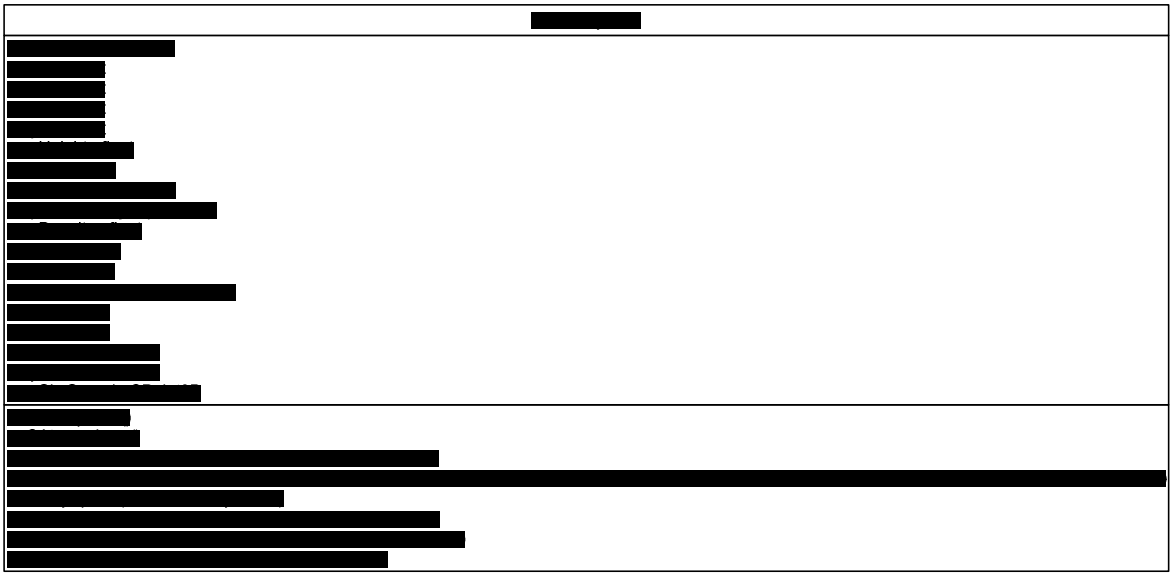


Диаграмма 4.16. Класс CAtmosphere.

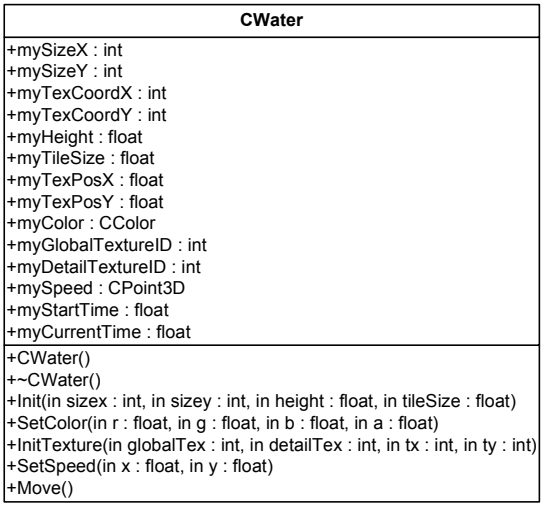


Диаграмма 4.17. Класс CWater.

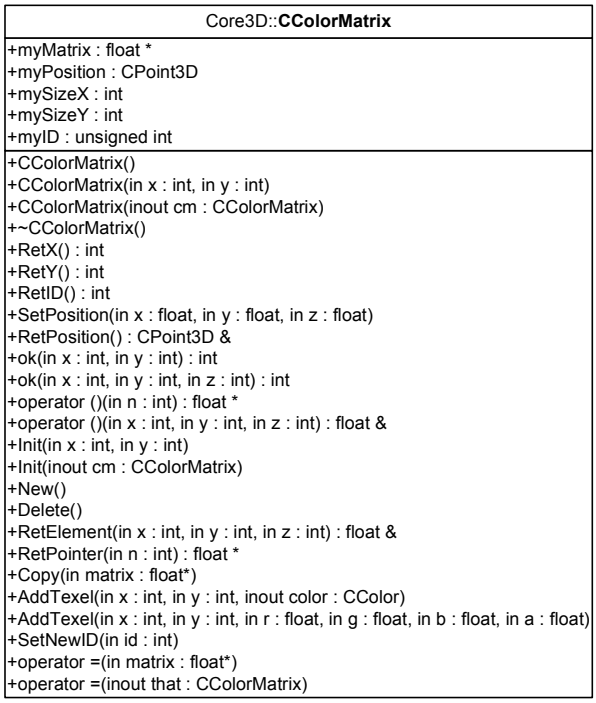


Диаграмма 4.18. Класс CColorMatrix.