МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

02.03.03 МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ И АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

БАКАЛАВРСКАЯ РАБОТА

на тему **Трехмерная обучающая игра для школьников «Правила поведения на дорогах»**

Студент Д.Е. Савкин _____

Руководитель О.М. Гущина _____

Допустить к защите
Заведующий кафедрой к.тех.н, доцент, А.В. Очеповский _____

«_____»_____20_____г.

Тольятти 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав. кафедрой «Прикладная
математика и информатика»
_____ А.В. Очеповский

«____»_____2016 г.

ЗАДАНИЕ
на выполнение бакалаврской работы

Студент Савкин Дмитрий Евгеньевич

1. Тема: Трехмерная обучающая игра для школьников «Правила поведения на дорогах»
2. Срок сдачи студентом законченной выпускной квалификационной работы 19.06.2016
3. Исходные данные к выпускной квалификационной работе: правила дорожного движения; требования к функциональным характеристикам: обеспечение дифференцированного подхода к обучению, использование трехмерных моделей; наличие интуитивно-понятного интерфейса; многопользовательский режим работы; при добавление новых функций наличие возможности моделирования приложения; архитектура приложения: клиент-серверная с использованием web-технологий; требования к средству реализации базы данных: MySQL.
4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов): проанализировать научную и учебную литературу по проблеме реализации обучающих приложений; рассмотреть основные принципы разработки обучающего игрового приложения; проанализировать подобные игровые приложения для определения основных функций и структурных компонентов моделируемого приложения; создать математическую модель реализации функций программного приложения; создать структуру приложения и описать алгоритмы работы основных его модулей; обосновать выбор средств реализации обучающего приложения; реализовать обучающее

игровое приложение программными средствами; протестировать реализованное программное приложение; апробировать реализованное приложение

5. Ориентировочный перечень графического и иллюстративного материала: блок-схемы работы приложения, графики, диаграммы, экранные формы, демонстрирующие работоспособность программного продукта; презентация

6. Дата выдачи задания « 11 » января 2016 г.

Заказчик,
директор МБУ СОШ №45 _____ Е.Н. Ошкина

Руководитель выпускной
квалификационной работы _____ О.М. Гущина

Задание принял к исполнению _____ Д.Е. Савкин

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав. кафедрой «Прикладная математика и информатика»
_____ А.В. Очеповский

«____»_____2016 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения бакалаврской работы

Студента Савкина Дмитрия Евгеньевича

по теме Трехмерная обучающая игра для школьников «Правила поведения на дорогах»

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
Выбор и утверждение темы ВКР	14.01.2016	14.01.2016	Выполнено	
Поиск и анализ литературы по проблеме реализации обучающих игровых приложений	15.01.2016	15.01.2016	Выполнено	
Описание принципов разработки обучающего игрового приложения	20.01.2016	20.01.2016	Выполнено	
Анализ подобных приложений для определения основных функций и структурных компонентов	25.01.2016	25.01.2016	Выполнено	

Описание математической модели реализуемого программного приложения	03.02.2016	03.02.2016	Выполнено	
Моделирование структуры приложения	10.02.2016	10.02.2016	Выполнено	
Обоснование выбора средств реализации обучающего приложения	15.02.2016	15.02.2016	Выполнено	
Описание алгоритмов работы основных модулей приложения	24.02.2016	24.02.2016	Выполнено	
Реализация обучающего приложения программными средствами	03.03.2016	03.03.2016	Выполнено	
Описание контрольного примера работы разработанного игрового приложения	18.03.2016	18.03.2016	Выполнено	
Тестирование реализованного программного приложения	27.03.2016	27.03.2016	Выполнено	
Апробация игрового приложения	02.04.2016	02.04.2016	Выполнено	
Оформление пояснительной записки ВКР	07.04.2016	07.04.2016	Выполнено	
Подготовка доклада и графического материала к защите	14.04.2016	14.04.2016	Выполнено	
Предварительная защита ВКР	20.05.2016	20.05.2016	Выполнено	
Корректировка ВКР согласно сделанным замечаниям	23.05.2016	23.05.2016	Выполнено	
Проверка ВКР в системе «Антиплагиат.ВУЗ»	28.05.2016	28.05.2016	Выполнено	
Сдача пояснительной записки ВКР и реализованного программного приложения	19.06.2016	19.06.2016	Выполнено	

Руководитель выпускной квалификационной работы _____ О.М. Гущина

Задание принял к исполнению _____ Д.Е. Савкин

Аннотация

Тема выпускной квалификационной работы: «Трехмерная обучающая игра для школьников «Правила поведения на дорогах»».

Объектом исследования бакалаврской работы является процесс обучения пользователя.

Предмет исследования - является процесс разработки обучающих игр. Данное исследование представлено введением, тремя главами и заключением.

Обоснование актуальности данной темы представлено во введении, формулируются в исследовании цель и задачи, определяются объект, предмет исследования.

Содержание первой главы посвящено анализу задачи для определения ключевых моментов по разработке обучающего игрового приложения.

Во второй главе проводится разработка и реализация проектных решений.

В третьей главе проводится тестирование разработанного игрового приложения. А также, написание руководства пользователя.

Заключение содержит основные выводы по работе, полученные в ходе выполнения выпускной квалификационной работы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Компьютерная игра — это программа, которая связывает игровой процесс с игроком.

Применение компьютерных обучающих игр в системе образования все больше набирает популярность. Компьютерная обучающая игра является игрой, которая имеет цели обучить игрока. Напротив, обучающую игру можно рассматривать как обучающую систему, в которой процесс обучения совмещен с игровым процессом. Хорошие обучающие игры включают преимущества обучающих систем, и одновременно имеют большой мотивационный потенциал. Качественный критерий обучающей игры представлен в виде соотношения игровых и обучающих компонентов, обеспечивающий целостность восприятия игры и возможность достижения целей обучения.

Метод внедрения обучающего процесса с игровым процессом ориентируется методикой взаимодействия в игровом приложении игровых и обучающих действий, т.е. способом внедрения обучающих курсов в игровой сценарий. В настоящее время, в существующих обучающих системах используются различные пути слияния обучающих и игровых сценариев. Со стороны соотношения обучающих и игровых компонентов выделяют обучающие системы с компонентами игры и игры с компонентами обучения. Обучающие системы с компонентами игры реализуют принцип «edutainment - especially computer games, with an educational aspect» - когда обучающая часть интегрируется в игровую среду. Существующие исследования говорят о том, что такие игры сильно уступают в мотивации, и в итоге не гарантируют достижения цели обучения. Например, в качестве обучающих игр также используются компьютерные игры, по преимуществу игры-стратегии. Такие как, например, SimCity или Civilization, которые разрабатывались как коммерческие продукты и не предназначались для обучения. Однако модели предметной области, реализованные в этих играх,

настолько богаты и реалистичны, что игры имеют самостоятельную обучающую значимость. Разницей, между двумя полярными подходами, являются обучающие игры, которые успешно сочетают обучающую и игровую составляющую. Но, существующие подходы по внедрению обучающего процесса в игровую среду, как правило, не могут существовать в одном продукте (ad-hoc подходы) и не могут быть перенесены на разработку новых продуктов.

Актуальность исследования заключается в том, что во многих учебных учреждениях все чаще используются обучающие компьютерные игры для упрощения обучения. Так как игры больше завлекают детей, чем привычные, традиционные методы обучения, то и необходимо разработать способ интеграции процесса обучения в компьютерные обучающие игры, позволяющего достигать в игре обучающих целей и сохранять при этом игровую привлекательность.

Объектом исследования является технология создания обучающего приложения.

Предметом исследования является процесс созданния трехмерной обучающей игры.

Целью данной выпускной квалификационной работы является разработка трехмерной обучающей игры для школьников по правилам поведения на дорогах.

Для достижения цели необходимо выполнить следующие **задачи**:

1. Проанализировать способы и методы разработки обучающих игр.
2. Спроектировать функциональную модель обучающей игры.
3. На основе требований заказчика, осуществить выбор реализации программного продукта.
4. Реализовать обучающую систему.
5. Протестировать полученные результаты.
6. Написать руководство пользователя.

В ходе выполнения выпускной квалификационной работы будет реализован программный продукт в виде игры.

Выпускная квалификационная работа представлена введением, тремя главами, заключением, списком используемой литературы, приложением.

Во введении описывается актуальность проводимого исследования, формируется цель и ставятся задачи, которые необходимо решить для достижения цели.

В первой главе описывается анализ задачи для определения ключевых моментов по разработке обучающего игрового приложения.

Во второй главе проводится разработка и реализация проектных решений.

В третьей главе приводится тестирование и руководство пользователя.

В заключении приводятся основные выводы по работе, полученные в ходе выполнения выпускной квалификационной работы.

ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

Предметная область – совокупность предметов, их отношений и свойств, которые рассматриваются в рамках определённого контекста. Под контекстом также понимается область, которая является объектом определённой деятельности.

В данной выпускной квалификационной работе рассматривается предметная область разработка компьютерных обучающих игр.

Разработка обучающих компьютерных игр — процесс обучающих создания компьютерных игр. Разработкой обучающих игр занимается разработчик, который может быть, как одним человеком, так и целой компанией. Обычно крупномасштабные коммерческие игры разрабатываются командами разработчиков в пределах компании, специализирующихся на играх для персонального компьютера или консолей. Как полагается, разработку финансирует другая, более крупная компания-издатель, которая по окончанию разработки занимается изданием игры и связанными с ним тратами. Реже компании-издатели могут содержать внутренние команды разработчиков, или же компания-разработчик может разрабатывать игры за свой счет и распространять их без участия издателей, например, средствами цифровой дистрибуции (инди-игры).

В нашей стране ситуация с детским дорожно-транспортным травматизмом была и остаётся очень тревожной. Статистика дорожно-транспортных происшествий свидетельствует, что дети нередко оказываются в аварийных ситуациях. Причиной многих ДТП чаще всего становятся сами дети. Приводит к этому незнание элементарных основ Правил дорожного движения и безучастное отношение взрослых.

О того, насколько хорошо ребёнок усвоил правила безопасного поведения и как применяет их в реальной ситуации в улично-дорожной сети, зависит его здоровье. Для нас, взрослых самое ценное – здоровье и жизнь ребёнка.Очень важно, чтобы соблюдение правил безопасности стало нормой и образом жизни детей и взрослых.

Особое значение в решении этой проблемы имеет заблаговременная и правильная подготовка самых маленьких пешеходов, которых уже сейчас подстерегают серьёзные трудности и опасности. Поэтому изучение Правил дорожного движения целесообразно начинать ещё в школьном возрасте.

Известно, что привычки, закреплённые в детстве, остаются на всю жизнь, поэтому одной из важных проблем в обеспечении безопасности дорожного движения является профилактика детского дорожно-транспортного травматизма.

Актуальность и практическая значимость обучения и воспитания и в целом профилактики детского дорожно-транспортного травматизма подчёркивается высокими статистическими показателями.

В школьном возрасте для детей игра – основной вид развлечения. Ребёнок хочет играть, он играет и познаёт окружающий мир. Игра даёт возможность ребёнку проявить себя. Играя, он не только обучается, но и закрепляет полученные умения и навыки, что способствует формированию положительных привычек.

Таким образом актуальной проблемой данной предметной области является недостаточность обучающих материалов по ПДД для детей школьного возраста.

1.2 Сущность и постановка задачи разработки обучающей трехмерной игры

Для осуществления решения обозначенной проблемы проекта необходимо разработать трехмерную обучающую игру, представляющую собой совокупность

обучающих и игровых компонентов, которые обеспечивают взаимодействие человека с игрой, то есть, такие функции как:

- возможность ввода информации о пользователе;
- возможность интерактивного обучения правилам ПДД;
- возможность различного вывода результатов.

Также заказчиком были выдвинуты основные требования:

- обеспечение интуитивно понятного интерфейса;
- обеспечение современной графики;
- обеспечение хорошей оптимизации для запуска ЭВМ не старше десяти лет;
- обеспечение кроссплатформенности для будущего портирования приложения на мобильные платформы;
- возможность ввода Ф.И.О. и место обучения пользователя;
- возможность быстрого обучения пользователя;
- возможность вывода результата прохождения обучения и запись в базу данных.

Обучающая компьютерная игра – это комплекс обучающих компонентов, интегрированных в игровую среду.

В бакалаврской работе разрабатывается трехмерная обучающая игра для обучения пользователя. Задачей проектирования обучающей игры является создание условий для обучения пользователя.

Требования к функциональным характеристикам:

1) программа должна обеспечивать выполнение следующих функций:
   - ввод информации об пользователе;
   - наличие режима тренировки;
   - наличие режима экзамена;
   - запись результатов в базу данных;

2) исходные данные:

- данные пользователя;

3) отчетная документация:

- формирование отчетов по пройденным этапам тренировки и экзамена;

4) требования к надежности:

- предусмотреть контроль вводимой информации;
- обеспечить целостность хранимой информации;
- защита данных от несанкционированного доступа.

Создаваемая трехмерная обучающая игра должна будет использоваться в школьных учебных учреждениях, но также к ней будут иметь доступ и другие специалисты (для просмотра и изучения данных).

Задачей представленного исследования является разработка технической демонстрационной версии трехмерной компьютерной обучающей игры «Правила поведения на дорогах».

Техническая демонстрация (англ. tech demo) — это приближенная версия или неполная версия продукта, которая создается с целью наглядно продемонстрировать идею, производительность, метод, особенности какой-либо программы.

Обучающие компьютерные игры – это компьютерные программы, которые применяются в целях организации игрового процесса, ведущей задачей которого является восприятие пользователем конкретного образовательного материала. Также в них заложены второстепенные задачи, которые направлены на общее развитие пользователя.

Основой таких игр, в основном, выступает конкретная задача, направленная на получение конкретного результата в виде усвоенных знаний и умений. Главная характеристика заключается в том, что прохождение обучающей игры требует многократного повторения, т.к. целью игры является не просто приобретение новой информации, а формирование знаний, умений, способностей. Однако,

необходимо наличие развивающего потенциала в них, заключающегося в подборе материала, соответственно «уровню ближайшего развития».

Обучающие компьютерные игры можно разделить на категории по типу решаемых задач. Это могут быть игры, содержащие одну конкретную задачу или целый блок различных обучающих заданий, объединенный общей темой, или представленный различными предметными задачами. Итак, можно выделить следующую типологию:

- игры, которые содержат направленность на освоение компьютера. В качестве примера, можно привести игру, в которой пользователь получает начальные представления о языках программирования, играя в «ЛогоЧерепашки»;
- игры с математическим содержанием. Это различные игры с числами от простого узнавания цифр или определения количества предметов до выполнения математических действий; игры на формирование начальных математических представлений (понятия больше-меньше-равно, геометрические фигуры и т.д.);
- игры на освоение базовых элементов русского языка. Сюда относятся игры на запоминание букв, фонематические упражнения, игры на изучение структуры слова, тренажеры для обучения чтению, различные прописи-раскраски (особенно распространены программы для планшетных компьютеров);
- игры для освоения иностранных языков;
- игры, направленные на изучение окружающего мира. В содержание данные игр входит: запоминание цветов, животных и их потомства; игры из серии «кто что ест» и «кто где живет», а также ознакомление с днями недели, временами года, временем суток, с такими категориями как: транспорт, профессии, фрукты, овощи, целое и его части.

1.3 Анализ существующего аналога

Рассмотрим существующий аналогичный программный продукт – игру под названием «Дорога в школу», разработанную группой компаний «HI-TECH».

Главной задачей в игре является перемещение игрока от дома до школы, преодолевая разные препятствия, и угадывая знаки. Ее целью также является обучить детей правилам дорожного движения.

Игра является двухмерной флеш-игрой. После запуска приложения нас встречает главное меню, где нам даётся возможность ввести своё имя, и право выбрать за кого играть.(Рисунок 1.1)

Рисунок 1.1 - Главное меню игры «Дорога в школу»

После ввода имени и выбора персонажа, мы перемещаемся в двухмерный игровой мир, камера которого расположена сверху (Рисунок 1.2). Нам дается право управления персонажем, посредством клавиш управления стрелками вверх, вниз, влево, вправо на клавиатуре. Мы должны вести персонажа по игровому миру.

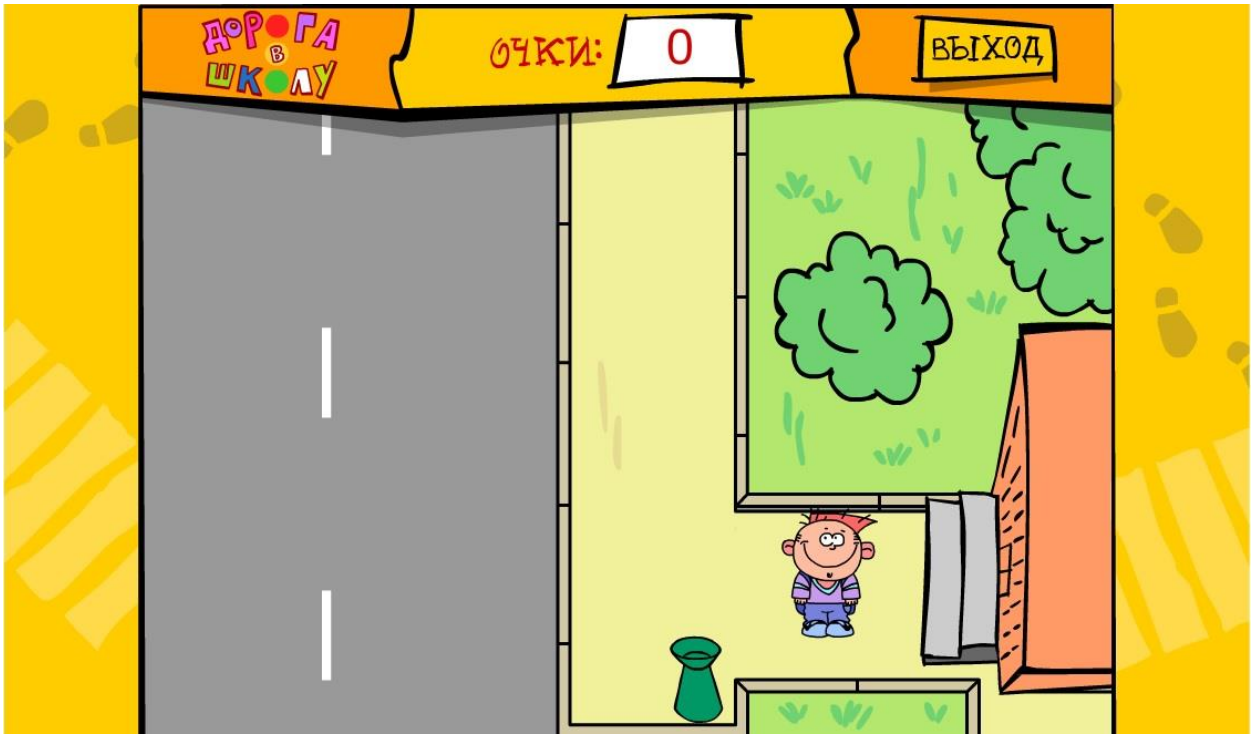

Рисунок 1.2 - Игровой мир игры «Дорога в школу»

Далее, по достижению конца уровня игроку даётся задание, он должен выбрать один из трех вариантов ответов на вопрос (Рисунок 1.3).

Рисунок 1.3 - Задание

За правильно данный ответ игроку даются очки. После ответа на вопрос игрок снова переходит в игровой мир, где надо пройти до конца уровня.

Рисунок 1.4 - Выбор правильного ответа

После всех пройденных уровней выводится на экран, выводится заключительная заставка, где написан конечный счёт, набранный за всю игру. Также на нём предлагается повторить игру, либо выйти из приложения (Рисунок 1.5).

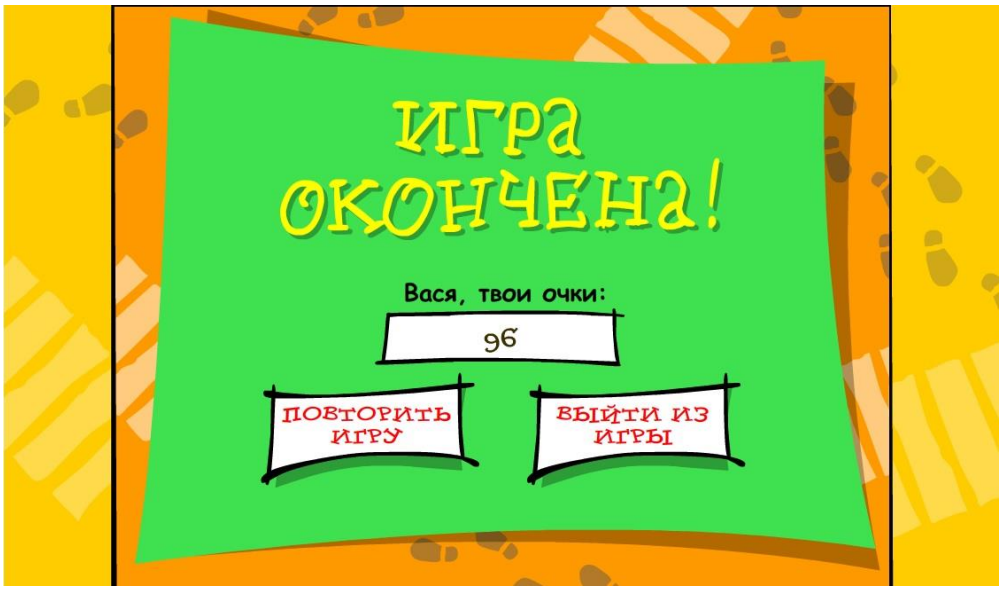

Рисунок 1.5 - Заключительная заставка

В этом разделе мы ознакомились с основными понятиями обучающих приложений, проанализировали детскую обучающую игру «Дорога в школу».

Таким образом, в ходе анализа были выявлены следующие недостатки существующих обучающих приложений на тематику ПДД:

- сложность в понимании для детей;
- необходимость доступа в сеть Интернет;
- использование устаревшей технологии Adobe Flash;
- требуется установка дополнительного программного обеспечения (браузер, Adobe Flash Player).

Поэтому, в ходе выполнения выпускной квалификационной работы будут учтены недостатки существующих продуктов по обучение правилам ПДД для детей школьного возраста.

Вывод первой главе был проведен анализ предметной области, а также её обозначение. Также была рассмотрена сущность и постановка задачи разработки обучающей трехмерной игры. И был приведен анализ существующего аналогичного приложения.

ГЛАВА 2 РЕАЛИЗАЦИЯ ОБУЧАЮЩЕЙ ИГРЫ

2.1 Выбор средств реализации проекта

Ключевым средством реализации данной работы является игровой движок Unity (Рисунок 2.1).

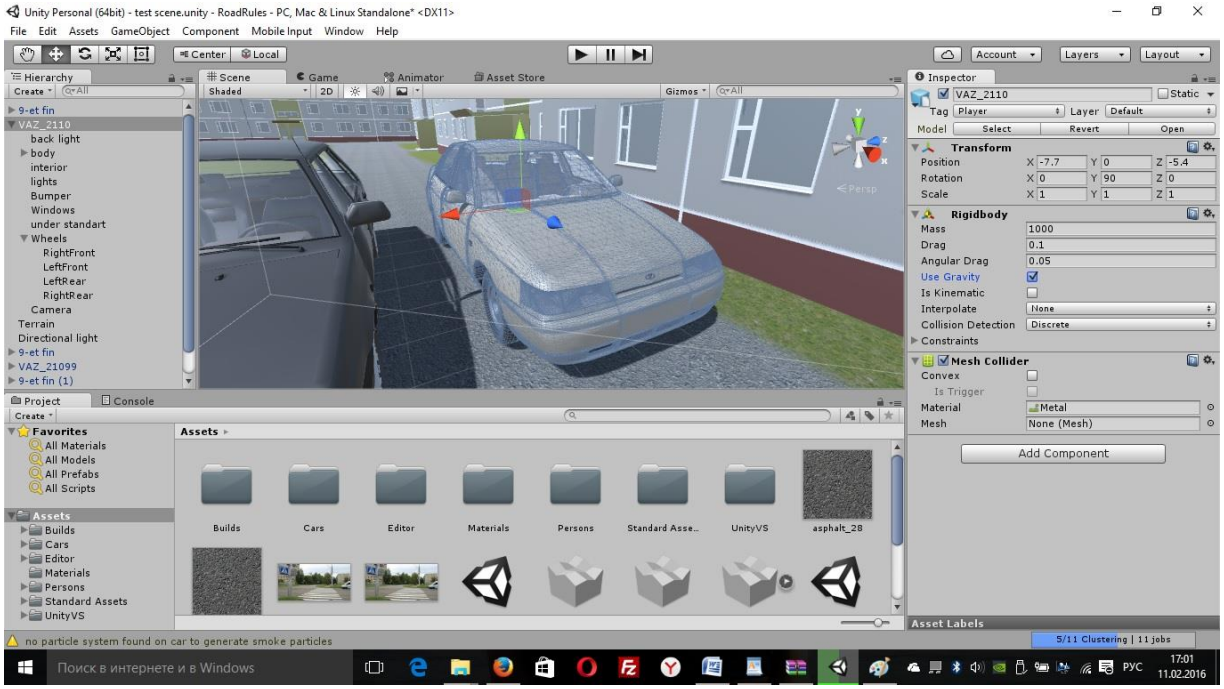

Рисунок 2.1 – Окно программы Unity

Unity — это кроссплатформенный инструментарий для реализации двухмерных(2D) и трехмерных(3D) программ и игр, работающий под операционными системами Windows, Linux и OS X. Созданные с помощью Unity приложения работают под операционными системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а также на игровых приставках Wii, PlayStation 3, PlayStation 4, PlayStation Vita, Xbox 360 и Xbox One. А также, есть возможность создавать интернет-приложения с помощью специального подключаемого модуля к браузеру Unity Web Player. Приложения, созданные с помощью Unity, поддерживают DirectX и OpenGL, также данная технология является бесплатной, что является основным преимуществом данной технологии.

Основные характеристики Unity:

- Скрипты, написанные на C#, UnityScript (модификация JavaScript) и Boo (разновидность языка Python со строгой типизацией для платформы.NET);
- игровой движок полностью связан со средой разработки Mono. Это позволяет прямо в редакторе испытывать игру;
- работа с ресурсами возможна через простой Drag&Drop. Интерфейс редактора настраиваемый;
- осуществлена система наследования объектов;
- поддержка импорта из очень большого количества форматов;
- встроенная поддержка сети;
- есть решение для совместной разработки — Version Control;
- также можно использовать подходящий пользователю способ контроля версий. К примеру, Tortoise SVN или Source Gear.

В качестве среды разработки и редактирования скриптов был выбран MonoDevelop от MonoDevelop Team. На рисунке 2.2 можно рассмотреть процесс написание скрипта главного меню.

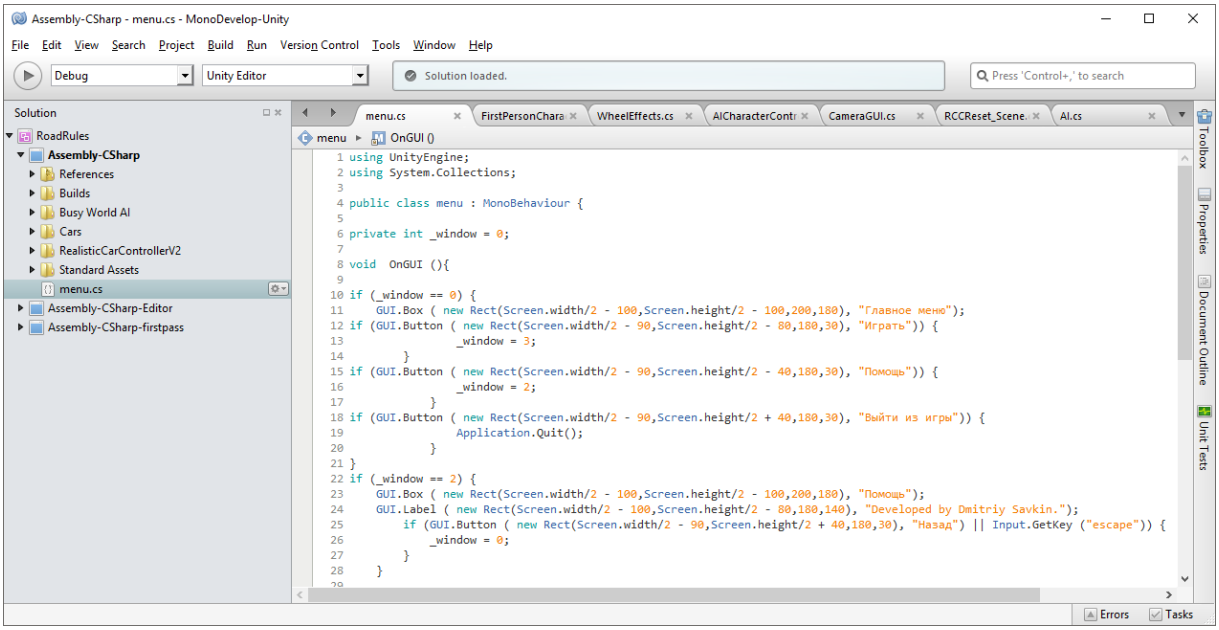

Рисунок 2.2 – Окно редактора MonoDeveloper

Основной платформой для разработки были выбраны операционные системы семейства Microsoft Windows (XP/Vista/7/8/10). В основном, языком программирования является C# на платформе.NET 2.0. Реализация не сложных скриптов выполняется на UnityScript, реже в – Boo. В качестве графической библиотеки используется Microsoft DirectX; основным шейдерным языков является Cg (англ. C for Graphics), разработанный компанией NVidia.

Для создания трехмерных моделей было принятое решение использовать продукт 3ds Max версии 2009 от Autodesk.

3ds Max – многофункциональная, мощная, профессиональная программная система для создания и редактирования трехмерной графики и анимации, разработанная компанией Autodesk. 3ds Max имеет самые современные инструменты, как для художников, так и для специалистов в области мультимедиа. Работает в семействе операционных систем Microsoft Windows, как в 32-битных, так и в 64-битных. Также является условно бесплатным. На рисунке 2.3 можно наблюдать запущенную программу с открытой трехмерной моделью автомобиля ВАЗ 10-ой серии, которая была взята с бесплатного источника и используется в данной работе.

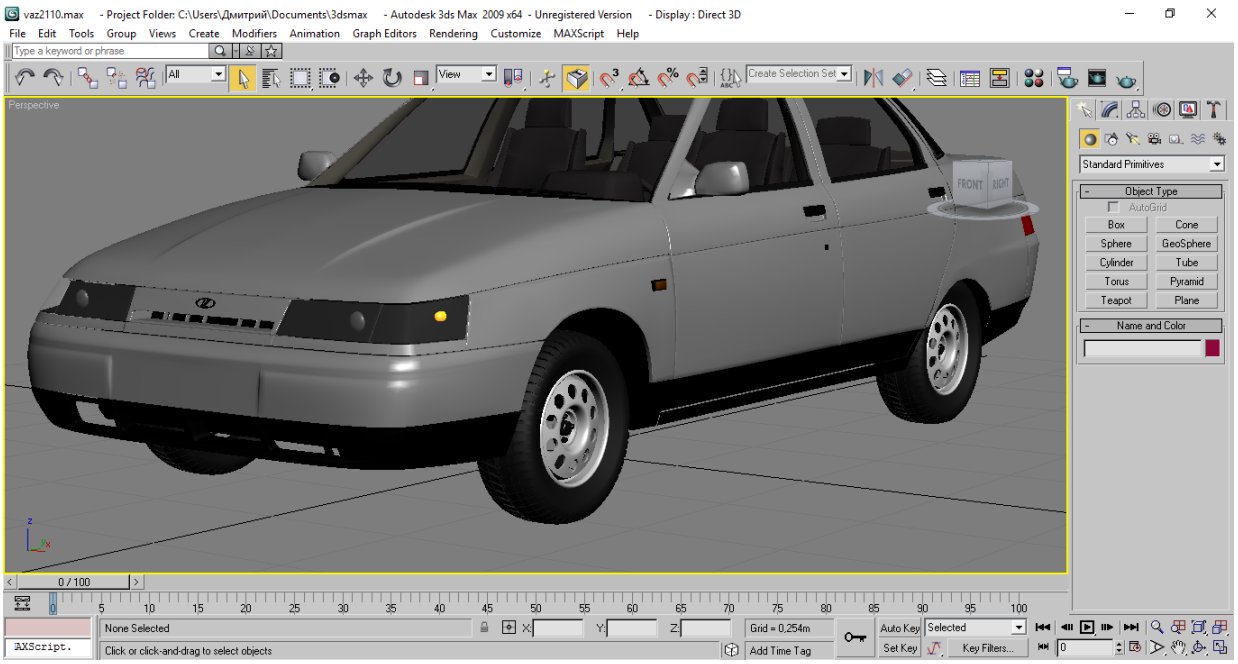

Рисунок 2.3 – ВАЗ 2110 в 3ds Max

В качестве системы управления базами данных(СУБД) выбор пал на SQLite версии 3. SQLite – это компактная мультиплатформенная СУБД, написанная на языке C, которая имеет не такой большой функционал, как MySQL, но идеально подходит для данного проекта. Также, SQLite не является отдельной программой, и не использует парадигму Клиент-Сервер. Оно встраивается в разрабатываемое приложение как библиотека(API). Движок SQLite и интерфейс к ней реализованы в одной библиотеке. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных, включая определения, таблицы, индексы и данные, в единственном стандартном файле на том устройстве, на котором выполняется программа. Также, данная СУБД является очень надёжной, так как при выпуске версии она проходит через ряд серьезнейших автоматических тестов. Самым большим достоинством данной СУБД является её компактность, и возможность использовать на любом устройстве, даже на смартфоне.

Также SQLite отличается относительно небольшим набором используемых типов данных.

Типы данных используемые в SQLite:

- Integer - Числовой тип данных. Хранит как целые положительные числа, так и отрицательные;
- Text - Текстовый тип данных который может хранить текстовые строки любой длины в кодировке UTF-8 или UTF-16. Этот же тип данных может использоваться для хранения даты и времени;
- Real – Числовой тип данных, хранящий как целые, так и дробные числа;
- Blob - Тип данных для хранения двоичных объектов. Может использоваться как для хранения файлов.

Все эти типы данных соответствуют данному программному проекту, несмотря на свой небольшой выбор.

Все программы являются свободно либо условно-свободно распространяемым ПО.

В данном разделе была приведена перечень и обоснования средств реализации данной компьютерной обучающей игры

2.2 Моделирование базы данных

Информационная модель – совокупность информации, характеризующая свойства и состояние объекта, процесса, явления, а также взаимосвязь с внешним миром.

При проектировании программного обеспечения выясняются запросы и пожелания заказчика, и определяется возможный подход к решению задачи. Задача анализируется. На основе этого анализа реализуется конкретная модель в конкретной программной среде. Результаты каждого этапа проектирования используются в качестве исходного материала следующего этапа.

При проектировании системы обработки данных больше внимания необходимо уделить организации данных. Понять организацию данных можно с помощью информационной модели.

Информационная модель - совокупность информации, характеризующая существенные свойства и состояния объекта, процесса, явления, а также взаимосвязь с внешним миром. Информационные модели нельзя потрогать или увидеть, они не имеют материального воплощения, потому что строятся только на информации.

Информационная модель (концептуальная модель) включает в себя совокупность входных и выходных документов, файлов входной оперативной, постоянной, промежуточной и результатной информации.

Концептуальная модель представляет объекты и их взаимосвязи без указывания способов их физического хранения.

Построение мощной и наглядной концептуальной модели данных позволяет точнее оценить специфику моделируемой предметной области во избежание возможных ошибок на стадии проектирования схем реляционной базы данных. На этапе концептуального моделирования производится очень важная документация, которая может оказаться полезной не только при проектировании схемы реляционной базы данных, но и при эксплуатации, сопровождении и развитии уже заполненной базы данных.

На рисунке 2.4 показана концептуальная модель, включающая в себя описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области и выявляемых в результате анализа данных.

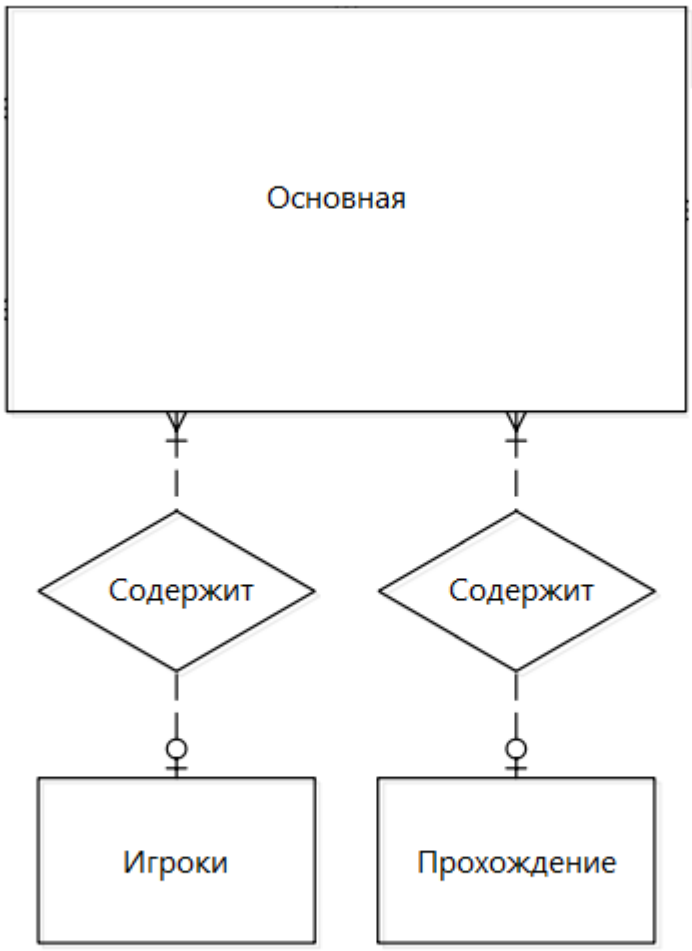

Рисунок 2.4 - Концептуальная модель данных

По окончании данного этапа получаем концептуальную модель, инвариантную к структуре базы данных. Данная модель отображает основные объекты и их характеристики.

На основе концептуальной модели была построена логическая модель данных. Данная модель описывает объекты предметной области, которые подлежат хранению в базе данных. Логическая модель строится на основе концептуальной модели данных.

Таблица 2.1- Основная таблица(main)

Название поля	**Описание поля**	**Ключ**	**Тип поля**	**Null значения**
id_game	Идентификатор игры	Primary key (Первичный ключ)	Integer	NOT NULL
id_player	Идентификатор игрока	Foreign key (вторичный ключ)	Integer	NOT NULL
id_level	Информация о школе	Foreign key (вторичный ключ)	Integer	NOT NULL
time	Прохождение обучения	-	Integer	NOT NULL

Таблица 2.2 - Игроки(player)

Название поля	**Описание поля**	**Ключ**	**Тип поля**	**Null значения**
id_player	Идентификатор пользователя	Primary key (первичный ключ)	Integer	NOT NULL
f_i_o	Ф.И.О	-	Text	NOT NULL
school	Школа	-	Text	NOT NULL
grade	Класс	-	Text	NOT NULL

Таблица 2.3 - Прохождение уровней(levels)

Название поля	**Описание поля**	**Ключ**	**Тип поля**	**Null значения**
id_level	Идентификатор прохождения уровней	Primary key (Первичный ключ)	Integer	NOT NULL
level_number	Пройденные уровни	-	Text	NOT NULL

Исходя из вышеперечисленных данных, была составлена логическая модель базы данных, которая представлена на рисунке 2.5.

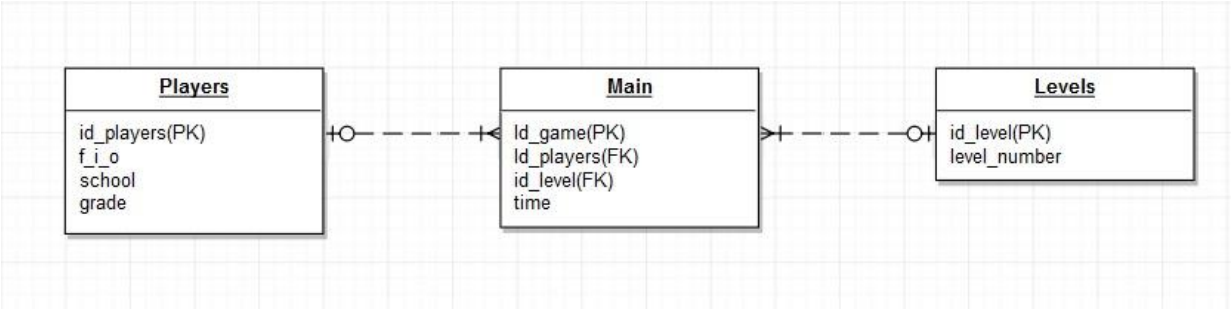

Рисунок 2.5 – Логическая модель данных

На основе составленной логической модели было осуществлено проектирование физической модели базы данных. Физическим аналогом сущности в разрабатываемой базе данных является таблица, а физическим аналогом атрибута – поле этой таблицы. Результатом этого процесса является физическая модель, содержащая полную информацию для генерации всех объектов в базе данных.

На рисунке 2.6 представлена физическая модель базы данных. Как видно из рисунка, поля в таблицах базы данных имеют тип соответствующий выбранной СУБД – SQLite.

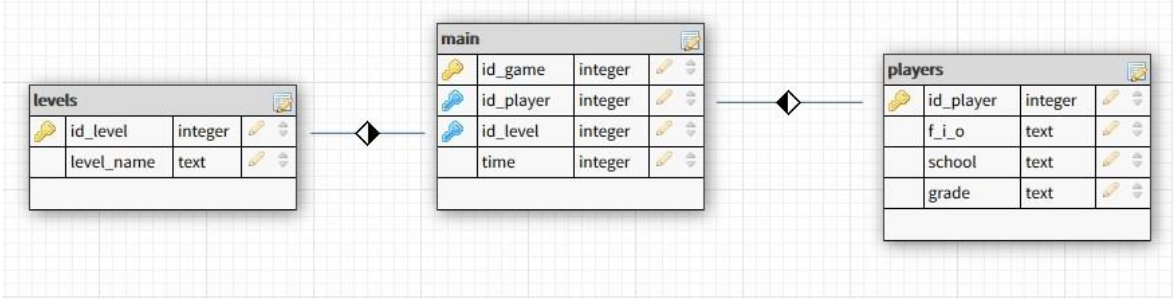

Рисунок – 2.6 Физическая модель данных

Реляционная модель данных – это логическая модель данных, описывающая:

- структуры данных в виде (изменяющихся во времени) наборов отношений;
- теоретико-множественные операции над данными: объединение, пересечение, разность и декартово произведение;
- специальные реляционные операции: селекция, проекция, соединение и деление;
- специальные правила, обеспечивающие целостность данных.

Особенность реляционной модели заключается в том, что объекты и взаимосвязи между ними представляются в базе данных единообразно в виде нормализованных отношений.

Исходя из построенных моделей данных, можно выделить основные сущности и их атрибуты, описывающие базу данных. Определив данные сущности, можно перейти к разработке компьютерного обучающего приложения.

2.3 Структурная схема обучающей компьютерной игры и технологическое обеспечение задачи

Структура системы - это его основа, способ разделения на страницы, разделы, блоки, а также способ их отображения с целью наиболее удобного использования сайта пользователем.

После того как выявлены наиболее важные операции, рассмотрим каждую из них более подробно. Алгоритмы построены средством приложения Сассоо,

предназначенного для построения блок-схем как вручную. Система строит блок-схемы согласно ГОСТу 19.701-90. - Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.

Система заполнения данных о пользователе разрабатываемой программы является основной операцией и устроена следующим образом: сначала формируется пользователь, затем вводятся данные о нем, после чего он сохраняется в базе данных.

Таким образом, процедура заполнения данных о пользователе является сложной, многоуровневой операцией, и представлена на рисунке 2.7.

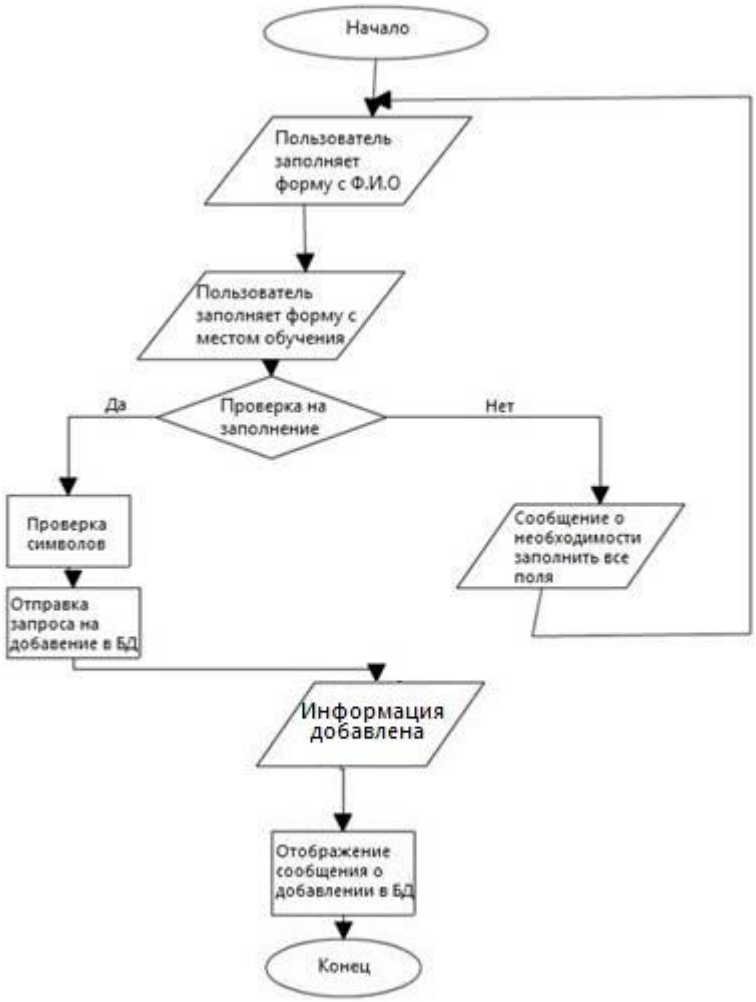

Рисунок 2.7 - Алгоритм заполнения данных о пользователе

2.4 Разработка внутриигрового мира и объектов

Начало разработки начинается с создания нового проекта в Unity. После того, как создали пустой проект, нам представляется окно с пустым игровым миром, и пустым полем иерархий объектов.

Для пустого проекта нам понадобятся трехмерные модели(объекты), чтобы заполнить ими пространство. Улицы были спроектированы с нуля, в программе 3Ds Max. На рисунке 2.8 представлен процесс моделирования трехмерной модели улицы. Некоторые модели, такие как светофоры и автомобили, были заимствованы с сайта http://www.3ddd.ru.

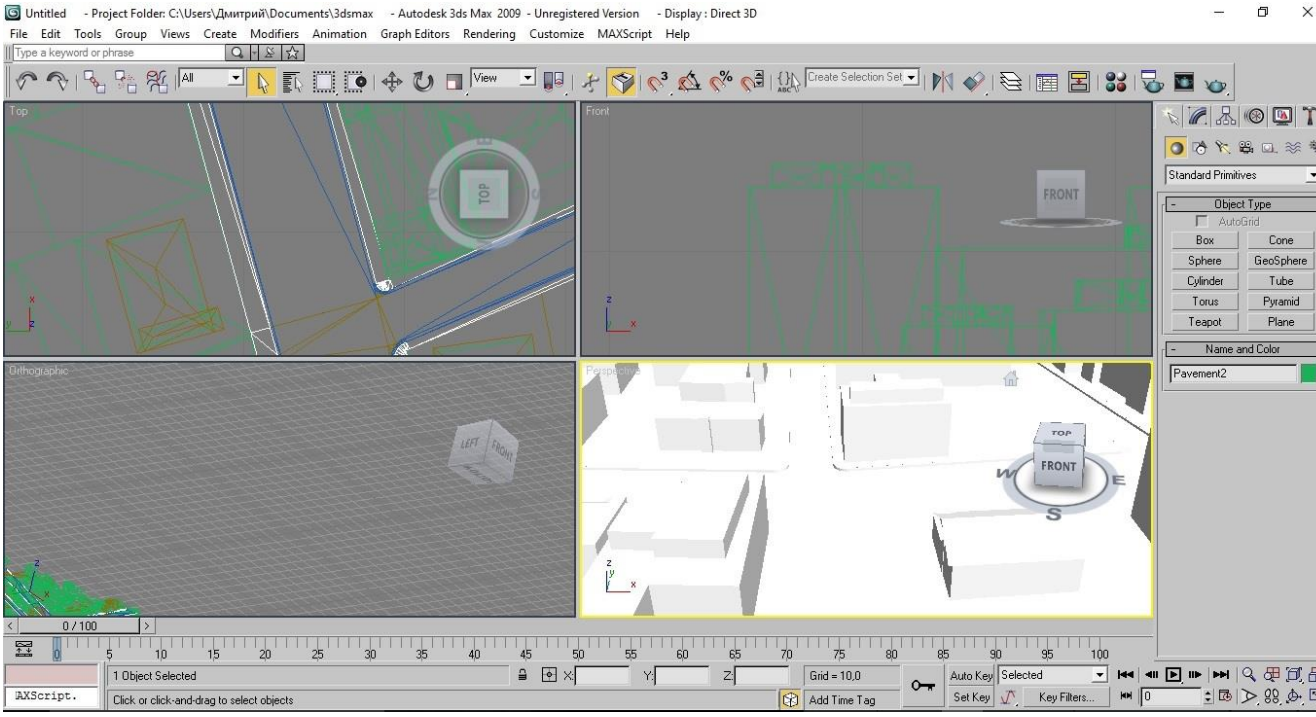

Рисунок 2.8 - Построение трёхмерной модели улицы.

После создания трехмерных объектов в Unity им присваиваются текстуры. Только там используются не просто текстуры, а так называемые Материалы (англ. Materials) . На рисунке 2.9 можно наблюдать меню материй, в котором содержатся разные параметры.

Рисунок 2.9 Меню материй

После наложения текстур на модели, придаём объектам физические свойства, такие как: гравитация и сила тяжести. Далее, для каждого объекта задаётся задаётся положение в пространстве. После того, как задали все необходимые свойства объектам, переходим к написанию сценариев(скриптов).

Далее, после написания всех скриптов, прикрепляем их к нужным объектам. В зависимости от написанного скрипта, меняем параметры, которые необходимы.

После всех проделанных пунктов приложение компилируется. Окно выбора параметров компиляции изображено на рисунке 2.10.

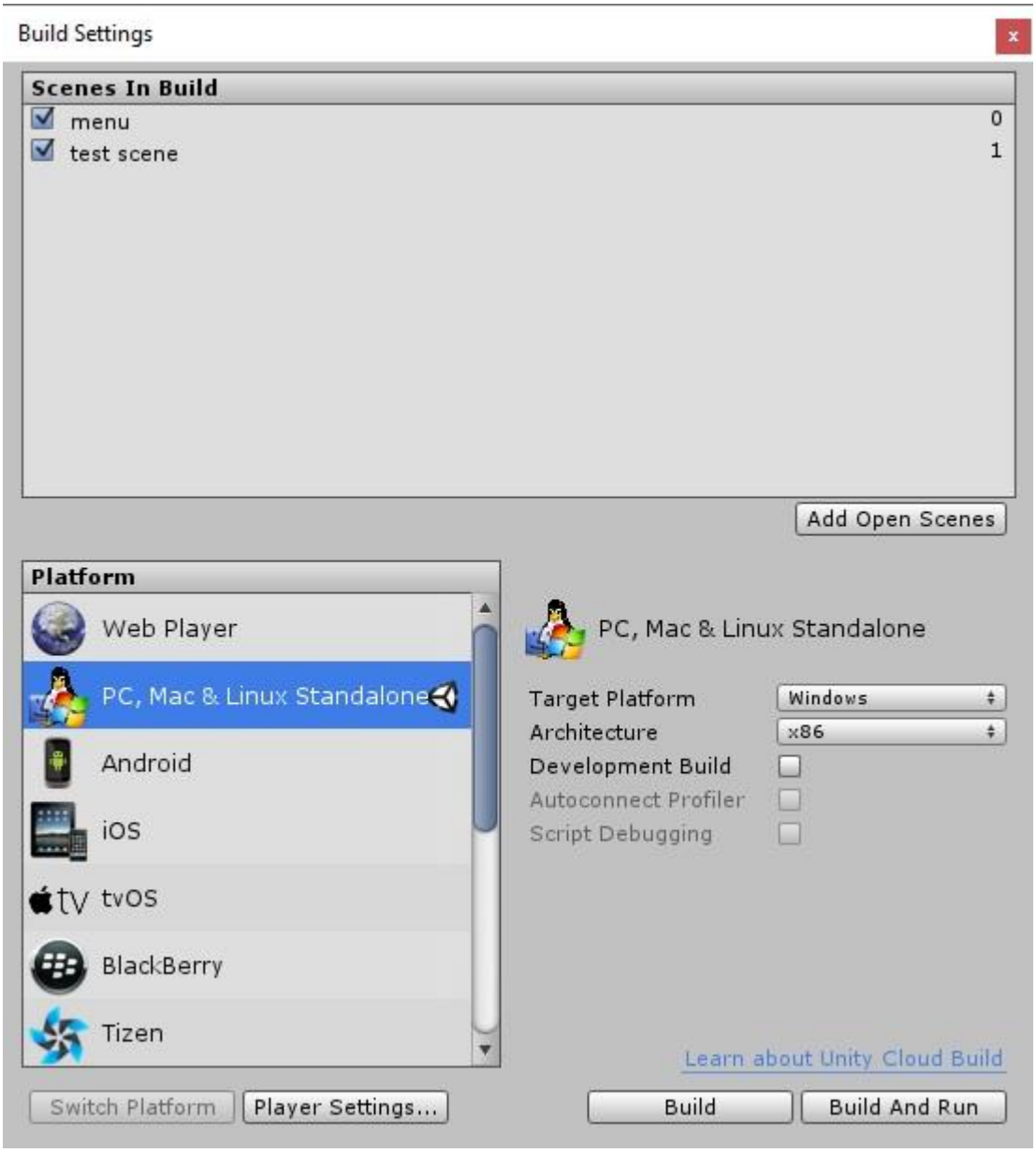

Рисунок 2.10 – Окно выбора параметров компиляции приложения.

2.5 Основные классы приложения

Игровой движок Unity имеет лицензию несвободного программного обеспечения с закрытым исходным кодом. Все программирование выполняется с помощью сценариев (скриптов). В качестве скриптовых языков на Unity

используются C#, JavaScript и Boo. В каждый из языков внесены модификации для поддержки полного функционала внутреннего скриптового языка UnityScript.

Сценарии являются ясным, легким и быстрым подходом программирования. Все три скриптовых языка используют платформу Open Source.NET (Mono).

Так как весь функционал реализуется с помощью скриптов, они должны быть наследованы от родительского класса MonoBehaviour, только после этого они могут быть интегрированы в конкретные внутриигровые объекты. Пример использования родительского класса в скрипте главного меню можно наблюдать на рисунке 2.11.

```
using UnityEngine;
using System.Collections;

public class menu : MonoBehaviour {

private int _window = 0;

void  OnGUI (){

if (_window == 0) {
    GUI.Box ( new Rect(Screen.width/2 - 100,Screen.height/2 - 100,200,180), "Главное меню");
if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 80,180,30), "Играть")) {
                _window = 3;
        }
if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 40,180,30), "Помощь")) {
                _window = 2;
            }
if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 + 40,180,30), "Выйти из игры")) {
                Application.Quit();
            }
}
```

Рисунок 2.11 – Создание дочернего класса Menu от MonoBehavior

Программирование на движке Unity является объектно-ориентированным.

Все внутриигровые объекты (персонажи, предметы и т.д.) являются экземплярами классов, описывающих их поведения. Все внутриигровые события (эффекты, сцены и т.д.) определяются скриптами. Игровой процесс получается из совместной работы менеджеров, каждый из которых отвечает за свою часть игрового процесса (геймплея):

Класс	Предназначение
Menu	Класс главного меню.
Pause	Класс меню паузы.
AI	Класс, отвечающий за поведение искусственного интеллекта.
WayPoint	Класс, отвечающий за определение пути у искусственного интеллекта.
TrafficLight	Класс, отвечающий за работу светофора.
Vector3	Класс, содержащий основные операции с вычислением векторов.

Таблица 2.4 – Таблица классов

Все классы реализованы на основе паттерна Singleton («одиночка»). Все они являются дочерними классами родительского закрытого класса MonoBehavior. Также, они имеют только один экземпляр и являются общими для всей игровой системы. На рисунке 2.12 изображена диаграмма классов.

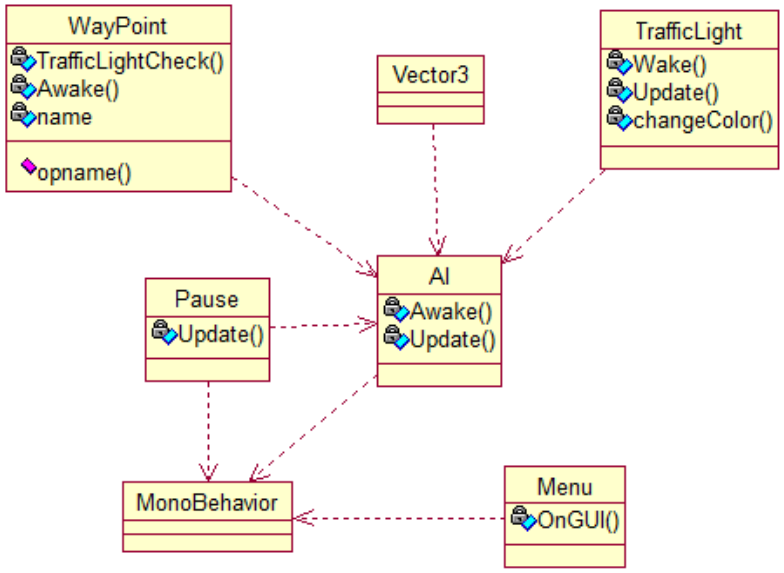

Рисунок 2.12 – Диаграмма классов

2.6 Основные алгоритмы приложения

Компьютерная игра является сложной системой, состоящей из отдельных подсистем, которые разрабатываются независимо, а затем интегрируются в единую программную архитектуру. В данной работе проект можно разбить на следующие подсистемы:

- система поиска пути;
- система работы светофора;
- система графического интерфейса пользователя;
- система взаимодействия объектов;
- и дополнительные системы управления приложением.

Рассмотрим эти системы по-отдельности.

Для определения расстояния между объектом и персонажем у искусственного интеллекта используется класс Vector3.

Vector3 – класс векторов, с помощью которого ведется представление не только векторов, но и точек в пространстве, так как с математической точки зрения эти понятия эквиваленты.

Класс Vector3 выполняет следующие операции:

- Присваивание векторов;
- Сравнение двух векторов;
- Сумма/разность двух векторов;
- Умножение/деление вектора на скаляр;
- Скалярное и векторное произведение векторов;
- Нахождение величины вектора;
- Нормализация вектора;
- Нахождение расстояния между двумя точками;

Код класса Vector3 представлен в Приложении Е.

Следующая рассматриваемая система – система поиска пути.

Поиск пути (англ. Pathfinding) — термин, который используется в информатике и искусственном интеллекте, который осуществляет определение компьютерной программой наилучшего, ближайшего пути между двумя точками. Пример поиска пути представлен на рисунке 2.13.

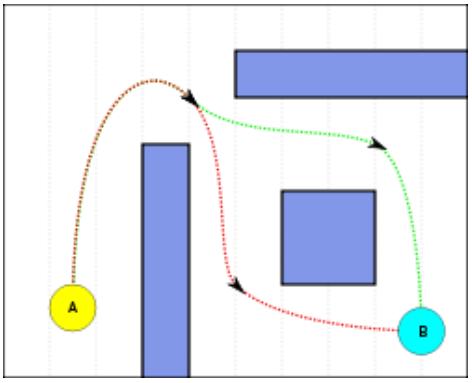

Рисунок 2.13 – Пример расчета пути в двумерном пространстве

Существует огромное множество алгоритмов поиска пути. В данной работе поиск пути реализован с помощью алгоритма навигационных сеток. Навигационная сетка (англ. Navmesh) - это абстрактная структура данных используемая в программах с использованием искусственного интеллекта, чтобы агенты движения нашли путь через большие геометрические и сложные трехмерные пространства. Объекты, не являющимися статическими препятствиями в окружающей среде, воспринимаются искусственным интеллектом как динамические преграды. Это дополнительно дает преимущество данному подходу решения задачи поиска пути в пространстве, так как агенты, которые имеют доступ к навигационной сетке, не будут рассматривать эти препятствия во время построения своей траектории движения. Метод навигационных сеток обеспечивает снижение вычислительных затрат и делает обнаружение столкновений между агентами и динамическими преградами менее затратным. Навигационные сетки обычно реализуются в виде графов, открывая их использование большому числу алгоритмов, определенных на этих структурах.

В отличие от классических решений задачи поиска пути в навигационных сетках вместо того, чтобы представлять пространство как ряд соединенных точек, создавать более точное представление конфигурационного пространства ИИ через связный граф выпуклых многоугольников. На каждом узле (полигоне) заранее известно, куда агент движения может переместиться из одной точки этого узла к любой другой точке этого же узла из-за его выпуклости. Таким образом, задача поиска пути через граф упрощается до поиска пути вдоль связного графа узлов.

Путь представляется серией многоугольников, которые агент должен пройти, чтобы достичь цели. Вместо того чтобы проверять каждую точку вдоль пути, ИИ имеет всю информацию, связанную с интерфейсом между узлами навигационной сетки. Это позволяет получить точный и гораздо более естественный вид движения.

На рисунке 2.14 изображен пример расчета пути на конфигурационном пространстве, представленным навигационной сеткой.

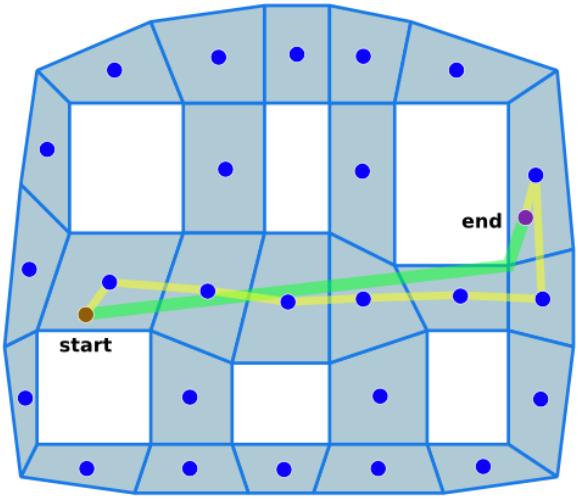

Рисунок 2.14 – Пример поиска пути на основе навигационной сетки

Следующая система, которая надлежит рассмотрению – это работа светофора. Так как в данном программном продукте реализован симулятор города, там также используются светофоры, для регулировки дорожного потока. Светофор имеет два световых сигнала красный и зеленый. При включении зеленого все персонажи игре начинают движение. При включении красного света,

все персонажи останавливаются. Данная система реализована на языке C#, и содержится в Приложении Д.

Следующий компонент приложения, который надлежит к рассмотрению – это графический интерфейс пользователя.

Графический пользовательский интерфейс (англ. Graphical user interface, GUI) — один из видов пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на мониторе, выполнены в виде графических изображений.

На рисунке 2.15 изображено главное меню приложения. Графический интерфейс пользователя является верхним слоем графической системы, что позволяет создавать на его фоне полноценные трехмерные сцены.

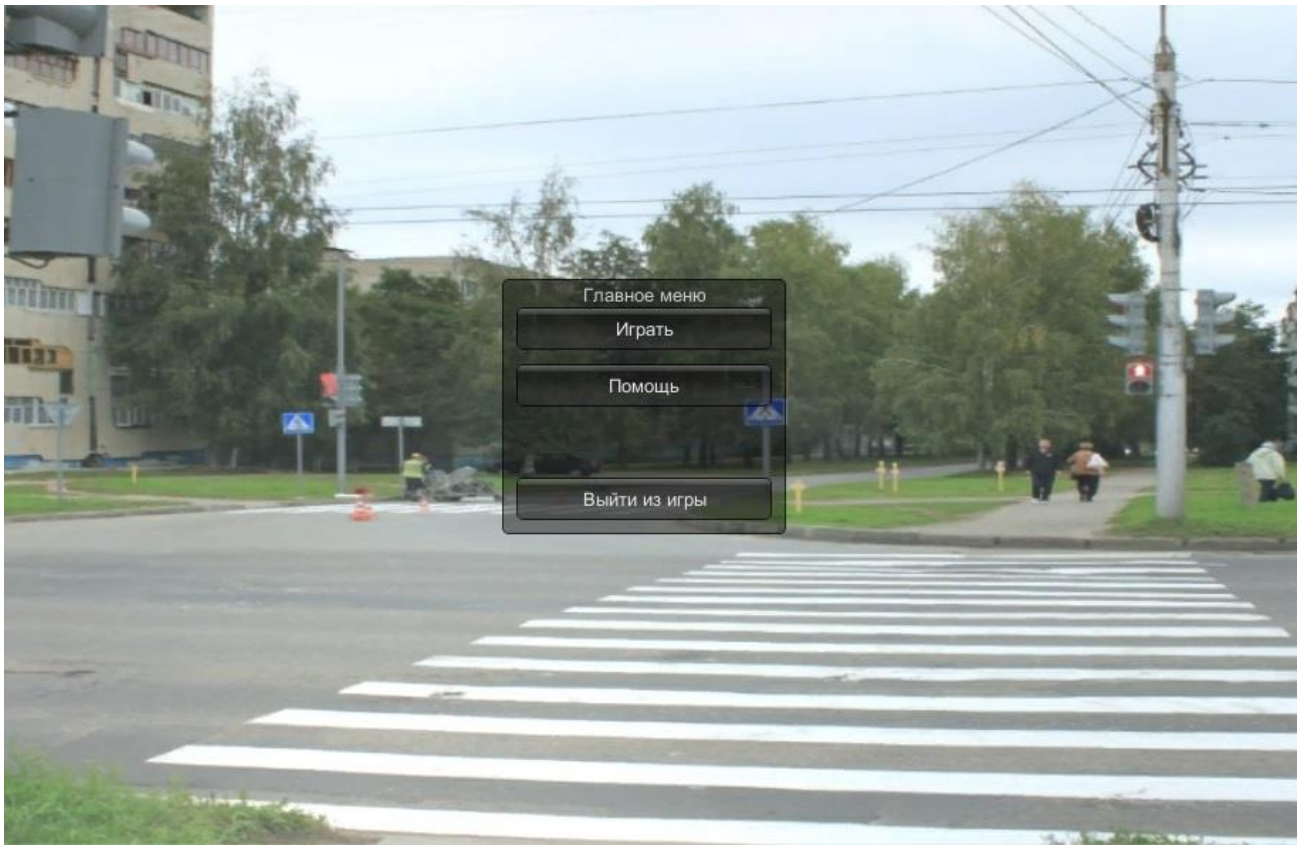

Рисунок 2.15 – Главное меню приложения

Одна из важнейших систем, без которых невозможно представить игровой цикл, – это система взаимодействия объектов.

Взаимодействие объектов – процесс воздействия объектов друг на друга. Данный процесс описываем механику игры. К системе взаимодействия объектов можно отнести:

- движущиеся объекты;
- интерактивные объекты (светофоры и т.д.);
- управление персонажем с помощью мыши/клавиатуры;
- коллизию объектов, обладающих физическими свойствами;
- звуковую подсистему (звуки шагов, машин, и т.д.);
- искусственный интеллект компьютерных персонажей (ботов).

Все эти элементы являются главными составляющими игрового цикла.

На рисунке 2.16 представлен скриншот игрового процесса. На нем представлено взаимодействие персонажа, управляемого пользователем, и ботов, контролируемого искусственным интеллектом.

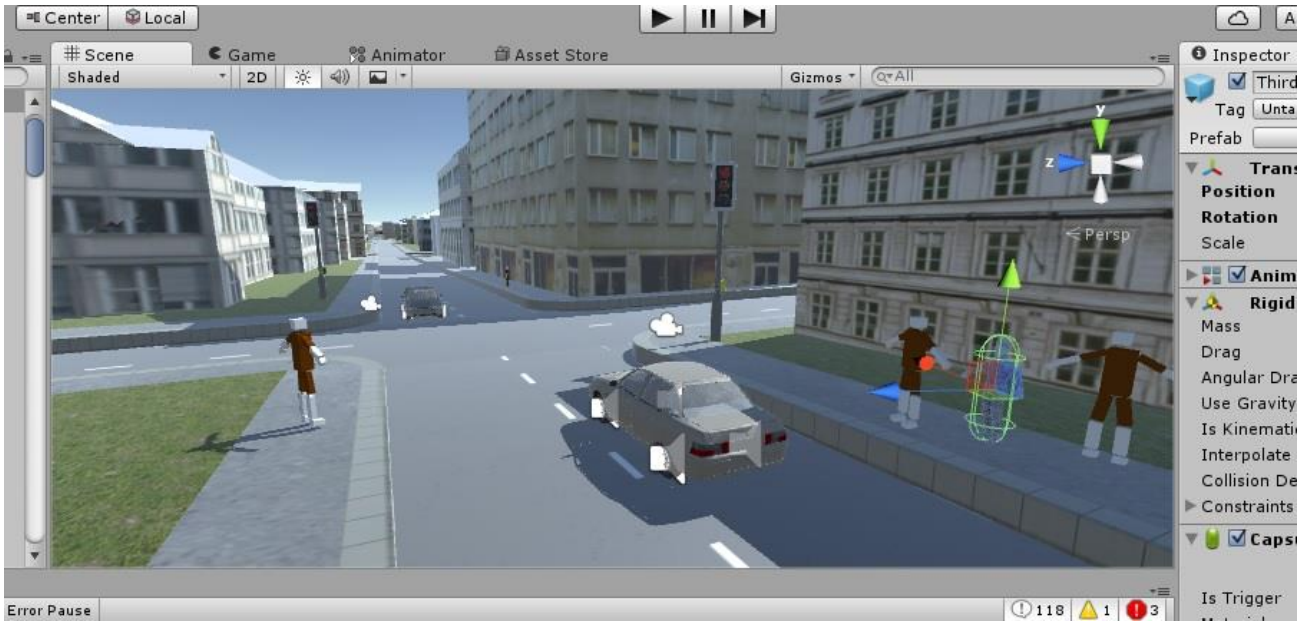

Рисунок 2.16 – Игровой процесс

Все игровые подсистемы были полностью реализованы и успешно интегрированы в единую игровую систему, которая позволяет запустить игровой процесс.

2.7 Архитектура системы и иерархия классов

В качестве основной парадигмы программирования используется объектно-ориентированный подход. ООП ориентировано на разработку крупных программных комплексов. Объектно-ориентированное проектирование состоит в описании структуры и поведения проектируемой системы, то есть, фактически, в ответе на два основных вопроса:

- из каких частей состоит система;
- за что ответствует каждой из частей.

Выделение частей производится таким образом, чтобы каждая имела минимальный по объёму и точно определенный набор выполняемых функций (обязанностей), и при этом взаимодействовала с другими частями как можно меньше.

Дальнейшее уточнение приводит к выделению более мелких фрагментов описания. По мере детализации описания и определения ответственности выявляются данные, которые необходимо хранить, наличие близких по поведению агентов, которые становятся кандидатами на реализацию в виде классов с общими предками. После выделения компонентов и определения интерфейсов между ними реализация каждого может проводиться практически независимо от остальных (разумеется, при соблюдении соответствующей технологической дисциплины).

Большое значение имеет правильное построение иерархии классов. Одна из общеизвестных проблем больших систем, построенных по ООП-технологии — так называемая проблема хрупкости базового класса. Она состоит в том, что на поздних этапах разработки, когда иерархия классов построена и на ее основе разработано большое количество кода, оказывается трудно или даже невозможно

внести какие-либо изменения в код основных классов иерархии (от которых порождены все или многие работающие в системе классы). Даже если вносимые изменения не затронут интерфейс базового класса, изменение его поведения может непредсказуемым образом отразиться на классах-потомках. В случае крупной системы разработчик основного класса просто не в состоянии предугадать последствия изменений, он даже не знает о том, как именно основной класс используется и от каких особенностей его поведения зависит корректность работы классов-потомков.

Объединение всех подсистем, реализованных с помощью ООП и представленных в виде иерархии классов, определяют внутреннюю структуру программного продукта.

На рисунке 2.17 изображен прототип архитектуры игрового движка, за основу который был взят при разработке данного приложения. Сплошной стрелкой отображена иерархия подсистем, а пунктирной – интерфейс между системами.

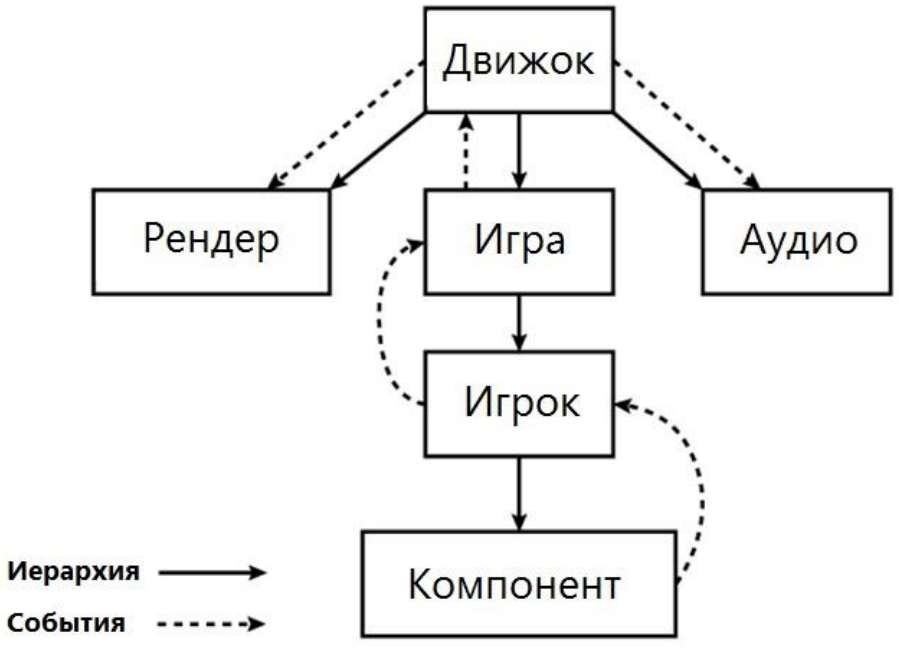

Рисунок 2.17 – Прототип архитектуры игрового движка

Исходя из прототипа архитектуры игрового движка, можно построить архитектуру самого игрового приложения. Архитектура приложения представлена на рисунке 2.18.

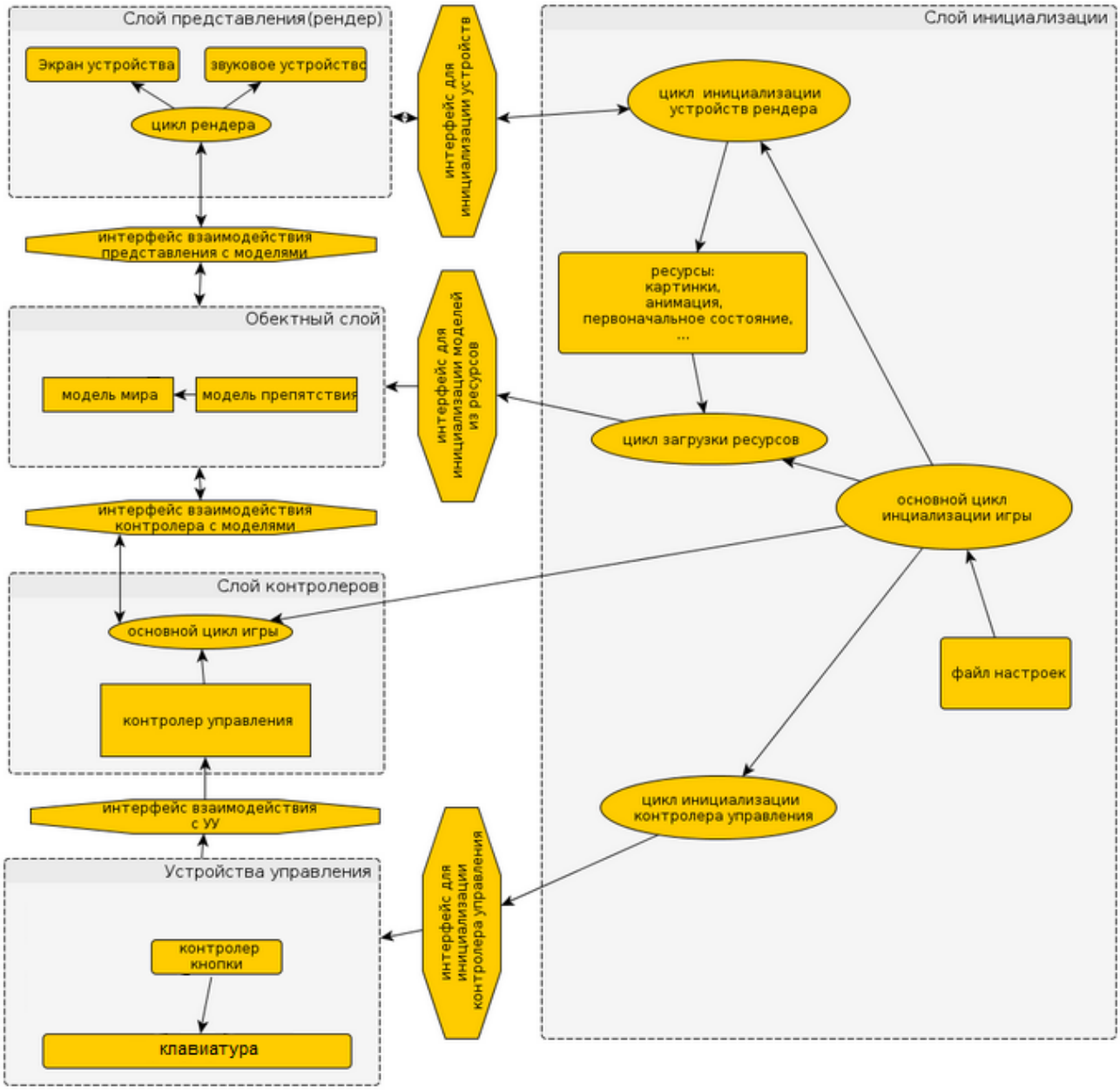

Рисунок 2.18 - Архитектура игрового приложения

Таким образом, в данном разделе было рассмотрено использование ООП при разработке программного продукта, и было показано, как устроена архитектура игрового движка.

2.8 Функциональная модель приложения

Перед тем, как создавать программу нужно продумать функциональную модель и логику приложения. Построим схему, изображающую функциональные части приложения, участвующие в процессе и свяжем эти части между собой. Рассмотрим ниже рисунок 2.19 схема работы системы.

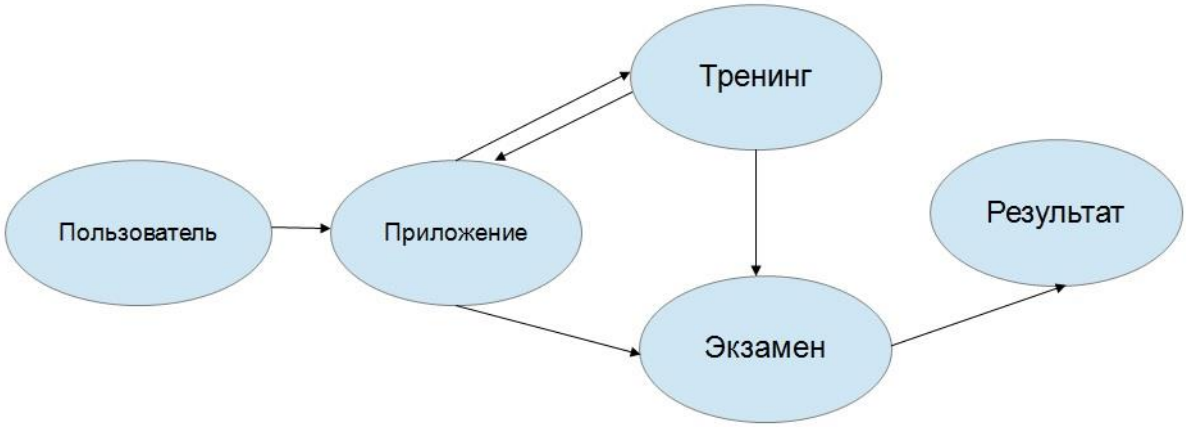

Рисунок 2.19 – Функциональная модель обучающей компьютерной игры

Функциональная модель будет иметь следующие модули:

1. Пользователь, обращающийся к приложению;
2. Заполнение данных о пользователе.
3. Прохождения тренинга.
4. Прохождение экзамена.
5. Формирование результата экзамена.

Описание этапов использования приложения.

На первом этапе пользователь обращается к приложению.

На втором этапе пользователь заполняет Ф.И.О. и выбирает режим игры, тестирование или экзамен.

На третьем этапе пользователь переходит в этап тестирования.

На четвертом этапе пользователь проходит экзамены.

На заключительном пятом этапе происходит формирование результатов прохождения экзамена.

Диаграмма прецедентов представлена на рисунке 2.20.

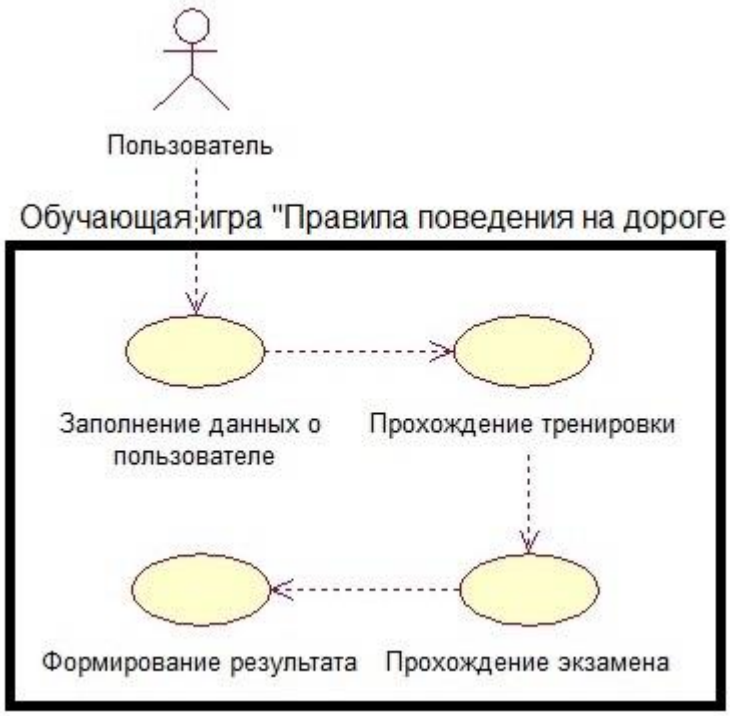

Рисунок 2.20 - Диаграмма прецедентов

И так, была представлена функциональная модель, где были выделены модули, которые были реализованы в конечном продукте. Также была представлена диаграмма прецедентов, показывающая основные этапы использования приложения.

Во второй главе был проведен анализ данных и алгоритмов, также были описаны логические и физические модели. Была описана архитектура системы и иерархии классов. И также была приведена функциональная модель приложения.

ГЛАВА 3 ТЕСТИРОВАНИЕ И РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

3.1 Тестирование реализованной программы

Тестирование программного обеспечения — процесс изучения, проверки программного продукта, который имеет две основные цели:

- убедиться, что разработанная программа соответствует требованиям;
- обнаружить ситуации, в которых работа программы не является правильным, или нежелательным.

Существует несколько уровней тестирования программного обеспечения:

- модульное тестирование — для теста берется минимально возможный для тестирования компонент, например, отдельный класс или функция;
- интеграционное тестирование — тестируются интерфейсы между компонентами, подсистемами или системами;
- системное тестирование — тестируется интегрированная система на ее соответствие требованиям;
- альфа-тестирование — имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком;
- Бета-тестирование — в некоторых случаях выполняется распространение предварительной версии для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок.

Компьютерная обучающая игра прошла все стадии тестирования на различных версиях операционных систем семейства Microsoft Windows (XP, 7, 8, 10) с разными конфигурациями оборудования (чипсет, производитель процессора, модель процессора, количество ядер, тактовая частота, размер кеша разных уровней, разрядность, объем оперативной и видео памяти и т.д.). Системные требования к программе удовлетворяют заявленным в техническом задании. При тестировании найденные ошибки были исправлены, большинство алгоритмов

было оптимизировано, что позволило увеличить производительность работы приложения и его стабильность.

Для тестирования всех компонентов использовалась утилита под названием Unity Test Tools.

Unity Test Tools – программа для тестирования разных элементов в разрабатываемой игре/программе, созданная компанией Unity Technologies.

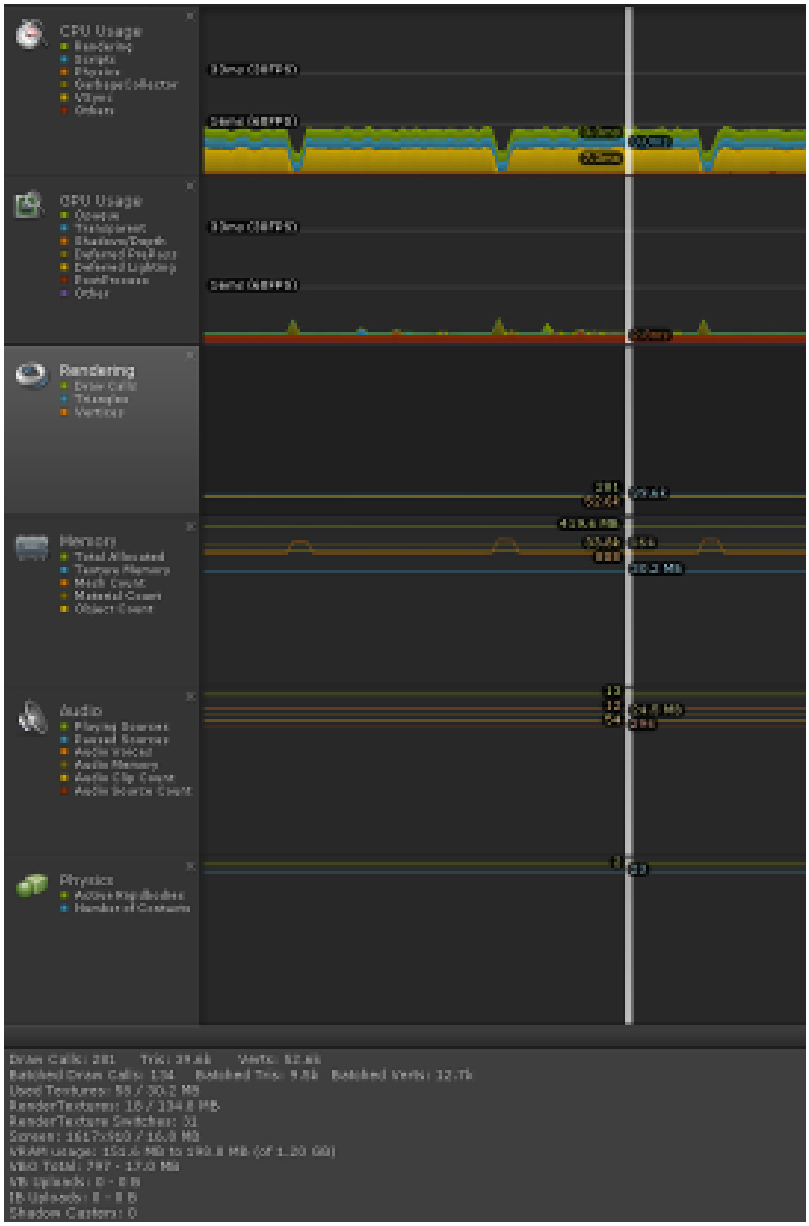

Рисунок 3.1 – Статистика ресурсопотребления ЭВМ при работе системы

На рисунке 3.1 представлена информация о требуемых ресурсах для работы системы:

- CPU Usage – использование Центрального Процессора;
- GPU Usage – использование Графического Процессора;
- Rendering – построение кадров, выводимых на монитор компьютера;
- Memory – использование Оперативной Памяти;
- Audio – использование аудио подсистемы компьютера;
- Physics – обработка физики;
- дополнительная информация (количество обращений к видеокарте, количество полигонов на сцене, разрешение экрана и т.д.).

В данном разделе было произведено тестирование всех элементов компьютерной обучающей игры.

3.2 Руководство пользователя по работе с обучающей системой

1. Общее назначение программы.

Программа является продуктом сферы компьютерных обучающих игр. Игра относится к жанру симуляторы. Симулятор – один из самых популярных жанров компьютерных игр. Возрастные ограничения для данной игры не установлены, и она может служить отличным способом времяпрепровождения для всех.

2. Установка, запуск, минимальные требования и состав программы.

Для полного функционирования программного продукта необходимо установить все необходимое программное обеспечение. Все необходимое ПО является бесплатным и легкодоступным, за исключением операционной системы. Для функционирования программы требуется установить следующие компоненты:

- Microsoft DirectX версии 9.0c или выше;
- Microsoft .Net Framework версии 4.0;
- Драйвера аудио устройств;

- Драйвера видео устройства.
- В состав программы входят:
- RoadRules.exe - исполняемый файл программы;
- RoadRules - каталог ресурсов программы.

Для запуска программы «Правила поведения на дорогах» требуется запустить исполняемый файл «RoadRules.exe». На рисунке 3.2 изображено окно запуска приложения, в котором пользователь может настроить следующие параметры:

- разрешение экрана;
- оконный или полноэкранный режим;
- качество графики.
- монитор

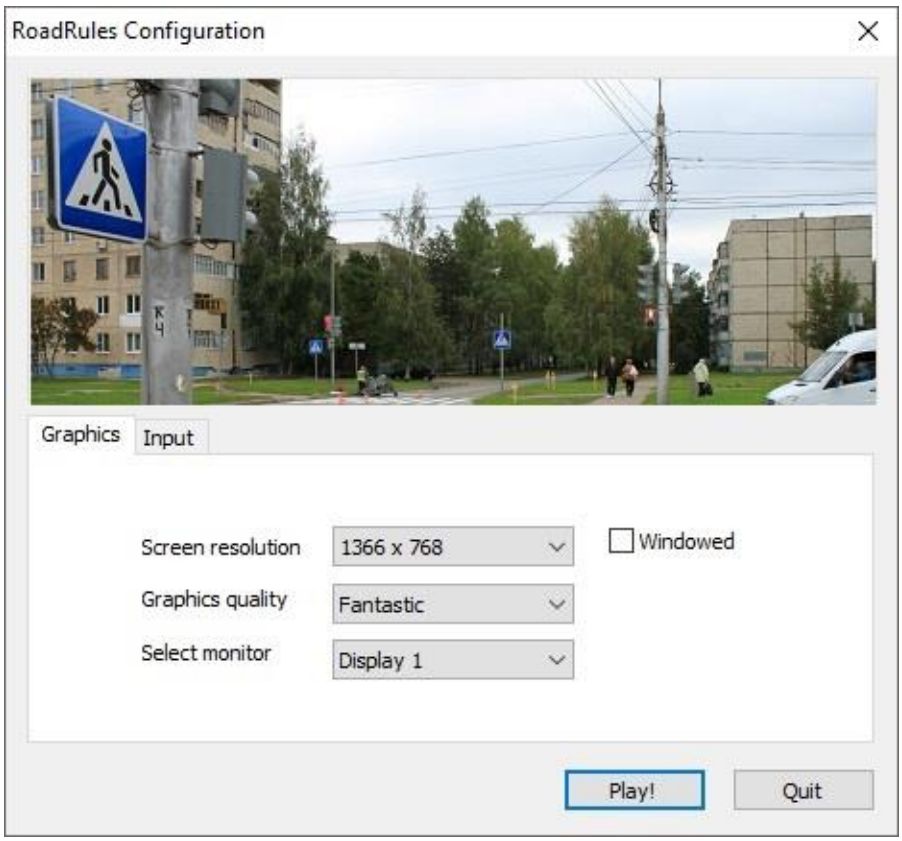

Рисунок 3.2 – Окно запуска приложения

После выбора оптимальных настроек необходимо нажать кнопку «Play!». После загрузки всех необходимых ресурсов в постоянную память компьютера приложение будет запущено. На рисунке 3.3 изображено главное меню приложения.

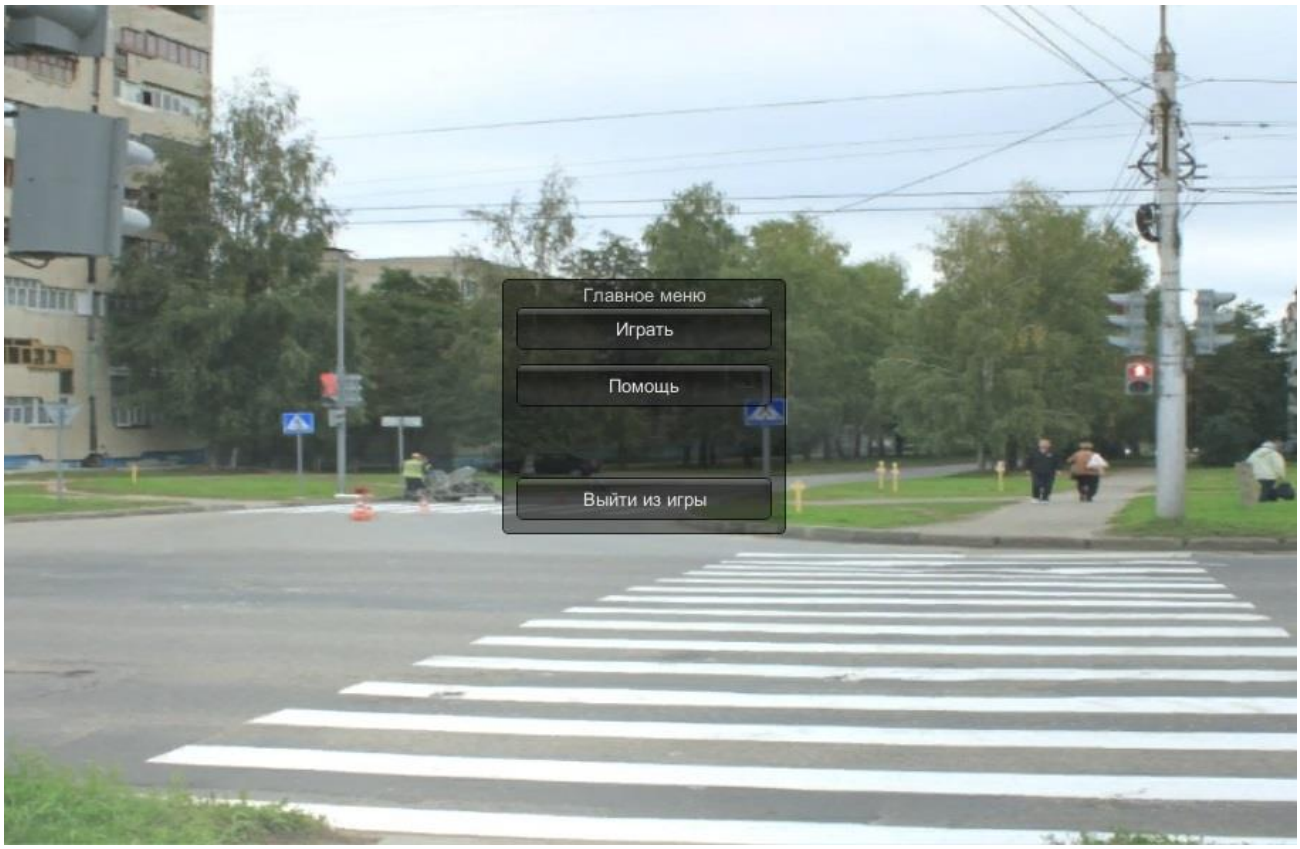

Рисунок 3.3 – Главное меню

Из главного меню пользователь может перейти к следующим пунктам:

- «Играть» – позволяет запустить игровой процесс;
- «Помощь» – меню помощи;
- «Выход» – выход из игры.

При переходе к игровому процессу приложение загрузит необходимые ресурсы и запустит уровень. Далее появляется игровой процесс. На рисунке 3.4

изображен процесс игры, в котором перед пользователем стоит задание пройти через улицу по пешеходной дороге.

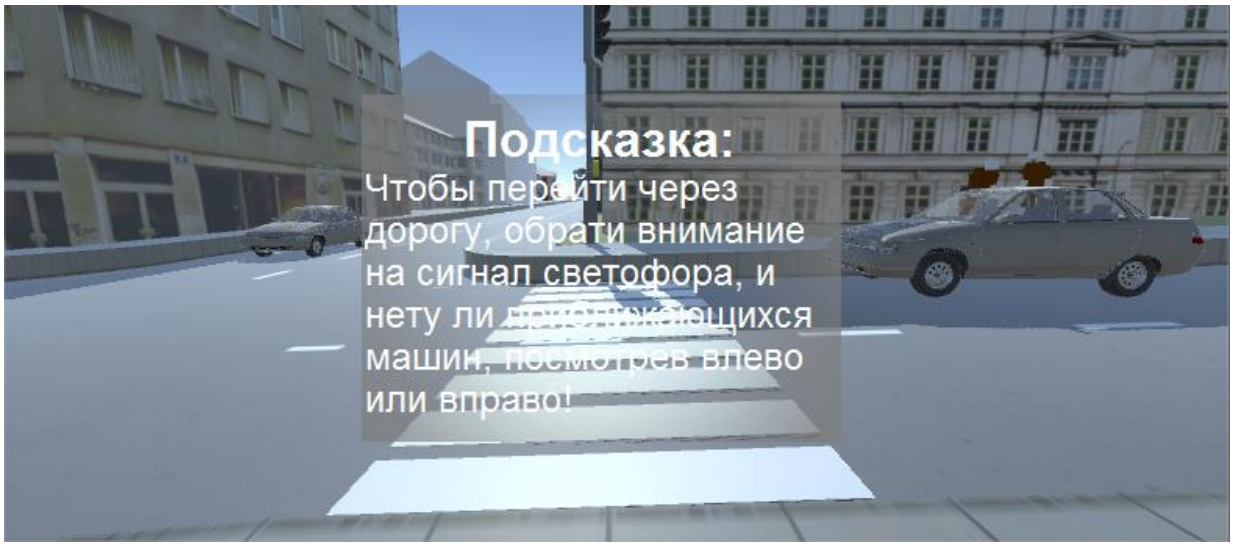

Рисунок 3.4 – Игровой процесс

После прохождения поставленного задания, на экран выдаётся сообщение о выполнении задания.

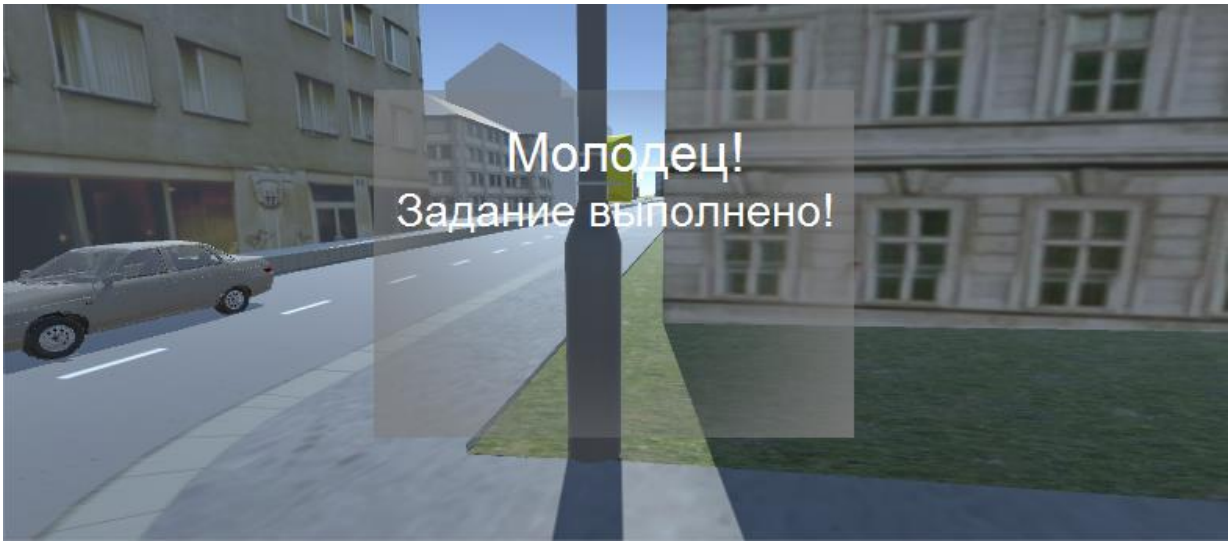

Рисунок 3.5 – Выполненное задание

Минимальные системные требования:

- Операционная система: Microsoft Windows XP/Vista/7/8;
- Процессор: Intel Core 2 Duo @ 1.8 Ghz / AMD Athlon64 X2 4000+;

- Оперативная память: 1 Gb;
- Жесткий диск: 10 Gb свободно;
- Видео память: 512 Mb;
- Видео карта: nVidia GeForce 8600 / AMD Radeon HD 4570;
- Звуковая карта: Совместимая с DirectX;
- Поддержка Microsoft DirectX 9.0c;
- Клавиатура, Мышь.

Рекомендуемые системные требования:

- Операционная система: Windows Vista/7/8/10;
- Процессор: Intel Core i3 @ 3.0 GHz / AMD Athlon64 X4 @ 3.6 GHz;
- Оперативная память: 2 Gb;
- Жесткий диск: 10 Gb свободно;
- Видео память: 1 Gb;
- Видео карта: nVidia GeForce GTX 450 / AMD Radeon HD 5850;
- Звуковая карта: Совместимая с DirectX;
- DirectX 11;
- Клавиатура, Мышь.

3. Элементы интерфейса и управление.

Пользователь имеет возможность играть от разных камер:

- Управление от первого лица;
- Управление от третьего лица (рисунок 3.5).

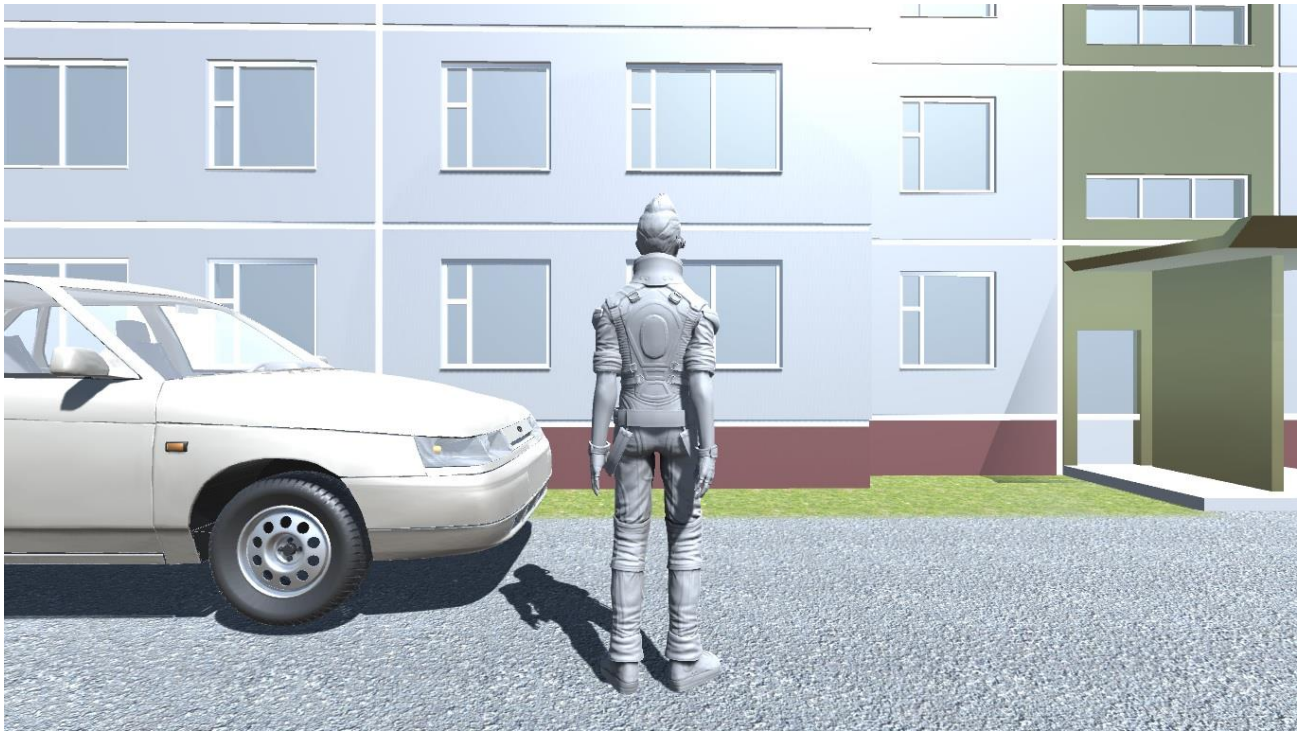

Рисунок 3.6 – Управление игроком от третьего лица

Управление игровым персонажем выполняется с помощью компьютерной мыши и клавиатуры.

Компьютерная мышь позволяет изменять ракурс камеры, следящей за игроком, и указать точку назначения движения персонажа по игровому уровню.

С помощью клавиатуры игрок совершает основные действия:

- Клавиша «W» – идти вперед;
- Клавиша «S» – идти назад;
- Клавиша «A» – идти влево;
- Клавиша «D» – идти вправо;
- Клавиши стрелок или движение мышью – изменение угла обзора камеры.

Задача игрока пройти весь уровень, выполняя условия тренажера.

Таким образом, проведя тестирование, были определены минимальные и рекомендуемые системные требования.

Вывод по главе 3

В результате проведения тестирования было показано, как программный продукт нагружает отдельные ресурсы ЭВМ. Также в главе было написано руководство пользователя, в котором указано:

как запустить программу;

минимальные системные требования;

рекомендуемые системные требования;

способ управления.

ЗАКЛЮЧЕНИЕ

Результатом выпускной квалификационной работы является готовый программный продукт «Трехмерная обучающая игра для школьников “Правила поведения на дорогах”». Реализованы подсистемы поиска пути, графического интерфейса пользователя и взаимодействия внутриигровых объектов. Все подсистемы отлажены, оптимизированы, протестированы и интегрированы в единую программную систему. В ходе тестирования и внедрения выяснилось, что разработанный продукт так же является отличным основанием для внедрения мотивационных программ. Основными достоинствами разработанного продукта является его простота и надежность, а также возможность быстрой доработки и модификации, по заданным критериям. Полученное в итоге приложение полностью удовлетворило все требования заказчика. Отдельно стоит отметить, что благодаря простой и универсальной структуре, полученный продукт имеет большой потенциал, что не исключает возможности расширения функционала и дальнейшего внедрения в других предприятиях.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Бабенко Е.В., Аноприенко А.Я. Организация модульного интерактивного приложения для трехмерного моделирования угольных шахт// Материалы III всеукраинской научно-технической конференции студентов, аспирантов и молодых ученых «Информационные управляющие системы и компьютерный мониторинг (ИУС и КМ 2012)» – 17-18 апреля 2012 г., Донецк, ДонНТУ, 2012. Т.3. С. 680-684.

2. Аноприенко А.Я., Бабенко Е.В. Навка Е.А. Оверчик О.М. Трехмерное интерактивное моделирование угольной шахты на базе системы «Blender» // Материалы четвертой международной научно-технической конференции «Моделирование и комьпьютерная графика» 5-8 октября 2011 года, Донецк, ДонНТУ, 2011. С. 279-285

3. Бабенко Е.В. Навка Е.А. Оверчик О.М. Перспективы использования свободного программного обеспечения для создания трехмерных интерактивных приложений // Материалы II всеукраинской научно-технической конференции студентов, аспирантов и молодых ученых «Информационные управляющие системы и компьютерный мониторинг (ИУС и КМ 2011)» – 12-13 апреля 2011 г., Донецк, ДонНТУ, 2011. Т.3. С. 184-187.

4. Трофимов В.А., доц. Николаев Е.Б., доц., Аноприенко А.Я., проф., Бабенко Е.В., Оверчик О.М., ст. гр. КЭМ-11м (ДонНТУ). Использование трехмерного интерактивного моделирования угольной шахты для создания тренажера по безопасности и охране труда. Материалы всеукраинской научно-технической конференции молодых ученых, студентов и аспирантов "Современные проблемы охраны труда и аэрологии горных предприятий"– 24 ноября 2011 г., Донецк, ДонНТУ, 2011.

5. Иванов В. П., Батраков А. С. Трехмерная компьютерная графика/Иванов В – М.: Издательский дом «Радио и связ», 1995. – 320 с.

6. Залогова Л.А. Компьютерная графика. Элективный курс. Практикум /Залогова Л.А – М.: БИНОМ Лаборатория знаний, 2005. – 245 с.

7. Энджел Э. Интерактивная компьютерная графика. Вводный курс на базе OpenGL/Энджел Э. – Вильямс, 2001. – 592 рр.

8. Шикин Е.В., Боресков А.В. Компьютерная графика. Полигональные модели / Шикин Е.В., Боресков А.В. – Диалог-МИФ, 2005. – 464 с.

9. Херн Д., Бейкер П. Компьютерная графика и стандарт OpenGL / Херн Д. – Вильямс, 2005. – 1168 с.

10. Хейфец А. Л., Логиновский А. Н., Буторина И. В., Васильева В. Н. Инженерная 3D компьютерная графика / Хейфец А. Л., Логиновский А. Н., Буторина И. В., Васильева В. Н. – Юрайт, 2011. – 516 с.

11. Рик Пэрент Компьютерная анимация/ Пэрент Р. – КУДИЦ-Образ, 2004. – 579 с.

12. Зеньковский В. А. 3D-эффекты при создании презентаций, сайтов и рекламных видеороликов /Зеньковский В. А. – БХВ-Петербург, 2011. – 650 с.

13. Unity 3D-Википедия [текст]/ интернет – ресурс. – режим доступа: www/URL: http://ru.wikipedia.org/wiki/Unity_.

14. Создание игр, Игровые движки, Конструкторы игр - Разработка игр- интернет – ресурс. – режим доступа: http://gcup.ru/.

15. Создание игр без программирования- интернет – ресурс. – режим доступа:http://make-games.ru/.

16. Unity3D по-русски – ресурс. – режим доступа: http://www.unity3d.ru/.

17. Конструкторы для создания компьютерных игр. Выпуск 3 – режим доступа: http://itcs.3dn.ru.

18. Создаем компьютерную игру. Создание игр на C++/DirectX– режим доступа: http://shatalov.su/.

19. Unity Documentation. Unity Manual – режим доступа: http://docs.unity3d.com/Manual/index.html

20. Jeremy Gibson, Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C#. –2014. – 944c.

21. Will Goldstone, Unity Game Development Essentials. –2009. – 316c.

22. Joe Hocking, Unity in Action: Multiplatform Game Development in C#. – 2015. – 352c.

23. Jonathan Weinberger. Learn Unity Programming with C#. –Technology in action, 2015. – 686c.

ПРИЛОЖЕНИЕ А

Скрипт главного меню

```
using UnityEngine;
using System.Collections;

public class menu : MonoBehaviour {

private int _window = 0;

void  OnGUI (){

if (_window == 0) {
GUI.Box ( new Rect(Screen.width/2 - 100,Screen.height/2 - 100,200,180), "Главное меню");
if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 80,180,30), "Играть")) {
                        _window = 3;
        }
if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 40,180,30), "Помощь")) {
                _window = 2;
        }
if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 + 40,180,30), "Выйти из игры")) {
                Application.Quit();
        }
}
if (_window == 2) {
GUI.Box ( new Rect(Screen.width/2 - 100,Screen.height/2 - 100,200,180), "Помощь");
GUI.Label ( new Rect(Screen.width/2 - 100,Screen.height/2 - 80,180,140), "Developed by Dmitriy Savkin.");
        if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 + 40,180,30), "Назад") || Input.GetKey ("escape")) {
        _window = 0;
```

```
            }
        }

            if (_window == 3) {
                GUI.Box (new Rect (Screen.width / 2 - 100, Screen.height / 2 - 100, 200, 180),
"Выберите режим игры");

                GUI.Button (new Rect (Screen.width / 2 - 90, Screen.height / 2 - 80, 180, 30),
"Тренинг");
                    //{
                    //      Application.LoadLevel (1);
                    //}
                GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 40,180,30),
"Экзамен");
                //{
                //      Application.LoadLevel (1);
                //      }
                if (GUI.Button (new Rect (Screen.width / 2 - 90, Screen.height / 2 + 40, 180, 30),
"Назад") || Input.GetKey ("escape")) {
                    _window = 0;
                }
            }
        }
```

ПРИЛОЖЕНИЕ Б

Скрипт меню паузы

```
using UnityEngine;
using System.Collections;

public class Pause : MonoBehaviour {



private bool _paused = false;

private int _window = 100;

private float _FloatVolume = 50;
private int IntVolume;

private float _FloatResolution = 2;
private int IntResolution;

private string StringWidth;
private string StringHeight;

private int width = 1600;
private int height = 900;
private bool FullScreen = true;

private bool Mouse;
private bool Keyboard;
public GameObject go;
```

```
	void Update () {


	if(Input.GetKeyUp(KeyCode.Escape)){

		if(!_paused){
			Time.timeScale = 0;
			_paused = true;
			_window = 0;
			}
		else{
			Time.timeScale = 1;
			_paused = false;
			_window = 100;
			}
		}
	}

	void  OnGUI (){

		if (_window == 0) { // Главное меню активировано при _window = 0
			GUI.Box ( new Rect(Screen.width/2 - 100,Screen.height/2 - 100,200,180), "Игровое меню");

		if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 80,180,30), "Продолжить")) {
			Time.timeScale = 1;
				_paused=false;
				_window = 100;
		}
		if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 40,180,30), "Опции"))
{
```

```
            _window = 1;
        }
            if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 0,180,30), "Главное меню")) {
                    Time.timeScale = 1;
                    _paused = false;
                    _window = 100;
            Application.LoadLevel (0);
        }
            if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 + 40,180,30), "Выйти из игры")) {
            Application.Quit();
        }
            }

            // Настройки
            if (_window == 1) {
                GUI.Box ( new Rect(Screen.width/2 - 100,Screen.height/2 - 100,200,180), "Опции");

                if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 80,180,30), "Видео")) { // Видео
                _window = 3;
            }
                if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 40,180,30), "Аудио")) { // Звук
            _window = 2;
        }
                if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 - 0,180,30), "Управление")) { // Управление
                    _window = 4;
        }
```

```
                if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 + 40,180,30), "Назад") || Input.GetKeyUp(KeyCode.Escape)) {
                _window = 0;
            }
            }
            // Звук
            if (_window == 2) {
                GUI.Box ( new Rect(Screen.width/2 - 100,Screen.height/2 - 100,200,180), "Аудио");
                GUI.Label ( new Rect(Screen.width/2 - 100,Screen.height/2 - 80,180,140), "Громкость:" ); // текст

                _FloatVolume = GUI.HorizontalSlider ( new Rect(Screen.width/2+50 - 90,Screen.height/2+6 - 80, 100, 20), _FloatVolume, 0, 100);
                IntVolume = (int)_FloatVolume;
                //audio.volume = IntVolume;

                if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 + 40,180,30), "Назад") || Input.GetKeyUp(KeyCode.Escape)) {
            _window = 1;
            }
            }
            // Видео
            if (_window == 3) {
                GUI.Box ( new Rect(Screen.width/2 - 100,Screen.height/2 - 100,200,180), "Видео");
                GUI.Label ( new Rect(Screen.width/2 - 90,Screen.height/2 - 80,180,30), "Разрешение:"); // текст

                _FloatResolution = GUI.HorizontalSlider ( new Rect(Screen.width/2 - 20,Screen.height/2 - 75,100,30), _FloatResolution, 0, 2);
                // Расчеты расширения
```

```
IntResolution = (int)_FloatResolution;
if (IntResolution == 0){
	width = 640;
	height = 480;
	StringWidth = width.ToString();
	StringHeight = height.ToString();
}
if (IntResolution == 1){
	width = 1024;
	height = 768;
	StringWidth = width.ToString();
	StringHeight = height.ToString();
}
if (IntResolution == 2){
	width = 1600;
	height = 900;
	StringWidth = width.ToString();
	StringHeight = height.ToString();
}
// Вывод на экран выбираемого расширения
GUI.Label ( new Rect(Screen.width/2 - 90,Screen.height/2 - 40,180,30), StringWidth); // ширина
GUI.Label ( new Rect(Screen.width/2 - 50,Screen.height/2 - 40,180,30), StringHeight); // высота

FullScreen = GUI.Toggle ( new Rect(Screen.width/2 - 90,Screen.height/2 - 0,180,30), FullScreen, "Полный экран");
//if (FullScreen == true) {}

if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 + 40,180,30), "Сохранить и вернуться")) {
```

```
                Screen.SetResolution (width, height, FullScreen);//A - ширина. B - высота. C - полноэкранный или оконный.
                _window = 1;
        }
                if (Input.GetKeyUp(KeyCode.Escape)) {
                _window = 1;
        }
        }
        if (_window == 4){
                GUI.Box ( new Rect(Screen.width/2 - 100,Screen.height/2 - 100,200,180), "Тип управления");
                Mouse = GUI.Toggle ( new Rect(Screen.width/2 - 90,Screen.height/2 - 80,180,30), Mouse, "Мышь");
                if (Mouse)
                        Keyboard = false;
                Keyboard = GUI.Toggle ( new Rect(Screen.width/2 - 90,Screen.height/2 - 40,180,30), Keyboard, "Клавиатура");
                if (Keyboard)
                        Mouse = false;
                if (!Mouse && !Keyboard)
                        Mouse = true;
                if (GUI.Button ( new Rect(Screen.width/2 - 90,Screen.height/2 + 40,180,30), "Сохранить и вернуться")) {
                        // Присвоение скриптам переменных
                        MoveMousePlayer mp = (MoveMousePlayer) go.GetComponent<MoveMousePlayer>();
                        MoveKeyboardPlayer kp = (MoveKeyboardPlayer) go.GetComponent<MoveKeyboardPlayer>();
                        // Проверка и перерасчет изменений
                        if(Mouse && mp == enabled){
                                mp.enabled = enabled;
                                kp.enabled = !enabled;
```

```
                }
                        if(Keyboard && kp == enabled){
                                kp.enabled = enabled;
                                mp.enabled = !enabled;
                }
                _window = 1;
                }
                        if (Input.GetKeyUp(KeyCode.Escape))
                                _window = 1;
                }
        }
}
```

ПРИЛОЖЕНИЕ В

Скрипт искусственного интеллекта

```
using UnityEngine;
using System.Collections;

public class AI : MonoBehaviour {

	public string Action = "idle";
	public string Walk = "walk";
	public string Run = "run";

	public float speed =0.04f;
	public float damping = 6;
	public GameObject ragdoll;

	public GameObject[] possibleNodes;
	public GameObject curNode;

	public GameObject preNode;
	private GameObject newNode;

	public bool doesActions = false;
	public bool dontBackTrack = false;
	public bool doesAnimations = true;

	void Awake(){
		curNode = FindStartNode(); //Поиск ближайшего узла, как начальная точка.

	}
```

```
public void Update(){


        if(doesAnimations == true){
            if(speed == 0){
                if(Action != null){
                    GetComponent<Animation>().Play(Action);      //Проигрывается анимация, зависящая от скорости движения ИИ
                }
            }else if(speed == 2f){
                if(Walk != null){
                    GetComponent<Animation>().Play(Walk);
                }
            }else if(speed >= 6f){
                if(Run != null){
                    GetComponent<Animation>().Play(Run);
                }
            }
        }


        var rotation = Quaternion.LookRotation(curNode.transform.position - transform.position);
        transform.rotation = Quaternion.Slerp(transform.rotation, rotation, Time.deltaTime * damping);

        if(Vector3.Distance(curNode.transform.position,transform.position)>1.5)
        {
```

```
                transform.Translate(Vector3.forward*speed*  Time.deltaTime);  //Если  ИИ
находится дальше, чем указано, то он продолжаеть двигаться в направлении curNode

            }else{
                if(doesActions == true){
                    if(curNode.GetComponent<Waypoint>().actionNode == true){
                        this.StartCoroutine(this.DoAction()); //Если ИИ делает действия,
то этот узел называется DoAction
                    }
                }

                if(curNode.GetComponent<Waypoint>().trafficNode  ==  true){  //этот  узел
смотрит дорожное движение
                    if(curNode.GetComponent<Waypoint>().redLight  ==  true){  //  это
определение красного цвета светофора
                        this.StartCoroutine(this.DoAction()); // в этот момент DoAction
означает ожидание
                    }
                }

                curNode = FindNewNode();//В противном случае выбирается другой узел
                newNode = null;
            }
            if (curNode == null) return;
        }

        public GameObject FindNewNode(){
            int ex = 10;
            possibleNodes  =  curNode.GetComponent<Waypoint>().possibleNodes;//находит
соседей текущего узла

            while(newNode == null && ex--> 0){
```

```
                    newNode = possibleNodes[Random.Range(0, possibleNodes.Length)];//Выбор одного из соседей curNodes, чтобы перейти к нему
                    if(dontBackTrack == true){
                        if(newNode == preNode){
                            newNode = null; //Если dontBackTrack активен и ИИ зарашивает отслеживание поиска нового узла, то он не будет этого делать
                        }
                    }

            }
            preNode = curNode;
            return newNode; //возвращаем новый узел, к которому мы переходим
        }

        public GameObject FindStartNode(){ //поиск всех узлов и нахождение самого близкого к ИИ
            var closest = gameObject;
            GameObject[] gos;
        gos = GameObject.FindGameObjectsWithTag("Node");
        float distance = Mathf.Infinity;
        Vector3 position = transform.position;
        foreach (GameObject go in gos) {
          Vector3 diff = go.transform.position - position;
            float curDistance = diff.sqrMagnitude;
          if (curDistance < distance) {
            closest = go;
            distance = curDistance;
                    }
            }

            return closest;
        }
```

```
    IEnumerator DoAction() {
            speed = 0; //Проигрывание анимации
    yield return new WaitForSeconds(Random.Range(3,12));//ожидание рандомного времени
    speed = 2f;//продолжение походки
    }


}
```

ПРИЛОЖЕНИЕ Г

Скрипт определения пути

```
using UnityEngine;
using System.Collections;

public class Waypoint : MonoBehaviour {

	public GameObject[] possibleNodes;

	public float searchRange = 10; // Как далеко будет луч в поиске осталньных узлов

	public bool manual = false;
	public bool actionNode = false;

	public bool VertNode = false;
	public bool trafficNode = false;
	public GameObject trafficLight;
	public bool redLight = false;

	public bool noCollisions = true; // твёрдые ли путевые точки

	void TrafficLightCheck(){
		if(trafficLight.GetComponent<TrafficLight>().redLight == true){
			redLight = true;
		}
		if(trafficLight.GetComponent<TrafficLight>().redLight == false){
			redLight = false;
		}

	}
```

```
    void Awake(){
        gameObject.GetComponent<Renderer>().enabled = false;

        if(trafficNode == true){
            InvokeRepeating("TrafficLightCheck", 1, 5);
        }


        if(manual == false){
            var fwd = transform.TransformDirection (Vector3.forward);//определение где есть перед, зад, лево, право
            var right = transform.TransformDirection (Vector3.right);
            var up = transform.TransformDirection (Vector3.up);


            RaycastHit hit = new RaycastHit();

            //Проверка спереди
            if (Physics.Raycast (transform.position, fwd, out hit, searchRange)) {
                    if(hit.transform.gameObject.tag == "Node"){
                        Debug.DrawLine(transform.position, hit.point, Color.red);
                        possibleNodes[0] = hit.transform.gameObject;

                    }
            }

            //Проверка сзади
            if (Physics.Raycast (transform.position, -fwd, out hit, searchRange)) {
                    if(hit.transform.gameObject.tag == "Node"){
                        Debug.DrawLine(transform.position, hit.point, Color.red);
```

```
                possibleNodes[1] = hit.transform.gameObject;

            }
        }

        //Проверка справа
        if (Physics.Raycast (transform.position, right, out hit, searchRange)) {
            if(hit.transform.gameObject.tag == "Node"){
                Debug.DrawLine(transform.position, hit.point, Color.red);
                possibleNodes[2] = hit.transform.gameObject;

            }
        }

        //Проверка слева
        if (Physics.Raycast (transform.position, -right, out hit, searchRange)) {
            if(hit.transform.gameObject.tag == "Node"){
                Debug.DrawLine(transform.position, hit.point, Color.red);
                possibleNodes[3] = hit.transform.gameObject;

            }
        }

        //проверка вверху
        if(VertNode == true){
            if (Physics.Raycast (transform.position, up, out hit, searchRange)) {
                if(hit.transform.gameObject.tag == "Node"){
                    Debug.DrawLine(transform.position, hit.point, Color.red);
                    possibleNodes[4] = hit.transform.gameObject;

                }
            }
```

```
                        //проверка внизу
                        if (Physics.Raycast (transform.position, -up, out hit, searchRange)) {
                                if(hit.transform.gameObject.tag == "Node"){
                                        Debug.DrawLine(transform.position, hit.point, Color.red);
                                        possibleNodes[5] = hit.transform.gameObject;

                                }
                        }









                }

        }
        if(noCollisions == false){
        GetComponent<Rigidbody>().detectCollisions = false;
        }
    }

}
```

ПРИЛОЖЕНИЕ Д

Скрипт работы светофора

```
using UnityEngine;
using System.Collections;

public class TrafficLight : MonoBehaviour {

    public Color color0 = Color.red;
    public Color color1 = Color.green;

  public float Timer;
  public float coolDown;
  public bool redLight = true;

  void Awake(){
      GetComponent<Light>().color = color0;//красный свет
      Timer = 7;// свет меняется каждые 7 секунд
      }

    void Update() {

      if(Timer > 0)
          Timer -= Time.deltaTime;

      if(Timer < 0)
          Timer = 0;

      if(Timer == 0)// свет должен быть сменён
          changeColour();
    }
```

```
public void changeColour(){//смена цвета светофора
        if(redLight == true){
                GetComponent<Light>().color = color1;
                redLight = false;
        }else{
                GetComponent<Light>().color = color0;
                redLight = true;
        }
        Timer = 10;
}
}
```

ПРИЛОЖЕНИЕ Е

Класс Vector3

```
const float EPSILON = 0.001f;

class vector3
{
public:

	union
	{
		struct
		{
			float x,y,z;
		};
		float v[3];
	};

	vector3() : x(0), y(0), z(0) {}
	vector3(vector3& v) : x(v.x), y(v.y), z(v.z) {}
	vector3(float _x, float _y, float _z) : x(_x),y(_y),z(_z){}

	// присваиваение векторов
	vector3& operator= (vector3& v)
	{
		x = v.x;
		y = v.y;
		z = v.z;
		return *this;
	}

	// сравнение векторов
	bool operator== (vector3& v)
	{
		if (fabs(x-v.x) < EPSILON)
			if (fabs(y-v.y) < EPSILON)
				if (fabs(z-v.z) < EPSILON)
					return true;
		return false;
	}

	// величина вектора (от magnitude - величина)
	float mag()
	{
		return sqrt(x*x+y*y+z*z);
	}
```

```
// отрицательный вектор (унарная операция!!!)
vector3 operator- ()
{
	return vector3(-x,-y,-z);
}

// сложение векторов
vector3 operator+ (vector3& v)
{
	return vector3(x+v.x,y+v.y,z+v.z);
}

// разность векторов
vector3 operator- (vector3& v)
{
	return vector3(x-v.x,y-v.y,z-v.z);
}

// умножение вектора на скаляр
vector3 operator* (float& a)
{
	return vector3(x*a,y*a,z*a);
}

// деление вектора на скаляр
vector3 operator/ (float& a)
{
	float b = 1.0f / a;
	return vector3 (x*b,y*b,z*b);
}

// скалярное произедение векторов (dot product)
float operator* (vector3& v)
{
	return x*v.x+y*v.y+z*v.z;
}

// векторное произведение векторов (cross product)
vector3 cross(vector3& v)
{
	return vector3(y*v.z - z*v.y, z*v.x - x*v.z, x*v.y - y*v.x);
}

vector3& operator+= (vector3& v)
{
	x += v.x; y += v.y; z += v.z;
	return *this;
```

```
	}

	vector3& operator-= (vector3& v)
	{
		x -= v.x; y -= v.y; z -= v.z;
		return *this;
	}

	vector3& operator*= (float& a)
	{
		x *= a; y *= a; z *= a;
		return *this;
	}

	vector3& operator/= (float& a)
	{
		float b = 1.0f/a;
		x *= b; y *= b; z *= b;
		return *this;
	}

	// нормализация вектора
	void normalize ()
	{
		float magnitude = mag();
		if (magnitude > 0)
		{
			float invertedMag = 1 / magnitude;
			x *= invertedMag;
			y *= invertedMag;
			z *= invertedMag;
		}
	}

	// расстояние между двумя точками (не векторами!!!)
	float distance (vector3& v)
	{
		float dx = x - v.x;
		float dy = y - v.y;
		float dz = z - v.z;
		return sqrt(dx*dx + dy*dy + dz*dz);
	}
};
```