Программа секции Computer Science – 2008 г.

$11^{00} – 13^{00}$

1. **Столяров Леонид (8В). Программно-аппаратный комплекс для проведения лабораторных работ по физике с использованием компьютера.**
2. **Никитенко Евгений ($7^{В}$). Разработка компьютерной игры «Снайпер» с возможностью передачи данных через интернет на языке программирования C++.**
3. **Борисова Татьяна ($10^{В}$). Разработка синтаксически ориентированного фильтра для сообщений об ошибках компиляции шаблонов C++.**
4. **Татаринов Андрей (NVidia). Генерация трехмерных эффектов огня, дыма и других волюметрических объектов с помощью шумов Перлина.**
5. **Янушковский Владимир ($9^{В}$). Перенос рабочего приложения с платформы Win32 на Qt и адаптация его к архитектуре библиотеки с использованием автоматного программирования.**
6. **Шабалов Сергей ($7^{В}$). Разработка компьютерной игры жанра шутера от третьего лица на языке C++ с использованием библиотеки DirectX 9.**

$13^{20} – 15^{20}$

7. **Пономарев Олег ($9^{А}$). Разработка ядра мультиагентной системы.**
8. **Скаков Юрий ($7^{В}$). Применение законов инерции и отражения в разработке программ.**
9. **Гурьянов Федор ($10^{В}$). Алгоритмы простых игр и секреты играбельности.**
10. **Янушковский Владимир ($9^{В}$). Использование компиляции «на лету» для быстрого и эффективного построения графиков функций.**
11. **Полещук Алексей (МГТУ им. Баумана). Система управления мобильным роботом для соревнований Eurobot 2008.**
12. **Зотов Алексей (МГТУ им. Баумана). Технологии командной работы в проектировании мобильных роботов для соревнований Eurobot.**
13. **Морозов Федор ($9^{В}$). Разработка программного комплекса для создания мультиагентных систем, использующих автоматный подход.**

Тезисы секции Computer Science – 2008 г.

1. Столяров Леонид (8В). Программно-аппаратный комплекс для проведения лабораторных работ по физике с использованием компьютера.

Целью работы является создание комплекса, позволяющего вычислить параметры маятника, измерять его параметры, обрабатывать данные, полученные в результате измерений, представлять результаты в виде графиков или осциллограмм. Имеется возможность расширять функциональность системы.

Комплекс состоит из программного обеспечения и аппаратных установок. Программное обеспечение поддерживает работу с различными установками. Одна из них – классическая модель маятника. Аппаратная установка подключается к компьютеру и обменивается данными по порту USB, на компьютере запускается программа с графическим интерфейсом пользователя и прово-

дится эксперимент. Программное обеспечение разработано по модульной архитектуре с использованием библиотек dll. Архитектура модульная и многоуровневая, т.е. существует общий слой библиотек для всех аппаратных установок, и существует слой библиотек для каждой аппаратной установки, содержащий элементы, специфичные для конкретной установки. Модульная архитектура даёт возможность написать собственные модули или изменить существующие для реализации нестандартных задач. Для этого имеется несколько уровней доступа к системе. Человек, хорошо знающий программирование, может написать собственный модуль «с нуля», изучив предоставленные средства. Человек, плохо знающий программирование может изменить или дописать существующие модули. Человек, почти не знающий программирование может изменить загрузчик, используя простейшие приёмы программирования, или написать специальный скрипт. В программное обеспечение входит программный эмулятор установки «маятник», который может работать так же, как аппаратная установка, если её нет. Обмен данными с аппаратной установкой осуществляется через специальный протокол с пакетами данных. Графический интерфейс написан на библиотеке MFC с использованием надстроек, включает в себя осциллограф, работу с файлами.

2. Борисова Татьяна (10^{B}). Разработка синтаксически ориентированного фильтра для сообщений об ошибках компиляции шаблонов C++.

При написании программ на C++ с использованием шаблонов или STL-классов у разработчиков часто возникает проблема, связанная с тем, что компилятор выдает слишком большое сообщение об ошибке, называемое «ошибкой-романом» (размер его может достигать нескольких десятков тысяч символов). Причем часто оно бывает очень запутанным и плохо поддается анализу, в результате чего поиск ошибок компиляции в коде на C++, использующем шаблоны, становится крайне затрудненным. Такое происходит не только при написании сложных систем шаблонных классов, но и в относительно небольших программах. Например, код

```
std::multimap <std::string, int> mm;
mm.insert (0);
```

вызывает появление сообщения об ошибке в Visual Studio 2005, размер которого 21 строка (и это один из самых безобидных примеров).

Целью проекта было облегчение труда разработчиков, представление ошибок-романов в более удобном и читаемом виде.

Задачей являлось написание плагина к популярной среде разработки Visual Studio, который представлял бы вывод компилятора в виде дерева, сравнивал бы синтаксическое дерево сообщения об ошибке с деревом возможно правильного варианта (если такой предоставляется компилятором) с целью поиска конкретного участка кода, где произошло несоответствие, выполнял бы синтаксическую подсветку текста сообщения об ошибке.

Прямых аналогов проекта нет. Существует проект под названием STLFilt, который является представляет собой скрипт, написанный на Perl, форматирующий стандартный вывод компилятора и скрывающий некоторые детали инициализации шаблонных классов. Его функциональности недостаточно, так как он не представляет вывод компилятора в виде дерева (только форматирование отступами), не сравнивает деревья сообщения об ошибке и возможного правильного варианта друг с другом и не выполняет синтаксическую подсветку сообщения об ошибке.

Проект подразделяется на 2 части: плагин к Visual Studio, отображающий в специальном окне (называемом Tool Window) сообщения об ошибках, предоставленные компилятором, в виде дерева, удобном для анализа и поиска ошибки. Вторая часть – это, собственно, анализатор, который находит в выводе компилятора сообщения об ошибке, в них – C++ декларации, которые вызвали ошибку и проводит их синтаксический анализ, получая на выходе синтаксические деревья частей сообщения об ошибке, содержащих C++ код, которые будут отображаться плагином.

Проект написан на языке C++ с использованием шаблонов, технологий объектно-ориентированного и обобщенного программирования, библиотек boost, STL. Также были использованы написанные ранее библиотеки для лексического анализа и ведения журнала программных ошибок (логгинга). Синтаксический разбор был реализован с помощью рекурсивного спуска.

В ближайшем будущем планируется завершение продукта до стадии непосредственного использования конечными пользователями и пробное внедрение его среди школьников в кабинете информатики. В дальнейшем планируется перенесение плагина на платформу Visual Studio 2008, поддержка других существующих компиляторов (например, gcc), написание плагинов для других сред разработки (Eclipse, Dev-CPP).

3. Морозов Федор (9^B). Разработка программного комплекса для создания мультиагентных систем, использующих автоматный подход.

Мультиагентная система (МАС) – система, образованная несколькими взаимодействующими агентами –сущностями с программируемым поведением, «наблюдающими» за окружающей средой и действующими в ней для достижения какой-либо цели.

Цель работы – создание программного комплекса для создания и изучения МАС без лишних трудозатрат.

Для описания поведения агентов используется текстовый язык автоматного программирования, для него разработан транслятор и виртуальная машина. Данный язык был выбран из-за того, что автомат (множество состояний, а также переходы из одного состояния в другое) является естественной формой описания сущностей. Синтаксический анализатор языка реализован на основе алгоритма рекурсивного спуска, лексический представляет собой конечный автомат.

Для взаимодействия с пользователем используется консоль, принимающая команды и транслирующая их в сообщения для других компонентов системы, для рисования агентов – визуализатор, реализованный с помощью своих и сторонних библиотек. Агенты взаимодействуют по некоторым законам, определенным соответствующими частями системы.

Также в систему включен набор инструментов для отладки и контроля ее работы, к примеру, протоколирование переходов между состояниями автомата поведения каждого агента.

4. Пономарев Олег (9^A). Разработка ядра мультиагентной системы.

Ядро предназначено для создания мультиагентных систем, состоящих из модулей и объектов. Главное свойство модулей – способность изменять свойства объектов системы. Таким образом, объект системы представляет из себя набор свойств, каждое из которых присоединяется к объекту и используется одним или несколькими модулями (например, при построении физической модели такие свойства, как координата и скорость используются модулями физики и графики, а такое свойство, как графическая модель объекта используется исключительно визуализатором (графическим модулем)). Модули могут взаимодействовать друг с другом (например, практически любому модулю приходится взаимодействовать с пользователем при помощи консоли). В ядро системы входят системный загрузчик, модуль «мессенджер», системная база данных, а также система отладки.

Основной задачей системного загрузчика является загрузка и выгрузка модулей системы, скомпилированных в библиотеки динамической линковки (DLL). Поддерживается динамическая загрузка и выгрузка модулей. Также этот модуль отвечает за размещение и хранение модулей в памяти. Системный загрузчик был разработан совместно с Морозовым Федором.

Модуль предоставляет возможность взаимодействия между модулями с помощью сообщений. Одним из основных требований к этому модулю было освобождение модулей системы от необходимости создавать функцию, содержащую цикл обработки сообщений. Для этого была реализована система подписки конкретных функций модулей (их слотов и запросов) на определённые типы сообщений. Каждая функция вызывается при поступлении соответствующего сообщения, а потоке обработки сообщений, который мессенджер создаёт для каждого модуля. При этом гарантируется, что одним модулем единовременно обрабатывается лишь одно сообщение, а это избавляет от необходимости делать модуль с расчётом на многопоточное использование.

Модуль базы данных предназначен для хранения объектов и их свойств, а также предоставление интерфейса (т.е. слотов и запросов) для добавления, изменения, чтения и удаления данных и объек-

тов. Любой объект системы сохраняется в базе данных, а обращение к нему и изменение его свойств осуществляется через интерфейс базы данных. Также стоит отметить, что для загрузчика, база данных – это самый обыкновенный модуль системы, то есть правила её загрузки и выгрузки ничем не отличаются от соответствующих правил для любого другого модуля.

Система отладки, в отличие от всех ранее описанных, является набором библиотек для статической линковки с модулями. Для использования системы логов достаточно лишь добавлять по одному её макросу в каждую функцию модуля, выполнение которой мы хотим отразить в файл системного журнала (файл логов). Также существует возможность добавлять в файл логов какие-либо сведения о работе программы. Файл логов, из которого можно узнать о работе системы (например, существует возможность посмотреть порядок вызовов всех функций модуля), генерируется после завершения работы системы. Модуль может быть скомпилирован с системой отладки в одном из трёх режимов (module debug, system debug или release).

5. Скаков Юрий (7^{B}). Применение законов инерции и отражения в разработке программ.

В докладе рассматривается создание двух проектов на языке C++ для операционной системы Windows.

Первая программа имитирует стандартную заставку Windows, рисующую два четырехугольника, каждая вершина которых по-своему отражается от границ окна и после этого продолжает движение. Четырехугольник на каждом шаге программы строится заново, с учетом текущих координат его вершин. После отражения вершины от «стенки» ее скорость меняется. В программе используется закон равенства угла падения углу отражения: с какой скоростью вершина ударяется о «стенку», с такой скоростью, только в другом направлении, она отражается от нее. В ходе работы над проектом добавлена возможность изменения количества углов самим пользователем.

Вторая программа – игра, содержащая возможность управления ускорением объекта с помощью клавиш со стрелками. Программа усовершенствована добавлением ограничений поля рисования объекта. В случае выхода объекта за границы поля выдается сообщение о вашем проигрыше. Закон инерции применяется во время постепенного уменьшения скорости, в результате в программе нет возможности резкой смены направления, чем игра и отличается от обычных программ с перемещением объектов.

В дальнейшем планируется возможности рисования нескольких многоугольников одновременно для первой программы и создание уровней для второй программы.

6. Никитенко Евгений (7^{B}). Создание компьютерной игры «Снайпер» с возможностью передачи данных через интернет на языке программирования C++.

Цель работы – создание графической компьютерной игры «Снайпер» на языке программирования C++ с помощью компилятора gcc в среде OS Windows. Цель игры – сбить все мишени за минимальное количество выстрелов. Количество выстрелов ограничено. Один из плюсов – пользователю можно создавать свои уровни с расширением .lvl. При разработке преследовалась цель максимально упростить их создание. Для отображения интерфейса используется графика в формате BMP, что придает игре красочность. Разработка игры помогла автору лучше понять язык C++.

Программа передает уровни, созданные пользователем, на FTP-сервер, чтобы автор мог добавлять их в игру. Для того, чтобы минимизировать задержки при передаче, создается второй поток, который запускает программу обмена данных с сервером. Таким образом, это не влияет на скорость работы игры.

7. Янушковский Владимир (9^{B}). Перенос рабочего приложения с платформы Win32 на Qt и адаптация его к архитектуре библиотеки с использованием автоматного программирования.

Платформы Win32 и Qt представляют собой две сильно отличающиеся архитектуры программирования. Win32 API – это интерфейс, основанный на структурах, функциях – то есть средствах языка C. Qt же построена на принципах объектно-ориентированного программирования, исполь-

зует иерархию классов, а также собственный оригинальный механизм сигналов и слотов. Но так как Qt предоставляет приложению возможность кроссплатформенности, может возникнуть потребность перенести Win32 приложение на неё, а, учитывая сильные архитектурные различия технологий, могут возникнуть проблемы. В докладе рассматриваются основные различия платформ и вопросы, которые могут возникнуть при таком переносе. Рассматриваются изменения в архитектуре приложения, которые следует произвести. Приводится пример рабочего приложения MathTools, которое переносится на платформу Qt.

8. Янушковский Владимир (9^{B}). Использование компиляции «на лету» для быстрого и эффективного построения графиков функций.

При построении графика требуется произвести множество однотипных вычислений (просчитать значение функции для каждой точки). Но если функция задана введённым пользователем выражением, то время расчётов может существенно увеличиться, так как в этом случае для подсчёта значения функции нужно не просто вызвать простую функцию, а разобрать строку с формулой. В докладе рассматривается возможность компиляции выражения в машинный код функции, которое сводит обсчёт значения в точке к простому вызову функции, что позволяет многократно увеличить скорость расчётов. В докладе приводится пример – приложение MathTools, в котором продемонстрированы разные способы вычислений (компиляция, компиляция во временный код, разбор строки в каждой точке), и преимущество первого становится очевидным.

9. Зотов Алексей, МГТУ им. Баумана. Технологии командной работы в проектировании мобильных роботов для соревнований Eurobot.

Если вам приходилось участвовать в серьёзных проектах по разработке, вы практически наверняка знакомы с мудростью поговорки «Первый блин комом». Только закончив работу, вы обладаете полноценным знанием о том, как её следовало делать. Очень редко удаётся сделать шаг назад и выполнить работу заново, но приятно было бы иметь такую возможность.

В проектах по разработке программного обеспечения особенно силён миф о том, что планировать их невозможно и даже вредно. Из-за такого подхода страдают разработчики, руководители и заказчик. Как же спланировать свою работу и работу окружающих вас людей, чтобы их и ваше время не тратилось впустую. Как увидеть текущее состояние проекта и понять, что уже сделано и что ещё предстоит.

В своём докладе я попытаюсь изложить несколько базовых принципов планирования и управления проектом проверенных на своём опыте.

10. Полещук Алексей, МГТУ им. Баумана. Система управления мобильным роботом для соревнований Eurobot 2008.

Мобильный робот модели JetBox предназначен для участия в соревнованиях автономных мобильных роботов по регламенту Eurobot 2008.

Робот обладает развитой системой управления, позволяющей ему планировать свои действия в соответствии с изменениями ситуации на игровом поле, корректировать свое местоположение относительно маркеров и осуществлять поиск игровых элементов.

Система управления имеет многоуровневую архитектуру, включающую в себя: контроллеры приводов колес, центральный контроллер, процессоры обрабатывающие сенсорную информацию. Соединения между модулями осуществляются с помощью интерфейсов CAN и SCI.

Контроллеры приводов осуществляют стабилизацию скорости каждого колеса в соответствии с поступающими командами. Центральный контроллер осуществляет планирование траектории движения робота и ее отслеживание, а также управляет прочими действиями в соответствии с изначально заданной программой и данными поступающими от сенсорной системы и приводов колес.

Стабилизация скорости каждого колеса реализуется с помощью нестационарной следящей системы с обратной связью по скорости. Следящая система включается в себя фильтр для обратной связи по скорости.

Траектория движения робота по полигону представляет собой ломаную линию, состоящую из отрезков прямых. Таким образом, процесс движения является последовательностью движений по прямым и разворотов. В процессе движения по прямой осуществляет стабилизация робота по линейному и угловому отклонениям, а также угловой скорости.

Исполняемая роботом последовательность действий является программой специального вида для виртуальной машины, исполняющейся на центральном контроллере. Использование такой архитектуры позволяет осуществить гибкое планирование операций с учетом меняющейся игровой ситуации. В частности такая архитектура позволяет надежно осуществлять коррекцию координат робота при стыковке к контейнеру с образцами, когда присутствует накопленная позиционная ошибка в процессе навигации по игровому полигону.

11. Гурьянов Федор (10^{B}). Алгоритмы простых игр и секреты играбельности.

Цель проекта – рассмотрение алгоритмов простых игр, не требующих изощренного программирования, но при этом достаточно увлекательных.

Рассматриваются реализация таких игр и идеи их создания.

Среди представленных игр – «Опасная дорога», «Арканоид», «Собери картинку», «CDDestroy», «BallShooter» и другие.

12. Шабалов Сергей (7^{B}). Разработка компьютерной игры жанра шутера от 3-его лица на языке C++ с использованием библиотеки DirectX 9.

Целью работы являлось создание компьютерной игры жанра шутера от 3-его лица на языке C++ с использованием библиотеки DirectX 9.

Выбор библиотеки DirectX связан с тем, что данная библиотека чаще всего используется для создания компьютерных игр такого рода.

Задачами работы являлось изучение библиотеки DirectX 9, разработка архитектуры движка, изучение особенностей жанра шутера от 3-его лица, разработка форматов игровых файлов, решение основных проблем, связанных с разработкой интерфейса и оптимизации движка с целью увеличения скорости рендеринга.

В докладе будут рассмотрена архитектура библиотеки DirectX, архитектура движка игры и схема наследования классов игровых объектов, разработаны форматы игровых файлов, решена проблема с выбором пункта меню и оптимизирована функция рендеринга.

Итогом данной работы является создание полноценной компьютерной игры, использующей DirectX – одну из популярнейших графических библиотек.

В дальнейшем планируется добавление возможности игры по сети с использованием DirectPlay и разработка редактора карт.