Министерство просвещения Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Уральский государственный педагогический университет»

Институт математики, физики, информатики

Кафедра информатики, информационных технологий
и методики обучения информатике

VR-ИГРА В ЖАНРЕ ГОЛОВОЛОМКА НА ПЛАТФОРМЕ UNITY

Выпускная квалификационная работа

Исполнитель: студент группы ИСиТ-2031
Якименко Н.Н.

Руководитель: Газейкина А.И .
канд. пед. наук, доцент
кафедры ИИТ и МОИ

Допущено к защите
Зав. кафедрой
«__»_____ 2024 г. _____

Екатеринбург – 2024

Реферат

Якименко Н.Н. VR-ИГРА В ЖАНРЕ ГОЛОВОЛОМКА НА ПЛАТФОРМЕ UNITY, выпускная квалификационная работа: 51 стр., 17 рис., 38 библ., приложений 0.

Ключевые слова: VR приложение, игровое приложение, Unity, головоломки.

Предмет разработки: игровое приложение на базе движка Unity для кроссплатформенной разработки игр в жанре головоломка, работающее под операционной системой Windows.

Цель разработки: разработка VR-игры в жанре головоломка с использованием платформы Unity для кроссплатформенной разработки игр.

В рамках выпускной квалификационной работы была изучена межплатформенная среда разработки компьютерных игр Unity. Продуманы игровые механики приложения. Реализовано игровое приложение на платформе Unity3D в соответствии с техническим заданием. Проведена апробация мобильного игрового приложения методом экспертных оценок.

Оглавление

ВВЕДЕНИЕ..... 4

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ VR-ИГР..... 6

1.1 История игр и виртуальной реальности 6

1.2 Выбор средств разработки игрового VR-приложения..... 12

1.3 Техническое задание 17

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИГРОВОГО VR-ПРИЛОЖЕНИЯ НА ПЛАТФОРМЕ UNITY 21

2.1 Модельные представления объекта разработки..... 21

2.2 Этапы разработки игрового VR-приложения 23

2.3 Тестирование игрового приложения. Результаты апробации 49

ЗАКЛЮЧЕНИЕ 51

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ..... 52

ПРИЛОЖЕНИЯ 57

Приложение 1. 57

Введение

В 21 веке игры стали частью жизни многих людей, зачастую они заменяют книги, газеты, фильмы. Игры позволяют человеку, стремления которого не могут быть реализованы в реальности, реализовать их в компьютерных играх. Они способны полностью погрузить в виртуальный мир, давая почти реальные ощущения наших взаимодействий с объектами. Поэтому существует огромный список и выбор игр, каждый может выбрать то, что подходит под его потребности, желания: можно строить города, покорять космос, выживать на необитаемом острове, пережить катастрофу, участвовать в спортивных мероприятиях, где действующим персонажем будет пользователь. Можно самостоятельно пройти историю персонажа, а не просто быть наблюдателем, как в книгах или кино.

С каждым годом технологии развиваются, а вместе и с ними и игры, они становятся более продвинутыми, изображение становится более кинематографичным и всесторонними, что делает их привлекательными для еще большего количества людей. Развитие технологий привело к тому, что игры начали создавать более иммерсивными, большое желание погрузиться как можно больше привело к началу технологий виртуальной реальности.

Виртуальная реальность (VR) – это одна из захватывающих и инновационных технологий нашего века. VR предлагает опыт полного погружения в виртуальный мир, обеспечивая перед нами бесконечные возможности в сферах жизни, работы, учебы, включая как и развлечение, и работу, и обучение.

Среди всех разновидностей приложений для виртуального мира, можно отметить, что самыми популярными являются игры, но также в последние годы в разработке появились приложения для учебного процесса, например, для обучения студентов-медиков, рассмотрение работы органов, расположения и вида при различных болезнях. VR используется и в обучении пилотов и водителей, позволяя изучить основы управления самолетами и машинами в безопасной среде, не подвергая опасности не их или окружающих людей.

Среди прочего эта технология используется и военными для отработки тактики и стратегии без рисков и моделирования различных сценариев проведения действий.

В этапах создания игры изучим основные теории и законы, лежащие в основе технологий виртуальной реальности, познакомимся с особенностями движка Unity, являющийся некоторым из популярных и мощных инструментов для разработки как приложений, так и игр.

Разработка VR-игры в жанре головоломки даст возможность не только погрузиться в виртуальный мир, но и развить навыки в создании интерактивного, захватывающего опыта. Эта работа представляет собой практическую исследовательскую задачу, которая открывает новую область разработки VR-игр и расширит навыки и знания в сфере IT-разработок.

Предмет разработки: игровое VR-приложение на платформе Unity для разработки игры в жанре головоломка, работающее под управлением операционной системы Windows.

Цель разработки: разработать VR-игру в жанре головоломка с использованием платформы Unity.

Задачи:

1. Провести анализ истории развития компьютерных игр, VR, изучить термины, необходимые для создания VR игры.
2. Проанализировать существующие игры данного жанра, платформы для разработки игр и их возможности, освоить среду разработки Unity и язык программирования C#, используемый в Unity.
3. В соответствии с техническим заданием провести разработку игрового VR-приложения на основе выбранных технологий и программных обеспечений.
4. Провести апробацию разработанного приложения методом экспертных оценок.

Глава 1. Теоретические основы разработки VR-игр

1.1 История игр и виртуальной реальности

История компьютерных игр начинается в 1947 году, когда в академической среде разрабатывались простые игры и симуляции. Компьютерные игры длительное время не были популярны, и только в 1970-х и 1980-х годах, когда появились доступные для широкой публики аркадные автоматы, игровые консоли и домашние компьютеры, компьютерные игры становятся частью поп-культуры [29].

В 1962 году группа программистов и сотрудников различных престижных университетов, например Гарвард, а также бывших студентов Массачусетского технологического университета создала одну занятную поделку на новейшем тогда DEC PDP-1. Этот компьютер был первым продуктом компании DEC, созданную также студентами MIT. До этого они работали над компьютерами серии TX, а именно TX-0, TX-1, TX-2. Это были одни из первых транзисторных компьютеров, однако они, как и все системы до этого, был лабораторным, не предназначенным для продажи. Используя наработки своих прошлых работ, компания DEC выпустила PDP-1. Выпустить им удалось не очень много копий, не больше 50, при этом одну из них компания подарила MIT.

На этой машине и создали Spacewar, компьютерную игру, которую многие называют первой. Понятно откуда это исходит, ведь компьютеры начали потихоньку вылезать из лабораторий, из-за чего эту игру кое-как, но смогли распространить. Команда, которая прозвала себя Hingham Institute, благодаря Spacewar смогла даже попасть в другие престижные компании и заведения, как, например, Стэндфордский университет и сама компания DEC. Таким образом, они и распространили игру за пределы компьютерной комнаты в MIT. Так что Spacewar! официально считается первой популярной компьютерной игрой [28].

Также ей приписывают звания первой из-за наследия, которая она оставила индустрии. Самая популярная игра своего времени, стоявшая на всех доступных в те года системах, в том числе на всех компьютерах PDP. Понятие порта игры с одной системы на другой появилось, когда Spacewar! Начала массово распространяться. Конечно, в первую очередь игрой интересовались программисты, которые понимали, как работает компьютер и имевшие доступ к системам с мониторами. По этой игре был проведён первый в истории турнир по компьютерным играм, а именно «Intergalactic Spacewar Olympics» в 1972 году [28].

Исследования демонстрируют длительный положительный эффект от видеоигр для таких основных психических процессов, как восприятие, внимание, память и принятие решений. Активные видеоигры требуют, чтобы игроки быстро двигались, следили за многими элементами одновременно, сразу же обрабатывали новую информации и выдавали решения за доли секунды. Многие из способностей, отточенных такими играми, психологи считают основными строительными блоками интеллекта. Они помогают улучшению пространственного внимания, развивают способность отслеживать движущиеся объекты, снижают импульсивность, помогают преодолеть дислексию. Видеоигры улучшают способность мозга решать несколько задач одновременно (мультитаскинг) и повышают гибкость ума. В Женевском университете профессор Дафна Бавельер сравнила визуальные способности геймеров и не-геймеров. Испытуемые должны были следить за положением нескольких объектов, движущихся на экране, и постоянно переключать свое внимание с одной части экрана на другую, в то же время оставаясь бдительными для других событий. Оказалось, что люди, которые играют в видеоигры, выполняют такое задание значительно лучше, чем те, кто этого не делают. Такая игра учит мозг обрабатывать входящую визуальную информацию эффективнее и быстрее [35].

В современном мире создание видеоигр является одним из наиболее крупных сегментов индустрии развлечений. Масштабы игровой индустрии

сопоставимы, например, с киноиндустрией. А по скорости роста за последние пять лет индустрия видеоигр существенно ее опережала. По степени влияния на потребителей и вовлеченности их в интерактивное окружение, предлагаемое видеоиграми, этот сегмент уже давно выделяется среди других видов развлечений [36].

Также среди тенденций развития видеоигр можно отметить появление официальных соревнований по видеоиграм – киберспорт. На текущий момент это целая ответвление, как и футбол. Есть свои команды, тренеры, спонсоры. Люди соревнуются в своих способностях, не только связанных с отточенными навыками, но и с гибкостью ума, стратегией и её реализации. В Российской Федерации на текущий момент киберспорт является признанным видом спорта.

История развития виртуальной реальности начинается примерно в одно время с компьютерными играми, в 1962 году Мортон Хейлинг, создал первое устройство виртуальной реальности, Sensorama [31].

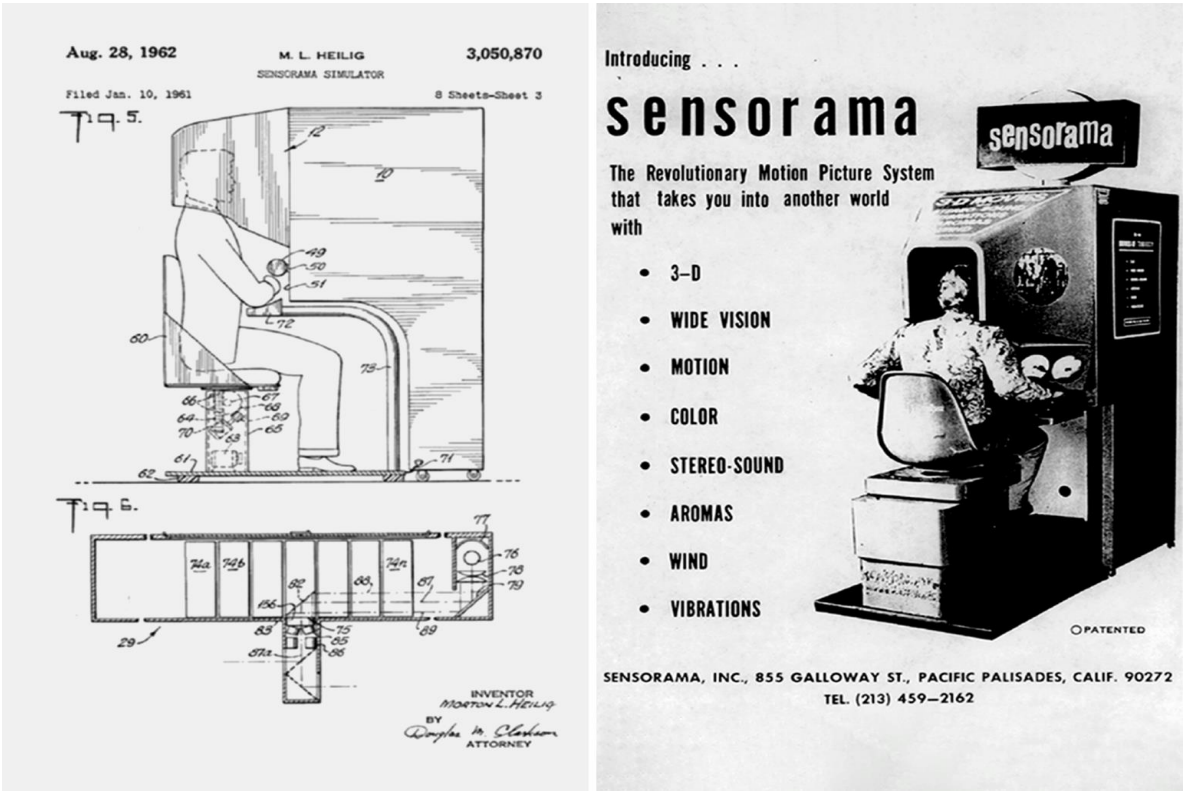

Рис. 1 Morton Heilig. The Sensorama. U.S. Patent #3050870 / Sensorama Inc.

Работало это так: зритель садился на специальное кресло, помещал голову в короб, где находились экран и динамики, и смотрел заранее записанные ролики. Например, можно было покататься на мотоцикле по Бруклину с эффектом полного погружения: динамики воспроизводили гул мотора, вентиляторы имитировали ветер, кресло вибрировало, а внутри ощущались запахи города [31].

Также Мортон Хейлинг разработал VR-очки, которые похожи на актуальные средства для VR, например Meta Quest 3.

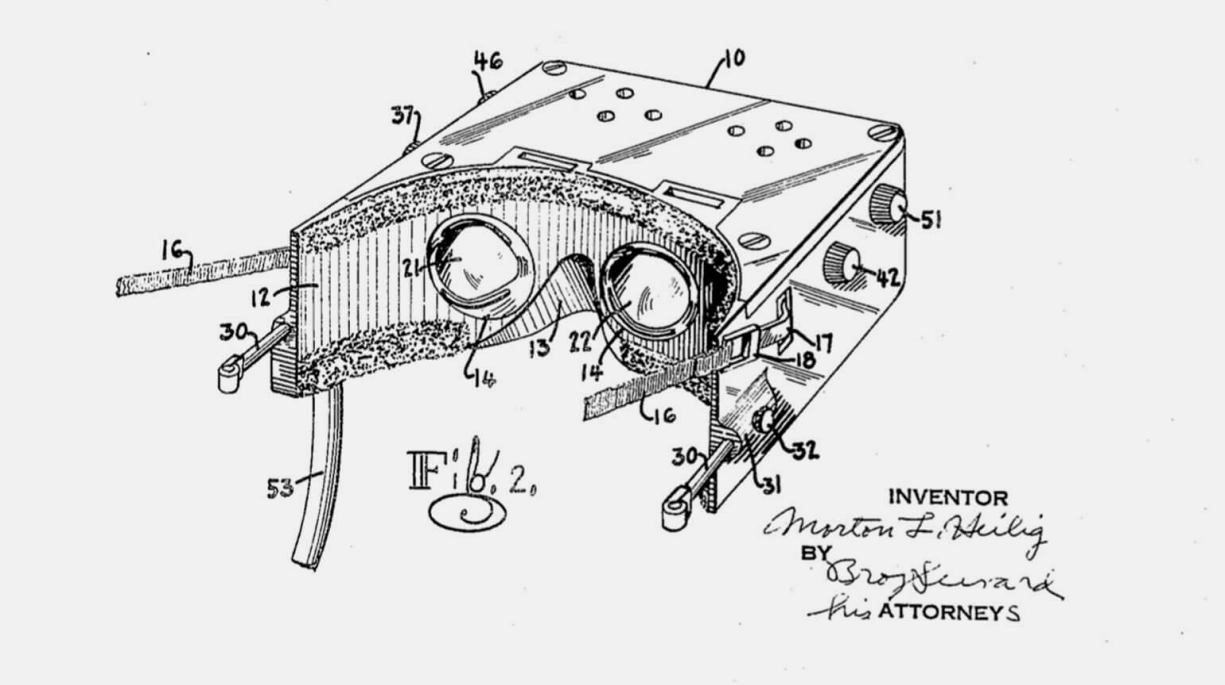

Рис. 2 Morton L Heilig. U.S. Patent 2955156A

Не все ранние вариации устройств виртуальной реальности нельзя назвать удачными, технологии времени были ограничены и работы было отложены. Так первые шлемы и первые игры к ним появились только в 1990-х, появилась игровая консоль Nintendo Virtual Boy, но из-за особенностей использования консоли, у многих возникала морская болезнь, что стало причиной закрыть проект.

Рис. 3. Nintendo Virtual Boy.

В 2012 году в VR-индустрии произошел прорыв. На выставке Electronic Entertainment Expo был представлен первый прототип Oculus Rift. VR-гарнитура имела высокое разрешение 1080x1200 и угол обзора 110 градусов. Устройство стало большим прорывом в сравнении с предыдущими подобными разработками. После такого события все крупные компании решили разрабатывать свои гарнитуры. Так виртуальная реальность стала примером воплощения смелых фантазий человека в реальность. Благодаря энтузиазму и жажде нового, технология получило своё развитие и популярность на текущий день.

Рис. 4 Oculus Rift на E3 2014

Рассмотрев историю игр, можно сказать, что игры отличаются постоянным развитием, их популярность растет с каждым годом и все компоненты игры, а именно графика, геймплей, сюжет, геймдизайн и множество визуальных эффектов, все они стремятся использовать самые передовые и инновационные технологии, видеоигры на текущий момент сильно влияют на культуру развлечений среди молодежи и людей среднего возраста. Тоже можно и сказать и про виртуальную реальность, в начале своего пути она не казалась перспективной, а также не возымела популярности в начале своего пути развития, но по истечению времени и развития технологий, виртуальная реальности популяризировалась и на текущий момент есть множество VR-гарнитур, некоторые из них могут использоваться даже в режиме AR, например «Apple Vision Pro». Помимо всего этого VR поспособствовал развитию видеоигр, предоставив новые горизонты для игроков и разработчиков. В дальнейшем можно ожидать развитие и следование тенденциям инноваций в области игр и VR. Графика и аудио будут улучшаться, возможно даже появятся новые жанры игр, а также виртуальная реальность покинет пределы игровой индустрии и станет популярна повсеместно.

1.2 Выбор средств разработки игрового VR-приложения

Игровой движок – программное обеспечение для создания компьютерной игры. Обычно к игровым движкам рекомендуется использовать или созданные для них IDE, или плагины для существующих.

Разделение игры и игрового движка часто расплывчато, и не всегда проводят чёткую границу между ними. Но в общем случае термин «игровой движок» применяется для того программного обеспечения, которое пригодно для повторного использования и расширения, и тем самым может быть рассмотрено как основание для разработки множества различных игр без существенных изменений [33].

Обобщённо говоря, игровой движок ответственен за организацию и поведение игровых объектов, а также за их отображение на экране. Ваша же задача - выбрать, как они будут выглядеть и как себя вести. Для этого движок предоставит вам возможность создавать и удалять объекты, задавать их параметры, добавлять логику и управлять ресурсами [https://dtf.ru/u/48338-akkaunt-ne-ispolzuetsya/236542-kak-razobratsya-v-igrovyh-dvizhkah].

Для разработки игры можно использовать большое количество игровых движков, все они отличаются по возможностям, языкам программирования, платформы для которой игра создается и самой реализации разработки игры. Также средства разработки разделяются на полноценное ПО и фреймоврки, в нашем случае мы будем использовать полноценное ПО для облегчения работы с VR и использованием готовых моделей, хочется выделить некоторые из таких движков, а именно:

- **Unity**
- **Unreal Engine**

Эти движки поддерживают необходимую технологию виртуальной реальности.

Unity

Unity - кроссплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет со-

здавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие [19].

Этот игровой движок доступен по бесплатной лицензии, что дает возможность энтузиастам и небольшим студиям использовать её для разработки своих игр, но есть свои ограничения, связанные с бесплатной лицензией, среди них можно выделить несколько:

- Показ лого Unity перед запуском приложения
- Коммерческий проект созданный в Unity не должен приносить больше 100 тысяч долларов в год.

Редактор Unity оснащен простым для пользователя Drag&Drop интерфейсом. Один из плюсов, плагин KALI так как его легко настраивать под свои нужды: состоит он из различных окон, с помощью чего отладку игры можно производить прямо в редакторе. Среда разработки использует для написания скриптов язык программирования C#. Расчёты физики производит физический движок PhysX от NVIDIA для 3D физики и Box2D для 2D физики. Графический API - DirectX [19].

В интерфейсе есть несколько разделов, которые отвечают за разные элементы разработки - ассеты (шаблоны элементов), игровые объекты, настройку их свойств и параметров:

- **Scene** - окно сцены, в котором выстраивается игровое пространство (элементы игрового мира, текстуры, фигурки персонажей и прочее).
- **Games** - окно игры, в котором можно посмотреть глазами пользователя, как будут двигаться элементы и работать игровые механики.
- **Hierarchy** - окно иерархии, в нем перечислен список всех элементов (GameObject), которые помещены в окно Scene.
- **Project** - система папок, в которых хранятся ассеты по категориям (текстуры, шрифты, звуки и т.д.).
- **Console** - программисты используют ее для отладки и поиска ошибок.

- **Inspector** - окно для изменения элементов игры, их размера, цвета, положения в пространстве и других характеристик [37].

Все эти разделы можно настроить по своему усмотрению, расположить и расставить в любое место, а также изменить занимаемое ими место.

Unreal Engine

Unreal Engine - игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Первой игрой на этом движке был шутер от первого лица «Unreal», выпущенный в 1998 году. Движок задумывался только для шутеров от первого лица, но в дальнейшей разработке его стали применять и к играм другого жанра и довольно успешно [21].

Ранее Unreal Engine был платным и распространялся при помощи ежемесячной подписки, но начиная с 2015 года он стал бесплатным, но с определенным условием, похожим с Unity, лицензия является бесплатной пока коммерческий проект приносит меньше одного миллиона долларов прибыли.

Игровой движок применяется в многих сферах и является кроссплатформенным. Среди особенностей можно отметить следующее:

- Ориентированность на 3D – движок разрабатывался исключительно для 3D, но ближе к текущему моменту времени компания разработчик «Epic Games» постепенно начали добавлять возможности для создания 2D игр.
- Оптимизация для языка программирования C++ - основным языком программирования в UE является C++. Это мощный и быстрый, но при этом довольно трудный, с которым очень сложно начинать с нуля. Использование данного ЯП и является основным отличием от первого движка Unity, игру создать будет сложнее, но если и получится, то она будет стабильнее, быстрее, эффективнее.
- Наличие языка Blueprints – Так как C++ является основным языком программирования, нужен был язык, с которым справится не только программист, но и аниматор. Поэтому для UE был создан Blueprints, визуальный язык программирования, позволяет создавать игровую ло-

гику при помощи блок-схем, однако она не эффективна в отличии от логики, написанной на C++. Blueprints облегчает разработку, но и сильно ограничивает ресурсы.

- Широкие графические возможности – в данном игровом движке есть огромное количество возможностей для создания трехмерной фотореалистичной графики. Множество текстур, спецэффектов и материалов, которые применяются к объектам, дабы изменить их вид. Графика гибко настраивается и по итогам можно создать и пиксельную игру и фотореалистичную.

На игровом движке Unreal Engine можно создать хорошую, оптимизированную игру, при знании языка C++, а также добавить к ней фотореализма.

Для разработки был выбран движок Unity, он является самым популярным среди независимых разработчиков, у него есть большая библиотека моделей и скриптов. Среди рассмотренных особенностей игровых движков, в Unity используется язык программирования C#, а также инструментарий визуального программирования «Script Graphs» являются более доступными к освоению, чем C++, на фоне небольшого опыта разработки, стабильность движка Unreal Engine не будет заметна, поэтому для разработки был выбран именно Unity, он популярен, прост и предоставляет множество шаблонов, плагинов и документаций для создания игр.

После выбора необходимого движка для разработки игр, необходимо выбрать технологию при использовании VR, а именно в Unity представлены следующие:

- Oculus
- OpenXR
- OpenVR

Первая технология Oculus представляет собой реализацию для гарнитур от компании «Meta Quest», нам она не подходит, гарнитура разработчика была создана другой компанией.

Вторая технология OpenXR является полностью бесплатной и разработана для виртуальной реальности и дополненной реальности, для которой в Unity представлены обучающие шаблоны для разработчиков, и она подходит для гарнитуры разработчика, ведь это является базовой технологией при запуске VR-гарнитуры, а именно использовался «Портал смешанной реальности», установленный по умолчанию в издание «Pro» у Windows 10,11.

Третья технология OpenVR разработана компанией Valve, является наиболее универсальным решением для большинства представленных VR-гарнитур которые сейчас существуют, а также имеет большую стабильность на фоне остальных.

Изучив представленные технологии, была выбрана OpenVR (Рис. 5.), а также был использован плагин для Unity под названием SteamVR. В нем представлены нужные документации, скрипты, а также является самым утилитарным решением, ведь приложение можно будет использовать для большинства гарнитур, а это и требуется от игрового приложения, быть совместимым для большинства систем.

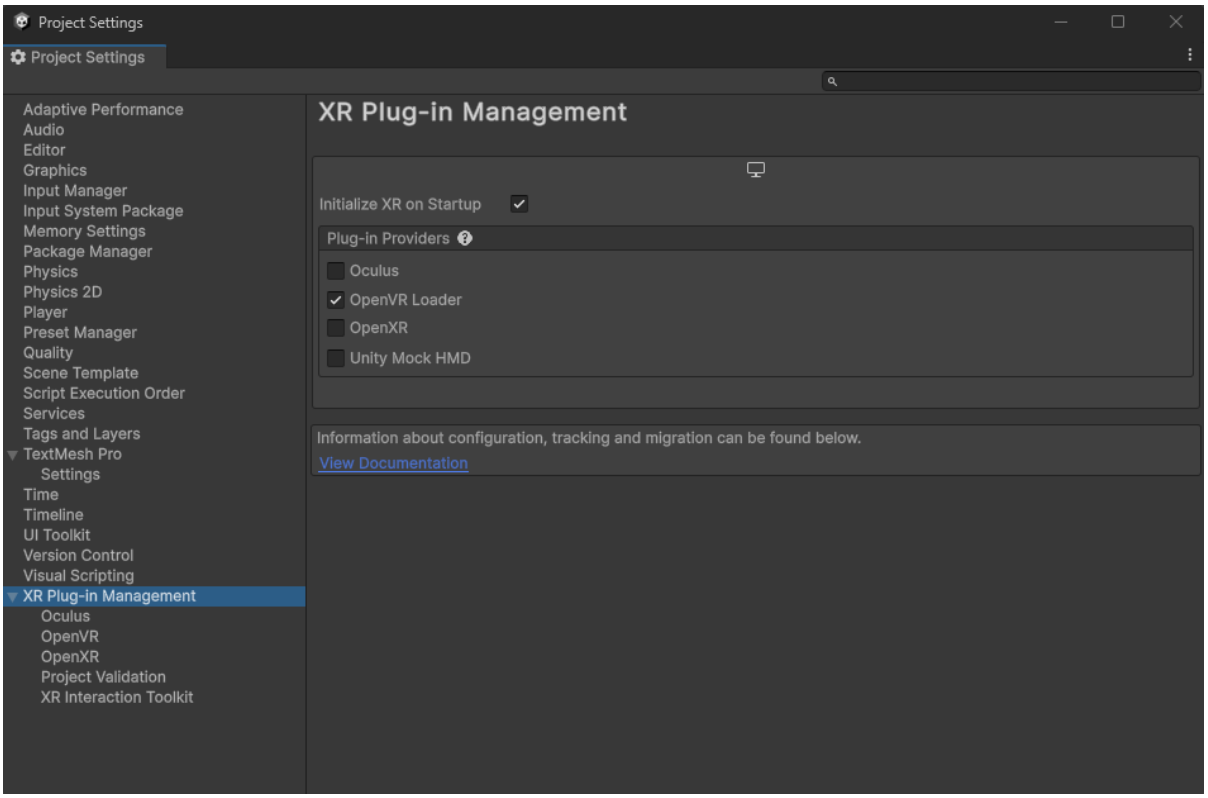

Рис. 5. Выбор технологии OpenVR

После тщательного анализа игровых движков и технологий виртуальной реальности можно сделать несколько выводов. Unity и Unreal Engine являются двумя наиболее популярными игровыми движками, у каждого из них свои уникальные возможности. В нашем случае был выбран Unity в качестве основного инструмента разработки игрового приложения из-за его интуитивного интерфейса, возможностей для создания VR-приложения и сильного сообщества разработчиков. В отношении выбора VR-технологий были рассмотрены OpenVR, OpenXR, Oculus. Среди них был выбран Valve OpenVR в качестве главной технологии реализации проекта из-за его универсальности и совместимости с различными VR-устройствами. OpenVR предоставляет единый интерфейс для работы с множеством гарнитур, что сильно упрощает процесс разработки и расширяет доступность проекта для аудитории.

Таким образом, выбрав Unity и OpenVR главными инструментами, обеспечивается оптимальное сочетание для создания доступного и качественного игрового VR-приложения.

1.3 Техническое задание

на разработку игрового приложения

«Игровое приложение на Unity в жанре VR головоломка»

Составлено на основе ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы» [38].

1. Общие сведения.

1.1. Название организации-заказчика.

ФГБОУ ВО «Уральский государственный педагогический университет»

1.2. Название продукта разработки.

«Игровое приложение на Unity в жанре VR головоломка»

1.3. Назначение продукта.

Для пользователей, которые увлекаются играми.

1.4. Плановые сроки начала и окончания работ.

В соответствии с планом выполнения ВКР (01.09.2023 – 20.05.2024).

2. Характеристика области применения продукта.

2.1. Процессы и структуры, в которых предполагается использование продукта разработки.

Игровое приложение предполагается опубликовать в магазине приложений Itch.io, с целью его апробации пользователями различных категорий, сопровождения и доработки.

2.2. Характеристика персонала (количество, квалификация, степень готовности)

Разработчик: владение межплатформенной средой разработки Unity3D, знание языка программирования C#, знание структуры геймдизайна.

Пользователь: Приложение Steam, VR гарнитура.

3. Требования к продукту разработки.

3.1. Требования к продукту в целом.

Игровое VR приложение, устойчиво работающее под управлением операционной системы Windows в приложении SteamVR.

3.2. Аппаратные требования.

Минимальные технические требования к ПК для работы с приложением:

1. Операционная система: Windows 10 и выше
2. Размер ОЗУ: 8 ГБ
3. Процессор: Intel i5-4590 или аналог от AMD
4. Видеокарта: Nvidia GTX 970 / AMD R9 290 и GTX 980 для ноутбуков
5. Порты: HDMI 1.3, три порта USB 3.0
6. Свободного места на HDD: 8 ГБ

Рекомендуемые технические требования к ПК для работы с приложением:

1. Операционная система: Windows 11
2. Размер ОЗУ: 16 ГБ

3. Процессор: Intel i5 8-го поколения и старше или аналог от AMD

4. Видеокарта: Nvidia RTX 20-го поколения и старше или аналог от AMD

5. Порты: DisplayPort 2.0, USB-A 3.2, USB-C 3.2

6. Свободного места на SSD: 8 ГБ

3.3. Указание системного программного обеспечения (операционные системы, браузеры, программные платформы и т.п.).

OS Windows 10 и выше

3.4. Указание программного обеспечения, используемого для реализации.

- Unity3D 2022.2.16f1
- Visual Studio 2022

3.5. Особенности реализации серверной и клиентской частей.

Не предусмотрено.

3.6. Форматы входных и выходных данных

- Входные данные: Данные о положении и ориентации головы, контроллеров, ввод с контроллеров, данные сенсоров и трекеров.
- Выходные данные: изменение отображаемых данных (управление персонажем, управление игровыми объектами, изменение настроек приложения).

3.7. Источники данных и порядок их ввода в систему (программу), порядок вывода, хранения.

Не предусмотрено.

3.8. Порядок взаимодействия с другими системами, возможности обмена информацией.

Не предусмотрено.

3.9. Меры защиты информации.

Не предусмотрено.

4. Требования к пользовательскому интерфейсу.

4.1. Общая характеристика пользовательского интерфейса.

WIMP интерфейс.

4.2. Размещение информации на экране, дизайн экрана.

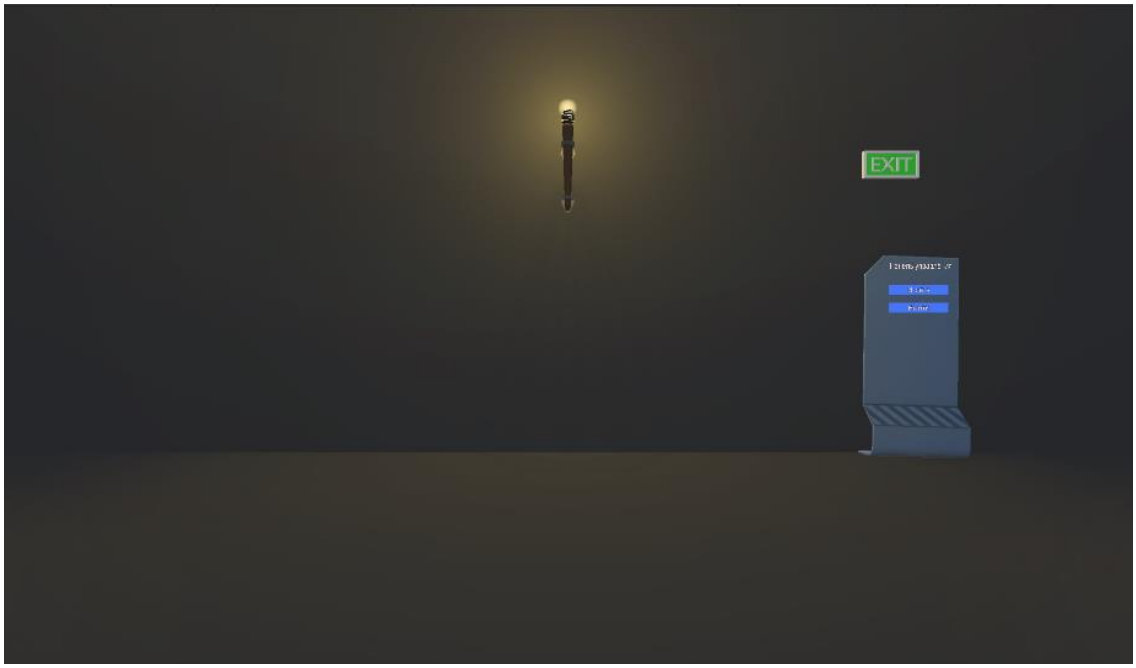

Рис. 6 Главное меню

4.3. Особенности ввода информации пользователем, представление выходных данных.

Взаимодействие с контроллерами: В VR-приложениях пользователи могут использовать контроллеры. Они могут держать контроллеры в руках и использовать кнопки, джойстики, сенсорные поверхности или жесты для выполнения различных действий, таких как перемещение, выбор объектов, взаимодействие с интерфейсом и другие.

Визуализация виртуального мира: VR-приложения создают виртуальный мир, который отображается на экране гарнитуры пользователя. Это может быть трехмерное окружение, сцена или интерфейс, который представляет виртуальную среду и объекты в ней.

5. Требования к документированию.

5.1. Перечень сопроводительной документации.

Руководство пользователя приложения.

5.2. Требования к содержанию отдельных документов.

Не предусмотрено.

Глава 2. Проектирование и разработка игрового VR-приложения на платформе Unity

2.1 Модельные представления объекта разработки

Функциональные диаграммы – это диаграммы отражающие взаимосвязи функций разрабатываемого игрового приложения. Каждый блок такой диаграммы (Activity) соответствует некоторой функции, для которой должны быть определены: исходные данные, результаты, управляющая информация и механизмы ее осуществления.

Для определения функционала возможностей, которые будут заложены в систему, и как будут происходить взаимодействия внутри системы, была построена функциональная модель.

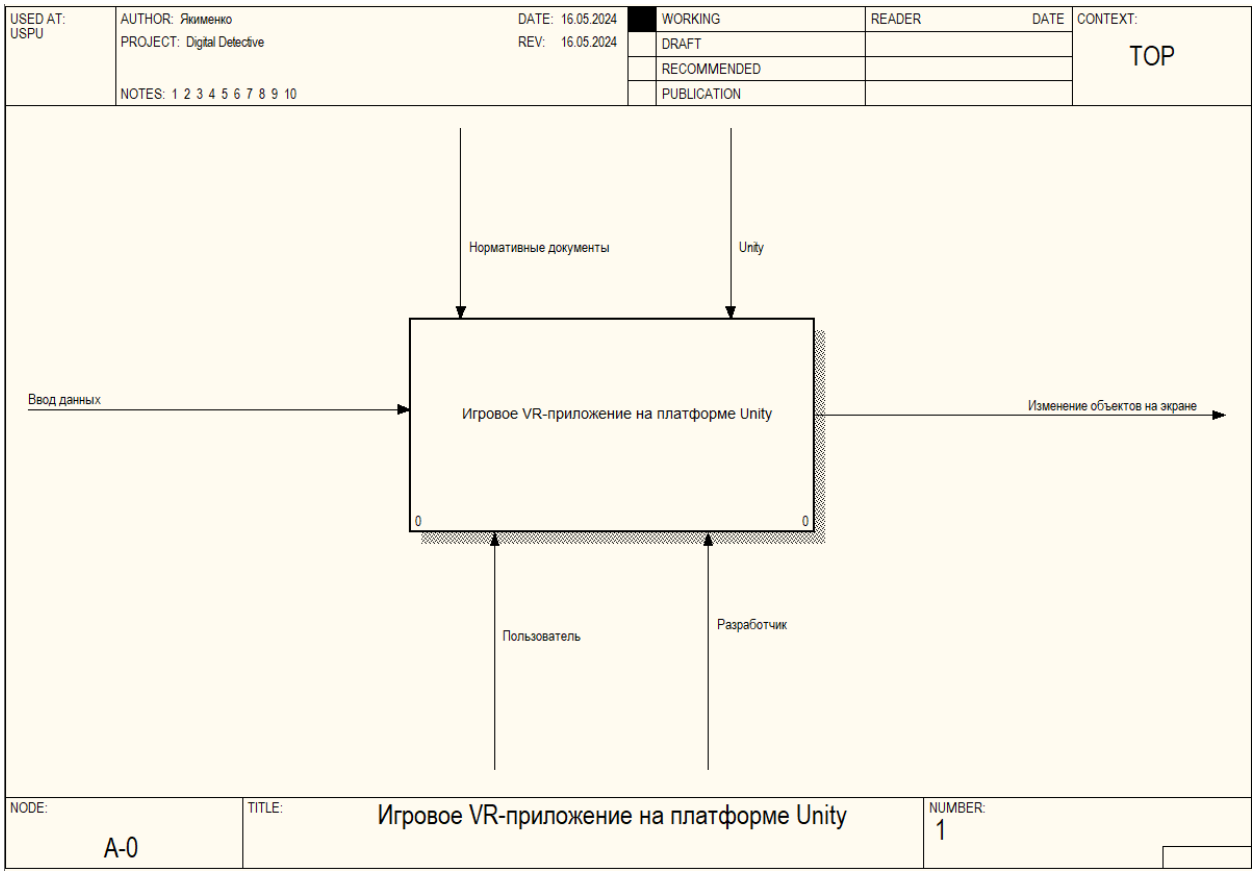

Рис. 1. Диаграмма IDEF0 для игрового VR-приложения на платформе Unity. Уровень A0 в функциональной модели

Исходя из данных диаграммы уровня А0, были определенны:

- входные данные – управляющая информация с сенсорного экрана устройства;
- управляющие данные – нормативные документы, фреймворк Unity;
- механизмы взаимодействия – пользователь, разработчик;
- выходные данные – графическая информация, а именно изменение объектов на экране;

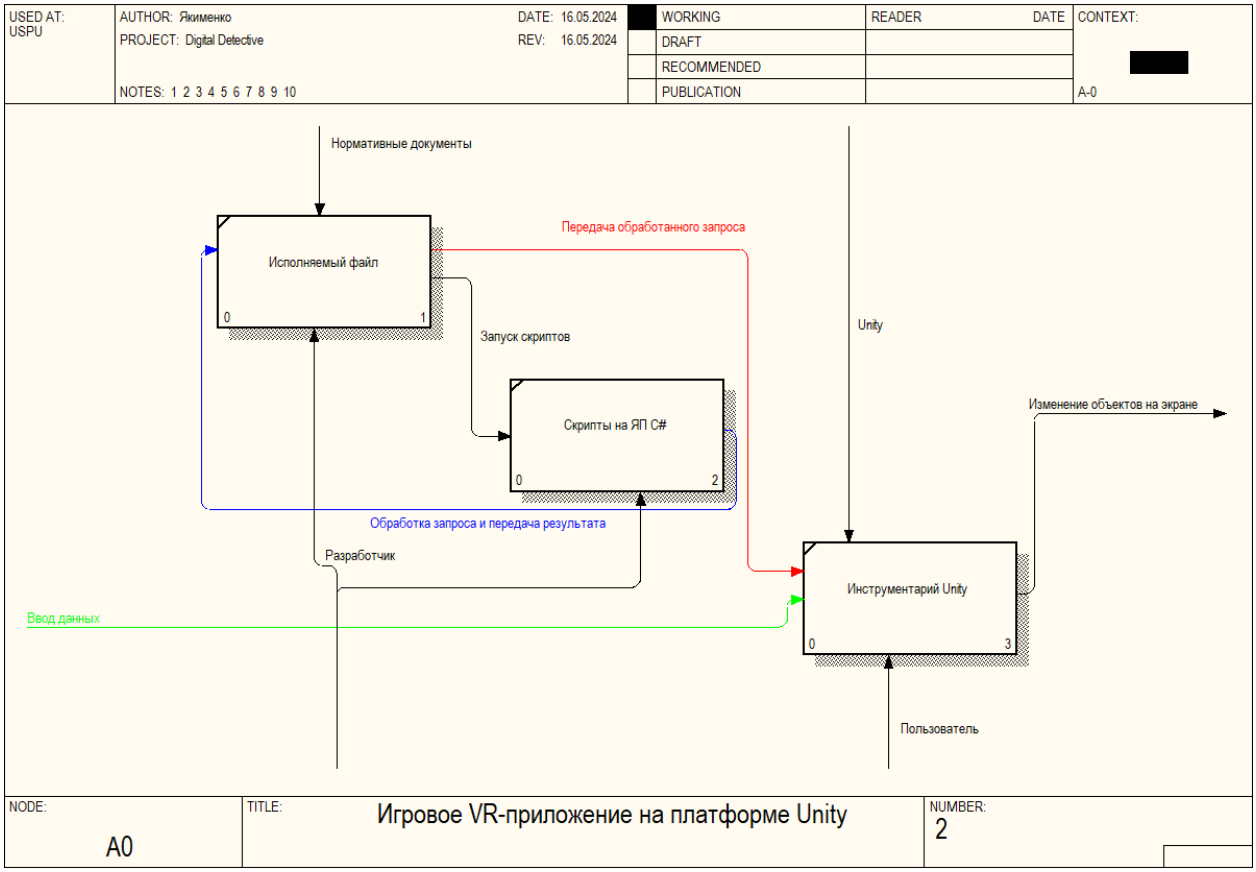

Рис. 2 Диаграмма IDEF0. Декомпозиция уровня A0 – уровень A1 в функциональной модели.

Исходя из данных диаграммы уровня A1, декомпозиции уровня A0, были определенны основные функции:

- Исполняемый файл
- Скрипты на языке программирования C#
- Инструментарий Unity

Функция «Исполняемый файл» получает на вход нормативные документы и взаимодействует с разработчиком. От функции «Инструментарий Unity»

в него приходит введенные сигналы пользователем и тогда на выходе в функцию «Скрипты на языке программирования C#» идет запуск скриптов. Так же на вход от функции «Скрипты на языке программирования C#» приходит обработка запроса и на выход в функцию «Инструментарий Unity» отходит передача обработанного запроса.

Функция «Скрипты на языке программирования C#» взаимодействует с разработчиком. На вход от функции «Исполняемый файл» приходит запуск скриптов, а на выходе в эту же функцию возвращается обработка запроса и передача результата.

Функция «Инструментарий Unity» получает на вход ввод данных и фреймворк Unity, взаимодействует с пользователем. Передает функции «Исполняемый файл» введенные сигналы пользователем и получает от этой функции передачу обработанного запроса. На выход идет изменение объектов на экране.

Разработка диаграмм IDEF0 для игрового приложения позволила структурировать и визуализировать процесс разработки, выявив основные этапы и их связь. Диаграммы способствовали повышению эффективности процесса разработки, но и дала основу для управления проектом.

2.2 Этапы разработки игрового VR-приложения

По результатам разработки было создано игровое приложение в жанре VR-головоломка на платформе Unity. При использовании игрового движка Unity можно обойтись без повторной реализации представленных инструментов для управления физики, звуков, сцен и дополнительных плагинов, и полностью посвятить время разработки своего продукта и программированию необходимых скриптов.

Игровое приложение в жанре VR-головоломка предполагает разнообразие уровней и находящихся в них головоломок. Некоторые вдохновители для создания VR-игры, связанные с VR и головоломками:

- The Talos Principle VR

- Keep Talking And Nobody Explodes
- I Expect You To Die
- The Room VR: A Dark Matter

Каждая из этих игр является ярким представителем жанра головоломка, не все из них были изначально задуманы для технологии VR, но в дальнейшем их адаптировали к данной технологии, добавив в них некоторые особенности взаимодействия, характерные для виртуальной реальности.

The Room VR: A Dark Matter

Серия игр The Room заслуживает внимания, а также заслужила репутации для игроков, её атмосфера крайне удивительна и иммерсивна. A Dark Matter рассказывает детективную историю со сверхъестественными мотивами. В ней игроку предстоит управлять детективом полиции, расследующего исчезновение египтолога в Лондоне 1908 года.

I Expect You To Die

Данная игра представлена в юмористическом характере, она полна увлекательных задач, игра состоит из проб и ошибок игрока, в ходе которой необходимо взаимодействовать со всеми объектами и выяснять их связь относительно друг друга.

Рассмотрев данные компьютерные игры, были взяты ориентиры по созданию головоломок для создания собственного продукта.

В процессе разработки можно выделить следующие этапы.

I. Установка.
II. Написание сценария.
III. Создание проекта и импорт плагинов.
IV. Разработка уровней.
V. Добавление логики на уровни.
VI. Исправление ошибок.
VII. Тестирование игры.
VIII. Оптимизация производительности
IX. Создание руководства пользователя

X. Документирование процесса разработки

Каждый этап играет свою роль в процессе создания игры, и они тесно связанные между собой. Рассмотрим каждый из них более подробно.

I. Установка.

Для того чтобы приступить к разработке продукта, нам необходимо зарегистрироваться на сайте, а также выбрать вариант лицензии, подходящий для нас, а именно «Personal» и установить среду разработки Unity, для установки среды нам понадобится приложение «Unity Hub», в котором необходимо авторизоваться. Далее необходимо выбрать подходящую для нас версию Unity, а именно 2023.2.16f1 и установить её. Также в процессе установки среды разработки нам понадобится редактор кода, Unity предлагает установить Microsoft Visual Studio, выберем его для установки.

После установки Unity, пройдет запуск установщика Visual Studio в котором также необходимо выбрать подходящую версию, для нас подойдет версия Microsoft Visual Studio 2022, также для корректного использования данного IDE с Unity, нам необходима библиотека, которая называется «Разработка игр с помощью Unity». После установки среды разработки и IDE, можно приступить к следующему этапу разработки.

II. Написание сценария.

Концептуально для головоломок сюжет является дополнением, приятным бонусом и мотивацией пройти игру, поэтому для игры был написан небольшой сюжет, который может заинтересовать игрока к прохождению.

Рабочее название игры: «Digital Detective»

Вступление: игрок оказывается в виртуальном мире, который представляет собой несвязанные между собой миры, «киберпространство». Главный герой – частный детектив, нанятый для расследования череды странных событий, казалось бы, несвязанных между собой, происходящих в «киберпространстве».

Задачи игрока: игрок расследует различные виртуальные локации, решая головоломки для раскрытия тайны и угрозы виртуальному миру. Он

должен пройти все уровни, каждый из которых представляет собой новую сцену с уникальными головоломками. Прохождение игры предполагает прохождение нескольких уровней, а после прохождений головоломок на каждом, предлагается перейти на следующий. После прохождения всех уровней игра вернется на уровень «Главное меню», из которого можно начать проходить игру с начала или выйти из неё.

Уровень 1: «Серверная»

Описание: игрок оказывается внутри небольшой серверной, где стены покрыты серверами и звучит гул машин. Необходимо найти способ активировать главный сервер, чтобы перейти к следующему уровню.

Головоломки: игроку необходимо перезапустить сервер, используя мастер-код, следуя логике.

Уровень 2: «Квартира»

Описание: игрок находится в «реальной» квартире, он должен исследовать комнаты, найти подсказки и решить загадки, чтобы раскрыть новую тайну.

Головоломки: логические задания, поиск скрытых предметов, декодирование сообщений.

Финал: Игрок раскрывает тайну, побеждает зло виртуального мира, вирус, восстанавливает мирный порядок. Возвращается в реальный мир с выполненной миссией и приличным гонораром в кармане, новой настороженностью по поводу «киберпространства». Но конец ли это?

III. Создание проекта и импорт плагинов.

Игровое приложение сделано в формате 3D (Рис. 7) с использованием VR технологий. Создание 3D-игры является сложным процессом, требующих больших затрат и времени, в отличии от 2D. Для создания 3D-игры в основном работают крупные команды, но и для небольших студий и независимых разработчиков это тоже является выполнимой задачей. В следствии нехватки ресурсов и времени, в приложении будут использоваться заготовки, представленные в магазине Unity.

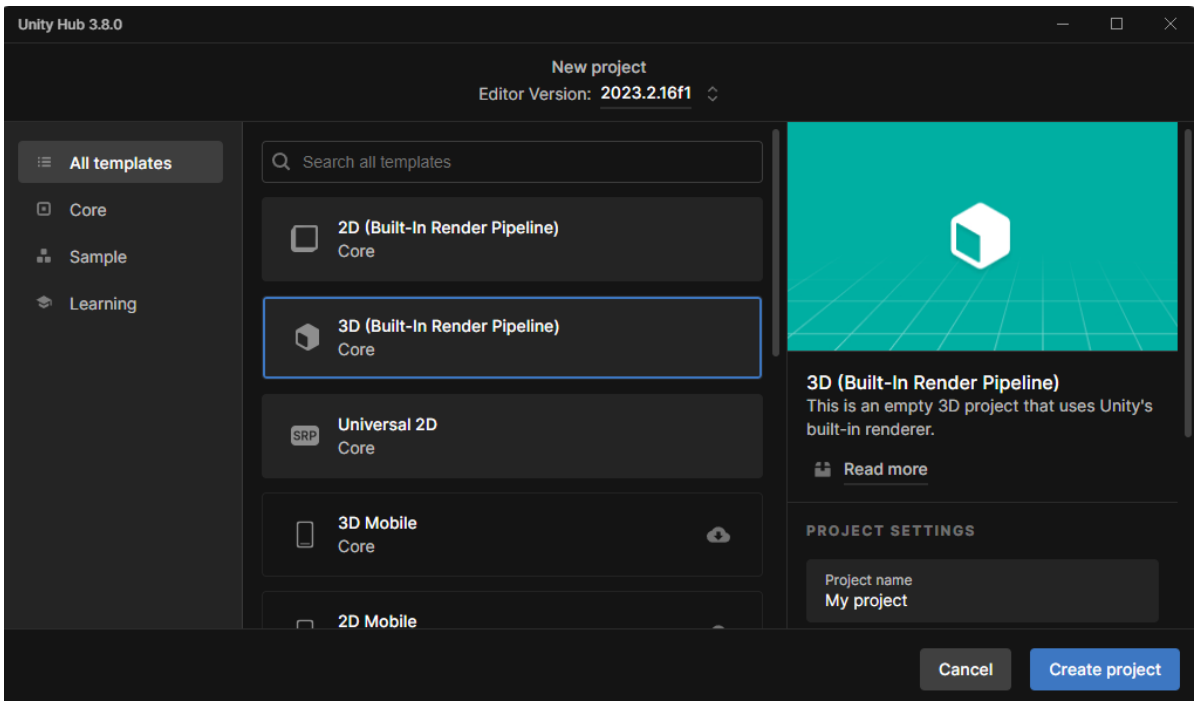

Рис. 7 Создание нового проекта в Unity формата 3D

При создании 3D-игр, в особенности VR-игр, необходимо множество ресурсов компьютера для обработки огромного количества информации, именно поэтому для них используется более дорогие системы, современные видеокарты и процессоры, а также не меньше 8 ГБ оперативной памяти. В основном использована анимация рук, а все объекты используют физическую модель и не имеют анимаций. Разрабатывая любое игровое приложение также следует помнить про оптимизацию, ведь неизвестно на каком ПК будет запущено приложение, ведь некоторые системы гораздо слабее на фоне других, так, например, в техническом задании выпускной квалификационной работы как минимальное требование была выставлена видеокарта Nvidia GTX 1070, на ней приложение будет работать, но не всегда стабильно.

После создания проекта формата 3D, необходимо импортировать необходимые плагины, среди которых есть базовые для VR, а также заготовки из магазина Unity.

Используемые плагины:

- SteamVR.
- Apartment Kit.
- LowPoly Server Room Props.

- VR.
- Engineering.
- OpenVR XR.

Данные плагины позволяют в полной мере реализовать необходимое в игровом приложении. Плагин «VR», «SteamVR» и «OpenVR XR» добавляют необходимый инструментарий для работы приложения в режиме VR, SteamVR добавляет базовые заготовки, например готовую систему камеры для VR и модели рук для игрока.

В плагинах «Apartment Kit» и «LowPoly Server Room Props» представлены 3D-модели для использования в проекте, они не являются готовыми и с ними нужно будет работать для того, чтобы внедрить их в VR-сцену игрового приложения, они позволят избежать лишних трат времени на создания 3D-моделей с нуля.

Все эти плагины являются бесплатными, и стандартная лицензия Unity позволяет их использовать в коммерческих проектах. При необходимости можно добавить дополнительные плагины, для расширения возможностей и так богатой среди разработки Unity.

IV. Разработка уровней.

В игровом приложении будет представлено три уровня, первым из которых является главное меню, оно не играет роли в сюжете и не несет никакой смысловой нагрузки, его главной целью является или начать игру, или выйти.

Главное меню представляет собой небольшую прямоугольную комнату (Рис. 8), в которой добавлены некоторые предметы для наполнения комнаты. Для создания данного уровня необходимо было создать шесть объектов которые использовались для стен, пола и потолка соответственно.

Рис. 8. Сцена главного меню

Были добавлены источники света, стенд в качестве пользовательского интерфейса, в котором располагаются две главные кнопки, «Играть» и «Выход».

Также в комнате присутствуют некоторые 3D-модели, с которыми можно взаимодействовать, один из таких — это радиоуправляемая машина вместе с пультом. Помимо машины, есть 2 стола, один деревянный, на нем расположен пульт управления, на другом ноутбук и подставка с карандашами. Также в главное меню добавлена музыка с открытой лицензией, которую можно использовать для коммерческого использования. На данном уровне не представлено головоломок. По итогам сборки сцены главного меню блок уровня (Рис. 9.) в среде разработки выглядит не слишком запутанно.

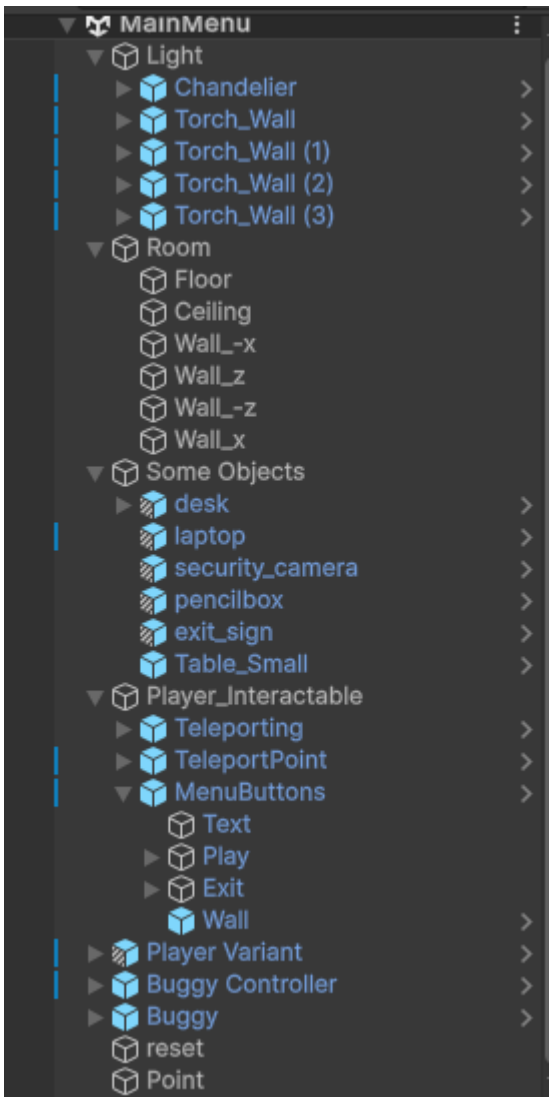

Рис. 9. Блок главного меню

Для перехода на следующий уровень необходимо нажать кнопку «Играть» на стенде в сцене главного меню.

Первый уровень представляет собой небольшую комнату, а именно «серверную» в которой расположен компьютерный стол, сервера, несколько дверей и панелей. Цель комнаты, выбраться из неё, для этого можно взаимодействовать с объектами, представленными на уровне. Сложность прогресса в уровне не является проблемой для людей, владеющих простой логикой и базовым владением иностранного языка, а именно английского.

Рис. 10. Сцена первого уровня игрового VR-приложения «Серверная»

В начале для создания первого уровня, был использован плагин под названием «LowPoly Server Room», о котором было сказано ранее, в составе которых были необходимые объекты для реализации уровня, в дальнейшем при помощи этих объектов (Рис. 11.) была собрана игровая сцена, использованная для первого уровня (Рис. 10.).

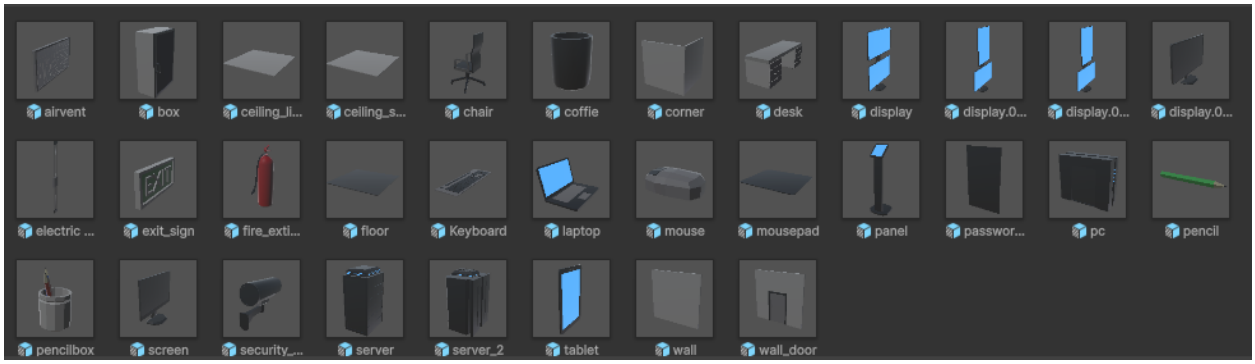

Рис. 11 Объекты из плагина «LowPoly Server Room»

Среди головоломок на первом уровне разработаны планшеты, взаимодействующие с объектами, панель для ввода кода, а также сервер, с которым могут взаимодействовать объекты. После решения всех головоломок активируется панель, при взаимодействии с которой игрок перейдет на следующий уровень, «Квартиру».

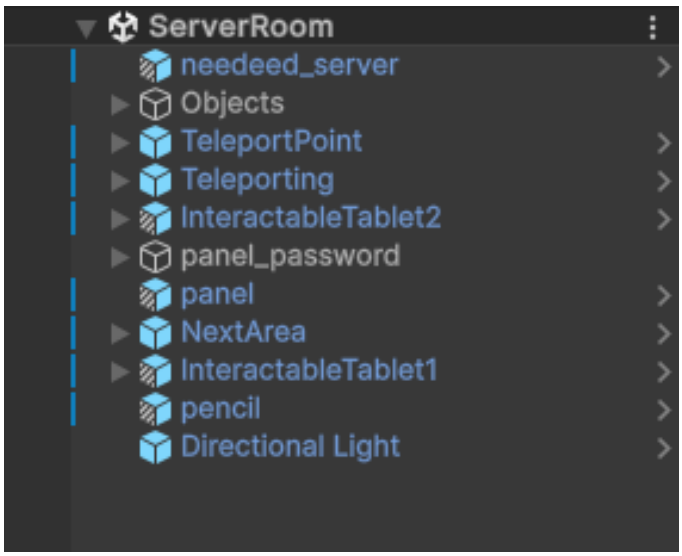

Рис. 12 Блок сцены первого уровня

Второй уровень создавался с целью погружения в находящуюся атмосферу, представляет собой обустроенную квартиру нашего времени, цель остается неизменной, выбраться из комнаты, в нашем случае квартиры и решить необходимые для этого головоломки. Выбранную локацию можно считать не особо крупным пространством, полностью наполненным функциональными объектами, с которыми можно взаимодействовать, но создается помеха, которая может запутать игрока в поисках и решениях головоломок.

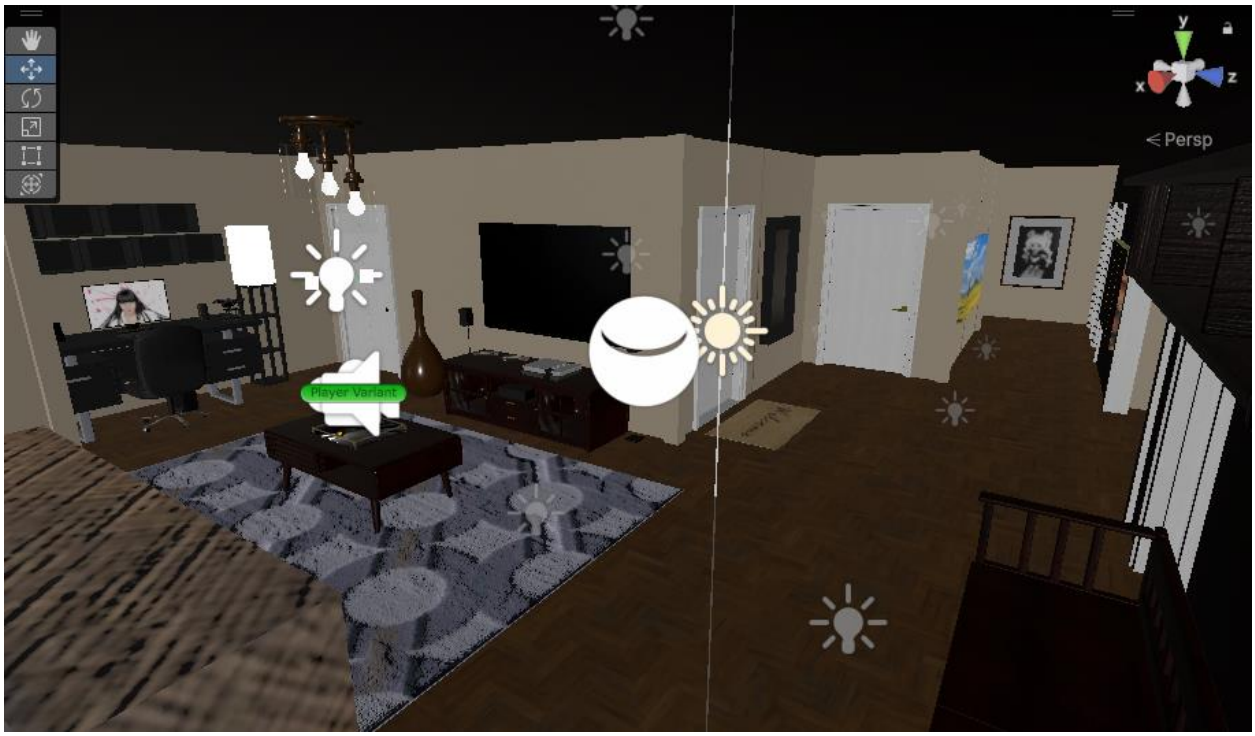

Рис. 13 Сцена второго уровня игрового VR-приложения «Квартира»

Для создания данной сцены были использованы объекты из плагина «Apartment kit» в которой присутствуют необходимые 3D-модели для создания квартир. В рамках которой была сделана одна квартира, но также реализовано здания, в котором можно обустроить четыре квартиры, но в сценарии это не посчиталось необходимым. Объекты из плагина можно считать неким конструктором, 3D модели не являются единым целым в следствии чего были созданы анимации для открытия дверей и выдвижных ящиков. Они двигают в пространстве в определенном диапазоне части объектов если это касается выдвижных ящиков у комода, тумбочек, холодильника и прочей домашней мебели. Двери же двигались целиком и в дальнейшем к ним будет применен скрипт для обработки логики взаимодействия.

V. Добавление логики на уровни.

Игровой геймплей заключается в управлении персонажем, в нашем случае это руки и взаимодействие с окружающим миром. В VR-приложении особенность взаимодействия состоит в том, что нужно физически «взять» объект и осмотреть его, используя VR-манипуляторы. UI как на обычных играх и мобильных, можно реализовать, но в случае данного приложения, в нем нет необходимости, нечего терять, например «здоровье» и «патроны». Или нет необходимости в направляющих маркерах, ведь уровни не являются огромными и в них трудно потеряться. Но маркеры могут пригодиться при создании головоломок в роли подсказок для игрока, которые могут быть вызваны определенной кнопкой.

При создании проекта были импортированы плагины: SteamVR, VR, OpenVR XR. Все они будут нужны для создания «игрока». В плагине SteamVR есть примеры скриптов, которые можно реализовать в полной мере, а некоторые являются лишь примером для создания собственных. Одними из таких скриптов, которые будут использоваться в игровом приложении это скрипт для камеры и рук. Также помимо скриптов есть модели для рук, которые также будут использованы. То есть заготовок для запуска игры в форма-

те VR уже предоставлен, необходимо написать скрипты для передвижения, взаимодействия и перехода уровня.

В плагине нет скрипта на передвижения, разработчиками данного плагина задумано использование телепортации и порталов, так как данный способ перемещений предотвращает морскую болезнь, вызванную слабым вестибулярным аппаратом, а также несоответствие передвижения в игре и реальности, потому что, шагая в игре, вы не шагаете на самом деле, это и вызывает диссонанс и потерю в пространстве. Но уверенным пользователям VR-гарнитур такой способ не очень подходит, ведь в нем теряется погружение в игру, нарушается атмосфера и игра будет ощущаться менее реально, поэтому был реализован скрипт для плавного передвижения в пространстве.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Valve.VR;
public class Locomotion : MonoBehaviour
{
    [Header("General Data")]
    public float minHeight = 1.0f;
    public float maxHeight = 2.0f;
    public float gravityStrength = 350.0f;
    public CharacterController characterController = null;
    public Transform cameraRig = null;
    public Transform head = null;
    [Header("Movement")]
    public float maxSpeed = 2.0f;
    public float joystickSensitivity = 0.1f;
    public SteamVR_Action_Vector2 joystickDir = null;
    [HideInInspector]
    public float currentSpeed = 0.0f;
    void Start()
    {
        characterController = GetComponent<CharacterController>();
    }
    void Update()
    {
        UpdateHeight();
        UpdateMovement();

    }

    void UpdateHeight()
    {
        float headHeight = Mathf.Clamp(head.localPosition.y, minHeight, maxHeight);
        characterController.height = headHeight;
        Vector3 newCenter = head.localPosition;
        newCenter.y = characterController.height / 2;
        newCenter.y += characterController.skinWidth;
        characterController.center = newCenter;
    }
    void UpdateMovement()
    {
        Quaternion movementDir = CalculateMovementDir();
        if (joystickDir.GetAxis(SteamVR_Input_Sources.LeftHand).magnitude == 0)
            currentSpeed = 0.0f;
        currentSpeed += joystickDir.GetAxis(SteamVR_Input_Sources.LeftHand).magnitude * joystickSensitivity;
        currentSpeed = Mathf.Clamp(currentSpeed, -maxSpeed, maxSpeed);
        Vector3 movementVec = movementDir * (currentSpeed * Vector3.forward);
        movementVec.y -= gravityStrength * Time.deltaTime;
        characterController.Move(movementVec * Time.deltaTime);
    }

    Quaternion CalculateMovementDir()
    {
        float rotation = Mathf.Atan2(joystickDir.GetAxis(SteamVR_Input_Sources.LeftHand).x,
joystickDir.GetAxis(SteamVR_Input_Sources.LeftHand).y);
        rotation *= Mathf.Rad2Deg;
        Vector3 moveDirEuler = new Vector3(0.0f, head.eulerAngles.y + rotation, 0.0f);
        return (Quaternion.Euler(moveDirEuler));
              }
```

Листинг. 1. Скрипт передвижения

В данном скрипте реализовано передвижения при помощи устройства управления игрового контроллера в виде рычага, или же стика. Помимо Камеры и двух рук у игрока есть и другие компоненты, например коллайдеры для физического взаимодействия у рук, тела и головы, потому что ходить и перемещаться сквозь стены нежелательно. Помимо передвижения, в скрипте реализована регулировка высоты персонажа, она зависит от высоты камеры, а камера фактически является VR-шлемом и полностью перемещается по всем осям. Но при этом есть ограничение по минимальной и максимальной высоте. Помимо необходимого, то есть VR-гарнитуры, к игроку также был добавлен компонент «Audio Listener», который позволяет выводить звук,

прикрепленный к объектам, потому что в игру будет использована, как и музыка, так и звуки. По итогам создания модели игрока его структура (Рис. 14) довольна громоздка, но отвечает всем необходимым условиям для создания игры.

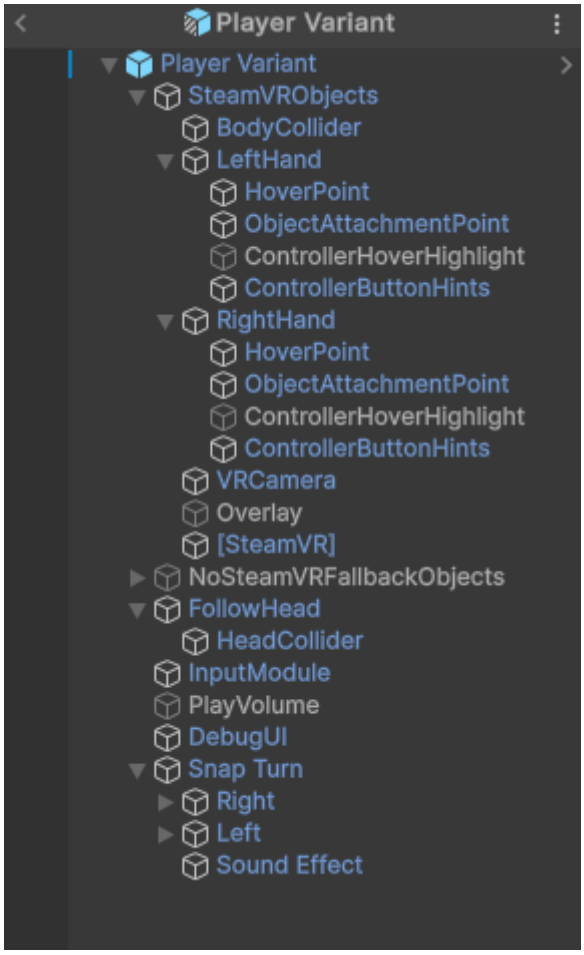

Рис. 14. Блок игрока

При написании сценария появилась идея реализовать озвучку и музыку при помощи нейросетей. Так для озвучки главного героя и других персонажей, была использована нейросеть от «SteosVoice» (https://cybervoice.io/en/). Она является бесплатной, но с определенными условиями, так первое условие, это то, что большинство голосов могут быть использованы только для некоммерческого проекта, второе условие, в день выдается определенное количество символов которая может озвучить нейросеть, данное ограничение не мешает сделать все в течении отведенного времени на подготовку и сдачу выпускной квалификационной работы. Так для озвучки главного героя был выбран голос Дмитрия Зарубина, также есть голоса дикторов и актеров дуб-

ляжа из игр и кино, но все они звучали как нейросеть, а выбранный голос наиболее реалистичен. Для создания музыки в игре была использована нейросеть «Aiva», она также является бесплатной, но с определенным условием, скачивание не более трех мелодий в месяц. При помощи данной нейросети была создана музыка в формате «Техно» с длительностью 3 минуты.

Для создания любого взаимодействия к 3D-модели надо прикрепить компонент Unity под названием Collider, они бывают разных типов, у большинства моделей которые являются сложными геометрическими фигурами используется «Mesh Collider», что является коллайдером для совокупности вершин и полигонов, Так при создании моделей необходимо учесть, создается модель для игр или нет, в зависимости от этого надо ограничивать количество полигонов, ведь каждый полигон обрабатывается физическим движком Unity, что вызывает нагрузку на компьютер, так в одной сцене не рекомендуется превышать рекомендуемый порог сто тысяч вершин, поэтому придуманы инструменты оптимизации, которые уменьшают количество полигонов в зависимости от дальности и видимости объектов, что дает возможность большее количество объектов на сцене.

Ко всем моделям, с которыми может взаимодействовать игрок, а это любая поверхность, были добавлены компоненты Collider, а тип компонента был определен моделью, «Round», «Box», «Plane», «Mesh». Что означает круглый, квадратный, плоский, или сложный компонент. Также среди общих объектов на сценах везде был использован свет, где-то направленный, где-то ненаправленный.

Для всех моделей был созданы скрипты взаимодействия и броска с учетом физики, веса объекта и придания ускорения от руки. Также в скрипте броска была реализована система, которая позволяет добавить события при поднятии объекта, отпускании и в процессе удерживания его в руке. Таким образом можно добавить любые звуки и дополнительные скрипты для взаимодействия с любой логикой. Базовый скрипт взаимодействия позволяет

включить подсветку при наведении на объект и загрузку набора управления, эта система пригодилась в создании объектов, которые становились активными при поднятии зависящего от них объекта.

В создании главного меню, необходимо добавить логику для машины с пультом и стенда с пользовательским интерфейсом, а также для объектов, с которыми возможны любые взаимодействия. Так для машины и пульта были написаны два скрипта «BuggyController» и «BuggyBuggy», в котором один используется для получения входных данных с манипулятора VR-гарнитуры, а второй получает эти входные данные и при помощи формул передвижения двигает саму машинку. Скрипты для машинки состоят из 350 строк кода. Для создания пользовательского интерфейса были написаны два скрипта, один (Листинг. 2.) из которых необходим для взаимодействия с кнопкой при помощи VR-гарнитуры, второй скрипт был написан для старта игры и соответственно выхода из неё.

```
public class MenuButtons : MonoBehaviour
{
    public void LoadLevel()
    {        SteamVR_LoadLevel.Begin("ServerRoom");    }
    public void Exit()
    {        Application.Quit();    }}
```

Листинг. 2. Кода для перехода на уровень или выхода из игры.

```
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.UI;
using System;
namespace Valve.VR.InteractionSystem
{

	[RequireComponent( typeof( Interactable ) )]
	public class UIElement : MonoBehaviour
	{
		public CustomEvents.UnityEventHand onHandClick;

	protected Hand currentHand;
		protected virtual void Awake()
		{
			Button button = GetComponent<Button>();
			if ( button )
			{
				button.onClick.AddListener( OnButtonClick );
			}
		}
		protected virtual void OnHandHoverBegin( Hand hand )
		{
			currentHand = hand;
			InputModule.instance.HoverBegin( gameObject );
			ControllerButtonHints.ShowButtonHint( hand, hand.uiInteractAction);
		}
	protected virtual void OnHandHoverEnd( Hand hand )
		{
			InputModule.instance.HoverEnd( gameObject );
			ControllerButtonHints.HideButtonHint( hand, hand.uiInteractAction);
			currentHand = null;
		}
	protected virtual void HandHoverUpdate( Hand hand )
		{
			if ( hand.uiInteractAction != null &&
hand.uiInteractAction.GetStateDown(hand.handType) )
			{
				InputModule.instance.Submit( gameObject );
				ControllerButtonHints.HideButtonHint( hand, hand.uiInteractAction);
			}
		}
	protected virtual void OnButtonClick()
		{
			onHandClick.Invoke( currentHand );
		}
	}
```

Листинг. 3. Код для взаимодействия с кнопками пользовательского интерфейса.

Логика была применена ко всем объектам и можно переходить к объектам следующего игрового уровня, а именно «Серверная».

В создании первого уровня логику необходимо применить к объектам, которые связаны с головоломками или с переходом на следующий уровень. На первом уровне всего несколько объектов необходимых для решения головоломок, а именно, карандаш, 2 планшета, сервер, панель для ввода кода и зона телепортации на следующий уровень.

Для данных объектов были добавлены необходимые теги, например, объект «Сервер», к нему могут применяться 2 тега, «выкл» и «вкл», компоненты «mesh collider», соответствующие скрипты для их активации, также был применен звук и добавлен компонент «Audio Source» для объекта «Сервер», который применен в головоломке используя 3D-эффект для звука.

На реализацию всей логики уровня, были разработано пять скриптов. Первый скрипт, нужен для первой головоломки, взаимодействия планшета и карандаша, добавленные к ним компоненты коллайдера, обладают свойством триггера, поэтому был добавлен компонент «Box Collider» с добавлением триггера (Рис. 15), что убирает его коллизию с другими объектами, так на планшете есть 2 коллайдера, один триггер, а другой реагирует на коллизию предметов.

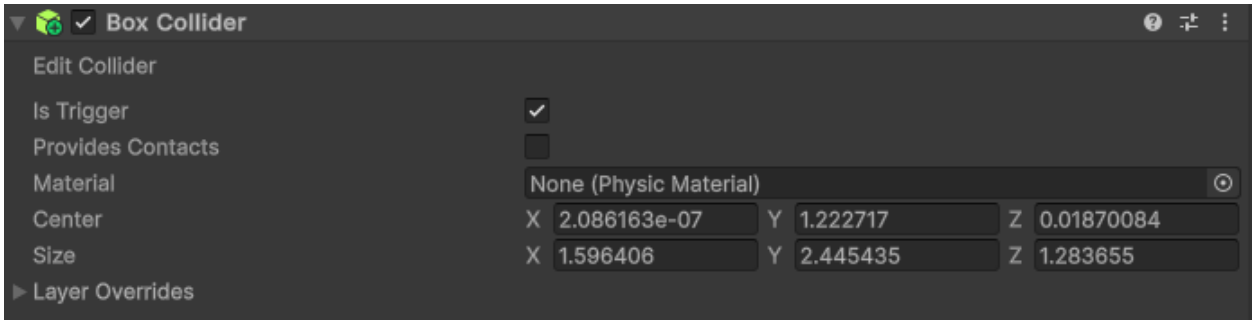

Рис. 15. Компонент «Box Collider»

Дополнительно к планшету был добавлен компонент «TextMeshPro» при помощи которого можно обратиться к пользователю и указать на правильность действий. Поэтому был написан скрипт, в котором происходит взаимодействие с объектами карандаш и планшет, а также с изменение компонента «TextMeshPro».

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

public class tablet_pen : MonoBehaviour
{
    public GameObject triggerObject; // Объект, тег которого нужно изменить
    public GameObject targetObject; // Объект, тег которого нужно изменить
    public string newTag; // Новый тег
    public TextMeshPro distanceTextUI; // Текстовое поле для вывода
    public tablet_behaivor tablet_behaivor;
    public void Start()
    {
        if (targetObject != null)
        {           targetObject.tag = "offserver";        }
        else
        {           Debug.LogError("Целевой объект не установлен!");        }
    }
    public void ChangeTag()
    {
        if (targetObject != null)
        {           targetObject.tag = newTag;        }
        else
        {           Debug.LogError("Целевой объект не установлен!");        }
    }

    public void OnTriggerEnter(Collider other)
    {
        if (other.gameObject == triggerObject)
        {
            distanceTextUI.text = "You did it! YAY!!!";
            ChangeTag();
            tablet_behaivor.FindObject();
        }
    }
}
```

Листинг. 4. Код для головоломки №1

Второй скрипт был разработан для объекта «Сервер», его задача была изменить тональность и направление звука, для того чтобы игрок мог найти нужный сервер. Так для реализации данного скрипта необходим компонент «Audio Source», к нему был добавлен звук серверной, распространяемый по бесплатной лицензии, и к нему применялись необходимые параметры (Листинг. 5Рис. 16). Так среди параметров есть исполняемый файл, выход для звука и различные эффекты, среди которых есть параметр зацикливания и громкости. Помимо компонента звука также был добавлен дополнительный триггер коллайдер для взаимодействия с планшетом.

```
    void Update()
    {
        // Проверка тега и управление pitch и spatial blend
        if (CompareTag(correctTag))
        {   targetPitch = maxPitch; // Устанавливаем максимальный pitch
            targetSpatialBlend = maxSpatialBlend; // Устанавливаем максимальный spatial blend        }
        else
        {   targetPitch = minPitch; // Устанавливаем минимальный pitch
            targetSpatialBlend = minSpatialBlend; // Устанавливаем минимальный spatial blend        }

        // Постепенно изменяем громкость, pitch и spatial blend звука к целевым значениям
        audioSource.pitch = Mathf.MoveTowards(audioSource.pitch, targetPitch, fadeSpeed * Time.deltaTime);
        audioSource.spatialBlend = Mathf.MoveTowards(audioSource.spatialBlend, targetSpatialBlend, fadeSpeed *
Time.deltaTime);
    }
```

Листинг. 5. Фрагмент скрипта для управления сервером.

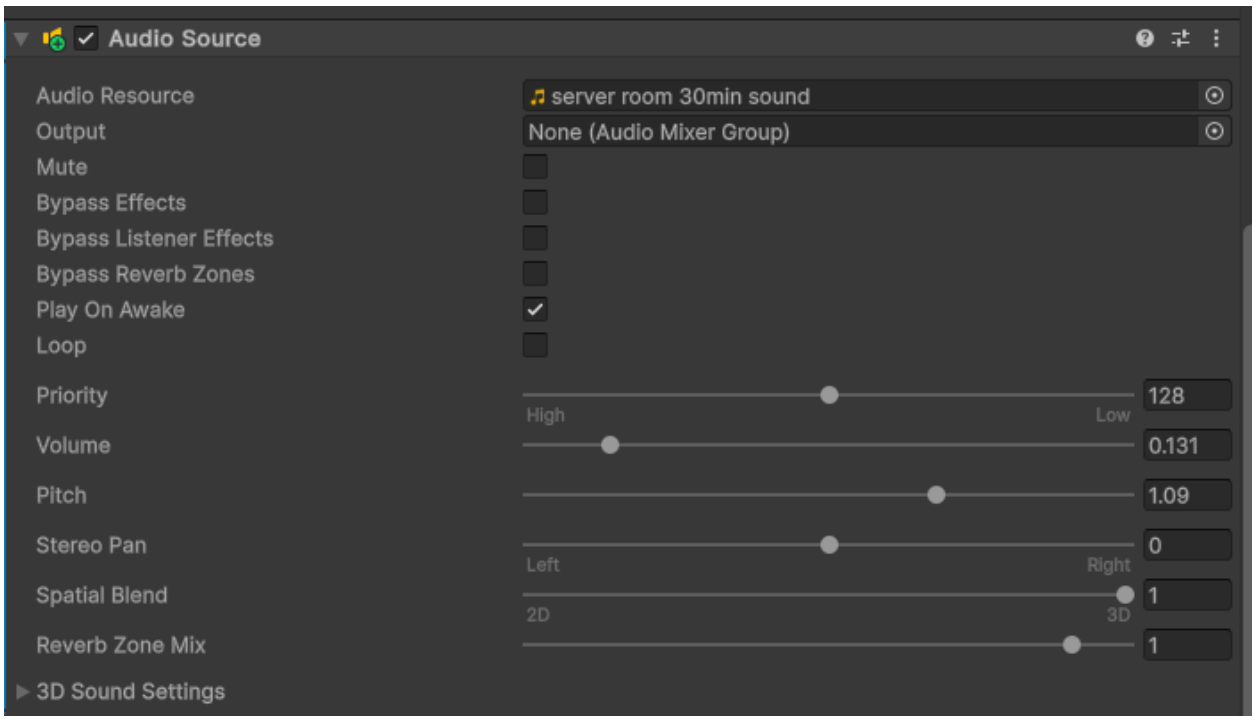

Рис. 16. Параметры компонента «Audio Source»

После того, как произошли события первых двух скриптов, можно рассматривать дальнейшие скрипты взаимодействий и головоломок, следующим скриптом является скрипт, который сравнивает какие коллайдеры взаимодействуют с планшетом, и если коллайдер другого объекта является объект с тегом «server», то на планшете изменялся текст, необходимый для дальнейшего. Также в скрипте есть метод, который вызывается другим скриптом (Листинг. 4), в котором производится поиска объекта с необходимым тегом. Текст изменяется, как и прежде компонентом «TextMesh Pro».

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

public class tablet_behaivor : MonoBehaviour
{
    public TextMeshPro distanceTextUI;
    private GameObject serverObject;
    public void FindObject()
    {
        serverObject = GameObject.FindGameObjectWithTag("server");
        if (serverObject == null)
        {
            Debug.LogError("'server'!");
        }
    }
    private void OnTriggerEnter(Collider other)
    {
        if (other.gameObject == serverObject)
        {           distanceTextUI.text = "356894";       }    }

}
```

Листинг. 6. Код планшета с паролем.

После получения правильного пароля на экране планшета, необходимо использовать панель. Панель для ввода кода была реализована при помощи нажимной кнопки, которая была размножено на необходимое количество раз, написан необходимый скрипт для управления кнопкой (Листинг. 7). Помимо скрипта нажимной кнопки, написан небольшой скрипт для управления панелью. К кнопка добавлен компонент «TextMesh Pro» для визуального обозначения действий кнопок, у панели добавлен компонент «Audio Source», при вводе правильного пароля будет проигрывается звук, предполагающий осознание правильности ввода и в дальнейшем панель для перехода на следующий уровень будет активирована.

```
using UnityEngine;
using System.Collections;
using TMPro;
namespace Valve.VR.InteractionSystem.Sample
{
    public class ButtonExample : MonoBehaviour
    {
        public HoverButton Button_1;
        public TextMeshPro text;
        public TeleportPoint TeleportPoint;
        private void Start()
        {
            Button.onButtonDown.AddListener(OnButtonDown);
            TeleportPoint.SetLocked(true);
        }

        private void OnButtonDown(Hand hand)
        {
            Print("1");
        }
        private void OnButtonDown(Hand hand)
        {
            EnterText();
        }
        private void OnButtonDown(Hand hand)
        {
            Clear();
        }

        private void Print(string text_)
        {
            text.text += text_;
        }
        private void EnterText()
        {
            if (text.text == "356894")
                //TeleportPoint.GetComponent<TeleportPoint>().SetLocked(false);
            TeleportPoint.SetLocked(false);
            else
                text.text = string.Empty;
        }
        private void Clear()
        {
            text.text = string.Empty;
        }
    }
}
```

Листинг. 7. Код для панели ввода пароля

```
using UnityEngine;
using System.Collections;
using UnityEngine.Events;

namespace Valve.VR.InteractionSystem
{
    [RequireComponent(typeof(Interactable))]
    public class HoverButton : MonoBehaviour
    {
        public Transform movingPart;
        public Vector3 localMoveDistance = new Vector3(0, -0.1f, 0);
        [Range(0, 1)]
        public float engageAtPercent = 0.95f;
        [Range(0, 1)]
        public float disengageAtPercent = 0.9f;
        public HandEvent onButtonDown;
        public HandEvent onButtonUp;
        public HandEvent onButtonIsPressed;
        public bool engaged = false;
        public bool buttonDown = false;
        public bool buttonUp = false;
        private Vector3 startPosition;
        private Vector3 endPosition;
        private Vector3 handEnteredPosition;
        private bool hovering;
        private Hand lastHoveredHand;
        private void Start()
        {
            if (movingPart == null && this.transform.childCount > 0)
                movingPart = this.transform.GetChild(0)
            startPosition = movingPart.localPosition;
            endPosition = startPosition + localMoveDistance;
            handEnteredPosition = endPosition;
        }

        private void HandHoverUpdate(Hand hand)
        {
            hovering = true;
            lastHoveredHand = hand;
            bool wasEngaged = engaged;
            float currentDistance = Vector3.Distance(movingPart.parent.InverseTransformPoint(hand.transform.position), endPosition);
            float enteredDistance = Vector3.Distance(handEnteredPosition, endPosition);

            if (currentDistance > enteredDistance)
            {               enteredDistance = currentDistance;
                handEnteredPosition = movingPart.parent.InverseTransformPoint(hand.transform.position);           }
            float distanceDifference = enteredDistance - currentDistance;
            float lerp = Mathf.InverseLerp(0, localMoveDistance.magnitude, distanceDifference);
            if (lerp > engageAtPercent)
                engaged = true;
            else if (lerp < disengageAtPercent)
                engaged = false;
            movingPart.localPosition = Vector3.Lerp(startPosition, endPosition, lerp);
            InvokeEvents(wasEngaged, engaged);
        }

        private void LateUpdate()
        {           if (hovering == false)
            {
                movingPart.localPosition = startPosition;
                handEnteredPosition = endPosition;
                InvokeEvents(engaged, false);
                engaged = false;
            }
            hovering = false;
        }

        private void InvokeEvents(bool wasEngaged, bool isEngaged)
        {
            buttonDown = wasEngaged == false && isEngaged == true;
            buttonUp = wasEngaged == true && isEngaged == false;
            if (buttonDown && onButtonDown != null)
                onButtonDown.Invoke(lastHoveredHand);
            if (buttonUp && onButtonUp != null)
                onButtonUp.Invoke(lastHoveredHand);
            if (isEngaged && onButtonIsPressed != null)
                onButtonIsPressed.Invoke(lastHoveredHand);
        }
    }
}
```

Листинг. 8. Код нажимной кнопки.

Для создания логики второго уровня «Квартира» необходимо сделать анимации дверей и всех открывающихся поверхностей, также необходимо создать головоломки для уровня.

Реализованы анимации открытия и закрытия дверей, которые активируются при взаимодействии игрока с коллайдером двери. Для некоторых дверей необходим ключ, который нужно поднести к ручке для активации взаимодействия с дверью. Для каждой двери есть несколько состояний: закрыто, открыто, закрыто ключом. Анимации выдвижных ящиков в квартире также реагируют на действия игрока, их можно закрыть и открыть, найти необходимые предметы, также как и с дверьми есть система ключ-замок и несколько состояний.

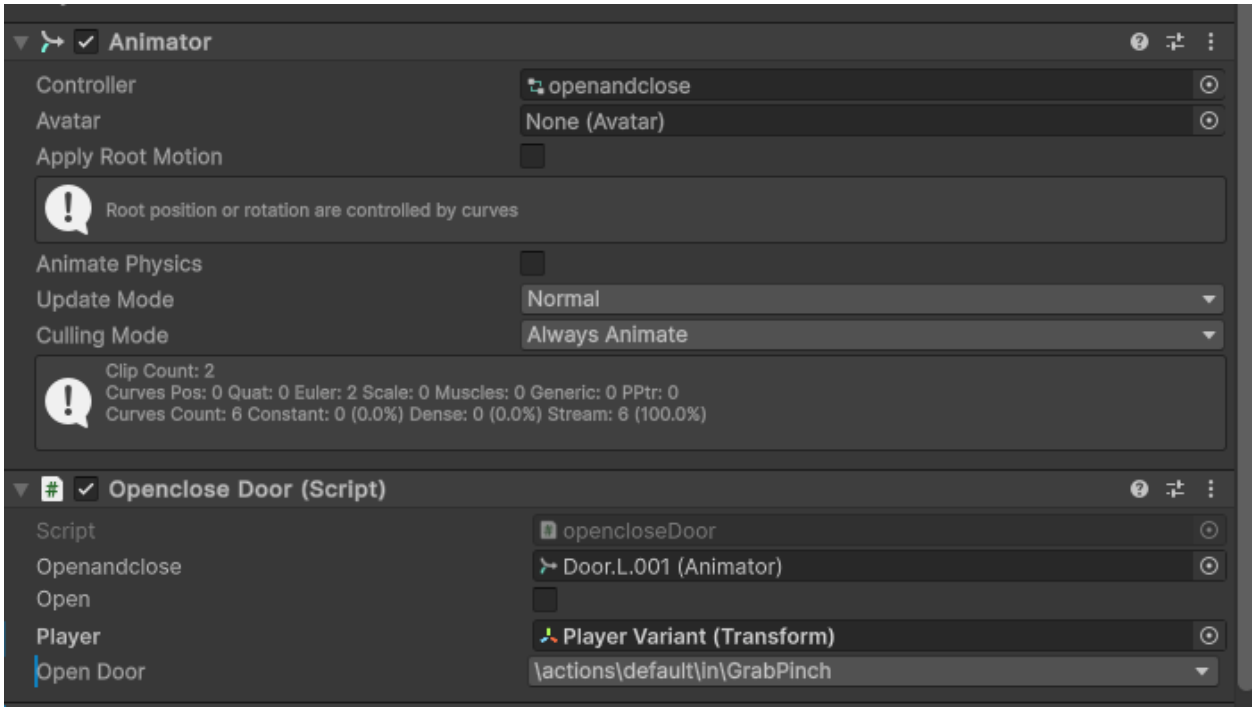

Рис. 17. Аниматор и скрипт для взаимодействия дверей.

Одной из головоломок уровня являются «Пятнашки», игровое поле состоит из пятнадцати плиток, которые необходимо разместить в правильном порядке, это головоломка состоит из двух скриптов:

- Скрипт перемещения плитки: отвечает за логику плитки, проверки свободной по соседству клетки и последующего перемещения на неё.

```
public class FifteenPuzzleTile : MonoBehaviour
{
    private Vector2Int position; // Позиция плитки на игровом поле
    private FifteenPuzzleManager puzzleManager; // Ссылка на менеджер игры
    public SteamVR_Action_Boolean OpenDoor = null;
    private Hand.AttachmentFlags attachmentFlags = Hand.defaultAttachmentFlags & (~Hand.AttachmentFlags.SnapOnAttach)
& (~Hand.AttachmentFlags.DetachOthers) & (~Hand.AttachmentFlags.VelocityMovement);
    void Start()
    {
        puzzleManager = FindObjectOfType<FifteenPuzzleManager>(); // Находим менеджер игры в сцене
    }

    public void SetPosition(Vector2Int newPosition)
    {
        position = newPosition;
    }
    public Vector2Int GetPosition()
    {
        return position;
    }
    private void HandHoverUpdate(Hand hand)
    {
        Debug.Log("Значения у тайла = "+puzzleManager.emptyPosition);
        if (puzzleManager.isSolved) return; // Если пятнашки уже решены, игровые действия недоступны
        if (OpenDoor.stateUp)
        {
            // Получаем текущую позицию пустой клетки
            Vector2Int emptyPos = puzzleManager.emptyPosition;
            Debug.Log(emptyPos);
            // Проверяем, можно ли переместить плитку на пустое место
            if (Mathf.Abs(emptyPos.x - position.x) + Mathf.Abs(emptyPos.y - position.y) == 1)
            {
                // Если да, то меняем позиции плитки и пустой клетки
                puzzleManager.emptyPosition = position;
                position = emptyPos;

                // Перемещаем плитку на новую позицию
                transform.localPosition = new Vector3(position.x * puzzleManager.spacing, position.y * -
puzzleManager.spacing, 0f);
            }        }    }}
```

Листинг. 9. Скрипт логики клетки в пятнашках.

- Скрипт размещения, перемешивания и проверки решения: отвечает за создания и логику головоломки, генерирует в определенных пределах пятнадцать плиток, изменяет их компонент «TextMesh Pro» и перемешивает их в случайном порядке, оставляя последнюю клетку пустой. Если все плитки будут расположены в правильном порядке, скрипт активирует соответствующее событие, например открытие ящика или сейфа.

```
public class FifteenPuzzleManager : MonoBehaviour
{
   public GameObject tilePrefab; // Префаб плитки
   public Transform puzzleParent; // Родительский объект для плиток
   public int gridSize = 4; // Размер сетки (4x4 для классической пятнашки)
   public float spacing = 1f; // Расстояние между плитками
   public Vector2Int emptyPosition; // Позиция пустой клетки
   public bool isSolved = false; // Флаг, указывающий на решение пятнашек
   private List<GameObject> tiles = new List<GameObject>(); // Список всех плиток
   void Start()
   {
      GenerateTiles();
      ShuffleTiles();
   }
   void Update()
   {
      if (!isSolved)
      {          CheckSolution();       }   }
   void GenerateTiles()
   {
      for (int y = 0; y < gridSize; y++)
      {
         for (int x = 0; x < gridSize; x++)
         {
            if (x == emptyPosition.x && y == emptyPosition.y)
            {
               continue;
            }

            GameObject tile = Instantiate(tilePrefab, puzzleParent);
            tile.transform.localPosition = new Vector3(x * spacing, y * -spacing, 0f);
            tile.GetComponent<FifteenPuzzleTile>().SetPosition(new Vector2Int(x, y));

            // Находим объект с компонентом TextMeshPro внутри префаба плитки
            GameObject textObject = tile.transform.Find("TextMeshPro").gameObject;
            if (textObject != null)
            {
               // Находим компонент TextMeshPro внутри объекта и устанавливаем текст
               TextMeshPro textMeshPro = textObject.GetComponent<TextMeshPro>();
               if (textMeshPro != null)
               {
                  int index = y * gridSize + x;
                  textMeshPro.text = index.ToString();
               }             }              tiles.Add(tile);          }       }   }
   void ShuffleTiles()
   {
      for (int i = 0; i < tiles.Count; i++)
      {
         int randomIndex = Random.Range(i, tiles.Count);
         GameObject temp = tiles[randomIndex];
         tiles[randomIndex] = tiles[i];
         tiles[i] = temp;       }
      for (int i = 0; i < tiles.Count; i++)       {
         Vector2Int position = new Vector2Int(i % gridSize, i / gridSize);
         tiles[i].GetComponent<FifteenPuzzleTile>().SetPosition(position);
         tiles[i].transform.localPosition = new Vector3(position.x * spacing, position.y * -spacing, 0f);       }          }
   void CheckSolution()
   {
       bool correct = false;
      for (int i = 0; i < tiles.Count; i++)
      {
         if (tiles[i].GetComponent<FifteenPuzzleTile>().GetPosition() == solution[i])
         {              correct = true;          }
         else          {              correct = false;          }
         if ( i == tiles.Count &&  correct)          {
            Debug.Log("Пятнашки решены!");
            isSolved = true;
         }       }   }}
```

Листинг. 10. Скрипт логики размещения плиток.

Замок и ключ:

- Скрипт замка: добавляется компонентом к нужному объекту (дверь или ящик), и присваивает состояние «закрыто ключом», включает проверку в коллайдер триггер двери ключа и сопутствующий звук открывания двери.

- Скрипт ключа: ключ – интерактивный объект, который необходимо найти и применить к нужному объекту, позволяющий открыть подходящий замок.

Эти элементы логики уровня «Квартира» создают приятный и интерактивный игровой опыт, взаимодействие с окружающим миром и решение головоломок.

VI. Исправление ошибок

Исправление ошибок – один из важнейших этапов разработки любого игрового приложения. Устранение багов обеспечивает стабильность работы и качество продукта. после разработки уровней и добавление к ним необходимой логики, нужно провести необходимые тесты для работоспособности головоломок и взаимодействий.

Путем взаимодействия с объектами уровня «Главное меню» критических багов не найдено, есть небольшие графические и логические, но они никак не препятствуют начать игру.

На уровне «Серверная» был обнаружен критический баг, нарушающий логику скрипта передвижения (**Ошибка! Источник ссылки не найден.**), был исправлен компонент «Player Controller», увеличен минимальный порог для поднятия модели игрока, также был обнаружен баг, связанный с отсутствием гравитации у игрока, также исправлено.

На уровне «Квартира» исправлены скрипты взаимодействия с объектами дверей и ящиков, в скрипте допущена ошибка порядка взаимодействия из-за чего двери и ящики не открывались, был исправлен метод управляющий рукой и дверью. Также в головоломке был критический баг, мешающий решению головоломки, был изменен способ взаимодействия с плитками.

Разработка игрового VR-приложения на платформе Unity, это комплексный и многоэтапный процесс, в который включены планирование, проектирование, реализацию и тестирование продукта. В ходе разработки достигнуты ключевые результаты:

- разработан сценарий

- разработаны игровые уровни.
- созданы необходимы элементы логики, а именно необходимые анимации и скрипты.
- исправлены найденные в процессе разработки багов.

В результате создано игровое VR-приложение, которое сочетает в себе поддержку большинства VR-гарнитур, реализовано интуитивное управление. Этот проект в рамках выпускной квалификационной работы демонстрирует большую часть возможностей игрового движка Unity и современных технологий VR-гарнитур. Проделанная работа удовлетворяет первоначальные требования технического задания и открывает перспективы для дальнейшего расширения и улучшение проекта.

2.3 Тестирование игрового приложения. Результаты апробации

После разработки игрового приложения был скомпилирован исполняемый exe файл данного приложения, затем отправлен группе студентов института математики, физики, информатики Уральского государственного педагогического университета, которые тестировали возможности и удобство использования данным проектом. Итоги апробации определялись путем экспертного оценивания: участникам тестирования (апробации) была выдана анкета с рядом критериев, каждый из которых можно оценить по 5-бальной шкале, где 0 – «очень плохо», а 5 – «очень хорошо».

Критерии, представленные в анкете, были следующие:

1. Корректность работы
2. Удобство пользовательского интерфейса
3. Сложность игровых заданий
4. Удовольствие от игры
5. Удовлетворение реализованным программно-аппаратным комплексом техническому заданию

Результаты апробации представлены в Таблица 1. Согласованность мнений экспертов по каждому из критериев выражается коэффициентом вариации C_v.

Таблица 1

Экспертные оценки, уровень согласованности экспертов

		Эксперт, оценка								
		1	2	3	4	5	6	Сред-нее	C_v	Степень согласованности
Критерий	Корректность работы	5	5	5	5	5	5	5	0%	Высокая
	Удобство пользовательского интерфейса	5	5	5	5	5	5	5	0%	Высокая
	Сложность игровых заданий	5	4	5	5	4	5	4,7	8%	Высокая
	Удовольствие от игры	5	5	5	5	4	5	4,8	5%	Высокая
	Удовлетворение реализованным программно-аппаратным комплексом ТЗ	5	5	5	5	5	5	5	0%	Высокая

Среди рекомендаций и замечаний экспертов по совершенствованию игрового приложения были следующие: «Не всегда понятный алгоритм действий, нужный для выполнения задания», «Мало подсказок».

Таким образом, по результатам апробации, система готова для дальнейшего распространения.

Заключение

Для выпускной квалификационной работы было разработано игровое VR-приложение на платформе Unity.

В ходе работы:

1. Проанализирована история развития видеоигр, VR, выделены современные тенденции в жанрах, дизайне и технологиях разработки игровых приложений.
2. Произведен анализ доступных технологий и программного обеспечения для реализации проекта, освоена среда разработки Unity и язык программирования C#, используемый для написания скриптов.
3. В соответствии с техническим заданием произведена разработка игрового VR-приложения на основе выбранных VR технологий и программного обеспечения для разработки, Unity и язык программирования C#.
4. Произведена апробация разработанного игрового VR-приложения методом экспертных оценок.

При разработке приложения была изучена и использовалась среда разработки Unity, предназначенная для создания кроссплатформенных игр и приложений. Разработанное игровое приложение может побудить желание игрока продолжать играть в головоломки, развиваться и не стагнировать.

Таким образом, следует считать, что результаты разработки соответствуют всем требованиям технического задания, поставленная цель достигнута. Работа носит законченный характер.

Список информационных источников

1. Bomb Defusal Manual : сайт. – URL: https://www.bombmanual.com/ (дата обращения: 26.06.2023)
2. Charming Puzzle Game 'Ghost Giant' Is Coming to PlayStation VR : сайт. – URL: https://variety.com/2018/gaming/news/ghost-giant-psvr-1202838621/ (дата обращения: 26.06.2023)
3. Ghost Giant physical edition launches May 7 in North America : сайт. – URL: https://www.gematsu.com/2019/04/ghost-giant-physical-edition-launches-may-7-in-north-america (дата обращения: 26.06.2023)
4. Philosophy, puzzles and Tetris in The Talos Principle next month : сайт. – URL: https://www.engadget.com/2014-11-04-philosophy-puzzles-and-tetris-in-the-talos-principle-next-month.html (дата обращения: 26.06.2023)
5. О свободе и нелинейности в играх : сайт. – URL: https://stopgame.ru/blogs/topic/106672/o_svobode_i_nelineynosti_v_igrah (дата обращения: 26.06.2023)
6. Первые шаги в Unity с OpenXR : сайт. – URL: https://habr.com/ru/companies/otus/articles/685506/ (дата обращения: 26.06.2023)
7. Running in 30 mins: Virtual Reality Development Basics with Unity and Oculus : сайт. – URL: https://medium.com/@fallenfate/running-in-30-mins-virtual-reality-development-basic-with-unity-and-oculus-f2300fd55842 (дата обращения: 26.06.2023)
8. SteamVR API : сайт. – URL: https://valvesoftware.github.io/steamvr_unity_plugin/api/Valve.VR.html (дата обращения: 26.06.2023)
9. VR development in Unity : сайт. – URL: https://docs.unity3d.com/Manual/VROverview.html (дата обращения: 26.06.2023)

10. Как делать головоломки в играх : сайт. – URL: https://dtf.ru/gamedev/1075017-kak-delat-golovolomki-v-igrah-rasskazyvaet-geymdizayner-supraland (дата обращения: 26.06.2023)
11. Геймдизайн VR-игр: особенности и рекомендации : сайт. – URL: https://dtf.ru/gamedev/1250012-geymdizayn-vr-igr-osobennosti-i-rekomendacii (дата обращения: 26.06.2023)
12. VR Design for Game Designers — Part 1 : сайт. – URL: https://medium.com/kids-digital/vr-design-for-game-designers-816efe83b93e (дата обращения: 26.06.2023)
13. VR Design for Game Designers — Part 2 : сайт. – URL: https://medium.com/kids-digital/part-2-cater-for-my-phone-5ec9b2665f24#.1rc9v24ib (дата обращения: 26.06.2023)
14. VR Design for Game Designers — Part 3 : сайт. – URL: https://medium.com/kids-digital/vr-design-for-game-designers-part-3-523fbe2d9ab0#.myh66upfc (дата обращения: 26.06.2023)
15. VR Gaming: What Are the Best Puzzle VR Games? : сайт. – URL: https://sensoriumxr.com/articles/best-puzzle-vr-games (дата обращения: 26.06.2023)
16. Лучшие головоломки в VR // Portal VR : сайт. – URL: https://portal-vr.ru/luchshie-golovolomki-v-vr/ (дата обращения: 07.06.2024)
17. Что такое ассеты Unity // SkillBox : сайт. – URL: https://skillbox.ru/media/gamedev/chto_takoe_assety_unity/ (дата обращения: 07.06.2024)
18. Unreal Engine // SkillFactory : сайт. – URL: https://blog.skillfactory.ru/glossary/unreal-engine/ (дата обращения: 07.06.2024)
19. Unity (игровой движок) // Википедия : сайт. – URL: https://ru.wikipedia.org/wiki/Unity_(игровой_движок) (дата обращения: 07.06.2024)

20. Unreal Engine : сайт. – URL: https://www.unrealengine.com/en-US (дата обращения: 07.06.2024)
21. Unreal Engine // Википедия : сайт. – URL: https://ru.wikipedia.org/wiki/Unreal_Engine (дата обращения: 07.06.2024)
22. Unigine // Википедия : сайт. – URL: https://ru.wikipedia.org/wiki/Unigine (дата обращения: 07.06.2024)
23. Визуальный скриптинг Unity // Unity : сайт. – URL: https://unity.com/ru/features/unity-visual-scripting (дата обращения: 07.06.2024)
24. Разработка компьютерных игр // Википедия : сайт. – URL: https://ru.wikipedia.org/wiki/Разработка_компьютерных_игр (дата обращения: 07.06.2024)
25. Как работает VR? Разбор // Хабр : сайт. – URL: https://habr.com/ru/companies/droider/articles/538748/ (дата обращения: 07.06.2024)
26. Виртуальная реальность // Википедия : сайт. – URL: https://ru.wikipedia.org/wiki/Виртуальная_реальность (дата обращения: 07.06.2024)
27. SteamVR : сайт. – URL: https://www.steamvr.com/ru/ (дата обращения: 07.06.2024)
28. История видеоигр. Часть 1 // StopGame : сайт. – URL: https://stopgame.ru/blogs/topic/109336/istoriya_videoigr_chast_1 (дата обращения: 07.06.2024)
29. История компьютерных игр // Википедия : сайт. – URL: https://ru.wikipedia.org/wiki/История_компьютерных_игр#Аркадные_автоматы (дата обращения: 07.06.2024)
30. VR в киберспорте: будущее или очередной пшик? // Чемпионат : сайт. – URL: https://www.championat.com/cybersport/article-3296181-kibersport-i-vr.html (дата обращения: 07.06.2024)

31. История VR: от неловких попыток до отслеживания жестов и полного погружения // SkillBox : сайт. – URL: https://skillbox.ru/media/code/istoriya-vr-ot-nelovkikh-popytok-do-otslezhivaniya-zhestov-i-polnogo-pogruzheniya/ (дата обращения: 07.06.2024)
32. Virtual reality game // Википедия : сайт. – URL: https://en.wikipedia.org/wiki/Virtual_reality_game (дата обращения: 07.06.2024)
33. Игровой движок // Википедия : сайт. – URL: https://ru.wikipedia.org/wiki/Игровой_движок#cite_note-1 (дата обращения: 07.06.2024)
34. Как разобраться в игровых движках // DTF : сайт. – URL: https://dtf.ru/u/48338-akkaunt-ne-ispolzuetsya/236542-kak-razobratsya-v-igrovyh-dvizhkah (дата обращения: 07.06.2024)
35. Почему играть - полезно? // PlayGround : сайт. – URL: https://www.playground.ru/misc/news/pochemu_igrat_polezno-227145 (дата обращения: 07.06.2024)
36. ИГРОВАЯ ИНДУСТРИЯ: ГЕЙМДЕВ (GAMEDEV) // ВШЭ : сайт. – URL: https://hsbi.hse.ru/articles/igrovaya-industriya-geymdev/ (дата обращения: 07.06.2024)
37. Unity // SkillFactory : сайт. – URL: https://blog.skillfactory.ru/glossary/unity/ (дата обращения: 07.06.2024)
38. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы: межгосударственный стандарт: утв. и введ. в действие Постановлением Государственного комитета СССР по стандартам от 24 марта 1989 г. No 661: дата введ. 1990-01-01: переиздание июнь 2009 г. / разработан и внесен Государственным комитетом СССР по стандартам, Министерством приборостроения, средств автоматизации и систем управления СССР. – М.: Стандартинформ, 2009. – Текст: электронный // Электронный фонт правовых и нормативно-технических документов:

[сайт]. – URL: https://docs.cntd.ru/document/1200006924 (дата обращения 27.05.2024).

Приложения

Приложение 1.