МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

РАБОТА ПРОВЕРЕНА

Рецензент
Руководитель отдела ИТ и связи
ООО «Модерн Гласс»

_____ А.Г. Григорьев

“___”_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2020 г.

# Разработка компьютерной игры в жанре «Шутер» на платформе Unity

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2020.308-014 .ВКР

Научный руководитель,
к.ф.-м.н., доцент кафедры СП,
_____ С.А. Иванов

Автор работы,
студент группы КЭ-402
_____А.А. Комар

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
“___”_____ 2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет**
**(национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
09.02.2020

**ЗАДАНИЕ**
**на выполнение выпускной квалификационной работы бакалавра**
студенту группы КЭ-402
Комар Анастасии Алексеевны,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

**1. Тема работы** (утверждена приказом ректора от 24.04.2020 № 627)
Разработка компьютерной игры в жанре «Шутер» на платформе Unity.
**2. Срок сдачи студентом законченной работы:** 05.06.2020.
**3. Исходные данные к работе**
3.1. Robert Nystorm. Game Programming 1-е изд. – Patterns Genver Benning, 2014. – 354 с.
3.2. Торн А. Искусство создания сценариев Unity. – ДМК-Пресс, 2016ю – 360 с
**4. Перечень подлежащих разработке вопросов**
4.1. Провести анализ литературы.
4.2. Спроектировать игровое приложение.
4.3. Выполнить реализацию игрового приложения.
4.4. Провести тестирование игрового приложения.
**5. Дата выдачи задания:** 03.02.2020.

**Научный руководитель,**
к.ф.-м.н., доцент кафедры СП С.А. Иванов

**Задание принял к исполнению** А.А. Комар

**ОГЛАВЛЕНИЕ**

# ВВЕДЕНИЕ

## Актуальность темы

Виртуальная и дополненная реальность являются одной из самых перспективных сфер деятельности в будущем. Так как с помощью данных технологий можно смоделировать любую ситуацию, данную область использует в медицине, в строительстве, в дизайне и даже в кинематографе. Еще одна из наибольших распространенных областей, применяющих VR-индустрия.

Современные технологии дают игроку возможность переключаться между разными видами реальностей. Виртуальная реальность (VR) позволяет погружаться в виртуальный мир с помощью специальных устройств, при этом игрок не может видеть того, что происходит в реальном мире [3]. Благодаря этому создается эффект присутствия в совершенно другом месте. Дополненная реальность (AR) не меняет окружение человека, а лишь привносит в него искусственные элементы, например, информацию о погоде или всплывающее перед глазами уведомления о получении e-mail. Ключевой момент заключается в том, что цифровой контент не прикреплен к пространству. Смешанная реальность (MR) означает, что в реальный мир добавляются виртуальные предметы, которые «прикреплены» к своему месту в пространстве для того, чтобы смотрящий воспринимал их как реальные. Среди устройств смешанной реальности можно отметить Microsoft Hololense и Magic Leap.

Чтобы погрузиться в виртуальную реальность, используют такие оборудования, как VR-очки или VR-шлемы. В настоящее время существует большая линейка шлемов виртуальной реальности, которые делятся на Light Mobile VR, Premium Mobile VR  и компьютерные VR-шлемы [18].

**Обзор инструментов**

Существует множество видеоигр с использованием технологии VR и соответственно множество оборудований, позволяющих игрокам ощущать эффект присутствия в игре. В настоящее время средства виртуальной реальности подразделяют на 3 вида:

1) компьютерная VR;
2) мобильная VR;
3) веб-реализация VR.

Компьютерная VR – технология, полученная трехмерной компьютерной средой, взаимодействующая с человеком через специальные оборудования [14]. Наглядным примером является VR-шлемы HTC Vive и Oculus Rift. Особенностью компьютерной VR в том, что пользователь может не только осматривать окружение виртуальной реальности, но и взаимодействовать и выполнять различные движения. Добиться такого эффекта позволяют специальные контролеры, предусмотренные вместе с VR-очками, которые представлены на рисунке 1.


**Рис. 1.** VR-гарнитура «Oculus Rift»

Рассмотрим мобильную виртуальную реальность. Главное отличие от компьютерной VR заключается в том, что в качестве базового устройства используется смартфон, поэтому этот способ погрузиться в виртуальный мир более доступен. Передвижение игрока также осуществляется с

помощью контроллера, который можно приобрести отдельно от VR-шлема. На рисунке 2 изображен принцип взаимодействия с виртуальной реальностью через смартфон.

**Рис. 2.** Мобильные VR-очки

На рисунке 3 показано как выглядит игра на смартфоне.

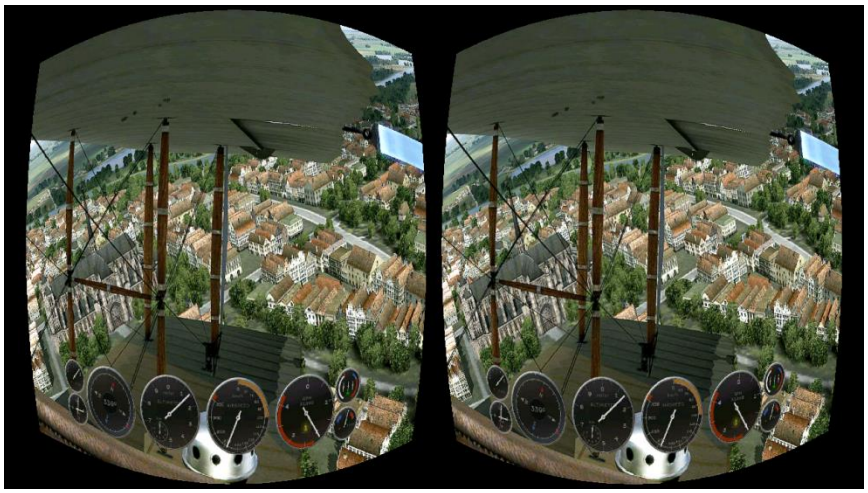

**Рис. 3.** Скриншот игры на Android

Также существует веб-реализация виртуальной реальности, т.е. базовым устройством может быть, как и смартфон, так и компьютер. Самыми популярными веб-приложения для создания VR-сцен являются Vizor и Mozila A-Frame.

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Обзор аналогов

Рассмотрим аналогичные проекты, применяющие технологии виртуальной реальности на компьютерной платформе.

**Arizona Sunshine**

Одной из самых популярных VR-игр в жанре «шутер» является игра «Arizona Sunshine» [4]. Данная игра является  шутером от первого лица, созданный исключительно для виртуальной реальности и погружающий игрока в постапокалиптический мир, захваченный зомби. Скриншот игры представлен на рисунке 4.

**Рис. 4.** Скриншот игры «Arizona Sunshine»

«Arizona Sunshine» – survive-игра. Ее суть заключается в том, что целью игрока является сохранение жизни виртуального персонажа на фоне множества угрожающих ему опасностей. В данном случае главной опасностью игрока оказываются зомби.

Взаимодействие игрока с виртуальным миром осуществляется с помощью контроллеров отслеживающих движения рук. Контроллеры используются для управления виртуальным огнестрельным оружием, а также для взаимодействия с инвентарем игрока и окружающей средой. Передви-

жение по миру на большие расстояния происходит за счет искусственной локомоции, т.е. с помощью кнопок на контроллере.

**Serious Sam VR**

Еще одним примером является игра «Serious Sam VR». На рисунке 5 изображен скриншот игры.

**Рис. 5.** Скриншот игры «Serious Sam 3 VR»

Данная игра создана в жанре трехмерного шутера от первого лица из серии игр «Serious Sam», разрабатывающаяся эксклюзивно для очков виртуальной реальности [10]. Игра представляется игроку в качестве симулятора для новобранцев Армии Обороны Земли, которые сражаются с инопланетными расами. Все взаимодействие с объектами и управление оружием осуществляется с помощью контроллеров, а передвижение в игре частично статично, т.е. персонаж не может передвигаться по уровням на далекое расстояние. Игрок может осуществлять только небольшие движения в рамках области, которая отслеживается датчиками VR-шлема. Данный принцип не совсем оптимален, так как у игрока нет возможности перемещаться по карте, что ограничивает его взаимодействие с виртуальным миром.

**Superhot VR**

Superhot – игра в жанре «шутер» от первого лица. Хоть она и основана на традиционной игровой механике шутеров, когда игрок пытается уничтожить вражеские цели с помощью оружия, данная игра не является типичной для игр такого жанра. На рисунке 6 показан скриншот «Superhot».

**Рис. 6.** Скриншот игры «Superhot VR»

Динамика в игре спокойна, что не соответствует традиционным шутерам. Особенность этой игры заключается в том, что время начинает двигаться только тогда, когда начинает двигаться сам игрок. Оставаясь без движения, время в игре останавливается. Из-за данной специфики передвижение на дальние расстояния бессмысленны, поэтому данное действие здесь отсутствует.

Основываясь на результатах обзора аналогов можно сделать вывод, что игры данного жанра очень популярны и предоставляют широкий спектр реализации различных идей. Также существует несколько вариантов передвижения по виртуальному миру, дающие возможность определится с наиболее удобным вариантом перемещения.

## 1.2. Обзор решений для реализации проекта

На сегодняшний день индустрия видеоигр успела набрать огромную популярность и развить современные технологии для получения качественного взаимодействия с игровым миром. Соответственно существует множество сред разработки для создания видеоигр, созданных с целью наиболее простого и удобного способа реализации. Рассмотрим несколько примеров программного обеспечения для создания видеоигр.

1) Unity – межплатформенная среда разработки компьютерных игр [11]. Особенностью этого игрового движка является то, что с помощью данной среды можно создать приложения и игры, работающие на более чем 25 различных платформ, включающих персональные компьютеры, игровые консоли, мобильные устройства и другие. На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и независимыми студиями.

2) Unreal Engine – игровой движок, который первоначально разрабатывался для создания шутеров от первого лица. Однако его последующие версии успешно применялись в играх самых различных жанров, в том числе игры в жанре «стелс», в жанре «файтинг» и массовых многопользовательских ролевых онлайн-играх. Достоинством данной среды является улучшенная графика и простота разработки.

3) Godot Engine – открытый кроссплатформенный 2D и 3D игровой движок. Среда позволяет разработчикам создавать игры с нуля, не пользуясь никакими инструментами, за исключением тех, которые необходимы для создания игрового контента, т.е. элементов графики, музыкальных треков и т.д. Процесс программирования также не требует внешних инструментов, но при необходимости использовать внешний редактор это можно легко сделать.

На таблице 1 приведены данные о вышеописанных игровых движках.

**Табл. 1.** Характеристика сред разработки игр

| № | Характеристики | Среды разработки игр | | |
|---|---|---|---|---|
| | | **Unity** | **Unreal Engine** | **Godot** |
| 1 | Мультиплатформенность | Да | Да | Да |
| 2 | Бесплатный доступ | Да | Да | Да |
| 3 | Наличие обучающих материалов | Да | Да | Нет |
| 4 | Большое количество игровых ресурсов | Да | Да | Нет |
| 5 | Удобный визуальный скриптинг | Нет | Да | Да |
| 6 | Проблемы с производительностью | Да | Да | Да |
| 7 | Удобство пользовательского интерфейса | Да | Нет | Нет |
| 8 | Поддержка VR | Да | Да | Нет |

Проанализировав данные, приведенные в таблице, средой разработки был выбран игровой движок Unity [12]. Недостаток данного программного обеспечения незначителен, так как визуальное программирование в данной работе не предусмотрено.

**Вывод**

В ходе работы был произведен обзор существующих аналогов, в результате которого были выявлены их недостатки и достоинства. Также на основе полученных данных были выбраны инструменты, которые будут использоваться для реализации поставленных задач.

# 2. ПРОЕКТИРОВАНИЕ

## 2.1. Техническое предложение

Данное игровое приложение должно быть выполнено в жанре «Шутер» от первого лица с использованием технологий виртуальной реальности. Шутер – жанр компьютерных игр, действия которые основываются на сражениях с противниками с помощью огнестрельного или другого типа оружия.

**Концепция игры**

Действия игры разворачиваются на острове, где орудуют пираты, готовые атаковать любого, ради наживы. С ними и происходит главное противостояние игрока. По пути игроку попадаются различные виды оружий: пистолеты, холодное оружие, гранаты. Чем дальше игрок проходит в джунгли, тем больше врагов ему попадается на пути и соответственно больше оружия. Все взаимодействие с оружием и предметами происходит с помощью контроллеров, а повороты камеры выполняются за счет поворота головы. Передвигаться также можно либо в пределах ограниченной области, либо с помощью котроллеров, выполняя телепортацию на дальние расстояния. Цель игры пройти всех врагов и найти способ выбраться с острова.

**Игровые объекты**

В игре должны быть реализованы следующие объекты взаимодействия с игроком.

1) Оружие, которое можно собрать после победы над врагом. Также различное оружие может находиться в произвольных местах на пути к цели.

2) Боеприпасы и медикаменты для восстановления арсенала и здоровья.

3) Помимо объектов взаимодействия, в игре имеются объекты окружения, как палатки, ящики, деревья, камни и т.д., а также сами противники.

**Интерфейс**

В игре имеется главное меню, окно, всплывающее после поражения и интерфейс игрового поля.

В главном меню должны быть реализованы кнопки:

1) играть;
2) опции;
3) выход.

Кнопка «Играть» позволяет пользователю перейти в игровое поле и начать играть. С помощью кнопки «Опции» игрок переходит в новое окно, в котором появляется возможность отключить звук и настроить время суток: темное или светлое. Кнопка «Выход» позволяет выйти игроку из игры. Также эта кнопка расположена в сплывающем окне после поражения, давая возможность не заходить в главное меню и быстро воспроизвести выход.

В игровом поле должны отображаться запасы боеприпасов, панель, отображающая здоровье персонажа, а также карта, чтобы игрок мог ориентироваться и видеть возможные пути передвижения.

После смерти персонажа, перед игроком всплывает окно, в котором расположено три кнопки: «Начать заново», «Главное меню» и «Выход из игры». При нажатии кнопки «Начать заново», игрок переносится в место, откуда началась игра, при этом запас боеприпасов и здоровье полностью восстановлены.

**Игровой процесс**

Игра является однопользовательской и предусматривает вид от первого лица. Игрок, находясь на острове, исследует окрестности в поисках каких-либо предметов. На пути игроку могут попасться либо оружие, такое как пистолет, автомат, лук гранаты или ножи, а также боеприпасы и медикаменты, для пополнения запасов. После того, как игрок найдет первое огнестрельное оружие, на его пути встретятся разбойники, которым он должен оказать противостояние.

Игроку дается возможность убить противников тихо и остаться незамеченным, либо с помощью огнестрельного оружия или разрывных снарядов, и привлечь внимание других врагов. Для того чтобы незаметно убить врагов, игрок может использовать нож, топор или лук со стрелами. Если он будет использовать огнестрельное оружие или разрывные снаряды, к которым относятся гранаты и болоны с бензином, расположенные в различных местах баз бандитов, то игрок не сможет остаться незамеченным и к нему сбегутся все враги, находящиеся поблизости.

## 2.2. Определение требований

Проектируя игровое приложение, были сформулированы функциональные и нефункциональные требования.

**Функциональные требования**

Функциональные требованию описывают задачи, которые должна выполнить разрабатываемая система, демонстрируя ее поведение.

Разрабатываемое игровое приложение должно удовлетворять следующим требованиям.

1) Игровое приложение должно представить пользователю главное меню, с помощью которого игрок быстро сможет освоиться с пользовательским интерфейсом.

2) Игровое приложение должно предоставить пользователю возможность регулировать настройки звуки.

3) Игровое приложение должно предоставить пользователю возможность перезапустить игру в любой момент времени.

4) Игровое приложение должно предоставить пользователю все игровые способности.

**Нефункциональные требования**

Нефункциональные требования описываю характеристики системы, а также ограничения, накладываемые на нее. В ходе работы были определены следующие нефункциональные требования.

1) Игровое приложение должно быть реализовано на языке программирования C# с помощью игровой платформа Unity3D.

2) Игровое приложение должно быть реализовано для устройств вывода и ввода, использующих технологии виртуальной реальности.

3) Игровое приложение должно функционировать на операционной системе Windows с версией не ниже 7.

4) Игровое приложение должно взаимодействовать с контроллерами и шлемом «HTC Vive».

## 2.3. Диаграмма вариантов использования

В ходе анализа требований, была построена диаграмма, с помощью языка графического описания для объектного моделирования UML [9], которая представлена на рисунке 7.

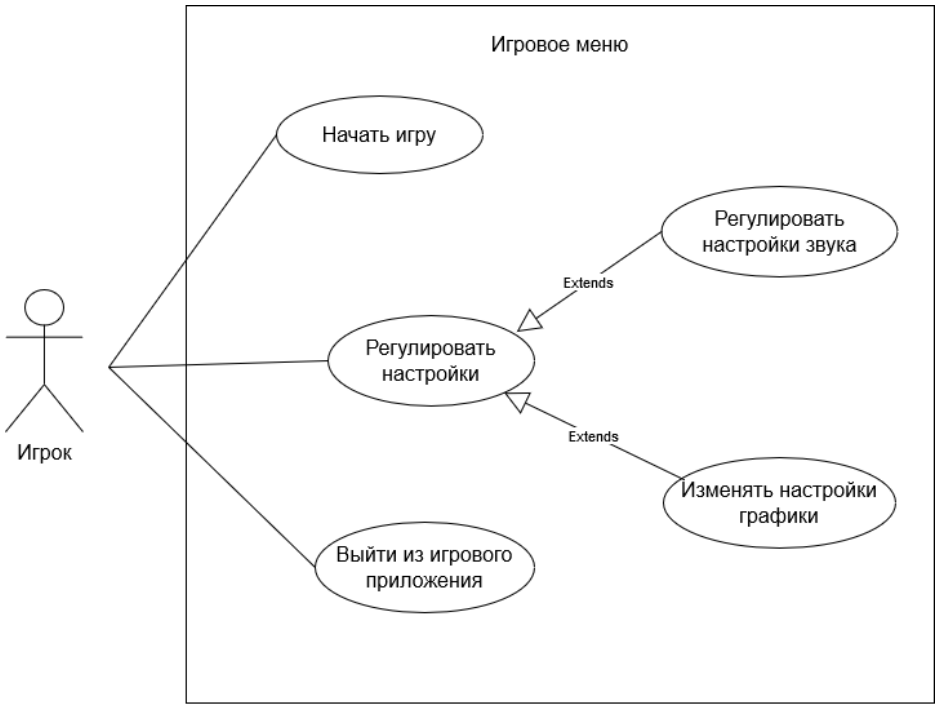


**Рис. 7.** Диаграмма вариантов использования в игровом меню

Для игрока определены следующие варианты использования.

1) Начать игру – игрок появляется на игровом поле и начинает игровой процесс.

2) Регулировать настройки – игрок изменяет настройки игрового приложения.

3) Выйти из игрового приложения – игрок заканчивает игровой процесс и выходит из игры.

На рисунке 8 представлена диаграмма вариантов использования в игровой сцене.

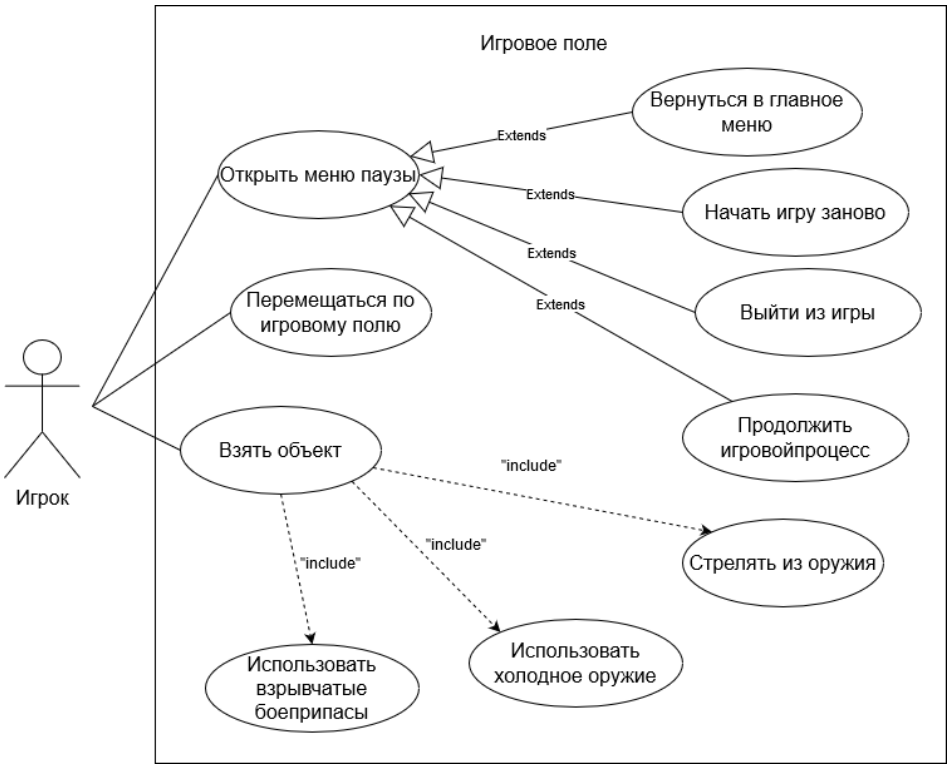


**Рис. 8.** Диаграмма вариантов использования в игровом поле

Рассмотрим варианты использования в игровом поле.

1) Открыть меню паузы – игрок приостанавливает игровой процесс и получает возможность вернуться в главное меню, начать игровой процесс заново, закончить игру или продолжить игровой процесс.

2) Перемещаться по игровому полю – игрок перемещается по полю либо с помощью контроллеров, используя указатель, производящий телепорт в заданную точку, либо с помощью физического передвижения в ограниченной области видимости базовых станций.

3) Взять объект – игрок захватывает объекты с помощью контроллера и держит в виртуальной руке. Исходя из этого прецедента, пользователь сможет стрелять из оружия, использовать холодное оружие или взрывчатые боеприпасы.

**Вывод**

В ходе проектирования игрового приложения были сформирована основная концепция игры, которая была представлена в техническом предложении. Также был проведен анализ требований, в котором были приведены функциональные и нефункциональные требования. На основе функциональных требований были построены диаграммы вариантов использования в игровом меню и поле.

# 3. РЕАЛИЗАЦИЯ

Для создания игрового приложения была выбрана межплатформенная среда разработки Unity. Преимущество данного редактора заключается в том, что он имеет удобный интерфейс, бесплатный доступ и наличие большого количества контента.

Для графической составляющей игры были использованы компоненты, взятые из информационного ресурса Assets Store [16].

Также в качестве взаимодействия с виртуальной реальностью использовался набор инструментов SteamVR.

## 3.1. Файловая структура приложения

На рисунке 9 представлена файловая структура игрового приложения. Проект состоит из набора каталогов, которые содержат анимации, текстуры, модели, скрипты, аудио и т.д.

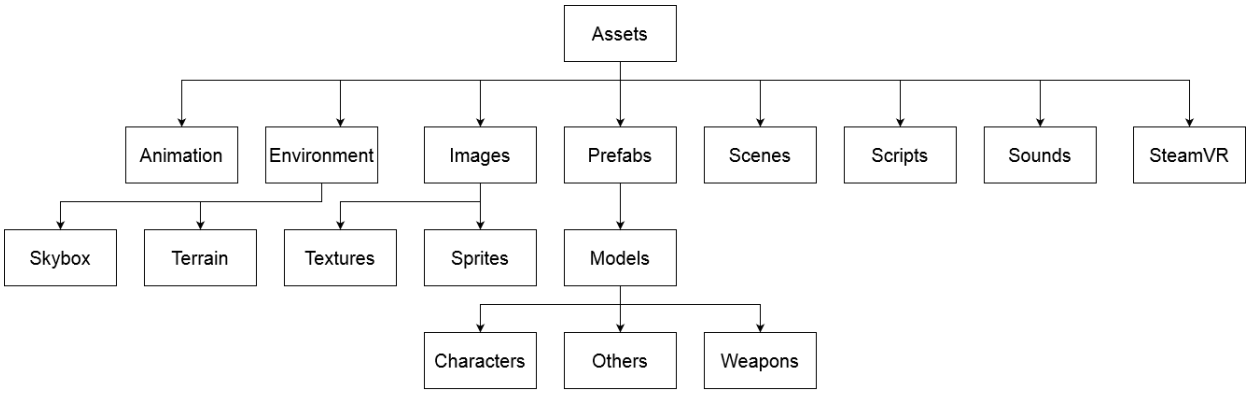


**Рис. 9.** Файловая структура приложения

В директории Animation содержатся анимации персонажей.

В директории Environment находятся панорамы неба, а также окружение игры.

В директории Image содержатся текстуры, спрайты и элементы интерфейса.

В директории Prefabs наборы готовых объектов и модели. К ним относятся модели персонажей, оружие и другие объекты.

В директории Scenes находятся сцены с игровым полем и меню.

В директории Scripts содержатся скрипты, используемые в игровом приложении.

В директории Sounds содержатся музыкальные файлы.

В директории SteamVR находятся скрипты и объекты, позволяющие отслеживать движения игрока.

## 3.2. Диаграмма компонентов

Диаграмма компонентов отображена на рисунке 10.

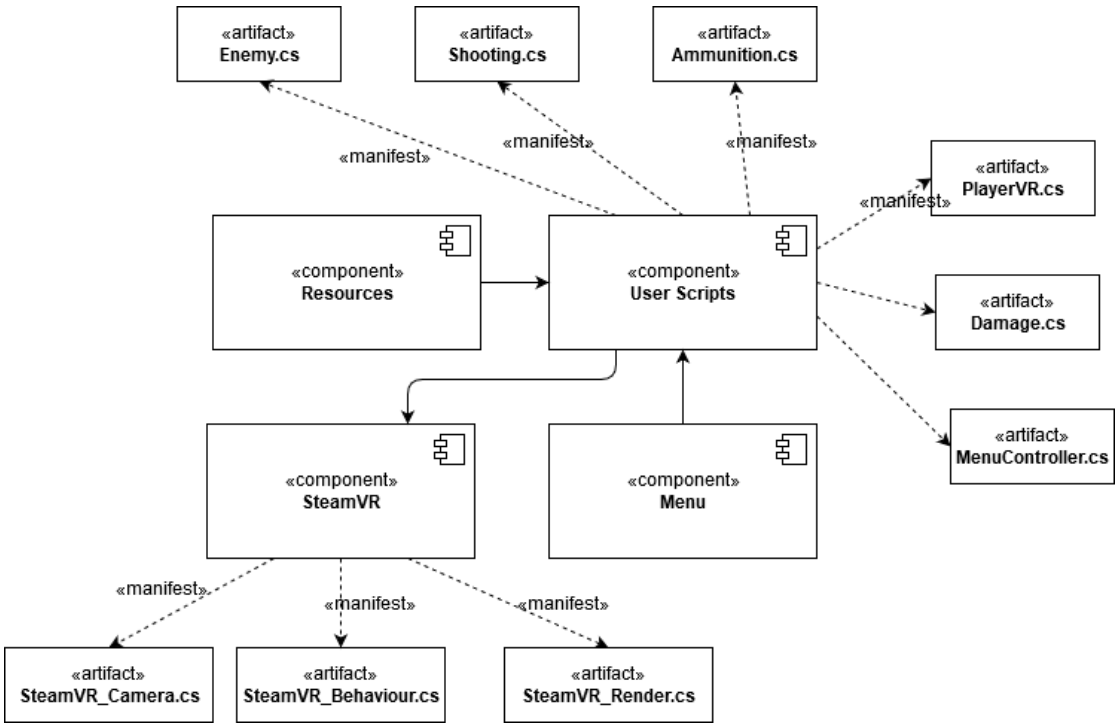


**Рис. 10.** Диаграмма компонентов

Приложение состоит из четырех основных логических блоков.

1) User Scripts.
2) Menu.
3) Resources.
4) SteamVR.

Компонент User Scripts представлен в виде ряда артефактов – файлов с расширением *.cs, основными из которых являются: «Enemy.cs», «Shooting.cs», «Ammunition.cs», «PlayerVR.cs», «Damage.cs», и «MenuController.cs».

Компонент Menu содержит элементы главного меню игры и игровое меню.

Компонент Resources содержит файлы с графическим оформлением сцен и готовых моделей.

Компонент SteamVR содержит ряд артефактов – систем объектов, каждая из которых отвечает за взаимодействие со шлемом и контроллерами виртуальной реальности.

### 3.3. Диаграмма классов

На рисунке 11 представлена диаграмма классов игрового приложения.

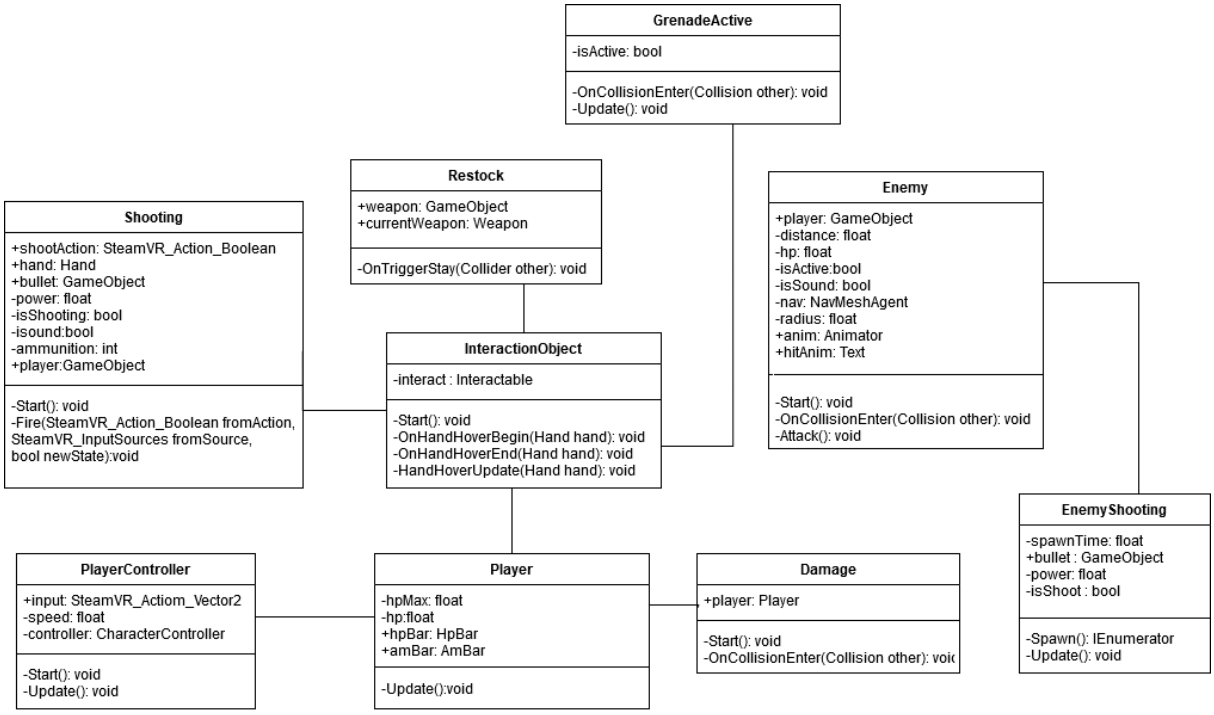


**Рис. 11.** Диаграмма классов

Класс Player отвечает за поведение игрока.

Класс PlayerController производит управление персонажем и обработку нажатий кнопок на контроллере.

Класс InteractionObject отвечает за взаимодействия с игровыми объектами.

Класс Damage - класс, отвечающий за нанесение урона игроку.

Класс Shooting производит стрельбу из оружия игроком.

Класс Restock пополняет боеприпасы и позволяет восстанавливать здоровье игрока.

Класс GrenadeAction –класс, который активирует гранату.

Класс Enemy позволяет передвигаться врагам и следовать за игроком.

Класс EnemyShooting производит стрельбу врагов.

### 3.4. Реализация поведения игрока

Для передвижения игрока с помощью котроллеров был создан скрипт PlayerController, который представлен на рисунке 12.

```
void Update(){

     if(input.axis.magnitude > 0.1f) {

          Vector3 direction = Play-
          er.instance.hmdTransform.TransformDirection(new Vec-
          tor3(input.axis.x, 0, input.axis.y));

      controller.Move(speed * Time.deltaTime * Vec-
     tor3.ProjectOnPlane(direction, Vector3.up) - new Vec-
     tor3(0, 9.81f, 0) * Time.deltaTime);

     }

}
```

**Рис. 12.** Передвижение игрока с помощью контроллеров

Для начала необходимо поставить условие, чтобы величина вектора оставалась положительной, иначе игрок будет падать бесконечно. После этого реализуем движение игрока, в котором, помимо движения по оси X и

Z, игрок может перемещаться вверх, например по лестнице. Также, необходимо учесть гравитацию и вычесть вектор, который является его направлением.

Скрипт InteractionObject дает возможность захватывать объекты и держать их в руке. На рисунке 13 представлен листинг этого класса.

```
 void OnHandHoverBegin(Hand hand){
        hand.ShowGrabHint();
    }
    void OnHandHoverEnd(Hand hand){
        hand.HideGrabHint();
    }
    void OnHandHoverUpdate(Hand hand){
     GrabTypes grabType = hand.GetGrabStarting();
     bool isGrabEnding = hand.IsGrabEnding(gameObject);
     if (interact.attachedToHand == null && grabType !=  GrabTypes.None){
                hand.AttachObject(gameObject, grabType);
                hand.HoverLock(interact);
                hand.HideGrabHint();
            }
          else if (isGrabEnding)
          {
                 hand.DetachObject(gameObject);
                 hand.HoverUnlock(interact);
        }
    }
```

**Рис. 13.** Захват объектов с помощью контроллеров

Функции OnHandHoverBegin и OnHandHoverEnd прячут и показывают контроллеры, чтобы не получилось ситуации, когда при захвате объ-

ектов, в руке оказалось два объекта. Функция OnHandHoverUpdate позволяет держать объект в виртуальной руке и отпускать его, когда, кнопка на контроллере отжата.

Также одной из самых необходимых функций в игре является возможность стрелять из оружия. Пример функций, отвечающий за стрельбу, приведен на рисунке 14.

```
private void Fire(){

     GameObject b = Instantiate(bullet,  GameOb-
ject.Find("SpawnPoint").transform.position, Quaterni-
on.identity);

     Rigidbody rb = b.GetComponent<Rigidbody>();

     ammunition = ammunition - 1;

     rb.AddForce(transform.forward * power);

}

void Update(){

     if(interact.attachedToHand != null)

     {

           SteamVR_Input_Sources source = inter-
act.attachedToHand.handType;

           if (shootAction[source].stateDown)

           {

               Fire();

           }

     }

}
```

**Рис. 14.** Стрельба из оружия

В функции Update происходит обработка кнопок контроллера в случае, когда нажата кнопка «Trigger». Когда виртуальная рука прикасается к объекту и нажимается данная кнопка, происходит вызов функции Fire. Функция Fire предназначена для создания пули, наделяет ее физическими

свойствами и предает ей движение, направленное в ту же сторону, куда нацелено оружие.

### 3.5. Реализация поведения врагов

В игровом приложении существует два вида врагов: с холодным и огнестрельным оружием. На рисунке 15 представлено основное поведение врагов.

```
void Update(){
        distance = Vector3.Distance(player.transform.position,
transform.position);


        if ((distance <= radius) && (hp > 0) ){

            nav.enabled = true;

            nav.SetDestination(player.transform.position);

            gameOb-
ject.GetComponent<Animator>().SetTrigger(anim);

            isActive = true;

        }
        if (distance > radius){
            nav.enabled = false;
            gameOb-
ject.GetComponent<Animator>().SetTrigger("Idle");
            isActive = false;
        }
        if (hp <= 0){
            nav.enabled = false;
            gameOb-
ject.GetComponent<Animator>().SetTrigger("Die");
            isActive = false;
        }

        if (isSound == true){
            nav.enabled = true;
            nav.SetDestination(player.transform.position);
            gameOb-
ject.GetComponent<Animator>().SetTrigger(anim);
            isActive = true;
      }
}
```

**Рис. 15.** Поведение врагов в зависимости от положения игрока

В данном фрагменте кода измеряется расстояние врагов от игрока. Если игрок подошел слишком близко к нему и попал в его поле зрения, то враг начинает атаку. Также, если игрок далеко отбежит от врага, то он перестает за ним следовать. Помимо этого, на разные ситуации, у врагов проигрывается своя анимация.

Один из основных видов врагов – стреляющий противник. На рисунке 16 представлен код, производящий появление пули и стрельбу врага.

```
IEnumerator Spawn(){


        while (hp != 0){
Debug.Log("SHOOT");

            GameObject b = Instantiate(bullet, GameOb-
ject.Find("BulletSpawn").transform.position, Quaterni-
on.identity);
            Rigidbody rb = b.GetComponent<Rigidbody>();
            rb.AddForce(transform.forward * power);
            yield return new WaitForSeconds(spawnTime);
}

    }

void Update()
    {
        Enemy en = enemy.GetComponent<Enemy>();
        active = en.isActive;
        distance = en.distance;
        hp = en.hp;


        if
(anim.GetCurrentAnimatorStateInfo(0).IsName("Shoot"))
        {

            while (temp >= 1)
            {
                temp = 0;
                StartCoroutine(Spawn());
            }
        }
    }
```

**Рис. 16.** Стрельба врагов

Чтобы реализовать появление и поведение пули, необходимо использовать коротуны. Коротуны используются, чтобы запускать функции, которые должны работать параллельно в течение определенного времени [1]. Функция Update связывается с основным кодом поведения противников, а также вызывает функцию Spawn, чтобы произвести стрельбу только тогда, когда это необходимо.

### 3.6. Создание интерфейса

Интерфейс в игре также реализован в Unity с помощью системы User Interface [6], позволяя, таким образом, пользователю создавать интерфейс в редакторе.

Основные виды объектов, задействованные в создании пользовательского интерфейса.

1) UI.Button – выполняет функции кнопок, которые используются для взаимодействия игрока с меню.

2) UI.Image – используется для отображения жизни игрока, которое меняется в зависимости от его количества.

3) UI.Text – отображает различную информацию в виде текста.

Объекты пользовательского интерфейса, используемые в игровом приложении:

1) главное меню;

2) меню паузы;

3) шкала здоровья;

4) панель количества патрон;

При запуске игрового приложения запускается игровое меню, с помощью которого, можно запустить игру, регулировать настройки и выполнить выход. Меню состоит из блоков, выполняющих роль кнопок, которые сделаны из примитивов редактора. Кнопки реализованы посредствам кубов и UI.Button. На рисунке 17 показано как выглядит меню в игровом приложении.

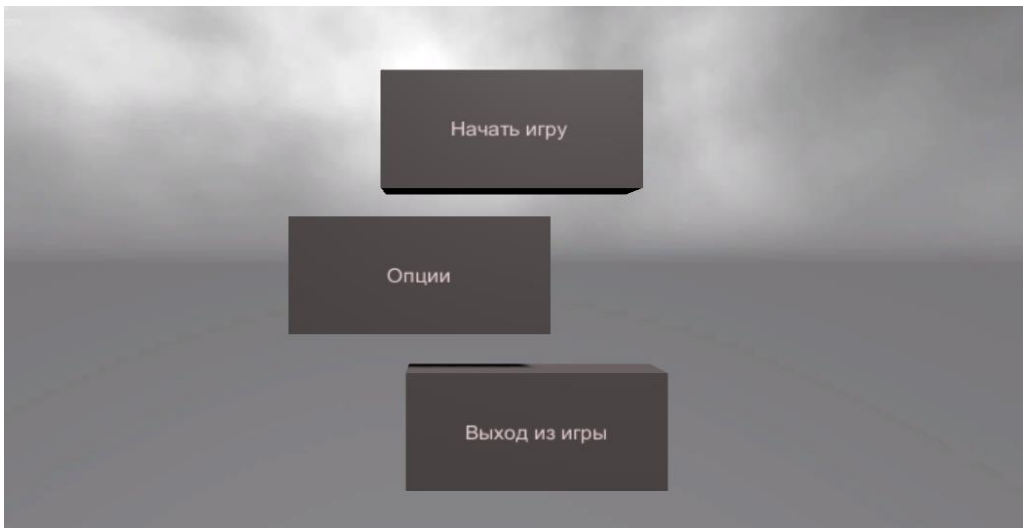


**Рис. 17.** Окно главного меню в игре

Чтобы нажать на кнопку, игрок должен подойти к блокам, поднести виртуальную руку к нужной кнопке и зажать курок на контроллере, тогда кнопка произведет соответствующее действие.

В меню имеются вложенные кнопки. При нажатии на кнопку «Опции», слева появятся еще два блока: настройка звука и графики. Пример вложенных кнопок опции представлен на рисунке 18.

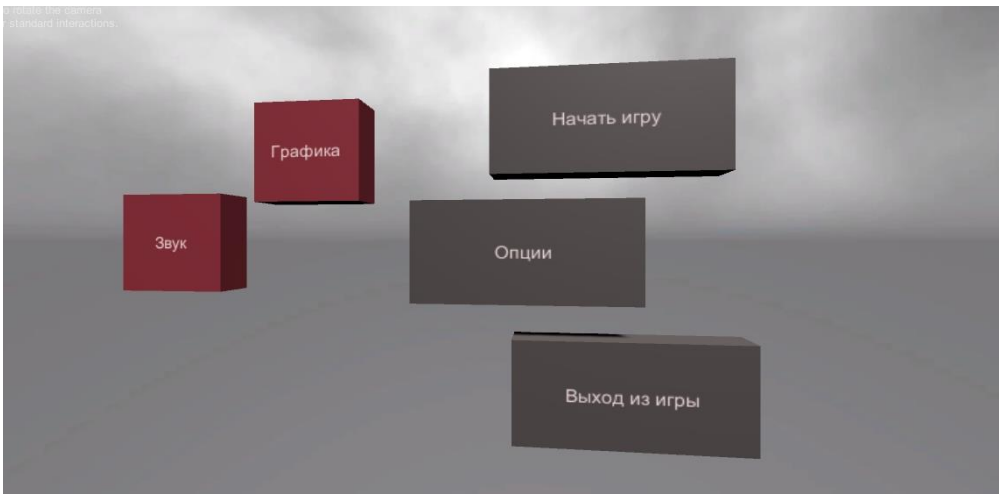


**Рис. 18.** Игровое меню при нажатии на кнопку «Опции»

Блоки «Звук» и «Графика» так же имеют вложенные кнопки, с помощью который можно менять их настройки. Таким образом, меню со всеми вложенными кнопками показано на рисунке 19.


**Рис. 19.** Меню со всеми возможными вариантами выбора

Также после поражения персонажа появляются панели, в которых есть возможность начать игру заново, вернуться в главное меню или выйти из игрового приложения (рисунок 20). Кроме того, данные панели можно вызвать при нажатии кнопки «Система» на контроллере, чтобы остановить игру или выбрать одно из действий.


**Рис. 20.** Всплывающие панели паузы

На рисунке 21 представлена, как выглядит шкала жизни в игровом приложении.

**Рис. 21.** Отображение значения здоровья

Также в игровом приложении отображается количество боеприпасов. Количество патрон расположено около оружия и следует за ним. Текст появляется только тогда, когда игрок берет оружие в руки (рисунок 22).


**Рис. 22.** Отображение количества патронов

**Вывод**

В ходе выполнения реализации игрового приложения была описана файловая структура и диаграмма классов, представляющая основные классы приложения. Также была описана реализация игрового процесса и интерфейс игры.

# 4. ТЕСТИРОВАНИЕ

Для тестирования игрового приложения использовалось функциональное тестирование и проверка эргономичности.

## 4.1. Функциональное тестирование

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения решать нужные задачи для пользователей в определенных условиях [7]. Результаты тестирования представлены в таблице 2.

**Табл. 2.** Функциональное тестирование

| № | Название теста | Шаги | Ожидаемый результат | Тест пройден? |
|---|---|---|---|---|
| 1 | Работа кнопки «Играть» | 1. Запустить игру.<br>2. Нажать на кнопку «Играть» в главном меню. | После нажатия на кнопку открывается сцена с игровым полем | Да |
| 2 | Работа кнопки «Опции» | 1. Запустить игру.<br>2. Нажать на кнопку «Опции». | После нажатия на кнопку откроется меню, в котором можно регулировать звук и время суток. | Да |
| 3 | Работа кнопки «Выход» | 1. Запустить игру.<br>2. Нажать кнопку «Выход». | После нажатия на кнопку, игровое приложение заканчивает свою работу. | Да |
| 4 | Регулирование звука | 1. Нажать на кнопку «Опции».<br>2. Нажать на кнопку «Звук». | Появляется ползунок, который при передвижении регулирует звук в игре. | Да |
| 5 | Изменение времени суток | 1. Нажать на кнопку «Опции».<br>2. Нажать на кнопку «Свет». | Появляется две кнопки, которые отвечают за день и ночь. После этого меняется панорама неба. | Да |

Продолжение табл. 2.

| № | Название теста | Шаги | Ожидаемый результат | Тест пройден? |
|---|---|---|---|---|
| 6 | Передвижение игрока | 1. Нажать на кнопку «Играть».<br>2. Зажать кнопку «Захват» на контроллере.<br>3. Направить указатель на место, куда должен переместиться игрок | Направив в указанное место и отжав кнопку «Захват», игрок должен переместиться в указанное место. | Да |
| 7 | Захват предметов | 1. Подойти к объекту, который можно взять.<br>2. Протянуть виртуальную руку к объекту.<br>3. Зажать курок на контроллере. | После зажатия в виртуальной руке должен оказаться объект. | Да |
| 8 | Стрельба из оружия | 1. Захватить оружие в руке с помощью курка.<br>2. Нажать на сенсорную панель на контроллере. | После нажатия на сенсорную панель оружие должно выстрелить из виртуальной руки. | Да |
| 9 | Восстановление запасов и здоровья | 1. Захватить объект «Патроны» с помощью курка на контроллере.<br>2. Захватить объект «Медикаменты» с помощью курка на контроллере. | После захвата объекты должны исчезнуть, а здоровье или запас припасов должны восстановиться. | Да |
| 10 | Смерть персонажа | 1. Понизить значение здоровья до нуля. | Когда значение здоровье равняется нулю, на игровом поле появляется меню с возможность начать заново, выйти в главное меню или выйти из игры. | Да |

Окончание табл. 2.

| № | Название теста | Шаги | Ожидаемый результат | Тест пройден? |
|---|---|---|---|---|
| 11 | Поражение врагов | 1. Убить противника с помощью огнестрельного или холодного оружия. | После нанесенного урона, полученного тремя выстрелами или ножевым ранением, противник должен умереть. | Да |
| 12 | Стрельба противника | 1. Подойти к противнику. | После приближения к противнику, враг должен следовать за игроком и воспроизводить стрельбу. | Да |
| 13 | Вызов окна паузы | 1. Нажать кнопку «Меню» на контроллере во время игрового процесса. | Процесс игры должен остановиться. После этого возникает окно, в котором появляется возможность начать игру заново, выйти в главное меню или выйти из игры | Да |
| 14 | Взрыв разрывного снаряда | 1. Взять гранату.<br>2. Бросить объект. | После соприкосновения с землей должно пройти пару секунд, чтобы граната взорвалась, и противники поблизости погибли. | Да |

### 4.2. Юзабилити-тестирование

Юзабилити-тестирование – это метод тестирования, проверяющий удобство и понятность в использовании продукта для пользователя.

Чтобы проверить игровое приложение на удобство, участникам тестирования необходимо выполнить следующие задачи.

1) Отключить звук в игре.
2) Начать игру.
3) Переместиться в любую доступную точку.
4) Схватить огнестрельное оружие.
5) Выстрелить из оружия.

6) Выстрелить в противника.
7) Бросить объект и схватить новый.
8) Приостановить игру.
9) Выйти из игры.

Все участники тестирования выполнили задания без особых затруднений. Было отмечено простое и интуитивное взаимодействие с игровой средой и высокое погружение в виртуальную реальность. Проверка эргономичности прошла успешно.

**Вывод**

Основываясь на функциональных требованиях, было проведено функциональное тестирование и проверка эргономичности. В набор тестов входила проверка взаимодействия пользователя с игровой средой, а также взаимодействие с пользовательским интерфейсом. Все тесты были выполнены успешно.

# ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы бакалавра была реализована игра в жанре «Шутер» на платформе Unity с применением виртуальной реальности.

Также были выполнены поставленные задачи:

1) выполнен анализ актуальности темы;
2) произведен анализ существующих аналогов и обзор средств реализации игрового приложения;
3) описана концепция игры;
4) спроектировано игровое приложение;
5) выполнена реализация игрового приложения;
6) проведено тестирование игры.

**ЛИТЕРАТУРА**

1. Торн А. Искусство создания сценариев в Unity. – М.: ДМК-Пресс, 2016. – 360 с.

2. Josuttis N. M. The C++ standard library: a tutorial and reference. – Ad-dison-Wesley, 2012. – 592 p.

3. Линовес Д. Виртуальная реальность в Unity. – М.: ДМК-Пресс, 2016 г. – 316 с.

4. The Global Games Market 2017. [Электронный ресурс] URL: https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/ (дата обращения 09.05.2020).

5. Unity – Manual: PlayerPrefs. [Электронный ресурс] URL: https://docs.unity3d.com/ScriptReference/PlayerPrefs.html (дата обращения: 09.05.2020).

6. User Interface (UI). [Электронный ресурс] URL: https://unity3d.com/ru/learn/tutorials/s/user-interface-ui (дата обращения: 09.05.2020).

7. Липаев В. Тестирование компонентов и комплексов программ – М.: Директ-Медиа, 2015. – 528 с.

8. Албахари Д., Албахари Б. C# 6.0. Справочник. Полное описание языка. – М.: Вильямс, 2017. – 1040 с.

9. Гома Х. UML-проектирование систем реального времени, распределенных и параллельных приложений. – М.: ДМК Пресс, 2011. – 704 с.

10. Официальный сайт Google Play. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.halfbrick.jetpackjoyride (дата обращения: 09.05.2020).

11. Официальный сайт Unity. [Электронный ресурс] URL: https://unity3d.com (дата обращения: 09.05.2020).

12. Язев Ю. Обзор самых популярных движков для разработки игр. [Электронный ресурс] URL: https://xakep.ru/2014/09/05/game-developmentengines-review/ (дата обращения: 09.05.2020).

13. Aras Pranckevičius. Entity Component Systems & Data Oriented Design Unity Training Academy 2018-2019. [Электронный ресурс] URL: http://aras-p.info/texts/files/2018Academy%20-%20ECS-DoD.pdf (дата обращения: 09.05.2020).

14. Robert Nystrom. Game Programming 1 издание. – Patterns Genever Benning, 2014. – 354 с.

15. The Entity-Component-System - An awesome game-design pattern in C++. [Электронный ресурс] URL: https://www.gamasutra.com/blogs/ TobiasStein/20171122/310172/The_EntityComponentSystem__An_awesome_game design_pattern_in_C_Part_1.php (дата обращения: 09.05.2020).

16. Магазин ассетов Unity. [Электронный ресурс] URL: https://assetstore.unity.com/ (дата обращения: 09.05.2020).

17. Zheng L., Dong Y., Yang F. C++ Programming. – Walter de Gruyter GmbH & Co KG, 2019. – 486 p.

18. Анализ рынка виртуальной реальности. [Электронный ресурс] URL: https://habrahabr.ru/post/318868/ (дата обращения: 09.05.2020).