Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
**Томский государственный университет систем управления и радиоэлектроники (ТУСУР)**
Кафедра автоматизации обработки информации (АОИ)

# РАЗРАБОТКА VR-ИГРЫ В ЖАНРЕ КВЕСТ

Пояснительная записка к курсовой работе по дисциплине «Информатика и программирование»

Студент гр. 428-4
_____ Бертман П.Д.
«___» _____ 2020 г.

Руководитель
Доцент каф. АОИ ТУСУР, к.ф.– м.н
_____ _____ Зариковская Н.В.
(оценка) (подпись) (Фамилия И.О)
«___» _____ 2020 г.

Томск 2020

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**Томский государственный университет систем управления и радиоэлектроники (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

# ЗАДАНИЕ

на курсовую работу по дисциплине «Информатика и программирование»

студенту Бертману Павлу Дмитриевичу группы 428-4

факультета систем управления

1. Тема работы: Разработка VR-игры в жанре квест
2. Срок сдачи работы на кафедру: «12» июня 2020 г.
3. Содержание работы:

3.1 Анализ задачи;

3.2 Описание жизненного цикла разработки и роль тестировщика в нём;

3.3 Схема блоков приложения;

3.4 Разработка плана тестирования игры;

3.5 Разработка тест-дизайна игры;

3.6 Использованные технологии;

3.7 Реализация игрового места;

3.8 Руководство пользователя и описание интерфейсов

Задание принял к исполнению: _____

Бертман Павел Дмитриевич «21» февраля 2020г.

# Оглавление

# 1 Введение

Виртуальная реальность (далее - VR) – мир, созданный при помощи технических средств, который передаётся человеку через его ощущения: зрение, слух, тактильные ощущения. Технологии VR прошли огромный путь от первых экспериментов в 60-ых годах XX века до современных шлемов виртуальной реальности. Цель технологии VR - симуляция чувственных данных, которые формируют «реальный» опыт.

Желаемый эффект использования технологии всегда один: погрузить пользователя в виртуальное окружение настолько, чтобы тот начал воспринимать его «как настоящее» (или «почти как настоящее»).

VR изменил и продолжит менять многие области человеческой деятельности по мере внедрения данной технологии. Кардинальные изменения уже были привнесены в индустрию развлечений, архитектуру и дизайн, частично в сферу образования: виртуальная реальность применяется для обучения профессиям, где имеются повышенные риски для человека и эксплуатируемых механизмов. Также VR позволяет презентовать товар или услугу, создать ещё не построенный объект для проверки какого-либо бизнес-проекта.

На сегодняшний день существует несколько типов VR-систем:

— мобильный VR, где основные функции на себя берёт смартфон, который устанавливается в специальную гарнитуру. Данная система удобна в использовании и доступна, если речь заходит о цене, однако, вполне вероятно, что уровень эффекта погружения, как в других системах, может быть не достигнут;

— полноценные VR-шлемы, которые не требуют подключения мобильного устройства и включают в себя все необходимые технические компоненты в одном корпусе: центральный процессор, графический процессор, дисплеи, динамики. Данное устройство полностью автономно и пользуется популярностью благодаря удобству в использовании и

доступности на рынке, однако, цена на шлем больше цены гарнитуры для мобильного VR;

— стационарный VR, который подключается к персональному компьютеру, помимо шлема имеются джойстики. Обеспечивает полное погружение в виртуальный мир.

Возможности VR безграничны и продолжают развиваться, технология с каждым годом становится всё доступнее, о чём говорит рост спроса и выручка сегмента VR. Целью работы является разработка VR-приложения, а назначением самого приложения является перенос пользователя в мир, сгенерированный компьютером, для развития логического мышления и воображения путём решения головоломок. Приложение несёт в себе развлекательный характер, однако, может быть расширено для моделирования функционала реалити-квестов.

Для реализации поставленной цели необходимо решить следующие задачи:

— разработка плана тестирования игры;

— разработка тест-дизайна игры;

— разработка игрового пространства и реализация логики некоторых локаций игры;

— произведения тестирования приложения;

— написания отчета по обнаруженным дефектам во время первой итерации тестирования приложения;

— написания отчета по обнаруженным дефектам во время второй итерации тестирования приложения.

# 2 Разработка VR-игры

## 2.1 Анализ задачи

Перед составлением плана тестирования необходимо определиться с типами тестирования приложения. Для этого нам нужно ознакомиться с особенностями нашего проекта. Такой подход к созданию плана тестирования позволит нам создать наиболее логично выстроить этапы тестирования продукта.

Для составления данного технического документа необходимо изучить необходимо изучить структуру и особенности составления такого типа документов. Это позволит нам создать технически правильную документацию для нашего проекта.

Далее на основании плана тестирования и технического задания игры было необходимо реализовать тест-дизайн. Тест-дизайн – это документ, содержащий в себе все тест-тесты, по которым будет проходить тестирование приложения.

На основании технического задания нужно было создать игровое место «Коридор» и «Комната 5», а также написать все необходимые скрипты для взаимодействия в данных местах, чтобы реализовать эту задачу нужно было изучить возможности трёхмерного редактора Blender и игрового движка Unity.

После проведения тестирования данного продукта было необходимо составить отчёт с подробным описанием дефектов найденных дефектов приложения после первой итерации тестирования приложения и после второй итерации тестирования приложения.

## 2.2 Описание жизненного цикла и роль тестировщика в нём

Модель жизненного цикла программного обеспечения — это структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта.

Жизненный цикл разработки приложения делится на 5 основных этапов:

— сбор и анализ требований – это начальный подготовительный этап, в течение которого необходимо убедиться в возможности реализации проекта. На данном этапе проект необходимо описать и утвердить, написать и утвердить устав и техническое задание, распределить задачи между участниками;

— проектирование – это этап, на котором разрабатывается план работы, проектируется дизайн будущего приложения;

— разработка и программирование – этап, на котором происходит непосредственная реализация проекта командой разработчиков;

— тестирование – это не менее важный этап, чем предыдущие, поскольку позволяет выявить и исправить критические дефекты в разработанном программном обеспечении до ввода в эксплуатацию;

— эксплуатация: финал проекта.

На рисунке 1 представлена модель жизненного цикла разработки программного обеспечения (далее - ПО).


Рисунок 1 – Модель жизненного цикла разработки ПО

Тестировщик – человек, занимающийся тестированием программного обеспечения с целью выявления ошибок в его работе и их последующего исправления. Основная задача тестировщика — найти в программе, приложении, игре или другом продукте все возможные ошибки и проблемы.

Тестировщик моделирует различные ситуации, которые могут возникнуть в процессе использования предмета тестирования, чтобы разработчики смогли исправить обнаруженные дефекты. Таким образом он удостоверяется в надежности продукта с технической и пользовательской точки зрения. Итогом его работы является максимально подробный отчет о проведенном тестировании, в котором должен быть указан анализ и причины возникших проблем.

### 2.3 Схема блоков приложения

В разработанном приложении линейная схема прохождения, на основе этого можно сформировать схему, представленную на рисунке 2.

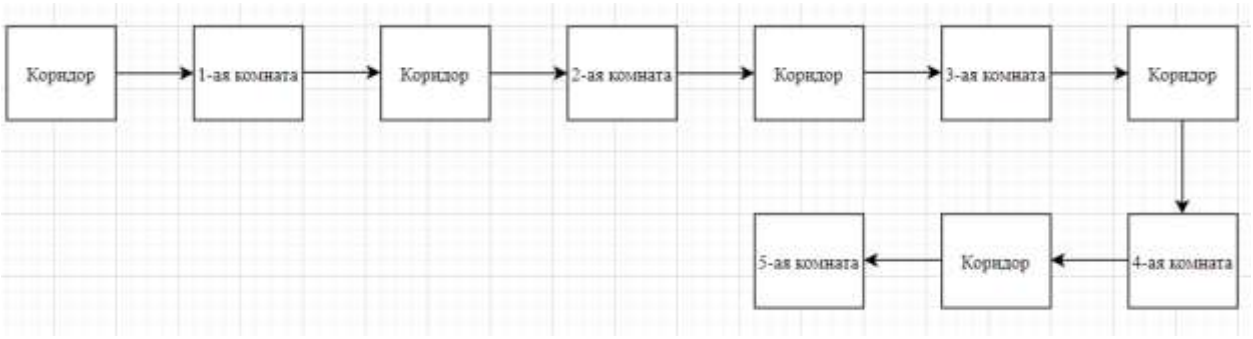


Рисунок 2 – Схема приложения

Попасть в следующую комнату можно только после прохождения предыдущей, при этом необходимо вернуться в игровую локацию «Коридор» и прослушать аудиокассету.

### 2.4 Разработка плана тестирования

Целью тестирования VR игры является выявления критических моментов во время пользования данным приложением. Для этого необходимо провести проверку работы всех написанных скриптов, проверить удобство приложения для пользователя, а также выявление пожеланий пользователя по расширению или исправлению функционала приложения. Перед началом работы тестировщик должен разработать всю необходимую тестовую документацию, в эту документацию как раз и входит план тестирования приложения.

План тестирования представляет собой документ, на основе которого строится всё тестирование программного продукта. Данный документ описывает планирование процесса тестирования. Он определяет необходимый список видов тестирования, которые могут использоваться при проверке качества приложения, требования к техническим ресурсам, среде, на базе которой должно работать приложение.

Для тестирования приложения был выбран метод написания тест-кейсов, которые по своей структуре симулируют поведения пользователя в среде. Далее были выбраны типы тестирования необходимые для тестирования приложения. Это помогает нам логическое представление о том, каким образом будет проходить тестирование.

На основе изученных материалов были выбраны следующие типы тестирования продукта:

— функциональное тестирование - главной задачей функционального тестирования является проверка разработанного программного обеспечения на соответствие требованиям Заказчика;

— регрессионное тестирование - направлено на обнаружение ошибок в проверенной части функционала приложения;

— тестирование конфигурации - данный вид тестирования позволяет проверить работоспособность системы при различных операционных системах и аппаратных конфигурациях;

— тестирование UI – данный вид тестирование помогает убедится в том, что разрабатываемое приложение будет понятно и полезно пользователям;

— тестирование UX - нужно для проверки существующего интерфейс на удобство пользовательских сценариев.

После выбора типов тестирования, что будут использованы для проверки работоспособности игры, перед тестировщиком возникает следующая задача - описать эти самые типы в плане тестирования, в разделе

«Стратегия тестирования». Данный раздел содержит подробное описание каждого типа.

После подробного описания типов тестирования в разделе «Стратегия тестирования» можно приступать к созданию примерного календарного плана тестирования продукта и внести эти данные в раздел «Основные этапы проекта».

Таблица 1 – Основные этапы проекта

| Задача | Начало | Окончание |
|---|---|---|
| Подготовить тест план | 13.05.2020 | 16.05.2020 |
| Подготовить тест дизайн | 26.05.2020 | - |
| Провести основное тестирование | - | - |
| Составить отчёт по найденным дефектам | - | - |

Для того, чтобы во время тестирования результаты были корректными и однозначными, тестировщику было необходимо описать раздел «Система градации серьёзности дефекта», который содержит в своей структуре описание серьёзности дефектов.

Разделы «Старт тестирования» и «Окончание тестирования» описывают при каких условиях проект можно приступать к тестированию продукта и какие требования нужно соблюсти для завершения тестирования.

Конечный итог проведения тестирования описывается в разделе «Результаты».

### 2.5 Разработка тест-дизайна

Следующим этапом подготовки к тестированию приложения, после составления плана тестирования, служит написание тест-дизайна.

Тест дизайн — один из первоначальных этапов тестирования программного обеспечения, этап планирования и проектирования тестов. Тест

дизайн представляет собой продумывание и написание тестовых случаев (тест-кейс), в соответствии с требованиями проекта, критериями качества будущего продукта и финальными целями тестирования. Тест-кейс — это часть документации составляемый тестировщиком, описывающая определённую последовательность действий, с помощью которых можно проверить определенный функционал приложения.

Тест-дизайн выполнен в таблицах Microsoft Excel с целью удобного функционирования в них тестировщику приложения. В данной таблице реализованы критерии для определения приоритета, в котором необходимо устранить дефекты системы, а также подробно реализовано функциональное тестирование с указанием на каких платформах тестирование уже проводилось.

Для начала формируется базовый сценарий поведения в приложении. Базовый сценарий – тип поведения пользователя, при котором пользователь идёт по чётко прописанной линии, не совершая побочный действий, можно сказать, что это симулятор «идеального пользователя».

При создании тест-дизайна приложения нужно учитывать возможность пользователя пойти не по задуманному сценарию разработчика, а следовательно, совершить действия, не входящие в базовый сценарий развития событий.

После написания базового сценария, тестировщик приступает к написанию альтернативных сценариев – они представляют собой разновидность действий пользователя в одной и той же ситуации. Чем больше альтернативных сценариев будет разработано тестировщиком приложения, тем меньше будет у пользователя возможностей нарваться на дефекты и недочеты в работе приложения, которые могут негативно сказаться на настрое пользователя.

Далее на рисунке 3 приведен пример функционально тестирования «Меню» игры.

| Объект проверки | Проверка | | Платформа (Android 7.0) | Платформа (Android 8.0) | Платформа (Android 10.0) | ID бага, комментарий | Тестировщик | Приоритет |
|---|---|---|---|---|---|---|---|---|
| | Шаг | Ожидаемый результат | | | | | | |
| БС1 Работоспособность меню приложения | | | | | | | | |
| БС1 | **Участники: запускают приложение с помощью нажатия на иконку приложения** | | Не тестировался | Не тестировался | Протестировано | | Бертман | |
| БС1 | **Предусловия: все технические характеристики устройства пользователя удовлетворяют требованиям приложения** | Приложение отображает главный экран с 3-мя кнопками: **«Новая игра»**, **«Продолжить»** и **«Выход»**. | Не тестировался | Не тестировался | Протестировано | | Бертман | |
| БС1 | Пользователь находится на главном экране приложения, нажимает первую по расположению кнопку меню **«Новая игра»** | Запускается игровая сцена. Пользователь появляется в локации «Коридор» напротив тумбы с первой кассетой и магнитофоном. | Не тестировался | Не тестировался | Протестировано | | Бертман | |
| БС1 | Сценарий завершён | | Не тестировался | Не тестировался | Протестировано | | Бертман | |
| АС1 Пользователь нажимает кнопку "Продолжить" в меню | | | | | | | | |
| БС1 | Пользователь находится на главном экране приложения, нажимает вторую по расположению кнопку меню **«Продолжить»** | Система не реагирует, так как не выполнены необходимые условия для загрузки автосохранения | Не тестировался | Не тестировался | Протестировано | | Бертман | |
| АС1 | Сценарий завершён | | Не тестировался | Не тестировался | Протестировано | | Бертман | |
| АС2 Пользователь нажимает кнопку «Выход» в меню | | | | | | | | |
| БС1 | Пользователь находится на главном экране приложения, нажимает первую по расположению кнопку меню **«Выход»** | Появляется меню с надписью «Вы действительно хотите выйти?» и имеющим 2 вариантами выбора: «Выход» и «Отмена» | Не тестировался | Не тестировался | Протестировано | | Бертман | |
| АС2 | Пользователь нажимает кнопку всплывающего меню **«Выход»** | Происходит завершение работы приложения | Не тестировался | Не тестировался | Протестировано | | Бертман | |
| АС2 | Пользователь нажимает кнопку всплывающего меню **«Отмена»** | Выполняется переход к сценарию БС1 | Не тестировался | Не тестировался | Протестировано | | Бертман | |
| АС2 | Сценарий завершён | | Не тестировался | Не тестировался | Протестировано | | Бертман | |

Рисунок 3 – Фрагмент тест-дизайн

## 2.6 Использованные технологии

Плана тестирования и тест дизайн были созданы на базе технического задания, составленном для приложения. План тестирования разрабатывался в Microsoft Word. Тест дизайн разрабатывался в Microsoft Excel.

Microsoft Word — текстовый процессор, предназначенный для создания, просмотра, редактирования и форматирования текстов статей, деловых бумаг, а также иных документов, с локальным применением простейших форм таблично-матричных алгоритмов.

Microsoft Excel — программа для работы с электронными таблицами, созданная корпорацией Microsoft для Microsoft Windows, Windows NT и Mac OS, а также Android, iOS и Windows Phone.

Для тестирования приложения использовался смартфон с системой Android 10, эмулятор системы Android Android Studio для проверки работоспособности приложения на других версиях андроид, а также очки Google Cardboard для симуляции виртуальной реальности.

Android Studio — интегрированная среда разработки (IDE) для работы с платформой Android.

Google Cardboard — стандарт компании Google в области виртуальной реальности, в основе которого лежит шлем, главной частью которого служит обычный смартфон. Название связано с простейшим вариантом шлема виртуальной реальности, который, по замыслу разработчиков, можно собрать из двух линз, застёжек и картона.

Все игровые модели создавались в программе Blender версии 2.82, с использованием таких аддонов как, BlenderKit Asset Library, UV Layout, Node Wrangler, для облегчения процесса разработки.

Blender — профессиональное свободное и открытое программное обеспечение для создания трёхмерной компьютерной графики, включающее в себя средства моделирования.

После окончания работ по визуальной части игры, модели переносились в 3D-редактор Unity, для проектирования логической части игры – скриптов,

которые будут отвечать за взаимодействие пользователя с игровым пространством.

Unity – межплатформенная среда разработки компьютерных игр. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие.

### 2.7 Реализация игровой комнаты и коридора

На основе технического задания было разработано игровое место «Коридор» и «Комната №5». «Коридор» является основным местом действия игрока, именно отсюда он и начинает свое первое взаимодействие в игре. На рисунке 4 представлена модель «Коридора». Данное место имеет собственное игровое пространство, созданное с помощь моделей, звуковое сопровождение, предметы с которыми можно взаимодействовать.


Рисунок 4 – Коридор

«Комната 5» - представляет собой заключительное место игрового процесса, для этой комнаты дополнительно был реализован менеджер, отвечающий за логику работы функций в пятой комнате. На рисунке 5 представлена модель «Комнаты №5».


Рисунок 5 – Пятая комната

Листинг программы менеджера, отвечающего за основную логику работы функций в комнате представлен в приложении А.

### 2.8 Руководство пользователя и описание интерфейса

Ниже представлено описание некоторых функций, реализованных в игре. Одной из основных функций в игре является перемещение, без него игрок не сможет взаимодействовать с объектами, находящимися вдалеке. Перемещение игрока производиться с помощью телепортов. Точки, куда игрок может переместиться находятся на полу, посмотрев на одну из них, и сфокусировав на несколько секунд взгляд, игра переместит пользователя в то место, где находиться точка телепортации, на которую был сфокусирован взгляд. Изначальное положение игрока представлено ниже на рисунке 6, далее на рисунке 7 показана работа функции перемещения.

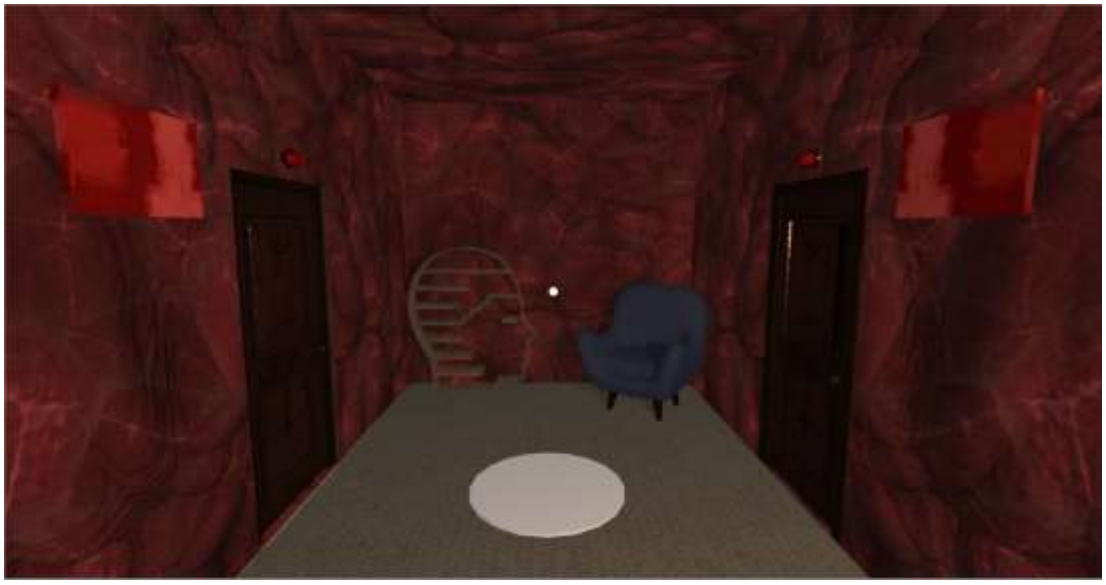

Рисунок 6 – Игрок находится в начальном положении

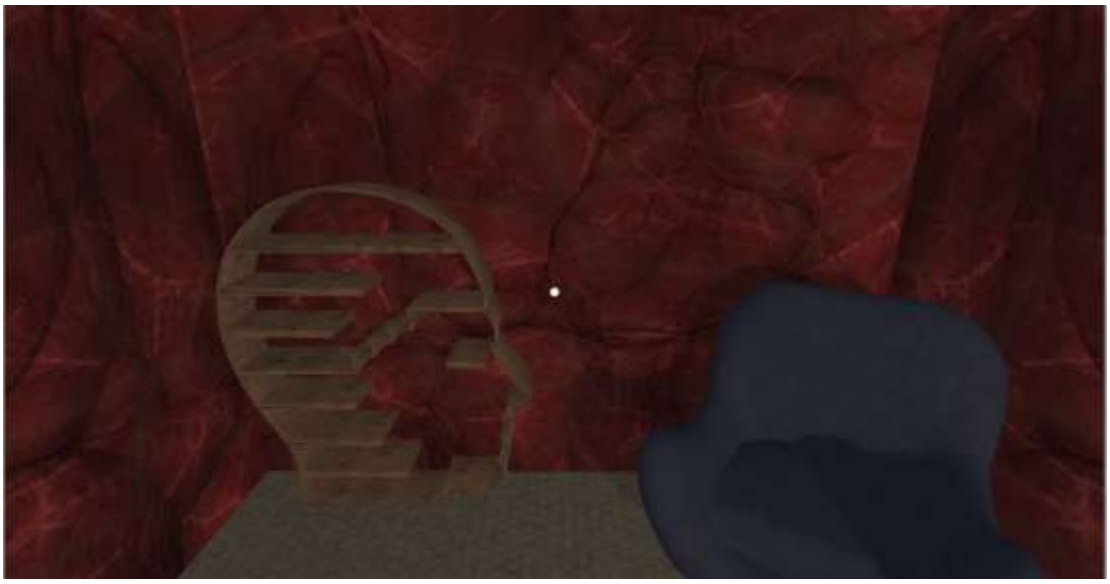

Рисунок 7 – Игрок переместился в другую точку

Следующей основной функцией является взаимодействие с предметами игрового пространства. В игровом пространстве находятся как предметы, с которыми игрок может контактировать, так и те с которыми сделать это невозможно. Все взаимодействия пользователя с игровым миром совершаются фокусировкой взгляда игрового поинтера на объектах.

Если взаимодействие с объектом возможно, сработает анимация поинтера и соответствующее событие. На рисунках 8 и 9, представленных ниже, показано, как именно ведет себя данный курсор-поинтер при взаимодействии с активными объектами.

Рисунок 8 – Игровой поинтер не наведён на объект, с которым возможно взаимодействие

Рисунок 9 – Сработала анимация игрового поинтера при наведении на объект, с которым возможно взаимодействие

Таким образом, можно сказать, что перемещение игрока по игровому пространству и его дальнейшее взаимодействие с активными предметами – это две основополагающие механики, связанные между собой, без которых становится невозможным прохождение игры.

# 3 Заключение

В результате работы была реализована VR-игра в жанре квест. Для достижения данной цели были выполнены следующие задачи:

— разработан плана тестирования игры;

— разработан тест-дизайна игры;

— разработано игровое пространства и реализована логики некоторых локаций игры;

— произведено тестирования приложения;

— написан отчет по обнаруженным дефектам во время первой итерации тестирования приложения;

— написан отчет по обнаруженным дефектам во время второй итерации тестирования приложения.

Были получены навыки оформления документов для тестирования приложения, составления тест дизайна приложения и составления актов о найденных багах. Были приобретены навыки работы в программах по работе с документами Microsoft Word и Microsoft Excel, программы для трёхмерного моделирования Blender, программы для создания игр – Unity. Также был получен опыт работы с VR-технологией, на которой была разработана VR-игра в жанре квест.

## Список использованных источников

1. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. — Минск: Четыре четверти, 2017. — 312 с.

2. Тестирование. Фундаментальная теория [Электронный ресурс] - Режим доступа: https://habr.com/ru/post/279535/ (дата обращения: 26.05.20)

3. Тестирование программного обеспечения - основные понятия и определения [Электронный ресурс] - Режим доступа: https://habr.com/ru/post/279535/ (дата обращения: 26.05.20)

4. План тестирования [Электронный ресурс] - Режим доступа: https://software-testing.ru/forum/index.php?/topic/31705-plan-testirovaniia/ (дата обращения: 28.05.20)

5. Особенности тестирования VR продуктов [Электронный ресурс] - Режим доступа: https://ue4daily.com/blog/VR-QA-howto (дата обращения: 28.05.20)

6. Кто такие тест — дизайнеры и зачем они нужны. [Электронный ресурс]. — Режим доступа: https://www.software-testing.ru/library/testing/test-analysis/2500-roles-and- (Дата обращения 26.05.2020)

# Приложение А

```
public class FifthRoomManager : MonoBehaviour
{
    public GameObject player;
    public GameObject room;
    public AudioClip roomSound;
    public AudioClip doorOpenSound;
    public AudioClip doorCloseSound;
    public GameObject finishWindow;
    public GameObject corridor;

    public GameObject shkaf;
    public GameObject puzzle;
    public GameObject water;
    public GameObject podskaz;
    public GameObject button;
    public GameObject shkafvniz;

    private AudioSource playerSourse;

    private GameObject[] tile = new GameObject[9];

    private bool openDoor;
    private bool key;
    private bool p;
    private bool endRoom;
    private bool buttonF;

    public void HidePodskaz()
    {
        podskaz.SetActive(false);
    }

    void Start()
    {
        playerSourse =
player.GetComponent<AudioSource>();
        playerSourse.Stop();
        playerSourse.clip = roomSound;
        playerSourse.Play();

        buttonF = false;
        endRoom = false;
        key = false;
        room.SetActive(true);
```

```
        openDoor = true;
        GetComponent<Animation>().Play("Door");

GetComponent<AudioSource>().PlayOneShot(doorOpenSound);

        for (int i = 0; i < tile.Length; i++)
        {
            tile[i] =
puzzle.transform.GetChild(i).gameObject;
        }
    }

    void Update()
    {
        if (openDoor)
        {
            if (player.transform.position.z < -1.5)
            {

GetComponent<Animation>().Play("DoorClose");

GetComponent<AudioSource>().PlayOneShot(doorCloseSound)
;
                Invoke("HidePodskaz", 2f);
                openDoor = false;
            }
        }
        else
        {
            if (player.transform.GetChild(0).childCount
> 2)
            {
                if (!key)
                {

shkaf.transform.GetChild(0).GetComponent<MeshCollider>(
).enabled = true;

shkaf.transform.GetChild(1).GetComponent<MeshCollider>(
).enabled = true;
                    key = true;
                }
            }
```

```
            if
(!shkaf.transform.GetChild(3).gameObject.activeSelf &&
key)
            {
                for (int i = 0; i < 9; i++)
                {

tile[i].GetComponent<MeshCollider>().enabled = true;
                }
            }

            if (key)
            {
                p = true;

                for (int i = 0; i < 9; i++)
                {
                    if (tile[i].transform.eulerAngles
!= new Vector3(270f, 0f, 0f))
                    {
                        p = false;
                    }
                }
            }

            if (key && p && !endRoom)
            {
                for (int i = 0; i < 9; i++)
                {

tile[i].GetComponent<MeshCollider>().enabled = false;
                }


water.GetComponent<Animation>().Play("Water");

water.GetComponent<AudioSource>().Play();

water.transform.GetChild(0).GetComponent<Animation>().P
lay("ArmWithKey");
                endRoom = true;
                key = false;
            }
```

```
            if (water.transform.GetChild(0).childCount
== 0)
            {

shkafvniz.GetComponent<MeshCollider>().enabled = true;

                if
(button.transform.GetComponent<MeshCollider>().enabled
== false)
                {
                    buttonF = true;
                }
            }

            if (buttonF)
            {
                room.SetActive(false);
                finishWindow.SetActive(true);
                corridor.SetActive(false);
                playerSourse.Stop();
            }
        }
    }
    }
```