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

ДОПУСТИТЬ К ЗАЩИТЕ

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

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка игры в жанре «Платформер» на платформе Unity

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-432.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.
_____ И.И. Клебанов

Автор работы,
студент группы КЭ-402
_____ А.М. Шелакин

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

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

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-402
Шелакину Анатолию Михайловичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 746-13/12) Разработка игры в жанре «Платформер» на платформе Unity.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Хокинг Дж. Unity in Action: Разработка многоплатформенных игр на C#. // М.: ДМК Пресс, 2018. – 376 с.
3.2. Джексон С. Овладение разработкой 2D-игр в Unity. – М.: Питер, 2015. // 400 с.
3.3. Ферроне Х. Изучение C# через разработку игр на Unity. // М.: Питер, 2016. – 448 с.

4. **Перечень подлежащих разработке вопросов**

4.1. Провести анализ предметной области и обзор аналогов.
4.2. Спроектировать приложение.
4.3. Реализовать и протестировать приложение.

5. **Дата выдачи задания:** 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н. И.И. Клебанов

Задание принял к исполнению А.М. Шелакин

ГЛОССАРИЙ

1. *2D-Платформер* – жанр компьютерных игр, основная задача игрока в котором – пройти уровни, преодолевая препятствия, передвигаясь по разным платформам, благодаря чему жанр и получил данное название [1].
2. *Unity* – игровой движок, обладающий широким спектром для интеграции, создания, поддержки и развития игровых продуктов, которые могут использоваться на любом этапе разработки продукта [2].
3. *Геймплей* – совокупность задач, которые игра ставит перед игроком, и действий, которые игрок может выполнять в игре [3].
4. *Ассет* – набор ресурсов, которые используются в игре. В эту категорию может входить все что угодно, но обычно под ними имеются в виду готовые программные модули, спрайты, модели, анимации, звуки [4].
5. *Префаб* – тип ассета, позволяющий хранить все компоненты и значения свойств игрового объекта, а также создавать его экземпляры. При этом экземпляры копируют свойства оригинального ассета [5, 6].
6. *Спрайт* – графический 2D-объект (изображения, картинки) [7].
7. *Скрипт* – программа, которая содержит последовательность действий, созданных для автоматического выполнения задачи [8].
8. *Триггер* – механизм, проверяющий присутствие каких-либо объектов игрового мира в заданном пространстве или расстояние от этих объектов до специальной точки [9].
9. *Коллизия* – столкновение объектов в пространстве [10].

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Разработка компьютерной игры может быть актуальной для дипломной работы как с точки зрения актуальности для индустрии, так и с точки зрения приобретения новых навыков и возможности творчества.

Игры становятся все более популярными и приносят значительные доходы, поэтому исследование и разработка игр могут быть актуальными для студентов, интересующихся этой областью.

Разработка компьютерной игры требует знания различных технологий, программирования, дизайна и других навыков, которые могут быть полезны для студента в будущей карьере.

Разработка компьютерной игры может быть использована для исследования различных аспектов, таких как взаимодействие пользователя с интерфейсом, психологические аспекты игрового процесса и другие.

Постановка задачи

Целью выпускной квалификационной работы является разработка компьютерной игры. Для достижения поставленной цели необходимо решить следующие задачи.

1. Анализ предметной области проекта: изучить игры в схожем жанре и средства для реализации игры.
2. Проектирование игры: разработать дизайн игры, включая игровую механику, уровни, интерфейс пользователя, архитектуру игрового движка и другие аспекты.
3. Реализация игрового приложения.
4. Тестирование и отладка: провести тестирование игры, выявить и исправить ошибки и недочеты.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 42 страницы, объем списка литературы – 15 источников.

В глоссарии даны определения важным терминам, характерным для работы в среде разработки Unity.

В первой главе проводится анализ предметной области проекта, анализ аналогичных проектов в этой области и анализ существующих решений для реализации проекта.

Вторая глава посвящена описанию требований к разрабатываемой системе, а также производится проектирование архитектуры разрабатываемой системы. Были произведены общее описания архитектуры системы и описание компонентов, составляющих эту систему, а также разработаны макеты пользовательских интерфейсов.

Третья глава содержит реализацию основных аспектов игрового приложения с демонстрацией основных функций и механик игрового приложения.

В четвертой главе приводятся описания и результаты функционального и юзабилити тестирований, по результатам которых были произведены исправления и доработки в итоговый проект.

В приложении А содержатся таблицы, описывающие основные варианты использования игрового приложения.

В приложении Б содержатся скриншоты финальной версии игры, демонстрирующие меню и игровой процесс.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Предметная область проекта

В данной работе необходимо разработать двухмерную компьютерную игру в жанре «Платформер». «Платформер» – жанр игры, суть которого заключается в прохождении игроком уровней, состоящих из платформ, этажам или другим объектам, а также препятствий и противников (при их наличии).

Главной задачей в играх этого жанра является прохождение игроком от начальной точки уровня (уровней может быть несколько) до конечной. Процесс может дополняться собиранием различных предметов для совершенствования персонажа, преодолением разнообразных ловушек и битвами с противниками.

Данный жанр видеоигр не стоит на месте и, как и прочие жанры в индустрии, активно развивается. Разработчики пытаются внести что-то новое в уже привычный игровой процесс: различные механики, переплетение игрового процесса с другими жанрами, уникальный визуальный/аудиальный стиль.

Сложность разрабатываемой игры заключается в требовании к правильности и точности выполнения игроком своих действий, избеганием препятствий и уничтожением противников. Далее приводится анализ аналогичных проектов в этой предметной области.

1.2. Анализ аналогичных проектов

Перед, непосредственно, началом работы необходимо провести анализ уже существующих игр данного жанра. Путем выделения уникальных решений таких проектов, этот этап поможет в реализации собственного игрового приложения.

В роли аналогов были рассмотрены следующие игры: Celeste, Contra и Super Mario Bros.

Celeste [11]

Celeste – сюжетная игра в жанре «Платформер», выпущенная в 2018 году. В игре нет противников и, как следствие, боевой системы. Этот проект относится к поджанру «precision-платформер», игры которого делают упор на передвижение, точные прыжки и избегание опасностей, а не на бой. Персонаж в Celeste имеет весьма разнообразный набор способностей для передвижения, которые меняются из локации в локацию, благодаря чему игроку не надоедает проходить уровни в игре. Скриншот игрового процесса представлен на рисунке 1.

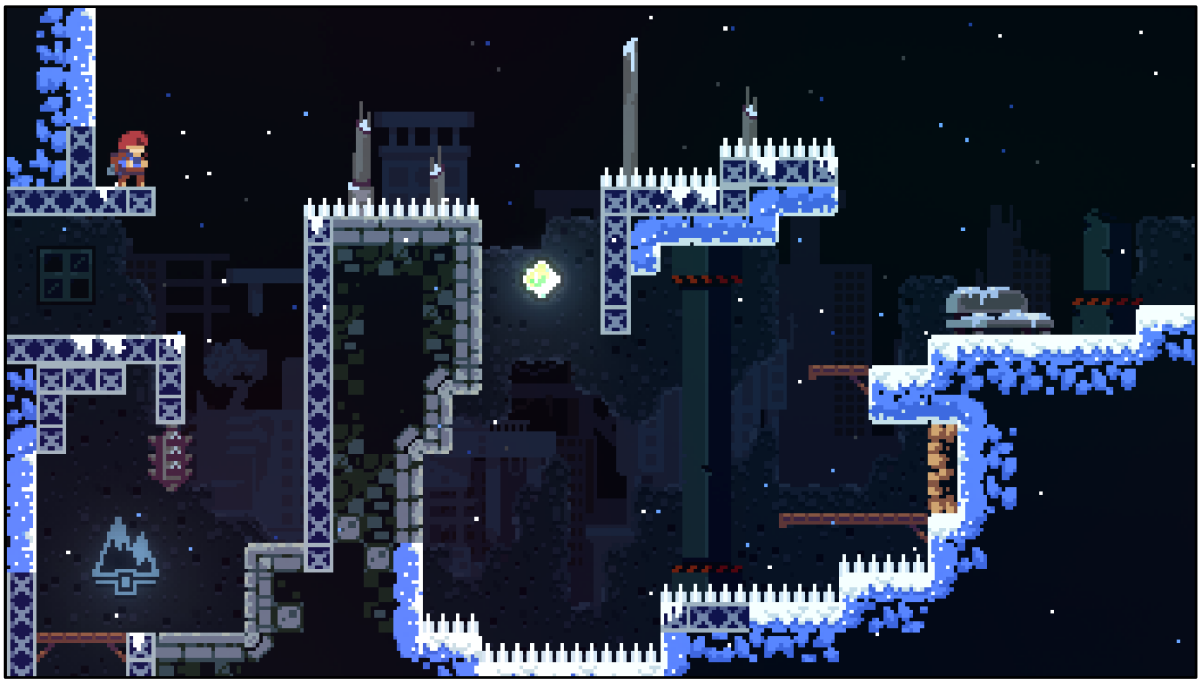

Рисунок 1 – Игра «Celeste»

Contra

Contra – это культовая видеоигра в жанре «action-платформер», которая стала популярной благодаря своему динамичному геймплею, высокому уровню сложности и качественной графике для того времени. Игра имеет разнообразное оружие и бонусы, которые можно собирать на уровнях для улучшения атакующих возможностей персонажей, локации и врагов, создающих разнообразие и динамику в игровом процессе и несколько уровней сложности, от легкого до экстремально сложного, что делает игру вызовом

даже для опытных игроков. Contra продолжает оставаться популярной и в настоящее время благодаря своей классической механике и ностальгическому шарму. Скриншот игры представлен на рисунке 2.

Рисунок 2 – Игра «Contra»

Super Mario Bros [12]

Super Mario Bros. – игра, внесшая большой вклад как в развитие игровой индустрии в целом, так и в разработку платформеров в частности. Эта игра – двухмерный платформер, выпущенный в 1985 году. Игровой процесс заключается в прохождении уровней, путем прыжков по платформам, поиском секретов и противостоянию противникам, единственным средством к которому является прыжок на противника сверху. Дополнительную сложность в прохождении игры добавляет инерция персонажа его постепенный разгон, что создает не самый точный контроль игроком своего игрового аватара. Игровой процесс представлен на рисунке 3.

Рисунок 3 – Игра «Super Mario Bros»

1.3. Анализ существующих решений для реализации проекта

В настоящее время существует множество платформ для разработки компьютерных игр. Они значительно упрощают процесс разработки приложения из-за экономии времени и трудовых ресурсов. В данном разделе будет произведен анализ различных игровых движков, доступных для пользования каждому желающему. Среди них будет выбрано наиболее подходящее для разработки решение.

Godot Engine [13]

Godot Engine – полностью бесплатный открытый кроссплатформенный движок. Основным языком программирования в Godot является GDScript, написанный, непосредственно разработчиками движка. Также имеется поддержка C#, однако разработчик рекомендует использовать именно собственный язык для большей стабильности. Godot предоставляет различные инструменты для разработки платформеров, включая:

1) уникальную узловую систему для создания сцен, что делает управление объектами и компонентами игры более удобным и гибким;

2) встроенный физический движок, который обеспечивает поддержку коллизий, гравитации, динамики объектов и других физических эффектов;

3) мощную систему анимации, которая позволяет создавать анимированные персонажи, объекты и эффекты;

4) простую обработку ввода с клавиатуры, мыши и геймпадов, что позволяет легко управлять персонажем в игре.

Также Godot имеет качественную собственную документацию, в которой есть примеры кода как на собственном языке, так и на C#. Однако из-за недостаточного количества обучающих материалов (как от разработчиков, так и от сообщества) выбор не был сделан в пользу Godot Engine.

GameMaker Studio [14]

GameMaker – один из самых популярных игровых движков, позволяющий разрабатывать приложения под множество платформ. Бесплатная версия (Free) позволяет создавать игры только под игровую платформу Opera GX.games. По сравнению с ней, Creator версия имеет множество преимуществ, включая управление ресурсами и компиляцию для настольных компьютеров. Также, в Creator версии можно покупать отдельные модули, расширяющие функциональность программы. GameMaker Studio предоставляет разработчикам множество инструментов для создания платформеров. Вот некоторые из основных инструментов, которые могут быть полезны при разработке платформеров в GameMaker Studio:

1) удобный интерфейс для создания игр без написания кода, а также язык программирования GML (GameMaker Language) для создания более сложной логики игры;

2) встроенный редактор спрайтов и инструменты анимации, которые упрощают создание и анимирование персонажей, объекты и фоны;

3) инструменты для интеграции звуков и музыки, что позволяет создавать атмосферные звуковые эффекты;

4) удобный редактор помещений, который позволяет создавать и управлять различными уровнями и сценами.

Движок однозначно предоставляет хороший набор инструментов для создания собственных проектов (особенно для новичков в разработке игр), однако, из-за недостаточного базового функционала у бесплатной версии, не подходит для использования при разработке.

Unity 3D [15]

Unity 3D – инструмент для создания 2D и 3D игр, а также интерактивного контента. В составе версий имеет Community Edition (бесплатная), Professional Edition и решение для студий, в котором обещана специальная поддержка. В Community Edition включены все функции и большинство платформ, под которые разрабатывают приложения. Unity предоставляет различные инструменты для разработки платформеров, включая:

1) редактор сцен, который позволяет создавать уровни и настраивать их параметры;

2) компоненты физики, которые обеспечивают реалистичное поведение персонажей и объектов;

3) систему анимации, которая позволяет создавать анимированные персонажи и объекты;

4) систему частиц, которая позволяет создавать эффекты взрывов, дыма и других спецэффектов;

5) инструменты для работы со звуком, включая возможность добавления звуковых эффектов и музыки;

6) систему скриптинга, которая позволяет написать собственные скрипты для управления поведением персонажей и объектов;

7) различные готовые ресурсы, такие как модели персонажей, текстуры и звуковые эффекты, которые можно использовать в своих проектах;

8) система управления персонажем, включая возможность настройки управления с клавиатуры, геймпада и сенсорного экрана;

9) встроенные компоненты для обработки коллизий, триггеров и других физических взаимодействий;

10) инструменты для оптимизации производительности игры, включая возможность управления ресурсами и оптимизации кода.

Вывод по первой главе

На данном этапе разработки игрового приложения был произведен анализ аналогичных проектов в области разрабатываемой системы, а также рассмотрение решений для реализации проекта.

Исходя из полученных данных, можно прийти к следующим выводам. Сутью игрового цикла, в своей основе, является полное прохождение уровней с избеганием имеющихся препятствий и противодействию противникам. Сами препятствия должны быть интересными и бросать игроку вызов к их преодолению. Визуальный стиль должен быть красочным и целостным, поскольку именно внешняя оболочка игрового приложения в первую очередь привлекает игроков. Аудиальный стиль должен быть ненавязчивым и соответствовать выбранной стилистике.

Исходя из рассмотренных решений для реализации проекта, можно прийти к тому, что игровой движок Unity 3D является наиболее подходящей платформой для разработки игрового приложения при условии недостаточного опыта в сфере создания компьютерных игр.

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1. Требования к проектируемой системе

В ходе проектирования были сформулированные следующие функциональные и нефункциональные требования к разрабатываемому приложению.

Функциональные требования проектируемой системы

Ниже представлены функциональные требования.

1. Игрок должен иметь возможность управлять персонажем, перемещая его по уровню и выполняя различные действия.
2. На каждом уровне должны быть различные препятствия, которые игрок должен преодолеть.
3. Противники в игре должны иметь искусственный интеллект, чтобы противодействовать игроку.
4. Игра должна иметь систему сохранения прогресса, чтобы игрок мог вернуться к прохождению уровня позже.
5. Игра должна иметь красочную графику и звуковое сопровождение, чтобы создать атмосферу и увлекательность игры.

Нефункциональные требования к проектируемой системе

Ниже представлены нефункциональные требования.

1. Игра должна иметь быструю скорость загрузки, чтобы игроки не теряли интерес из-за долгого ожидания.
2. Игра должна иметь интуитивно понятный и простой интерфейс, чтобы игроки могли быстро освоить управление и начать играть.
3. Игра должна иметь возможность регулирования громкости звукового сопровождения и музыки.
4. Игра должна иметь несколько уровней, каждый из которых состоит из некоторого количества комнат (экранов), спроектированных вручную.

2.2. Диаграмма вариантов использования

Исходя из составленных функциональных требований была составлена модель взаимодействия внешнего актера с разрабатываемым приложением. Модель представлена в виде диаграммы вариантов использования, изображенной на рисунке 4.

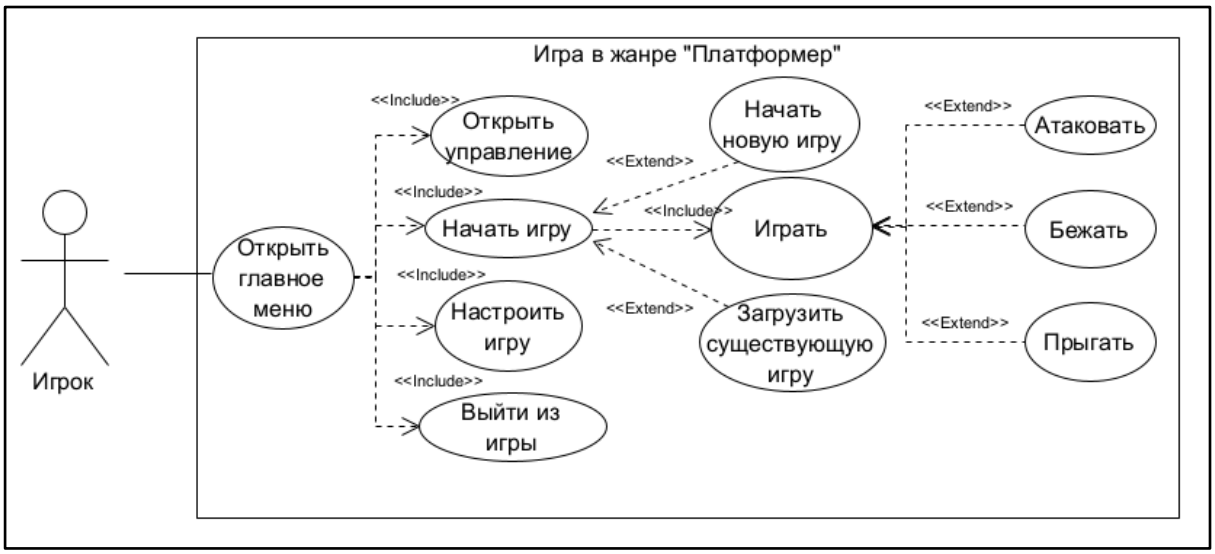

Рисунок 4 – Диаграмма вариантов использования

С системой может взаимодействовать лишь один актер – игрок. Взаимодействие происходит через совершение следующих действий.

1. «Открыть главное меню» – при запуске игрового приложения главное меню запускается автоматически.
2. «Открыть управление» – посмотреть все варианты управления игровым персонажем.
3. «Открыть настройки» – открыть отдельное меню настроек, в котором можно будет изменять уровень громкости внутриигровых звуков и музыки.
4. «Выйти из игры» – завершить работу приложения.
5. «Начать игру» – запуск игрового процесса.
6. «Начать новую игру» – игровой процесс запускается с первого уровня.

7. «Загрузить существующую игру» – открыть меню с выбором любого из уже открытых уровней.

8. «Управлять персонажем» – непосредственно основа игрового процесса.

9. «Атаковать» – игровой персонаж производит выстрелы из активированного на данный момент оружия.

10. «Бежать» – перемещаться по уровню путем передвижения по земле или платформе.

11. «Прыгать» – совершить прыжок.

Спецификации основных вариантов использования приведены в приложении А.

2.3. Архитектура системы

На рисунке 5 представлена диаграмма компонентов игрового приложения. Данные компоненты содержат в себе классы, описывающие поведение, свойства, параметры и зависимости игровых объектов.

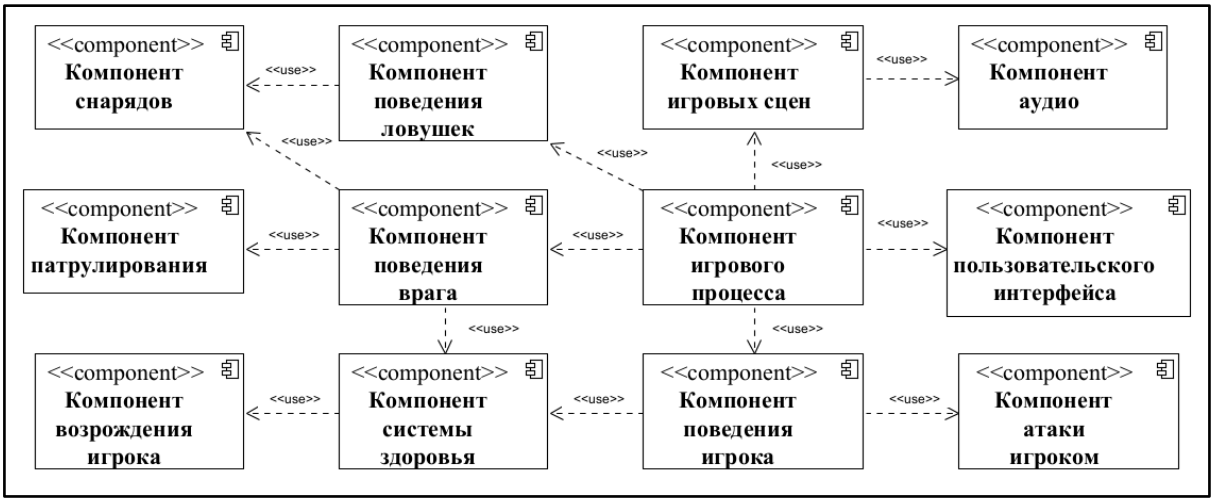

Рисунок 5 – Диаграмма компонентов

Проектируемая система содержит следующие компоненты.

1. Компонент снарядов содержит классы, реализующие поведение снарядов игрока и вражеских снарядов (ловушек и противников).

2. Компонент патрулирования содержит класс, реализующий перемещение противников от крайней левой точки до крайней правой точки.

3. Компонент поведения врага содержит набор классов, реализующих анимацию противников и их реакции на игрового персонажа.

4. Компонент аудио содержит класс, реализующий воспроизведение фоновой музыки и звуковых эффектов.

5. Компонент игровых сцен содержит класс, реализующий корректное переключением между сценами уровней и сценой главного меню.

6. Компонент пользовательского интерфейса содержит набор классов, реализующих поведение главного меню, меню паузы и меню конца игры.

7. Компонент поведения ловушек содержит набор классов, реализующих работу всех имеющихся в игре ловушек.

8. Компонент атаки игроком содержит класс, реализующий боевые способности игрового персонажа.

9. Компонент возрождения игрока содержит набор классов, реализующих работу системы сохранений на конкретном уровне.

10. Компонент системы здоровья содержит класс, реализующий работу системы здоровья как у игрового персонажа, так и у неигровых персонажей (противников).

11. Компонент поведения игрока содержит набор классов, реализующих управление игровым персонажем и его поведение при различных игровых обстоятельствах.

12. Компонент игрового процесса содержит набор классов, реализующих корректную работу, непосредственно, игрового процесса.

2.4. Макеты интерфейса

При запуске игры загружается сцена главного меню (рисунок 6), которое содержит кнопки «Load», «New Game», «Settings», «Help» и «Quit». При нажатии на кнопку «New Game» загружается сцена с первым уровнем.

При нажатии на кнопку «Load Game» открывается меню выбора уровня (рисунок 7). При нажатии на кнопку «Settings» открывается меню настроек звука (рисунок 8). При нажатии на кнопку «Help» открывается окно с информацией об управлении (рисунок 9). При нажатии на кнопку «Quit» игра закрывается. Также пользователь в каждом из второстепенных меню может вернуться в главное меню посредством нажатия на кнопку «Back».

Рисунок 6 – Макет интерфейса главного меню

В меню выбора уровня пользователь может выбрать любой из доступных уровней, которые открываются по мере прохождения игры.

Рисунок 7 – Макет интерфейса меню выбора уровня

Рисунок 8 – Макет интерфейса меню настроек звука

В окне с информацией об управлении представлена раскладка управления персонажем. Бег осуществляется через кнопки «A» и «D» либо стрелочками, атака осуществляется нажатием левой кнопки мыши, а прыжок происходит после нажатия кнопки «Пробел».

Рисунок 9 – Макет окна с информацией об управлении

Находясь на игровом уровне, игрок имеет возможность открыть меню паузы (рисунок 10) нажатием на кнопку «Esc». Это меню ставит игру на паузу и содержит кнопки «Resume», «Music», «Sounds», «Main Menu» и «Quit».

При нажатии на кнопку «Resume» игра возвращает игрока в сцену игрового поля. При нажатии на «Music» уменьшается громкость внутриигровой музыки. При нажатии на «Sounds» уменьшается громкость внутриигровых эффектов (звуков прыжков, стрельбы, ранений и т.д.). При нажатии на «Main Menu» игрок выйдет в главное меню. При нажатии на «Quit» завершается игра и закрывается игровое приложение.

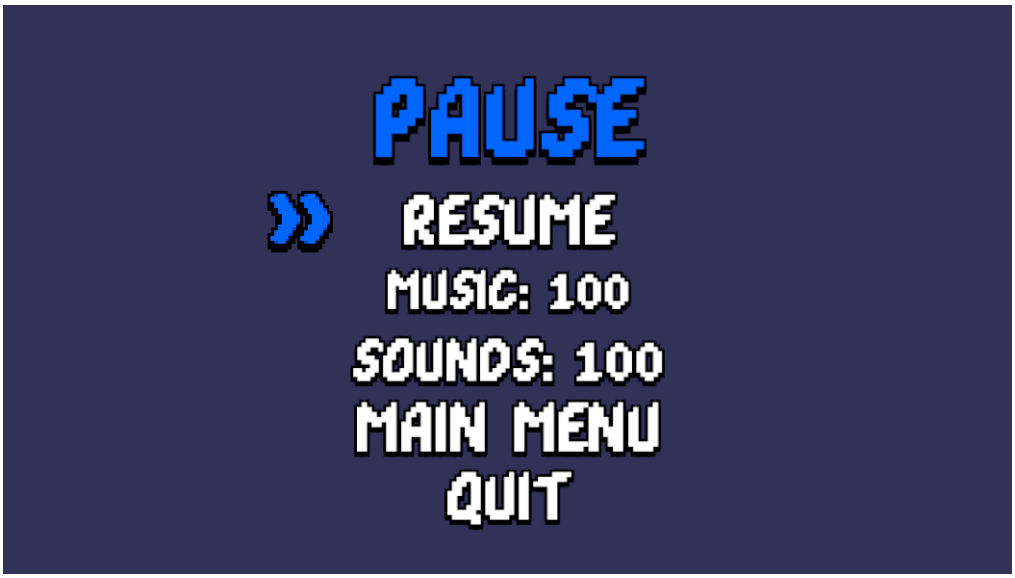

Рисунок 10 – Макет интерфейса меню паузы

Также, находясь на игровом уровне у игрового персонажа могут закончиться очки здоровья и количество доступных возрождений, что приведет к концу игры и соответствующему меню (рисунок 11). В меню конца игры содержатся кнопки «Restart», «Main Menu» и «Quit». При нажатии на кнопку «Restart» заново загружается сцена с текущим уровнем. При нажатии на кнопку «Main Menu» загружается сцена с главным меню. При нажатии на кнопку «Quit» игра закрывается.

Рисунок 11 – Макет интерфейса меню конца игры

Игровой интерфейс (рисунок 12) представлен в виде трех показателей. Шкала здоровья отображает текущее количество очков здоровья. Под шкалой здоровья отображается количество жизней. Далее отображаются предметы, которые игрок собрал на протяжении текущего игрового сеанса.

Рисунок 12 – Макет игрового интерфейса

Вывод по второй главе

На данном этапе разработки игрового приложения для представления сформированной архитектуры системы были составлены диаграмма вариантов использования, диаграмма компонентов и диаграмма деятельности. Для пользовательского интерфейса были составлены соответствующие макеты.

3. РЕАЛИЗАЦИЯ

Для реализации компьютерной 2D игры была выбрана платформа Unity. Она обладает всеми необходимым инструментарием для создания 2D игр, а также поддержкой широкого круга платформ, подробной официальной документацией и официальными уроками, а также материалами по разработке продуктов.

Инструменты Unity, которые использовались для реализации игрового приложения, представлены ниже.

1. Animator – инструмент для создания спрайтовых анимаций, и привязки к ним определенных скриптовых методов.

2. Tilemap – инструмент, с помощью которого можно довольно быстро создавать двухмерные уровни путем выставления спрайтов по установленной сетке.

3. Box Collider 2D – компонент, позволяющий реализовать пересечение и столкновение объектов на сцене.

4. Rigidbody 2D – компонент, реализующий поведение объектов согласно физике.

5. Physics Material 2D – компонент, позволяющий задать поверхностям свойства физических объектов.

Для реализации скриптов игры использовался язык программирования C#. Для написания и отладки кода использовалась интегрированная среда программирования Visual Studio 2022 Community Edition.

Также для поиска графических элементов, спрайтов, персонажей и фонов были использованы различные интернет-ресурсы и собственный магазин ассетов Unity, из которых были взяты бесплатные графические элементы, которые могут быть свободно использованы без нарушения авторских прав.

3.1. Реализация персонажа

Передвижение игрового аватара осуществляется при помощи клавиш A и D на клавиатуре, а прыжок персонажа осуществляется с помощью клавиши Space в том случае, если персонаж либо стоит на поверхности, либо у него есть возможность сделать дополнительный прыжок (появляется при подборе предмета на двойной прыжок), либо пока не истек таймер буферизованного прыжка. Буфер прыжка используется для более отзывчивого прыжка с края любой платформы. Реализация передвижения, прыжка и метода определяющего, стоит ли игровой персонаж на земле, приведена в листинге 1.

Листинг 1 – Реализация передвижения игрового персонажа

```
private void Move(float input)
{
    body.velocity = new Vector2(horizontalInput * speed, body.velocity.y);
    if (input>0) transform.localScale = new Vector3(scale, scale, scale);
    else if(input<0) transform.localScale=new Vector3(-scale,scale,scale);
    animator.SetBool("isRunning", input != 0);
    animator.SetBool("isGrounded", isGrounded());
}

private void Jump()
{
    if (coyoteCounter <= 0 && jumpCounter <= 0) return;
    SoundManager.instance.PlaySound(jumpSound);
    if (isGrounded())
    {
        body.velocity = new Vector2(body.velocity.x, jumpPower);
    }
    else
    {
        if (coyoteCounter > 0)
        {
            body.velocity = new Vector2(body.velocity.x, jumpPower);
        }
        else if (jumpCounter > 0)
        {
            body.velocity = new Vector2(body.velocity.x, jumpPower);
            jumpCounter--;
        }
    }
    coyoteCounter = 0;
}

public bool isGrounded()
{
    return Physics2D.BoxCast(transform.position, boxSize, 0,
                    -transform.up, castDistanceY, groundLayer); ;
}
```

Реализация механики здоровья состоит из функции получения урона (представлена в листинге 2), функции неуязвимости после получения урона (представлена в листинге 3) и функции возрождения (листинг 4).

Листинг 2 – Реализация механики получения урона

```
public void TakeDamage(float _damage)
{
    if (isInvulnerable) return;
    currentHealth = Mathf.Clamp(currentHealth - _damage, 0, startHealth);
    if (currentHealth > 0)
    {
        StartCoroutine(Invunerabilty());
        SoundManager.instance.PlaySound(hurtSound);
    }
    else
    {
        if (!dead)
        {
            foreach (Behaviour component in components)
                component.enabled = false;
            gameObject.layer = 0;
            animator.SetTrigger("dead");
            animator.SetBool("isGrounded", true);
            dead = true;
            SoundManager.instance.PlaySound(deathSound);
        }
    }
}
```

Неуязвимость после получения урона реализована путем ожидания назначенного времени неуязвимости и мигания спрайта игрока красным цветом.

Листинг 3 – Реализация механики неуязвимости

```
private IEnumerator Invunerabilty()
{
    isInvulnerable = true;
    Physics2D.IgnoreLayerCollision(8, 9, true);
    for (int i = 0; i < numberOfFlashes; i++)
    {
        spriteRenderer.color = Color.red;
        yield return new
            WaitForSeconds(iFramesDuration / (numberOfFlashes * 2));
        spriteRenderer.color = Color.white;
        yield return new
            WaitForSeconds(iFramesDuration / (numberOfFlashes * 2));
    }
    Physics2D.IgnoreLayerCollision(8, 9, false);
    isInvulnerable = false;
}
```

Листинг 4 – Реализация возрождения

```
public void Respawn()
{
    dead = false;
    AddHealth(startHealth);
    gameObject.layer = 8;
    animator.ResetTrigger("dead");
    animator.Play("Idle");
    StartCoroutine(Invunerabilty());
    foreach (Behaviour component in components)
        component.enabled = true;
}
```

3.2. Реализация стрельбы

Стрельба осуществляется путем нажатия на левую кнопку мыши. В целях оптимизации ресурсов, множественные снаряды реализованы методом пула объектов. Метод заключается в том, что, вместо множественного создания нового объекта для каждого снаряда, существует заготовленный деактивированный набор снарядов, которые циклично активируются и снова деактивируются по истечении времени жизни объекта. Реализация стрельбы игрового персонажа (листинг 5) и реализация поведения пули (листинг 6) представлены ниже.

Листинг 5 – Реализация стрельбы

```
private void Attack()
{
    SoundManager.instance.PlaySound(shotSound);
    cooldownTimer = 0;
    bullets[FindBullet()].transform.position = firePoint.position;
    bullets[FindBullet()].GetComponent<Projectile>().
        SetDirection(Mathf.Sign(playerMovement.transform.localScale.x) *
                                    transform.localScale.x);
}

private int FindBullet()
{
    for (int i = 0; i < bullets.Length; i++)
    {
        if (!bullets[i].activeInHierarchy) return i;
    }
    return 0;
}
```

Листинг 6 – Реализация поведения пули

```
private void Update()
{
    if (hit) return;
    float movementSpeed = speed * Time.deltaTime * direction;
    transform.Translate(movementSpeed, 0, 0);
    lifetimeCooldown += Time.deltaTime;
    if(lifetimeCooldown > lifetime)
        gameObject.SetActive(false);
}

private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.transform.tag != "Checkpoint" &&
        collision.transform.tag != "Level Trigger" &&
        collision.transform.tag != "Bullet")
    {
        hit = true;
        boxCollider.enabled = false;
        animator.SetTrigger("explode");
    }
    if (collision.tag == "Enemy")
        collision.GetComponent<Health>().TakeDamage(damage);
}

public void SetDirection(float _direction)
{
    lifetimeCooldown = 0;
    direction = _direction;
    gameObject.SetActive(true);
    hit = false;
    boxCollider.enabled = true;
    float localScaleX = transform.localScale.x;
    if(Mathf.Sign(localScaleX) != _direction)
        localScaleX = -localScaleX;
    transform.localScale = new
        Vector3(localScaleX, transform.localScale.y,
                transform.localScale.z);
}
```

3.3. Реализация поведения противников

В игре есть два вида противников: атакующие в ближнем бою и атакующие издалека. Поведение противников реализовано так, что, при попадании игрового персонажа в их поле зрения, противник производит атаку в конкретный момент своей анимации атаки. Также, помимо поведения самих противников, реализована механика их патрулирования. Реализация определения игрового персонажа в поле зрения (листинг 7) и реализация патрулирования (листинг 8) представлены ниже.

Листинг 7 – Реализация определения игрока в поле зрения противника

```
private bool PlayerInSight()
{
    RaycastHit2D hit = Physics2D.BoxCast(boxCollider.bounds.center +
            transform.right * range * transform.localScale.x *
            colliderDistance,
        new Vector3(boxCollider.bounds.size.x * range,
        boxCollider.bounds.size.y, boxCollider.bounds.size.z),
        0, Vector2.left, 0, playerLayer);
    if(hit.collider != null)
        playerHealth = hit.transform.GetComponent<Health>();
    return hit.collider != null;
}
```

Листинг 8 – Реализация патрулирования

```
private void Update()
{
    if (movingLeft)
    {
        if(enemy.position.x >= leftEdge.position.x) MoveInDirection(-1);
        else DirectionChange();
    }
    else
    {
        if (enemy.position.x <= rightEdge.position.x) MoveInDirection(1);
        else DirectionChange();
    }
}

private void DirectionChange()
{
    animator.SetBool("isMoving", false);
    idleTimer += Time.deltaTime;
    if(idleTimer > idleDuration) movingLeft = !movingLeft;
}

private void MoveInDirection(int _direction)
{
    idleTimer = 0;
    animator.SetBool("isMoving", true);
    enemy.localScale = new Vector3(Mathf.Abs(initScale.x) * _direction,
                                   initScale.y, initScale.z);
    enemy.position = new Vector3(enemy.position.x + Time.deltaTime *
            _direction * speed, enemy.position.y, enemy.position.z);
}
```

3.4. Реализация звукового сопровождения

Работа со звуком производится за счет компонента `AudioSource`, который управляет всем звуком, воспроизводимым игровым приложением. Реализация метода изменения громкости источника (звуки или музыка) представлена в листинге 9.

Листинг 9 – Реализация аудио сопровождения

```
private void ChangeSourceVolume(float _change, string _volumeName,
    AudioSource _source)
{
    float currentVolume = PlayerPrefs.GetFloat(_volumeName);
    currentVolume += _change;
    if (currentVolume > 1) currentVolume = 0;
    else if (currentVolume < 0) currentVolume = 1;
    _source.volume = currentVolume;
    PlayerPrefs.SetFloat(_volumeName, currentVolume);
}
```

3.5. Реализация камеры

Камера реализована с помощью объекта `CameraController`. Камера прикреплена к комнате, в которой находится игрок и перемещается тогда, когда он переходит в другую комнату. Реализация камеры представлена в листинге 10.

Листинг 10 – Реализация камеры

```
public class CameraController : MonoBehaviour
{
    [Header("Скорость перемещения камеры")]
    [SerializeField] private float speed;
    private float currentPosX;
    private Vector3 velocity = Vector3.zero;
    private void Update()
    {
        transform.position = Vector3.SmoothDamp(transform.position,
            new Vector3(currentPosX, transform.position.y,
                  transform.position.z),
            ref velocity, speed);
    }
    public void MoveToNewRoom(Transform _newRoom)
    {
        currentPosX = _newRoom.position.x;
    }
}
```

Вывод по третьей главе

На данном этапе, в соответствии с заранее определенными требованиями, была реализована двухмерная игра в жанре «Платформер». Была описана реализация основных классов и объектов с приведением исходного кода игры.

4. ТЕСТИРОВАНИЕ

4.1. Функциональное тестирование

В ходе данного тестирования проверялось соответствие реализованного игрового приложения на соответствие сформулированным функциональным требованиям. Данные тесты проверяют работу базовых механик игры и взаимодействие между компонентами системы. Результаты тестирования представлены в таблице 1.

Таблица 1 – Функциональное тестирование игрового приложения

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Проверка запуска приложения.	Запуск игрового приложения.	Приложение запускается.	Да
2	Проверка работоспособности главного меню.	Поочередное нажатие всех имеющихся в главном меню кнопок.	Каждая кнопка выполняет свою задачу.	Да
3	Проверка работы системы передвижения.	Передвижения персонажа путем нажатия кнопок «A», «D», «Пробел» и кнопок с горизонтальными стрелками.	Персонаж двигается согласно нажатым кнопкам.	Да
4	Проверка корректности поведения игрового интерфейса.	Получить урон; Подобрать предмет.	При получении урона происходит соответствующее изменение на шкале здоровья; При подборе предмета он начинает отображать под шкалой здоровья.	Да
5	Проверка работоспособности внутриигровых меню.	Нажать кнопку «Esc»; Получить игровым персонажем летальный урон; Попеременное нажатие имеющихся кнопок в отображенном меню.	При нажатии «Esc» становится активным меню паузы; При смерти игрового персонажа становится активным меню конца игры. Каждая кнопка в меню выполняет свою задачу.	Да
6	Проверка работы стрельбы.	Нажать левую кнопку мыши.	Персонаж производит выстрел.	Да
7	Проверка получения урона игровым персонажем.	Подойти вплотную к противнику ближнего боя; Попасть под выстрел противника дальнего боя.	Персонаж получается соответствующий урон.	Да

Окончание таблицы 1

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
8	Проверка получения урона противниками.	Попадание выстрелом в противника.	Противник получил урон.	Да
9	Проверка корректного поведения камеры.	Переход из одной комнаты в другую.	Камера переместилась на положение новой комнаты.	Да
10	Проверка системы сохранения.	Активировать точку сохранения; Получить игровым персонажем летальный урон.	Точка сохранения становится активной; Персонаж возродился на точке сохранения с уменьшенным количеством жизней.	Да
11	Проверка подбора объектов игровым персонажем.	Подобрать игровым персонажем предмет на уровне; Использовать подобранный предмет.	Предмет пропадает с уровня и становится активным в инвентаре игрока; Персонаж выполняет действие в соответствии с подобранным предметом.	Да

4.2. Юзабилити тестирование

Юзабилити тестирование проводилось среди друзей и членов семьи. Всего в тестировании приняли участие пять человек. В ходе данного тестирования были обнаружены неочевидные при функциональном тестировании ошибки. Также были получены предложения по улучшению игры, предполагающие повторную балансировку игровых параметров. Ниже приведен список изменений игрового приложения после проведения юзабилити тестирования.

1. Игровая камера вела себя некорректно при остановке игрового персонажа на стыке между комнатами.

2. Ловушки не деактивировались на сцене в случае, если комната, в которой находится последняя точка сохранения отлична от той, в которой умер игровой персонаж.

3. Игровой персонаж не умирал при падении в пропасть.

4. При активации меню паузы или меню конца игровой процесс продолжал быть активен.

5. После победы над противником игрок все еще мог получить урон при соприкосновении с ним.

6. Противники продолжали атаковать игрового персонажа, который уже получил летальный урон, но не успел возродиться.

7. Игровой персонаж не мог допрыгнуть до определенной поверхности.

8. Игровой персонаж мог выполнить повторный прыжок в воздухе в том случае, если между интервал между нажатиями кнопки «Space» был незначительным.

9. Персонаж мог случайно застревать в какой-либо части ровной плоской поверхности.

Данные ошибки были устранены. При повторном тестировании ошибки не были обнаружены. В приложении Б представлены скриншоты итогового игрового приложения.

Вывод по четвертой главе

Были проведены функциональное и юзабилити тестирования. В ходе тестирований были обнаружены ошибки разной степени критичности, которые в последствии были устранены. Повторные тестирования показали, что все тестирования были пройдены.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано игровое приложение в жанре «Платформер» в среде разработки Unity. Система была разработана в соответствии со всеми сформулированными в ходе работы требованиями. В ходе работы были выполнены следующие задачи:

1) анализ аналогов;
2) анализ существующих средств для решения поставленной задачи;
3) формулировка требований к приложению;
4) проектирование приложения;
5) реализация приложения;
6) тестирование приложения;
7) обзор литературы по предметной области.

Результаты исследования показали, что создание игры на платформе Unity позволяет эффективно реализовывать игровые механики, обеспечивая высокое качество геймплея и визуального исполнения.

В процессе работы были изучены основные принципы разработки игр, оптимизации производительности и создания пользовательского интерфейса.

В будущем планируется дальнейшая разработка, расширение функционала, полировка и замена всех имеющихся внутриигровых ассетов (взятых из ресурсов готовых ассетов без авторских прав) на собственные.

Полученный опыт и знания могут быть использованы для дальнейшего совершенствования навыков разработки игр и создания увлекательных проектов в будущем.

ЛИТЕРАТУРА

1. Gamersgate – Платформеры. [Электронный ресурс] URL: https://gamersgate.ru/reviews/zhanr-kompyuternykh-igr-platfo/ (дата обращения: 25.05.2024 г.).
2. Unity. [Электронный ресурс] URL: https://unity.com/ru/pages/more-than-an-engine (дата обращения: 25.05.2024 г.).
3. Ernest Adams, Joris Dormans. Game Mechanics: Advanced Game Design – New Riders, 2012. – 353 с.
4. Как правильно использовать ассеты? [Электронный ресурс] URL: https://dtf.ru/s/superjam/237159-kak-pravilno-ispolzovat-assety (дата обращения: 06.02.2024 г.).
5. Unity User Manual. [Электронный ресурс] URL: https://docs.unity3d.com/ru/530/Manual/Prefabs.html (дата обращения: 25.05.2024 г.).
6. Unity User Manual. [Электронный ресурс] URL: https://unity.com/ru/prefabs (дата обращения: 25.05.2024 г.).
7. Что такое спрайт. [Электронный ресурс] URL: https://dic.academic.ru/dic.nsf/ruwiki/42140 (дата обращения: 05.02.2024 г.).
8. Что такое скрипт. [Электронный ресурс] URL: https://ru.hostings.info/termins/chto–takoe–skript.html (дата обращения: 05.02.2024 г.)
9. Триггер (компьютерные игры). [Электронный ресурс] URL: https://dic.academic.ru/dic.nsf/ruwiki/247708 (дата обращения: 07.02.2024 г.).
10. Что такое коллизия. [Электронный ресурс] URL: https://media-xyz.com/ru/articles/1404-kollaidery-i-khitboksy-zakony-stolknoveniia-i (дата обращения: 05.02.2024 г.).
11. Официальный сайт игры Celeste. [Электронный ресурс] URL: https://www.celestegame.com/ (дата обращения: 07.02.2024 г.).

12. Официальная страница игры Super Mario Bros. [Электронный ресурс] URL: https://www.nintendo.ru/-/NES/Super-Mario-Bros-803853.html (дата обращения: 07.02.2024 г.).

13. Официальный сайт Godot. [Электронный ресурс] URL: https://godotengine.org/ (дата обращения: 07.02.2024 г.).

14. Официальный сайт GameMaker. [Электронный ресурс] URL: https://gamemaker.io/ru (дата обращения: 07.02.2024 г.)

15. Официальный сайт Unity. [Электронный ресурс] URL: https://unity.com/ (дата обращения: 07.02.2024 г.).

ПРИЛОЖЕНИЯ

Приложение А. Спецификации вариантов использования

Спецификации вариантов использования (ВИ) системы представлена в таблицах 1–7.

Таблица 1 – Спецификация ВИ «Начать игру»

Прецедент: Начать игру
ID: 1
Краткое описание: Запуск игры позволяет начать проходить игровой уровень.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Успешный запуск игрового приложения. 2. Пользователь должен находиться в главном меню.
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает кнопку «New Game». 2. Приложение запускает игровой уровень и начинается игровой процесс.
Постусловия: Нет
Альтернативные потоки: Нет

Таблица 2 – Спецификация ВИ «Играть»

Прецедент: Играть
ID: 2
Краткое описание: Управление персонажем и прохождение игровых уровней.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Успешный запуск игрового уровня.
Основной поток: 1. Вариант использования начинается, когда завершается вариант использования 1. 2. Пользователь управляет игровым персонажем.
Постусловия: Нет
Альтернативные потоки: Нет

Таблица 3 – Спецификация ВИ «Атаковать»

Прецедент: Атаковать
ID: 3
Краткое описание: Персонаж выстреливает из оружия снарядом, который наносит урон противникам.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Пользователь проходит игровой уровень.
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает левую кнопку мыши. 2. Игровой персонаж выстреливает снарядом.
Постусловия: Нет
Альтернативные потоки: Нет

Таблица 4 – Спецификация ВИ «Бежать»

Прецедент: Бежать
ID: 4
Краткое описание: Игровой персонаж перемещается по уровню.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Пользователь проходит игровой уровень.
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает кнопки «A» или «D» на клавиатуре. 2. Игровой персонаж перемещается по уровню.
Постусловия: Нет
Альтернативные потоки: Нет

Таблица 5 – Спецификация ВИ «Прыгать»

Прецедент: Прыгать
ID: 5
Краткое описание: Игровой персонаж совершает прыжок.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Пользователь проходит игровой уровень.
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает кнопку «Пробел» на клавиатуре. 2. Игровой персонаж совершает прыжок.
Постусловия: Нет
Альтернативные потоки: Нет

Таблица 6 – Спецификация ВИ «Настроить игру»

Прецедент: Настроить игру
ID: 6
Краткое описание: Пользователь настраивает уровень громкости игры.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Пользователь открыл меню настроек. 2. Пользователь открыл меню паузы.
Основной поток: 1. Вариант использования начинается, когда пользователь открывает меню настроек. 2. Пользователь может отрегулировать громкость внутриигровых звуков и музыки.
Постусловия: 1. Программа сохраняет настройки.
Альтернативные потоки: Нет

Таблица 7 – Спецификация ВИ «Выйти из игры»

Прецедент: Выйти из игры
ID: 7
Краткое описание: Пользователь закрывает игровое приложение.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Пользователь находится в главном меню. 2. Пользователь открыл меню паузы. 3. Активно меню конца игры.
Основной поток: 1. Вариант использования начинается, когда пользователь находится либо в главном меню, либо в меню паузы, либо в меню конца игры. 2. Пользователь может закрыть игровое приложение.
Постусловия: Нет
Альтернативные потоки: Нет

Приложение Б. Скриншоты финальной версии игры

На рисунках 1–7 представлены скриншоты итогового игрового приложения.

Рисунок 1 – Главное меню

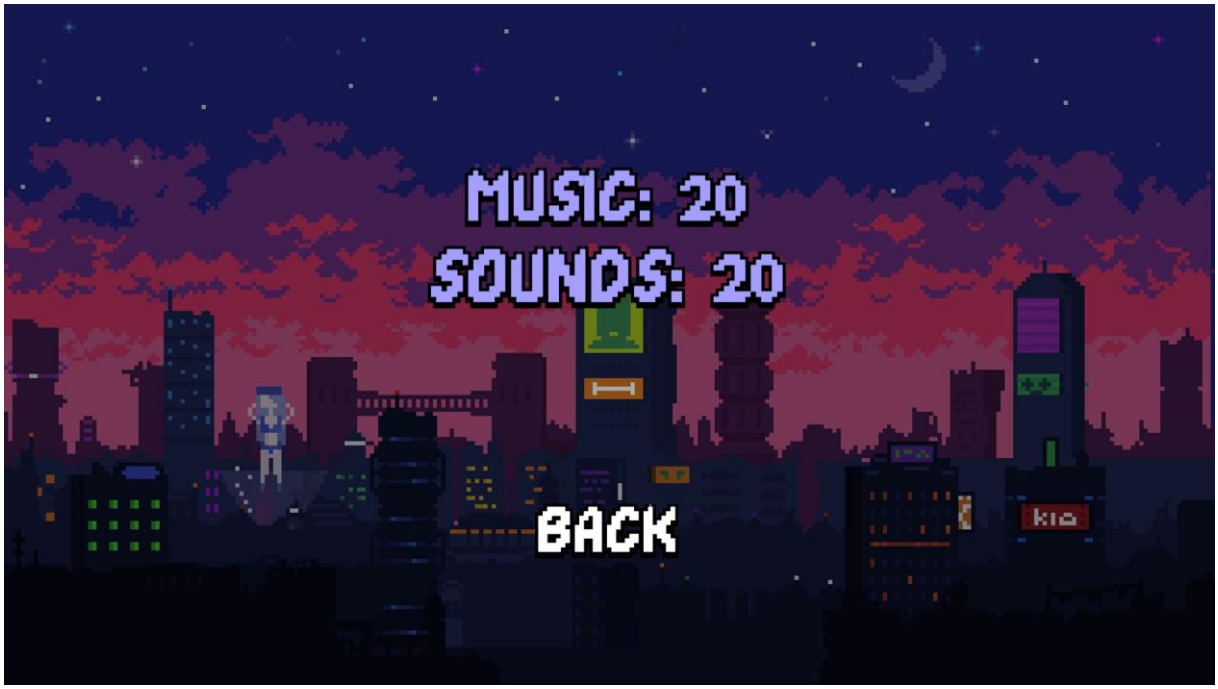

Рисунок 2 – Меню настроек громкости

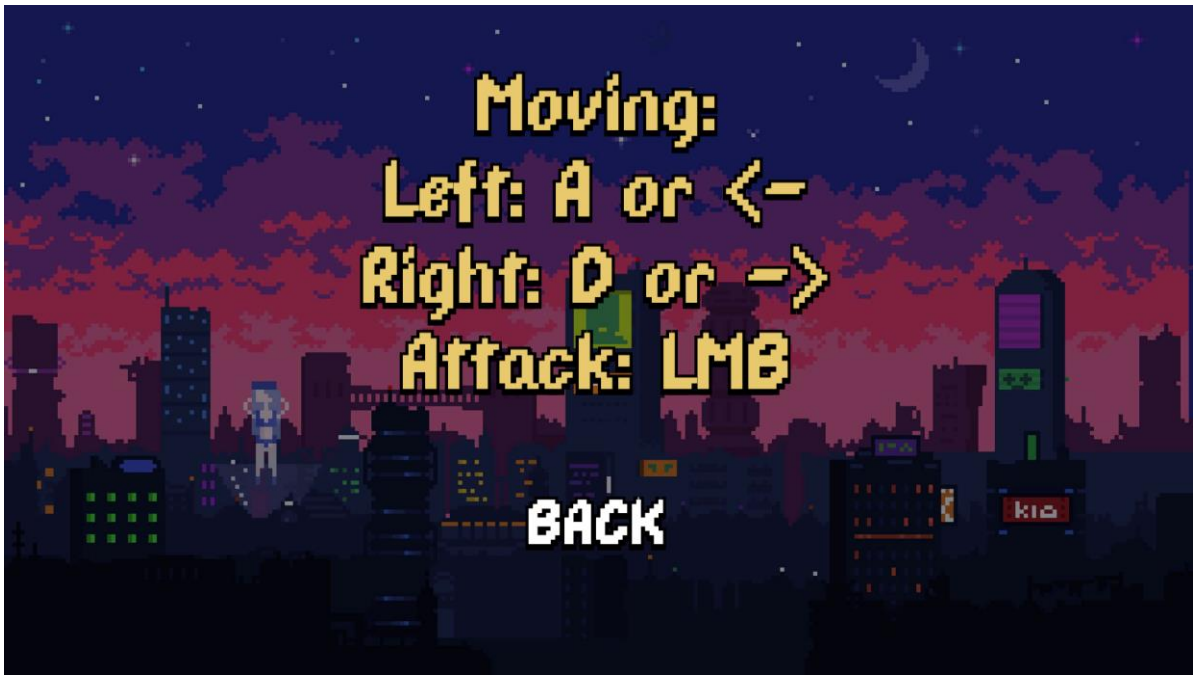

Рисунок 3 – Меню с указанием элементов управления

Рисунок 4 – Меню выбор уровня для прохождения

Рисунок 5 – Пример игрового процесса

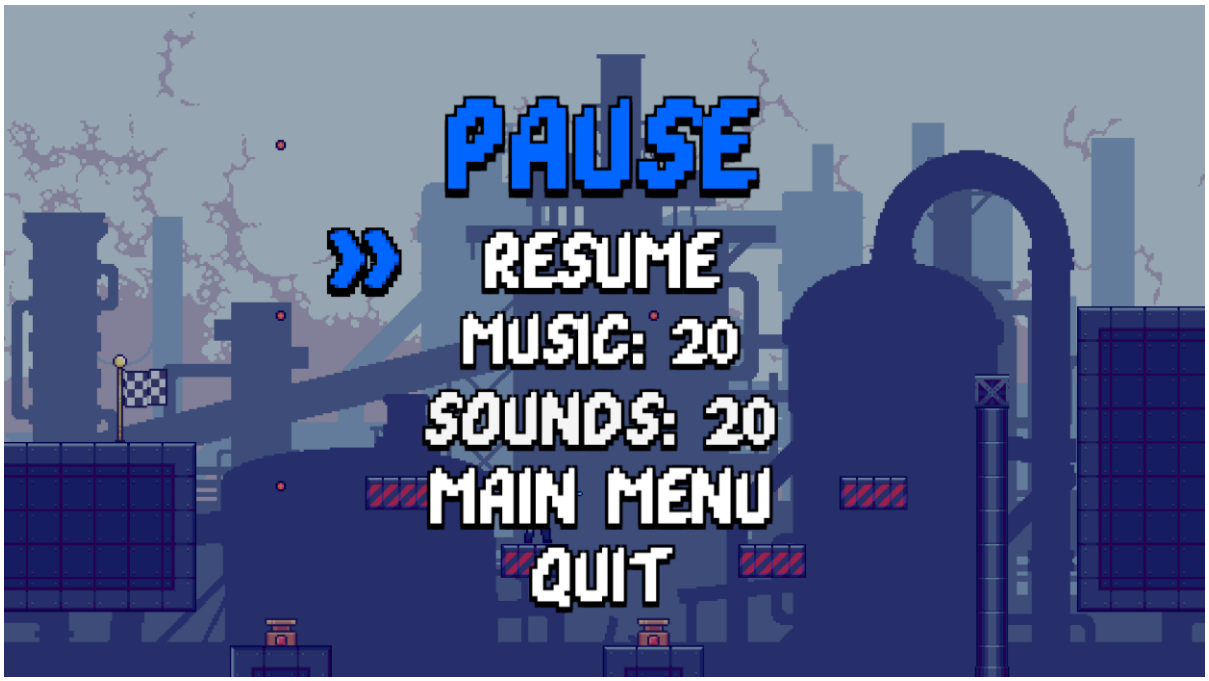

Рисунок 6 – Меню паузы

Рисунок 7 – Меню конца игры