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 «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д. В. Топольский
«___»_____ 2022 г.

РАЗРАБОТКА СЮЖЕТНОЙ 2D-ИГРЫ В ЖАНРЕ ПРИКЛЮЧЕНИЙ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-09.03.01.2022.204 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____Е. С. Ярош
«___»_____2022 г.

Автор работы,
студентка группы КЭ-405
_____Д. С. Балашова
«___»_____2022 г.

Нормоконтролёр,
к.п.н., доцент каф. ЭВМ
_____ М. А. Алтухова
«___»_____2022 г.

Челябинск-2022

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 «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д. В. Топольский
«___»_____2022 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студентке группы КЭ-405
Балашовой Дарье Сергеевне,
обучающейся по направлению
09.03.01 «Информатика и вычислительная техника»

1) **Тема работы:** «Разработка сюжетной 2D-игры в жанре приключений» утверждена приказом по университету № 308/141 от 12.12.2021.

2) **Срок сдачи студентом законченной работы:** «24» мая 2022 г.

3) **Исходные данные к работе:**

– разработка игры должна осуществляться для ОС Windows;

– разрабатываемый продукт должен быть написан на языке C#;

– возможность сохранения и загрузки игрового прогресса;

– возможность взаимодействия игрока с игровыми предметами;

– возможность взаимодействия игрока с неигровыми персонажами.

4) **Перечень подлежащих разработке вопросов:**

– анализ существующих аналогов;

– анализ и выбор средств реализации проекта;

– реализация проекта;

– тестирование и оценка работоспособности и готовности продукта.

5) **Дата выдачи задания:** «1» декабря 2021 г.

Руководитель работы _____/ Е. С. Ярош /

Студентка _____/ Д. С. Балашова /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы. Анализ существующих аналогов и выбор средств реализации	07.02.2022	
Определение требований, проектирование игрового проекта	07.03.2022	
Реализация системы	04.04.2022	
Тестирование, отладка, эксперименты	10.05.2022	
Компоновка текста работы и сдача на нормоконтроль	16.05.2022	
Подготовка презентации и доклада	24.05.2022	

Руководитель работы _____/Е. С. Ярош/

Студентка _____/Д. С. Балашова/

АННОТАЦИЯ

Д. С. Балашова. Разработка сюжетной 2D-игры в жанре приключений. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2022, 47 с., 15 ил., библиогр. список – 18 наим.

В рамках выпускной квалификационной работы выполнена разработка сюжетной 2D-игры в жанре приключений.

В данной работе был проведен сравнительный анализ существующих игровых проектов, а также основных технологический решений с целью выбора подходящего игрового движка для разработки, определены требования к разрабатываемому продукту, произведено проектирование, разработка и тестирование готового игрового продукта.

Результатом выполненной работы является полноценно функционирующая компьютерная игра.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Темой выпускной квалификационной работы является разработка сюжетной 2D-игры в жанре приключений.

Игровая индустрия является одной из самых быстро развивающихся отраслей компьютерных технологий и глобального спектра развлечений. Компьютерные игры как самый настоящий культурный феномен обретают чрезвычайную популярность среди различных возрастных категорий людей по всему миру. Видеоигры – это не только развлечение и способ ухода от рутинной реальности. Это еще и возможность для указания на проблемы современного общества с целью обращения на них внимания.

По оценкам аналитической компании Newzoo, к концу 2021 года по всему миру аудитория видеоигр приблизилась к отметке в 2,96 млрд. человек, а это на 5,4% больше, чем в 2020 году [1]. Аналитики ожидают, что количество геймеров продолжит расти и дальше.

По данным PwC, за 2021 год рынок видеоигр в России вырос до 158 млрд рублей, увеличившись на 8%. Согласно прогнозу PwC, в период до 2025 года российский рынок гейминга будет расти в среднем на 5% в год, а его объем достигнет значения в 186,5 млрд рублей [2]. Однако события, происходящие на данный момент в мире, поставили под сомнение многие планы по развитию российской игровой индустрии. К примеру, один из крупнейших онлайн-сервисов цифрового распространения игр “Steam” приостановил выплаты разработчикам из России, а многие авторы лишились возможности оплатить необходимый софт для разработки своих проектов. Кроме того, не остались в стороне и потребители игровой отрасли, поскольку некоторые зарубежные игровые студии и издатели объявили о приостановке деятельности в России.

Таким образом, тема моей выпускной квалификационной работы является как никогда актуальной из соображений внесения вклада в развитие российской игровой индустрии, несмотря на ограничения.

Целью выпускной квалификационной работы является создание 2D-игры в жанре приключений, содержащей в себе сюжетную часть.

Для достижения поставленной цели, необходимо решить следующие задачи:

1) проанализировать существующие игровые проекты подобного жанра;

2) осуществить анализ игровых движков для разработки;

3) разработать проект игры;

4) реализовать компьютерную игру;

5) провести тестирование реализованной компьютерной игры.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор существующих аналогов

Для реализации данного проекта были выбраны следующие жанры:

1) платформер;

2) приключение.

Платформер (platformer, platform game) – жанр компьютерных игр, основу игрового процесса которого составляют прыжки, сбор предметов, необходимых для прохождения игры и борьба с врагами.

Приключение (adventure) – жанр компьютерных игр, представляющий собой интерактивную историю с главным персонажем, управляемым игроком. Важнейшим элементом игры в данном жанре является исследование мира.

В рамках исследования предметной области были рассмотрены следующие наиболее популярные аналоги игровых проектов подобного жанра:

1) Hollow Knight;

2) Cuphead;

3) Little Nightmares;

4) Limbo;

5) Seasons after Fall;

6) Inmost;

7) Gris;

8) Ori and the Blind Forest;

9) Night in The Woods;

10) Skull: The Hero Slayer.

Hollow Knight [3] – это мультиплатформенная компьютерная игра в жанрах метроидвания (поджанр приключенческого боевика) и платформера, выпущенная инди-студией Team Cherry на движке Unity.

Преимущества:

– огромный игровой мир, открытый для исследований;

– продуманный сюжет продолжительностью в 40 часов;

– кроссплатформенность;

– различные игровые механики.

Недостатки:

– управление заточено исключительно под геймпад;

– не слишком удачно сделана платформенная часть, неудобство прыжков на мелкие уступы.

Cuphead [4] – это мультиплатформенная компьютерная игра в жанрах платформера и «run and gun», разработанная канадской командой разработчиков StudioMDHR Entertainment на движке Unity.

Преимущества:

– кроссплатформенность

– уникальная графика в ретро-стиле (каждая анимация в игре была отрисована вручную);

– разнообразный геймплей;

– интересные разнообразные боссы.

Недостатки:

– данная игра отличается высокой сложностью прохождения.

Little Nightmares [5] – мультиплатформенная компьютерная игра в жанрах платформера и приключений, разработанная шведской компанией Tarsier Studios на движке Unreal Engine 4 и выпущенная компанией Bandsi Namco Entertainment.

Преимущества:

– уникальная и атмосферная графика;

– кроссплатформенность;

– продуманный сюжет;

– разнообразный геймплей.

Недостатки:

– не всегда отзывчивое управление на клавиатуре.

Limbo [6] – мультиплатформенная компьютерная игра в жанрах платформер-головоломка с элементами survival horror, разработанная независимой датской студией Playdead на движке Box2D.

Преимущества:

– атмосферная графика в черно-белых тонах;

– кроссплатформенность;

– продуманный нестандартный сюжет;

– большое количество игровых механик, логических загадок.

Недостатки:

– запутанный, нарочито неясный заключительный акт;

– высокая требовательность к ПК.

Seasons After Fall [7] – мультиплатформенная компьютерная игра в жанрах платформер-головоломка, разработанная компанией Swing Swing Submarine на движке Unity и изданная Focus Home Interactive.

Преимущества:

– приятная красочная графика;

– музыкальное сопровождение;

– кроссплатформенность;

– интересная игровая механика со сменой времен года.

Недостатки:

– порой не самое отзывчивое управление;

– малое количество контента.

Inmost [8] – мультиплатформенная компьютерная игра в жанрах приключение и платформер-головоломка, разработанная литовской компанией Hidden Layer Games на движке Unity и изданная Chucklefish.

Преимущества:

– большое количество разных механик;

– живой мир пиксель-арта с огромным количеством анимаций;

– продуманный сюжет;

– кроссплатформенность;

– атмосферное музыкальное сопровождение.

Недостатки:

– в некоторых сценах интуитивно не особо понятно, что требуется от игрока.

Gris [9] – мультиплатформенная компьютерная игра в жанрах головоломка-платформер и приключение, разработанная испанской командой независимых разработчиков Nomada Studio на движке Unity и изданная Devolver Digital.

Преимущества:

– красочная графика в акварельных тонах;

– кроссплатформенность;

– разнообразный геймплей с интересными способностями героини;

– атмосферное музыкальное сопровождение.

Недостатки:

– пазлы и прыжки местами выглядят слишком отвлеченными от эмоциональной составляющей части истории;

– небольшая продолжительность игры.

Ori and the Blind Forest [10] – мультиплатформенная компьютерная игра в жанрах платформер и метроидвания, разработанная студией Moon Studios на движке Unity и изданная Microsoft Studios.

Преимущества:

– красочная уникальная графика в синих тонах;

– кроссплатформенность;

– отточенная игровая механика и отзывчивое управление;

– приятное музыкальное сопровождение.

Недостатки:

– небольшая продолжительность игры.

Night in the Woods [11] – мультиплатформенная компьютерная играв в жанрах платформер и приключение, разработанная на движке Unity и выпущенная студией Infinite Fall.

Преимущества:

– красочная графика;

– приятное музыкальное сопровождение;

– кроссплатформенность;

– продуманный сюжет с большим количеством интересных историй.

Недостатки:

– однообразность игрового процесса;

– не слишком продуманная концовка.

Skull: The Hero Slayer [12] – мультиплатформенная компьютерная игра в жанрах платформер и приключения, разработанная южнокорейской студией SouthPAW Games на движке Unity и изданная NEOWIZ.

Преимущества:

– красочная графика в стиле пиксель-арта;

– проработанный сюжет и персонажи;

– богатый возможностями геймплей;

– кроссплатформенность.

Недостатки:

– довольно большое количество багов.

Проведя анализ существующих разработок, можно сделать вывод, что для успешности игры необходимо разнообразить разрабатываемый продукт как в области геймплея, так и в области сюжета.

1.2 Анализ основных технологических решений

Разработка и редактирование кода игры будет происходить в Visual Studio 2022 Enterprise, поскольку данная среда обладает рядом функциональных

возможностей, позволяющих ускорить и упростить разработку проекта. К примеру, по умолчанию Visual Studio форматирует код по мере его ввода, применяя цветовое кодирование, что делает его более удобным для чтения. Также широкие возможности инструментов отладки позволяют отслеживать ошибки и странное поведение кода.

Для модификации спрайтов будет использоваться программное обеспечение Adobe Photoshop 2022, поскольку это самый популярный и многофункциональный графический редактор, обладающий огромным количеством инструментов для работы с изображениями.

Основным средством для разработки компьютерной игры является игровой движок – рабочая среда разработки игр. Проанализируем основные и наиболее популярные игровые движки:

1) Unity;

2) Unreal Engine 4;

3) GameMaker: Studio 2.

Unity [13] – это межплатформенная среда для разработки 2D- и 3D-игр, разработанная американской компанией Unity Technologies. На данный момент является одним из самых популярных движков. Для написания скриптов движок использует язык C#.

Преимущества:

– редактор Unity имеет простой Drag&Drop интерфейс, благодаря которому можно производить отладку игры прямо в нем;

– наличие визуальной среды разработки, включающей в себя как инструментарий визуального моделирования, так и интегрированную среду, цепочку сборки, направленную на повышение производительности этапов создания прототипов и тестирования;

– модульная система компонентов Unity, при которой объекты создаются посредством объединения функциональных блоков, что облегчает создание прототипов;

– наличие огромного количества готовых ассетов;

– кроссплатформенная поддержка.

Недостатки:

– ограничение визуального редактора при работе с многокомпонентными схемами;

– отсутствие поддержки Unity ссылок на внешние библиотеки;

– игры, созданные на Unity, довольно тяжеловесны, поэтому уступают в производительности для мобильных платформ.

Unreal Engine 4 [14] – игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. UE4, поддерживает сразу два языка программирования для разработки игр: C++ и визуальный язык Blueprints.

Достоинства:

– визуальный язык Blueprints, в котором игровая логика выстраивается с помощью связанных между собой блоков, позволяющий реализовать некий игровой процесс без знаний программирования;

– кроссплатформенная поддержка;

– активное сообщество с огромным количеством opensource-инструментов;

– наличие большого инструментария для моделирования уровней;

– наличие большого количества готовых ассетов.

Недостатки:

– для разработки на данном движке требуется достаточно производительная система;

– громоздкий интерфейс.

GameMaker: Studio 2 [15] – один из самых популярных игровых движков, разрабатываемый и поддерживаемый компанией YoYo Games. Основным языком программирования является собственный скриптовый язык (GML).

Достоинства:

– простота освоения, благодаря системе визуального скриптового инструмента Drag&Drop;

– собственный упрощенный язык Game Maker Language (GML), который не требует углубленного изучения кода;

– кроссплатформенная поддержка;

– наличие инструментария для создания кат-сцен.

Недостатки:

– несмотря на упрощенный экспорт игр на другие платформы, нет поддержи одновременного выпуска патчей;

– из-за недостатка функциональности не может конкурировать с более мощными движками;

– медленная работа движка из-за элементарного языка программирования.

1.3 Вывод

В качестве языка программирования мною был выбран язык программирования C#, потому что разрабатываемая игра небольшая и нетребовательная, а данный язык имеет преимущество по скорости разработки перед языком C++. Таким образом, можно сделать вывод, что из рассмотренных технологических решений наиболее подходящим для реализации игрового проекта является Unity, поскольку основным языком сценариев данного движка является C#. Наличие огромной библиотеки готовых бесплатных ассетов и плагинов для данного игрового движка значительно упрощает разработку игры, что также является преимуществом по отношению к выбору Unity как среды для реализации.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

К общим требованиям для персонального компьютера можно отнести следующие минимальные системные требования:

– операционная система (ОС): Windows 7 SP1+, 8, 10, 11, только 64-разрядные версии; Mac OS X 10.12+; Ubuntu 16.04, 18.04; CentOS 7;

– процессор: 4-ядерный процессор линеек Intel Core i3, i5, i7, AMD FX 8000;

– оперативная память: 4 ГБ;

– видеокарта: GT1030 и выше или встроенная графика уровня Intel HD Graphics 610 и выше;

– DirectX: версии 11;

– звуковая карта: совместимая с DirectX.

Данные системные требования являются минимальными для продукции, производимой на движке Unity.

2.1 Функциональные требования

Определим следующие функциональные требования:

1) наличие одного игрового персонажа под управлением игрока;

2) наличие одного уровня сложности;

3) наличие одной логической задачи;

4) ведение счета за каждую победу над врагом и решение логической задачи;

5) наличие игрового меню.

Взаимодействие с игровым меню включает в себя следующие возможности: запуск игры, настройки, выход из игры.

1) возможность сохранения игрового процесса и продолжения прохождения с места остановки игры;

2) взаимодействие с игровыми предметами;

3) следование камеры за персонажем, которым управляет игрок;

4) наличие звукового сопровождения для бега, прыжка и удара.

Необходимо реализовать следующие подсистемы:

– запуск игры;

– меню настроек игры;

– выход из игры.

Функциональные требования к подсистеме «Запуск игры»:

Местоположение игрового персонажа устанавливается на точку возрождения в начальной локации.

Функциональные требования к подсистеме «Меню настроек игры»:

1) изменение громкости звука в игре;

2) изменение качества графики.

Функциональные требования к подсистеме «Выход из игры»:

1) при использовании из игрового меню происходит переход в главное меню игры;

2) при использовании из главного меню игры происходит завершение игрового процесса и выход из игры.

Логическая задача заключается в сборке фрагментов пазла в единую картинку. Опишем условия для данной задачи:

1) поле пазла состоит из 6 фрагментов;

2) каждый фрагмент пазла исключительно прямоугольной формы и одного размера;

3) поворот частей пазла осуществляется по клику компьютерной мыши.

2.2 Нефункциональные требования

Определим нефункциональные требования:

1) разработка игры на платформе игрового движка Unity версии 2021.3.3f1;

2) скрипты должны быть написаны на языке программирования C#;

3) сюжетная часть, заключающаяся в путешествии главного персонажа по собственному сну с целью борьбы со страхами и переживаниями;

4) наличие одного вида игровых предметов, а именно сундуков;

5) наличие игрового сундука, содержащего в себе логическую задачу;

6) неигровые персонажи и игровые предметы находятся на заранее установленных точках, вариативность не предусматривается.

3 ПРОЕКТИРОВАНИЕ

3.1 Архитектура предлагаемого решения

Разработку игрового проекта можно разделить на следующие части:

1) работа с игровой логикой;
2) работа с анимациями;
3) работа с локациями;
4) работа с игровым интерфейсом;
5) работа с главным меню;
6) работа со звуком.

В модуле работы с игровой логикой с помощью написания скриптов создаются компоненты, позволяющие активировать игровые события, изменять параметры, а также поведение тех или иных объектов на сцене.

В модуле работы с анимациями предполагается реализация анимаций для действий различных объектов. Основными являются анимации игрового персонажа (состояние покоя, бег, прыжок, атака), неигровых персонажей (ходьба), а также игровых предметов (открывание сундуков). Анимации будут реализованы спрайтами с помощью встроенного компонента аниматора.

В данном проекте предполагается одна локация, представляющая собой подземелье, которое снится главному персонажу игры. На локации будут расположены сундуки с логическими загадками, за решение которых присуждаются очки к общему счету игры.

В игровом интерфейсе будет отображено количество жизней и общий счет за победу над врагом и решение задач. Также будет предусмотрена возможность постановки игры на паузу.

Модуль главного меню будет являться отдельной сценой, загружающей информацию о прогрессе, с возможностью изменения игровых параметров.

Звуковое сопровождение игры будет цикличным для сцены главного меню и игровой сцены. Для главного персонажа будет свой набор звуков, который будет проигрываться в зависимости от действия.

Файловая структура игры представлена на рисунке 1.

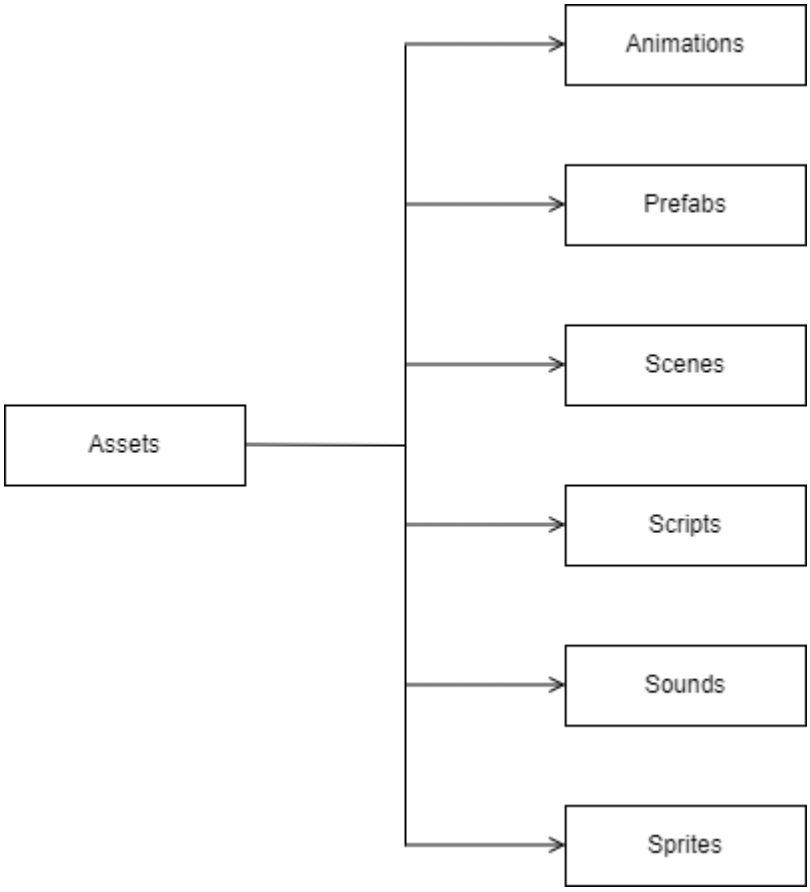

Рисунок 1 – Файловая структура игры

Файловая структура включает в себя следующие каталоги:

1) animations – контроллеры и анимации игровых объектов;

2) prefabs – шаблоны игровых объектов с настроенными параметрами;

3) scenes – сцены, описывающие организацию объектов в памяти;

4) scripts – скрипты, реализующие игровую логику;

5) sounds – звуки, используемые в игре;

6) sprites – все используемые спрайты.

4 РЕАЛИЗАЦИЯ

Все спрайты, используемые в реализации графики игрового проекта, взяты с официального сайта Unity Asset Stor [16]. Звуковое сопровождение также взято со стоковых сайтов звуков с открытой лицензией на использование [17].

4.1 Главное меню

При запуске игры перед игроком предстает сцена главного меню, отображенного на рисунке 2.

Рисунок 2 – Главное меню

Навигация по главному меню осуществляется с помощью компьютерной мыши. Главное меню включает в себя следующие опции:

– играть;

– выход;

– настройки.

Используя опцию «играть», игрок может начать игру. Функциональная возможность «выход» завершает игровой процесс, закрывая игру. С помощью

опции «настройки» игроку предоставляется возможность выбора настройки параметров игры, представленных на рисунке 3.

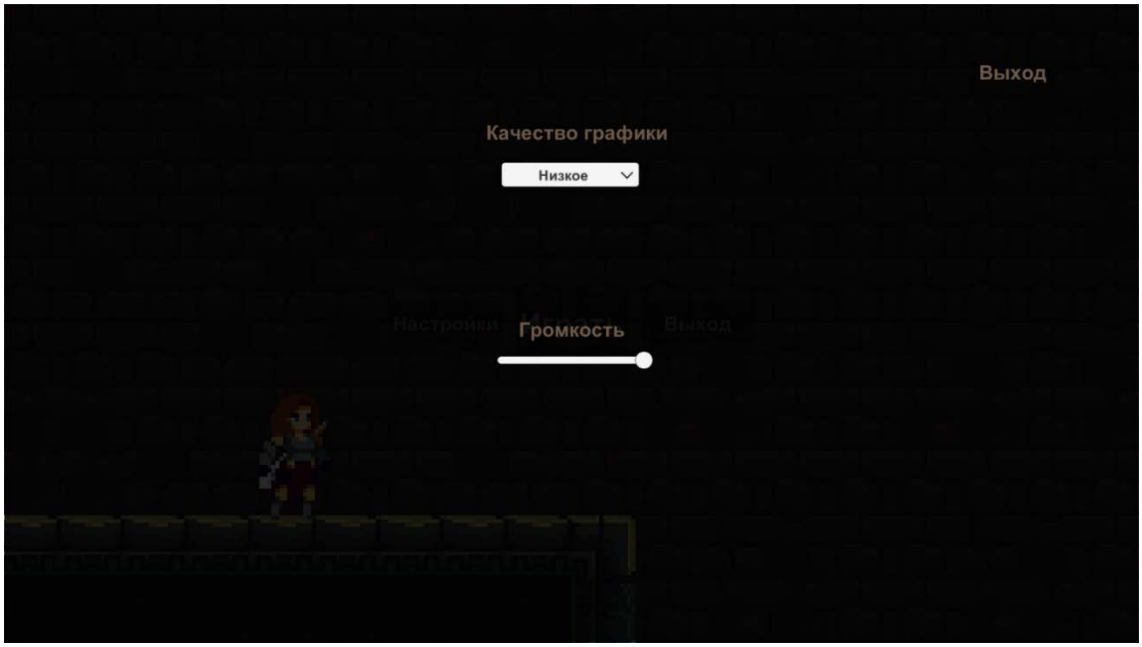

Рисунок 3 – Меню настроек

С помощью данного меню можно изменять следующие параметры:

– качество графики игры (низкое, среднее, высокое);

– громкость звукового сопровождения как в главном меню, так и в игре;

– выход в главное меню.

4.2 Игровой персонаж

Главным персонажем игры является девушка, путешествующая по своему сну. Использующийся спрайт отображен на рисунке 4.

Рисунок 4 – Спрайт главного персонажа игры [18]

Персонаж под управлением игрока имеет следующие параметры:

– ограниченное количество жизней;

– возможность свободного передвижения по карте;

– взаимодействие с объектами;

– анимированное перемещение со звуковым сопровождением.

Анимации передвижений главного персонажа были реализованы с помощью встроенного компонента Animator. Структура переходов анимаций представлена на рисунке 5.

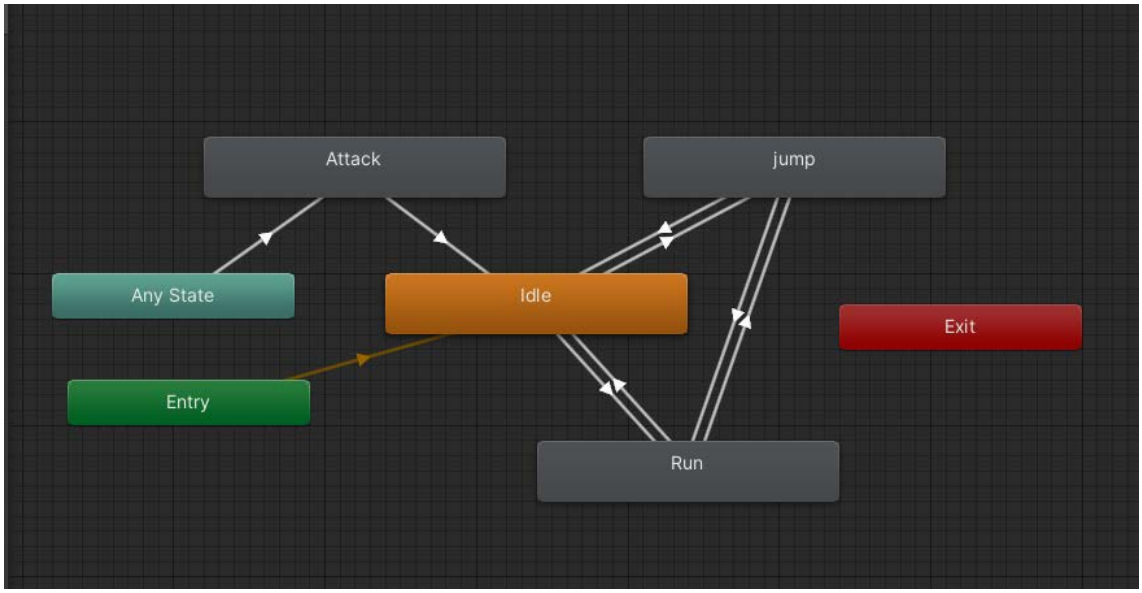

Рисунок 5 – Структура переходов анимаций

4.3 Неигровые персонажи

В игровом проекте были реализованы два вида неигровых персонажей, представленных на рисунке 6. Данные персонажи передвигаются по четко заданной территории и могут наносить урон главному персонажу.

Рисунок 6 – Спрайты неигровых персонажей

4.4 Игровой процесс

Скриншот игрового процесса отображен на рисунке 7.

Рисунок 7 – Игровой процесс

На интерфейсе игрового процесса показано количество жизней главного персонажа, его счет. Также во время игры игрок имеет возможность поставить игру на паузу путем выхода в игровое меню, представленное на рисунке 8.

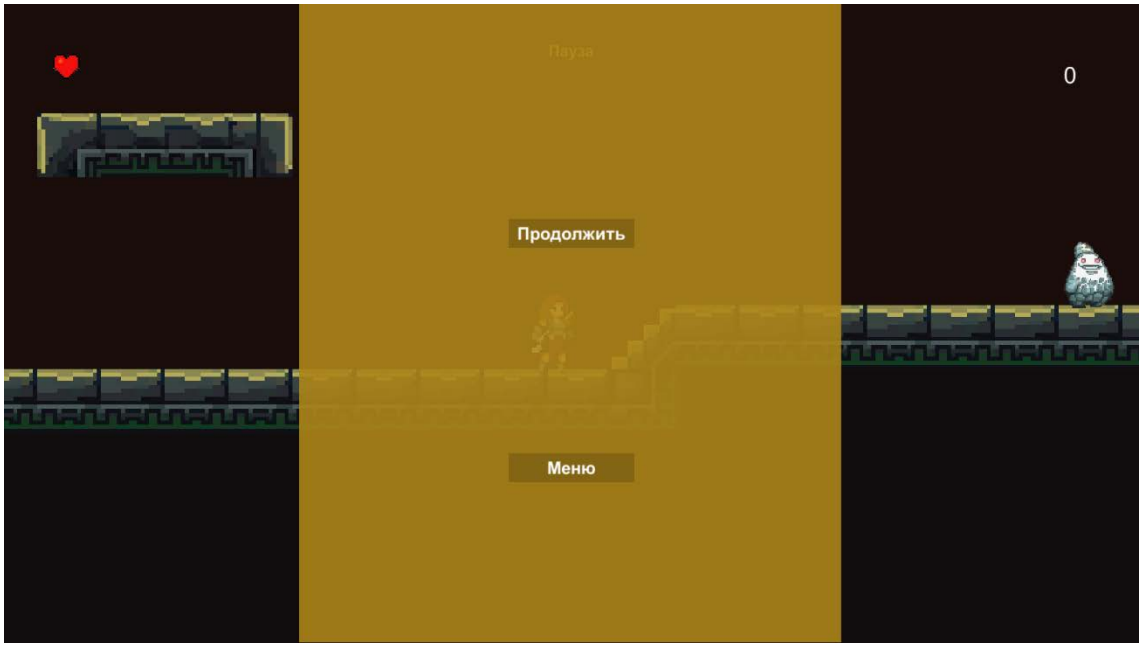

Рисунок 8 – Игровое меню

Игровое меню предоставляет следующие функциональные возможности:

– продолжение игры;

– выход в главное меню.

Игрок может свободно перемещаться по карте, сражаться с врагами и взаимодействовать с игровыми сундуками, если находится в радиусе их взаимодействия. За каждую победу над врагом увеличивается общий счет игрока.

Пример задачи на логику представлен на рисунке 9.

Рисунок 9 – Логическая задача

В данном случае имеется одна логическая задача, которая заключается в том, чтобы собрать части пазла в единую картинку. Условия для данной задачи описаны в п. 2.1. За решение задачи также увеличивается общий счет игрока.

4.5 Игровая логика

Для реализации игровой логики было создано 10 скриптов на языке C#. Описание скриптов представлено в таблице 1.

Таблица 1 – Описание скриптов

Наименование скрипта	**Назначение**	**№ листинга**
Player	Скрипт, отвечающий за поведение и передвижение главного персонажа по карте	А.1 приложения А
PlayerCombat	Скрипт, отвечающий за возможность атаковать врага	А.2 приложения А
Enemy	Скрипт, отвечающий за поведение и передвижение врага	А.3 приложения А
GameControl	Скрипт, отвечающий за логическую задачу в виде пазла	А.4 приложения А
TouchRotate	Скрипт, отвечающий за поворот частей пазла при клике	А.5 приложения А
ChestLogic	Скрипт, отвечающий за игровой объект «сундук»	А.6 приложения А
VolumeValue	Скрипт, отвечающий за настройку параметра «громкость»	А.7 приложения А
QualityLogic	Скрипт, отвечающий за настройку параметра «графика»	А.8 приложения А
SceneLogic	Скрипт, отвечающий за переключение между сценами	А.9 приложения А
SavePosition	Скрипт, отвечающий за сохранение позиции игрока	А.10 приложения А

Описание методов для скрипта Player представлено в таблице 2.

Таблица 2 – Описание методов для скрипта Player

Наименование метода	**Назначение**
Start	Получение компонентов
Update	Проверка движения игрока
FixedUpdate	Передвижение персонажа
Flip	Поворот персонажа
CheckGround	Проверка того, находится ли игрок на земле
OnTriggerEnter2D	Касание врага

Передвижение персонажа осуществляется с использованием компонента Rigidbody2D. Данный компонент сообщает движку Unity, что для игрового объекта Player должно имитироваться действие законов физики. Параметры, заданные в компоненте Rigidbody2D представлены на рисунке 10.

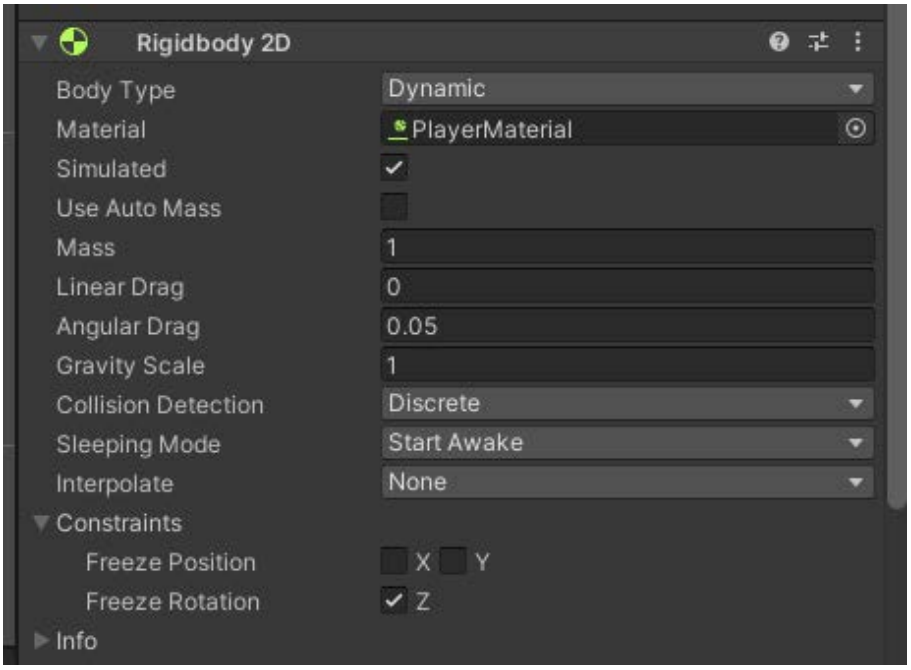

Рисунок 10 – Параметры компонента Rigidbody2D

Параметры массы (Mass), углового сопротивления (Angular Drag) и гравитации (Gravity Scale) задаются по умолчанию для реалистичного перемещения твердых тел.

Траектория движения персонажа задается путем изменения значения «x» вектора Vector2 и присваивания его к transform position игрока. Траектория движения игрока представлена в листинге 1.

Листинг 1 – Траектория движения игрока

```
private void FixedUpdate()
    {
        Vector2 targetVelocity = new Vector2(horizontalMove * 10f,
rb.velocity.y);
        rb.velocity = targetVelocity;
    }
```

При осуществлении прыжка учитываются параметры силы прыжка (jumpForce), силы тяжести (Gravity Scale), а также скорость (speed) в случае, если игрок в этот момент перемещается в пространстве.

Касание игроком врага осуществляется с помощью функции для обработки столкновений OnTriggerEnter2D. Она выполняется, когда радиус коллайдера игрока соприкасается с радиусом коллайдера врага. Радиусы коллайдеров задаются в объектах персонажей. Таким образом, задается граница области, позволяющая определить факт соприкосновения.

Описание методов для скрипта PlayerCombat представлено в таблице 3.

Таблица 3 – Описание методов для скрипта PlayerCombat

Наименование метода	**Назначение**
Start	Получение компонентов
Update	При нажатии на клавишу «K» происходит осуществление атаки
Attack	Атака главного персонажа
OnDrawGizmosSelectes	Отображение круга attackRange для наглядной настройки параметров
PuzzleBonus	Бонус к общему счету за решение логической задачи

Атака главного персонажа осуществляется с использованием компонента физического движка, встроенного в Unity, Collider2D. Заданные параметры отражены в рисунке 11.

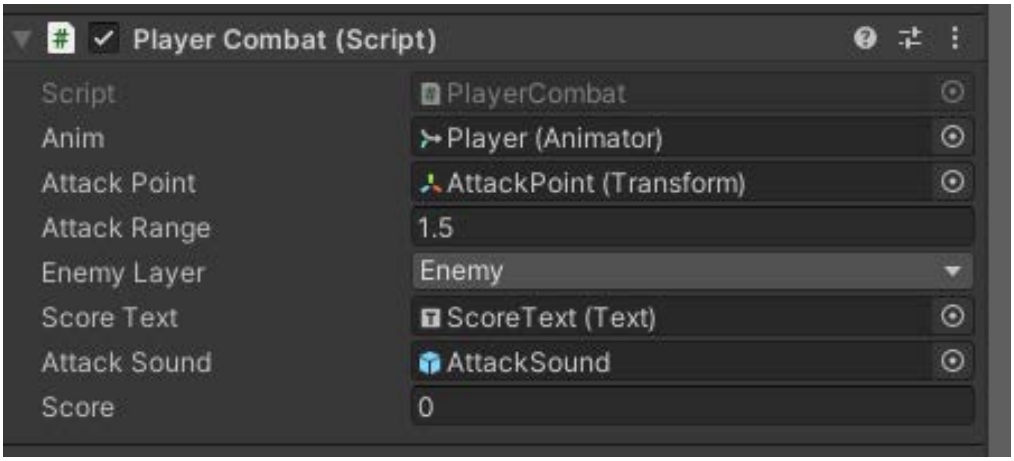

Рисунок 11 – Параметры скрипта PlayerCombat

Для атаки главного персонажа задаются параметры позиции атаки (Attack Point) и радиуса атаки (Attack Range). Позиция атаки определяется положением игрока в пространстве, а радиус атаки задается в объекте игрока.

Описание методов для скрипта Enemy представлено в таблице 4.

Таблица 4 – Описание методов для скрипта Enemy

Наименование метода	Назначение
Update	Выполнение для каждого кадра метода Chill
Chill	Траектория движения врага
EnemyDestroy	Разрушение врага

Алгоритм траектории движения врага представлен на рисунке 12. Сначала осуществляется проверка того, дошел ли враг до крайних точек радиуса действия, который задается через переменную positionOfPatrol в объекте игрока. Далее описываем траекторию с помощью Vector2, моделируя направление движения.

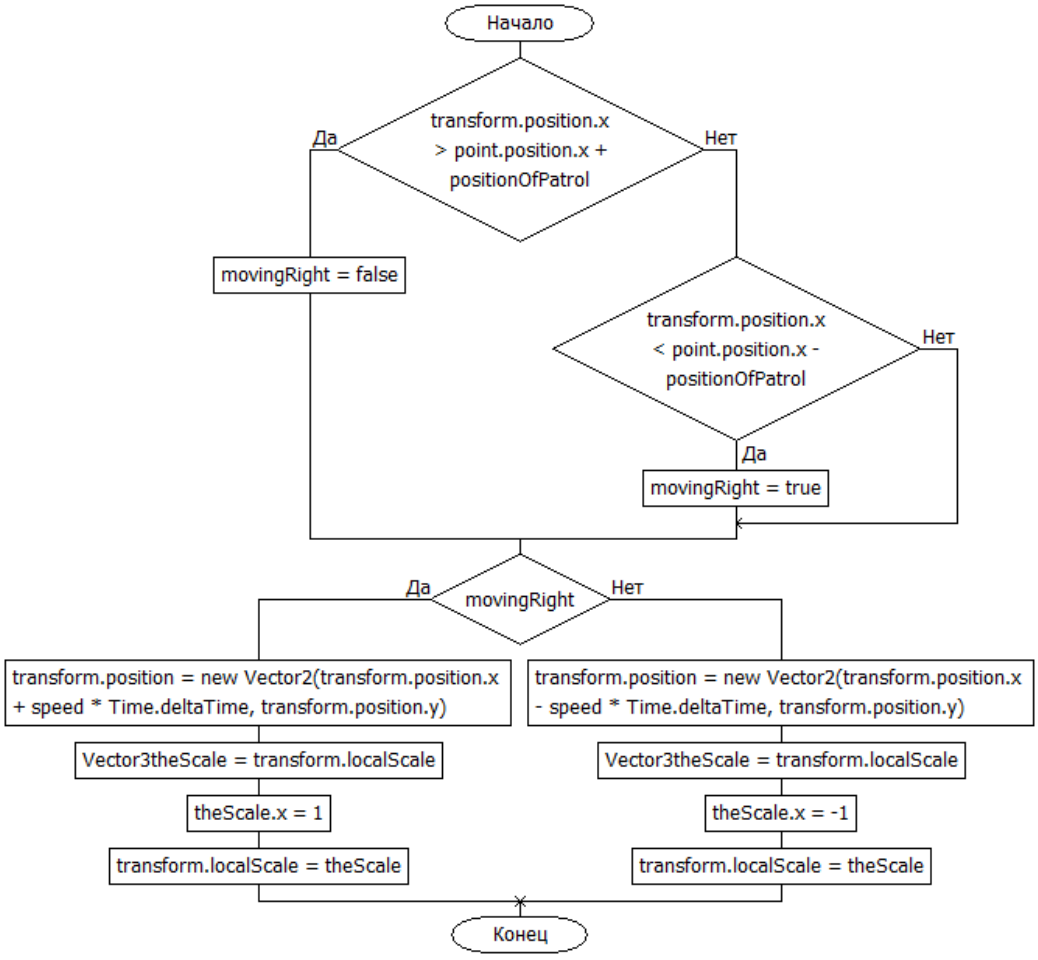

Рисунок 12 – Алгоритм траектории движения врага

Описание методов для скрипта GameControl представлено в таблице 5.

Таблица 5 – Описание методов для скрипта GameControl

Наименование метода	**Назначение**
Start	Получение компонентов
Update	Проверка положения пазла

Проверка положения пазла осуществляется путем определения положения частей пазла по оси z.

Описание методов для скрипта TouchRotate представлено в таблице 6.

Таблица 6 – Описание методов для скрипта TouchRotate

Наименование метода	**Назначение**
Rotate	Поворот частей пазла

Описание методов для скрипта ChestLogic представлено в таблице 7. Касание игроком сундука осуществляется с помощью функции для обработки столкновений OnTriggerEnter2D. Она выполняется, когда коллайдер игрока соприкасается с коллайдером сундука. Также воспроизводится анимация открывания.

Таблица 7 – Описание методов для скрипта ChestLogic

Наименование метода	**Назначение**
OnTriggerEnter2D	Проверка на касание игроком сундука
Puzzle	Активация пазла

Описание методов для скрипта VolumeValue представлено в таблице 8.

Таблица 8 – Описание методов для скрипта VolumeValue

Наименование метода	**Назначение**
Update	Изменение переменной через Slider
SetVolume	Приравнивание громкости к переменной

Описание методов для скрипта QualityLogic представлено в таблице 9.

Таблица 9 – Описание методов для скрипта QualityLogic

Наименование метода	**Назначение**
SetQuality	Настройка качества графики

Описание методов для скрипта SceneLogic представлено в таблице 10.

Таблица 10 – Описание методов для скрипта SceneLogic

Наименование метода	**Назначение**
Start	Получение компонентов
Menu	Переключение на сцену «Меню»
Play	Переключение на сцену «Играть»
Pause	Переключение на сцену «Пауза»
Continue	Переключение на сцену «Продолжить»
Exit	Переключение на сцену «Выход»

Описание методов для скрипта SavePosition представлено в таблице 11.

Таблица 11 – Описание методов для скрипта SavePosition

Наименование метода	**Назначение**
Start	Получение сохраненных координатов
Save	Сохранение местоположения

Сохранение местоположения происходит путем сохранения текущих значений playerX и playerY с ключами «x» и «y», соответственно, в хранилище PlayerPrefs.

5 ТЕСТИРОВАНИЕ

5.1 Методология тестирования

В рамках тестирования компьютерной игры использовалось нагрузочное тестирование с помощью одной из самых популярных утилит MSI Afterburner, поскольку она предоставляет удобный доступ ко всем настройкам графической подсистемы компьютера.

5.2 Проведение процедуры тестирования

Тестирование было проведено на персональном компьютере со следующими техническими характеристиками:

1) процессор: Intel(R) Core (TM) i5-10400F;

2) оперативная память: 16,00 Гб;

3) видеокарта: Nvidia GeForse GTX-1060 6 Гб;

4) система: Windows 11 версия 21H2 (сборка 22000.613) 64-Bit.

Во время тестирования были произведены замеры среднего количества FPS (количество сменяемых кадров за одну секунду) в зависимости от изменения настроек графики. Полученные данные представлены на рисунках 13-15. В результате во время прохождения выдается количество FPS в пределах 143-144, при этом CPU и GPU не подвержены значительной нагрузке.

Рисунок 13 – Результаты тестирования при настройках графики «Низкое»

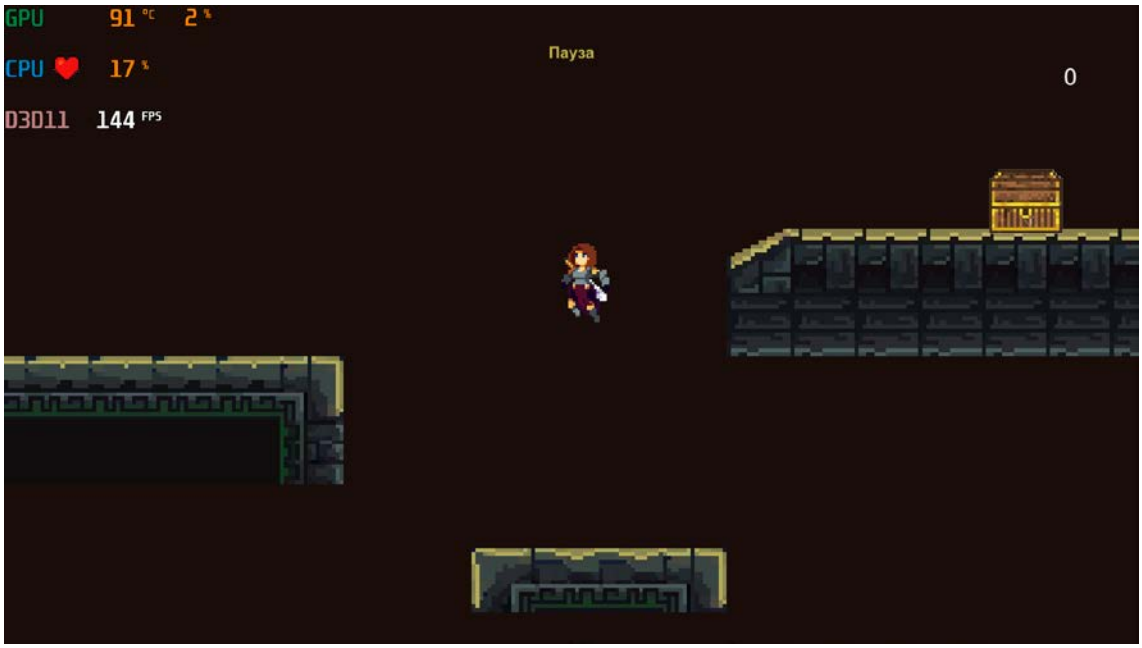

Рисунок 14 – Результаты тестирования при настройках графики «Среднее»

Рисунок 15 – Результаты тестирования при настройках графики «Высокое»

Также во время тестирования было выявлено, что:

1) все анимации проигрываются в соответствии с тем, что и предполагалось;

2) возможность сохранения работает корректно;

3) за каждую победу и решение задач даются очки к общему счету в соответствии с тем, что и прописано в скриптах;

4) настройки параметров игры работают корректно;

5) при выходе из игры приложение успешно закрывается и не отслеживается в диспетчере задач, что совпадает с функционалом опции выхода из игры.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы была спроектирована и разработана сюжетная компьютерная 2D-игра на игровом движке Unity.

Для этого были решены следующие задачи:

– проанализированы существующие игровые проекты подобного жанра;

– осуществлен анализ игровых движков для разработки;

– разработан проект игры;

– реализована компьютерная игра;

– проведено тестирование реализованной компьютерной игры.

У реализованного проекта имеются следующие перспективы развития:

– адаптация для других платформ;

– увеличение количества локаций;

– модификация и увеличение игровых механик;

– добавление новых логических задач;

– добавление неигровых персонажей;

– улучшение сюжетной части путем добавления диалогов с неигровыми персонажами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. The Games Market and Beyond in 2021: The Year in Numbers. – Текст. Изображение : электронные // Newzoo : [сайт]. – 22 декабря 2021. – URL: https://newzoo.com/insights/articles/the-games-market-in-2021-the-year-in-numbers-esports-cloud-gaming (дата обращения: 23.04.2022).

2. Что ждет рынок видеоигр в 2022 году? – Текст. Изображение : электронные // Финам : [сайт]. – 19 января 2022. – URL: https://www.finam.ru/analysis/newsitem/chto-zhdet-rynok-videoigr-v-2022-godu-20220119-183308/ (дата обращения: 23.04.2022).

3. Hollow Knight. – Текст. Изображение : электронные // Steam : [сайт]. – 24 февраля 2017. – URL: https://store.steampowered.com/app/367520/Hollow_Knight/?l=russian (дата обращения: 25.05.2022).

4. Cuphead. – Текст. Изображение : электронные // Steam : [сайт]. – 29 сентября 2017. – URL: https://store.steampowered.com/app/268910/Cuphead/ (дата обращения: 25.05.2022).

5. Little Nightmares. – Текст. Изображение : электронные // Steam : [сайт]. – 28 апреля 2017. – URL: https://store.steampowered.com/app/424840/Little_Nightmares/ (дата обращения: 25.05.2022).

6. Limbo. – Текст. Изображение : электронные // Steam : [сайт]. – 2 августа 2011. – URL: https://store.steampowered.com/app/48000/LIMBO/?l=russian (дата обращения: 25.05.2022).

7. Seasons after Fall. – Текст. Изображение : электронные // Steam : [сайт]. – 2 сентября 2016. – URL: https://store.steampowered.com/app/366320/Seasons_after_Fall/ (дата обращения: 25.05.2022).

8. Inmost. – Текст. Изображение : электронные // Steam : [сайт]. – 21 августа 2020. – URL: https://store.steampowered.com/app/938560/INMOST/ (дата обращения: 25.05.2022).

9. Gris. – Текст. Изображение : электронные // Steam : [сайт]. – 13 декабря 2018. – URL: https://store.steampowered.com/app/683320/GRIS/ (дата обращения: 25.05.2022).

10. Ori and the Blind Forest. – Текст. Изображение : электронные // Steam : [сайт]. – 9 июня 2014. – URL: https://store.steampowered.com/app/387290/Ori_and_the_Blind_Forest_Definitive_Edition/ (дата обращения: 25.05.2022).

11. Night in the Woods. – Текст. Изображение : электронные // Steam : [сайт]. – 21 февраля 2017. – URL: https://store.steampowered.com/app/481510/Night_in_the_Woods/ (дата обращения: 25.05.2022).

12. Skul: The Hero Slayer. – Текст. Изображение : электронные // Steam : [сайт]. – 21 января 2021. – URL: https://store.steampowered.com/app/1147560/Skul_The_Hero_Slayer/ (дата обращения: 25.05.2022).

13. Unity [официальный сайт] / Unity Technologies. – США, 2022. – . – URL: https://unity.com/ru (дата обращения: 25.05.2022). – Текст. Изображение : электронные.

14. Unreal Engine [официальный сайт] / Epic Games. – США, 2022. – . – URL: https://www.unrealengine.com/en-US/ (дата обращения: 25.05.2022). – Текст. Изображение : электронные.

15. GameMaker [официальный сайт] / YoYo Games Ltd. – Великобритания, 2022. – . – URL: https://gamemaker.io/ru (дата обращения: 25.05.2022). – Текст. Изображение : электронные.

16. Unity Asset Stor [официальный сайт] / Unity Technologies. – США, 2022. – . – URL: https://assetstore.unity.com/ (дата обращения: 25.05.2022). – Текст. Изображение : электронные.

17. Free Sound. – Текст. Изображение : электронные // Music Techology Research Group : [сайт]. – 2022. – URL: https://freesound.org/ (дата обращения: 25.05.2022).

18. Warrior Free Asset. – Текст. Изображение (подвижное ; двумерное) : электронные // Unity Technologies : [сайт]. – 27 мая 2021. – URL: https://assetstore.unity.com/packages/2d/characters/warrior-free-asset-195707 (дата обращения: 25.05.2022).

ПРИЛОЖЕНИЕ А
Фрагменты исходного кода игры

Листинг А.1 – Исходный код скрипта Player

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Player : MonoBehaviour
{
    //Объявление переменных
    public Rigidbody2D rb;
    public Animator anim;

    public float horizontalMove = 2f;

    public bool facingRight;

    [Header("Параметры предвижения игрока")]
    [Range(0, 10f)] public float jumpForce = 8f;
    [Range(0, 10f)] public float speed = 3f;


    [Space]
    [Header("Параметры касания земли")]
    public bool isGraunded = false;
    [Range(-5f, 5f)] public float checkGroundOffsetY = -1.8f;
    [Range(0, 5f)] public float checkGroundRadius = 0.3f;


    public GameObject jumpSound;

    //Получение компонентов
    private void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
    }

    //Вызывается раз за кадр
    private void Update()
    {
        //Проверка того, касается ли игрок земли для последующего осуществления
прыжка
        if (isGraunded && Input.GetKeyDown(KeyCode.Space))
        {
            Instantiate(jumpSound, gameObject.transform.position,
Quaternion.identity);
            rb.AddForce(transform.up * jumpForce, ForceMode2D.Impulse);
        }
        horizontalMove = Input.GetAxisRaw("Horizontal") * speed;

        if (horizontalMove < 0 && facingRight)
        {
            Flip();
        }
        else if (horizontalMove > 0 && !facingRight)
        {
            Flip();
        }
        CheckGround();
```

```
        //Проверка того, двигается ли игрок
        if (horizontalMove != 0)
        {
            anim.SetBool("isRunning", true);
        }
        else
        {
            anim.SetBool("isRunning", false);
        }

        if (isGraunded)
            anim.SetBool("isJumping", false);
        else
            anim.SetBool("isJumping", true);
    }

    //Передвижение персонажа с использованием компонента Rigidboy2D
    private void FixedUpdate()
    {
        Vector2 targetVelocity = new Vector2(horizontalMove * 10f,
rb.velocity.y);
        rb.velocity = targetVelocity;
    }

    //Поворот персонажа
    private void Flip()
    {
        facingRight = !facingRight;

        Vector3 theScale = transform.localScale;
        theScale.x *= -1;
        transform.localScale = theScale;
    }

    //Проверка того, находится ли игрок на земле
    private void CheckGround()
    {
        Collider2D[] colliders = Physics2D.OverlapCircleAll(new
Vector2(transform.position.x, transform.position.y + checkGroundOffsetY),
checkGroundRadius);

        if (colliders.Length > 1)
        {
            isGraunded = true;
        }
        else
        {
            isGraunded = false;
        }
    }

    //Касание врага через коллайдер
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.tag == "Enemy")
        {
            SceneManager.LoadScene(1);
        }
    }
}
```

Листинг А.2 – Исходный код скрипта PlayerCombat

```
using UnityEngine;
using UnityEngine.UI;

public class PlayerCombat : MonoBehaviour
{
    //Объявление переменных
    public Animator anim;

    public Transform attackPoint;

    public float attackRange;
    public LayerMask enemyLayer;

    public Text scoreText;

    public GameObject attackSound;

    public int score;

    //Получение компонентов
    private void Start()
    {
        scoreText.text = score.ToString();
    }

    //Осуществление атаки при нажатии на клавишу «K»
    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.K))
        {
            Instantiate(attackSound, gameObject.transform.position,
Quaternion.identity);
            Attack();
        }
    }

    //Атака главного персонажа
    public void Attack()
    {
        anim.SetTrigger("attack");

        Collider2D[] hitEnemies =
Physics2D.OverlapCircleAll(attackPoint.position, attackRange, enemyLayer);

        foreach (Collider2D enemy in hitEnemies)
        {
            enemy.GetComponent<Enemy>().EnemyDestroy();
            print("sefd");
            score++;
            scoreText.text = score.ToString();
        }
    }

    //Отображение круга attackRange
    private void OnDrawGizmosSelected()
    {
        if (attackPoint == null)
            return;
```

```
        Gizmos.DrawWireSphere(attackPoint.position, attackRange);
    }
    //Бонус к общему счету за решение задачи
    public void PuzzleBonus()
    {
        score += 200;
        scoreText.text = score.ToString();
    }
}
```

Листинг А.3 – Исходный код скрипта Enemy

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

public class Enemy : MonoBehaviour
{
    //Объявление переменных
    public float speed;
    public float positionOfPatrol;

    public bool movingRight = true;
    public bool facingRight;


    public Transform point;

    private void Update()
    {
        Chill();
    }

    //Траектория движения
    public void Chill()
    {
        //Проверка того, дошел ли враг до места назначения
        if (transform.position.x > point.position.x + positionOfPatrol)
        {
            movingRight = false;
        }
        else if (transform.position.x < point.position.x - positionOfPatrol)
        {
            movingRight = true;
        }

        //Поворот врага
        if (movingRight)
        {
            transform.position = new Vector2(transform.position.x + speed *
Time.deltaTime, transform.position.y);
            Vector3 theScale = transform.localScale;
            theScale.x = 1;
            transform.localScale = theScale;
        }

        else
        {
            transform.position = new Vector2(transform.position.x - speed *
Time.deltaTime, transform.position.y);
            Vector3 theScale = transform.localScale;
            theScale.x = -1;
```

```
            transform.localScale = theScale;
        }
    }

    //Удаление врага со сцены
    public void EnemyDestroy()
    {
        gameObject.SetActive(false);
    }
}
```

Листинг А.4 – Исходный код скрипта GameControl

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

//Логическая задача в виде пазла
public class GameControl : MonoBehaviour
{
    //Объявление переменных
    [SerializeField]
    private Transform[] pictures;
    public static bool youWin;
    private PlayerCombat player;
    public GameObject puzzlePanel;
    private int receidBonus = 0;

    //Получение компонентов
    private void Start()
    {
        player =
GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerCombat>();
        youWin = false;
    }

    //Проверка положения пазла
    private void Update()
    {
        if (pictures[0].rotation.z == 0 &&
            pictures[1].rotation.z == 0 &&
            pictures[2].rotation.z == 0 &&
            pictures[3].rotation.z == 0 &&
            pictures[4].rotation.z == 0 &&
            pictures[5].rotation.z == 0 )
        {
            puzzlePanel.SetActive(false);
            Time.timeScale = 1f;
            if (receidBonus == 0)
            {
                player.PuzzleBonus();
                receidBonus = 1;
            }
        }
    }
}
```

Листинг А.5 – Исходный код скрипта TouchRotate

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

//Поворот частей пазла
```

```
public class TouchRotate : MonoBehaviour
{
    public void Rotate()
    {
        if (!GameControl.youWin)
            transform.Rotate(0f, 0f, 90f);
    }
}
```

Листинг А.6 – Исходный код скрипта ChestLogic

```
using UnityEngine;

public class ChestLogic : MonoBehaviour
{
    public Animator anim;

    public GameObject puzzlePanel;

    //Проверка на касание игроком объекта
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.tag == "Player")
        {
            anim.SetTrigger("open");

        }
    }

    public void Puzzle()
    {
        puzzlePanel.SetActive(true);
    }
}
```

Листинг А.7 – Исходный код скрипта VolumeValue

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

//Настройка громкости
public class VolumeValue : MonoBehaviour
{
    //Объявление переменных
    public AudioSource audioSource;
    private float musicVolume = 1f;

    private void Update()
    {
        audioSource.volume = musicVolume;
    }

    public void SetVolume(float volume)
    {
        musicVolume = volume;
    }
}
```

Листинг А.8 – Исходный код скрипта QualityLogic

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

```
//Настройка графики
public class QualityLogic : MonoBehaviour
{
    public void SetQuality(int qualityIndex)
    {
        QualitySettings.SetQualityLevel(qualityIndex);
    }
}
```

Листинг А.9 – Исходный код скрипта SceneLogic

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

public class SceneLogic : MonoBehaviour
{
    //Переключение сцен
    private void Start()
    {
        Time.timeScale = 1.0f;
    }
    public void Menu()
    {
        SceneManager.LoadScene(0);
    }

    public void Play()
    {
        SceneManager.LoadScene(1);
    }

    public void Pause()
    {
        Time.timeScale = 0f;
    }

    public void Contuine()
    {
        Time.timeScale = 1f;
    }

    public void Exit()
    {
        Application.Quit();
    }
}
```

Листинг А.10 – Исходный код скрипта SavePosition

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

public class SavePosition : MonoBehaviour
{
    private float playerX;
    private float playerY;

    private Vector2 spawnPosition;

    //Получение сохраненных координатов
    private void Start()
```

```
    {
        playerX = PlayerPrefs.GetFloat("x");
        playerY = PlayerPrefs.GetFloat("y");

        spawnPosition = new Vector2(playerX, playerY);

        gameObject.transform.position = spawnPosition;
    }

    //Сохранение местоположения
    public void Save()
    {
        playerX = gameObject.transform.position.x;
        playerY = gameObject.transform.position.y;

        PlayerPrefs.SetFloat("x", playerX);
        PlayerPrefs.SetFloat("y", playerY);
    }
}
```