

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального
назначения»

Специализация: «Эксплуатация вычислительных машин, комплексов,
систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

Мобильное игровое приложение

Студент Тупичак 14.06.22 Тупичак Дмитрий Александрович
(подпись, дата) (ФИО полностью)

Руководитель Бикташев 14.06.22 Бикташев Р. А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр Бычков 14.06.22 Бычков А. С.
(подпись, дата) (фамилия, инициалы)

Рецензент Печерский 14.06.22 Печерский С. В.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2022 г. № 13)

Заведующий кафедрой
(подпись)

(фамилия, инициалы)

Митрохин М.А.

Работа защищена с отметкой удовлетворительно (протокол заседания ГЭК от

24 июня 2022 № 16)

Секретарь ГЭК

Гурин Е.И.

Пенза 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

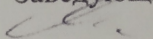
вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

«26» 04 2022 г.

**ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ**

1. Студент Тупичак Дмитрий Александрович гр. 17BO2 факультета ВТ

2. Тема работы Создание игрового мобильного приложения

Тема утверждена приказом ПГУ №357/о от "26" апреля 2022 г.

3. Руководитель работы Бикташев Р.А.

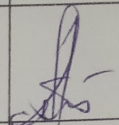
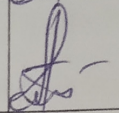
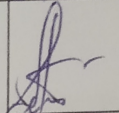
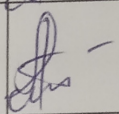
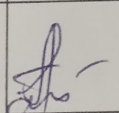
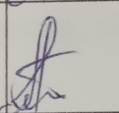
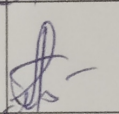
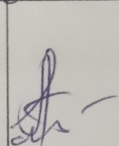
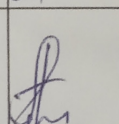
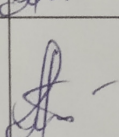
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Приложение представляет собой игровое программное обеспечение, работающее на устройствах с операционной системой Android.

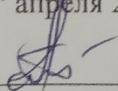
5. Перечень подлежащих разработке вопросов

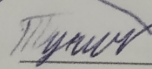
Требуется произвести анализ операционной системы Android, определить функциональный состав системы, осуществить выбор инструментов для разработки игрового мобильного приложения, функционирующего на основе операционной системы Android, определить алгоритмы работы создания мобильного приложения и его разработку. В ходе дипломного проектирования требуется выполнить задание по безопасности жизнедеятельности и произвести расчет экономической целесообразности проект

6. Календарный график выполнения работы

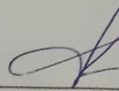
№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	10	26.04.2022- 28.04.2022	
2	Подбор элементной базы системы	13	29.04.2022- 30.04.2022	
3	Разработка аппаратной части системы	40	01.05.2022- 06.05.2022	
4	Настройка и отладка системы	17	07.05.2022- 10.05.2022	
5	Разработка программной части	60	11.05.2022- 19.05.2022	
6	Отладка программной части	15	20.05.2022- 22.05.2022	
7	Разработка раздела БЖД	19	23.05.2022- 26.05.2022	
8	Анализ экономических показателей	20	27.05.2022- 30.06.2022	
9	Разработка графической части	15	31.05.2022- 2.06.2022	
10	Написание пояснительной записки	31	3.06.2022- 11.06.2022	

Дата выдачи задания " 26 " апреля 2022 г.

Руководитель работы  26.04.22 Бикташев Р. А.
(подпись, дата)

Задание к исполнению принял студент  26.04.22 Тупичак Д.А.
(подпись, дата)

Работу к защите допустить

Декан факультета  23.06.22 Фионова Л.Р.
(подпись, дата)

Реферат

Работа содержит 112 листов, 21 рисунок, 3 таблицы, 10 источников, 3 приложения.

Ключевые слова: Blender, Unity, Game development, прототипирование, рендеринг, Android.

Объектом исследования является разработка прототипа мобильного игрового приложения для операционной системы Android.

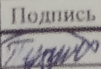
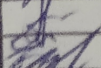
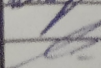
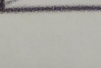
Разработка велась на языке C#.

В процессе исследования проводилось изучение принципов работы игр, возможности их оптимизации, разработка компонентов игры, тестирование игры на конечном устройстве.

Цель работы — создать мобильное игровое приложение для операционной системы Android при помощи таких инструментов, как Blender и Unity 3D, имеющей 3D дизайн, не имеющее системы автоматического боя, в котором игроки не имеют игрового преимущества за деньги.

В пояснительной записке приведен анализ предметной области, обзор технологии разработки и отладки программы.

В результате проведенной работы был создан прототип мобильного игрового приложения.

					ПГУ 1.090501.03.001 ПЗ			
Изм.	Лист	№ докум.	№	Подпись	Дата			
Разраб.		Тупичак Д. А.				Мобильное игровое приложение	Лит.	Лист
Провер.		Бикташев Р. А.				Пояснительная записка		
Н. Контр.		Бычков А. С.					2	68
Утверд.		Митрохин М. А.				ФВТ, каф. ВТ гр. 17В02		

Содержание

Перечень обозначений.....	7
Введение.....	8
1 Анализ предметной области.....	11
1.1 Сбор материала по теме.....	12
1.1.1 Архитектура Android.....	12
1.1.2 Эмулятор для разработки приложений Android SDK.....	15
1.1.3 Компоненты пользовательского интерфейса Android.....	16
1.1.4 Программное обеспечение для разработки игрового мобильного приложения.....	25
1.1.5 Сложности и ограничения при создании игрового мобильного приложения.....	32
1.1.6 Минимальные системные требования для воспроизведения игровых мобильных приложений.....	34
1.1.7 Этапы создания игрового мобильного приложения.....	35
2 Разработка собственного игрового мобильного приложения.....	38
2.1 Подготовка объектов к игре.....	40
2.2 Реализация программного интерфейса.....	41
2.3 Структура игрового мобильного приложения.....	42
2.4 Прототипирование.....	43
2.5 Реализация приложения.....	46
3 Оптимизации графики игры с помощью шейдинга в Unity 3D.....	57
4 Экономическое основание разработки.....	63
4.1 Расчёт затрат на создание программного обеспечения.....	63
4.2 Установление цены программного обеспечения.....	65

					<i>ПГУ 2.090501.03.001 ПЗ</i>	<i>Лист</i>
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		6

5 Безопасность жизнедеятельности.....	69
5.1 Неблагоприятные факторы воздействия.....	69
5.2 Противодействие вредным факторам.....	70
5.2.1 Организация безопасных условий труда на рабочем месте.....	70
5.2.2 Расчёт освещённости.....	72
5.2.3 Параметры микроклимата.....	74
Заключение.....	75
Список использованных источников.....	76
Приложение А – Листинг программы.....	77
Приложение Б – Диаграммы UML.....	98
Приложение В – Презентация.....	100

					ПГУ 1.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

Перечень обозначений

Ассет – файл проекта (модель, код, текстура, звуковой файл)

Коллайдер – компонент, обеспечивающий обнаружение столкновений

Меш - полигональная сетка 3д модели

Полигональная сетка – совокупность вершин, рёбер, граней, которые определяют форму многогранного объекта в трёхмерной графике

Полигонаж – количество полигонов

Префаб – скомбинированная из нескольких файлов (компонентов) конструкция

Прототип – упрощенная модель игры, проета

Сэмплер – подпрограмма, сопоставляющая координаты вершин трехмерной модели с координатами текстуры

Текстура – растровое (пиксельное) изображение, накладываемое на поверхность трёхмерной модели для придания ей цвета, окраски, иллюзии рельефа.

Шейдер – короткая программа, визуализирующая графические данные

Шейдинг – процесс визуализации

					ПГУ 1.090501.03.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В настоящее время мобильные игровые приложения являются одним из популярных досугов, которые помогают нам быстро расслабиться, сбросить напряжение или скрасить рутинное времяпровождение в очередях, дороге и т.п. Популярность и спрос на игровые приложения настолько велики, что разработчики, рассчитывая на большой денежный прирост, часто готовы использовать «избитые» шаблоны. Такие площадки, как App Store и Google Play Market заполнены приложениями с одинаковой механикой и сюжетом, разница между которыми состоит только в наполняющем их контенте — музыкальном сопровождении, изображениях, моделях, декорациях и т. д., что вызывает разочарование пользователей. Поэтому одной из проблем игровой индустрии является создание все более оригинальных игр, соответствующих различным вкусам пользователей.

Мобильные игры развивают определенные качества человека. Чтобы разобраться в некоторых играх, пройти их или победить в матче (если игра сессионная) нужно думать и уметь быстро принимать решения, как поступить в той или иной ситуации, как победить в битве с наименьшими потерями, как получить больше бонусов, и так далее. Чем больше человек проводит времени в игре, тем лучше у него развиваются такие качества, как логическое мышление или скорость реакции. Современные игры оснащаются системами, автоматизирующими получение прогресса в игре без применения усилий игрока в силу его отсутствия в самом процессе. Чисто технически, игра играет сама в себя, и в этом кроется ещё одна проблема: получая бонусы, игрок не развивает полезные навыки. Игра должна оставаться игрой — приносящей удовольствие, развивающей полезные навыки, восстанавливающей хорошее настроение и силы.

					ПГУ 1.090501.03.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

Во многих мобильных играх можно создать своего персонажа или свой победоносный отряд из героев, которые будут смело сражаться с чудищами разных форм и размеров. И даже если в один момент они смогут стать сильнее всех, то игроки в прекрасный момент могут столкнуться с ещё более ужасным монстром, ставшим самым могущественным, не прилагая к этому никаких усилий. Большинство предметов, навыков и игровой валюты человек может просто купить за реальные деньги, обеспечив себе в конечном итоге далеко не самое честное преимущество, а игроки, получившие преимущество таким способом, часто впадают в зависимость и лишаются значительного количества денежных средств. Решение очевидно — создавать игры, в которых если и есть покупки за реальные деньги, то дающие лишь незначительное преимущество в боях или не дающее его вовсе. Разработчики игр не должны злоупотреблять финансовыми возможностями игроков.

Наряду с этим, существует такая проблема, как недружественный интерфейс игры. Максимальное использование экрана устройства напрягает пользователя в поисках нужного объекта. Связанные между собой элементы интерфейса могут быть разбросаны по всему окну в беспорядке или скомпонованы так, что с ними неудобно взаимодействовать. Наличие большого количества текстовых пояснений также может мешать получать удовольствие от игры. Интерфейс может не давать игроку обратную связь и, нажимая на кнопку он может не понимать, что в результате произошло и почему. Отсутствие контекстуальных подсказок и напоминаний также может усложнять процесс прохождения игры для привыкших к удобству пользователей.

Наконец, заключительной проблемой могут быть ошибки оптимизации приложения: игра может занимать место, нагревать устройство, тормозить процесс игры из-за большого кода, громоздких моделей. Чем проще и легче компоненты игры, тем быстрее мобильное устройство будет их обрабатывать и воспроизводить.

					ПГУ 1.090501.03.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

На сегодняшний день не существует стандартных инструментов разработки мобильных приложений. Каждый разработчик пытается сделать операционную систему на устройстве уникальной, и поэтому возникают проблемы совместимости между различными приложениями.

Таким образом, актуальность темы дипломного проекта очевидна. Требования пользователей к качеству игр растут, поэтому создание новых игровых приложений является востребованным среди разработчиков.

Предмет исследования - технология разработки игрового приложения для операционной системы Android с помощью Blender и Unity 3D.

Объект исследования – игровое приложение для операционной системы Android.

Цель работы – создание игрового мобильного приложения для операционной системы Android, привлекательное для пользователя своим дизайном, несложным управлением и дружелюбным интерфейсом, не имеющее системы автоматического боя и игрового преимущества за деньги. Предполагается оптимизировать приложение для использования на любом мобильном устройстве.

Для достижения этой цели необходимо решить следующие задачи:

- 1) изучить преимущества операционной системы Android для игровых мобильных приложений;
- 2) рассмотреть вопрос об инструментах создания игровых приложений для операционной системы Android и использовать их для создания приложения;
- 3) рассчитать экономические показатели программного продукта;
- 4) выполнить описание техники безопасности программиста при разработке приложения.

					ПГУ 1.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

1 Анализ предметной области

В связи с большим количеством устройств, работающих с помощью операционной системы Android, разработка приложения для этой системы является наиболее актуальной. Операционная система Android основана на платформе Linux для мобильных устройств, разработанных Open Handset Alliance. Она позволяет создавать приложения на основе Java, которые управляют устройством через библиотеки Google. Кроме того, с помощью Android Development Kit можно писать приложения на C# и других языках программирования. Существует ряд других преимуществ Google, которые отличают данную операционную систему.

Система даёт новые возможности, так как может объединять информацию из Интернета с данными телефона, контактной информацией и другими данными.

Эта операционная система позволяет получить доступ к основным функциям мобильного устройства, используя API стандартный вызов.

Для Android нет никакой разницы между основными телефонными приложениями и программным обеспечением других производителей.

Эта операционная система позволяет быстро и легко разрабатывать другие приложения. В SDK есть все необходимое для создания и запуска приложений.

Большая часть кода лицензируется в соответствии с открытой и неограниченной лицензией Apache 2, что позволяет свободно использовать исходный код для создания собственных систем.

Рассмотрим операционную систему Android подробнее.

					ПГУ 1.090501.03.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

1.1 Сбор материала по теме

1.1.1 Архитектура Android

Android - платформа, которая абстрагирует разработчика от ядра, и позволяет ему создать код в Java. Android имеет несколько полезных функций. Во-первых, это основа предлагает большой набор API для создания различных типов приложений, а также обеспечивает возможность повторного использования и замены компонентов, предлагаемых приложениями платформы и сторонних производителей. Во-вторых, наличие виртуальной машины Android Runtime, которая отвечает за запуск приложений на Android. Кроме того, услуги множества разработчиков графических библиотек для 2D и 3D приложений, поддерживают мультимедийные форматы (Ogg Vorbis, MP3, MPEG-4, H.264, PNG), API для доступа к камере, GPS, компас, акселерометр, сенсорный экран, джойстик или клавиатура. Существует даже специальный API для воспроизведения фоновых звуковых эффектов, что полезно для нас в разработке игр.

Android-архитектура формируется из множества компонентов [1]. Каждый компонент основан на элементах нижнего уровня. На рис. 1 представлен обзор основных компонентов Android. В нижней части рисунка 1 показано, что ядро Linux обеспечивает основные драйверы для аппаратных компонентов системы. Кроме того, ядро отвечает за память, управление процессами, сетевую поддержку и т.д. Среда выполнения Android, построенная вокруг ядра, отвечает за разработку и осуществление Android приложений. Каждая программа работает в своем собственном процессе со своей собственной виртуальной машиной Android Runtime (ART), которая пришла на смену старой виртуальной машине Dalvik в Android 5.0 и представляет собой новый тип компиляции. Суть данного метода в том, что система компилирует приложения до его запуска, во время установки и пользователь запускает программу с уже преобразованным кодом, что значительно ускоряет процесс её

					ПГУ 1.090501.03.001 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

загрузки, прежде всего из-за малого количества потребляемой оперативной памяти.

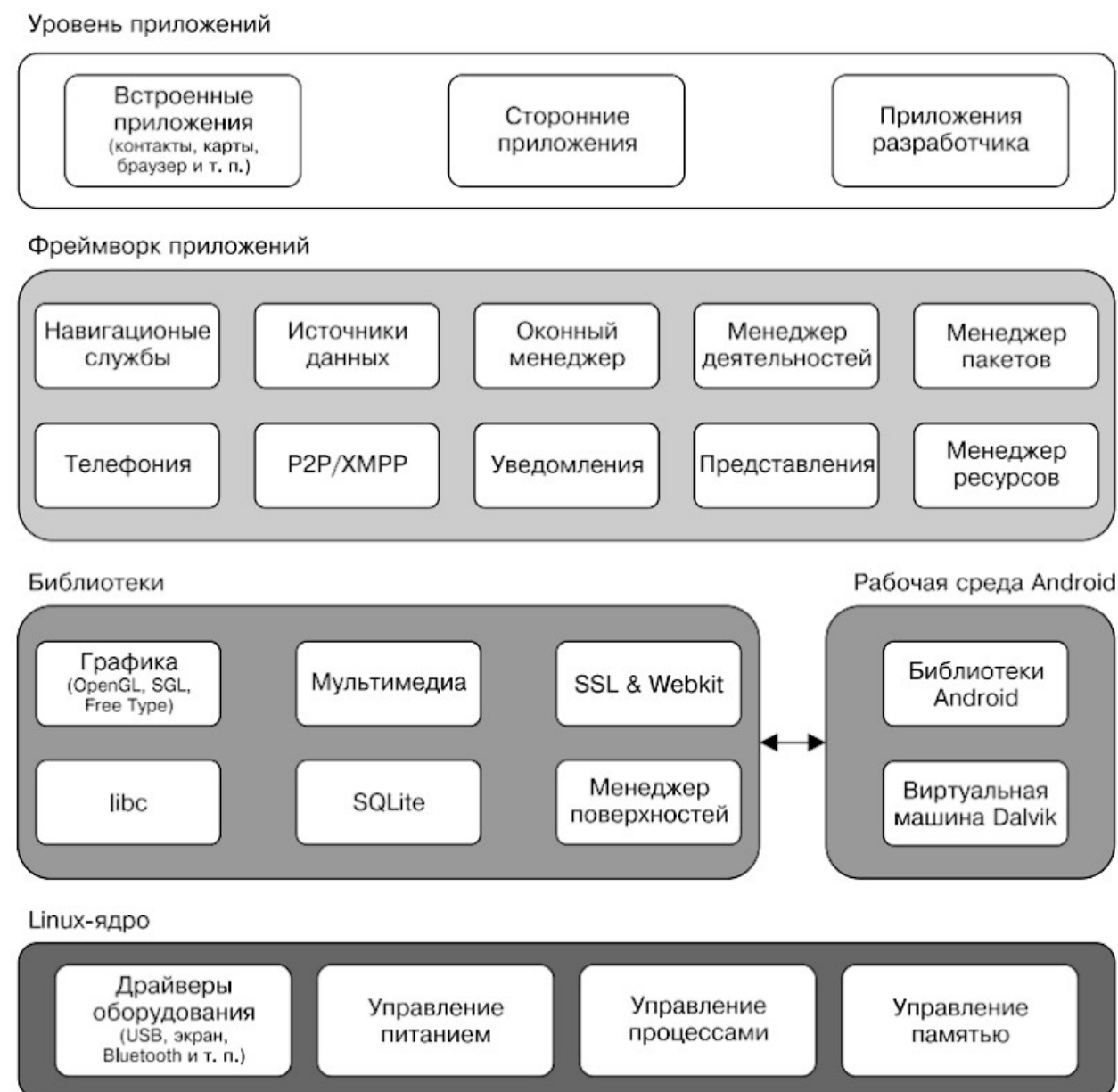


Рисунок 1 – Архитектура Андроид

В ART, как и в Dalvik используется свой формат байткода DEX. Java-файлы с классом преобразованы в формат DEX с помощью специального инструмента DX, доступном в SDK. Формат DEX занимает гораздо меньше места в памяти, чем классический тип файлов CLASS, что достигается за счет высокой степени сжатия, разделения на таблицы и объединения несколько

					ПГУ 1.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		14

CLASS-файлов. ART виртуальная машина взаимодействует с библиотеками ядра, которые предлагают базовые функциональные возможности для Java-программ. Эти библиотеки имеют большой, но не полный перечень классов, доступных через Java SE. До Android 2.2 (Froyo) весь код был интерпретирован. В Froyo было введено отслеживание JIT-компилятора, который может компилировать части байткода в машинный код на лету. Это значительно повышает производительность приложений, которые требуют большего количества вычислений. JIT-компилятор может использовать возможности процессоров, которые специально разработаны для сложных вычислений, таких как операции с плавающей запятой. Каждое приложение, работающее в экземпляре виртуальной машины ART, имеет в своем распоряжении от 16 до 24Mb RAM. Это необходимо иметь в виду, используя картинки и звуковые ресурсы. В дополнение к основным библиотекам, которые предлагают некоторые функциональные возможности Java SE, есть также множество собственных библиотек в C / C ++, которые помогают создать основу для применения структуры. Эти системные библиотеки в основном отвечают за комплекс функций (рисование графики, воспроизведение звука, доступ к базе данных). API развернул их с помощью классов Java в рамках приложения, которое используется для написания игр. Фреймворк связывает системные библиотеки и среды выполнения, создавая тем самым привязку к Android. Структура управляет приложениями и предлагает сложную среду, в которой они работают. Разработчики создают приложения для этой структуры с набором API-интерфейсов в Java, охватывающих различные области: разработка пользовательского интерфейса, фоновые службы и так далее.

					ПГУ 1.090501.03.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

1.1.2 Эмулятор для разработки приложений Android SDK

Для разработки приложений используется высокоуровневый прикладной интерфейс программирования Java для Android, при помощи которого можно создавать приложения для конечных пользователей Android. Рассмотрим особенности эмулятора Android, фундаментальные компоненты Android и пакеты, входящие в состав SDK. Комплект для разработки программного обеспечения (SDK) для Android поставляется с Android Studio, плагином, который называется набором инструментальных средств для разработки на Android (ADT). Этот инструмент разработки IDE (IDE) для создания, отладки и тестирования приложений на Java. Android SDK может использоваться без ADT; вместо инструментов, можно использовать инструменты командной строки. Эмулятор поддерживает использование обоих подходов, и с его помощью можно запустить, исправить и проверить приложения. Полностью функциональный эмулятор для Android воспроизводит наиболее изученные характеристики устройства. Среди тех функций, которые не могут быть имитированы в эмуляторе, являются USB-подключение, работа камеры и видео, имитация работы наушников, батареи и технологии Bluetooth. Android Emulator базируется на технологии с открытым исходным кодом "имитация" процессора под названием QEMU, который был разработан Беллар. Та же самая технология может эмулировать одну операционную систему на другую, независимо от того, какая используется процессором. QEMU обеспечивает эмуляцию уровня процессора. При использовании эмулятор Android имитирует процессор, который функционирует на базе ARM (Advanced RISC Machine, Advanced RISC машина). ARM - 32-разрядная архитектура микропроцессоров на базе RISC (Reduced Instruction Set Computer, компьютер с сокращенным набором команд), которая за счет сокращения количества команд достигает простоты конструкции и повышения производительности. Эмулятор использует процессор в такой моделируемой версии Linux, которая используется в Android.

					ПГУ 1.090501.03.001 ПЗ	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

ARM широко используется в мобильных устройствах и встроенных электронных устройствах, где важно распределить небольшое количество энергии. Многие имеющиеся в продаже мобильные устройства имеют процессоры с этой архитектурой.

1.1.3 Компоненты пользовательского интерфейса ANDROID

Рассмотрим пользовательский интерфейс, используемый в рамках ОС Android и сравним его с другими полнофункциональными UI-структурами, применяемыми на локальных компьютерах. UI-фреймворк Android принадлежит к четвертому поколению, если считать первое поколение традиционного прикладного программирования интерфейса Microsoft Windows, основанный на C и MFC (классов Microsoft Foundation, Microsoft библиотеки базовых классов, основанных на C++) - второй. В этом случае, Swing UI-фреймворк, основанный на Java, будет третьим поколением. Android UI, JavaFX, Microsoft Silverlight и язык Mozilla XML (XUL) пользовательские интерфейсы включают новый тип UI-фреймворк четвертого поколения, который является декларативным UI и поддерживает независимую тематизацию. При программировании в пользовательском интерфейсе Android используется объявление интерфейса в файлах XML. Тогда определение представления XML загружаются в приложение с пользовательским интерфейсом, как в Windows. Даже меню приложения загружаются из файла XML. Экраны (окна) Android часто называют деятельностью, которые включают в себя несколько типов и пользователи должны выполнить логический элемент процесса. Виды (представления) являются основными элементами, которые находятся в пользовательском интерфейсе Android. Формы могут быть объединены в группы (группы Видов). Для внутренней организации этого вида, они используют в течение длительного времени известную в программировании концепцию холста и взаимодействие пользователя с системой. Такие композитные представления, которые включают в себя виды и группы видов,

					ПГУ 1.090501.03.001 ПЗ	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

работают на основе специальной логики компонента пользовательского интерфейса Android. Одним из ключевых понятий базы является управление Android-Power жизненным циклом окон событий (окна деятельности). Протоколы системы используются, так что Android может управлять ситуацией пока пользователь, выполняет действия, чтобы скрыть, восстановить, остановить и закрыть окно явления. Основные компоненты Android Фреймворк для пользовательского интерфейса Android, вместе с другими компонентами Android основан на новой сущности, называется намерение. Намерение - сложный феномен, который сочетает в себе идеи, такие как сообщения, отображаемые в окне, действия модель "публикации и подписки", обмен межпроцессной информацией и регистрами приложений. В зависимости от того, какой браузер установлен на мобильном устройстве, Android будет выбрать наиболее подходящее для отображения сайта. Кроме того, широко поддерживаются ресурсы Android, которым знакомы строкт и растровые изображения, и по крайней мере некоторые известные элементы, такие как определение представления на основе XML. Использование ресурсов в этих рамках, осуществляется по-новому, с помощью чего работа ресурса становится более простой, понятной и удобной.

Еще одной инновационной концепцией в Android является контент-провайдер. Поставщик контента относится к абстракции источника данных, который может быть представлен в качестве эмиттера и REST обслуживания (репрезентативная State Transfer). В основе этой базы данных лежит абстракция SQLite и делает это свойство контент-провайдеров мощным инструментом для разработки приложений. Было отмечено, что XML играет важную роль в описании пользовательских интерфейсов для Android. Хотя Android поддерживает и диалоговые окна, все они являются асинхронными. Работа с такими асинхронными диалогами особенно трудна для разработчиков, которые привыкли работать с синхронными модальными диалоговыми окнами, которые

					ПГУ 1.090501.03.001 ПЗ	Лист
						18
Изм.	Лист	№ докум.	Подпись	Дата		

используются в некоторых оконных фреймворках. Android также поддерживает анимацию - эта функция включена в совместно со стекком пользовательского интерфейса, который основан на видах объектов. В Android используется анимация двух видов: с построением промежуточных кадров (tweening animation) и покадровая (frame-by-frame animation). Промежуточной анимацией называют картинки, которые отображаются между основными. На компьютере эти картинки создают изменения средних значений в определенные промежутки времени и перерисовки фона. Кадровая анимация состоит из последовательности кадров, которые нарисованы один за другим через равные промежутки времени. На Android используются обе версии анимации, используя функцию обратного вызова анимации, интерполяторов и матрицу преобразования. Кроме того, Android может идентифицировать эти типы анимации в файле XML-ресурсов. Графическая библиотека, которая основана на таком процессе, поддерживает стандартные матричные преобразования, что позволяет масштабировать, перемещать и поворачивать изображения. Объект камеры, которые присутствуют в графической библиотеке обеспечивает поддержку глубины и проекции, так что на двумерном интерфейсе удастся имитировать трехмерные эффекты. В Android доступна и трехмерная графика. Это связано с тем, что введен стандарт OpenGL ES 1.0. Как и OpenGL, он основан на плоском языке API C. Так как интерфейс прикладного программирования Android SDK основан на Java, OpenGL для доступа к нему должен использовать Java-связывание. В Java ME является обязательным для OpenGL ES и уже определено с помощью запроса на спецификации Java QSR 239 и для Android OpenGL ES использует тот же тип Java-связывания.

На Android, используются новые подходы, связанные с идеей информации на кончиках пальцев (information at your fingertips), доступ к которой осуществляется через домашнюю страницу. Первая из этих идей называется живые каталоги (живые папки). Используя живые каталоги, могут

					ПГУ 1.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

быть опубликованы коллекция элементов в папке, расположенной на главной странице сайта. Содержание этой коллекции меняется от базового изменения в его базе данных. Новые данные могут быть отправлены на устройство или со съемных носителей, или из Интернета. Вторая идея связана с домашней страницей - домашний виджет (home screen widget). Главные виджеты используются для отрисовывания с помощью пользовательского интерфейса виджета информации на домашней странице. Эта информация может быть изменена через регулярные промежутки времени. Примером таких данных может быть несколько сообщений электронной почты, которые хранятся на устройстве. Встроенный поиск Android (Integrated Android Search) - третья идея, связанная с использованием домашней страницы. Этот тип поиска позволяет искать информацию как на устройстве, так и в Интернете. В Android эта функция не ограничивается только одним поиском и может давать команды с помощью элемента управления поиска. Кроме того, так называемые Поддерживаемые жесты Android, то есть интерпретация движений пальцев пользователя работы с устройством. Android позволяет записывать любую последовательность движений пальцев на экране и сохранять их в виде жестов. Затем такие жесты могут использоваться приложениями для определения конкретных действий. Кроме инструментов Android SDK, есть и другие независимые инновации, которые делают процесс разработки интересной и простой. Некоторые примеры таких явлений - XML / VM, PhoneGap и титана. Titanium позволяет использовать HTML технологию в программировании для браузера на основе WebKit Android.

Еще один интересный компонентом Android SDK являются услуги на основе местоположения. В этом разделе SDK предоставляет разработчикам интерфейсов прикладного программирования, приложение, предназначенное для работы с картами, а также для получения информации в режиме реального времени, связанной с расположением устройства. Компоненты дисплея и

					ПГУ 1.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

компоненты, связанные с телефонией Интерфейсы Android API предназначены для работы с аудио-, видео - и телефонными компонентами. До версии 1.5 на Android можно было только записывать аудио, а видео - нет. Версия 1.5 дала возможность записывать аудио и видео. Это было сделано с помощью MediaRecorder. В версиях Android 2.0 и выше применяется Pico двигатель для преобразования текста в речь (TTS). Важно отметить, что во всех этих концепциях Android, объединенных в единый файл XML, который определяет пакет приложений. Этот файл представляет собой файл описания. В файле описания Android содержатся определения действий, регистрируются поставщики содержимого и поставщики служб и обозначаются права доступа.

Чтобы получить более полное представление о строении платформы, необходимо иметь представление о пакетах Java. Ниже приводится краткое описание важных пакетов, составляющих Android SDK:

- **Android.app** - реализует модель приложения для Android. Среди основных классов - приложение, которое описывает начальную и конечную семантику, а также целый ряд классов, связанных с явлениями, элементы управления, диалоговых окон, окон предупреждений и уведомлений;

- **Android.Bluetooth** - содержит классы для работы с технологией Bluetooth. Основные классы BluetoothAdapter, BluetoothDevice, BluetoothSocket, BluetoothServerSocket и BluetoothClass. BluetoothAdapter класс может быть использован для управления Bluetooth-адаптером, установленным на локальном компьютере. Этот адаптер может включить, отключить или запустить процесс обнаружения. Класс BluetoothDevice является удаленным устройством Bluetooth, к которому можно подключиться. Для связи Bluetooth имеет два сокета, используемых между устройствами. Bluetooth Class представляет собой тип устройства Bluetooth, к которому вы подключены;

					ПГУ 1.090501.03.001 ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

- Android. content. pm. - предоставляет классы для работы, связанные с помощью диспетчера пакетов. Он содержит информацию о разрешениях, установленных пакетах, установленных поставщиками, услуг и компонентов, таких как акции, а также установленных приложений

- Android. content. res - предоставляет доступ к файлам ресурсов, как структурированными неструктурированным. Основные классы AssetManager (для неструктурированных ресурсов) и материальных ресурсов;

- Android.database - реализует идею абстрагирования базы данных. Основной интерфейс называется Cursor;

- Android.database.sqlite - реализует концепцию пакета базы данных Android, с использованием в качестве физической базы данных SQLite. Основные классы SQLiteCursor, SQLiteDatabase, SQLiteQuery, SQLiteQueryBuilder и 25 SQLiteStatement ;

Класс абстрактного пакета базы данных:

- Android.gesture - в этом пакете располагаются все классы и интерфейсы, необходимые для работы с определенными пользователем жестами. Основные классы Gesture, GestureLibrary, GestureOverlayView, GestureStore, GestureStroke, GesturePoint;

- Android. graphics - содержит класс Canvas, Camera, Color, Matrix, Movie, Paint, Path, Rasterizer, Shader, SweepGradient nTypeFace;

- Android.graphics.drawable - предназначен для работы с протоколами рисования и фоновых изображений, обеспечивает эффекты анимации при работе с рисованными объектами;

- Android.graphics.drawable. shapes - предназначены для работы с контурами, в том числе ArcShape, OvalShape, PathShape, RectShape и RoundRectShape;

					ПГУ 1.090501.03.001 ПЗ	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

- **Android.hardware** - позволяет использовать так называемые естественные классы, предназначенные для работы с камерой. Класс камеры является распространенным устройством - камерой, а класс **android.graphics.Camera** - графическая концепция, не имеющая никакого отношения к реальной физической камере;

- **Android.location** - содержит классы **Address**, **GeoCoder**, **Location**, **LocationManager** и **LocationProvider**. Класс **Address** является упрощением-замещением **Language XAL** (расширяемый язык адреса). **Geocoder** позволяет узнать адрес координат объекта (широта и долгота), и наоборот. В **Location** представлена информация о широте и долготе;

- **Android.media** - содержит классы **MediaPlayer**, **MediaRecorder**, **Ringtone**, **AudioManager** и **FaceDetector**. Класс **MediaPlayer** предназначен для потоковой поддержки аудио и видео. Класс **Ringtone** используется для воспроизведения коротких аудиоотсчетов, которые могут быть использованы в мелодии или уведомлениях. **AudioManager** отвечает за регулировку громкости. **FaceDetector** может быть использован для обнаружения человеческих лиц на точке (растровые) рисунки;

- **android.net** - реализует базовую сеть на уровне сокетов API. Основные классы включают **Uri**, **ConnectivityManager**, локальный сокет и местный **ServerSocket**. Следует также отметить, что **Android** поддерживает уровень **HTTPS**-браузера и сетевой уровень. Кроме того, **Android** поддерживает **JavaScript** в браузере;

- **android.net.WiFi** - управляет подключением **Wi-Fi**. Основные классы **WifiManager** и **WifiConfiguration**. Класс **WifiManager** отвечает за составление списка настроенных сетей и работает с текущей активной сетью **Wi-Fi**;

					ПГУ 1.090501.03.001 ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

- Android. OpenGL - содержит вспомогательные классы, которые используются при выполнении операций OpenGL ES. Классы Basic OpenGL ES являются частью другого набора пакетов, взятых из JSR 239.

- Android. os - служба операционной системы, доступ к которой осуществляется с помощью языка Java. Некоторые важные классы - BatteryManager, Биндер, FileObserver, Хэндлер, Looper и PowerManager. Binder класс обеспечивают обмен информацией между процессами. FileObserver ведет учет изменений в файлах. Класс Handler используется для выполнения задач в потоке сообщений, и Looper начинается само сообщение потока;

- Android.preference - позволяет приложениям предоставить пользователям возможность управлять своими настройками для этого приложения в единой форме. Основные классы PreferenceActivity, PreferenceScreen;

- Android.provider - включает в себя набор готовых контент-провайдеров, связанных с android. content. ContentProvider. Среди поставщиков контента - Контакты, MediaStore, браузер и настройки. Этот набор интерфейсов и классов содержит метаданные для описания базовой структуры данных;

- Android. sax - обеспечивает эффективный набор простых API для XML (SAX), вспомогательные классы, предназначенные для синтаксического анализа. Основные классы Element, RootElement некоторые ElementListener интерфейсы.

- Android. speech - содержит константы для распознавания речи. Этот пакет входит только в версии 1.6 и выше;

- Android.speech.tts - обеспечивает поддержку для преобразования текста в речь. Основной класс - TextToSpeech. В Android имеется механизм PICO TTS (преобразования текста в речь, синтезатор речи) производства SVOX;

					ПГУ 1.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		24

- `android.telephony` - содержит классы `CellLocation`, `PhoneNumberUtils` и `TelephonyManager`. `TelephonyManager` класс для определения местоположения, из которого был сделан вызов, номер телефона, имя поставщика услуг, тип сети, тип телефона и серийный номер модуля идентификации абонента (`Subscriber Identity Module, SIM`);

- `Android. телефонии. GSM` - позволяет собирать информацию об адресах ячеек, основанных на местоположении данных вышек сотовой связи, но также содержит классы, ответственные за работу с SMS-сообщениями.

- `Android.telephony. CDMA` - поддерживает CDMA-телефонию;

- `Android. text` - содержит классы для обработки текста;

- `Android. text. method` - предоставляет классы для ввода текста в различные элементы управления;

- `Android. text. style` - обеспечивает разнообразие методов обработки текста;
- `Android.Utils` - содержит классы, `DebugUtils`, `TimeUtils` и `Xml`;

- `Android.view` - содержит классы, меню `View`, `ViewGroup`, а также некоторые из процессов слушателей и обратных вызовов;

- `Android. view. animation` – обеспечивает поддержку анимации в промежуточных кадрах.

- `Android. view. InputMethod` - реализует ввод-вывод системной архитектуры. В этом пакете содержится только в версиях 1.5 и выше;

- `Android. WebKit` - содержит классы, связанные с веб-браузером. Среди основных классов `WebView`, `CacheManager` и `CookieManager`;

- `Android. widget` - содержит все классы элементов управления пользовательского интерфейса, которые получены главным образом с точки зрения класса. Основные виджеты - Кнопка, Галочка, хронометр, `AnalogClock`,

					ПГУ 1.090501.03.001 ПЗ	Лист
						25
Изм.	Лист	№ докум.	Подпись	Дата		

DatePicker, EditText, ListView, FrameLayout, GridView, ImageButton, MediaController, ProgressBar, RadioButton, RadioGroup, RatingButton, скроллер, Scrollview, Spinner, TabWidget, TextView, TimePicker, VideoViewn ZoomButton;

- com. google. android. maps - содержит класс MapView, MapController и MapActivity, необходимые для работы с Google Maps.

Все эти многочисленные пакеты составляют богатую вычислительную платформу, предназначенную для написания программ для мобильных устройств, а Android является наиболее подходящей операционной системой для создания мобильных приложений. Очевидно, что их создание будет наиболее востребованным на рынке программного обеспечения в ближайшем будущем

1.1.4 Программное обеспечение для разработки игрового мобильного приложения

Для создания объектов и окружения игрового приложения используются модели, созданные в программах, специализирующихся на отдельных областях: 3D-редакторы, редакторы графики, музыкальные синтезаторы, игровые движки и многое другое. Для создания скриптовой составляющей используются редакторы кода. Именно от них зависит поведение персонажей, врагов, контрольных объектов и окружения в целом. Самый распространенный редактор кода на сегодняшний день – Microsoft Visual Studio.

					ПГУ 1.090501.03.001 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

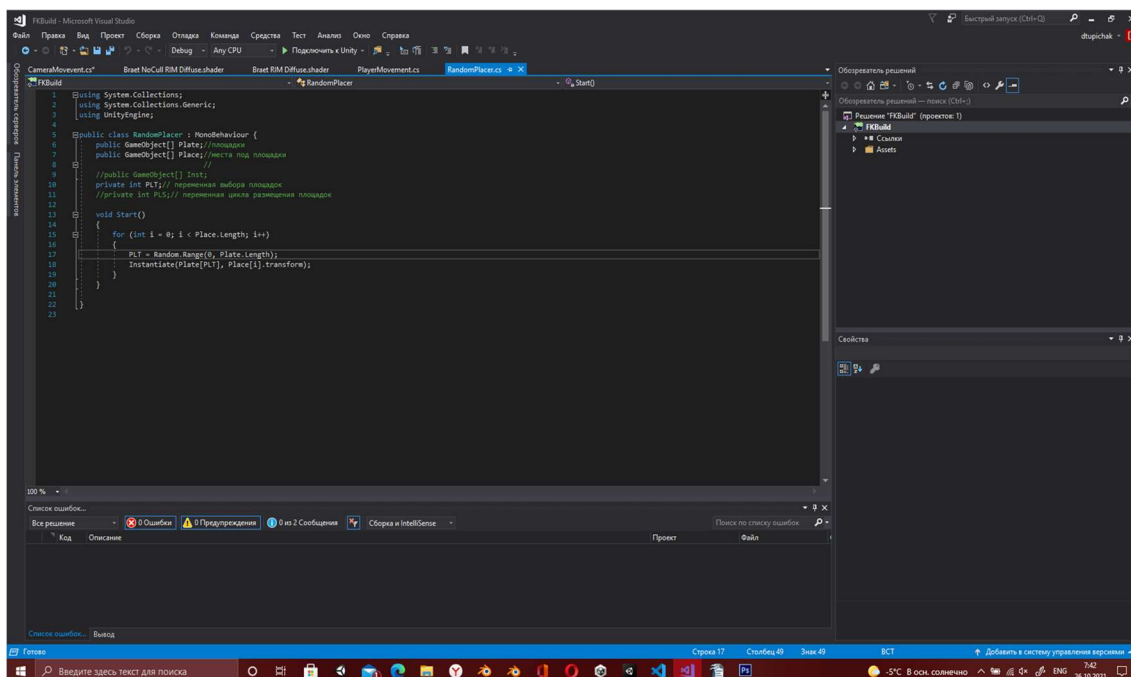


Рисунок 2 - Microsoft Visual Studio

Не менее популярным редактором кода является Microsoft Visual Studio Code, который является портативной версией Visual Studio. Он имеет гибкий и легко настраиваемый интерфейс и способен работать с любыми языками программирования.

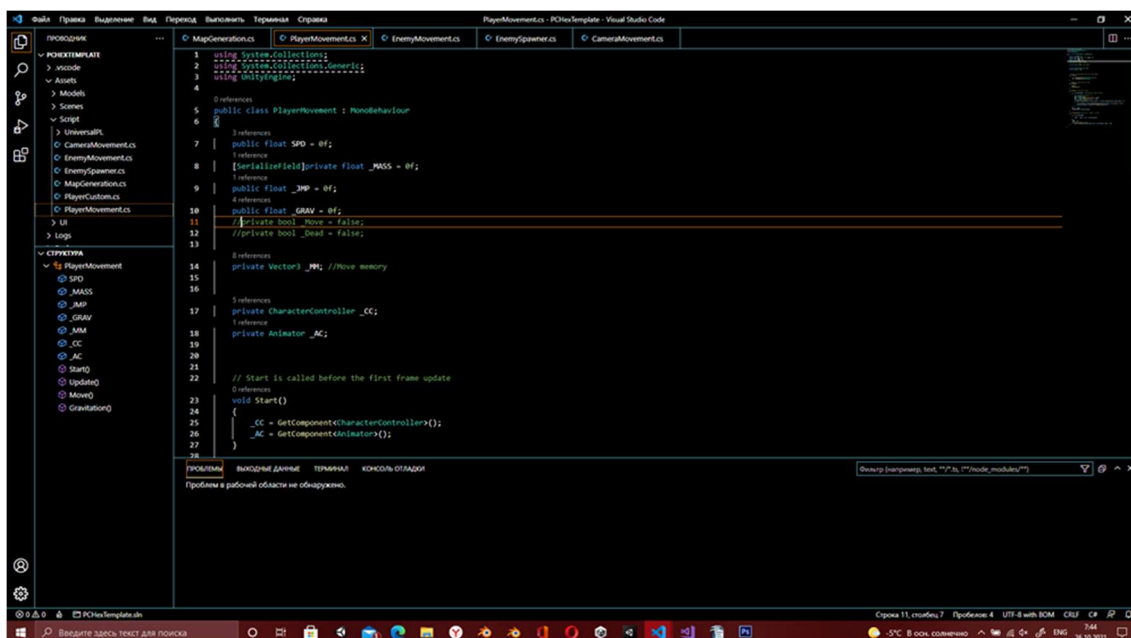


Рисунок 3 – Microsoft Visual Studio Code

						Лист
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 1.090501.03.001 ПЗ	27

Для создания изображений и текстур для трехмерных моделей используются всевозможные графические редакторы. Самыми распространенными являются Krita, Paint, Paint Tool SAI и ADOBE PhotoShop.

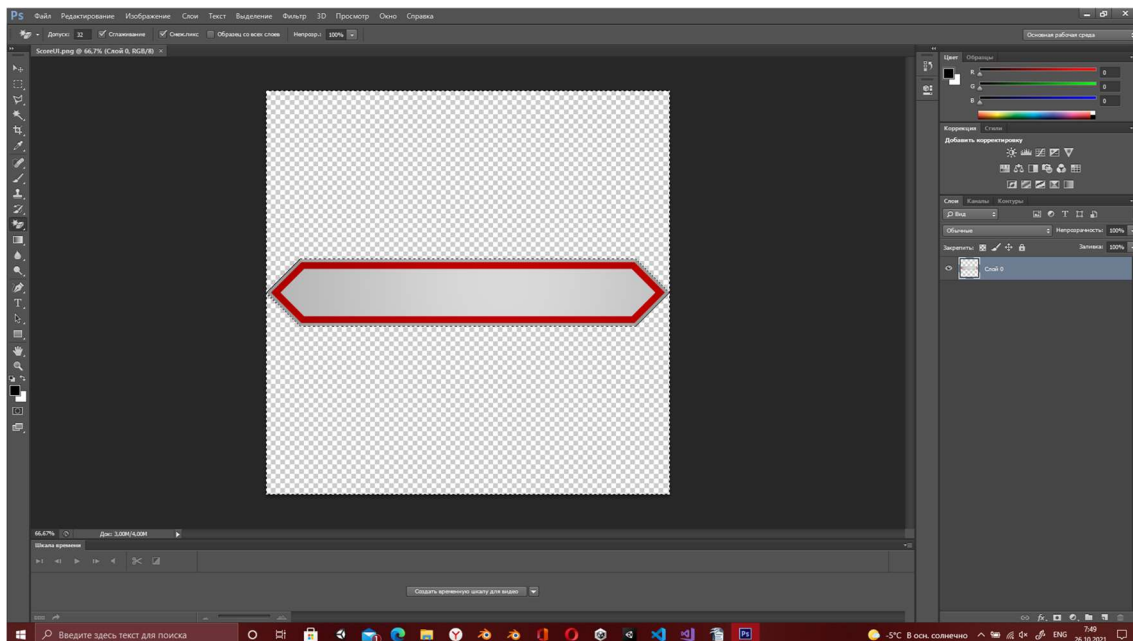


Рисунок 4 — Adobe PhotoShop

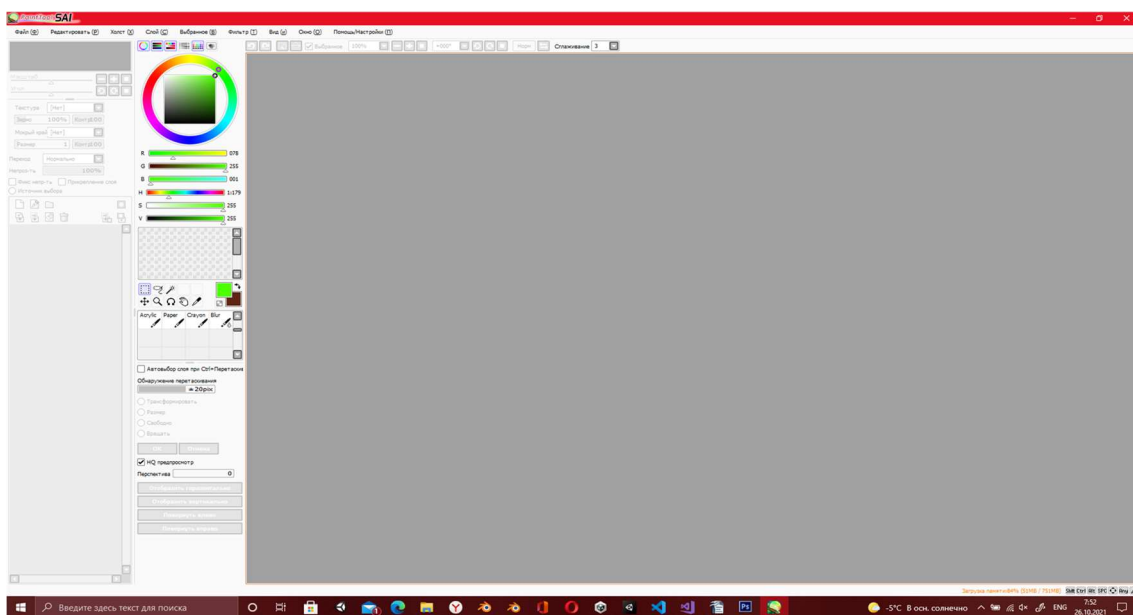


Рисунок 5 — Paint Tool SAI

					ПГУ 1.090501.03.001 ПЗ	Лист
						28
Изм.	Лист	№ докум.	Подпись	Дата		

движки, преобразующие весь ранее созданный материал, непосредственно, в играбельный проект. Существуют различные программы, с помощью которых реализуются проекты мобильных игр. Все они имеют свои преимущества и недостатки. Чтобы определиться с платформой, следует проанализировать самые популярные из них.

Unity – межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies [3].

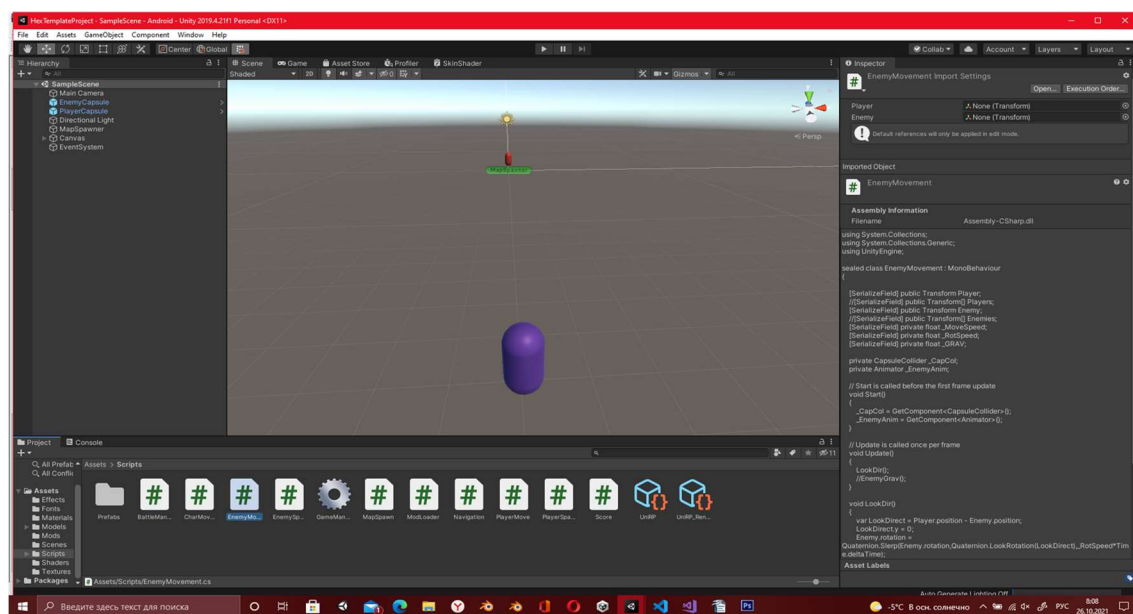


Рисунок 8 – Внешний вид программы Unity

Она позволяет создавать игры на более чем 25 платформах, включая игры на компьютерах, консолях, приставках, смартфонах. Редактор Unity имеет простой, легко настраиваемый «Drag&Drop» интерфейс. Он состоит из различных окон, благодаря которым можно производить отладку игры прямо редакторе. Движок использует для написания скриптов C# и JavaScript. Расчёты физики производит физический движок PhysX от NVIDIA. Графический API — DirectX (на данный момент DX 11, поддерживается DX 12). Благодаря понятным инструкциям на русском языке имеется большое количество разработчиков, которые пользуются этой программой.

					ПГУ 1.090501.03.001 ПЗ	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

Как показано на рисунке, программа состоит из различных окон, так, что можно редактировать игру прямо в редакторе. Любое приложение состоит из сцен и содержит различные игровые объекты. Можно сделать вывод, что преимуществами этого игрового движка являются:

- кроссплатформенность;
- простота освоения и использования;
- документация на русском языке;
- бесплатная лицензия и бесплатный магазин ассетов.

К недостаткам Unity можно отнести закрытый исходный код и, следовательно, невозможность дополнять эту программу новыми возможностями. Существуют ограничения визуального редактора со сложными схемами, и тогда в сценах визуальная работа затрудняется. Кроме того, в этой программе отсутствуют ссылки на внешние библиотеки и поэтому работу с ними приходится настраивать самостоятельно.

Unreal Engine – кроссплатформенный инструмент, разработанный американской компанией Epic Games для профессиональной разработки игр,

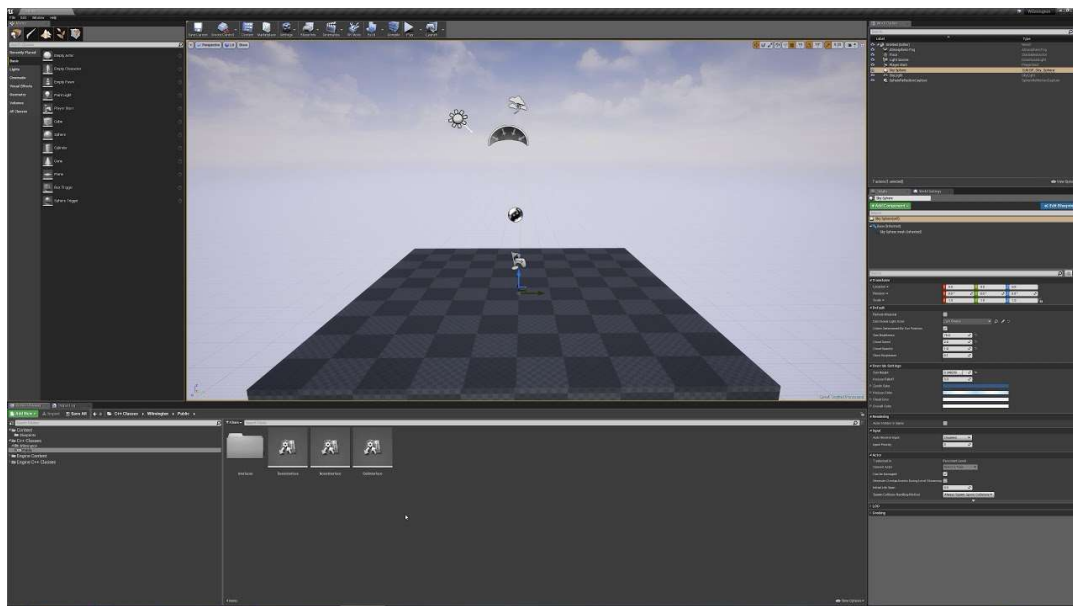


Рисунок 8 – Unreal Engine

					ПГУ 1.090501.03.001 ПЗ	Лист
						31
Изм.	Лист	№ докум.	Подпись	Дата		

Это программное обеспечение имеет профессиональную техническую поддержку и документацию на русском языке, поэтому также имеет большое число разработчиков. Что касается недостатков, то сюда относятся и невозможность вносить изменения в исходный код, и сложность в освоении этой программой. Кроме того, компания взимает комиссию со стоимости игры и с внутриигровых покупок. Перечислим достоинства этой программы:

- кроссплатформенность;
- постоянная техническая поддержка;
- документация на русском языке.

Недостатками можно считать сложность в освоении и взимание комиссии за пользование играми. Представим это сравнение в виде таблицы.

Таблица 1 - Сравнение игровых движков

Игровой движок	Встроенная физика	Графика	Кросс-платформенность	Языки	Простота в освоении	Ресурсоёмкость
Unity3D	+	3D, 2D	+	Java Script, C#	+	+
Unreal Engine	+	3D, 2D	+	C++	—	—

Проанализировав два самых популярных в среде разработчиков игровых движка, было выявлено, что оба программных продукта имеют много преимуществ, однако для реализации проекта был выбран Unity3D из-за более простого использования и доступа к бесплатному магазину ассетов. Кроме того, из таблицы видно, что Unreal Engine является более ресурсоёмкой программой

					ПГУ 1.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		32

и подключает все пакеты расширений одновременно, в то время как Unity 3D позволяет пользователю выбирать тот пакет, который ему нужен.

1.1.5 Сложности и ограничения при создании игрового мобильного приложения

Структурно каждая игра является совокупностью игрового движка и связанного с ней контента - сборника картинок, 3D – моделей, музыки и кодов, объединенных в систему с помощью игрового движка. Каждый год устройства совершенствуются, позволяя разработчикам игр использовать контент все более и более высокого качества. Но, к сожалению, телефон все еще не компьютер, поэтому, частоты процессоров и видеочипов мобильных телефонов и планшетных компьютеров все еще меньше, и ограничения на использование строго определенного количества объектов со строго определенными параметрами все еще остаются актуальными.

В современных играх количество вызовов отрисовок (Draw calls) объектов на игровую сцену раньше составляло от 70 до 75. Современные игры способны вызвать от 150 до 200 объектов соответственно. Под объектом подразумевается полигональная сетка и 1 (и только 1) покрывающий ее материал. Помимо этого, вызовом отрисовки также может быть и элемент пользовательского интерфейса.

В трехмерных играх во избежание перегрузки и несвоевременного выхода графического процессора из строя используют модели с ограниченным количеством полигонов — общее количество от 300 до 1500 треугольников на модель является своеобразным стандартом. В более ресурсоемких играх количество треугольников может достигать порядка 2500 треугольников на модель.

Шейдер — программа, исполняемая графическим процессором или видеокартой при выводе изображения на экран с помощью рендерера

					<i>ПГУ 1.090501.03.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		33

(программы, визуализирующей работу шейдера). В мобильных играх шейдер не должен содержать большое количество обрабатывающих изображение процедур, не должен отбрасывать тень и в идеале должен отрисовывать полисетку только с лицевой стороны.

В игре, предназначенной для мобильных устройств, не должен использоваться динамический источник света, во избежание зависаний приложения при проведении большого количества дополнительных расчетов и преждевременного выхода из строя на этот раз не только графического процессора.

Если игра разрабатывается для Play Market, оптимальный размер игры не должен превышать 100 Мб. В противном случае игра разбивается на несколько частей, и они скачиваются отдельно. Некоторые фирмы — разработчики, такие, как Gameloft, Com2Us, EA, Blizzard и т. д. Могут создавать специальный файл-загрузчик, который начинает скачивать данные игры только после его запуска, причем поскольку данные скачиваются прямо с сервера разработчиков, это позволяет сэкономить много времени.

При создании кода стоит опасаться большого количества операторов условия. Вместо этого — лучше поставить проверку на выполнение условия в конец процедуры или функции. Или воспользоваться командой Return для прерывания ее выполнения. Можно задействовать тернарные операции, кейсы и бадибэг — логику. При использовании циклов важно убедиться, что они всегда должны иметь конечное количество итераций. Единственное, что может не иметь конечное количество итераций — вызов процедуры Update (в Unity 3D), обновляющей данные игры каждый кадр.

Таким образом, проанализировав трудности и ограничения при создании игровых мобильных приложений, можно сделать вывод, что проектирование и разработка приложений подобного рода требует много умений и навыков, а сама игра похожа на искусство. Кроме того, устройство, на которое рассчитано

					ПГУ 1.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		34

игровое приложение, может сильно повлиять на процесс его разработки и использования.

1.1.6 Минимальные системные требования для воспроизведения мобильных игровых приложений

Существует минимально требуемый набор данных, который обеспечивает стабильность процессу игры. Следует обратить внимание на экран смартфона и его параметры, процессор и его мощность, оперативную память и её количество. Важна и совместимость приложения с мобильным устройством. Обычно описание игры содержит информацию о совместимости её с процессорами телефонов, также, могут быть указаны дополнительные данные о семействе ядра и версии операционной системы: минимальная 2.3, но лучше от 4.0. Лучшим вариантом для ресурсоёмких игр являются смартфоны с размером оперативной памяти 3-4 Гб, но не меньше 1 Гб, что обезопасит от постоянных ошибок, зависаний, самопроизвольного закрывания. Это важно, так как процессор обрабатывает данные игры и отвечает за её отображение на дисплее. Основными характеристиками мощности смартфона, на которые стоит обратить внимание являются частота процессора и количество ядер. Для ресурсоёмких игровых приложений необходимо 4 и 8 ядер с большой частотой. Экран смартфона должен иметь высокое разрешение и большие размеры. Это связано не только с удобством использования, но и с качеством картинки или действия. Лучше отдавать предпочтение экранам с разрешением от HD128x720 и диагональю от 5 дюймов. Современные игровые телефоны требуют аккумулятора большой ёмкости. Оптимальный минимум такого устройства – это батарея не менее 3000 мАч. Такого объема достаточно для 3-4 часовой непрерывной игры, даже учитывая наличие 3D-графики, и 4-6 часов простых игр.

					<i>ПГУ 1.090501.03.001 ПЗ</i>	Лист
						35
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

1.1.7 Этапы создания игрового мобильного приложения

С целью создания собственного мобильного приложения были исследованы основные этапы её разработки [10].

Сначала определяется выбор жанра — условной характеристики, указывающей главным образом на ключевые действия игрока. В настоящее время в целях привлечения новых игроков разработчики создают игры, используя несколько жанров одновременно. Одним из популярных на сегодняшний день жанром является Action (Экшн) — игра с высокой концентрацией, быстрым принятием решений и большим разнообразием механик. Все игры этого жанра можно разбить на следующие поджанры:

- Action Adventure (наличие сюжета и испытания на любой вкус)
- Slasher (характеризуется использованием оружия ближнего боя)
- Shooter (основная черта - использования оружия дальнего боя)
- Stealth (отличается от предшественников «тихими» способами игры)
- Survival/Horror (главная цель - выживание)
- Action-RPG (улучшение героя с последующими сражениями против других игроков)
- пошаговые RPG (основа - игроки ходят по очереди)
- Roguelike (непредсказуемая игра с высокой сложностью)
- Стратегии в реальном времени (RTS)
- MOBA (Multi-online battle arena)
- Экономическая стратегия
- Симуляторы
- Гонки
- Платформеры
- Квесты

Кроме того, согласно статистике аналитической компании Sensor Tower,

					ПГУ 2.090501.03.001 ПЗ	Лист
						36
Изм.	Лист	№ докум.	Подпись	Дата		

которая опубликовала отчёт самых скачиваемых игр в мире во втором квартале 2021 года, самыми популярными являются игры казуальных жанров, которые отличаются упрощённым управлением.



Рисунок 10 - Самые скачиваемые игры второго квартала 2021 года

Следующим шагом для создания игры является её прототипирование - создание проектной и продуктной документации. Этот этап определяет техническое задание для каждого из разработчиков игры, будь то художник, дизайнер, 3D – художник, звукорежиссер или программист - задания распределяются среди всех.

Основными чертами технической документации являются:

- Структурированность (Задачи адаптированы под обладающего определенной специализацией человека или группы людей).

- Защищенность от разночтений (Четко сформулированное техническое задание со всеми его деталями).

- Регулярная актуализация (обновление и сопровождение документации после релиза проекта и внесения правок в его структуру).

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		37

Следующая стадия — разработка прототипа игры, где все объекты представлены двухмерными (или трехмерными) примитивами — кубами, шарами, капсулами и фигурами, не требующими сложного подхода к моделированию. Очень важно в ходе создания прототипа игры реализовывать только управление, физику, положение камеры, работу кнопок интерфейса.

После испытаний прототип уничтожается, а на его месте начинается создание финальной версии игры. На этом этапе добавляются более мелкие механики, создаются наброски, эскизы, по которым отрисовываются модели, текстуры, пишется музыка. Как правило, перед проведением «вертикального среза» геймплея в игре есть всего по 2-3 объекта каждого типа. После проведения пробных тестов контент подвергается изменению или удалению. Конечный продукт содержит в себе намного больше элементов.

Четвертая стадия — тестирование. Различают:

-Альфа-тест — игра тестируется самими разработчиками на наличие багов, зависаний и технических ошибок. Перед тестированием составляется подробный список, по каким пунктам игра должна быть протестирована перед тем, как перейти в следующую стадию тестов. В случае обнаружения неполадок по какому-либо пункту неисправность незамедлительно устраняют.

-Закрытое бета-тестирование - разработчиками набирается команда тестеров, в которую может входить определенное количество человек, как обычные пользователи, так и профессиональные тестеры. В случае обнаружения ошибки команда разработчиков устраняет ошибку.

-Открытое бета-тестирование — тестирование игры любым желающим, количество людей не ограничено. В случае обнаружения ошибки игрок может отправить сообщение команде разработчиков с указанием на то, в каких случаях она возникает.

Последняя стадия — релиз и сопровождение, когда игра выходит в общий доступ и приобрести ее сможет любой желающий. При этом

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		38

разработчики продолжают работу над своим проектом, время от времени выпуская к ней модификации и DLC.

Представленная технология создания игр и была использована для создания игрового мобильного приложения

					ПГУ 2.090501.03.001 ПЗ	Лист
						39
Изм.	Лист	№ докум.	Подпись	Дата		

2 Разработка собственного мобильного игрового приложения

Предлагаемое игровое приложение – игра в жанрах экшн и казуальный, где персонаж пытается пережить своего соперника или соперников (в зависимости от выбора). В игре появляются монстры, которые атакуют игрока или команду. Каждый раз, когда члены одной команды погибают, игроки другой получают балл. Сцена обновляется через каждые 5 секунд и начинается новый раунд. Выигрывает та команда или игрок, которая наберет 3 балла. Игра предназначена для различной целевой аудитории, не имеет смысловой нагрузки и используется в целях развлечения пользователя.

Игра является мобильным приложением для платформы Android. В качестве инструмента разработки игрового мобильного приложения был выбран игровой движок Unity 3D, поскольку эта программа поддерживается SDK и имеет бесплатную лицензию. Камера в игре расположена под углом пятьдесят градусов. Управление в игре осуществляется с помощью джойстика, который активируется при касании экрана пользователем. При касании дисплея модель игрока начинает двигаться по карте туда, куда укажет пользователь. Если игрок убирает палец с дисплея телефона, его персонаж стоит. Это положение необходимо игроку, так как, только из такого положения он может атаковать монстра и набрать очки. В бегущем состоянии игрок будет преследоваться монстрами. Исходя из концепции игры были определены действия, которые должна предоставлять разрабатываемая система:

- пользователь должен иметь возможность входить и выходить из игры;

					ПГУ 2.090501.03.001 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		

-пользователь должен играть в игру сам и не полагаться на систему автобоя;

-интерфейс игры должен правильно отражать состояние событий и быстро реагировать на происходящие изменения.

2.1 Подготовка объектов к игре

Для создания объектов и окружения разработанного игрового приложения были использованы модели, созданные в программах, специализирующихся на отдельных областях приложения: 3D-редакторы, редакторы графики, музыкальные синтезаторы, игровые движки и многое другое.

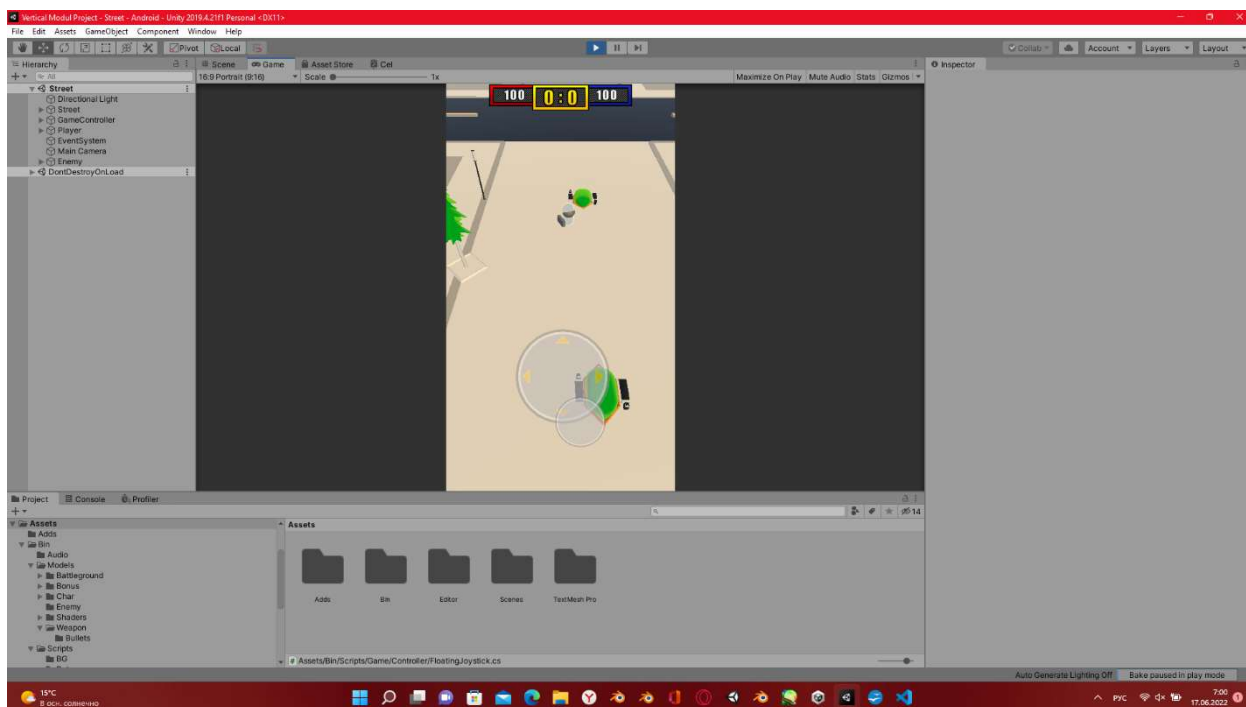


Рисунок 12 – Управление персонажем

Для создания и анимирования трехмерных моделей и графического интерфейса в данном учебном проекте использовалась программа Blender, так

					ПГУ 2.090501.03.001 ПЗ	Лист
						41
Изм.	Лист	№ докум.	Подпись	Дата		

как этот программный продукт является доступным в использовании и лучше всего подходит для визуализации объектов. Для объединения моделей интерфейса в игру была выбрана межплатформенная среда Unity 3D, так как программа имеет понятный интерфейс, бесплатную лицензию, качественную документацию на русском языке, техническую поддержку разработчиков.

В программе Blender были созданы модели объектов окружения и интерфейса. В качестве окружения была выбрана длинная и узкая улица, которая хорошо подходит под вертикальные проекты. Сначала из кубов и плоскостей была создана грубая форма – драфт (фигура, составленная из примитивных фигур). Затем на плоскостях были смоделированы контуры бордюров, а кубы были трансформированы в грубые формы зданий. Для мобильной игры этого было достаточно. В новом окне Blender был создан пустой текстурный атлас разрешением 2048 на 2048 точек и состоящий из 100 пустых белых кадров – для заполнения их текстурами. Одни участки были окрашены в монотонные цвета, другие получили градиентальную заливку, третьи – полноценную текстуру. Монотонная текстура была наложена на маленькие объекты и на те участки карты, которые находились вне зоны видимости. Текстуры с градиентальной заливкой были использованы для имитации дополнительного объема. Стоит отметить, что в более ресурсоемких играх для этого используют специальные текстуры – карты нормалей. Для асфальта, плитки и стекол были созданы полноценные текстуры, так как именно эти элементы будут видны игроку чаще всего.

Для элементов интерфейса также была использована программа Blender. В настройках визуализации было настроено разрешение 2048 на 2048 пикселей, был задан прозрачный задний фон. В качестве основы для кнопок, панелей и ползунка были использованы плоскости, которые в режиме редактирования были преобразованы.

					ПГУ 2.090501.03.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

2.2 Реализация программного интерфейса

Интерфейс игрового приложения реализован с помощью системы Unity User Interface. Интерфейс приложения состоит из главного меню, меню игрового поля и меню настройки. При запуске игры открывается главное меню с тремя кнопками «Играть», «Настройки» и «Выход». При нажатии кнопки «Играть» - начинается игровой процесс и открывается сцена боевого поля. Она состоит из статических, динамических объектов и элементов интерфейса. Через равные промежутки времени на каждой из частей появляется от одного до четырех игроков (в зависимости от режима), и две половины карты заполняются монстрами. Начинается бой. Здесь есть счётчик, который подсчитывает количество здоровья и баллы игроков. Кроме того, на поле боя есть меню «Пауза», при нажатии которого игрок может решить, остаться ему в игре или покинуть поле боя. При нажатии кнопки «Настройки» появляется экран с выбором качества графики, звука, громкости. При нажатии кнопки «Выход» игрок может покинуть игру.

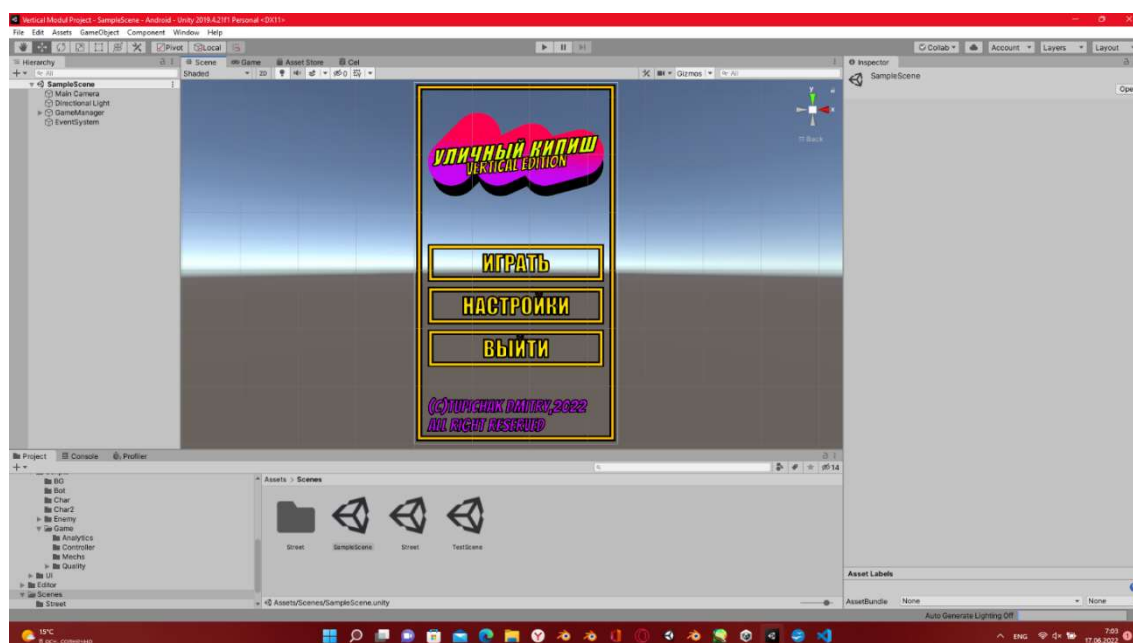


Рисунок 11 – Главное меню

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		43

2.3 Структура игрового мобильного приложения

Разработка игры была начата с создания каталогов в директории Assets: Models, Shaders, Scripts, UI (User Interface). Внутри папки Models были созданы каталоги Char, Enemy, Street, Weapon. В папку Scripts были добавлены каталоги Player, Enemy, Game.

В папке Scenes находятся все сцены, из которых состоит данное приложение. Структура проекта наглядно показана в UML-диаграмме.

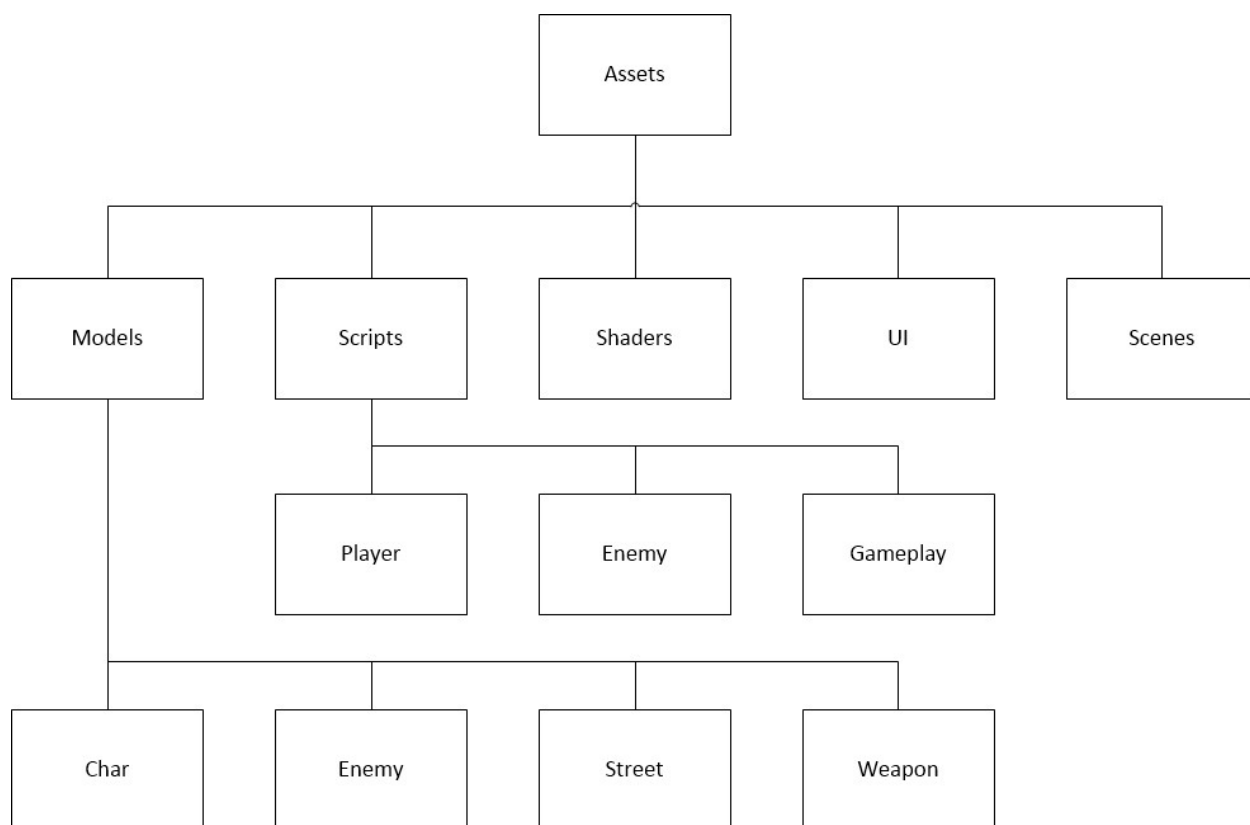


Рисунок 13 – Файловая структура приложения

В директории Models находятся модели персонажей (папка Char), врагов (папка Enemy), окружения (папка Street), оружия (папка Weapon).

В директории Scripts расположены скрипты, взаимодействующие с персонажем (папка Char), врагом (папка Enemy) или игрой (папка Game).

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		44

В директории Shaders находятся шейдеры – наборы команд для графического интерфейса.

В директории UI находятся элементы графического интерфейса, задействованные в игре.

2.4 Прототипирование

Одной из важных составляющих реализации проекта является прототипирование (визуализация). Прототипирование – это предварительная реализация, (сборка, разработка) базовой функциональности будущего проекта для анализа системы в целом. На этом этапе создаётся работающая система и отрабатываются все предложенные геймдизайнерами идеи с целью отбора нужных и важных объектов для разрабатываемого проекта. На этом этапе разработки игрового мобильного приложения происходит проверка основных механик, физики и работы кодов. Для более наглядного отображения работы отдельных элементов используют примитивные фигуры – капсулы, шары, цилиндры, кубы и тому подобное. Такие фигуры позволяют понять разработчикам, где и как у игровых объектов находятся физические границы – коллайдеры. Это позволяет выявлять ошибки при навигации и в работе физического движка на ранних стадиях.

Явными моделями можно представить только те объекты, физика и геометрия которых влияет на процесс игры незначительно или совсем не влияет. Любой прототип игры напоминает лишь болванку с реализованной идеей и набором заложенных механик. Идея в любой момент может измениться, а механики либо добавляются, либо удаляются в зависимости от их влияния на реализацию проекта.

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		45

В разрабатываемом проекте персонажи представлены капсулами визуализирующими их физические границы. Это нужно было для того, чтобы протестировать их физические границы. Проверялись следующие параметры:

- как взаимодействует персонаж с объектами окружения;
- как он взаимодействует с вражеской единицей;
- как перемещается персонаж по карте в двух осях;
- как реагирует герой на появление врага;
- разворачивается ли он и атакует, когда появляется враг;
- находится ли он в спокойном положении, если врага не видно;
- атакует ли враг противника, когда видит его;
- атакует ли враг противника, когда не видит его;
- атакует ли враг противника, когда находится в движении;
- как перемещается враг по карте, когда не видит противника;
- как изменяются значения интерфейса при получении урона врагом;
- как изменяются значения интерфейса при получении урона персонажем;
- виден ли персонаж во время управления им игроком;
- как изменяется экран победы и поражения;

Прототипы должны соответствовать следующим требованиям:

- Персонаж должен атаковать, когда находится без движения и враг виден;
- Персонаж не должен атаковать, когда находится в движении и враг виден
- Персонаж не должен атаковать, когда находится без движения и враг не виден;
- Персонаж не должен атаковать, когда находится в движении и враг не виден;

					ПГУ 2.090501.03.001 ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

-Враг должен атаковать, когда находится без движения и персонаж виден;

-Враг не должен атаковать, когда находится в движении и персонаж виден;

- Враг не должен атаковать, когда находится без движения и персонаж не виден;

- Враг не должен атаковать, когда находится в движении и персонаж не виден;

- При прикосновении нижней части дисплея должен появиться джойстик для управления персонажем;

- Персонаж должен быть виден всегда, даже с активированным джойстиком;

- Интерфейс должен обновлять данные при любом изменении в процессе игры;

- В конце игры должен появиться экран победы или поражения;

- Игрок может выйти из игры, не закончив её.

В результате проведения прототипирования выяснилось, что все требования, выдвинутые для прототипов, были выполнены, что означает переход к следующему этапу – замене прототипов персонажами, отрисованными в Blender.

2.5 Реализация приложения

За происходящие в игре действия отвечают скрипты, включающие в себя состоящие из набора методов классы. Диаграмма классов представлена на рисунке 14.

					ПГУ 2.090501.03.001 ПЗ	Лист
						47
Изм.	Лист	№ докум.	Подпись	Дата		

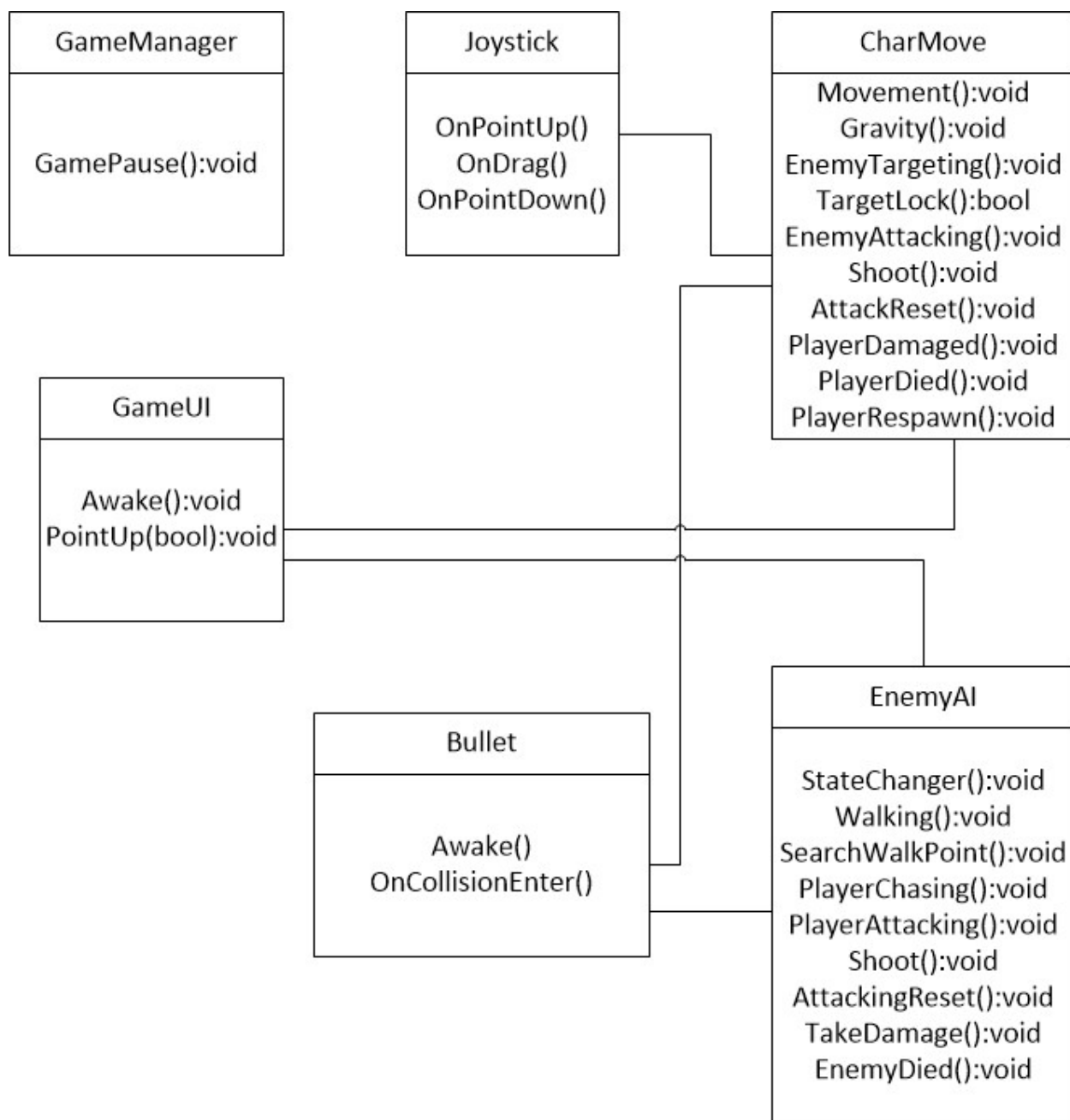


Рисунок 14 – Диаграмма классов в приложении

Класс Game Manager реализует основную логику приложения, сохраняет и изменяет данные настроек.

Класс Game UI отвечает за отображение интерфейса пользователя и обновления выводимых данных.

Класс Joystick отвечает за визуализацию джойстика на экране при прикосновении к нему и к передаче координат для моделирования вектора перемещения с помощью метода Movement класса Char Move.

Класс Char Move отвечает за перемещение персонажа по карте и за атаку врагов.

За перемещение персонажей по карте отвечают два файла проекта – Joystick.cs, CharMove.cs.

Joystick.cs определяет положение пальца игрока, соотносит его с центром джойстика и выводит его изображение на экран.

Процедура on PointerDown передаёт начальные координаты джойстика для процедуры onDrag., которая передает координаты смещения джойстика в файл ProJoyMovement для построения вектора перемещения игрока. Процедура OnPointerDown убирает джойстик с экрана.

Наиболее интересными являются следующие коды:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
public class FloatingJoystick : MonoBehaviour, IDragHandler,
IPointerDownHandler, IPointerUpHandler
{
    public JoyStickDirect JSDir = JoyStickDirect.Both;
    public RectTransform Back;
    public RectTransform Handle;
    [Range(0,2f)]public float HandleLimit = 1f;
```

					ПГУ 2.090501.03.001 ПЗ	Лист
						49
Изм.	Лист	№ докум.	Подпись	Дата		

```

private Vector2 _input = Vector2.zero;

public float Vertical {get{return _input.y;}}
public float Horizontal {get{return _input.x;}}
Vector2 JoyPos = Vector2.zero;

public void OnPointerDown(PointerEventData PED)
{
    Back.gameObject.SetActive(true);
    OnDrag(PED);
    JoyPos = PED.position;
    Back.position = PED.position;
    Handle.anchoredPosition = Vector2.zero;
}

public void OnDrag(PointerEventData PED)
{
    Vector2 JoyDir = PED.position - JoyPos;
    _input = (JoyDir.magnitude > Back.sizeDelta.x/2f) ? JoyDir.normalized :
JoyDir/(Back.sizeDelta.x/2f);
    if (JSDir == JoyStickDirect.Horizontal) _input = new Vector2(_input.x,0f);
    if (JSDir == JoyStickDirect.Vertical) _input = new Vector2(0f,_input.y);
    Handle.anchoredPosition = (_input * Back.sizeDelta.x/2f) * HandleLimit;
}

public void OnPointerUp(PointerEventData PED)
{
    Back.gameObject.SetActive(false);

```

					ИГУ 2.090501.03.001 ИЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

```

    _input = Vector2.zero;
    Handle.anchoredPosition = Vector2.zero;
}
}

```

Pro JoyMovement получает на вход три компонента и более чем десять переменных для реализации перемещения и стрельбы персонажа.

```

//MovementStats=====
[SerializeField]private float _charMass;//Масса
[SerializeField]private float _charSpd;//Линейная скорость
[SerializeField]private float _charAngSpd;//Угловая скорость
//TurretStats=====
[SerializeField, Range(10f, 200f)] private float _targetingRange =
50f;//Радиус обзора
public Transform target;//Атакуемая цель
private Collider[] _targets;//Враги в радиусе обзора
public LayerMask EnemyLayer;//Слой врагов
public bool _enemySight;//Враг виден
//ShootingStats=====
public bool _attackHold;//Перезарядка
public float _reloadRime;//Время перезарядки
public GameObject Bullet;//Снаряд
public Transform BulletSpawnPoint;//место выхода снарядов
[Range(0f,10f)]public float spread;//Разброс при стрельбе
public float BulletSpeed;//Скорость снаряда

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

```

//Stats=====
[SerializeField]private int HP;//Здоровье
[SerializeField]private int ATK;//Атака
[SerializeField]private int DEF;//Защита
public int maxHP = 100;//Максимальное количество хп
private bool _alive = true;//Жив ли игрок
//CalculatedStats=====
public float _horz,_vert,_grav;
private Vector3 _charPos;//Позиция в мире
private Quaternion _charRot;//Направление позиции в мире
//Constans=====
private string _horzDir = "Horizontal";
private string _vertDir = "Vertical";
//Components=====
private CharacterController _PCC;//Контроллер персонажа
private Animator _PA;//Аниматор персонажа
//JoypadComponent=====
public FloatingJoystick FJController;//Подключение джойстика
// Start is called before the first frame update

```

В начале игры происходит получение данных от компонентов Character Controller (физический коллайдер персонажа, определяющий его объем, допустимую высоту преодолеваемых препятствий и их допустимый угол наклона) и Animator (компонент, определяющий анимацию персонажа во время выполнения какого – либо действия).

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		52

```

void Start()
{
    _PCC = GetComponent<CharacterController>();
    _PA = GetComponent<Animator>();
}

```

Процедура CharMove не получает на вход переменных, но берёт значения из ранее упомянутого класса FloatingJoystick и по ним создаёт вектор перемещения персонажа в пространстве.

```

void CharMove()
{
    if (_PCC.isGrounded)
    {
        target = null;
        _charPos = Vector3.zero;
        _charPos.x = FJController.Horizontal;
        _charPos.z = FJController.Vertical;
        _charPos = _charPos.normalized;

        Vector3 newDirect =
Vector3.RotateTowards(transform.forward, _charPos, _charAngSpd, 0);
        transform.rotation = Quaternion.LookRotation(newDirect);
    }
    _charPos.y = _grav;
    _PCC.Move(_charPos * _charSpd * Time.deltaTime);
}

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

Получившийся вектор перемещения дополняется силой тяжести, искусственно созданной с помощью направленного вниз вектора и рассчитанной с помощью процедуры Gravity.

```
void Gravity()
{
    if (!_PCC.isGrounded)
    {
        _grav -= _charMass * Time.deltaTime;
        return;
    }
    _grav = -1f;
}
```

Если персонаж находится без движения, а в радиусе поражения появляется враг, функция EnemyTargeting делает врага целью, принуждая персонажа развернуться в его сторону и начать атаку.

```
void EnemyTargeting()
{
    if (_charPos.magnitude > 1f) return;
    _enemySight = Physics.CheckSphere(transform.position,
    _targetingRange, EnemyLayer);
    if (_enemySight) EnemyAttacking();
}
```

					ПГУ 2.090501.03.001 ПЗ	Лист
						54
Изм.	Лист	№ докум.	Подпись	Дата		

За атаку отвечает процедура Enemy Attacking, проверяющая, не вышла ли цель за радиус поражения, и, при получении отрицательного значения, с помощью процедуры Shoot генерируется снаряд и вектор его перемещения.

```
void EnemyAttacking()
{
    _targets =
Physics.OverlapSphere(transform.position,_targetingRange,EnemyLayer);
    target = _targets[0].GetComponent<Transform>();
    transform.LookAt(target);
    if (!_attackHold)
    {
        //Использование снарядов для атаки
        Shoot();
        //Reloading script
        _attackHold = true;
        Invoke(nameof(AttackReset),_reloadRime);
    }
}

void Shoot()
{
    Vector3 TargetPoint = target.position;
    Vector3   direcionWithoutSpread   =   TargetPoint   -
BulletSpawnPoint.position;
    float XSpread = UnityEngine.Random.Range(-spread,spread);
    float Yspread = UnityEngine.Random.Range(-spread,spread);
```

					ПГУ 2.090501.03.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

```

Vector3 directionWithSpread = directionWithoutSpread + new
Vector3(Xspread,Yspread,0);

GameObject CurrentBullet =
Instantiate(Bullet,BulletSpawnPoint.position,BulletSpawnPoint.rotation);

CurrentBullet.transform.forward = directionWithSpread.normalized;

CurrentBullet.GetComponent<Rigidbody>().AddForce(directionWithSpread.norm
alized * BulletSpeed,ForceMode.Impulse);

}

```

Снаряду задаются такие параметры как скорость, сила атаки, время жизни (чтобы в случае промаха снаряд не занимал место в памяти устройства). Для придания проекту сложности добавлена случайная дробная переменная *spread*, имитирующая разброс при стрельбе.

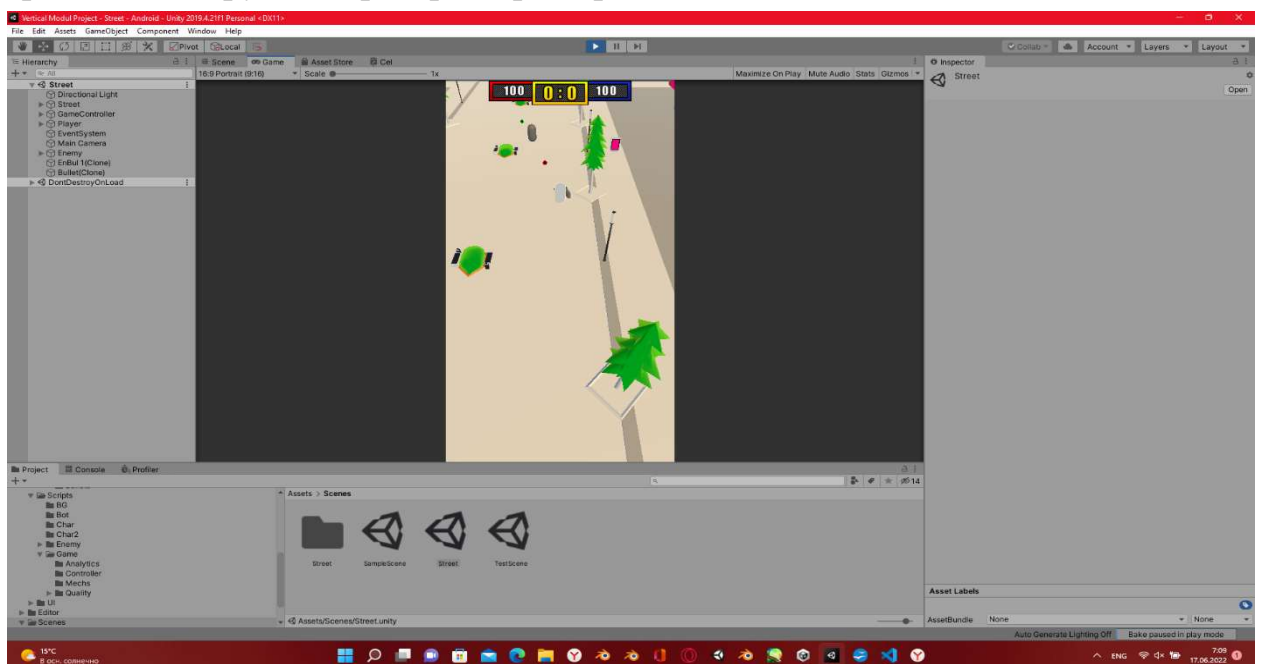


Рисунок 15 – Атака врага игроком

Выстрелы происходят через определённые промежутки времени заданной переменной *ReloadTime* с помощью корутины – действия,

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		56

совершаемого с определенной периодичностью:

```
Invoke(nameof(AttackReset),_reloadRime);
```

Поведение врага на карте определяется компонентами Navigation Mesh Agent и EnemyAI.cs. определяет радиус обнаружения препятствий, максимальный угол склона и высоту порога, по которым враг может перемещаться, прокладывает возможные пути перемещения, делая невозможным прохождение врага сквозь препятствия, застревания между ними и выход за пределы игрового поля. EnemyAI.cs генерирует вектор перемещения врага в соответствии с полученными с помощью процедур Walking и SearchWalkPoint данными.

```
void Walking()
```

```
{  
    target = null;  
    if (!_walkPointSet) SearchWalkPoint();  
    _agent.SetDestination(_walkPoint);  
    Vector3 WalkPointDist = transform.position - _walkPoint;  
    if (WalkPointDist.magnitude < 1f) _walkPointSet = false;  
}
```

```
void SearchWalkPoint()
```

```
{  
    float RandZ = Random.Range(-_walkPointRange,_walkPointRange);  
    float RandX = Random.Range(-_walkPointRange,_walkPointRange);  
    _walkPoint = new Vector3(transform.position.x +  
    RandX,transform.position.y,transform.position.z + RandZ);  
    if (Physics.Raycast(_walkPoint, -transform.up,2f,groundLayer))  
    _walkPointSet = true;
```

					ПГУ 2.090501.03.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

}

Обнаружение персонажа происходит с помощью встроенного физического движка, область действия которого задана радиусом, запускающего алгоритм преследования и атаки цели. Алгоритм преследования создаёт кратчайший путь до персонажа, опираясь на полученные с помощью компонента NavMeshAgent данные об окружении. Процедура атаки врага идентична процедуре атаки персонажа.

И у персонажа, и у врага есть целочисленные переменные, отвечающие за здоровье и уменьшающие своё значение попадании снаряда. При достижении переменной значения 0 и меньше, объект «персонаж» или «враг» убирается с карты.

Подсчет количества здоровья и побед играющих команд (игроков) производится с помощью класса Score. Каждый раз, когда игрок теряет или восстанавливает здоровье или его побеждает игрок другой команды, класс Score фиксирует это и выводит обновленное значение на экран. Игра завершается, когда одна из команд наберет три балла.

За декорирование карты и размещение точек возрождения монстров отвечает класс StreetConfig. В качестве переменных этого класса выступают координаты пространства, которые в целях украшения карты в начале игры, заполняются декоративными элементами. Когда игра начинается, на месте заданных точек генерируются декоративные объекты (здания, скамейки, кусты, деревья).

					ПГУ 2.090501.03.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

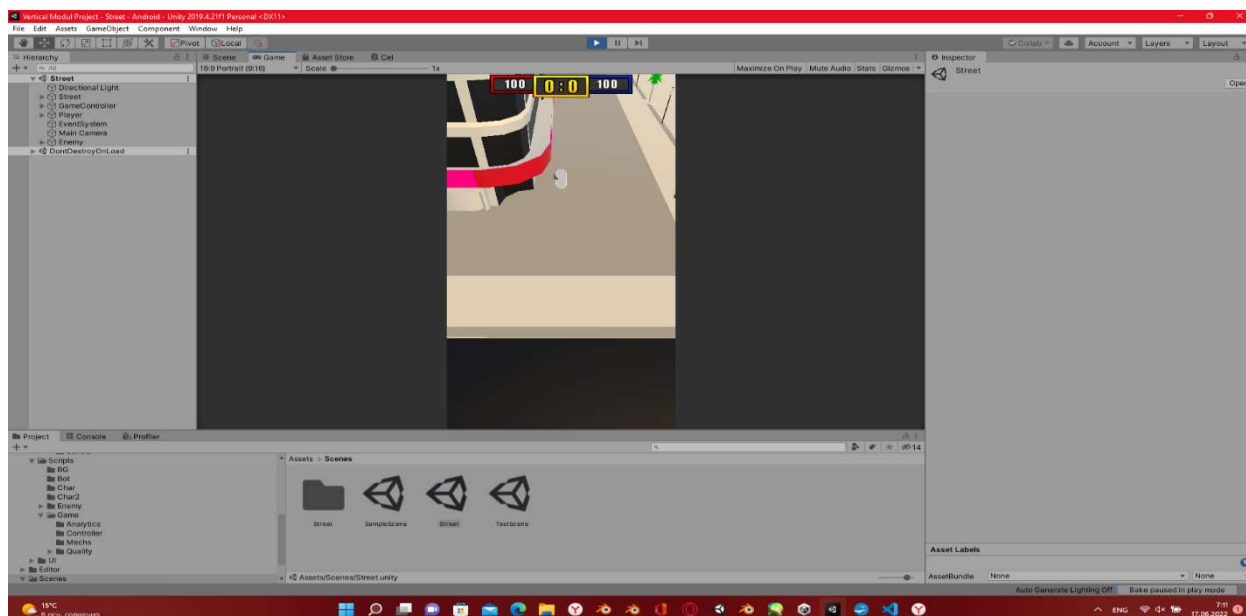


Рисунок 16 – Заполнение карты декором

Таким образом, в ходе реализации учебного проекта были продемонстрированы каталоги приложения: построена диаграмма классов и файловая структура игрового мобильного приложения. Были описаны методы некоторых ключевых классов, описана реализация интерфейса. Для тестирования учебного проекта было проведено прототипирование. Все требования этого этапа были выполнены успешно. Кроме этого, была описана подготовка объектов к игре с помощью программы Blender.

					ПГУ 2.090501.03.001 ПЗ	Лист
						59
Изм.	Лист	№ докум.	Подпись	Дата		

3 Технология оптимизации графики игры с помощью шейдинга в Unity3D

Unity – игровой движок, разработанный Unity Technologies, на котором можно разработать игры и приложения для любых устройств. Unity поддерживает такие языки как Java и C#, имеет собственный набор ядер почти для всех платформ, оснащён собственным магазином ассетов для быстрого прототипирования игр, поддерживает большое количество пакетов расширения. Разработка приложений ведётся для платформ Windows, MacOS, Wii, Apple iOS, Android, PS3 и Xbox360.

Основной интерфейс состоит из нескольких частей:

- окно отображаемой в игре сцены;
- окно иерархии расположенных в нем объектов;
- окно используемых в проекте ассетов;
- окно инспектора для их настройки.

Ассетом может быть абсолютно любой файл, свойства которого можно настроить и использовать в проекте. Это может быть рисунок, модель, скрипт, шейдер, материал или скомбинированная из них конструкция – префаб. В инспекторе ассетом можно настраивать различные свойства, способы обработки и выводы. Также инспектор позволяет настраивать размещённые в пространстве сцены и объекты, назначая им положение и размер, вращая их или назначая им компоненты, расширяющие их значимость в игре и позволяющие настраивать им дополнительные свойства. Существует двенадцать групп компонентов – дополненная реальность, аудио, эффекты, слои, физика, навигация и др. Скрипты, приведённые в качестве ассетов и написанные пользователем, также являются компонентами.

Для создания необычных решений иногда недостаточно встроенного в Unity инструментария, и, тогда на помощь разработчикам приходят пакеты

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		60

расширений. Чтобы добавить пакет расширений, нужно открыть меню windows и выбрать пункт Package Manager. В открывающемся окне следует найти нужную опцию, нажать на кнопку install и, подождав окончания установки, приступить к использованию.

Некоторые пакеты просто добавляют наборы компонентов, некоторые могут обрабатывать добавленные в игру ассеты, (например, sprite 2D, позволяющий обрабатывать двухмерные изображения и атласы спрайтов для дальнейшего их использования в качестве кнопок и других элементов интерфейса или скреплять в анимированные изображения). Некоторые могут использоваться для разработки прикладных ассетов, например, ShaderGraph, позволяющий создавать шейдеры – наборы команд для графического процессора устройства – путём визуального программирования. Один из нужных пакетов для создания красивого изображения на любом устройстве является Universal Render Pipeline. Такие программы пишутся на языке HLSL (High Level Shader Language), который был разработан для пакета Direct x9 и Directx11, которые не поддерживаются платформой Android. Для возможности создавать красивые изображения был разработан пакет – рендерер Universal Render Pipeline (программа для обработки объектов изображения с помощью шейдеров) и пакет разработки шейдеров с возможностью их визуализации через этот рендер. Расширение представляет собой небольшую встроенную программу, большая часть которой занимает рабочая область с размещёнными на ней и соединёнными между собой кривыми линиями блоками. При создании нового шейдера на рабочей области имеется только один блок, отвечающий за полученный результат. Этот результат можно увидеть в небольшом окошке, расположенном в правом нижнем углу рабочей области Main Preview. ShaderGraph позволяет при создании шейдера не только оперировать уже имеющимися блоками, но и создавать пользовательские переменные, будь то число, массив чисел, матрица, цвет или текстура. Помимо

					ПГУ 2.090501.03.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

этого, пакет допускает использование собственных функций, написанных на HLSL. Получаемые результаты можно нормализовать (блок Normalize), складывать (блок Add), смешивать (блок Multiply), разделять (блок Subdivide) и многое другое. Некоторые из представленных блоков способны не только изменить цвет и текстуру объекта, но и добавить необычные эффекты, например, Fresnel Effect, добавляющий свечение по краям объекта, анимировать их и даже изменить геометрию объекта. Итоговый результат во многом зависит от смелости и фантазии разработчиков игры.

В качестве примера созданного в ShaderGraph ассета можно рассмотреть шейдер CelShader, который используется в данном проекте как основной. Особенностью такого шейдера является то, что он получает от источника света только цвет и только один вектор направления, что создаёт резкую границу между освещённой и затенённой сторонами, как это часто бывает в мультфильмах и аниме. В качестве входных данных используются переменные с плавающей точкой ShadowSize, ShadowBlend, ShadowIntencity,

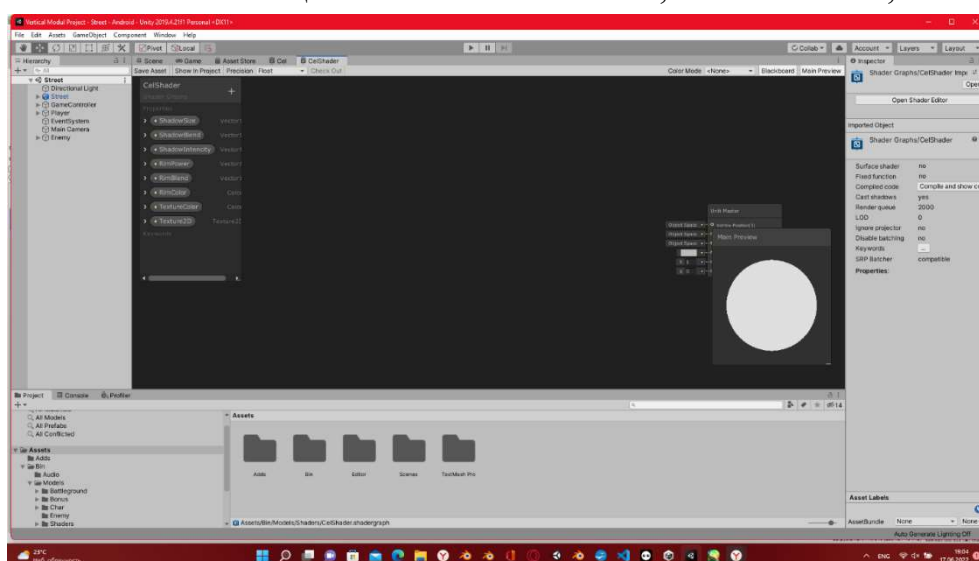


Рисунок 17 – Объявление переменных

RimPower, переменные цвета TextureColor и Rimcolor, переменная графического изображения Texture2D.

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

Для определения координат обрабатываемого шейдером объекта в пространстве сцены добавляется блок Position, в выпадающем меню «Space» выбирается World.

Для получения данных направления, интенсивности цвета на языке HLSL пишется функция и определяются её входные и выходные параметры, а полученный на выходе вектор нормализуется с помощью блоков Normal Vector и Dot Product, позволяющих свести все полученные векторы направления света в один.

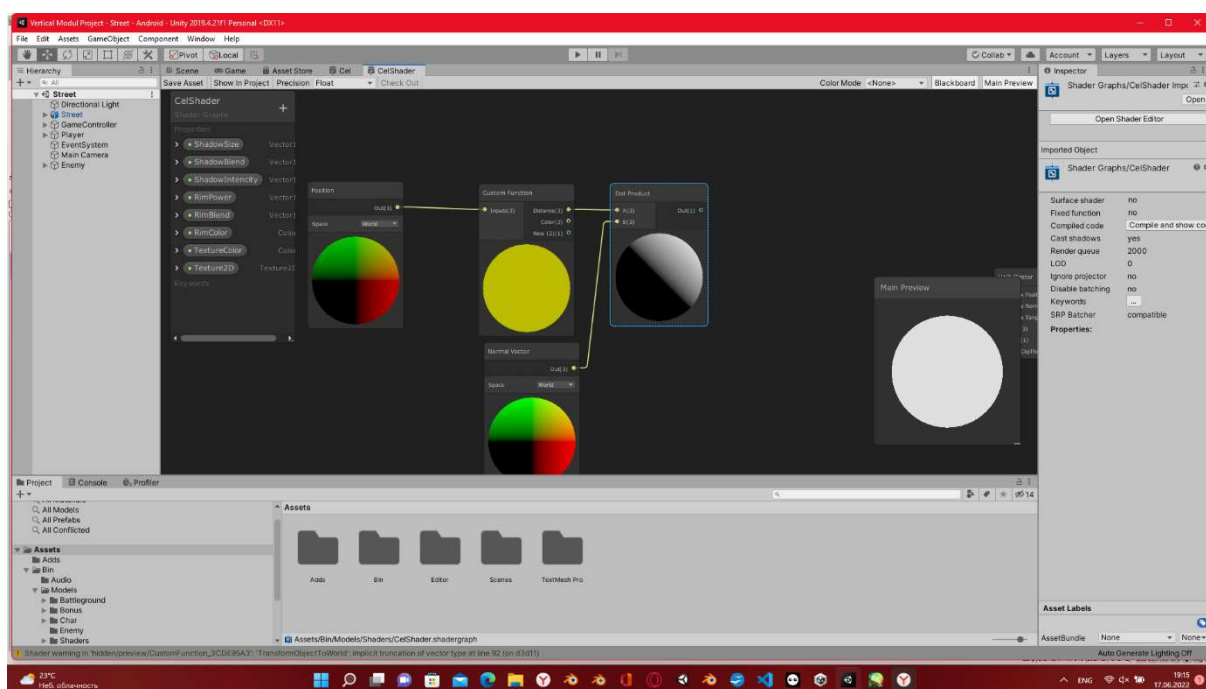


Рисунок 19 – Резкость перехода между освещенной и затенённой стороной

Добавляется блок SmoothStep для контроля резкости между значениями A и B, в нашем случае это будет переход между освещённой и затенённой сторонами, резкость которого настраивается с помощью переменных Shadow Size (размер тени) и Shadow Blend (смесь света и тени).

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		63

Добавляется блок Minus One, делающий все входящие в него положительные значения отрицательными и наоборот, от блока Smooth Step создается ответвление, ведущее к блоку Minus One. Так получается альтернативные вектор источника света, освещающий тело с шейдером с обратной стороны. С помощью блока Multiply происходит смешивание с переменной Shadow Intensity и полученный результат складывается с чистым результатом, исходящим из блока Smooth Step, чтобы получить освещённый объект с одной стороны и слегка затенённый с другой.

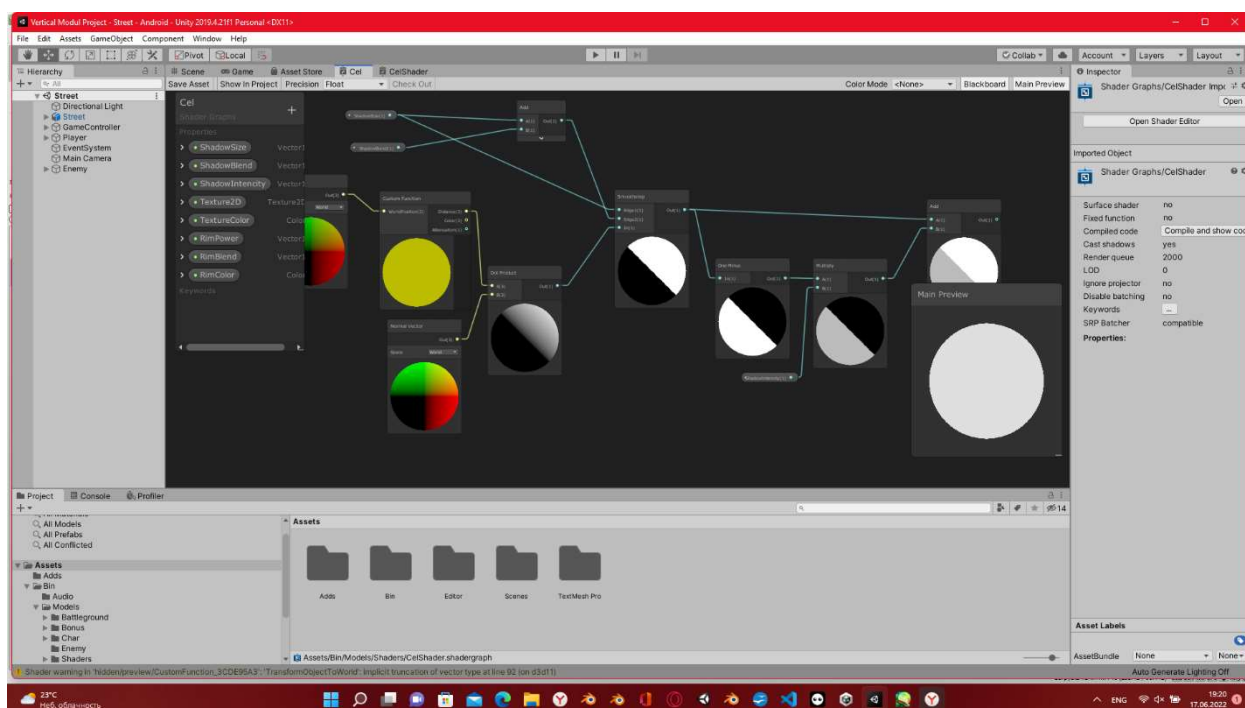


Рисунок 20 – Создание имитации тени

Добавляется переменная изображения Texture 2D и блок - сэмплер Sample Texture 2D. Сэмплер - программа, сопоставляющая точки UV – развертки модели с координатами текстуры, тем самым, размещая участки изображения, использованного в качестве текстуры, на модель. С помощью блока Multiply и переменной TextureColor окрашивается текстура. С помощью ещё одного блока Multiply смешивается окрашенная текстура и получаемые от источника данные. Чтобы находящиеся в тени объекты можно было

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		64

различить, добавляется вся получившаяся конструкция в блок Fresnel Effect, с помощью переменных RimPower и RimBlend, отвечающих за интенсивность и величину подсветки объекта по краям, и блока SmoothStep, отвечающего за резкость перехода, создается настройка резкости, а с помощью переменной цвета RimColor и блока Multiply – настройка цвета.

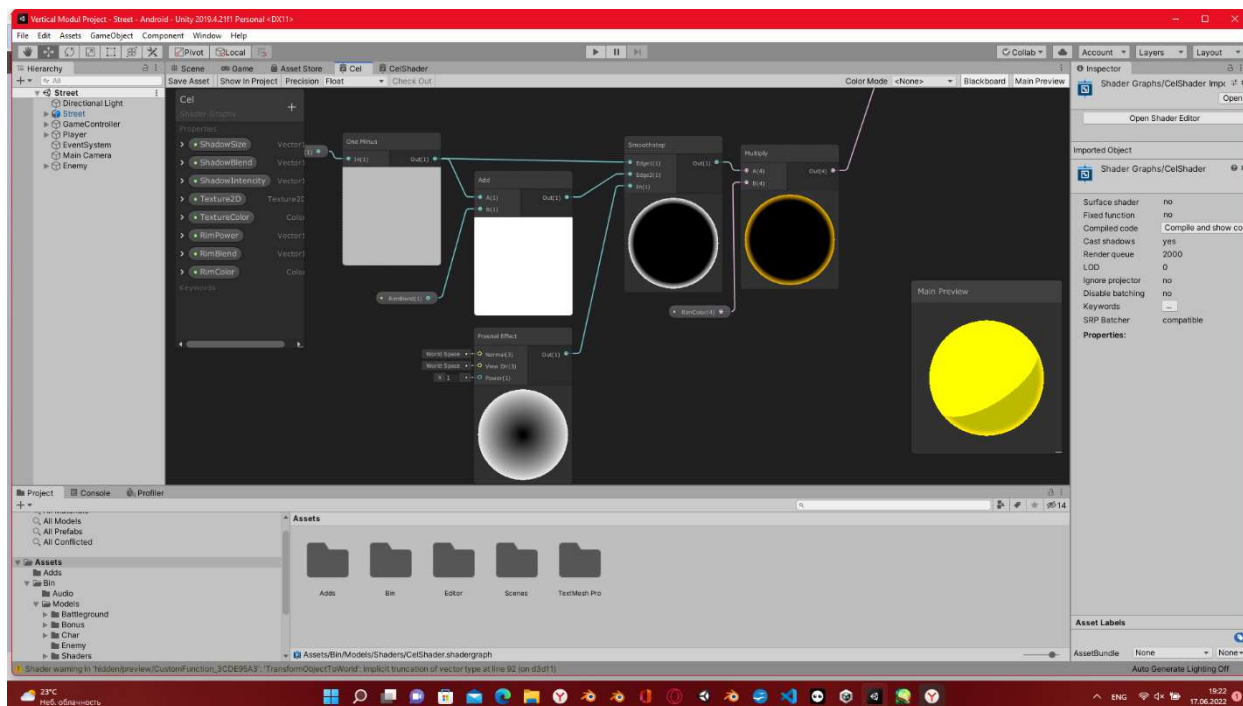


Рисунок 21 – Создание подсветки по краям

С помощью блока Add получается готовый шейдер, результат выводится в графу Color. Сохраняется получившийся шейдер с помощью кнопки Save Asset, готовый шейдер присваивается всем материалам окружения и настраивается.

Таким образом, создается шейдер, принимающий от пользователя значения цвета, текстуры, интенсивности и смешивания тени и подсветки по краям. Данный шейдер берет значения координат, интенсивности и цвета источника освещения сцены, нормализует вектор источника света и создает на его основе имитацию тени, тем самым сильно облегчая работу графическому процессору мобильного устройства.

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		65

4 Экономическое основание разработки

4.1 Расчет затрат на создание программного обеспечения

Трудоемкость разработки программного продукта можно рассчитать исходя из основных этапов. Таким образом, общая формула расчета затрат труда в человеко-часах (чел./ч.) будет следующая:

$$T_{\text{пр}} = T_{\text{п}} + T_{\text{а}} + T_{\text{бс}} + T_{\text{к}} + T_{\text{от}} + T_{\text{д}} \text{ (чел./ч.)} \quad (1)$$

где $T_{\text{п}}$ - трудозатраты на изучение описания задачи, (чел./ч.);

$T_{\text{а}}$ - трудозатраты на разработку алгоритма решения задачи, (чел./ч.);

$T_{\text{бс}}$ - трудозатраты на разработку блок-схемы программы, (чел./ч.);

$T_{\text{к}}$ - трудозатраты на непосредственное написание программы, (чел./ч.);

$T_{\text{от}}$ - трудозатраты на отладку программы на ЭВМ, (чел./ч.);

$T_{\text{д}}$ - трудозатраты на подготовку документации по задаче, (чел./ч.).

Слагаемые трудозатрат определяются через условное количество команд в программных модулях:

$$O = o * C * (1 + \sum_{i=1}^n P) = 100 * 1,6 * (1 + (0,3 + 0,3 + 0,3)) = 304 \text{ (чел./ч.)} \quad (2)$$

где o - предполагаемое число оператораций, $o=100$;

C - коэффициент сложности программы, $C = 1,6$;

P - коэффициент коррекции программы при ее разработке, $P = 0,3$;

$n = 3$ - количество коррекций программы в ходе ее разработки.

Трудозатраты на изучение описания задачи $T_{\text{п}}$ определяются по

$$\text{формуле: } T_{\text{п}} = \frac{O}{48 * K} * B = \frac{304}{48 * 0,8} * 1,5 = 7,6 \text{ (чел./ч)} \quad (3)$$

где K - степень подготовленности исполнителя к выполнению работы, $K=0,8$;

B - коэффициент увеличения затрат по сложности задачи, $B=1,5$.

Трудозатраты на разработку алгоритма решения задачи:

					ПГУ 2.090501.03.001 ПЗ	Лист
						66
Изм.	Лист	№ докум.	Подпись	Дата		

$$T_a = \frac{O}{12 * K} = \left(\frac{304}{12 * 0,8} \right) = 20 \text{ (чел./ч)} \quad (4)$$

Трудозатраты на разработку блок-схемы программы:

$$T_{\text{бс}} = \frac{O}{10 * K} = \left(\frac{304}{10 * 0,8} \right) = 24 \text{ (чел./ч)} \quad (5)$$

Трудозатраты на непосредственное написание программы:

$$T_n = \frac{O}{68 * K} = \left(\frac{304}{68 * 0,8} \right) = 3,57 \text{ (чел./ч.)} \quad (6)$$

Трудозатраты на отладку программы на ЭВМ:

$$T_{\text{отл}} = \frac{O}{8 * K} = \left(\frac{304}{8 * 0,8} \right) = 30,4 \text{ (чел./ч.)} \quad (7)$$

Трудозатраты на подготовку документации:

$$T_d = \frac{O}{15 * K} = \left(\frac{304}{15 * 0,8} \right) = 16 \text{ (чел./ч)} \quad (8)$$

Полученные данные приведены в таблице 2.

Таблица 2 - Трудоемкость работ на разработку приложения

№	Этапы проектирования	В	К	Формула расчёта	Трудоёмкость (чел.-ч)
1	Изучение описания задачи $T_{\text{п}}$	1,5	0,8	$T_{\text{п}} = \frac{O}{48 * K} * B$	7,6
2	Алгоритм решения задачи T_a	1,5	0,8	$T_a = \frac{O}{12 * K}$	20
3	Блок-схема алгоритма $T_{\text{бс}}$	1,5	0,8	$T_{\text{бс}} = \frac{O}{10 * K}$	24
4	Написание программы по блок-схеме T_n	1,5	0,8	$T_n = \frac{O}{68 * K}$	3,57
5	Отладка программы на ЭВМ $T_{\text{отл}}$	1,5	0,8	$T_{\text{отл}} = \frac{O}{8 * K}$	30,4
6	Подготовка	1,5	0,8	$T = \frac{O}{15 * K}$	16

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		67

	документации Т _д				
7	Итого:				102

Таким образом, сумма трудозатрат на разработку игрового мобильного приложения составляет 102 (чел./ч.)

4.2 Расчет цены программного обеспечения

Себестоимость проекта С_{пр} определяется суммированием материальных ресурсов, энергоресурсов, оплаты труда разработчика программного продукта, амортизацией, расходами на подготовку и освоение производства, общехозяйственными расходами. Стоимостная оценка труда разработчиков программного проекта может быть определена по формуле:

$$C_{\text{тр}} = O * T_{\text{пр}} * \left[\left(1 + \frac{P_d}{100} \right) * \left(1 + \frac{P_{\text{сц}}}{100} \right) + \frac{P_{\text{нр}}}{100} \right] \quad (9)$$

где О – средняя заработная плата проектировщика в час, руб./ч.;

T_{пр} - трудоемкость проектирования, чел.-ч.; T_{пр} = 17,75

P_д - процент дополнительной заработной платы; P_д = 10%

P_{сц} - процент отчислений на социальные нужды, включая социальное и медицинское страхование, фонд занятости, пенсионный фонд; P_{сц} = 30%

Средняя заработная плата проектировщика в час может быть определена по формуле:

$$O = \frac{Z}{K_{\text{дн}} * T_{\text{см}}} * K_{\text{раз}} = \frac{58413}{20 * 8} * 1 = 365 \text{ (руб)} \quad (10)$$

где Z_{мар} - средняя заработная плата инженера-проектировщика в месяц, 58413руб.;

K_{дн} - количество рабочих дней в месяце, K_{дн} = 20 (дн.);

K_{раз} - разрядный коэффициент, K_{раз} = 1,00;

T_{см} - продолжительность смены, T_{см} = 8 (часов).

Дополнительный фонд заработной платы – 10%.

					ПГУ 2.090501.03.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

Страховые взносы - 30%.

Теперь определим стоимостную оценку труда инженера- проектировщика за выполнение проекта:

$$C_{\text{тр}} = 365 * 102 * \left[\left(1 + \frac{10}{100} \right) * \left(1 + \frac{30}{100} \right) + \frac{30}{100} \right] = 64408 \text{ руб.}$$

Расчет затрат на техническое обеспечения проекта

Для разработки программного обеспечения необходимы технические средства:

- Монитор 1 шт. - 9500 руб.
- системный блок 1 шт. - 13000 руб.
- клавиатура 1 шт. - 3000 руб.
- мышь 1 шт. - 500 руб.
- принтер— шт. - 5000 руб.
- Windows 10 (лицензионный) — шт. - 15000 руб.
- Unity 3D – бесплатная лицензия
- Blender – бесплатная лицензия

Таким образом, затраты на техническое и программное обеспечение $C_{\text{оф}}$ составляют 41000 руб.

Основные фонды переносят свою стоимость на создаваемую продукцию в виде амортизационных отчислений. Срок использования технических средств 3 года. Затраты на амортизацию:

$$Z_{\text{ам}} = S_{\text{бал}} * \frac{H_{\text{ам}}}{100} \quad (11)$$

$$Z_{\text{ам}} = 41000 * \frac{12,5}{100} = 5125$$

где $H_{\text{ам}} = 12,5\%$;

$S_{\text{бал}} = 41000$ (руб).

Прочие затраты

Затраты на оплату машинного времени при отладке программы

					ПГУ 2.090501.03.001 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

определяются путем умножения фактического времени отладки программы на цену машино-часа арендного времени:

$$З_{\text{мв}} = C_{\text{час}} * t_{\text{эвм}} \quad (12)$$

где $C_{\text{час}}$ - цена машино-часа арендного времени, тен/час = 30 руб;

$t_{\text{эвм}}$ – фактическое время отладки приложения на ЭВМ.

Фактическое время отладки вычисляется по формуле:

$$t_{\text{эвм}} = t_{\text{н}} + t_{\text{до}} + t_{\text{от}} \quad (13)$$

$$t_{\text{эвм}} = 3,57 + 16 + 30,4 = 50$$

$$З_{\text{мв}} = 300 * 50 = 1500$$

Расчет затрат на текущий ремонт.

Затраты на текущий и профилактический ремонт принимаются равными 5% от стоимости ЭВМ

$$З_{\text{тр}} = 0,05 * S_{\text{бал}} \quad (14)$$

$$З_{\text{тр}} = 0,05 * 41000 = 2050$$

Затраты на материалы, необходимые для обеспечения нормальной работы ПЭВМ составляют около 1% от стоимости ЭВМ:

$$З_{\text{мв}} = 0,01 * S_{\text{бал}} \quad (15)$$

$$З_{\text{мв}} = 0,01 * 41000 = 410 \text{ руб.}$$

Итого прочие затраты:

$$З_{\text{пр}} = З_{\text{мв}} + З_{\text{тр}} + З_{\text{м}} = 1500 + 2050 + 410 = 3960 \text{ (руб)}$$

Затраты на материальные ресурсы

1. Литература - 2000 руб.

2. Канцтовары – 200 руб.

Итого: 2200 руб.

					ПГУ 2.090501.03.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

Расчет затрат на энергоресурсы рассчитывается по формуле:

$$C_{\text{эп}} = N * n * Ц * t = 0,70 * 1 * 3,76 * 50 = 132 \text{ (руб.) (16)}$$

где N - мощность технических средств - 0,70 (кВт);

n = 1 - количество используемых технических средств;

Ц - цена одного киловатта электроэнергии 3,76 (руб.);

t - время работы в часах - 50 (часов).

Все затраты представлены в таблице 3.

Таблица 3 - Калькуляция себестоимости и отпускной цены

№	Затраты	Сумма (руб)
1	Материалы	2200
2	Энергоресурсы	132
3	Основная заработная плата	64000
4	Дополнительная заработная плата	408
5	Амортизационные нужды	5125
6	Прочие затраты	3960
7	Расходы на подготовку и освоиваивание производства 0,5% от $\sum 1+6$	3791
8	Общехозяйственные расходы 3% от $\sum 1+7$	2388
9	Производственная себестоимость 1-8	82004
10	Прибыль предприятия 20% от 9	16400
11	Цена продукции 9+10	98404
12	НДС 20% от 11	19680
13	Отпускная цена 11 + 12	118085
14	Налог на прибыль 20% от (11-9)	2501
15	Чистая прибыль 11-9-14	13900

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		71

Цена одной копии программного продукта определяется по формуле:

$$C_k = C_{\text{прод}} / N \quad (17)$$

где $C_{\text{прод}}$ - цена продукции;

N - количество копий.

Цена одной копии составит:

$$C_k = C_{\text{прод}} / N = 118085 / 1 = 118085 \text{ (руб)}$$

Таким образом, себестоимость программного продукта составит 82004 рублей.

5 Безопасность жизнедеятельности

5.1 Неблагоприятные факторы воздействия

Одним из самых вредных факторов, воздействующих на оператора при эксплуатации ПК является ухудшение зрения из-за перенапряжения глаз.

При работе на ПК функция зрительного анализатора сопровождается постоянным и значительным напряжением. Причиной является такой принцип чтения информации, когда ПК оператор не наклоняет голову как при обычном чтении, а смотрит прямо на экран. Причем источником текста является люминесцирующее вещество экрана, поэтому пользователь не считывает отраженный текст, а смотрит непосредственно на источник света, что вынуждает глаза и орган зрения в целом работать в стрессовом режиме длительное время.

К физическим вредным и опасным факторам относятся повышенный уровень инфракрасного и электромагнитного излучения, повышенный уровень запыленности воздуха рабочей зоны и статического электричества, пониженное содержание отрицательных аэроионов в воздухе рабочей зоны и повышенное содержание положительных аэронов, повышенный уровень

					ПГУ 2.090501.03.001 ПЗ	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		

ослепленности и блескости, неравномерность распределения яркости в поле зрения, повышенная яркость светового изображения.

К химическим вредным и опасным факторам относят: повышенное содержание в воздухе рабочей зоны двуокиси углерода, озона, аммиака, фенола и формальдегида.

К психофизиологическим вредным и опасным факторам относят напряжение зрения и внимания, интеллектуальные, эмоциональные и длительные статические нагрузки, монотонность труда, большой объем информации, обрабатываемый в единицу времени, нерациональная организация рабочего места.

5.2 Противодействие вредным факторам

Для предупреждения утомляемости операторов ПК рекомендуется организовывать рабочую смену путём чередования работы с использованием компьютера и без него. Если же работа требует постоянного действия с монитором, рекомендуется организовывать перерывы на 10-15 минут через каждые 45-60 минут работы.

Для снижения нервно – эмоционального напряжения, устранения влияния гиподинамии и гипокинезии необходимо устраивать физкультурные паузы с целенаправленным воздействием на ту или иную группу мышц.

5.2.1 Организация безопасных условий труда на рабочем месте

При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора) должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов — не менее 1,2

					ПГУ 2.090501.03.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

м. Согласно СанПиН 2.2.2/2.4.134003, площадь на одно рабочее место пользователей ПЭВМ с ВДТ на базе электроннолучевой трубки (ЭЛТ) должна составлять не менее 6 м².

Если компьютер оснащен жидкокристаллическим или плазменным монитором, площадь может составлять 4,5 м². При использовании ПЭВМ с ВДТ на базе ЭЛТ (без вспомогательных устройств - принтер, сканер и др.), отвечающих требованиям международных стандартов безопасности компьютеров, с продолжительностью работы менее 4-х часов в день допускается минимальная площадь 4,5 м² на одно рабочее место пользователя (взрослого и учащегося высшего профессионального образования).

Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам и естественный свет падал преимущественно слева. Рабочие места с ПЭВМ при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, рекомендуется изолировать друг от друга перегородками высотой 1,5-2,0 м. Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов. Клавиатуру следует располагать на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю, или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы. Для удобства считывания документов следует применять подвижные подставки (пюпитры), которые должны размещаться в одной плоскости и на одной высоте с экраном. Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) следует выбирать с учетом

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		74

роста пользователя, характера и продолжительности работы с ПЭВМ. Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья. При этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию. Поверхности сиденья, спинки и других элементов стула (кресла) должны быть полумягкими, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений. Рабочее место пользователя ПЭВМ следует оборудовать подставкой для ног, имеющей ширину не менее 300 мм, глубину не менее 400 мм, регулировку по высоте в пределах до 150 мм и по углу наклона опорной поверхности подставки до 20 градусов. Поверхность подставки должна быть рифленой и по переднему краю бортик высотой 10 мм.

5.2.2 Расчёт освещенности на рабочем месте

В помещении, где находятся компьютеры, окна рекомендуется ориентировать на север и северо-восток. Если работа происходит в помещениях без естественного освещения, то устанавливается искусственное освещение в соответствии с требованиями нормативных документов: освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500 лк. Оно не должно создавать бликов на экране. Освещенность самого экрана не должна быть более 300 лк. Следует ограничивать прямую блескость от источников освещения, при этом яркость светящихся поверхностей вокруг должна быть не более 200кд/м².

Чтобы рассчитать число и мощность светильников, нужных для обеспечения освещенности для операторов вычислительного центра, исходят из того, что помещение имеет общую систему освещения, при этом типом

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		75

источников света являются люминесцентные лампы прямого света мощностью 35-65 Вт типа ЛБ. Яркость светильников в зоне углов излучения от 50 до 90° с вертикалью в продольной и поперечной плоскостях должна составлять не более 200 кд/м², защитный угол светильников должен быть не менее 40°. Общее освещение следует выполнять в виде сплошных и прерывистых линий светильников, расположенных сбоку от рабочих мест, параллельно линии зрения пользователя при рядом расположении ПК. Расчет освещённости осуществляется методом коэффициента использования светового потока (отношение светового потока, падающего на расчётную поверхность к суммарному потоку всех ламп). Расположим светильники в два ряда параллельно длинной стороне помещения, имеющего размеры 8×4м и высотой 3м. Светильники в рядах расположены с зазором в 1,5м, расстояние между рядами 1,5м установлены на потолке. Высота рабочих мест составляет 0,75 м, поэтому высота подвеса светильников над рабочей поверхностью будет равна 2, 25 м.

В соответствии со СНиП 23-05-93 освещенность на поверхности стола в зоне размещения документа должна быть 300-500лк. Световой поток группы ламп вычисляется по следующей формуле:

$$\Phi_{\text{св}} = (E_{\text{н}} * S * k_3 * Z) / (N * \eta),$$

где $E_{\text{н}}$ – требуемый нормативный уровень освещенности рабочей поверхности (300лк);

S – площадь помещения (32 м²);

$k_3=1,5$ -коэффициент запаса;

$Z = 1, 1$ – коэффициент неравномерности освещения;

N – количество светильников;

η – коэффициент использования светового потока, выбирается из таблиц в зависимости от типа светильника, размеров помещения, коэффициентов отражения стен $r_{\text{с}}$ и потолка $r_{\text{п}}$, показателя помещения i ;

					ПГУ 2.090501.03.001 ПЗ	Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

$r_{\text{п}} = 0.7$ (цвет поверхности - белый)

$r_{\text{с}} = 0.5$ (цвет поверхности – светлый)

Количество светильников рассчитывается по следующей формуле:

$$N = S/h^2 = 32 / (2.25)^2 = 6,3 \text{ (шт)}$$

Показатель помещения определяется по формуле:

$$i = (A \cdot B) / ((A+B) \cdot h) = (8 \cdot 4) / ((8+4) \cdot 2.25) = 1,18$$

На основании значений $r_{\text{п}}$, $r_{\text{с}}$, и i по таблице определяется $h=0,42$ и рассчитывается световой поток группы ламп светильника:

$$\Phi_{\text{св}} = (300 \cdot 32 \cdot 1,5 \cdot 1,18) / (6 \cdot 0,42) = 6743 \text{ лм (для 4 ламп); 1686 лм - световой поток одной лампы}$$

Этому значению соответствует лампа ЛД40 мощностью 40 Вт со световым потоком 2100 лм

Общая потребляемая мощность:

$$P_0 = 24 \cdot 40 = 960 \text{ Вт}$$

Рассчитывается запас мощности с учетом возможных потерь до 25%:

$$P_p = 960 \cdot 0,25 = 240 \text{ Вт}$$

Значит общая мощность сети должна быть:

$$P = P_0 + P_p = 960 + 240 = 1200 \text{ Вт}$$

Расчёт освещённости позволяет обеспечить возможность нормальной деятельности операторов в условиях отсутствия или недостаточности естественного освещения, сохранить их зрение и повысить производительность их труда.

5.2.3 Параметры микроклимата

В помещениях с ПЭВМ должна проводиться ежедневная влажная уборка и проветривание после каждого часа работы. Помещение следует оборудовать системами отопления, вентиляции и кондиционирования

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		77

воздуха. Оптимальные параметры микроклимата в помещении с компьютерами: температура воздуха – от 19 до 21;

Относительная влажность от 62 до 55%

Скорость движения воздуха – не более 0,1 м/с.

					ПГУ 2.090501.03.001 ПЗ	Лист
						78
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения выпускной квалификационной работы был получен материал, анализ которого позволил сделать вывод, что операционная система Android в настоящее время является наиболее востребованной на рынке программного обеспечения.

В первой главе выполнен обзор операционной системы Android, определен ряд её преимуществ для использования под игровое мобильное приложение. Затем был рассмотрен ряд графических технологий специализирующихся на создания игр и выполнено обоснование выбора средств проектирования собственного игрового приложения. Далее были выделены основные этапы проектирования приложения, обоснованы минимальные системные требования для воспроизведения мобильного приложения. Был выполнен обзор актуальных программ для реализации игровых приложений и сделан выбор в пользу Unity3D.

Вторая глава содержит описание разработанного приложения: его идею, концепцию, пользовательский интерфейс, структуру приложения и логику разработки игрового мобильного приложения.

В третьей главе дан обзор игрового движка Unity3D, рассмотрены его рабочие области и возможности управления расширениями и настройками объектов, находящимися на сцене, описана технология оптимизации графики игры с помощью шейдинга.

Четвертая глава посвящена экономической составляющей проекта. Описаны расчёты затрат на разработку, произведён расчёт начальной цены приложения.

Пятая глава посвящена вопросам безопасности жизнедеятельности при работе оператора с ЭВМ.

Таким образом, в ходе выполнения дипломного проекта все задачи были выполнены и цель достигнута.

					ПГУ 2.090501.03.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		79

Список использованных источников

1. Дейтел П., Дейтл Х, Дейтл Э. Android для разработчиков. 2-е изд.: Питер, 2015. 384 с.
2. Дикинсон К. Оптимизация игр в Unity 5. – М.: Издательство ДМК, 2017. – 306с.
3. Ларкович С.Н. Unity на практике. – Санкт-Петербург, Издательство НиТ, 2021. -272с.
4. Мэннинг Д., Батфилд-Эддисон П. Unity для разработчика. Мобильные мультиплатформенные игры. – Издательство Питер, 2018. – 352с.
5. СанПиН 2.2.4.335916 «Санитарно-эпидемиологические требования к физическим факторам на рабочих местах», утвержденные Постановлением Главного государственного санитарного врача РФ от 21.06.2016 № 81.
6. СанПиН 2.2.2/2.4.134003 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» (утверждены Постановлением Главного государственного санитарного врача РФ от 03.06.2003 № 118).
7. Lee Zhi Eng Building a Game with Unity and Blender. - Packt Publishing, 2015, - 250p.
8. Schell J. The Art of Game Design. – Morgan Kaufmann Publishers. 2015. - 518p.
9. Thorn A. Pro. Unity Game Development with C#. - Apress, 2014, - 335p.
10. Этапы создания компьютерной игры // Этапы создания компьютерной игры (Сайт GameisArt.ru) URL: http://gameisart.ru/game_dev_create.html

					ПГУ 2.090501.03.001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А

(Обязательное)

Мобильное игровое приложение

Листинг программы

					ПГУ 2.090501.03.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		

Код CharMove.cs

```
using UnityEngine;
```

```
public class CharMove : MonoBehaviour
```

```
{
```

```
    //MovementStats=====
```

```
    [SerializeField]private float _charMass;//Масса
```

```
    [SerializeField]private float _charSpd;//Линейная скорость
```

```
    [SerializeField]private float _charAngSpd;//Угловая скорость
```

```
    //TurretStats=====
```

```
    [SerializeField, Range(10f, 200f)] private float _targetingRange = 50f;//Радиус  
    обзора
```

```
    public Transform target;//Атакуемая цель
```

```
    private Collider[] _targets;//Враги в радиусе обзора
```

```
    public LayerMask EnemyLayer;//Слой врагов
```

```
    public bool _enemySight;//Враг виден
```

```
    //ShootingStats=====
```

```
    public bool _attackHold;//Перезарядка
```

```
    public float _reloadRime;//Время перезарядки
```

```
    public GameObject Bullet;//Снаряд
```

					ПГУ 2.090501.03.001 ПЗ	Лист
						82
Изм.	Лист	№ докум.	Подпись	Дата		

```

public Transform BulletSpawnPoint;//место выхода снарядов

[Range(0f,10f)]public float spread;//Разброс при стрельбе

public float BulletSpeed;//Скорость снаряда

//Stats=====

[SerializeField]private int HP;//Здоровье

[SerializeField]private int ATK;//Атака

[SerializeField]private int DEF;//Защита

public int maxHP = 100;//Максимальное количество хп

private bool _alive = true;//Жив ли игрок

//CalculatedStats=====

public float _horz,_vert,_grav;

private Vector3 _charPos;//Позиция в мире

private Quaternion _charRot;//Направление позиции в мире

//Constans=====

private string _horzDir = "Horizontal";

private string _vertDir = "Vertical";

//Components=====

private CharacterController _PCC;//Контроллер персонажа

private Animator _PA;//Аниматор персонажа

//JoypadComponent=====

public FloatingJoystick FJController;//Подключение джойстика

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						83
Изм.	Лист	№ докум.	Подпись	Дата		

// Start is called before the first frame update

void Start()

{

 _PCC = GetComponent<CharacterController>();

 _PA = GetComponent<Animator>();

}

// Update is called once per frame

void Update()

{

 Gravity();

 Movement();

 EnemyTargeting();

}

//Physics=====

=====

void Gravity()

{

 if (!_PCC.isGrounded)

 {

					ПГУ 2.090501.03.001 ПЗ	Лист
						84
Изм.	Лист	№ докум.	Подпись	Дата		

```

    _grav -= _charMass * Time.deltaTime;

    return;

}

_grav = -1f;

}

```

```

//Movement=====
=====

```

```

void Movement()

{

    if (_PCC.isGrounded)

    {

        target = null;

        _charPos = Vector3.zero;

        _charPos.x = FJController.Horizontal;

        _charPos.z = FJController.Vertical;

        _charPos = _charPos.normalized;

        Vector3 newDirect =

Vector3.RotateTowards(transform.forward, _charPos, _charAngSpd, 0);

        transform.rotation = Quaternion.LookRotation(newDirect);

```

					ИГУ 2.090501.03.001 ПЗ	Лист
						85
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }

    _charPos.y = _grav;

    _PCC.Move(_charPos * _charSpd * Time.deltaTime);

}

//Turret=====

=====

void EnemyTargeting()

{

    if (_charPos.magnitude > 1f) return;

    _enemySight =
Physics.CheckSphere(transform.position, _targetingRange, EnemyLayer);

    if (_enemySight) EnemyAttacking();

}

bool TargetLock()

{

    if (target == null) return false;

    Vector3 a = transform.localPosition;

    Vector3 b = target.localPosition;

    if (Vector3.Distance(a,b) > _targetingRange)

    {

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						86
Изм.	Лист	№ докум.	Подпись	Дата		

```

        target = null;

        return false;

    }

    return false;

    //Что делать, если враг покинет зону поражения
}

void EnemyAttacking()

{

    _targets =
Physics.OverlapSphere(transform.position,_targetingRange,EnemyLayer);

    target = _targets[0].GetComponent<Transform>();

    transform.LookAt(target);

    if (!_attackHold)

    {

        //Использование снарядов для атаки

        Shoot();

        //Reloading script

        _attackHold = true;

        Invoke(nameof(AttackReset),_reloadRime);

    }

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						87
Изм.	Лист	№ докум.	Подпись	Дата		

```

}

void Shoot()
{

    Vector3 TargetPoint = target.position;

    Vector3 directionWithoutSpread = TargetPoint - BulletSpawnPoint.position;

    float XSpread = UnityEngine.Random.Range(-spread,spread);

    float Yspread = UnityEngine.Random.Range(-spread,spread);

    Vector3 directionWithSpread = directionWithoutSpread + new
Vector3(XSpread,Yspread,0);

    GameObject CurrentBullet =
Instantiate(Bullet,BulletSpawnPoint.position,BulletSpawnPoint.rotation);

    CurrentBullet.transform.forward = directionWithSpread.normalized;

    CurrentBullet.GetComponent<Rigidbody>().AddForce(directionWithSpread.norm
alized * BulletSpeed,ForceMode.Impulse);

}

void AttackReset()
{

    _attackHold = false;

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						88
Изм.	Лист	№ докум.	Подпись	Дата		

}

//Stats=====

=====

void PlayerDamaged(int Damage)

{

HP -= Damage;

if (HP <= 0)

{

_alive = false;

Invoke(nameof(PlayerDead),2f);

}

}

void PlayerHealed(int Heal)

{

//Отхил игрока

HP = HP + Heal;

if (HP > maxHP) HP = maxHP;

}

					ПГУ 2.090501.03.001 ПЗ	Лист
						89
Изм.	Лист	№ докум.	Подпись	Дата		

```

void PlayerDead()

{

    Destroy(gameObject);

}

}

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

Код EnemyAI.cs

```
using UnityEngine;
```

```
using UnityEngine.AI;
```

```
public class EnemyAI : MonoBehaviour
```

```
{
```

```
    private NavMeshAgent _agent;
```

```
    public LayerMask playerLayer,groundLayer;
```

```
    public Transform target;
```

```
    public Collider[] targets;
```

```
    //public bool targetLock;
```

```
    private Vector3 _walkPoint;
```

```
    private bool _walkPointSet;
```

```
    private bool _alive = true;
```

```
    private bool _attackHold;
```

```
    [SerializeField]private float _attackHoldTime;
```

					ПГУ 2.090501.03.001 ПЗ	Лист
						91
Изм.	Лист	№ докум.	Подпись	Дата		

```
[SerializeField, Range(10f, 300f)]private float _walkPointRange = 100f;
```

```
[SerializeField, Range(10f, 300f)]private float _sightRange = 100f;
```

```
[SerializeField, Range(10f, 300f)]private float _attackRange = 100f;
```

```
bool _sightPlayer;
```

```
bool _attackPlayer;
```

```
[SerializeField]private int ATK;
```

```
[SerializeField]private int HP;
```

```
public GameObject Bullet;
```

```
public Transform BullSpawnPoint;
```

```
public float bulletSpeed;
```

```
[Range(0,10)]public float spread = 2f;
```

```
// Start is called before the first frame update
```

```
void Start()
```

```
{
```

```
    _agent = GetComponent<NavMeshAgent>();
```

```
}
```

					ИГУ 2.090501.03.001 ИЗ	Лист
						92
Изм.	Лист	№ докум.	Подпись	Дата		

```

// Update is called once per frame

void Update()

{
    StateChanger();
}

void StateChanger()

{
    _sightPlayer =
Physics.CheckSphere(transform.position,_sightRange,playerLayer);

    _attackPlayer =
Physics.CheckSphere(transform.position,_attackRange,playerLayer);

    if (!_sightPlayer && !_attackPlayer) Walking();

    if (_sightPlayer && !_attackPlayer) PlayerChasing();

    if (_sightPlayer && _attackPlayer) PlayerAttacking();

}

void Walking()

{
    target = null;

    if (!_walkPointSet) SearchWalkPoint();

    _agent.SetDestination(_walkPoint);

```

					ИГУ 2.090501.03.001 ИЗ	Лист
						93
Изм.	Лист	№ докум.	Подпись	Дата		

```

Vector3 WalkPointDist = transform.position - _walkPoint;

if (WalkPointDist.magnitude < 1f) _walkPointSet = false;

}

void SearchWalkPoint()

{

    float RandZ = Random.Range(-_walkPointRange,_walkPointRange);

    float RandX = Random.Range(-_walkPointRange,_walkPointRange);

    _walkPoint = new Vector3(transform.position.x +
RandX,transform.position.y,transform.position.z + RandZ);

    if (Physics.Raycast(_walkPoint, -transform.up,2f,groundLayer))
_walkPointSet = true;

}

void PlayerChasing()

{

    if (target == null) targets =
Physics.OverlapSphere(transform.position,_sightRange,playerLayer);

    target = targets[0].GetComponent<Transform>();

    _agent.SetDestination(target.position);

}

```

					ИГУ 2.090501.03.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		94

```

bool TargetLock()

{

    if (target == null) return false;

    Vector3 a = transform.localPosition;

    Vector3 b = target.localPosition;

    if (Vector3.Distance(a,b) > _sightRange)

    {

        target = null;

        return false;

    }

    return false;

}


void PlayerAttacking()

{

    _agent.SetDestination(transform.position);

    transform.LookAt(target);

    if (!_attackHold)

    {

        //Скрипт атаки

        //var Bull = Instantiate(Bullet,BullSpawn.position,BullSpawn.rotation)

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

```

    Shoot();

    //Перезарядка
    _attackHold = true;

    Invoke(nameof(AttackingReset),_attackHoldTime);

}

}

void Shoot()
{
    Vector3 TargetPoint = target.position;

    Vector3 direcionWithoutSpread = TargetPoint - BullSpawnPoint.position;

    float XSpread = UnityEngine.Random.Range(-spread,spread);

    float Yspread = UnityEngine.Random.Range(-spread,spread);

    Vector3 directionWithSpread = direcionWithoutSpread + new
Vector3(XSpread,Yspread,0);

    GameObject CurrentBullet =
Instantiate(Bullet,BullSpawnPoint.position,BullSpawnPoint.rotation);

    CurrentBullet.transform.forward = directionWithSpread.normalized;

    CurrentBullet.GetComponent<Rigidbody>().AddForce(directionWithSpread.norm
alized * bulletSpeed,ForceMode.Impulse);

}

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						96
Изм.	Лист	№ докум.	Подпись	Дата		

```
void AttackingReset()
```

```
{
```

```
    _attackHold = false;
```

```
}
```

```
void TakeDamage(int Damage)
```

```
{
```

```
    HP -= Damage;
```

```
    if (HP <= 0)
```

```
    {
```

```
        _alive = false;
```

```
        Invoke(nameof(EnemyDead),2f);
```

```
    }
```

```
}
```

```
void EnemyDead()
```

```
{
```

```
    Destroy(gameObject);
```

```
}
```

					ПГУ 2.090501.03.001 ПЗ	Лист
						97
Изм.	Лист	№ докум.	Подпись	Дата		

Код Joystick.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
```

```
public enum JoyStickDirect{Horizontal,Vertical,Both}
```

```
public class Joystick :
MonoBehaviour,IDragHandler,IPointerDownHandler,IPointerUpHandler
{
    public JoyStickDirect JSDir = JoyStickDirect.Both;
    public RectTransform Back;
    public RectTransform Handle;
    [Range(0,2f)]public float HandleLimit = 1f;
    private Vector2 _input = Vector2.zero;

    public float Vertical {get{return _input.y;}}
    public float Horizontal {get{return _input.x;}}
    Vector2 JoyPos = Vector2.zero;

    public void OnPointerDown(PointerEventData PED)
    {
        Back.gameObject.SetActive(true);
        OnDrag(PED);
    }
}
```

					ИГУ 2.090501.03.001 ИЗ	Лист
						98
Изм.	Лист	№ докум.	Подпись	Дата		

```

        JoyPos = PED.position;
        Back.position = PED.position;
        Handle.anchoredPosition = Vector2.zero;
    }

    public void OnDrag(PointerEventData PED)
    {
        Vector2 JoyDir = PED.position - JoyPos;
        _input = (JoyDir.magnitude > Back.sizeDelta.x/2f) ? JoyDir.normalized
: JoyDir/(Back.sizeDelta.x/2f);
        if (JSDir == JoyStickDirect.Horizontal) _input = new
Vector2(_input.x,0f);
        if (JSDir == JoyStickDirect.Vertical) _input = new Vector2(0f,_input.y);
        Handle.anchoredPosition = (_input * Back.sizeDelta.x/2f) *
HandleLimit;
    }

    public void OnPointerUp(PointerEventData PED)
    {
        Back.gameObject.SetActive(false);
        _input = Vector2.zero;
        Handle.anchoredPosition = Vector2.zero;
    }
}

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						99
Изм.	Лист	№ докум.	Подпись	Дата		

Код StreetConfig.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class StreetConfig : MonoBehaviour
{
    [SerializeField]private GameObject[]
    _LongVert,_ShortVert,_LongShop,_ShortShop;
    [SerializeField]private GameObject[]
    _LongVertPlace,_ShortVertPlace,_LongShopPlace,_ShortShopPlace;
    [SerializeField]private GameObject[] _EnemySpawner;
    [SerializeField]private int _WaveNum;
    [SerializeField]private int _Diff;
    public bool Recharge = true;
    // Start is called before the first frame update
    void Start()
    {
        int i,s;
        for (i = 0;i < _LongVertPlace.Length; i++)
        {
            s = UnityEngine.Random.Range(0,_LongVert.Length);
            Instantiate(_LongVert[s],_LongVertPlace[i].transform);
        }
        for (i = 0;i < _ShortVertPlace.Length; i++)
        {
```

					ИГУ 2.090501.03.001 ИЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		

```

        s = UnityEngine.Random.Range(0, _ShortVert.Length);
        Instantiate(_ShortVert[s], _ShortVertPlace[i].transform);
    }
    for (i = 0; i < _LongShopPlace.Length; i++)
    {
        s = UnityEngine.Random.Range(0, _LongShop.Length);
        Instantiate(_LongShop[s], _LongShopPlace[i].transform);
    }
    for (i = 0; i < _ShortShopPlace.Length; i++)
    {
        s = UnityEngine.Random.Range(0, _ShortShop.Length);
        Instantiate(_ShortShop[s], _ShortShopPlace[i].transform);
    }
}

// Update is called once per frame

}

```

					ПГУ 2.090501.03.001 ПЗ	Лист
						101
Изм.	Лист	№ докум.	Подпись	Дата		

Код Bullet.cs

```
using UnityEngine;
```

```
public class BullSet : MonoBehaviour
```

```
{
```

```
    public float live = 2;
```

```
    // Start is called before the first frame update
```

```
    void Awake()
```

```
    {
```

```
        Destroy(gameObject, live);
```

```
    }
```

```
    // Update is called once per frame
```

```
    void OnCollisionEnter(Collision collision)
```

```
    {
```

```
        Destroy(gameObject);
```

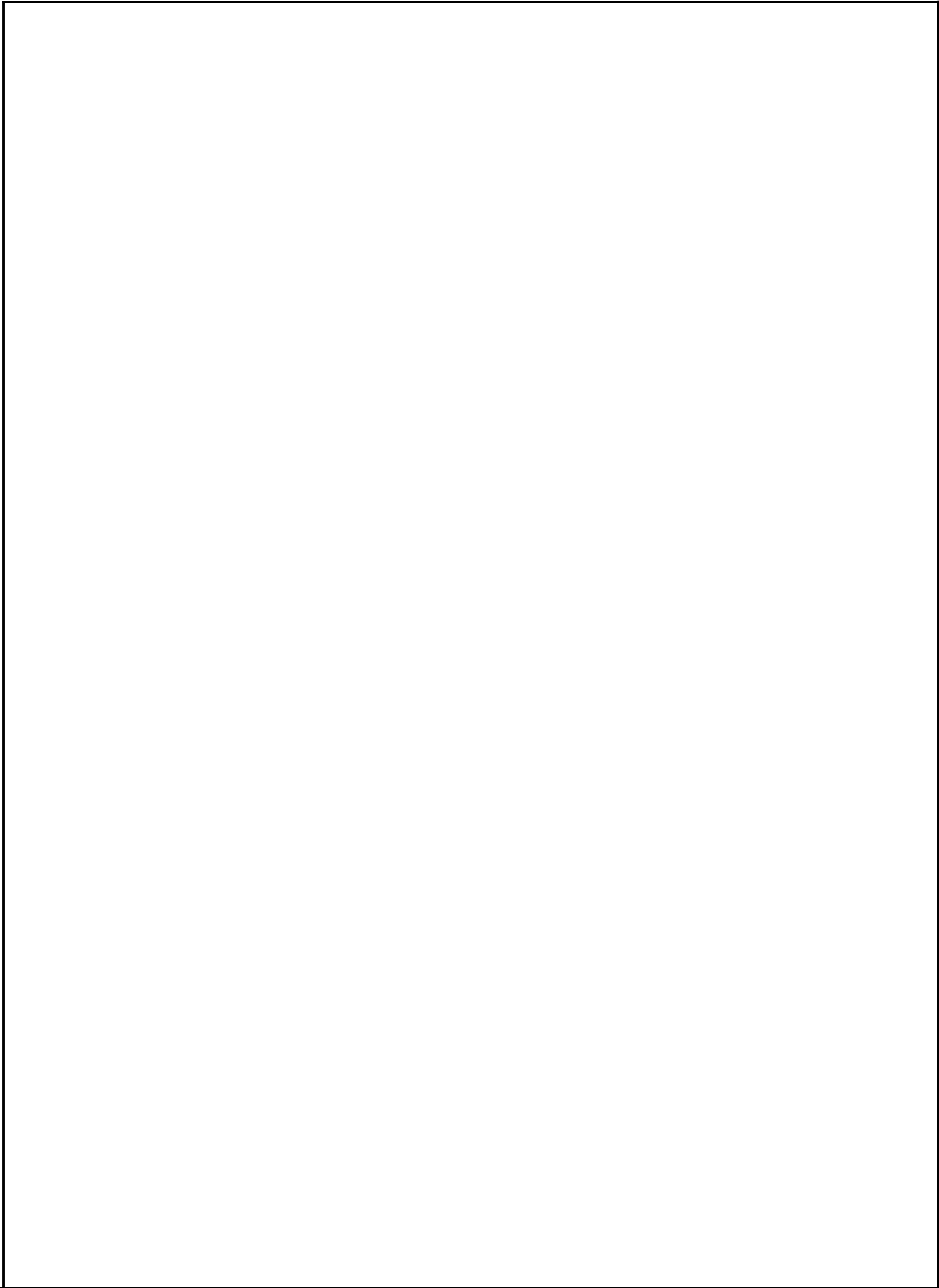
```
    }
```

```
}
```

					ПГУ 2.090501.03.001 ПЗ	Лист
						102
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение Б
(Обязательное)
Мобильное игровое приложение
Схемы программ

					ПГУ 2.090501.03.001 ПЗ	Лист
						103
Изм.	Лист	№ докум.	Подпись	Дата		



					<i>ПГУ 2.090501.03.001 ПЗ</i>	<i>Лист</i>
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		<i>104</i>

Приложение В
(Обязательное)
Мобильное игровое приложение
Презентация

					ПГУ 2.090501.03.001 ПЗ	Лист
						105
Изм.	Лист	№ докум.	Подпись	Дата		



Мобильное игровое приложение

Студент группы 17В02
Тупичак Дмитрий



Рисунок В1 – Первый слайд



ЦЕЛЬ

- создание игрового мобильного приложения для операционной системы Android, привлекательное для пользователя своим дизайном, несложным управлением и дружественным интерфейсом, не имеющее системы автоматического боя и игрового преимущества за деньги. Предполагается оптимизировать приложение для использования на любом мобильном устройстве.

Рисунок В2 – Второй слайд

					ПГУ 2.090501.03.001 ПЗ	Лист
						106
Изм.	Лист	№ докум.	Подпись	Дата		



АКТУАЛЬНОСТЬ

- Чем больше качественных мобильных игровых приложений появится в России, тем конкурентоспособнее она станет на рынке игровой индустрии.

Рисунок В3 – Третий слайд



ЗАДАЧИ

- 1) изучить преимущества операционной системы Android для игровых мобильных приложений;
- 2) рассмотреть вопрос об инструментах создания игровых приложений для операционной системы Android и использовать их для создания приложения;
- 3) рассчитать экономические показатели программного продукта;
- 4) выполнить описание техники безопасности программиста при разработке приложения.

Рисунок В4 – Четвертый слайд

					ПГУ 2.090501.03.001 ПЗ	Лист
						107
Изм.	Лист	№ докум.	Подпись	Дата		



ОБЪЕКТ И ПРЕДМЕТ ИССЛЕДОВАНИЯ

- Объект исследования – игровое приложение для операционной системы Android.
- Предмет исследования: технология разработки игрового приложения для операционной системы Android с помощью Blender и Unity 3D.

Рисунок В5 – Пятый слайд



ПРЕИМУЩЕСТВА ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID

- объединяет информацию из Интернета с данными телефона, контактной информацией и другими данными
- даёт доступ к основным функциям мобильного устройства
- позволяет быстро и легко разрабатывать другие приложения благодаря SDK
- позволяет свободно использовать исходный код для создания собственных систем.

Рисунок В6 – Шестой слайд

					ПГУ 2.090501.03.001 ПЗ	Лист
						108
Изм.	Лист	№ докум.	Подпись	Дата		

ПРОГРАММЫ ДЛЯ РАЗРАБОТКИ ИГРОВЫХ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

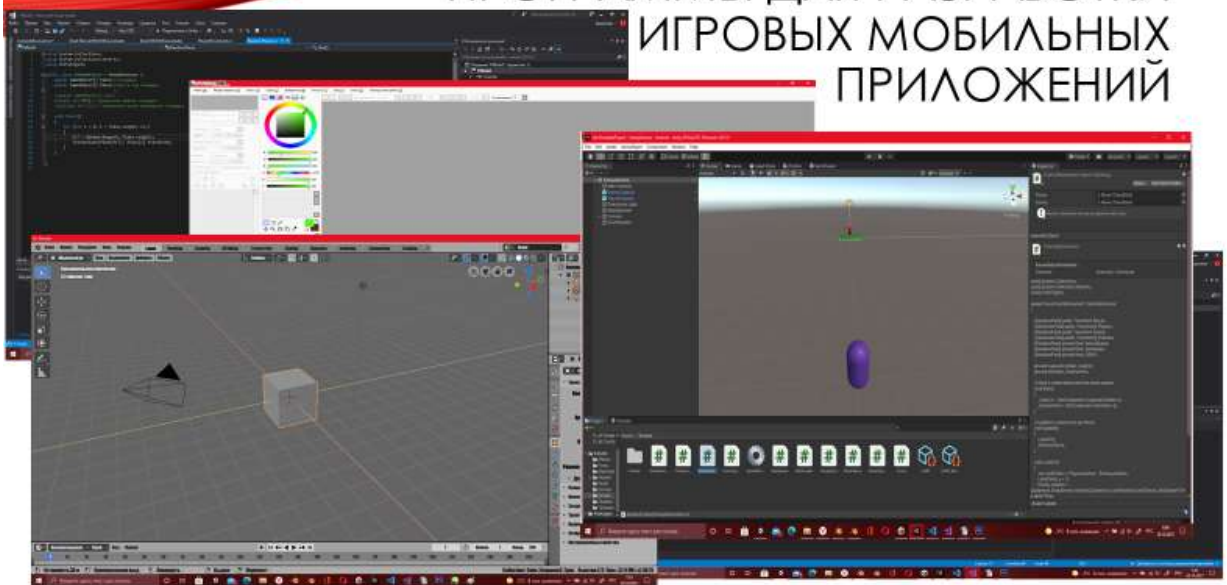
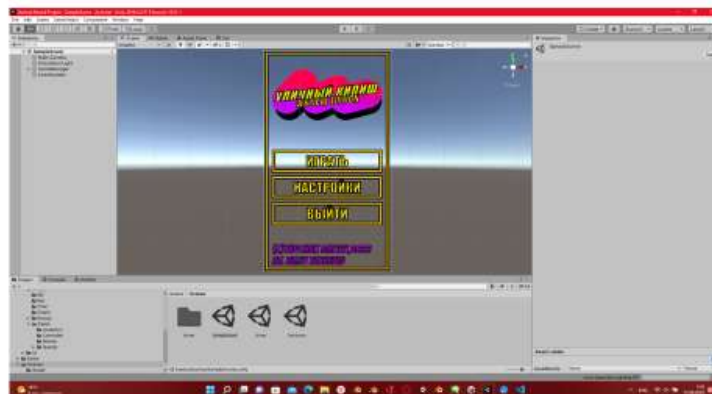


Рисунок В7 – Седьмой слайд

СОЗДАНИЕ СОБСТВЕННОГО МОБИЛЬНОГО ПРИЛОЖЕНИЯ

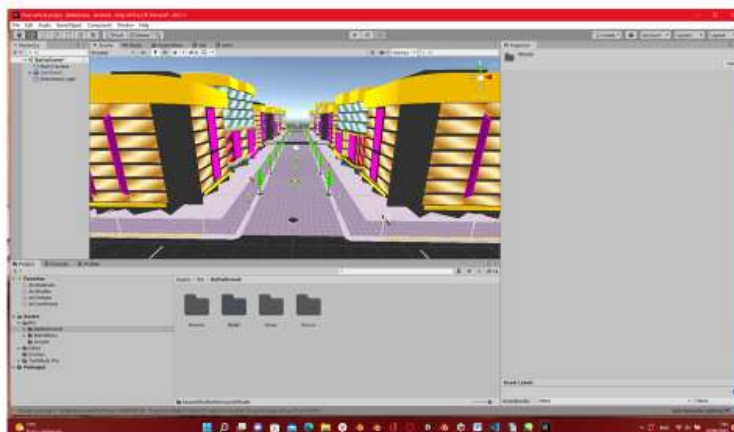


Главное меню

Рисунок В8 – Восьмой слайд

					ПГУ 2.090501.03.001 ПЗ	Лист
						109
Изм.	Лист	№ докум.	Подпись	Дата		

РЕАЛИЗАЦИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ



Проработанная карта

Рисунок В9 – Девятый слайд

СТРУКТУРА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

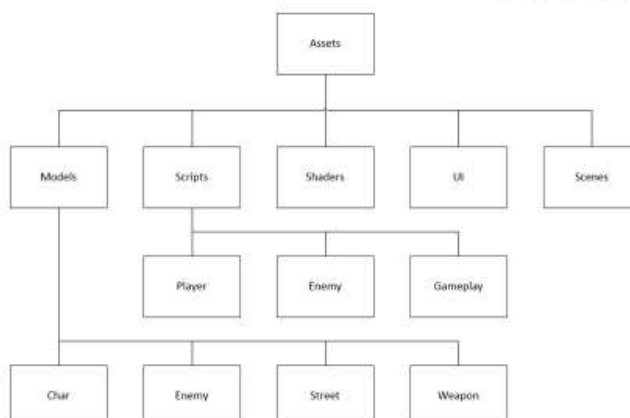


Рисунок В10 – Десятый слайд

					ПГУ 2.090501.03.001 ПЗ	Лист
						110
Изм.	Лист	№ докум.	Подпись	Дата		



СЛОЖНОСТИ И ОГРАНИЧЕНИЯ ПРИ РАЗРАБОТКЕ

- Ограничения на вызовы отрисовок (от 150 до 200)
- Ограничения на количество полигонов модели в 3D – играх (от 300 до 1500(2500))
- Ограничения на размер игры (не более 100Мб)
- Запрет на использование динамического источника света
- Ограничения на операторы условия – команда return, кейсы, бадибэг логика
- Ограничения на количество обрабатывающих изображение процедур (в шейдере)

Рисунок В11 – Одинадцатый слайд



МИНИМАЛЬНЫЕ СИСТЕМНЫЕ ТРЕБОВАНИЯ К УСТРОЙСТВАМ

- Совместимость устройства и приложения
- Объем оперативной памяти (3-4 Гб)
- 4-8 ядер с большой частотой
- Разрешение экрана HD128x720
- Диагональ экрана от 5 дюймов
- Батарея не менее 3000 мАч

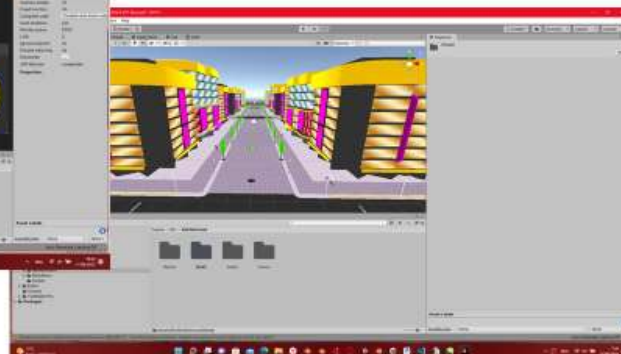
Рисунок В12 – Двенадцатый слайд

					ПГУ 2.090501.03.001 ПЗ	Лист
						111
Изм.	Лист	№ докум.	Подпись	Дата		

ОПТИМИЗАЦИЯ ГРАФИКИ ИГРЫ (ШЕЙДИНГ)



Шейдер, сделанный в ShaderGraph



Результаты его обработки

Рисунок В13 – Тринадцатый слайд

ЗАКЛЮЧЕНИЕ

- 1) были изучены преимущества операционной системы Android для игровых мобильных приложений;
- 2) рассмотрен вопрос об инструментах создания игровых приложений для операционной системы Android и использованы для создания приложения;
- В ходе исследования было создано игровое мобильное приложение.
- Цели достигнуты, задачи выполнены.

Рисунок В14 – Четырнадцатый слайд

					ПГУ 2.090501.03.001 ПЗ	Лист
						112
Изм.	Лист	№ докум.	Подпись	Дата		