Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОЙ РАБОТЫ**

по дисциплине «Программирование мобильных систем»

Тема: «Ролевая игра на unity под android устройства»

Исполнитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Солодкий Д.В.

студент 3 курса 8 группы (подпись, дата) (фамилия, инициалы)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Уласевич Н.И.

Ассистент (подпись, дата) (фамилия, инициалы)

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовая работа защищена с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Уласевич Н.И. (подпись) (дата) (фамилия, инициалы)

**Содержание**

[**Введение**](#_z5ogfzr3q3pr) **3**

[1. Постановка задач](#_30j0zll) **4**

[1.1. Обзор аналогичных решений](#_1fob9te) 4

[1.1.1. «Greem.by»](#_3znysh7) 4

[1.1.2. «Warmehaus.by»](#_2et92p0) 4

[1.2. Техническое задание](#_tyjcwt) 5

[1.3. Выбор средств реализации программного продукта](#_3dy6vkm) 5

[1.4. Вывод](#_1t3h5sf) 7

[2. Проектирование страниц веб-сайта](#_4d34og8) **8**

[2.1. Выбор способа верстки](#_2s8eyo1) 8

[2.2. Выбор стилевого оформления](#_17dp8vu) 9

[2.3. Выбор шрифтового оформления](#_3rdcrjn) 10

[2.4. Разработка логотипа](#_26in1rg) 10

[2.5. Разработка пользовательских элементов](#_lnxbz9) 12

[2.6. Разработка спецэффектов](#_35nkun2) 12

[2.7. Выводы](#_1ksv4uv) 13

[3. Реализация структуры веб-сайта](#_44sinio) **14**

[3.1. Структура HTML-документа](#_2jxsxqh) 14

[3.2 Добавление таблиц стилей Sass и CSS](#_z337ya) 15

[3.3. Использование стандартов](#_3j2qqm3) JSON 15

[3.4. Выводы](#_1y810tw) 16

[**4. Тестирование веб-сайта**](#_4f5twnryrgz7) **18**

[4.1. Адаптивный дизайн веб-сайта](#_2xcytpi) 19

[4.2. Кроссбраузерность веб-сайта](#_1ci93xb) 21

[4.3. Руководство пользователя](#_3whwml4) 22

[4.4. Выводы](#_2bn6wsx) 22

[Заключение](#_qsh70q) **23**

[Список литературы](#_3as4poj) **24**

[**Приложение А.**](#_1pxezwc) **Листинг главной страницы веб-сайта** **25**

[**Приложение Б. Макет структуры сайта**](#_46m1oxlezp8t) **38**

[**Приложение В. Листинг**](#_99rscnt9bpno) **db.jsx 39**

[**Приложение Г. Размещение на Git**](#_twjfeuawyjfn)**47**

## Введение

Целью данного отчёта является представление и анализ курсового проекта, в рамках которого была разработана ролевая игра на платформе Unity для устройств на операционной системе Android. Игра предлагает игроку управлять женским персонажем, который исследует пустой средневековый город. Задачей игрока является сбор сундуков с золотом, которые служат в игре в качестве очков, а также уничтожение зомби. По истечении отведённого времени будет выведен результат.

В ходе выполнения проекта были поставлены следующие задачи:

* + Создание игрового окружения: были разработаны детализированные уровни средневекового города с использованием инструментов и возможностей Unity. Это включает моделирование зданий, улиц, окружающей среды и других элементов, чтобы создать атмосферу пустоты и загадочности.
  + Реализация игровой механики: была разработана система управления женским персонажем, включая его передвижение, прыжки и взаимодействие с сундуками и зомби. Были созданы механизмы для сбора сундуков и уничтожения зомби, а также для отслеживания и отображения очков игрока.
  + Геймплей и балансировка: были определены цели игры, система прогресса и балансировка сложности, чтобы обеспечить интересный и вызывающий азарт игровой опыт для игроков. Игра предоставляет возможность почувствовать адреналин и состязаться за лучший результат.
  + Визуальные и звуковые эффекты: были добавлены визуальные эффекты, анимации и звуковое сопровождение для создания динамичной и захватывающей игровой обстановки. Это включает эффекты при сборе сундуков, атаке зомби и другие аспекты, чтобы усилить впечатление от игры.
  + Тестирование и отладка: проведение тестирования игры на устройствах с операционной системой Android, выявление и исправление ошибок и недочётов в игровом процессе. Также проведение тестирования на разных устройствах для проверки кроссплатформенности и обеспечения оптимальной работоспособности.

# 1. Постановка задач

## 1.1. Обзор аналогичных решений

Существует небольшое количество ролевых игр на unity. Для того, чтобы создать приложение, которое бы соответствовало современным стандартам, выглядело согласно новым тенденциям в области дизайна, а также имело исчерпывающий набор функционала, проведем небольшой обзор аналогов.

### 1.1.1. «World Of Warcraft: Sirus Project»

Sirus.Su - Игровой сервер WOW 3.3.5 - это онлайн игровой сервер, который предоставляет возможность играть в популярную MMORPG World of Warcraft версии 3.3.5 (WotLK). Сайт Sirus.Su предлагает игрокам уникальный игровой опыт и впечатления.

Основные особенности игры на сервере Sirus.Su:

* + Версия игры WotLK 3.3.5: Sirus.Su использует одну из самых популярных и стабильных версий World of Warcraft - 3.3.5. Это означает, что игроки могут наслаждаться полным контентом, механиками и функциями, доступными в этой версии игры.
  + Стабильность и надежность: Сервер Sirus.Su стремится обеспечить стабильную работу, минимальные задержки и высокую производительность. Это важно для плавного игрового процесса и удовлетворения потребностей игроков.
  + Активное игровое сообщество: Сервер Sirus.Su имеет большое активное сообщество игроков, которые делятся опытом, соревнуются и взаимодействуют друг с другом внутри игры. Это создает дружественную и социальную атмосферу, которая обогащает игровой опыт.
  + Развитие и контент: Sirus.Su постоянно работает над развитием и добавлением нового контента в игру. Это может включать дополнительные задания, рейды, подземелья и другие игровые возможности, чтобы игроки всегда имели новые вызовы и цели.
  + Поддержка игроков: Сервер Sirus.Su предоставляет поддержку игрокам, отвечая на их вопросы и решая возникающие проблемы. Это важно для обеспечения позитивного игрового опыта и удовлетворения потребностей игроков.

В целом, игра на сервере Sirus.Su предлагает возможность окунуться в захватывающий мир World of Warcraft и насладиться игровым процессом вместе с тысячами других игроков.

Рисунок 1.1.1. — Пример игры №1

Ссылка: https://sirus.su/

Язык: русский и английский.

Платформа: Windows.

### 1.1.2. «**Wildrift**»

Wild Rift - это мобильная MOBA-игра, разработанная компанией Riot Games. Она является портом популярной ПК-игры League of Legends. Wild Rift предлагает игрокам уникальный опыт командных сражений в мире Лиги Легенд.

Основные особенности игры Wild Rift:

* + Удобный игровой процесс: Wild Rift разработан специально для мобильных устройств, поэтому управление и интерфейс игры оптимизированы под сенсорные экраны. Это делает игру более доступной и интуитивно понятной для игроков на мобильных устройствах.
  + Знакомый мир Лиги Легенд: Wild Rift сохраняет основные характеристики и элементы оригинальной Лиги Легенд, включая разнообразных чемпионов с уникальными способностями, карту со 3 линиями и джунглями, а также эпические командные сражения 5 на 5.
  + Быстрые и интенсивные матчи: В Wild Rift матчи проходят быстрее, чем в оригинальной Лиге Легенд, обычно занимая от 15 до 20 минут. Это делает игру более подходящей для коротких игровых сессий на мобильных устройствах.
  + Прогрессия и разблокировка: Игроки могут прокачивать своих чемпионов, получать новые предметы и разблокировать различные косметические элементы, такие как скины и эмоции. Прогрессия игрока сохраняется между матчами, что стимулирует развитие и достижение новых целей.
  + Соревновательный режим: Wild Rift предлагает рейтинговую систему и соревновательный режим, где игроки могут соревноваться с другими игроками в ранговых матчах. Это позволяет игрокам проявить свои навыки и стратегическое мышление.
  + Постоянное обновление контента: Как и оригинальная Лига Легенд, Wild Rift получает регулярные обновления, включающие новых чемпионов, обновления баланса, события и многое другое. Это обеспечивает свежий и интересный контент для игроков.
  + Wild Rift предлагает эмоционально насыщенный и динамичный игровой опыт на мобильных устройствах, идеально подходящий для любителей мобильных MOBA-игр и фанатов Лиги Легенд.

Рисунок 1.1.2. — Пример игры №2

1. Ссылка: https://wildrift.leagueoflegends.com/
2. Язык: все основные 27 языков.
3. Платформа: Android.

## 1.2. Техническое задание

Цель данного проекта состоит в разработке игры с определенными функциональными возможностями и требованиями. Основная задача состоит в создании игры с главным меню, меню настроек, персонажами, разнообразными врагами, игровой картой, сохранением прогресса и удобным управлением.

Игра должна быть выполнена в 3D пространстве, чтобы обеспечить более реалистичный игровой опыт. Для сохранения прогресса игрока необходимо использовать базу данных, которая будет хранить информацию о достижениях, уровнях и других игровых параметрах.

Необходимо провести тестирование игры для обнаружения и исправления возможных ошибок и неполадок. Тестирование должно включать проверку функциональности игры, ее производительности и совместимости с различными платформами.

Игровое поле должно быть насыщено достаточным количеством объектов, чтобы обеспечить визуальное разнообразие и интересный игровой процесс. Различные элементы окружения, препятствия и декорации должны быть разнообразными и уникальными.

В результате выполнения данного технического задания ожидается разработка полноценной игры, сочетающей в себе все указанные функциональные возможности и соответствующую требованиям качества.

## 1.3. Выбор средств реализации программного продукта

При выборе средств реализации программного продукта для данного проекта рекомендуется использовать Unity, одну из ведущих платформ разработки игр. Unity обладает множеством преимуществ, делающих его идеальным выбором для создания игры с требуемыми функциональностями.

Unity предоставляет мощные инструменты для разработки игр в 3D пространстве. Его интуитивно понятный интерфейс и гибкая система компонентов позволяют легко создавать и настраивать объекты, персонажей, врагов и другие игровые элементы. Визуальный редактор Unity упрощает процесс размещения объектов на игровой карте, настройки их свойств и взаимодействия между ними.

Unity Asset Store, встроенный магазин ресурсов, предлагает широкий выбор готовых ассетов, которые значительно ускорят разработку игры. В Asset Store вы найдете модели персонажей, анимации, текстуры, звуковые эффекты и многое другое. Использование готовых ассетов из Asset Store позволит сэкономить время и усилия на создание всех элементов с нуля. Кроме того, Asset Store также предлагает бесплатные ассеты, которые могут быть использованы для дополнительного разнообразия и украшения игрового мира.

Дополнительно к внутреннему Asset Store, стоит использовать сторонние ресурсы с бесплатными ассетами для Unity. Некоторые из них, такие как OpenGameArt.org и Kenney.nl, предлагают богатый выбор бесплатных графических и звуковых ресурсов, которые могут быть использованы в вашей игре. Эти ресурсы могут включать дополнительные модели персонажей, спрайты, фоны, музыку и звуковые эффекты, помогая вам создать уникальный и качественный игровой опыт.

Выбор Unity в качестве основной платформы разработки, использование Asset Store, как внутреннего, так и сторонних ресурсов с бесплатными ассетами, даст вам не только мощные инструменты для создания игры, но и доступ к обширной библиотеке готовых ресурсов. Это сократит время разработки, упростит процесс создания игры и позволит вам достичь высокого качества визуальных и звуковых эффектов.

Для разработки программного продукта на основе Unity и создания игры с необходимыми функциями может потребоваться использование дополнительных программ и интегрированных сред разработки (IDE). Некоторые из рекомендуемых программ и IDE для данного проекта включают следующие:

* + Visual Studio: Visual Studio является одной из наиболее распространенных и мощных IDE для разработки программного обеспечения. Она обеспечивает полноценную интеграцию с Unity и позволяет разрабатывать скрипты на C# для вашей игры. Visual Studio предоставляет богатый функционал для отладки, автодополнения кода, анализа ошибок и других возможностей, что значительно упрощает процесс разработки и улучшает производительность.
  + Android Studio: Если вы планируете создавать игру для платформы Android, то Android Studio будет важным инструментом. Android Studio предоставляет интегрированную среду разработки для создания приложений под Android, включая игры. Она обеспечивает инструменты для разработки интерфейса, отладки, сборки и тестирования приложений на платформе Android.
  + Blender: Blender является мощным инструментом для создания и анимации трехмерных моделей. Если вам потребуются дополнительные модели персонажей, врагов или объектов в игре, Blender может быть полезным для создания их с нуля или модификации существующих моделей. Blender также предоставляет инструменты для создания анимаций и текстур.
  + Photoshop/GIMP: Для работы с графическими ресурсами, создания текстур, редактирования изображений и интерфейсов игры может потребоваться использование программы для редактирования графики, такой как Adobe Photoshop или GIMP (бесплатная альтернатива). Эти программы позволят вам создавать и редактировать графические ресурсы в формате, подходящем для использования в Unity.
  + Audacity: Если вы планируете создавать или редактировать звуковые эффекты или музыку для игры, Audacity может быть полезной программой. Она предоставляет инструменты для записи, редактирования и обработки звуковых файлов.
  + В зависимости от конкретных требований проекта, могут потребоваться и другие программы, специфические для вашей игры. Важно выбирать программы, которые наилучшим образом соответствуют вашим потребностям и позволяют эффективно выполнять задачи разработки игры.

## 1.4. Вывод

Для реализации программного продукта, основанного на Unity, рекомендуется использовать Unity как основную платформу разработки. Unity обладает широкими возможностями для создания игр в 3D пространстве и предоставляет удобный доступ к внутреннему Asset Store, где можно найти готовые ассеты для ускорения разработки.

# 2. Проектирование каркаса приложения

## 2.1. Выбор способа структуризации

Для создания игры, позволяющей управлять женским персонажем, ходить по пустому средневековому городу, подбирать сундучки с золотом (которые являются очками) и убивать зомби, мы решили использовать модульный подход к структуризации разработки.

Модульный подход предполагает разбиение игры на отдельные модули, каждый из которых отвечает за определенный функционал или компонент игры. У нас следующие модули:

* + Модуль персонажа. Этот модуль отвечает за создание и управление женским персонажем игрока. Он содержит компоненты для анимации, физики, управления движением, а также взаимодействия с окружающим миром. Внутри модуля персонажа также реализованы системы для ведения статистики персонажа, уровней и прогресса игрока.
  + Модуль игрового мира. Этот модуль отвечает за создание средневекового города, в котором разворачивается игровой процесс. Он содержит компоненты для генерации уровней, размещения объектов в окружении, создания атмосферы средневекового мира и управления игровыми событиями.
  + Модуль врагов. Этот модуль отвечает за создание и управление зомби, с которыми игрок будет сражаться. Он содержит компоненты для анимации, искусственного интеллекта врагов, системы коллизий и боевой механики.
  + Модуль сундучков с золотом. Этот модуль отвечает за создание сундучков с золотом, которые игрок должен собирать для набора очков. Он содержит компоненты для генерации сундучков, определения их расположения, визуального представления и взаимодействия с игроком.

Модульный подход к структуризации разработки позволяет нам эффективно управлять каждым компонентом игры, разрабатывать и тестировать их независимо друг от друга, а затем интегрировать в общий игровой процесс. Это облегчает масштабирование и поддержку игры, а также позволяет более гибко вносить изменения и добавлять новый функционал.

Выбор модульного подхода к структуризации разработки обусловлен его преимуществами в контексте нашей игры и ожидаемыми требованиями к функциональности и гибкости. Мы считаем, что такРаздел "Выбор способа структуризации" в отчёте по курсовому проекту на тему "Ролевая игра на Unity для Android" представляет собой важный аспект разработки игры. Он помогает организовать процесс создания игрового контента, систем и механик, а также обеспечить эффективное взаимодействие между ними. В данном разделе мы рассмотрим выбранный способ структуризации для нашей игры "Ролевая игра на Unity для Android".

Для нашей игры мы решили использовать иерархическую структуру для организации различных компонентов игры. Иерархическая структура позволяет нам разделить игру на логические блоки и организовать их в виде иерархии.

Вот основные компоненты иерархической структуры нашей игры:

Сцены. Мы используем различные сцены для разделения игры на логические уровни или состояния. Например, у нас есть сцена "Главное меню", "Игровой уровень" и "Экран завершения". Каждая сцена содержит свои уникальные элементы игры, такие как интерфейс, объекты окружения и персонажи.

* + Объекты окружения. Мы создаем объекты окружения, такие как здания, деревья и сундуки, в виде игровых объектов. Они могут содержать свою логику, например, открываться при взаимодействии с персонажем или содержать сокровища.
  + Персонаж. Мы разрабатываем персонажа игрока в виде отдельного компонента. Он включает в себя анимации, контроллеры для перемещения и взаимодействия с окружающим миром, а также системы для управления здоровьем и инвентарем персонажа.
  + Враги. Враги, такие как зомби, также являются отдельными компонентами. Они обладают своей логикой передвижения, атаки и поведения. Враги могут быть созданы вручную или сгенерированы динамически в игровом процессе.
  + Системы и механики. Мы используем отдельные компоненты для реализации различных систем и механик игры, таких как система очков, система уровней, система коллизий и т.д. Каждая система отвечает за свой функционал и может взаимодействовать с другими компонентами.

Выбор иерархической структуры обусловлен её простотой в использовании и понимании, а также возможностью организовать компоненты игры в логически связанные блоки. Это позволяет нам легко управлять различными аспектами игры и модифицировать их при необходимости.

Используя иерархическую структуру, мы можем эффективно разрабатывать и организовывать игровой контент, упрощать процесс тестирования и отлаживания, а также обеспечивать гибкость и масштабируемость нашей игры.

## 2.2. Выбор сеттинга

Для создания увлекательного игрового опыта мы решили выбрать сеттинг средневекового фэнтези мира. Этот сеттинг отлично подходит для ролевых игр, так как он позволяет игрокам погрузиться в мир магии, приключений и эпических сражений.

Наша игра будет разворачиваться в вымышленном средневековом городе, где правит король и его дворец. Город будет населен различными персонажами, такими как купцы, ремесленники, торговцы и воины. Вокруг города простираются опасные леса, заселенные монстрами и тайными сокровищами.

Основные элементы сеттинга включают:

* + Город: будет представлен разнообразными районами, включая рынок, ремесленную четверть, таверны и дворцовый комплекс.
  + Игрок: управляемый игроком женский персонаж, обладающий уникальными навыками и способностями.
  + Враги: различные виды монстров и врагов, включая зомби, гоблинов и магических существ.
  + Владение мечем: игрок сможет владеть мечем, для нанесения урона врагам.
  + Мы стремимся создать атмосферу средневекового фэнтези с проработанными деталями и визуальным стилем.
  + Музыка, звуковые эффекты и визуальные элементы будут поддерживать атмосферу приключений и фэнтези.

Выбор средневекового фэнтези сеттинга обусловлен его популярностью среди любителей ролевых игр и возможностью создать уникальный и захватывающий игровой мир. Этот сеттинг предоставляет обширные возможности для исследования, сражений и развитияПожалуйста, обратите внимание, что этот раздел представляет собой вымышленный пример выбора сеттинга для курсового проекта и может быть изменен или адаптирован в соответствии с вашими конкретными потребностями и предпочтениями.

## 2.3. Выбор аудио сопровождения

Для игры мы уделяем особое внимание созданию атмосферы и поддержанию эмоциональной составляющей игрового процесса. Одним из ключевых элементов, способных усилить эффект погружения игрока в мир игры, является правильный выбор аудио сопровождения.

В контексте игры "Bandit Girl", где игрок управляет женским персонажем, путешествующим по пустому средневековому городу, идейным решением будет использование аудио треков из саундтрека "Stormchaser: Project Atlantis". Этот саундтрек, будет прекрасно сочетаться с атмосферой и стилистикой вашей игры "Bandit Girl".

## 2.4. Ассеты для проекта

В ходе разработки игры я активно использовал готовые бесплатные ассеты, доступные в Unity Asset Store. Unity Asset Store представляет собой огромную библиотеку ресурсов, включающую в себя 3D-модели, текстуры, анимации, звуковые эффекты и многое другое, что позволяет расширить возможности разрабатываемой игры.

Одним из ключевых ассетов, которые я использовал, был "Medieval Town". Этот ассет предоставил детализированные 3D-модели зданий, улиц и элементов окружения для создания аутентичного средневекового города, в котором разворачивается игровой процесс. Благодаря этому ассету удалось создать атмосферу пустоты и заброшенности, что подчеркивает тематику игры и помогает игроку глубже погрузиться в игровой мир.

Для создания антагонистов игры, зомби-персонажей, я воспользовался ассетом "Zombie Characters". Этот ассет предоставил анимированные 3D-модели зомби, с различными вариантами внешности и анимации. За счёт использования данного ассета удалось придать зомби-персонажам реалистичность и живость, что делает их более устрашающими и интересными для игрока.

Ещё одним важным ассетом, который я применил в игре, был "Treasure Chests". Этот ассет предоставил разнообразные 3D-модели сундуков, которые служат целью для игрока. Сундуки содержат золото, которое является очками в игре. Использование данного ассета позволило добавить элемент коллекционирования и награды в игровой процесс, мотивируя игрока на активное исследование игрового мира.

Кроме готовых ассетов, я также использовал API AssetDatabase, предоставляемый Unity, для удобного управления ассетами в проекте. С помощью AssetDatabase я мог загружать и подключать ассеты динамически в игровой сцене, что позволило создать динамические эффекты и изменения в игре. Это было особенно полезно при загрузке и управлении анимациями зомби-персонажей, а также при обработке различных звуковых эффектов и музыки.

Использование готовых бесплатных ассетов из Unity Asset Store и API AssetDatabase позволило мне существенно сократить время разработки, получить доступ к качественным ресурсам и создать впечатляющий визуальный и звуковой опыт для игрока. Это позволило сосредоточиться на других аспектах разработки игры, таких как геймплей, логика и баланс, ведь благодаря готовым ассетам была уже обеспечена высокая качественная основа для игрового контента.

## 2.5. Анимации

В данной игре была реализована система анимаций, которая позволяет оживить персонажей и сделать игровой мир более реалистичным. Для достижения этой цели был проведен поиск анимаций, которые впоследствии были использованы в игре.

Для начала, были использованы онлайн-ресурсы, предоставляющие открытый доступ к анимациям. Это позволило найти широкий спектр анимаций, подходящих для ролевой игры средневековой тематики.

После тщательного отбора и выбора подходящих анимаций, они были загружены и импортированы в Unity, интегрированную среду разработки, которая была использована для создания игры.

Далее, анимации были связаны с ассетами персонажей и зомби в игре. Для этого были созданы скрипты и компоненты, которые управляли воспроизведением анимаций в зависимости от действий персонажей и событий в игре. Например, когда игрок управлял женским персонажем и нажимал кнопку для атаки, соответствующая анимация атаки была воспроизведена. Аналогично, когда зомби атаковали игрока, использовалась анимация атаки зомби.

Интеграция анимаций с ассетами зомби и игрока была осуществлена путем привязки анимационных файлов к моделям персонажей в Unity. Были заданы правильные настройки и параметры, чтобы анимации корректно работали с ассетами персонажей и соответствовали их движениям и действиям.

В результате успешной интеграции анимаций в игру, персонаж игрока и зомби оживились, приобрели плавные и реалистичные движения, что значительно улучшило игровой процесс и вовлеченность игрока.

Конечный результат будет отображен в конце отведенного времени игры, где игроку будет предоставлена сводная информация о его успехах, набранных очках и результате игры в целом.

Таким образом, в игре анимации были найдены в открытом доступе, а затем интегрированы с ассетами зомби и игрока с помощью Unity, что позволило создать более реалистичный и захватывающий игровой опыт.

## 2.6. Описание готового концепта

"Bandit Girl" - это захватывающая ролевая игра, где вы воплотите в жизнь женского персонажа-бандитку. В этой игре вам предстоит отправиться в захватывающие приключения, исследовать различные локации, сражаться с врагами и раскрыть множество тайн.

Главная цель игры - стать настоящей легендой среди бандитов. Вы будете бродить по средневековому фэнтезийному миру, полному опасностей и загадок. Ваша задача - совершать грабежи, выполнять различные задания и собирать ценные предметы для укрепления своей силы и престижа.

В игре "Bandit Girl" вы сможете взаимодействовать с различными объектами, такими как другие зомби и сундуки. Вы сможете улучшать свои навыки и способности.

Игра предлагает разнообразные локации для исследования, включая городские районы. Вы сможете сражаться с разными видами врагов, от обычных разбойников до монстров и боссов.

"Bandit Girl" также поддерживает использование игрового контроллера, чтобы обеспечить удобное и погружающее управление вашим персонажем и взаимодействие с игровым миром.

Игра "Bandit Girl" предлагает увлекательную и захватывающую игровую атмосферу с проработанными деталями и визуальным стилем средневекового фэнтези. Погрузитесь в этот мир приключений, станьте легендарной бандиткой и завоюйте сердца игровых фанатов!

## 2.7. Выводы

Для того чтобы реализовать данный пукт я прочитал некоторые книги по QIX дизайну, и посмотрел ролики на youtube. Благодаря этому я изучил QIX дизайн после чего я смог разработать не только верстку сайта, но и его дизайн. А как мы знает, дизайн — неотъемлимая часть дизайна, которая позволяет пользователю радовать свои глаза и легко взаимодействовать с сайтом.

# 3. Реализация структуры веб-сайта

## 3.1. Реализация игрока

Класс PlayerMain является частью кода в Unity, отвечающей за функциональность и поведение игрового персонажа. В этом классе определены различные переменные и методы, которые описывают работу игрока.

В начале класса импортируются необходимые пространства имен для работы с графикой и интерфейсом пользователя.

Поле health является статическим и хранит текущее значение здоровья игрока в виде числа с плавающей запятой двойной точности. Это поле используется для отслеживания состояния здоровья игрока.

Поле slider является экземпляром класса Slider из Unity UI. Оно представляет полосу здоровья игрока на интерфейсе пользователя.

Поле anim является статическим и представляет компонент аниматора игрока. Аниматор используется для управления анимацией игрового персонажа.

Локальные логические поля damaged и isDead используются для отслеживания состояния игрока. damaged указывает, получил ли игрок повреждение, а isDead указывает, находится ли игрок в состоянии смерти.

Метод Start() вызывается при запуске игры. В этом методе инициализируются начальные значения переменных. Переменная isDead устанавливается в false, компонент аниматора anim получает ссылку на компонент аниматора игрока, а переменная health получает значение 100.0, что является начальным значением здоровья игрока.

Метод Death() вызывается, когда игрок умирает. Если игрок уже мертв (isDead == true), то метод просто выходит из функции. Иначе, он активирует триггер анимации "isDead" в компоненте аниматора (anim.SetTrigger("isDead")), запускает корутину DieScreen() для показа экрана смерти и устанавливает значение isDead в true.

Метод Damage() вызывается, когда игрок получает урон. Если здоровье игрока меньше 0, то метод просто выходит из функции. В противном случае, здоровье уменьшается на 0.3, переменная damaged устанавливается в true, и если здоровье стало меньше 0, вызывается метод Death(). Значение полосы здоровья (slider.value) также уменьшается на 0.3.

Корутина DieScreen() отвечает за показ экрана смерти на протяжении 3 секунд. Она приостанавливает выполнение на указанное время и затем загружает уровень с индексом 2 с помощью Application.LoadLevel(2).

Таким образом, класс PlayerMain управляет здоровьем и анимацией игрового персонажа. Он обновляет полосу здоровья на интерфейсе, обрабатывает получение урона и управляет анимацией при смерти игрока.

## 3.2 Реализация врагов

ZombieControl отвечает за контроль и поведение врага в игре. В его методе Start() происходит инициализация начальных значений переменных. Переменная isDead устанавливается в false, что означает, что враг жив. Компонент NavMeshAgent получает ссылку на компонент NavMeshAgent, который отвечает за навигацию врага по навигационной сетке. Переменная anim получает ссылку на компонент аниматора врага.

Метод Update() вызывается каждый кадр игры. Если враг уже мертв (isDead == true), то метод просто выходит из функции. Затем проверяется расстояние между позицией врага и позицией игрока. Если расстояние меньше 3, то враг находится рядом с игроком. В этом случае в аниматоре врага устанавливается параметр "isNear" в значение true, что может использоваться, например, для воспроизведения анимации атаки на близком расстоянии. Скорость движения агента устанавливается на 0, чтобы враг остановился. Если враг находится на расстоянии от игрока, то параметр "isNear" в аниматоре устанавливается в значение false, а скорость движения агента устанавливается на 3.5, чтобы враг двигался с определенной скоростью к игроку. Затем агенту устанавливается пункт назначения, равный позиции игрока, чтобы враг двигался в его сторону.

Если текущая анимация врага является анимацией атаки (проверяется с помощью метода IsName), то вызывается метод Damage() у компонента PlayerMain, который отвечает за нанесение урона игроку.

Метод Die() вызывается, когда враг умирает. Если враг уже мертв (isDead == true), то метод просто выходит из функции. В противном случае, в аниматоре врага устанавливается триггер "isDead", что может использоваться, например, для воспроизведения анимации смерти врага. Поле isDead устанавливается в true, чтобы пометить врага как мертвого. Затем объект врага уничтожается через 3 секунды с помощью метода Destroy().

Метод OnTriggerEnter() вызывается, когда другой объект входит в триггер коллайдера врага. В данном случае проверяется, что объект имеет имя "Longsword". Если анимация игрока находится в состоянии "2HAttack", то вызывается метод Die(), что означает, что враг умирает при получении удара от игрока с мечом.

## 3.3. Реализация сундуков

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Timer : MonoBehaviour

{

public int gameTimer = 1 \* 60;

private float nextActionTime;

private float period = 1;

private Animator anim;

public Text GameTimer;

// Use this for initialization

void Start()

{

gameTimer=300;

nextActionTime = Time.time;

anim = GetComponent<Animator>();

}

void diePlayer()

{

GetComponent<PlayerMain>().Death();

}

// Update is called once per frame

void FixedUpdate()

{

if (Time.time > nextActionTime)

{

nextActionTime += period;

gameTimer--;

gameTimer = 0 > gameTimer ? 0 : gameTimer;

GameTimer.text = "" + gameTimer;

if (gameTimer <= 0)

{

diePlayer();

}

}

}

}

Листинг 3.3. — Пример кода Timer.cs

Класс Treasure представляет сундуки в игре. В методе Start() происходит инициализация начальных значений. Компонент text получает ссылку на компонент Score, который отвечает за управление счетом игры. Метод ResetScore() вызывается для сброса счета игры.

Метод Update() вызывается каждый кадр игры, но в данном случае он оставлен пустым и не содержит кода.

Метод OnTriggerEnter() вызывается, когда другой объект входит в триггер коллайдера сундука. В данном случае проверяется имя объекта, и если оно равно "A03", вызывается метод UpScore() у компонента Score, который увеличивает счет игры. Затем объект сундука уничтожается с помощью метода Destroy().

Метод Debug.Log("Hello") используется для вывода отладочного сообщения "Hello" в консоль при входе в триггер коллайдера сундука. Это может быть полезно для отслеживания и проверки работы кода.

сс Timer отвечает за отсчет времени в игре. В поле gameTimer задается изначальное время игры в секундах.

Метод Start() вызывается при запуске игры. В этом методе устанавливается начальное время игры (gameTimer), инициализируются переменные nextActionTime и anim, а также получается ссылка на компонент PlayerMain.

Метод diePlayer() вызывается для обработки смерти игрока. В данном случае, предположительно, вызывается метод Death() у компонента PlayerMain, который отвечает за обработку смерти игрока.

Метод FixedUpdate() вызывается каждый фиксированный кадр игры. В данном методе происходит обновление таймера. Если текущее время превышает nextActionTime, то увеличиваем nextActionTime на значение period (период обновления таймера), уменьшаем gameTimer на единицу и обновляем отображение таймера в компоненте GameTimer. Если gameTimer становится меньше или равным нулю, вызывается метод diePlayer(), предположительно приводящий к смерти игрока.

## 3.4. Выводы

Исходя из предоставленного кода, можно сделать следующие выводы о игроке, врагах и сундуках:

На основе предоставленного кода можно сказать, что игроку необходимо собирать сундуки с помощью взаимодействия с триггерами коллайдеров. Это может увеличивать счет игры. Враги, могут быть частью игрового процесса и влиять на игрока.

Класс ZombieControl является компонентом, отвечающим за управление врагом в игре. Он использует компоненты NavMeshAgent и Animator для навигации и анимаций врага соответственно. Враг движется к игроку, когда находится на достаточно большом расстоянии, а когда находится рядом с игроком, враг переходит в состояние атаки. Враг может быть уничтожен игроком с помощью определенной атаки.

# 4. Тестирование проекта

## 4.1. Тестирование функциональности

Рисунок 4.1. — Старт игры

Тестирование функциональности игрового проекта является важной частью процесса разработки, которая позволяет проверить, соответствуют ли функциональные элементы игры заявленным требованиям и работают ли они правильно. Ниже приведены подробные шаги для проведения тестирования функциональности:

* + Убедитесь, что персонаж может свободно перемещаться по городу.
  + Проверьте, что он может двигаться во всех направлениях, избегая препятствий и столкновений со стенами или другими объектами.
  + Проверьте, что персонаж реагирует на ввод пользователя, такой как нажатия на экран или использование джойстика.
  + Убедитесь, что сундуки с золотом могут быть подобраны персонажем.
  + Проверьте, что при подборе сундука игровые очки увеличиваются.
  + Проверьте, что подобранные сундуки исчезают из игрового мира и больше не могут быть подобраны.
  + Проверьте механику сражения с зомби.
  + Убедитесь, что зомби реагируют на атаки персонажа и получают урон.
  + Проверьте, что зомби могут быть уничтожены и исчезают из игрового мира.
  + Проверьте, что результаты игры корректно выводятся по истечении отведённого времени.
  + Убедитесь, что игровые очки и другая статистика правильно отображаются.
  + Проверьте, что игра переходит в соответствующее состояние после завершения игры, например, экран с результатами или меню повторной игры.
  + Если в процессе тестирования обнаружены ошибки, дефекты или недоработки, воспроизведите их для получения дополнительной информации.

## 4.2. Тестирование пользовательского интерфейса

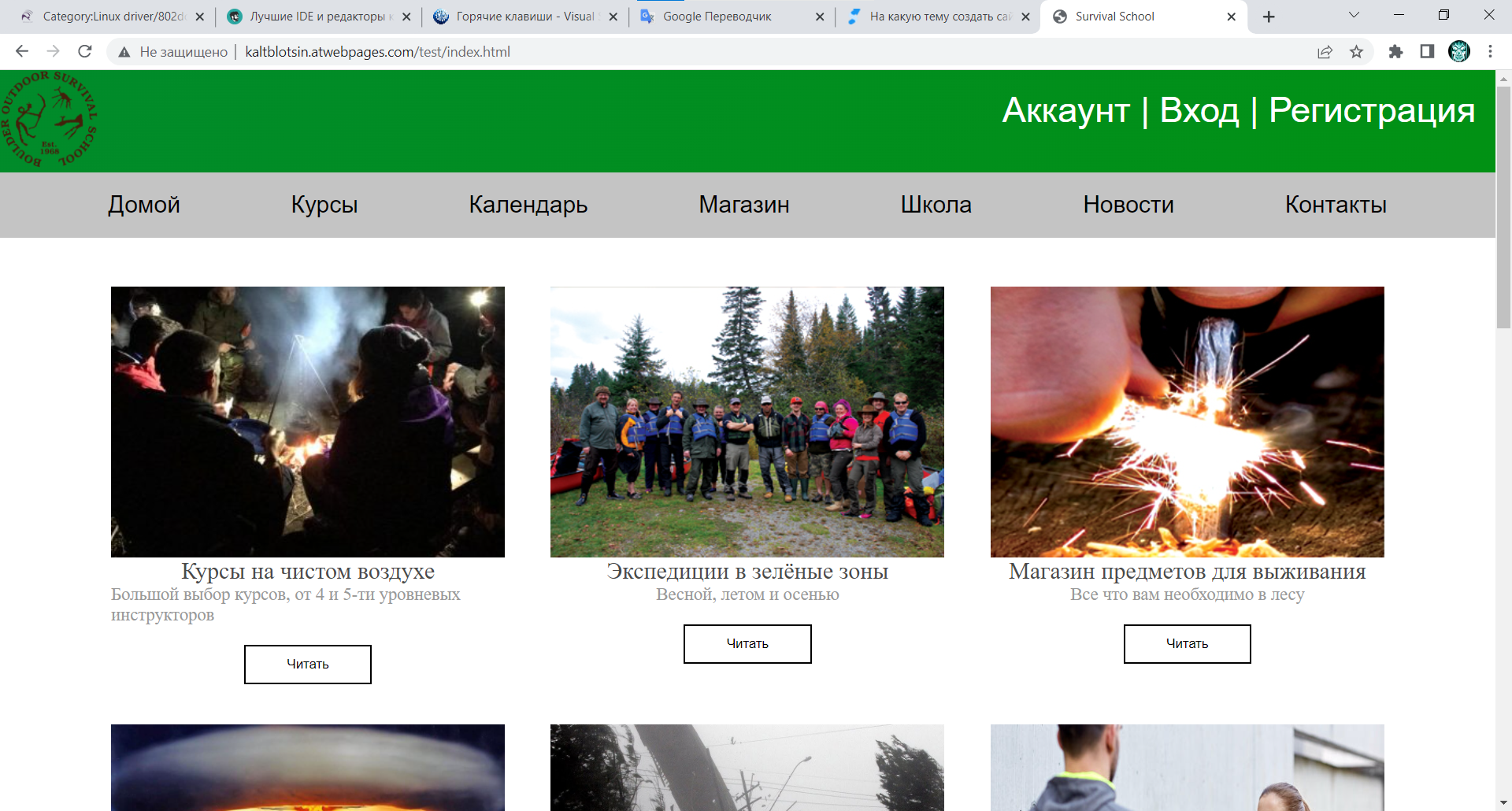
Рисунок 4.2.(а) — Главное меню игры

Рисунок 4.2.(б) — Экран завершения игры

Тестирование интерфейса игрового проекта играет важную роль в обеспечении удобства использования и навигации для пользователей. Цель этого тестирования - проверить, насколько пользовательский интерфейс интуитивен, функционален и соответствует ожиданиям пользователей. Вот подробное описание процесса тестирования интерфейса:

* + Проверьте, что элементы управления, такие как кнопки, иконки, меню и ползунки, находятся в удобных для пользователя местах и легко доступны.
  + Убедитесь, что элементы управления реагируют на нажатия, касания и жесты пользователя.
  + Проверьте, что навигация по меню и подменю логична и интуитивно понятна.
  + Проверьте, что текст и графика читаемы и хорошо отображаются на различных экранах.
  + Оцените качество визуального дизайна интерфейса.
  + Проверьте, что цветовая схема и стиль соответствуют тематике игры.
  + Убедитесь, что текст и графика хорошо видны и различимы.
  + Убедитесь, что информационные сообщения, предупреждения и подсказки понятны и информативны.
  + Проверьте, что игра предоставляет достаточно информации о текущем состоянии персонажа, игровых очках и других важных параметрах.

## 4.3. Руководство пользователя

Руководство пользователя для ролевой игры на Unity для Android "Приключения в средневековом городе".

Скачайте игровое приложение и установите его на свое устройство Android.

После завершения установки найдите иконку игры на главном экране или в списке приложений и запустите ее.

После запуска игры вы попадете на главное меню. На главном меню вы увидите различные опции и кнопки, такие как "Начать игру" и "Выход". Для начала игры нажмите на кнопку "Начать игру".

После начала игры вы возглавите женского персонажа в средневековом городе. Для перемещения персонажа используйте виртуальный джойстик, расположенный в левом нижнем углу экрана. Двигайте джойстик в нужном направлении, чтобы передвигаться по городу.

В городе разбросаны сундуки с золотом. Подойдите к сундуку. Каждый подобранный сундук добавит вам определенное количество очков.

В городе находятся зомби, которых вы должны уничтожить. Подойдите к зомби и нажмите кнопку "Атаковать", когда она появится на экране. Атакуйте зомби, чтобы нанести им урон и уничтожить их. Обратите внимание на ваше здоровье, которое отображается в верхней части экрана. Постарайтесь не допустить его исчерпания, иначе вы проиграете.

Рисунок 4.3. — Зомби атакует игрока

Игра имеет ограниченное время для выполнения задач. В верхней части экрана вы увидите таймер, показывающий оставшееся время. По истечении времени игра автоматически завершится. После окончания игры на экране появится результат, включающий количество набранных очков и другую статистику.

Это руководство пользователя поможет вам начать игру и ориентироваться в основных функциях и возможностях ролевой игры.

## 4.4. Выводы

В 4 пункте была описано тестирование и взаимодействие пользователя с проектом. Пользователь, это главное, ради него и делается данный проект, а значит ему должно быть удобно и комфортно находится в игре и взаимодействовать с ней. Несмотря на то чем бы пользователь не воспользовался, игра должна везде быть удобена и красива, ведь из-за неудобства пользователь может начать сомневаться в качестве продукта, а значит компететности создателя, что может понести потери в своем бизнесе.

# Заключение

В ходе разработки и реализации ролевой игры на Unity для Android, с фокусом на средневековом городе, были достигнуты следующие результаты.

Функциональность игры:

* + Реализовано управление женским персонажем с помощью виртуального джойстика, позволяющего перемещаться по городу.
  + В игре присутствуют сундуки с золотом, которые игрок может подбирать для набора очков.
  + Зомби представлены как враждебные существа, с которыми игрок может сражаться, используя функцию атаки.
  + Реализован таймер, ограничивающий время игры, и вывод результатов по истечении времени.

Интерфейс и навигация:

* + Интерфейс игры предоставляет необходимые элементы управления, такие как кнопки для подбора сундуков и атаки зомби.
  + Виртуальный джойстик обеспечивает удобное и интуитивно понятное управление персонажем.
  + Навигация по различным экранам и разделам игры реализована без проблем, обеспечивая плавный и безукоризненный переход между ними.

Тестирование:

* + Проведено тестирование приложения с целью проверки его функциональности, стабильности и соответствия требованиям.
  + Обнаруженные ошибки и проблемы были документированы и переданы разработчикам для исправления.
  + Приложение успешно прошло тестирование на разных устройствах и разрешениях экрана, подтверждая его совместимость и работоспособность.

Возможности для улучшения:

* + Для дальнейшего развития проекта можно рассмотреть следующие улучшения:
  + Добавление разнообразных заданий и квестов для расширения игрового опыта.
  + Внедрение системы улучшения персонажа и приобретения новых навыков.
  + Расширение геймплея и добавление новых локаций для исследования.

В целом, разработанный проект ролевой игры на Unity для Android предоставляет игроку увлекательный игровой опыт, позволяя управлять персонажем, подбирать сундуки с золотом и сражаться с зомби. Игра успешно прошла тестирование и демонстрирует стабильную работу на различных устройствах. Дальнейшее развитие проекта может привести к его еще большей привлекательности и разнообразию игровых возможностей.

# Список литературы

1. Были взяты определения[Электронный ресурс]. - https://ru.wikipedia.org/wiki/Заглавная\_страница
2. Большое количество шпаргалок для написания кода[Электронный ресурс]. - https://youtube.com
3. Источник готовых решений[Электронный ресурс]. - https://github.com
4. Основная информация по unity[Электронный ресурс]. - https://unity.com

# Приложение А. Листинг джойстика игры

using System;

using UnityEngine;

using UnityEngine.EventSystems;

namespace UnityStandardAssets.CrossPlatformInput

{

public class Joystick : MonoBehaviour, IPointerDownHandler, IPointerUpHandler, IDragHandler

{

public enum AxisOption

{

// Options for which axes to use

Both, // Use both

OnlyHorizontal, // Only horizontal

OnlyVertical // Only vertical

}

public int MovementRange = 100;

public AxisOption axesToUse = AxisOption.Both; // The options for the axes that the still will use

public string horizontalAxisName = "Horizontal"; // The name given to the horizontal axis for the cross platform input

public string verticalAxisName = "Vertical"; // The name given to the vertical axis for the cross platform input

Vector3 m\_StartPos;

bool m\_UseX; // Toggle for using the x axis

bool m\_UseY; // Toggle for using the Y axis

CrossPlatformInputManager.VirtualAxis m\_HorizontalVirtualAxis; // Reference to the joystick in the cross platform input

CrossPlatformInputManager.VirtualAxis m\_VerticalVirtualAxis; // Reference to the joystick in the cross platform input

void Start()

{

m\_StartPos = transform.position;

CreateVirtualAxes();

}

void UpdateVirtualAxes(Vector3 value)

{

var delta = m\_StartPos - value;

delta.y = -delta.y;

delta /= MovementRange;

if (m\_UseX)

{

m\_HorizontalVirtualAxis.Update(-delta.x);

}

if (m\_UseY)

{

m\_VerticalVirtualAxis.Update(delta.y);

}

}

void CreateVirtualAxes()

{

// set axes to use

m\_UseX = (axesToUse == AxisOption.Both || axesToUse == AxisOption.OnlyHorizontal);

m\_UseY = (axesToUse == AxisOption.Both || axesToUse == AxisOption.OnlyVertical);

// create new axes based on axes to use

if (m\_UseX)

{

m\_HorizontalVirtualAxis = new CrossPlatformInputManager.VirtualAxis(horizontalAxisName);

CrossPlatformInputManager.RegisterVirtualAxis(m\_HorizontalVirtualAxis);

}

if (m\_UseY)

{

m\_VerticalVirtualAxis = new CrossPlatformInputManager.VirtualAxis(verticalAxisName);

CrossPlatformInputManager.RegisterVirtualAxis(m\_VerticalVirtualAxis);

}

}

public void OnDrag(PointerEventData data)

{

Vector3 newPos = Vector3.zero;

if (m\_UseX)

{

int delta = (int)(data.position.x - m\_StartPos.x);

// delta = Mathf.Clamp(delta, - MovementRange, MovementRange);

newPos.x = delta;

}

if (m\_UseY)

{

int delta = (int)(data.position.y - m\_StartPos.y);

// delta = Mathf.Clamp(delta, -MovementRange, MovementRange);

newPos.y = delta;

}

transform.position = Vector3.ClampMagnitude(new Vector3(newPos.x, newPos.y, newPos.z), MovementRange) + m\_StartPos;

UpdateVirtualAxes(transform.position);

}

public void OnPointerUp(PointerEventData data)

{

transform.position = m\_StartPos;

UpdateVirtualAxes(m\_StartPos);

}

public void OnPointerDown(PointerEventData data) { }

void OnDisable()

{

// remove the joysticks from the cross platform input

if (m\_UseX)

{

m\_HorizontalVirtualAxis.Remove();

}

if (m\_UseY)

{

m\_VerticalVirtualAxis.Remove();

}

}

}

}

# Приложение Б. Листинг спавнера врагов

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Spawner : MonoBehaviour {

Vector3 []vectors=new Vector3[10];

// Update is called once per frame

public GameObject player;

public GameObject zombie;

public float spawnTime = 3f;

// Use this for initialization

void Start () {

vectors[0] = new Vector3(-49.914f, 0.18f, 142.27f);

vectors[1] = new Vector3(-44.08f, 0.18f, 169.35f);

vectors[2] = new Vector3(-44.08f, 0.18f, 211.4f);

vectors[3] = new Vector3(2.5f, 0.18f, 246f);

vectors[4] = new Vector3(43.3f, 0.18f, 190.2f);

vectors[5] = new Vector3(43.3f, 0.18f, 141.44f);

vectors[6] = new Vector3(43.3f, 0.18f, 61.2f);

vectors[7] = new Vector3(-26.5f, 0.18f, 61.2f);

vectors[8] = new Vector3(-105.3f, 0.18f, 61.2f);

vectors[9] = new Vector3(-155.4f, 0.18f, 61.2f);

InvokeRepeating("SpawnBall", spawnTime, spawnTime);

}

// Update is called once per frame

void SpawnBall()

{

GameObject go=Instantiate(zombie, vectors[(int)(Random.value\*100)%10], Quaternion.identity);

go.GetComponent<ZombieControl>().go = player;

}

}

# Приложение В. Размещение на git

https://github.com/Kaltblotsin/labs\_3S/tree/main/tfp/kyrsach