|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Информационная безопасность

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К курсовому проекту***

***НА ТЕМУ:***

***Система моделирования дорожного движения***

Студент ИУ8-31 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Северов В.С

(Подпись, дата)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Бородин А.А

(Подпись, дата)

*2017 г.*

**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУ8

Басараб М. А.

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине Технологии и методы программирования

***Студент группы ИУ8-31:***

Северов Владимир Сергеевич

***Тема курсового проекта:***

Разработка системы моделирования дорожного движения.

***Направленность курсового проекта:***

учебная

***Источник тематики:***

кафедра

***График выполнения курсового проекта:***

25% к 3 нед., 50% к 9 нед., 75% к 12 нед., 100% к 15 нед.

***Техническое задание:***

Создать модель движения транспорта по дорогам общего пользования, которая позволяет создавать, менять карты городов и изменять дорожную ситуацию в режиме реального времени. Также она должна иметь гибкую архитектуру для создания новых объектов (пешеходные переходы, ж/д переезды, дорожные работы, высокоскоростные магистрали) и иметь высокую производительность при большом количестве транспортных средств.

***Оформление курсового проекта:***

Расчетно-пояснительная записка на \_\_\_\_\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « 12 » сентября 2017 г.

**Руководитель курсового проекта**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Бородин А.А

(Подпись, дата)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Северов В.С

(Подпись, дата)

Оглавление

[Цель 4](#_Toc501396034)

[Основные определения 5](#_Toc501396035)

[Введение 6](#_Toc501396036)

[Требования к проекту 7](#_Toc501396037)

[Проектирование системы 8](#_Toc501396038)

[Система управления 8](#_Toc501396039)

[Класс «WorldScene» 8](#_Toc501396040)

[Класс «WorldController» 9](#_Toc501396041)

[Объекты карты 9](#_Toc501396042)

[Сегменты поверхности карты 9](#_Toc501396043)

[Статические объекты 11](#_Toc501396044)

[Динамические объекты 12](#_Toc501396045)

[Выбор технологий 14](#_Toc501396046)

[Выбор языка программирования 14](#_Toc501396047)

[Выбор используемых библиотек 16](#_Toc501396048)

[Описание технический решений 17](#_Toc501396049)

[Заключение 18](#_Toc501396050)

[Список используемых источников 19](#_Toc501396051)

# Цель

Цель проекта заключается в создании модели движения транспорта по дорогам общего пользования для тестирования системы управления светофорами и машинами, проверки качества дорожных развязок и тому подобное.

# Основные определения

* RTM – Road Traffic Model, название проекта.
* Cocos2d-x – открытая кроссплатформенная библиотека для разработки игр.
* ПО – программное обеспечение.
* IDE – Integrated Development Environment, среда разработки ПО.
* Дорожная развязка – комплекс дорожных сооружений (мостов, туннелей, дорог), предназначенный для минимизации пересечений транспортных потоков и, как следствие, для увеличения пропускной способности дорог.
* Транспортный поток – это упорядоченное транспортной сетью движение транспортных средств.

# Введение

Проект решает несколько важных проблем, причем как настоящие, так и будущие.

Рассмотрим настоящие проблемы. В данный момент по всему миру крупные города страдают от большого количества дорожных заторов. Но что нужно для решения данной проблемы? Во-первых, необходимо правильно рассчитать время работы каждого сигнала светофора или же создать искусственный интеллект для автоматических расчётов, зависящих от дорожной ситуации. Во-вторых, необходимо создавать качественные дорожные развязки, которые также помогут улучшать дорожную ситуацию. Но как же данный проект поможет решать эти проблемы? Ответ на этот вопрос достаточно прост. Данное приложение позволит создавать модели различных дорожных развязок, тестировать системы управления светофорами и воссоздавать потоки машин такие же, как и в реальности!

Также, данный проект поможет решать не только настоящие проблемы, но и будущие. Например, существуют проблемы по написанию искусственного интеллекта для автомобилей. В настоящий момент уже создано программное обеспечение, позволяющее машине ехать в своей полосе, тормозить при появлении препятствий, парковаться параллельным способом. Но ведь это далеко не предел! В скором будущем достаточно будет указать начальную и конечную точки маршрута, а всё остальной автомобиль сделает сам. Он будет перестраиваться, ждать зеленого сигнала светофора и делать многое другое абсолютно самостоятельно. А тестировать новые версии прошивок можно будет в нашей программе без особых проблем!

# Требования к проекту

* Простое создание новых карт городов и изменение готовых
* Простое изменение дорожной ситуации (увеличение количества машин, создание барьеров на дороге)
* Гибкая архитектура для создания новых объектов (пешеходные переходы, ж/д переезды, дорожные работы, высокоскоростные магистрали)
* Высокая производительность при большом количестве транспортных средств (95% загруженности дороги)

# Проектирование системы

Для того чтобы реализовать данный проект, было решено разделить его на две основные части: система управления и объекты карты.

## Система управления

В данном случае системой управления является набор классов, который выполняет некоторые автоматические вычисления (например, обработку столкновений для более быстрых расчетов), а также является связующим элементом между пользователем и программой. В свою очередь система управления состоит из двух частей: так называемая сцена («WorldScene») и контроллер, управляющий миром, в котором всё происходит («WorldController»).

### Класс «WorldScene»

Класс сцены является наследником готового класса "Scene" из библиотеки «Cocos2d». Он содержит в себе различные обработчики событий, а также интерфейс для работы с графической составляющей приложения. Благодаря ему возможно с помощью клавиатуры загружать различные карты, создавать новые объекты, менять масштаб отображения и скорость воспроизведения. Также класс сцены содержит в себе различные слои для отображения объектов разных типов. Например, есть отдельный слой для отображения фона, поскольку он никак не взаимодействует с другими объектами. Сами же слои являются объектами готового класса «Layer» из библиотеки «Cocos2d». Поэтому они содержат различные наследуемые методы: для изменения масштабов, для отображения изображений объектов, для перемещения самого слоя относительно родительского окна и так далее.

### Класс «WorldController»

Класс контроллера – связующее звено между сценой и всеми объектами, а также является сердцем всей программы. С его помощью можно загружать различные карты, добавлять различные объекты в мир и многое другое. Тут же храниться функция для обновления всей карты. Этот метод проверяет, не пора ли сгенерировать новые объекты, не вышел ли кто за пределы карты, не случилась ли где-то авария и не пора ли её устранить. Также тут происходит вызов функций обновления у всех изменяющихся объектов. То есть всем объектам сообщается, что прошло определенное количество времени и пора совершить какое-то действие, описанное в программном коде обновляемого объекта.

## Объекты карты

Существует три основных типа объектов: сегменты поверхности, статические и динамические объекты. Но обо всем по порядку.

### Сегменты поверхности карты

Для реализации данного типа было создано 2 класса: «CoatingObject» и «CoatingUnion».

Первый из этих классов описывает параметры поверхности, по которой движутся различные объекты, причем значения задаются не на всю область, на определенную секцию карты. Благодаря этому движущиеся объекты всегда могут понять, насколько скользкое покрытие в том или ином месте, в каких направлениях можно проехать, а в каких лучше не стоит. Таким образом, можно задавать направление движения автомобилей, однако последнее слово всегда остается за «водителем». Для того чтобы описать новый тип поверхности, надо создать наследуемый класс, описав необходимые параметры в конструкторе. Но если заранее это сделать невозможно, и решения принимаются «на ходу», то достаточно будет вызвать конструктор класса «CoatingObject» с необходимыми параметрами. Иерархия, реализованная в данной программе, показана на рисунке 1.

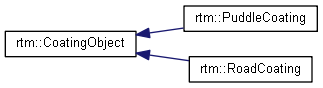


Рисунок 1. Наследуемые классы от класса «CoatingObject»

Второй из перечисленных классов является надстройкой над первым классом. Он позволяет объединять несколько сегментов поверхности в общую структуру, описывая общие параметры. Попутно в конструкторе можно переопределить разрешенные направления движения у каждого объекта класса «CoatingObject», тем самым задавая направление движения для движущихся объектов. Это позволяет заранее «собрать по кусочкам» ту или иную структуру и легко использовать её в любой момент выполнения программы. В данной программе было реализовано несколько стандартных объединений дорог, таких как: односторонняя многополосная дорога, многополосный поворот и многополосные перекрестки. Данную иерархию можно наблюдать на рисунке 2.

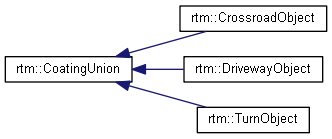


Рисунок 2. Наследуемые классы от класса «CoatingUnion»

Односторонняя многополосная дорога имеет необходимые методы для получения информации о полосах движения. То есть их количество, номер полосы, находящихся по координатам (полезно для ориентирования машин) и так далее. Многополосный поворот в свою очередь имеет всего одной свойство, в какую сторону продолжается движение. Однако перекрестки устроены сложнее. Во-первых, тут содержится для различных конструктора для двух типов перекрестка: обычного и Т-образного. Во-вторых, тут храниться направление, в котором нельзя двигаться (в случае Т-образного перекрестка), которое помогает движущимся объектам понять, необходимо ли поворачивать или можно продолжить движение прямо. В-третьих, перекрестки могут содержать управляющий блок (светофор), который будет показывать, в каких направления разрешено ехать в данный момент. Данное размещение управляющего блока необходимо для грамотной регулировки и корректного отображения.

Рассмотрим «светофоры» поподробнее. Данный класс не очень сложен, ведь в нем содержаться всего две основные функции, зависящие от выбранного способа управления: инициализатор и метод для обновления. Инициализатор загружает необходимые текстуры и задает первоначальные значения сигналов. Метод для обновления в свою очередь содержит различные инструкции (зависящие от выбранного способа управления) для изменения сигналов.

### Статические объекты

Особенность объектов данного типа в том, что они не меняются со временем. То есть они не движутся и не имеют никакой анимации. Также объекты данного типа осязаемы, то есть движущийся объекты могут столкнуться с ними. В данном проекте был создан класс «StaticObject», имеющий свойства, описанные выше. У него есть один дочерний класс «MapObject», являющийся надстройкой. Его отличие от «StaticObject» в том, что координаты указываются не в пикселях, а в количестве секций карты, что упрощает работу. Далее от класса «MapObject» наследуются различные типы стандартных объектов, которые можно быстро отобразить на карте. В данном проекте было реализовано два стандартных класса: «BuildingObject» и «BushObject». Первый из этих классов является классом объектов в виде зданий. Второй же является классом объектов в виде кустарниковых растений. Де-факто, оба класса несут в себе лишь декоративную нагрузку. Иерархию статических объектов можно увидеть на рисунке 3.

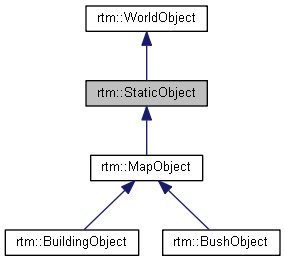


Рисунок 3. Наследуемые классы от класса «StaticObject»

### Динамические объекты

Особенность объектов данного типа в том, что они меняются со временем. Также объекты данного типа, как и статические объекты, осязаемы, то есть могут сталкиваться с другими объектами. В данном проекте был создан класс «DynamicObject», имеющий свойства, описанные выше. У него есть один дочерний класс «VehicleObject», описывающий некоторые общие свойства транспортных средств. Для примера, в нем содержаться стандартные функции для движения с ускорением, торможения за определенное расстояние, поворота и перестроения из полосы в полосу. Для выполнения этих маневров достаточно указать начальные параметры, а остальное всё сделается автоматически. Также в данном классе есть методы для проверки наличия объектов в стандартных зонах видимости. Однако если они не устраивают, можно подставить в обобщенный метод свои параметры и проверить совершенно любую область. В свою очередь от класса «VehicleObject» наследуются дочерние классы, которые задают поведение транспортного средства. Для установки свой логики выполнения того или иного маневра, достаточно переопределить нужный метод. Таким же несложным способом можно добавить новые маневры. В данном проекте был создан один вид транспорта, автомобиль, в котором была определена ничем не выделяющаяся модель поведения. Иерархию динамических объектов можно увидеть на рисунке 4.

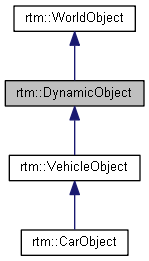


Рисунок 4. Наследуемые классы от класса «DynamicObject»

# Выбор технологий

## Выбор языка программирования

Рассмотрим язык программирования C++. У него много плюсов:

* Поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, ООП, обобщённое программирование и метапрограммирование;
* Предсказуемое выполнение программ. Весь код, неявно генерируемый компилятором, определён в стандарте. Также строго определены места программы, в которых этот код выполняется;
* Автоматический вызов деструкторов объектов при их уничтожении, причём в порядке, обратном вызову конструкторов;
* Используя шаблоны, возможно создавать обобщённые контейнеры и алгоритмы для разных типов данных, а также специализировать и вычислять на этапе компиляции;
* Кроссплатформенность: стандарт языка накладывает минимальные требования на ЭВМ для запуска скомпилированных программ;
* Эффективность. Язык спроектирован так, чтобы дать программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы. Ни одна из языковых возможностей, приводящая к дополнительным накладным расходам, не является обязательной для использования — при необходимости язык позволяет обеспечить максимальную эффективность программы;
* Имеется возможность работы на низком уровне с памятью, адресами.

Но также есть и минусы:

* Синтаксис, провоцирующий ошибки;
* Некоторые преобразования типов неинтуитивны;
* Иногда шаблоны приводят к порождению кода очень большого объёма;
* Отсутствие встроенной системы сборки мусора.

Как альтернативный язык программирования рассмотрим Java. У него также есть некоторые плюсы:

* Кроссплатформенность. Для запуска Java-приложения достаточно установить виртуальную Java-машину (JVM);
* Многофункциональные стандартные библиотеки;
* Встроенный сборщик мусора;
* Встроенные библиотеки для работы с сетью.

Из недостатков можно назвать:

* Низкое, в сравнении с другими языками, быстродействие, повышенные требования к объему оперативной памяти;
* Постоянное развитие языка вызывает наличие как устаревших, так и новых средств, имеющих одно и то же функциональное назначение;
* Java является полностью объектно-ориентированным языком. Например, C++ тоже является объектно-ориентированным, но в нем есть возможность писать программы не в объектно-ориентированном стиле, а в Java так нельзя;
* Язык реализован с использованием интерпретации байт-кода. То есть программа сначала транслируется в машинно-независимый Р-код, а потом интерпретируется некоторой программой-интерпретатором (виртуальная Java-машина, JVM).

Также можно рассмотреть язык программирования Python. Его плюсы:

* Python – универсальный язык, простой и интуитивно-понятный. Кривая обучения у него сравнительно пологая, на этом языке возможно быстрее писать программы;
* В Python встроен фреймворк для тестирования, входной барьер у которого очень низок. Фреймворк обеспечивает хорошее тестовое покрытие;
* Python – простой и распространенный язык, не только понятный многим программистам, но и легкий для специалистов по статистике, на нем можно написать такой инструмент, в котором будут интегрированы все этапы рабочего процесса.

Его минусы:

* Хотя в Python и есть «приятные» библиотеки, визуализация тут устроена гораздо сложнее, а ее результаты порой не слишком наглядны;
* Для выполнения скриптов требуется много ресурсов;
* Отступы могут производиться как за счёт пробелов, так и за счёт табуляции. При совместной разработке это вызывает проблемы.

Подведем итоги. Для нашего проекта лучше всего использовать язык программирования C++, так как этот язык является достаточно гибким, на нём можно написать оптимальную программу и качественно её визуализировать.

## Выбор используемых библиотек

Для данного проекта было решено использовать всего две библиотеки: «FastTrigo» и «Cocos2d-x». Первая библиотека является более быстрой реализацией стандартных тригонометрических функций и используется для ускорения вычислений. Вторая – открытой кроссплатформенной библиотекой для разработки игр. Она позволяет просто и быстро отображать и перемещать изображения в окне программы. Также данная библиотека имеет методы для взаимодействий объектов между собой. Поэтому «Cocos2d-x» может помочь упростить написание данного проекта.

# Описание технический решений

Во время выполнения данного проекта возникло несколько проблем, однако стоит рассмотреть только три из них.

Первая проблема связана с тем, что изначально проект рассчитывался на два человека. Соответственно и объем работ был рассчитан на два человека. Но вследствие некоторых обстоятельств сначала напарник не мог писать код, а потом ему пришлось и вовсе покинуть данный проект. Но так как работа уже была начата, было решено оставить старую структуру и продолжить разработку более усердно.

Вторая проблема возникла во время выбора игрового движка. Сначала была выбрана утилита для разработки игр с трёхмерной графикой. Однако в данном движке было очень сложно сделать одновременную поддержку столкновений и своей физики. И так как это заняло бы много времени (больше, чем было дано преподавателем), было решено использовать другую утилиту.

Третья проблема заключалась в том, что точности вещественных чисел не хватало для точных перемещений объектов, так что в итоге объекты могли пропускать нужные координаты, на которых необходимо принять то или иное решение. Решить это проблему получилось достаточно просто: была написана функция для сравнения координат с некоторым допуском. Тем самым точности вещественных чисел стало хватать.

# Заключение

В итоге, несмотря на возникшие проблемы, удалось реализовать успешный проект, выполняющий необходимые функции: есть дороги, повороты и перекрестки, есть автомобили, которые едут по дорожным полосам, есть светофоры, которые регулируют перекрестки, есть приоритеты движения на нерегулируемых перекрестках. Данный набор методов достаточен для выполнения задач, как регулировка светофоров и проверка пропускной способности той или иной системы дорог.

В дальнейшем планируется сделать удобный редактор карт, чтобы любой человек мог создать модель, соответствующую реальности. Также планируется разработать новые типы объектов, например, трамвайные переезды, перекрестки с круговым движением (однако сейчас их уже можно создать из нескольких объектов), мосты и многоуровневые развязки. Также планируется повысить точность вычислений при улучшении быстродействия программы.

# Список используемых источников

1. Открытый исходный код утилиты «Cocos2d-x». Доступно на: <https://github.com/cocos2d/cocos2d-x> (дата обращения: 15.12.2017).
2. Документация к утилите «Cocos2d». Доступно на: [http://www.cocos2d‑x.org/docs/cocos2d-x/en/index.html](http://www.cocos2dx.org/docs/cocos2d-x/en/index.html) (дата обращения: 15.12.2017).
3. Плюсы и минусы языка программирования C++. Доступно на: <http://cpplus.my1.ru/publ/o_jazyke_c/dostoinstva_i_nedostatki_jazyka/1-1-0-6> (дата обращения: 15.12.2017).
4. Сравнение языков программирования R и Python. Доступно на: <https://habrahabr.ru/company/piter/blog/263457/> (дата обращения: 15.12.2017).
5. Недостатки Python и их описание. Доступно на: <http://python-3.ru/page/nedostatki-python-i-ih-opisanie> (дата обращения: 15.12.2017).