THE SOFTWARE FRAMEWORK FOR MULTI-AGENT SYSTEMS RESEARCH

Research Paper for Intel Science and Engineering Fair
(ISEF Competition – 2003), Cleveland, Ohio, USA

ANDREY TATARINOV
VASILY FEDOSEEV
VSEVOLOD USTINOV

The aim of the project was to create software toolkit for developing and exploring multi-agent systems (named *Elendor*), which are widely used in AI research. The toolkit includes: IDE for 2 programming languages (source editor, compilers, virtual machines and debuggers), the tools for multi-agent systems support and 3D visualization. The open component architecture is used as a main approach. All the components are plug-in modules and their loading and linking is based on the ideology of Component Object Model.

The Core consists of: (1) the Linker module, which is responsible for generic inter-component and inter-language communication; (2) the event-driven Execution Dispatcher, which controls threads and processes; (3) the objects Database; (4) Structural Event Journaling and Exception Dispatcher and (5) System Console for user direct control.

The Shell consists of: (6) Agent Server, which provides agents' creation and interaction; (7) the Physical module for modeling of virtual world where agents can move and collide; (8) Visualization module based on OpenGL which allows monitoring agents' behavior and (9) the Sound Engine.

All the modules are platform-independent and based on Software Abstraction Layer, which virtualizes processing of OS-specific tasks (OS-in-OS concept). Dou to this approach, 2 Mb of source code was ported from Win32 to Linux platform within 1 month.

The languages used now to describe agents behavior and configuration is Smalltalk and simple C-like language. The translators compile sources to byte-code, which is executed by Execution Dispatcher through Linker, when agent's method is called. The system can also support other programming languages.

On the whole, the project's possibilities allow user to create and improve agent-controlling algorithms, and multi-agent systems for efficient education and research.

Русская версия

1. ПОСТАНОВКА ЗАДАЧИ

В настоящее время получила распространение технология мультиагентных систем. Мультиагентная система – это система, состоящая из агентов и объектов. Главное свойство агентов – способность принимать решения о каких-либо действиях в зависимости от внешних факторов. Под понятием интеллекта агентов подразумеваются только поведенческие аспекты его деятельности. Объекты же статичны и не обладают таким свойством. Фактически, объекты являются частью мультиагентной среды – виртуального мира, в котором живут и действуют агенты.

Мультиагентные системы применяются в различных областях – например, для исследования свойств и поведения различных популяций в замкнутых средах, или для создания игрового мира, где каждый персонаж является агентом.

Но зачастую программист, занимающийся разработкой мультиагентных систем, вынужден также разрабатывать средства, позволяющие отлаживать созданные им алгоритмы поведения агентов и исследовать их свойства.

Таким образом, появилась задача разработки комплекса программных средств разработки мультиагентных систем (*Elendor*), используя который программист сможет не заботиться о средствах исследования мультиагентных систем, сконцентрировавшись на разработке алгоритмов интеллекта агентов.

Фактически, комплекс является набором инструментов, позволяющих не только проводить исследовательскую работу, связанную с искусственным интеллектом, но и выполнять широкий круг задач, где возможно применение агентов и мультиагентных систем. Комплекс можно сравнить с рабочим местом, которое его пользователь сможет настраивать для своих задач. В руки пользователя попадает гибкое и настраиваемое средство, функциональные возможности которого могут быть расширены или изменены в соответствии с задачами, которые собирается выполнять пользователь с его помощью.

Комплекс является своеобразным «конструктором», в котором пользователь может смоделировать условия нужной ему задачи. Возможность запрограммировать не только все аспекты поведения агентов, но и любой из аспектов поведения комплекса и его компонентов позволяет пользователю планировать и конструировать условия проведения исследований. Комплекс позволяет не только в реальном времени демонстрировать результаты исследования, но и записывать результаты в текстовом виде с сохранением всех числовых значений, что зачастую бывает более важным при исследовании.

Комплекс построен на гибкой открытой архитектуре, позволяющей подключать новые модули с новыми функциональными возможностями или изменять и расширять уже существующие модули. Все интерфейсы модулей стандартизованы и открыты, чтобы предоставить пользователю возможность самому добавлять в комплекс новые средства и возможности.

Важным требованием к комплексу является повышенная надежность при работе, необходимая при исследовании длительных процессов. Все ошибки, произошедшие во время работы комплекса, должны обрабатываться, не вызывая прекращения выполнения текущих запущенных задач комплекса. Работоспособность комплекса должна поддерживаться даже при отсутствии некоторых модулей, при невозможности загрузить указанные в модульной конфигурации компоненты.

Комплекс должен включать в себя средства разработки интеллекта агентов, такие как редактор исходного текста алгоритмов, а также компиляторы и отладчики внешних языков комплекса, выбранных в качестве основных языков алгоритмического моделирования. Этими языками в настоящее время являются Smalltalk и Elendor C++, являющийся подмножеством стандартного языка C++. Базой для разработки мультиагентных систем является стандартная библиотека, в которую входит сервер агентов – компонент комплекса, позволяющий поддерживать существование агентов и их взаимодействия на уровне общения друг с другом. Также комплекс должен включать в себя средства, позволяющие моделировать мультиагентную среду и настраивать ее физические свойства, а также помещать в нее агентов. В мультиагентной среде агенты могут взаимодействовать друг с другом через физические взаимодействия.

Важной частью комплекса являются модули, позволяющие наблюдать за действиями агентов в мультиагентной среде и исследовать свойства агентов. Это такие модули, как графический и звуковой, позволяющие следить за действиями агентов в реальном времени, а также модуль статистики, позволяющий собирать информацию об агентах.

2. Концептуальная основа комплекса

Архитектура комплекса базируется на таких понятиях, как расширяемость и модульность, комплекс является надежным и переносимым на разные платформы. Поэтому можно

утверждать, что основой комплекса является концепция OS-in-OS (Operation System in Operation System, операционная система в операционной системе). Существует ядро и модульная оболочка вокруг него. Ядро комплекса позволяет отделить и абстрагировать его компоненты от операционной системы, на которой они в данный момент работают.

Ядро позволяет организовать работу модулей как приложений операционной системы, используя многозадачность, рассылку сообщений, регистрацию событий, диспетчеризацию оперативной памяти внутри системы. Все эти процессы происходят без непосредственного участия внешней операционной системы. Все вызовы функций операционной системы инкапсулированы в специальных модулях, являющихся прослойкой абстрагирования от операционной системы. Эти модули являются сменными и позволяют переносить комплекс на разные платформы без изменения основных его компонентов, организуя независимость модулей и процессов, происходящих в них, от операционной системы.

Комплекс выполняет задачу по построению внутренней мультиагентной операционной системы, где на каждый агент ложатся задачи по выполнению определенных функций, таких как, например, построение интеллектуальных менеджеров информации, позволяющих организовать удобный доступ к информации, находящейся в модуле хранения данных. Каждое приложение в этой операционной системе является агентом, и комплекс предоставляет средства для разработки таких приложений – редактор исходного текста, трансляторы языков программирования. Комплекс позволяет наблюдать за работой его «приложений» - за действиями агентов в мультиагентной системе.

Агент в мультиагентной системе сконструирован как набор представлений в различных модулях. Для физического модуля это набор его физических параметров, для графического – трехмерная геометрическая форма агента, для языковой подсистемы – набор скомпилированных функций этого агента. Эти проекции на различные модули взаимно ортогональны друг другу, и возможно использование агента с отсутствующей проекцией или проекциями. В этом случае модули, работающие с этими проекциями, не будут оказывать влияния на деятельность агента.

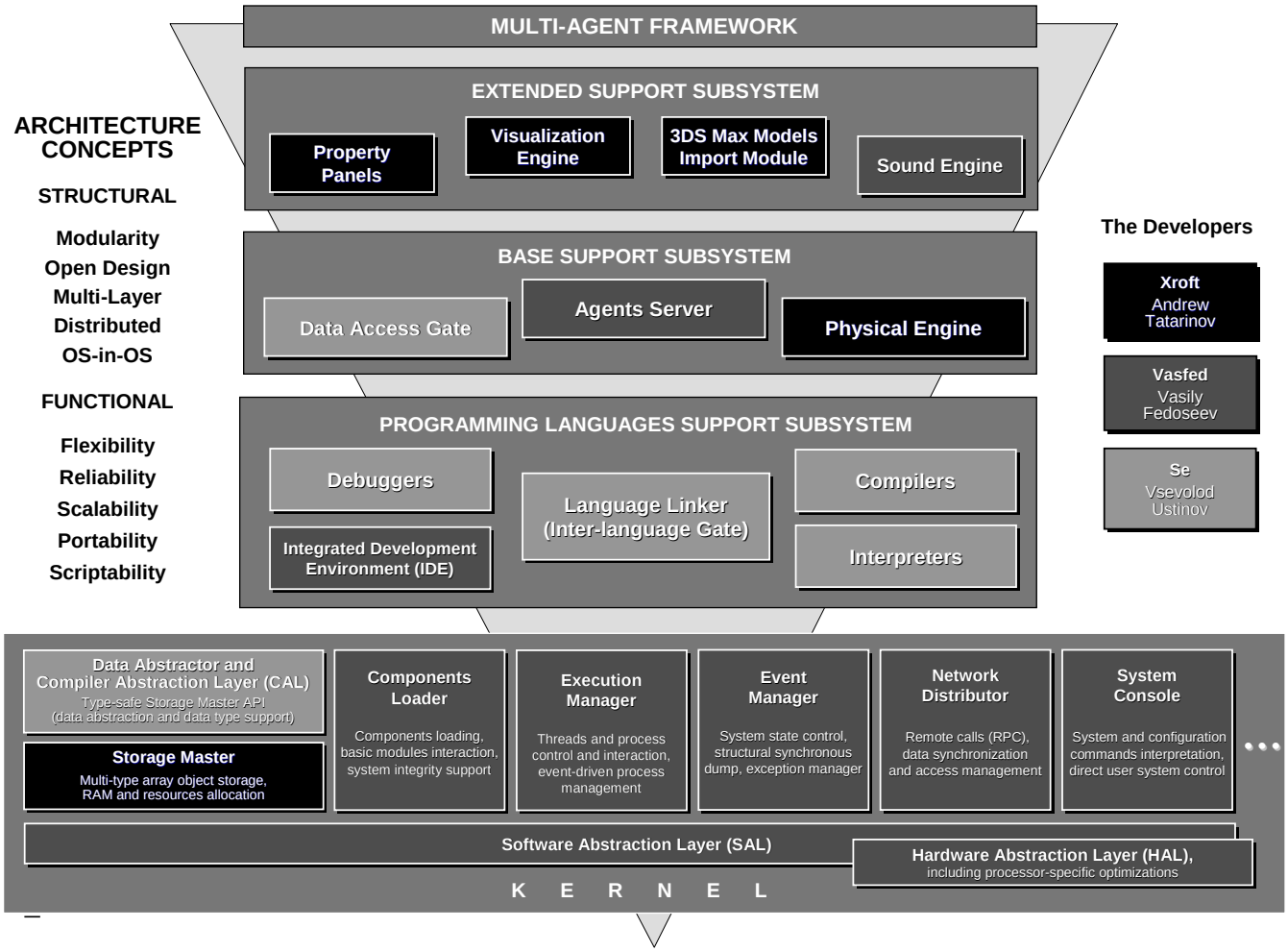
Таким образом, возможна настройка цикла жизнедеятельности агента – будет ли он участвовать в физических взаимодействиях или будет выполнять сложные вычисления, оказывающие влияние на работу остальных агентов системы. Агенты необязательно должны быть представлены в виде геометрических форм. Может существовать агент, например, выполняющий функции атмосферы, или агент, собирающий информацию о жизни других агентов.

Понятие агента в мультиагентной операционной системе универсально и позволяет использовать его в различных областях научной и технической деятельности, используя комплекс как средство создания, разработки и исследования поведения агентов и как средство организации внутренней операционной системы, работающей с агентами.

3. Архитектура комплекса

Архитектура комплекса является открытой, гибкой для настройки и поддерживает перенос на другие платформы. Пользователь получает возможность полностью контролировать состояние комплекса, расширять и дополнять его функциональные возможности, подстраивая комплекс под свои задачи. Также архитектура комплекса обеспечивает надежность его работы в условиях, когда произошла ошибка времени исполнения или при отсутствии каких-либо компонентов, нужных при работе. В этом случае работа комплекса не будет приостановлена, и уже загруженные компоненты будут продолжать свою работу.

При работе с комплексом пользователь получает возможность выбрать подходящую для решения его задач платформу и с малыми временными затратами перенести комплекс на нее. Все компоненты комплекса не зависят от платформы, на которой они в данный момент работают, а архитектура построена таким образом, что позволяет вынести всю зависимую от платформы часть в ряд небольших сменных модулей, которые пользователь может изменить для перехода на другую платформу.



Pic 3.1. Elendor System Architecture

Основой архитектуры комплекса является ядро (pic 3.1), позволяющее комплексу работать на разных платформах, подключать к комплексу различные компоненты, построенные по идеологии COM, управлять работой комплекса, организовывать сетевое взаимодействие различных его частей и распределенные вычисления.

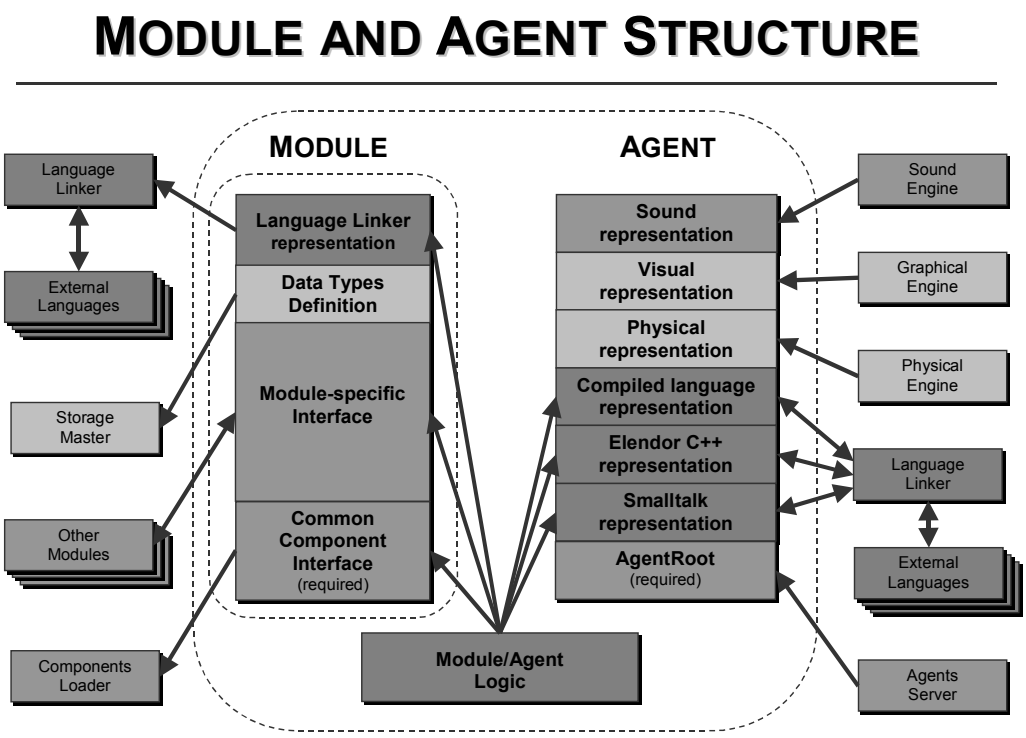
Также на ядро ложатся функции по выводу информации о текущем состоянии комплекса и регистрации различных событий, распределению оперативной памяти, используемой при работе и организации работы с пользователем на низком уровне через ввод команд, влияющих на работу комплекса.

Ядро может осуществлять свою работу и при отсутствии некоторых своих модулей. Такая функция нужна для обеспечения повышенной работоспособности и устойчивости в непредвиденных ситуациях.

Каждый компонент комплекса (включая модули ядра и модули уровней SAL, HAL и CAL) имеет открытый интерфейс, являющийся его API (Application Programming Interface). Это позволяет пользователю использовать любой из существующих компонентов наравне с остальными, либо заменять уже созданные модули своими.

3.1. Elendor SDK (Software Development Kit)

Каждый компонент имеет набор интерфейсов разного уровня (pic 3.2). Общий интерфейс модуля (Common Component Interface) позволяет модулю-загрузчику подключить его к комплексу. Собственно интерфейс модуля (Module-specific Interface) позволяет использовать функции, входящие в его состав, и позволяет модулю выполнять свои задачи. Описания типов данных (Data type Definitions) позволяют компоненту экспортировать в модуль хранения данных свои типы данных. Интерфейс для модуля-компоновщика (Language Linker Representation) позволяет вызывать функции компонента из программ, написанных на внешних языках.



Pic 3.2. Module And Agent Structure

Для создания компонентов комплекса был разработан специальный SDK, включающий в себя описания API всех компонентов, включенных в стандартный набор комплекса, Elendor Module Wizard для Visual C++ 7.1 и шаблоны компонента для Linux-разработчиков. Они позволяют значительно ускорить процесс создания новых модулей и их подключения к комплексу, так как созданный компонент уже будет содержать оптимальную структуру входящих в него файлов, алгоритмы подключения и загрузки основных модулей ядра, а также набор обязательных директив, необходимых при создании компонента, таких как его GUID, имя экспортируемого интерфейса и т.д.

SDK был использован при создании большинства компонентов, входящих на данный момент в состав комплекса. Это доказывает используемость комплекса и его удобство. SDK предоставляет сторонним разработчикам компонентов возможность участвовать в развитии комплекса наравне с его авторами.

3.2. Software Abstraction Layer

Уровень абстрагирования от операционной системы (Software Abstraction Layer, SAL) является важной частью комплекса, позволяющей переносить его на разные операционные системы. Модули этого уровня инкапсулируют в себе низкоуровневые функции, использующие обращения к операционной системе, предоставляя компонентам комплекса высокоуровневые интерфейсы. Используя эти интерфейсы, компоненты не обращаются напрямую к операционной системе, под которой они в данный момент работают.

Таким образом, компоненты становятся на уровне исходных текстов независимыми от операционной системы. Для переноса их на другую платформу нужно заменить существующие SAL-модули теми, в которых инкапсулированы вызовы функций операционной системы, на которую осуществляется перенос. Таким образом, при переносе на другую платформу исходный текст самих компонентов системы остается неизменным.

3.3. Модуль-загрузчик (loader)

Основой ядра и его единственной обязательной частью, без которой работа с комплексом осуществляться не может, является модуль-загрузчик. Этот модуль предоставляет пользователю возможность подключать к комплексу различные компоненты, позволяющие расширить функциональные возможности комплекса.

Все компоненты комплекса (модули) построены по идеологии COM (но не по технологии COM, взята лишь идея). Каждый модуль при подключении к комплексу модулем-загрузчиком предоставляет свой интерфейс, через который другие модули комплекса могут с ним работать, вызывая нужные функции интерфейса. Загрузчик позволяет получить интерфейс любого модуля, подключенного к комплексу, используя уникальный идентификатор нужного интерфейса.

Также модуль-загрузчик предоставляет возможность подключения компонентов, построенных по технологии COM.

3.4. Менеджер исполнения (execution manager)

Менеджер исполнения позволяет организовать работу комплекса, вызывая основные функции подключенных модулей на каждой итерации главного цикла комплекса. Каждый компонент системы может предоставить менеджеру исполнения функцию, которая должна вызываться во время работы комплекса, и установить приоритет выполнения этой функции, то есть, как часто функция будет вызываться, например, на каждой итерации главного цикла или через одну итерацию.

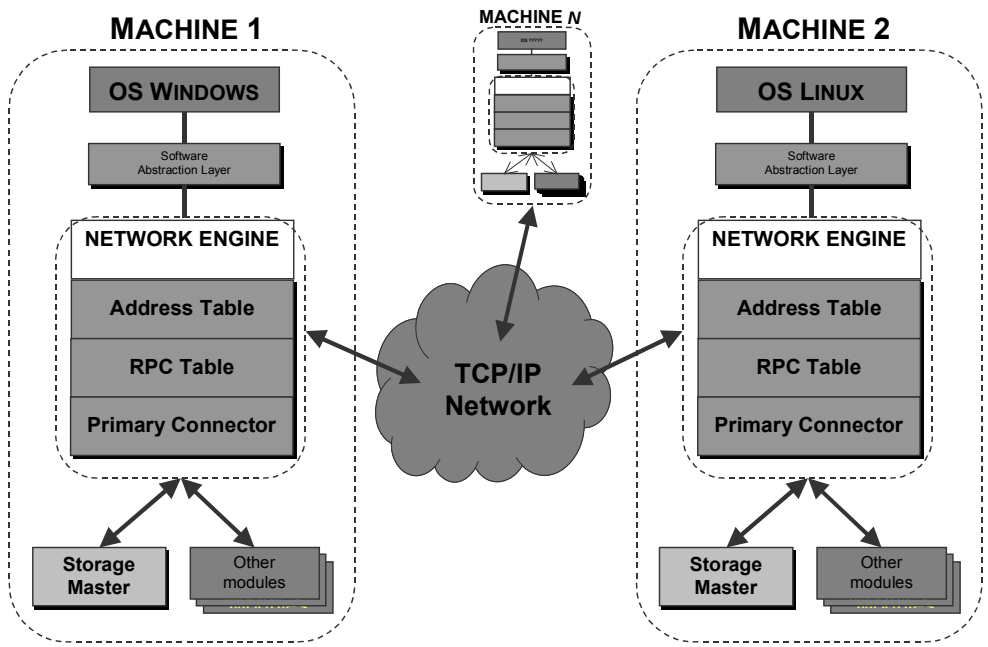
Менеджер исполнения позволяет управлять различными потоками, организуя многопоточность при работе комплекса. Каждый компонент может работать в своем потоке, таким образом, во время работы комплекса может выполняться сразу несколько задач. Также менеджер исполнения ведет статистику времени работы каждой функции, зарегистрированной для исполнения во время работы главного цикла.

В состав менеджера исполнения входит система рассылки сообщений о событиях, произошедших во время работы комплекса. Каждый компонент системы может получить информацию о произошедшем событии и обработать ее.

3.5. Сетевой модуль

Сетевой модуль позволяет организовать работу комплекса на нескольких компьютерах (pic 3.3). При этом возможно распределение вычислений на разных компьютерах, то есть распределение агентов, находящихся в мультиагентной системе, между ними. В этом случае каждый компьютер просчитывает информацию только о своих агентах и синхронизирует ее с остальными компьютерами в соответствии с заданными настройками.

Возможен запуск разных модулей комплекса на разных компьютерах в сети, то есть распределение вычислений между ними по различным задачам. На разных компьютерах могут производиться физические расчеты, исполняться алгоритмы агентов, а затем вся информация будет отправляться на компьютер (компьютеры), выполняющий визуализацию.



Pic 3.3. Network Support

Сетевой модуль позволяет организовать сетевую работу модулей хранения данных, находящихся на разных компьютерах в сети. В каждом таком модуле создается побитовая копия информации обо всех агентах, находящихся в мультиагентной системе в данный момент. Такой метод позволяет сэкономить системные ресурсы, тратящиеся на передачу данных по сети и на обращение к удаленным хранителям данных. Наряду с этим существует возможность указать сетевому модулю функции, позволяющие осуществлять передачу данных в тех случаях, когда побитовое копирование неприменимо.

Использование сетевого модуля позволяет организовать распределение вычислений на разных компьютерах и ускорить работу комплекса.

3.6. Модуль регистрации событий

Важной частью ядра комплекса является модуль регистрации событий. Он позволяет в текстовом виде отображать информацию о ходе загрузки и работы системы, выводить отладочную информацию и сообщения об ошибках, возникших в ходе работы комплекса.

Модуль регистрации событий позволяет обрабатывать исключительные ситуации, возникшие в ходе работы комплекса, и выводить информацию о них, отображая название модуля, в котором произошла исключительная ситуация, и участок исходного текста этого модуля, который породил исключение.

Также модуль регистрации системы позволяет вести протоколирование работы комплекса, то есть сохранять информацию о загрузке и выгрузке модулей, о занятой при работе оперативной памяти, о добавлении или удалении агентов, об их действиях в мультиагентной среде.

3.7. Модуль хранения данных (storage master)

Модуль хранения данных позволяет работать с доступной оперативной памятью, распределяя ее для создания внутри системы различных объектов. Этот модуль позволяет осу-

ществлять связь между компонентами системы с помощью работы с общими данными, используемыми несколькими компонентами для разных целей. Работа с общими данными позволяет избежать дублирования данных в различных модулях, а также значительно повысить скорость работы с памятью во многих случаях.

Модуль хранения данных предоставляет низкоуровневый интерфейс для работы с оперативной памятью. Он позволяет выделять память для работы комплекса, при этом возможно выделение памяти несколькими способами, и выбор каждого способа зависит от конкретной задачи, для которой потребовалась память.

Модуль базируется на нескольких типах аллокаторов (от английского слова allocate – размещать, распределять). Первый тип аллокаторов – это стековый аллокатор, в котором выделение и освобождение кусков памяти осуществляется по принципу стека, когда освободить можно лишь тот кусок памяти, который был выдан модулем хранения данных последним и находится последним в списке выделенных блоков памяти. Второй тип – это аллокатор объектов фиксированного размера, третий – аллокатор объектов маленького размера, являющийся диспетчером аллокаторов фиксированного размера, и четвертый – это аллокатор, использующий стандартные функции распределения памяти.

Модуль хранения данных предоставляет возможности по организации работы с данными при использовании многопоточности, а также централизует, стандартизует и упрощает процесс полного сохранения и загрузки комплекса или его частей.

Модуль хранения данных позволяет хранить информацию, инкапсулируя работу с указателями и работая только с хэндлами (handles) на данные. При работе с этим модулем происходит потеря безопасности типов данных, а это неприемлемо при работе с ними. Инкапсулировать работу с хэндлами и вернуть проверку типов данных позволяет использование группы классов, получившей название Smart Storage Master.

Этот набор классов включает в себя шаблонные контейнеры для объектов разных типов, поведение этих контейнеров определяется задаваемыми для них стратегиями (policies, Александреску, 2002), а также высокоуровневую оболочку для Storage Master, позволяющую регистрировать типы данных и работать с ними. Все функции по работе с типами данных вынесены в CAL (Compiler Abstraction Layer), что позволяет использование в модуле хранения данных компиляторозависимых типов.

Такая организация надстройки над модулем хранения данных позволяет восстановить безопасность типов данных на уровне языка C++, на котором написано большинство компонентов, использующих этот модуль, а также сравнивать удобство использования этого модуля с использованием стандартных операторов по работе с памятью.

3.8. Консоль

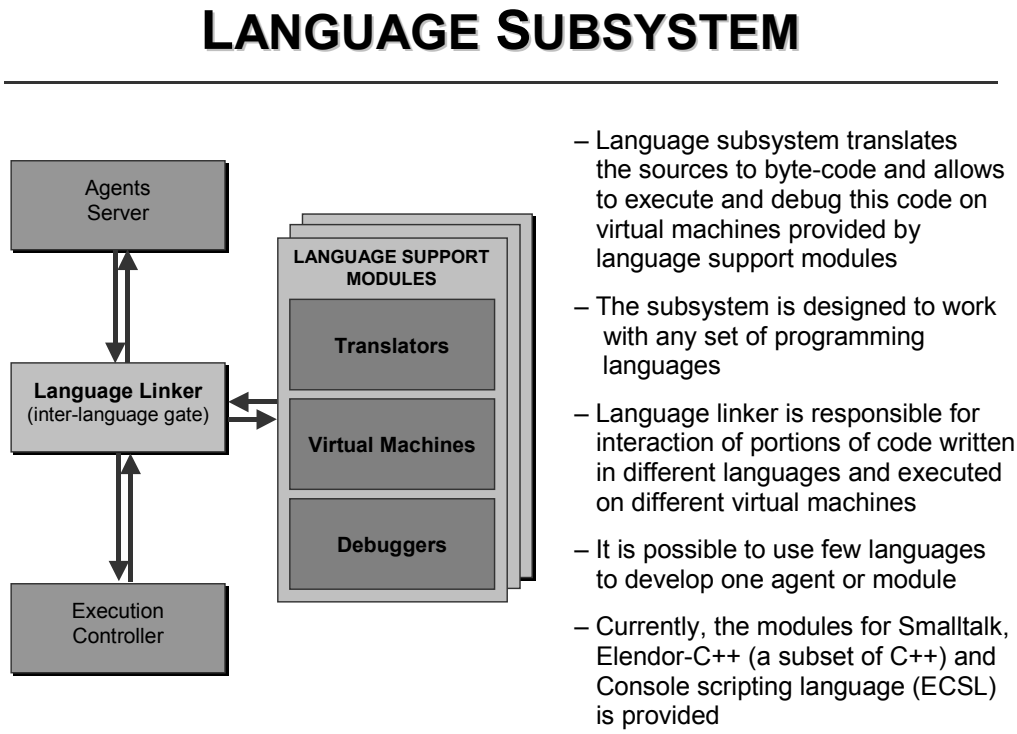
Консоль является модулем, позволяющим пользователю взаимодействовать с системой, загружая различные модули, вызывая их функции, добавляя агентов в мультиагентную систему и задавая их свойства. Консоль позволяет осуществлять текстовый ввод команд, оказывающих влияние на работу системы. С помощью этих команд пользователь может контролировать все аспекты работы комплекса.

Каждый модуль комплекса может экспортировать в консоль свой набор команд, вводя которые, пользователь может контролировать работу этого модуля. Эти команды регистрируются в консоли на этапе начальной инициализации компонентов комплекса, происходящей после их загрузки. Также модули выводят в консоль информацию о своей загрузке, выгрузке, о своем текущем состоянии и о произошедших ошибках.

Для удобства работы консоль поддерживает использование пакетных файлов, в которых содержится информация о командах, которые необходимо исполнить. Также пользователь имеет возможность расширить набор команд, используемых в консоли, для удобства работы с комплексом при выполнении конкретной задачи.

4. Языковая подсистема

Модули комплекса, входящие в языковую подсистему, позволяют создавать агентов в мультиагентной системе, описывая на внешних языках программирования, поддерживаемых комплексом, алгоритмы их искусственного интеллекта, связывать агентов с модулями комплекса, осуществляя межъязыковое взаимодействие, а также отлаживать написанную программу.



Pic. 4.1 Language Subsystem

Компоненты языковой системы осуществляют поддержку внешних языков комплекса. Для каждого внешнего языка, входящего в состав комплекса, существует свой набор таких модулей, состоящий из компилятора, виртуальной машины языка и отладчика (pic 4.1). Благодаря открытой архитектуре, открытым интерфейсам всех компонентов системы и возможности подключения новых модулей к комплексу возможно добавление пользователем своих языков. Для этого ему нужно разработать компилятор, виртуальную машину и отладчик языка, который должен быть использован в комплексе, и подключить их к комплексу, при этом возможно использование уже существующих виртуальных машин и отладчиков.

Модуль-компоновщик и среда разработки являются общими для разных языков и позволяют осуществлять взаимодействие между ними.

4.1. Модуль-компоновщик (linker)

Модуль-компоновщик, являясь одним из связующих звеньев между компонентами системы, позволяет осуществлять межъязыковое взаимодействие, то есть программа, написанная и скомпилированная с использованием внешних языков комплекса, получает возможность взаимодействовать с компонентами системы, заранее скомпилированными в динамически подключаемые библиотеки.

Каждый модуль может предоставить компоновщику особый интерфейс, функции которого будут зарегистрированы и станет возможным их вызов из созданных пользователем

программ на внешних языках. Также компоновщик регистрирует виртуальные машины всех языков, используемых при работе с комплексом. При поступлении запроса на вызов функции, написанной на одном из внешних языков, этот запрос переадресовывается виртуальной машине этого языка.

Благодаря такой структуре вызовов возможно описание разных методов класса агента на разных языках системы. Таблица функций класса не зависит от языка, на котором он написан. Информация о языке программирования становится важной только при вызове функции.

Модуль-компоновщик может не только выполнять функции шлюза между программами на разных языках, но и работать как модуль, позволяющий осуществлять связь между компонентами комплекса. Каждый компонент комплекса может получить интерфейс компоновщика и, используя его, вызвать функцию другого компонента, если она зарегистрирована в компоновщике. При этом модуль, вызывающий функцию, может и ничего не знать об интерфейсе компонента, у которого эта функция вызвана.

4.2. Компиляторы, виртуальные машины и отладчики языков

Благодаря модульной архитектуре в комплекс может быть включено несколько компиляторов разных языков. На данный момент в состав комплекса включены компиляторы языков Smalltalk и Elendor C++, но возможно добавление и других компиляторов, разработанных пользователем комплекса.

Компиляторы, уже входящие в состав комплекса, обрабатывают исходный текст программы в два этапа. Во время прохождения первого этапа методом конечных автоматов проводится лексический анализ текста программы. Текст представляется в виде набора логических лексем, удобных при разборе грамматики языка.

На втором этапе поток лексем обрабатывается синтаксическим анализатором и методом рекурсивного спуска и преобразуется во внутренний байт-код, используемый виртуальной машиной при исполнении функций, написанных на данном языке. Возможно использование динамической рекомпиляции программы во время ее работы, когда необходимо внести в алгоритм изменения и сразу же увидеть их результат. При этом старый байт-код функций заменяется новым, полученный при рекомпиляции.

Из-за того, что после компиляции функция представляется не в виде набора команд процессора, а в виде внутреннего байт-кода, обычный вызов функции невозможен. Для того чтобы правильным образом исполнить такую функцию, необходимо использование виртуальной машины. Виртуальная машина языка позволяет исполнять байт-код, являющийся набором простых низкоуровневых команд.

Для каждого языка необходимо создание своей виртуальной машины, но если при компиляции программ, написанных на разных языках, используется одинаковый байт-код, возможно использование одной виртуальной машины.

Отладчик является важным рабочим инструментом в руках пользователя комплекса. Этот компонент позволяет производить пошаговое исполнение программы, а также наблюдать за состоянием переменных-параметров агентов. Эти две возможности позволяют пользователю эффективно находить логические ошибки в программе и улучшать ее. Важным свойством уже включенных в состав комплекса отладчиков языков Smalltalk и Small C++ является возможность во время отладки проходить текст программы не по строкам, как реализовано в большинстве классических отладчиков, а по отдельным выражениям, что позволяет с большей точностью отслеживать происходящие при работе программы события.

4.3. Среда разработки (IDE)

Среда разработки (Integrated Developer’s Environment, IDE) является важной составной частью комплекса, так как именно этот компонент организовывает удобный пользовательский интерфейс для работы с компиляторами языков и их отладчиками. Пользователь может

и не применять возможности этого модуля, вызывая команды компилятору на обработку исходного текста программы напрямую из консоли, указав имя файла с исходным текстом. Но при такой работе пользователь лишается возможности удобного редактирования и отладки текста программы.

Среда разработки включает в себя текстовый редактор, позволяющий пользователю вводить исходный текст программы. Для языка Smalltalk реализована такая функция, как подсветка синтаксиса, которая позволяет выделять цветом различные по логике лексемы языка. Модуль подсветки синтаксиса является зависимым от языка, но подключается уже напрямую к среде разработки.

Использование среды разработки при отладке программы позволяет быстро выставлять точки остановки (breakpoints) на конкретные выражения в тексте, исполнять программу по шагам, просматривать локальные переменные. Использование горячих клавиш позволяет ускорить процесс работы в среде разработки.

5. Подсистема базовой поддержки работы мультиагентных систем

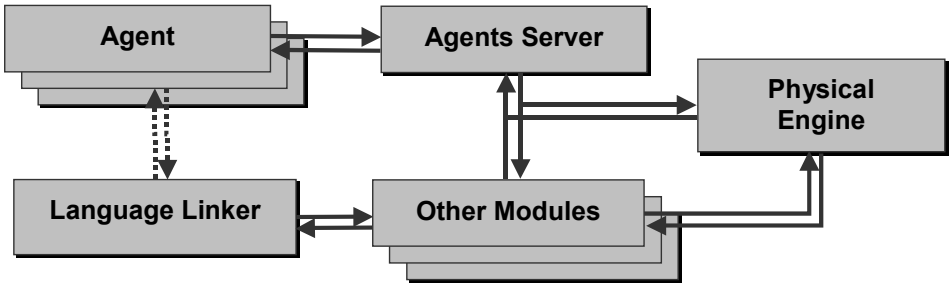
Подсистема базовой поддержки работы мультиагентных систем позволяет организовать жизненный цикл агентов в ней, а также обрабатывать взаимодействия агентов друг с другом и с объектами мультиагентной систему на разных уровнях.

Важной частью любой мультиагентной системы является сервер агентов (pic. 5.1) – модуль комплекса, позволяющий добавлять и удалять агентов и объектов. Фактически, в сервере агентов хранится вся информация о текущем состоянии мультиагентной системы, но не о каждом агенте в отдельности.

MULTI-AGENT SYSTEMS

AGENTS' PROPERTIES

– Agents don't concern any information about external modules
– Agents initialize interactions with each other via Agents Server (and may continue the interaction directly for optimization)
– Agents may use any module which has exported its interface to Language Linker
– Agents are composed from orthogonal projections to a subset of system modules at current system state



Pic 5.1. Multi-Agent Systems

Взаимодействия между агентами могут происходить на двух разных уровнях: на физическом уровне и на уровне общения между агентами. Агенты могут общаться друг с другом с

помощью рассылки сообщений друг другу. На физическом уровне агенты могут взаимодействовать посредством различных физических взаимодействий – столкновений, воздействий электромагнитных полей. Эти взаимодействия фиксируются и обрабатываются физическим модулем.

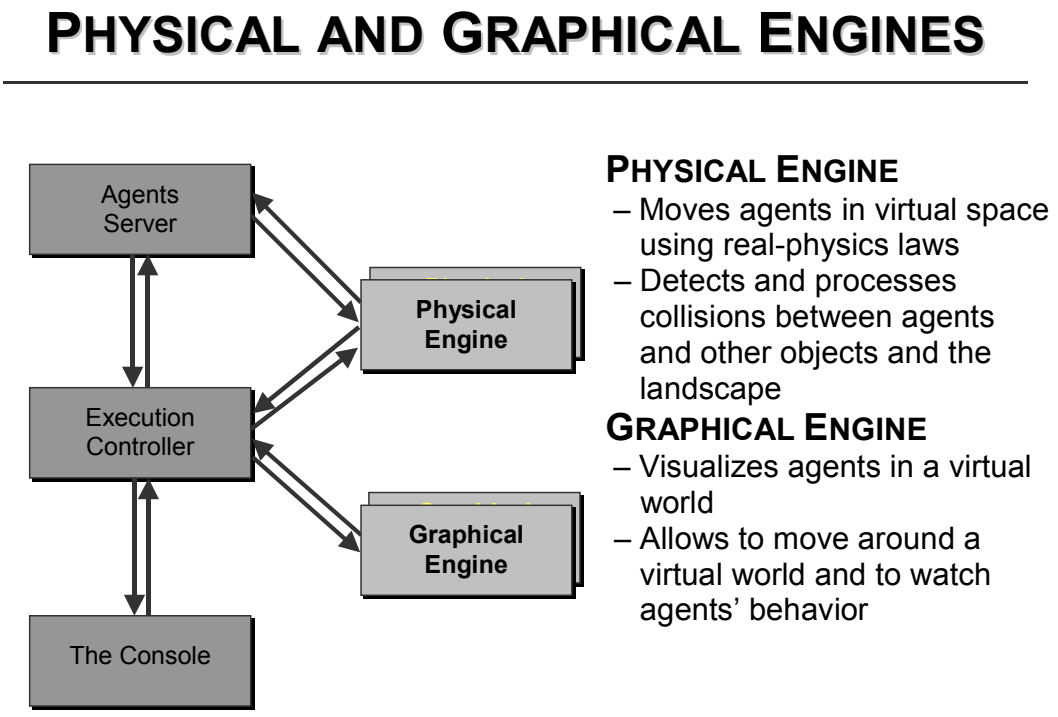
5.1. Структура агента и сервер агентов

Сервер агентов хранит информацию обо всех существующих в данный момент агентах и позволяет осуществлять их быстрый поиск по их типам или по каким-либо еще известным параметрам. Каждый агент содержит в себе список своих представлений в разных модулях комплекса (pic 3.2), комбинация которых определяет его точный тип. Эти представления могут быть либо объектами компилируемых внешних языков, либо данными, «приклеенными» другими модулями, например физическим (скорость и ускорение агента), графическим (трехмерная модель агента) или любым другим компонентом комплекса.

При вызове функций, «приклеенных» к агенту, вызов переадресовывается компоновщику, который находит среди зарегистрированных в нем функций нужную и производит ее вызов. Такая структура построения агента позволяет описывать его сразу на нескольких языках, причем не только на внешних языках, но и используя код, скомпилированный заранее.

5.2. Физический модуль

Физический модуль (pic 5.2) позволяет поместить агентов в мультиагентную среду – виртуальный мир с реальными физическими законами – и совершать там различные действия: перемещаться, сталкиваться друг с другом и с объектами среды, такими, как ландшафт.



Pic 5.2. Physical and Graphical Engines

В основу физической модели мультиагентной среды легло то утверждение, что все агенты с физической точки зрения являются материальными точками. Они не могут быть деформированными при столкновении или получить вращательный момент, начать вращаться. Эта

модель достаточно проста, но благодаря архитектуре комплекса физический модуль может быть заменен более совершенным, доработанным компонентом с измененной моделью, или его возможности могут быть дополнены пользователем комплекса с применением внешних языков системы.

Агенты в мультиагентной среде могут перемещаться равномерно или равноускоренно, при этом учитываются такие их параметры, как скорость и ускорение. Также агенты могут сталкиваться друг с другом. Проверка на столкновение двух агентов осуществляется с помощью проверки на пересечение пространств охватывающих параллелепипедов (bounding boxes) этих агентов. Если зафиксировано это пересечение, то считается, что произошло столкновение двух агентов. Это столкновение обрабатывается с помощью законов сохранения импульса и энергии, и в результате агенты получают новые значения скоростей.

Агент может столкнуться с ландшафтом. Столкновение с ландшафтом считается зафиксированным, если хотя бы одна из вершин охватывающего параллелепипеда оказалась под рельефом. При обработке такого столкновения считается, что ландшафт обладает бесконечной массой, и изменение его скорости незначительно мало, а у агента, столкнувшегося с ним, меняется только вектор скорости, но не ее модуль.

Физический модуль позволяет агентам взаимодействовать на физическом уровне путем столкновений, изменяя параметры их физических представлений.

6. Подсистема расширенной поддержки работы мультиагентных систем

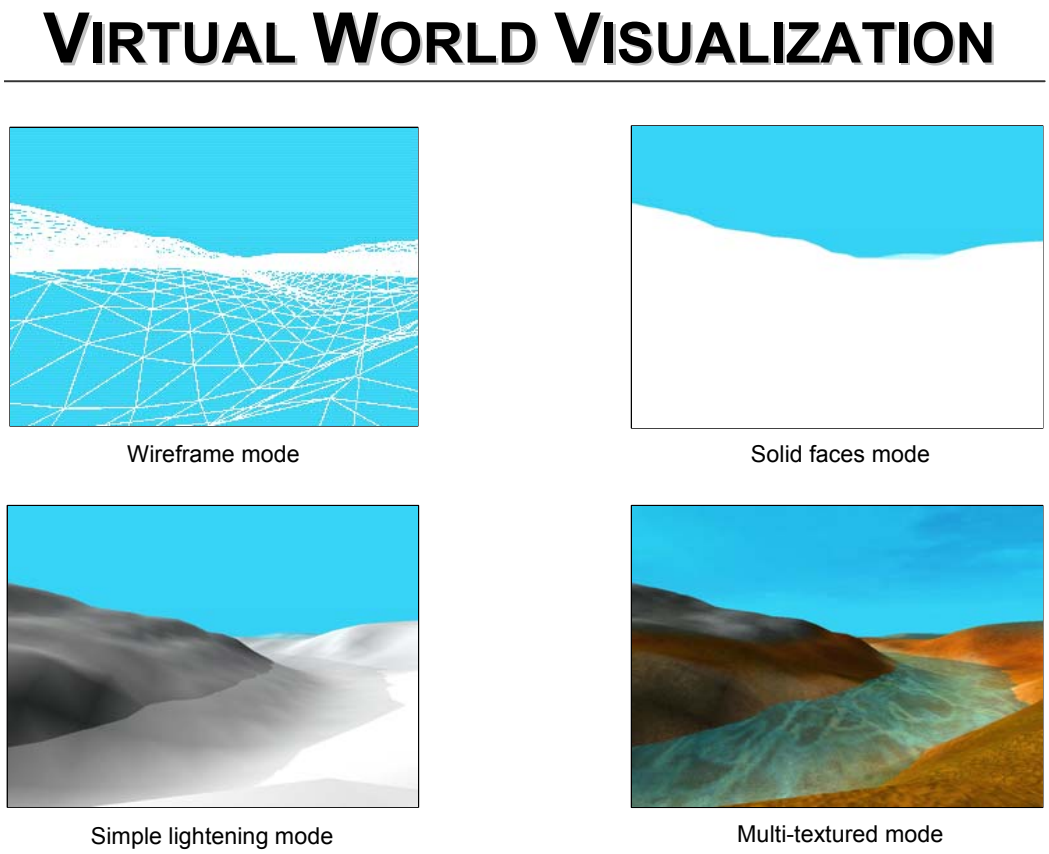
Подсистема расширенной поддержки работы мультиагентных систем предоставляет пользователю возможности по наблюдению за мультиагентными системами, позволяет наблюдать за действиями агентов в мультиагентной среде, следить за изменениями их характеристик в реальном времени.

6.1. Графический модуль и модуль экспорта моделей из 3DStudioMAX

Подсистема позволяет задать агенту трехмерную геометрическую модель, являющуюся его представлением в мультиагентной среде. Эта модель экспортируется из графического пакета 3DStudioMAX. Такой способ позволяет рисовать графические модели для агентов, используя для этого удобные средства, и быстро переносить эти модели в мультиагентную среду.

С геометрическими моделями работает графический модуль (pic 5.2). Этот модуль базируется на кроссплатформенной библиотеке OpenGL и позволяет выводить на экран трехмерное изображение видимой с данной позиции части мультиагентной среды и агентов, находящихся в ней. Благодаря открытой архитектуре графический модуль может быть заменен модулем, использующим Direct3D или другую библиотеку в качестве библиотеки трехмерной визуализации.

Графический модуль позволяет визуализировать трехмерные модели агентов, а также ландшафт, являющийся объектом в мультиагентной системе. При визуализации используются такие технологии, как текстурирование, трехмерное освещение. При рендеринге рельефа местности, заданного как регулярная сетка высот ландшафта, применяется технология уровней детализации, базирующаяся на четверичных деревьях (quad tree). Эта технология позволяет увеличивать размеры треугольников, находящихся на удалении от камеры, для уменьшения их числа и, соответственно, экономии системных ресурсов.



Pic 6.1. Virtual world visualisation

При работе с графическим модулем используются камеры наблюдения, позволяющие пользователю перемещаться по мультиагентной среде и выбирать различные позиции, с которых лучше видны происходящие в мультиагентной системе события. Менеджер камер, одна из частей графического модуля, позволяет работать с камерами, переключаться между ними, программировать траектории их движения, автоматизируя работу по выбору наилучшей позиции.

6.2. Звуковой модуль

Звуковой модуль позволяет создать звуковую картину происходящих в мультиагентной системе событий. Этот модуль базируется на кроссплатформенной библиотеке OpenAL, позволяющей осуществлять работу с трехмерным звуком. Источник звука может быть закреплен за каждым агентом, и при перемещении агента по мультиагентной среде или при изменении пользователем позиции текущей камеры наблюдения характер звука, издаваемого объектом, будет меняться в соответствии с законами распространения звука в пространстве.

Сервер агентов предоставляет своим агентам набор функций, позволяющих осуществлять контроль над их звуковыми представлениями в мультиагентной среде. Агент может постоянно, в течение всего своего жизненного цикла, издавать какой-либо звук, или звук может издаваться как реакция на какое-либо событие, произошедшее в мультиагентной системе.

6.3. Модуль статистики

Модуль статистики, являясь одним из важнейших инструментов при исследовании свойств мультиагентных систем, позволяет пользователю проводить наблюдения за различными параметрами и характеристиками агентов. Этот компонент комплекса предоставляет

пользователю удобный графический интерфейс, позволяющий представить цифровую информацию в читабельном визуальном представлении.

Модуль статистики позволяет собирать значения параметров агента, отображать их с помощью графических индикаторов, строить графики изменения различных параметров, подсчитывать средние значения переменных, давая пользователю возможность проследить динамику изменения состояния агента, и сохранять историю изменения отдельных параметров агента в файл, при этом возможно последующее воспроизведение поведения агента в соответствии с этой информацией.

Графический интерфейс этого модуля является гибким и настраиваемым. Он представляется в виде набора панелей, отображающих различные характеристики агентов. Набор характеристик, за которыми необходимо следить, можно изменять для каждой конкретной задачи.

Также модуль статистики позволяет не только наблюдать за агентами, но и проводить графическое сравнение графиков изменения их различных параметров со временем.

7. Примеры использования комплекса

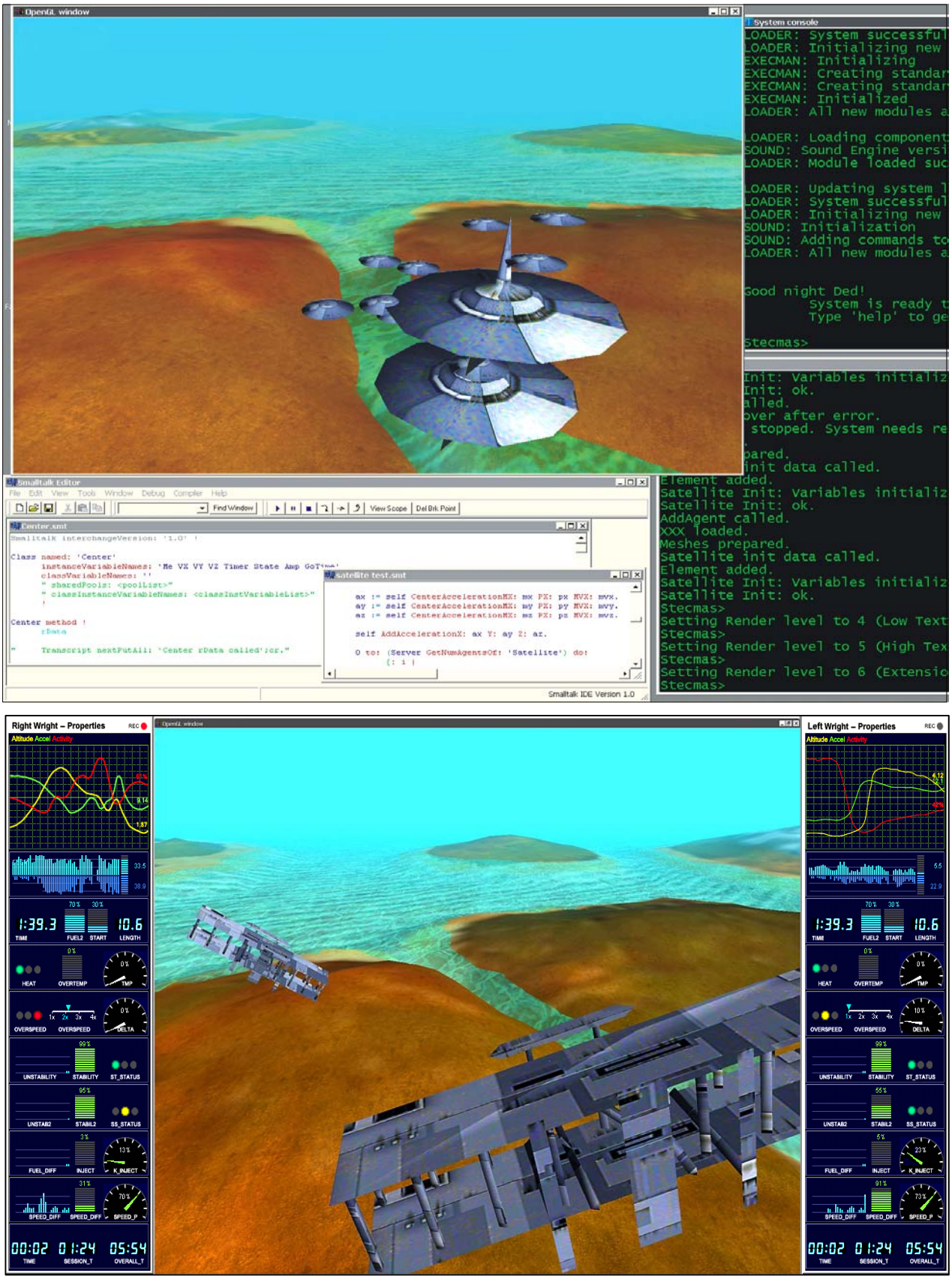
Комплекс может применяться в различных сферах научной и технической деятельности, таких как моделирование и исследование различных процессов, происходящих в природе, от характера движения молекул до поведения и взаимодействия популяций животных.

Комплекс уже получил применение при проведении конкурса боев программ, когда участники должны были за определенное время разработать алгоритм поведения летающих объектов, которые должны были, сталкиваясь с объектами противника, вытолкать их за пределы зону проведения сражений в мультиагентной среде. При проведении конкурса были организованы бои, в каждом из которых участвовали две команды по четыре летающих объекта, и поведение объектов каждой команды было запрограммировано одним из участников. Участники получили возможность наблюдать за этими боями (pic 7.1).

Также было проведено исследование характера поведения летающих объектов - спутников, стремящихся приблизится к центральному агенту, но пытающихся держаться на удалении друг от друга. В результате исследования было выявлено, что спутники пытались выстроиться в правильную стереометрическую фигуру, форма и линейные размеры которой зависели от количества спутников и коэффициентов притяжения к центральному агенту и отталкивания друг от друга.

Комплекс может применяться и для решения других задач, таких как создание компьютерных игр (средства, предоставляемые комплексом, позволяют создать виртуальный мир, в котором происходит действие игры, и описать поведение противников), моделирование динамики развития и взаимодействия популяций различных существ (при этом описывается поведение каждого существа и задаются начальные условия, такие как местность, в которой они живут), моделирование движения молекул в различных средах, например, в экране жидкокристаллического монитора.

SYSTEM AT WORK



Pic 7.1. System at Work

8. Список использованной литературы и сайтов

1. Бьярн Страуструп «Язык программирования C++»
2. Даниэль Грис «Конструирование компиляторов для ЭВМ»
3. Андрей Александреску «Современное проектирование на C++»
4. Герб Саттер «Exceptional C++»
5. Герб Саттер «More Exceptional C++»
6. Кент Бек «Экстремальное программирование»
7. OpenGL Programming Guide, Addison-Wesley Publishing
8. Юрий Тихомиров «OpenGL: программирование трехмерной графики»
9. Джон Бентли «Жемчужины программирования»
10. Дейл Роджерсон «Основы COM, 2-е издание»
11. Дональд Кнут «Искусство программирования». Том 1 «Фундаментальные алгоритмы».
12. Дональд Кнут «Искусство программирования». Том 3 «Сортировка и поиск».
13. Ахо, Теффри, Ульман, Сети «Компиляторы. Принципы, технологии, инструменты»
14. Хендрикс «Small C Compiler»

Материалы сайтов:

15. www.gotw.ca
16. www.cuj.com
17. www.smalltalk.com
18. www.opengl.org
19. www.pmg.narod.ru
20. www.nvidia.com
21. www.reactor.com
22. ANSI Site Smalltalk standard draft 1.9
23. MSDN Programming Reference Jan 2003

SYSTEM ARCHITECTURE