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 компьютерной игры «Night of the Knight» на платформе Unity

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-534.ВКР

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

Автор работы,
студент группы КЭ-402
_____ Т.А. Алейников

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
Алейникову Тихону Алексеевичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка компьютерной игры «Night of the Knight» на платформе Unity.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**
3.1. Среда разработки Unity. [Электронный ресурс] URL: https://unity.com/ru (дата обращения: 22.01.2024 г.).
3.2. Хокинг Дж. Unity в действии. Мультиплатформенная разработка на C#. – СПб: Питер, 2016. – 336 с.
4. **Перечень подлежащих разработке вопросов**
4.1. Анализ предметной области и обзор аналогов.
4.2. Проектирование игрового приложения.
4.3. Реализация игрового приложения.
4.4. Тестирование игрового приложения.
5. **Дата выдачи задания:** 29.01.2024 г.

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

Задание принял к исполнению Т.А. Алейников

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

21 век стал веком стремительного развития информационных технологий. Благодаря им, люди обрели возможность быстро решать поставленные задачи, экономить время и достигать максимально комфортного уровня жизни. Современные смартфоны, планшеты и другие гаджеты люди носят с собой повсюду, поэтому мобильные технологии стали неотъемлемой частью нашей жизни.

Развитие компьютерных технологий и их всеобщая доступность привели к росту популярности их использования не только в образовательных и вычислительных целях, но и в целях развлечений, таких как: просмотр фильмов, прослушивание музыки или общение в социальных сетях. Однако, наибольшую распространенность среди развлечений занимают видеоигры.

Компьютерные игры прочно вошли в жизнь человека, заняв почетное место лидера среди множества способов организации молодежного отдыха. Виртуальная реальность манит своими безграничными возможностями, от которых просто невозможно отказаться. Дети с самых малых лет не представляют свою жизнь без компьютера, планшета и телефона, которые, как правило, используют для видеоигр.

Согласно исследованию Entertainment Software Association (ESA) за 2021 год, средний возраст геймера в США составляет 31 год. Более 65% американских взрослых играют в компьютерные игры, а дети и подростки составляют около 25% от общего числа игроков [1]. При этом количество игроков в мире растет с каждым годом. По данным за 2023 год, компьютерными играми увлекаются более 3 млрд человек – около 40% населения Земли [2].

Игровая индустрия росла, а с ней также рос и заработок компаний. Однако не всем компаниям большой заработок пошел на пользу.

Многие крупные разработчики начали выпускать однотипные проекты, беря за основу свои предыдущие игры, зарабатывая на названии любимых игроками проектов. Это, в итоге, негативно сказалось на мнении публики и привело к сокращению продаж, а в последствии и к недовольству игровых инвесторов [3]. Рынок крупных игр значительно просел, а качество выходящих проектов стало желать лучшего. С связи с этим люди начали все чаще присматриваться к играм от независимых разработчиков и маленьких студий.

На различных крупных презентациях все чаще появляются независимые студии и разработчики. Их стали приглашать на выставки и шоу, что позволило им получать популярность и завоевывать новую аудиторию. Игры от таких разработчиков начали все чаще получать награды, они встали на одном уровне с крупными проектами от именитых студий. На их примере можно увидеть, что даже не имея крупного бюджета и большой команды разработчиков, сделать проект, который будет востребован среди игроков, совершенно возможно.

Основываясь на этом, можно сделать вывод, что на данный момент разработка компьютерных игр является актуальным направлением.

Постановка задачи

Целью выпускной квалификационной работы является разработка компьютерной 2D-игры в жанре платформер на платформе Unity. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ предметной области и обзор аналогов;
2) спроектировать игровое приложение;
3) проанализировать игровое приложение;
4) провести тестирование игрового приложения.

Структура и содержание работы

Работа состоит из введения, пяти глав, заключения, списка литературы и приложений. Объем работы составляет 47 страниц, объем списка литературы – 18 источников.

В первой главе описывается жанр разрабатываемой игры и свойственные ему особенности. Проводится обзор существующих аналогов. В ходе обзора выделяются особенности, достоинства и недостатки игр в жанре платформер. Также рассматриваются программные средства разработки для реализации разрабатываемого игрового приложения.

Вторая глава посвящена проектированию разрабатываемой компьютерной игры. В ней описываются функциональные и нефункциональные требования для разрабатываемого игрового приложения. Определена архитектура приложения, приведен эскизный проект для более четкого понимания концепции игры, определены варианты использования системы и составлены макеты интерфейса.

В третьей главе содержится описание архитектуры разрабатываемой системы. Приведены диаграммы вариантов использования и компонентов, а также макеты пользовательского интерфейса.

В четвертой главе описана реализация компонентов игрового приложения с приведенным исходным кодом игры.

В пятой главе приводятся описания и результаты тестирований реализованного игрового приложения.

В заключении описываются основные результаты, полученные при выполнении выпускной квалификационной работы, и приведены направления дальнейшего развития игры.

В приложениях приведены спецификация вариантов использования, исходный код для меню настроек и скриншоты с изображением итогового интерфейса игры.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области

В качестве жанра для разработки игры для данной работы был выбран жанр платформер. Платформер – это жанр игр, в котором основной процесс игры составляют бег и прыжки по платформам, этажам, выступам, лестницам или другим объектам, преодоление ловушек, сбор предметов и нередко бои с противниками и решение задач [4].

В большинстве случаев платформеры выполнены в 2D стилистике, а уровни представляются в виде вертикально и горизонтально прокручиваемого мира. Главной задачей в играх данного жанра является то, что главному герою необходимо пройти от начальной точки до конечной, попутно собирая предметы, побеждая врагов, решая побочные задачи и, конечно, избегая гибели персонажа.

Платформеры часто используют прыжки и лазание, чтобы ориентироваться в среде игрока и достигать своей цели. Уровни и окружающая среда, как правило, имеют неровную местность и подвешенные использования платформы разной высоты, что требует возможностей персонажа игрока для перемещения. Другие акробатические маневры также часто влияют на игровой процесс, например, лазание, раскачивание от таких предметов, как виноградные лозы или зацепы, прыжки со стен, рывок в воздухе, скольжение по воздуху, выстрел из пушки или прыгающие с трамплина или батута. Эти механики, даже в контексте других жанров, обычно называют платформингом.

Жанр платформер все еще коммерчески жизнеспособен, и ряд игр продается миллионными тиражами. С 2010 года множество бесконечно работающих платформ для мобильных устройств вернули жанру популярность [5].

Данный жанр, как и другие, не стоит на месте и развивается. В каждой своей игре разработчики пытаются внести в него что-то новое: кто-то делает

уникальный стиль игры, кто-то добавляет интересные игровые механики, дополняющие уже привычные всем игрокам.

1.2. Сравнительный анализ аналогов

Перед началом работы над проектом необходимо изучить существующие аналоги. Данный анализ позволяет выделить их особенности, преимущества и недостатки, что в дальнейшем поможет реализовать собственный проект.

Ori and the Blind Forest

Ori and the Blind Forest [6] – игра в жанрах платформер и метроидвания, разработанная студией Moon Studios [7]. Она обладает захватывающим сюжетом и отличным графическим оформлением. Скриншот из игры представлен на рисунке 1.

Рисунок 1 – Скриншот игры Ori and the Blind Forest

Герой, которым управляет игрок, изначально умеет только прыгать, но в процессе игры он сможет научиться карабкаться по стенам, нырять под воду, парить в воздухе, совершать двойные-тройные прыжки и использо-

вать энергию, чтобы выстреливать собой или отталкивать врагов и предметы. При этом игровой мир открыт и по мере изучения новых способностей, игрок получит возможность оказываться в местах, до которых прежде он добраться бы никак не смог.

К достоинствам игры можно отнести разнообразие игровых механик, простоту и сказочность ее сюжета, хорошо проработанный визуальный стиль игры, а также грамотно вписанное в сюжет исследование мира.

К недостаткам игры можно отнести нужду часто и быстро реагировать на противников и ловушки, а также резко управлять персонажем, что может быть утомительным для игрока.

Super Mario Bros

Super Mario Bros [8] – видеоигра в жанре платформер. Она была разработана и выпущена японской компанией Nintendo [9] в 1985 году для платформы Famicom. После успеха игры ее главный герой, Марио, стал символом компании Nintendo и одним из самых известных игровых персонажей в мире. Скриншот из игры представлен на рисунке 2.

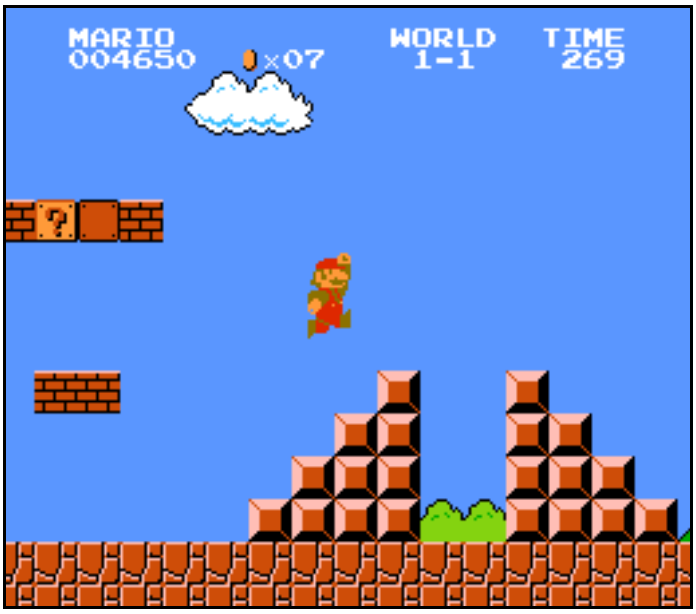

Рисунок 2 – Скриншот игры Super Mario Bros

Главный герой игры, Марио, является водопроводчиком в красном костюме. Главная задача игры – спасение принцессы. Чтобы ее спасти, нужно

дойти от точки А в точку Б. В игре имеются препятствия, которые игрок должен преодолеть. Это и платформы, и ловушки, и различные враги. В каждом уровне имеются по-своему уникальные виды этих препятствий. К тому же, главный герой способен разбивать блоки, подпрыгивая под ними и ударяясь головой. И хоть визуально блоки не отличаются друг от друга, из некоторых из них после разбиения могут выпасть бонусы, дающие улучшения для героя или пополняющие здоровье.

В связи с техническими ограничениями, имевшими место во время производства игры, сама игра была выполнена в пиксельном стиле. Однако мир игры все равно получился очень интересным и красочным. Каждая локация имеет отличные друг от друга стили, разные цвета платформ, разные задние фоны, разных врагов. Дизайны врагов до сих пор используются в играх Nintendo, посвященных вселенной игр про Марио.

К достоинствам игры можно отнести запоминающихся персонажей, разнообразный дизайн уровней и вариативность их прохождения.

К недостаткам игры можно отнести однообразный геймплей и высокую сложность игры для новых игроков.

Rayman Legends

Rayman Legends [10] – компьютерная игра в жанре платформер из игровой серии Rayman, рассказывающая о приключениях Рэймана и его друзей. Их цель – спасти пленников (10 принцесс и Малюток) от армии кошмаров. Данная игра была разработана компанией Ubisoft Monpellier [11]. Является прямым продолжением игры Rayman Origins. Скриншот игры приведен на рисунке 3.

Геймплей игры не имеет существенных отличий от геймплея предыдущей игры – главный герой спасает сказочные миры от наводнивших их кошмаров: вместе с друзьями он скачет по платформам и уворачивается от шипов, собирает люмов, освобождает из заточения малюток и входит в комнаты-испытания для решения остроумных головоломок. Однако, в игре есть много дополнительного контента – многопользовательские испытания,

мини-игры и переделанные уровни из Origins (40 штук). Сложность в игре вариативна: начало игры – относительно легкое для игрока, но ближе к концу ему приходится очень постараться, чтобы пройти уровень, не говоря уже о том, чтобы собрать все бонусы, темп происходящего многократно возрастет.

Рисунок 3 – Скриншот игры «Rayman Legends»

Благодаря уникальному стилю, интенсивному игровому процессу и умело проработанному дизайну, игра стала куда более популярной и любимой игроками, чем ее предшественник.

К достоинствам игры можно отнести тщательно продуманный дизайн, музыкальность игры, а также огромное количество дополнительного контента, придающего игре разнообразия.

Из недостатков можно выделить необходимость в хорошей реакции и способности учить уровни наизусть, что сильно сказывается при игре вдвоем, когда у одного из игроков банально не хватает игрового опыта и скорости реакции.

1.3. Обзор программного средства разработки

Для упрощения разработки игр существуют специализированные программные обеспечения – игровые движки. Они позволяют добавлять и рисовать уровни, элементы и персонажей, прописывать события, которые происходят в зависимости от действий, совершаемых главным героем. При этом разработчик, использующий игровой движок, занимается реализацией своих непосредственных идей, не растрачивая время на написание сотен строчек кода для очень многих рутинных моментов.

Игровые движки бывают разными, какие-то открытые для общего пользования, а какие доступны только для закрытого пользования. В данном разделе будут рассмотрены варианты игровых движков, доступных каждому желающему. Среди них будет выбран лучший для разработки игрового приложения.

Unreal Engine

Unreal Engine [12] – игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Первоначально он был предназначен для разработки шутеров от первого лица. Несмотря на это, его последующие версии успешно применялись в играх самых различных жанров, в том числе стелс-играх, файтингах и массовых многопользовательских ролевых онлайн-играх. Компания Epic Games 5 апреля 2022 года открыла доступ к игровому движку Unreal Engine 5 для всех разработчиков.

В данном движке используется язык программирования C++. При этом на нем имеется встроенный язык визуального программирования – Blueprints. Для более быстрого написания кода программисты обычно совмещают оба языка.

Unreal Engine может быть использован как для создания 2D, так и 3D игр. Но основное его направление – это трехмерные проекты.

Движок представляет несколько значительных усовершенствований в технологии рендеринга и инструментах для разработчиков, призванных

расширить границы возможного в мире нереального. Главной отличительной особенностью Unreal Engine является наличие у него эксклюзивных технологий, таких как Nanite и Lumen.

Nanite – технология виртуализированных микрополигонов, позволяющая добавлять высококачественные и детализированные объекты кинематографического качества непосредственно в движок с минимальной потерей детализации. Она позволяет показывать в кадре столько геометрии, сколько видит глаз, и зависит от разрешения: чем оно больше, тем выше становится детализация.

Lumen – это решение для глобального освещения, позволяющее создавать динамичное и реалистичное освещение в реальном времени, что обеспечивает более реалистичное взаимодействие света с поверхностями.

Игры, сделанные на Unreal Engine, могут быть портированы на все доступные платформы, но на мобильных устройствах встретить игры на данном движке достаточно сложно. Проекты, сделанные на Unreal Engine, требуют больших ресурсов, поэтому в основном его используют для созданий крупных проектов для компьютеров или консолей.

К достоинствам данного движка можно отнести большое количество встроенных технологий, возможность использовать визуальное программирование, а также полностью бесплатный функционал.

Из недостатков можно выделить то, что на данном движке из языков программирования доступен только C++, а также отсутствие документации на русском языке.

Unity

Unity [13, 14] – кроссплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие.

Доступные языки программирования, используемые при работе с движком – C# и JavaScript.

Unity подходит как для работы над 2D играми, так и над 3D играми. Большинство компонентов движка сделаны специально для работы над 2D или 3D играми. Это значительно уменьшает количество ошибок и облегчает работу над играми. Unity обладает всеми необходимыми средствами для удобной разработки игр. Например, в нем есть свой физический движок, который можно гибко настраивать прямо в инспекторе.

На данный момент Unity является самым популярным бесплатным движком программирования. На нем работает огромное количество разработчиков. Из-за этого у движка имеется огромное количество модификаций и уроков от сообщества. При чем Unity можно использовать для разработки не только игр, но и приложений с дополненной реальностью, таких как путеводитель, справочник, энциклопедия и многое другое.

К достоинствам движка Unity можно отнести наличие понятной официальной документации, кроссплатформенность движка, наличие множества уроков от сторонних разработчиков, огромное количество плагинов, разработанных ими, а также то, что основным языком программирования является язык C#.

Из недостатков движка Unity можно выделить отсутствие визуального программирования, не всегда корректную работу физического движка, а также дорогую стоимость платных планов движка.

2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОЙ СИСТЕМЕ

2.1. Требования к программной системе

Важную роль в разработке проекта играет анализ требований. Цель анализа требований – определить потребности пользователей и перевести их в конкретные, измеримые и достижимые требования, которые команда разработчиков программного обеспечения может использовать для проектирования и разработки системы. От его полноты и качества зависит успех проекта. Требования могут быть функциональными и нефункциональными.

Функциональные требования определяют действия, которые должна выполнять система, без учета ограничений, связанных с ее реализацией, то есть определяют поведение системы в процессе обработки информации.

Нефункциональные требования определяют свойства, которые система должна демонстрировать, или ограничения, которые она должна соблюдать, не относящиеся к поведению системы.

В ходе проектирования игрового приложения были определены следующие функциональные и нефункциональные требования к разрабатываемой системе.

Функциональные требования к проектируемой системе

Были выделены следующие функциональные требования:

1) системой должен управлять только один актер – игрок;
2) игрок должен иметь возможность управлять персонажем с момента начала игры;
3) игрок должен иметь возможность взаимодействовать с игровым окружением;
4) должна быть реализована возможность атаки у персонажа;
5) должна быть реализована система перехода на следующие уровни;
6) должна быть реализована система жизней у персонажа;
7) должна быть реализована возможность для игрока пройти уровень заново в случае смерти персонажа.

Нефункциональные требования к проектируемой системе

Были выделены следующие нефункциональные требования:

1) игровое приложение должно быть разработано на движке Unity;

2) игровое приложение должно работать на операционной системе Windows 10 и новее;

3) для разработки игрового приложения должен использоваться язык программирования C#.

2.2. Диаграмма вариантов использования

Для проектирования системы был использован язык графического описания UML [15]. В соответствии с требованиями, определенными ранее, была построена диаграмма вариантов использования, отражающая модель взаимодействия актера «Игрок» с разрабатываемым игровым приложением. Диаграмма приведена на рисунке 4.

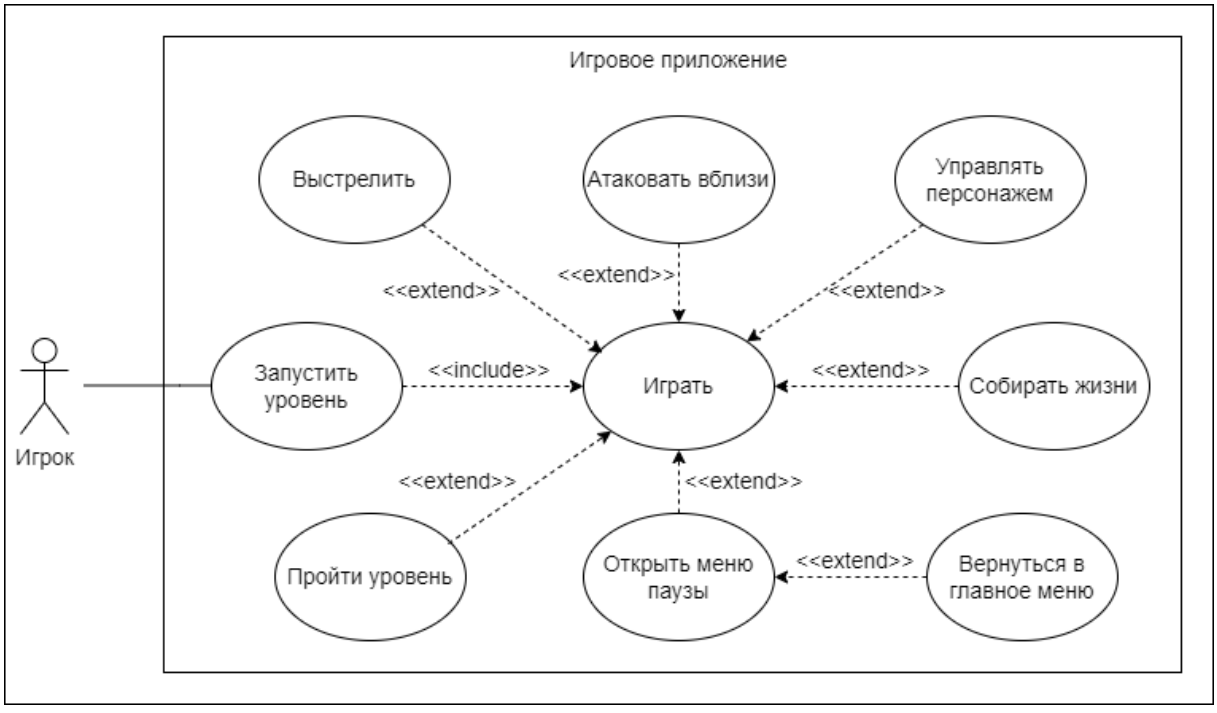

Рисунок 4 – Диаграмма вариантов использования

С системой взаимодействует только один основной актер – «Игрок», использующий игровое приложение. Задача игрока – пройти запущенный им игровой уровень до конца.

Ниже описаны действия, который может совершать игрок внутри игрового приложения.

1. «Запустить уровень» – запустить игровой уровень, нажав на кнопку «Start» в меню.

2. «Играть» – участвовать в самом игровом процессе. Во время него игроку нужно пройти уровень, управляя игровым персонажем.

3. «Открыть меню паузы» – вызвать меню паузы. В данном меню имеется возможность завершить игровой процесс, выйдя в главное меню.

4. «Выйти в главное меню» – завершить игровой процесс.

5. «Управлять персонажем» – игрок может управлять игровым персонажем с помощью клавиатуры, перемещая персонажа влево и вправо, а также совершая прыжки.

6. «Атаковать вблизи» – в процессе игры, игрок может уничтожать врагов с помощью ближней атаки.

7. «Выстрелить» – игрок может выстрелить во врага огненным шаром.

8. «Собирать жизни» – если игрок потерял одну или несколько единиц здоровья, он может восстановить его с помощью специальных бонусов, которые можно найти на уровне.

9. «Пройти уровень» – игрок получает уведомление о завершении уровня. Ему предоставляется возможность выбрать дальнейшие действия: выйти в главное меню или продолжить игру, перейдя таким образом на следующий уровень.

Спецификации основных вариантов использования приведены в таблицах 1–6 приложения А.

3. АРХИТЕКТУРА СИСТЕМЫ

3.1. Общее описание архитектуры системы

Система архитектурно состоит из сцен главного меню и уровней. При запуске игрового приложения первой сценой всегда загружается главное меню. В главном меню присутствует пользовательский интерфейс, состоящий из кнопок «Start», «Options» и «Exit». При нажатии кнопки «Start» загружается сцена первого игрового уровня, в «Options» есть возможность настроить разрешение экрана и качество картинки, а также возможность отключить звук в игре. При нажатии кнопки «Exit» игровое приложение завершает работу.

Любая сцена игрового уровня позволяет игроку перейти в меню паузы нажатием кнопки «Esc» на клавиатуре. В меню есть три кнопки – «Resume», «Turn on/off music» и «Menu». Первая кнопка позволяет выйти из меню паузы и продолжить игру. Вторая кнопка позволяет включить/выключить музыку на уровне. Третья кнопка переносит игрока в сцену главного меню, а игровой уровень закрывается.

Ниже представлена файловая система игры.

1. Animations – содержит в себе анимации игровых объектов.
2. Sounds – содержит в себе аудиофайлы, используемые игровым приложением.
3. Scenes – содержит в себе сцены главного меню и уровней.
4. Prefabs – содержит в себе готовые префабы персонажа, противников и предметов.
5. Scripts – содержит в себе скрипты, описывающие свойства и поведение объектов игры.
6. Resources – содержит в себе спрайтовые изображения (спрайты) игровых объектов.

3.2. Диаграмма компонентов

Диаграмма компонентов – это структурная диаграмма языка унифицированного моделирования, она описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами [16].

Диаграмма компонентов разрабатываемого игрового приложения приведена на рисунке 5.

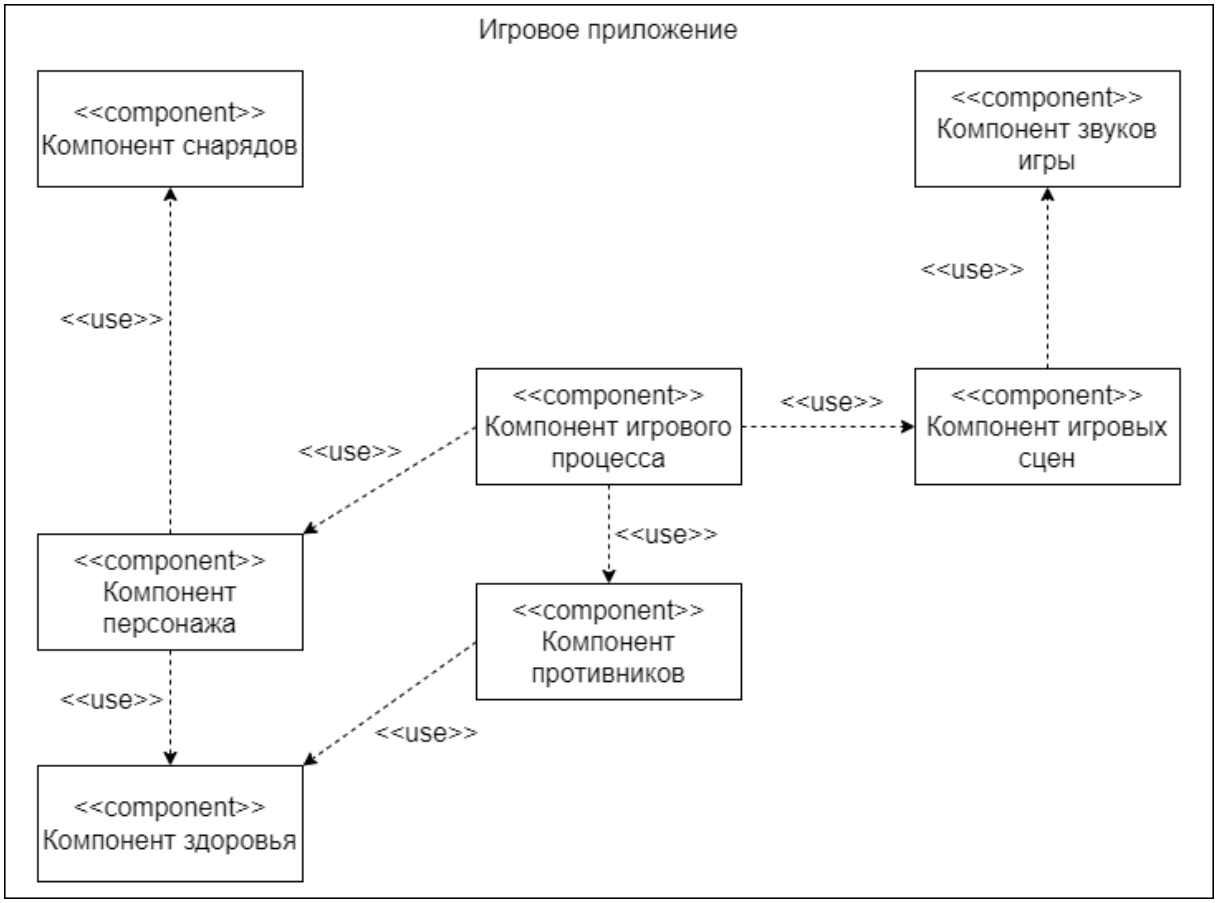

Рисунок 5 – Диаграмма компонентов игрового приложения

Компонент игрового процесса содержит набор классов, реализующих игровой процесс системы: управление персонажем, вызов отображения анимации, систему здоровья, игровую логику, задний фон и прочее.

Компонент игровых сцен содержит набор классов, реализующих главное меню, меню паузы, переключение между данными меню. А также отвечает за переход между сценами.

Компонент звуков игры содержит набор классов, реализующих систему звуков и добавление музыки для уровней, а также возможность отключить/включить музыку на уровне.

Компонент персонажа игрока содержит класс, позволяющий настраивать персонажа игрока, его внешний вид и свойства, например, его силу прыжка, скорость передвижения персонажа, вызов анимации при определенном действии.

Компонент противников содержит классы, описывающие поведение различных противников. Эти классы реализуют скорость передвижения противников, зону их передвижения, наносимый урон.

Компонент снарядов содержит набор классов, описывающий поведение, свойства и параметры снарядов для игрового персонажа.

Компонент здоровья содержит класс, описывающий и задающий параметры здоровья, а также вычитания здоровья при получении урона.

3.3. Представление пользовательского интерфейса

В данном разделе будут представлены макеты некоторых пользовательских интерфейсов, которые будут реализованы в игре. Данные макеты будут являться лишь примерным представлением итогового продукта и содержат в себе основные необходимые функции.

В ходе реализации или тестирования некоторые элементы интерфейсов могут быть изменены или переработаны, также может быть изменено их положение на экране.

Главное меню

В главном меню игры будут располагаться следующие элементы:

1) заголовок меню, информирующий игрока о том, что он находится в главном меню игры;

2) кнопка «Start» – запускает первый уровень игры;

3) кнопка «Options» – открывает меню настроек, где можно настроить режим экрана и качество картинки, а также включить (выключить) звуки в игре;

4) кнопка «Exit» – закрывает приложение.

Макет главного меню представлен на рисунке 6.

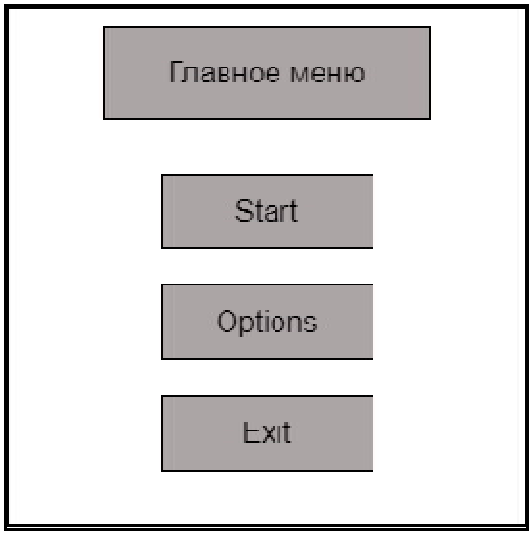

Рисунок 6 – Макет интерфейса главного меню

Меню паузы

В меню паузы будут располагаться следующие необходимые элементы:

1) кнопка «Resume» – возвращает игрока к игровому процессу;

2) кнопка «Turn on/off music» – включает/выключает музыку на уровне;

3) кнопка «Menu» – переносит игрока в главное меню.

Макет меню паузы представлен на рисунке 7.

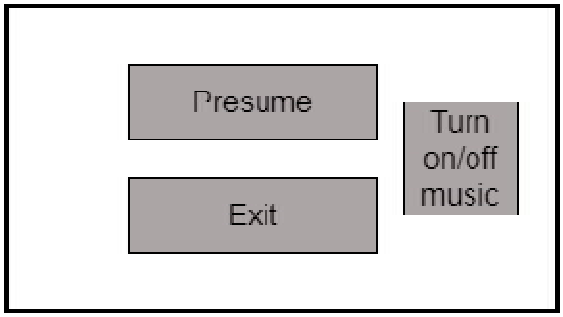

Рисунок 7 – Макет интерфейса меню паузы

Меню настроек

В меню настроек будут располагаться следующие необходимые элементы:

1) заголовок меню, информирующий игрока о том, что он находится в меню настроек;

2) кнопка «Screen resolution» – позволяет игроку выбрать расширение экрана;

3) кнопка «Full screen mode» – позволяет игроку включить (выключить) полноэкранный режим;

4) кнопка «Graphic settings» – позволяет игроку выбрать качество картинки для игры;

5) счетчик «Total Coins» – информирует игрока о том, сколько монет он собрал за всю игру;

6) кнопка «Save» – сохраняет изменения, сделанные в настройках и переносит игрока обратно в главное меню;

7) кнопка «Back» – переносит игрока обратно в главное меню без сохранения изменений.

Макет меню настроек представлен на рисунке 8.

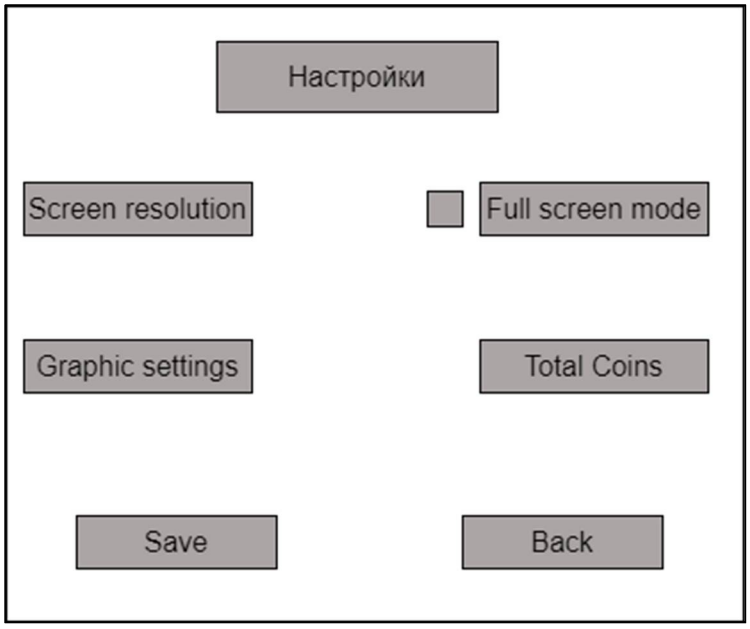

Рисунок 8 – Макет интерфейса меню настроек

Интерфейс игрового дисплея

Во время игрового процесса, когда игрок проходит игровые уровни, также есть элементы, которые необходимо отображать для игрока. Интерфейс игрового процесса в игре должен быть удобным, интуитивно понятным и эффективным.

Во время игры игрок должен видеть следующие элементы:

1) индикатор здоровья, информирующий игрока о текущем количестве здоровья у персонажа;

2) счетчик монет, показывающий количество собранных персонажем на уровне монет;

3) кнопку «Пауза», позволяющую игроку приостановить игровой процесс и перейти в меню паузы;

4) картинку «Charged», сообщающую игроку о том, что шар заряжен.

Макет игрового дисплея представлен на рисунке 9.

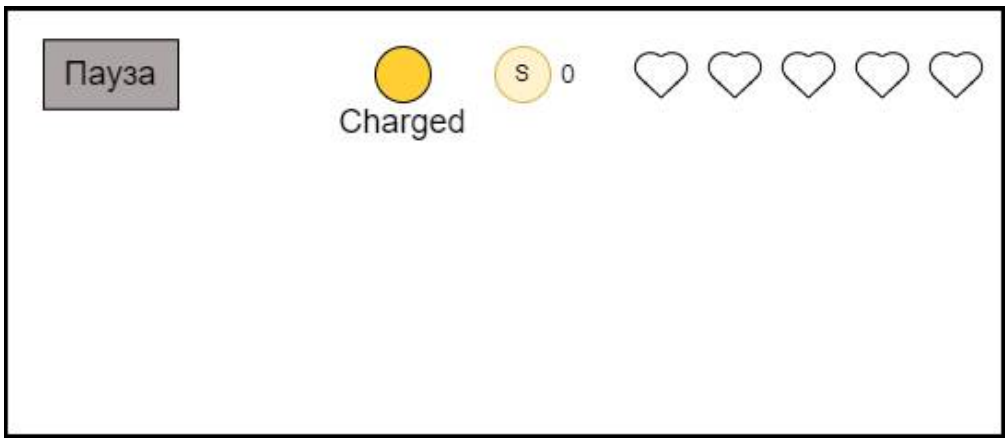

Рисунок 9 – Макет интерфейса игрового дисплея

4. РЕАЛИЗАЦИЯ СИСТЕМЫ

Для разработки игрового приложения было решено использовать игровой движок Unity. Несмотря на то, что основное направление движка – трехмерные проекты, Unity также обладает всеми необходимыми компонентами для разработки 2D-игр. Он позволяет создавать приложения, работающие под более чем 20 различными операционными системами. Различные технологии Unity позволяют делать разрабатываемые на нем игры более качественными. Их разработка при этом упрощается. Данный движок удобен тем, что имеет подробную документацию и большое количество официальных уроков и материалов по разработке игровых продуктов [17, 18].

Инструменты, используемые для разработки игрового приложения в Unity, представлены ниже.

1. Animation – инструмент, служащий для создания в Unity необходимой анимации из имеющихся у пользователя спрайтов.
2. Animator – инструмент, служащий для воспроизведения необходимой анимации в нужный момент в зависимости от действий пользователя в игре.
3. Rigidbody – инструмент, подключающий физическое поведение для объекта.
4. Box Collider 2D – компонент, реализующий столкновение для твердых 2D объектов.
5. Audio Source – компонент, используемый для воспроизведения звуков в разрабатываемом игровом приложении Unity.

Для разработки программного кода игры была выбрана среда программирования Visual Studio Code. В качестве языка программирования для разрабатываемого игрового приложения был выбран язык C#.

4.1. Файловая структура приложения

Данный проект состоит из списка каталогов. Каждый каталог содержит отдельную категорию элементов, использованных при разработке игрового приложения:

1) анимации;
2) префабы;
3) спрайты;
4) игровые сцены;
5) игровые скрипты;
6) звуки и музыка;
7) шрифты для текста.

Файловая структура приложения представлена на рисунке 10.

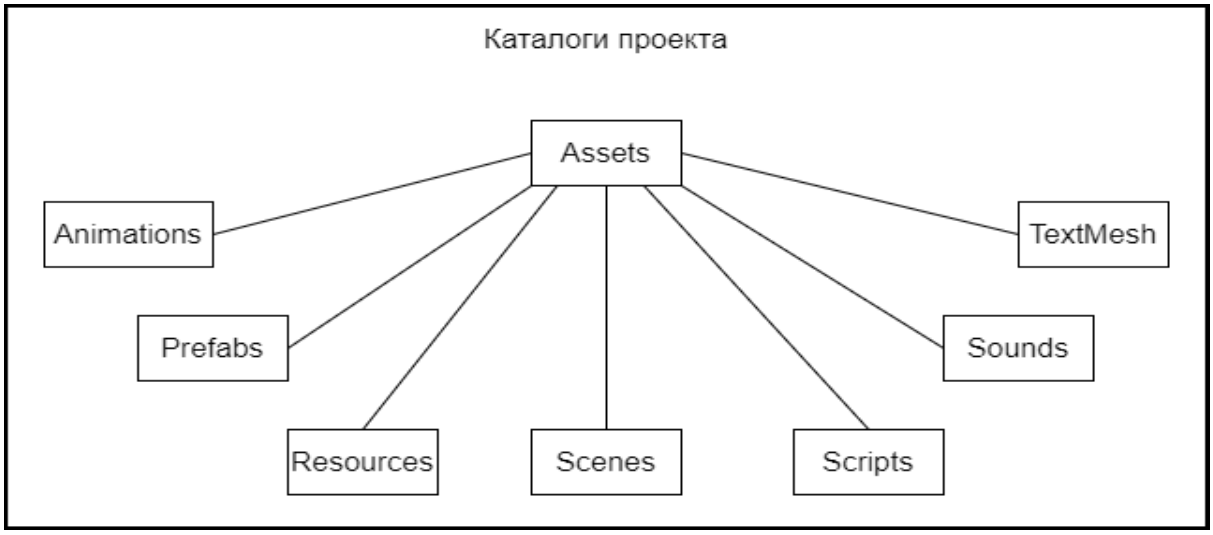

Рисунок 10 – Файловая структура

В каталоге Animations содержатся различные анимации персонажа и его врагов.

В каталоге Prefabs содержатся префабы игровых объектов.

В каталоге Resources содержатся спрайты персонажа, его врагов и используемых в игре объектов.

В каталоге Scenes содержатся игровые сцены (уровней, главного меню), необходимые для процесса игры.

В каталоге Scripts содержатся скрипты, используемые в игровом приложении.

В каталоге Sounds содержатся аудиофайлы, используемые в игровом приложении.

В каталоге TextMesh содержатся шрифты, используемые для текста в игре.

4.2. Реализация компонента персонажа

Данный компонент содержит в себе классы, описывающие свойства, поведение персонажа и его характеристики.

Основным классом является класс `Hero`. В данном классе реализовано передвижение игрового персонажа, возможность получения персонажем урона, восстановления здоровья, взаимодействия с игровыми объектами. Реализован вызов соответствующих анимаций при совершении персонажем различных действий, такие как бездействие, ходьба, прыжок, атака.

Для реализации функций перемещения и прыжка (листинг 1) был использован компонент `Rigidbody`. Реализация функции атаки приведена в листинге 2.

Листинг 1 – Реализация перемещения и прыжка персонажа

```
private void Run()
{
     if (isGrounded) State = States.run;

      dir.x = Input.GetAxis("Horizontal");
      transform.position = Vector3.MoveTowards(transform.position, trans-
form.position + dir, speed * Time.deltaTime);
      if (dir.x < 0 && FacingRight)
      {
        Flip();
      }
      else if (dir.x > 0 && !FacingRight)
      {
        Flip();
      }
}
private void Jump()
{
      rb.AddForce(transform.up * jumpForce, ForceMode2D.Impulse);
      PlaySound(sounds[0]);
}
```

Листинг 2 – Реализация атаки персонажем врагов

```
private void Attack()
{
      if(isGrounded && isRecharged)
      {
            State = States.attack;
            isAttacking = true;
            isRecharged = false;
            StartCoroutine(AttackAnimation());
            StartCoroutine(AttackCoolDown());
      }
             PlaySound(sounds[1]);
}
private void OnAttack()
{
      Collider2D[] colliders = Physics2D.OverlapCircleAll(attackPos.posi-
tion, attackRange, enemy);
      for (int i = 0; i < colliders.Length; i++)
      {
          colliders[i].GetComponent<Entity>().GetDamage();
      }
}
```

В классах `Hero` и `Health` реализована возможность подбирать предметы на уровне. В классе `Hero` реализована возможность собирать монеты, в то время как в классе `Health` реализована возможность восстановить здоровье персонажа при подборе соответствующего бонуса. Реализация данной возможности представлены в листингах 3 и 4.

Листинг 3 – Реализация сбора монет

```
Hero.cs
private void OnTriggerEnter2D(Collider2D coll)
{
      PlaySound(sounds[5]);
      if (coll.gameObject.tag == "Coin")
      {
        coins++;
        coinsText.text = coins.ToString();
        AddToTotalCoins(1);
        Destroy(coll.gameObject);
      }
}
```

Листинг 4 – Реализация сбора бонуса лечения

```
Health.cs
private void OnCollisionEnter2D(Collision2D collision)
      {
        if (collision.gameObject == Hero.Instance.gameObject)
           {
              Hero.Instance.SetHealth();
              Destroy(gameObject);
           }
      }
```

В классе Hero также реализована функция, определяющая, находится ли игрок на данный момент на земле. Это необходимо, чтобы знать, можно ли выполнить прыжок в данный момент. Реализация функции представлена в листинге 5.

Листинг 5 – Функция определения нахождения игрока на земле

```
private void CheckGround()
{
    Collider2D[] collider = Physics2D.OverlapCircleAll(transform.position,
0.3f);
    isGrounded = collider.Length > 1;
    if (!isAttacking && !isGrounded) State = States.jump;
}
```

При соприкосновении с врагом игрок получает урон. Для реализации данной функции был использован компонент Box Collider 2D. Код, дающий врагам возможность наносить игроку урон при соприкосновении с ним, приведен в листинге 6.

Листинг 6 – Реализация нанесения врагами урона игроку

```
private void OnCollisionEnter2D(Collision2D collision)
{
     if (collision.gameObject == Hero.Instance.gameObject)
     {
         Hero.Instance.GetDamage();
     }
}
public void GetDamage()
{
    health -= 1;
    PlaySound(sounds[2]);
}
```

В классе Hero, помимо функций управления персонажем, реализовано отображение количества здоровья и количества монет, собранных игроком за уровень. Код представлен в листинге 7.

Листинг 7 – Реализация отображения количества здоровья и монет

```
private void OnTriggerEnter2D(Collider2D coll)
{
    PlaySound(sounds[5]);
    if (coll.gameObject.tag == "Coin")
    {
        coins++;
        coinsText.text = coins.ToString();
        AddToTotalCoins(1);
        Destroy(coll.gameObject);
     }
}
```

```
private void FixedUpdate()
{
     CheckGround();
     if (health > numOfHearts)
     {
         health = numOfHearts;
     }
     for (int i = 0; i < hearts.Length; i++)
     {
         if(i < Mathf.RoundToInt(health))
         {
            hearts[i].sprite = fullHeart;
         }
         else
         {
             hearts[i].sprite = emptyHeart;
           }
           if(i < numOfHearts)
           {
              hearts[i].enabled = true;
          }
          else
          {
             hearts[i].enabled = false;
            }
        }
}
```

4.3. Реализация компонента выстрела снарядом

В классе `Shot` для персонажа была реализована дополнительная способность выстреливать огненным шаром в врагов. После выстрела снаряд заряжается в течение определенного времени. Свойство зарядки снаряда было реализовано в классе `Weapon`. Выстрелить им снова игрок может только когда шар заряжен. Реализация выстрела и перезарядки представлены в листингах 8 и 9.

Листинг 8 – Реализация выстрела снарядом

```
public float speed;
public float lifetime;
private float flyTimer;
public float distance;
public int damage;
Rigidbody2D rb;
public LayerMask whatIsSolid;
public GameObject hitEffect;

private void Awake()
    {
       flyTimer = lifetime;
    }
private void Update()
    {
       flyTimer -= Time.deltaTime;
       if(flyTimer <= 0)
       {
```

```
           Destroy(gameObject);
        }

        RaycastHit2D hitInfo = Physics2D.Raycast(transform.position, trans-
form.up, distance, whatIsSolid);
       if(hitInfo.collider != null)
       {
          if(hitInfo.collider.CompareTag("Enemy"))
          {
               hitInfo.collider.GetComponent<Enemy>().TakeDamage(damage);
          }
                 Instantiate(hitEffect, transform.position, Quaternion.iden-
tity);
               Destroy(gameObject);
       }
       transform.Translate(Vector2.right * speed * Time.deltaTime);
     }
}
```

Листинг 9 – Реализация перезарядки снаряда

```
public class Weapon : Sounds
{
public Transform firePoint;
public GameObject shotPrefab;
public Image chargeImage;
public Sprite chargedSprite; // Спрайт для заряженного состояния
public Sprite firedSprite; // Спрайт для состояния после выстрела
private float timeBtwShots;
public float startTimeBtwShots;

    private void Start()
    {
        chargeImage.gameObject.SetActive(true); // Активируем картинку при
старте
        chargeImage.sprite = chargedSprite;
    }
    void Update()
    {
         if (timeBtwShots <= 0) {
             chargeImage.gameObject.SetActive(true);
             chargeImage.sprite = chargedSprite;
            if(Input.GetButtonDown("Fire2"))
            {
                Instantiate(shotPrefab, firePoint.transform.position, fire-
Point.rotation);
                timeBtwShots = startTimeBtwShots;
                PlaySound(sounds[0]);

                // Меняем картинку на спрайт после выстрела
                chargeImage.sprite = firedSprite;
            }
         }
         else
         {
            timeBtwShots -= Time.deltaTime;
         }
     }
}
```

4.4. Реализация компонента меню паузы

В классе `PauseMenu` реализована функция паузы в игре. Метод `Pause` устанавливает флаг паузы, останавливает игровое время и отображает меню паузы. Метод `Resume` снимает флаг паузы, восстанавливает игровое время и скрывает меню паузы. Метод `LoadMenu` переносит игрока в главное меню. Реализация меню паузы представлена в листинге 10.

Листинг 10 – Реализация вызова меню паузы

```
public void Resume()
{
     pauseGameMenu.SetActive(false);
     Time.timeScale = 1f;
     PauseGame = false;
}
public void Pause()
{
     pauseGameMenu.SetActive(true);
     Time.timeScale = 0f;
     PauseGame = true;
}
public void LoadMenu()
{
     Time.timeScale = 1f;
     SceneManager.LoadScene("Menu")
}
```

4.5. Реализация компонента главного меню и меню настроек

Класс `MainMenu` отвечает за работу логики игрового меню. В нем реализована обработка кнопки для загрузки сцены с игрой, выхода из игры, а также удаления ключей, связанных со здоровьем и монетами при нажатии на кнопку «Start» для сброса сохранений количества здоровья и монет из предыдущей игры.

Реализация главного меню представлена в листингах 11 и 12.

Листинг 11 – Реализация свойств кнопок «Start» и «Exit»

```
public void PlayGame()
{
     ResetHealth();
     ResetCoins();
     SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
}
public void ExitGame()
{
     Debug.Log("Игра закрылась");
     Application.Quit();
}
```

Листинг 12 – Реализация сброса значений количества здоровья и монет

```
public void ResetHealth()
{
     PlayerPrefs.DeleteKey("PlayerHealth"); // Удаление ключа, связанного со здоровьем
     PlayerPrefs.Save();     // Вызов PlayerPrefs.Save(), чтобы изменения вступили в силу
}
public void ResetCoins()
{
// Удаление ключа, связанного со монетами
     PlayerPrefs.DeleteKey("TotalCoins");
     PlayerPrefs.Save();
}
```

Компонент настроек содержится в классе `Settings`. Он отвечает за настройку расширения экрана и качества графики и их сохранение. Также в данном классе реализовано выведение на экран количества собранных за всю игру монет.

Реализация всех вышеперечисленных параметров настроек представлена в листинге 1 приложения Б.

4.6. Реализация компонента музыки и звуков в игре

Для воспроизведения музыки и звуков в игровом приложении был использован компонент `AudioSource`. С помощью класса `Sounds` упрощено добавление звуков для каждого действия персонажа, управляемого игроком. Реализация данного класса представлена в листинге 13.

Листинг 13 – Реализация системы добавления звуков

```
public class Sounds : MonoBehaviour
{
     public AudioClip[] sounds;
     private AudioSource audioSrc => GetComponent<AudioSource>();
     public void PlaySound(AudioClip clip, float volume = 1f, bool destroyed = false,
float p1 = 0.85f, float p2 = 1.2f)
     {
        audioSrc.pitch = Random.Range(p1, p2);
        audioSrc.PlayOneShot(clip, volume);
     }
}
```

В меню паузы также реализована кнопка, позволяющая выключить и включить повторно музыку на игровых уровнях. После начала новой игры

настройка сохраняется. За принцип ее работы отвечает класс `Music`. Реализация данного класса представлена в листинге 14.

Листинг 14 – Реализация включения/отключения музыки

```
public class Music : MonoBehaviour
{
     public Sprite onMusic;
     public Sprite offMusic;
     public Image MusicButton;
     public bool isOn;
     public AudioSource ad;
     void Start()
     {
       isOn = true;
     }

     void Update()
     {
        if (PlayerPrefs.GetInt("music") == 0)
        {
            MusicButton.GetComponent<Image>().sprite = onMusic;
            ad.enabled = true;
            isOn = true;
        }
        else if (PlayerPrefs.GetInt("music") == 1)
        {
            MusicButton.GetComponent<Image>().sprite = offMusic;
            ad.enabled = false;
            isOn = false;
        }
     }
     public void offSaund()
     {
        if(!isOn)
        {
           PlayerPrefs.SetInt("music", 0);
        }
        else if (isOn)
        {
           PlayerPrefs.SetInt("music", 1);
        }
     }
}
```

5. ТЕСТИРОВАНИЕ СИСТЕМЫ

Для проведения тестирования разработанного игрового приложения были использованы следующие методы тестирования:

1) функциональное тестирование;
2) юзабилити тестирование.

5.1. Функциональное тестирование

В ходе данного тестирования проводилась проверка на соответствие игрового приложения функциональным требованиям, которые были ему предъявлены. Результаты функционального тестирования представлены в таблице 1.

Таблица 1 – Функциональное тестирование игрового приложения

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Запуск приложения.	Запустить приложение через .exe файл.	Приложение запускается.	Да
2	Проверка запуска игрового процесса.	Нажать на кнопку «Start» в главном меню.	Начинается игровой процесс.	Да
3	Передвижение персонажа вправо.	Во время прохождения уровня нажать на клавишу «D».	Персонаж передвигается вправо.	Да
4	Передвижение персонажа влево.	Во время прохождения уровня нажать на клавишу «A».	Персонаж передвигается влево.	Да
5	Прыжок персонажа вверх.	Во время прохождения уровня нажать на клавишу «Space».	Персонаж подпрыгивает вверх без возможности повторно прыгнуть в воздухе.	Да
6	Проверка вызова меню паузы.	Во время прохождения уровня нажать на клавишу «Esc».	Появляется меню паузы.	Да

Окончание таблицы 1

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
7	Проверка работоспособности стрельбы	Во время прохождения уровня нажать на правую кнопку мыши.	Персонаж выпускает огненный шар.	Да
8	Уничтожение противников	1. Подойти к противнику 2. Атаковать его нажав левую кнопку мыши	Противник уничтожен	Да
9	Получение урона	Подойти достаточно близко к противнику или к шипу	Персонаж получает урон.	Да
10	Подбор монеты	Подойти достаточно близко к монете	Персонаж получает монету. На UI обновляется количество монет.	Да
11	Подбор цветочка	1. Получить урон 2. Подобрать цветочек	Персонаж восполняет потерянную единицу здоровья. На UI обновляется количество здоровья.	Да
12	Возрождение персонажа	После смерти персонажа на возникшем экране «Game Over» нажать на кнопку «Click to Restart»	Персонаж появляется в начале уровня с полным уровнем здоровья, все враги, предметы на уровне восстановлены.	Да

Во время функционального тестирования были выявлены и исправлены неполадки. Данные неполадки приведены ниже.

1. При прикосновении к движущейся платформе, персонаж прилипал к ней и сопровождал ее. Для решения данной проблемы в коде была добавлена возможность для персонажа «открепиться» от платформы, когда он перестает физически с ней контактировать.

2. Огненный шар, выстреливаемый героем, не взаимодействовал с землей, по которой герой ходил, просто пролетая сквозь нее. Для блоков земли был добавлен слой «solid», с помощью которого для огненного шара была добавлена возможность взаимодействовать не только с врагами, но и с блоками земли.

5.2. Юзабилити тестирование

Целью юзабилити тестирования является обеспечение удовлетворительного пользовательского опыта и максимального комфорта при использовании продукта. В ходе данного тестирования был проведен анализ того, насколько эффективно пользователи могут взаимодействовать с игровым приложением. Юзабилити тестирование проводилось при участии друзей и знакомых из интернета. Всего в нем приняли участие пять человек. Это позволило получить разнообразные мнения и отзывы, так как каждый участник имел свой уникальный опыт и предпочтения в области визуала и функционала игрового приложения.

В ходе юзабилити тестирования были выявлены технические недостатки игры, которые не были обнаружены в ходе функционального тестирования, и получены предложения по ее улучшению. Участники тестирования запускали игровое приложение при различных разрешениях экрана, которые можно менять в настройках игры, и различных соотношениях сторон монитора.

Ниже приведен список изменений, внесенных в игровое приложение после реализации.

1. Внесены изменения в дизайн уровней. Было изменено расположение некоторых блоков делало неудобными прыжки для героя, сильно ограничивая пространство для них там, где это не было нужно. Задний фон доставлял чувство дискомфорта игрокам, поэтому он был изменен на более приятный.

2. Было немного увеличено время до падения падающих блоков под ногами персонажа, чтобы игрок успевал вовремя среагировать.

3. При выстреле огненным шаром на втором уровне помимо звука выстрела издавался звук подбора монетки. Баг исправлен.

На рисунках 1–5 приложения В представлены скриншоты итоговой игрового проекта после реализации и тестирования.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была разработана компьютерная игра в жанре платформер на Unity.

Для достижения данной цели были выполнены следующие поставленные задачи.

1. Проведен анализ предметной области, в ходе которого были рассмотрены аналогичные проекты, выявлены их плюсы и минусы, которые учитывались при разработке. Были выбраны подходящие средства разработки, учитывая требования проекта и его масштаб.

2. Сформирована концепция игрового приложения. Составлена диаграмма вариантов использования и определены функциональные и нефункциональные требования к проектируемой системе.

3. Была составлена подходящая архитектура игрового приложения, описана диаграмма компонентов, а также макеты меню и интерфейса.

4. Реализовано игровое приложение в соответствии с поставленными ранее функциональными и нефункциональными требованиями.

5. Проведены функциональное и юзабилити тестирования игрового приложения, что позволило убедиться в соответствии реализованного проекта поставленным требованиям.

В результате работы был подробно изучен жанр платформер и получен огромный опыт работы с игровым движком Unity, а также с использованием языка программирования C# для разработки игры на данном движке. В дальнейшем для игрового приложения планируется расширить количество уровней, ввести разнообразные подбираемые бонусы и разработать систему сохранения прогресса. Опыт, полученный при его создании, может стать основой не только для улучшения текущего, но и для создания будущих игровых проектов.

ЛИТЕРАТУРА

1. Официальный сайт Американской ассоциации производителей ПО и компьютерных игр (ESA). Важные факты об индустрии компьютерных игр в 2021 году. [Электронный ресурс] URL: https://www.theesa.com/resource/2021-essential-facts-about-the-video-game-industry/ (дата обращения: 22.01.2024 г.).

2. Ведущая платформа для поиска и исследования статистики в области программного обеспечения. [Электронный ресурс] URL: https://financeonline.com/number-of-gamers-worldwide/ (дата обращения: 22.01.2024 г.).

3. 3Dnews: CD Projekt RED выплатит почти $2 млн по иску от недовольных инвесторов – студию обвиняли во лжи о консольных версиях Cyberpunk 2077. [Электронный ресурс] URL: https://3dnews.ru/1079889/cdprojekt-red-vyplatit-pochti-2-mln-po-isku-ot-nedovolnykh-investorov-studiyuobvinyali-vo-lzhi-o-konsolnykh-versiyakh-cyberpunk-2077/ (дата обращения: 23.01.2024 г.).

4. What is a Platform Game? [Электронный ресурс] URL: https://www.lifewire.com/what-is-a-platform-game-812371/ (дата обращения: 23.01.2024 г.).

5. Платформер – Platform game. [Электронный ресурс] URL: https://ru.wikibrief.org/wiki/Platform_game/ (дата обращения: 23.01.2024 г.).

6. Официальная страница игры Ori and the Blind Forest. [Электронный ресурс] URL: https://www.orithegame.com/blind-forest/ (дата обращения: 23.01.2024 г.).

7. Официальный сайт компании Moon Studios. [Электронный ресурс] URL: https://www.orithegame.com/blind-forest/ (дата обращения: 23.01.2024 г.).

8. Официальная страница игры Super Mario Bros. [Электронный ресурс] URL: https://www.nintendo.ru/-/NES/Super-Mario-Bros-803853.html/ (дата обращения: 24.01.2024 г.).

9. Официальный сайт компании Nintendo. [Электронный ресурс] URL: https://www.nintendo.ru/ (дата обращения: 24.01.2024 г.).

10. Официальный сайт игры «Rayman Legends». [Электронный ресурс] URL: https://www.ubisoft.com/ru-ru/game/rayman/legends/ (дата обращения: 24.01.2024 г.).

11. Официальный сайт компании Ubisoft Montpellier. [Электронный ресурс] URL: https://www.ubisoft.com/en-us/studio/montpellier/ (дата обращения: 24.01.2024 г.).

12. Официальный сайт Unreal Engine. [Электронный ресурс] URL: https://www.unrealengine.com/en-US/ (дата обращения: 24.01.2024 г.).

13. Официальная документация Unity. [Электронный ресурс] URL: https://docs.unity3d.com/ru/2019.3/ScriptReference/Sprite.html/ (дата обращения: 24.01.2024 г.).

14. Официальный сайт Unity. [Электронный ресурс] URL: https://unity.com/ (дата обращения: 24.01.2024 г.).

15. Язык UML: что это такое и зачем он нужен. [Электронный ресурс] URL: https://skillbox.ru/media/code/yazyk-uml-chto-eto-takoe-i-zachem-on-nuzhen/ (дата обращения: 05.03.2024 г.).

16. UML: обзор основных типов диаграмма компонентов. Часть 2. [Электронный ресурс] URL: https://habr.com/ru/articles/756552/ (дата обращения: 05.03.2024 г.).

17. Unity for 2D. [Электронный ресурс] URL: https://unity.com/2d-solution-guide (дата обращения: 05.03.2024 г.).

18. Unity 2D. [Электронный ресурс] URL: https://unity.com/ru/solutions/2d (дата обращения: 05.03.2024 г.).

ПРИЛОЖЕНИЯ

Приложение А. Спецификация вариантов использования

Спецификация вариантов использования (ВИ) системы приведена в таблицах 1–6.

Таблица 1 – Спецификация ВИ «Запустить уровень»

Прецедент: Запустить уровень
Краткое описание: Запуск игрового процесса.
Главные актеры: Игрок
Второстепенные актеры: Нет
Предусловия: Успешный запуск игрового приложения. Игрок находится в главном меню.
Основной поток: Прецедент начинается, когда игрок нажимает на кнопку «Start». Приложение загружает игровой уровень и начинается игровой процесс.
Постусловия: отсутствуют
Альтернативные потоки: отсутствуют

Таблица 2 – Спецификация ВИ «Играть»

Прецедент: Играть
Краткое описание: О
Главные актеры: Игрок
Второстепенные актеры: Нет
Предусловия: 1. Успешный запуск игрового уровня.
Основной поток: . Вариант использования начинается после завершения варианта использования 1. 2. Игрок управляет игровым персонажем; 3. Игрок может проходить игру или выйти из нее через меню паузы.
Постусловия: отсутствуют
Альтернативные потоки: отсутствуют

Таблица 3 – Спецификация ВИ «Управлять персонажем»

Прецедент: Управлять персонажем
Краткое описание: Управление персонажем во время прохождения уровня.
Главные актеры: Игрок
Второстепенные актеры: Нет
Предусловия: Происходит игровой процесс
Основной поток: . Вариант использования начинается с нажатия кнопок, отвечающих за управление персонажем. 2. Персонаж, управляемый игроком, движется по локации в зависимости от нажатых кнопок.
Постусловия: отсутствуют
Альтернативные потоки: отсутствуют

Таблица 4 – Спецификация ВИ «Пройти уровень»

Прецедент: Пройти уровень
Краткое описание: И
Главные актеры: Игрок
Второстепенные актеры: Нет
Предусловия: отсутствуют
Основной поток: . 2. Выводятся кнопки «Продолжить» и «Меню». 3. Игрок нажимает на кнопку «Продолжить» и загружается следующий уровень.
Постусловия: отсутствуют
Альтернативные потоки: отсутствуют

Таблица 5 – Спецификация ВИ «Открыть меню паузы»

Прецедент: Открыть меню паузы
Краткое описание: Открытие меню паузы
Главные актеры: Игрок
Второстепенные актеры: Нет
Предусловия: отсутствуют
Основной поток: . Вариант использования начинается, когда игрок нажимает на кнопку для открытия меню паузы. 2. Процесс игры приостанавливается и на экране выводится меню паузы; 3. Игрок может продолжить игру, зайти в настройки или выйти из игры.
Постусловия: отсутствуют
Альтернативные потоки: отсутствуют

Таблица 6 – Спецификация ВИ «Выйти в главное меню»

Прецедент: Выйти в главное меню
Краткое описание: Выход в главное меню.
Главные актеры: Игрок
Второстепенные актеры: Нет
Предусловия: Игрок находится в меню паузы
Основной поток: . Вариант использования начинается, когда игрок нажимает на кнопку возврата в главное меню. 2. Игроку открывается главное меню.
Постусловия: отсутствуют
Альтернативные потоки: отсутствуют

Приложение Б. Настройки игрового приложения

Реализация настроек игры представлена в листинге 1.

Листинг 1 – Настройка игры

```
public Dropdown resolutionDropdown;
public Dropdown qualityDropdown;
public TMP_Text totalCoinsText;
     Resolution[] resolutions;

void Start()
{
    resolutionDropdown.ClearOptions();
    List<string> options = new List<string>();
    resolutions = Screen.resolutions;
    int currentResolutionIndex = 0;

    for(int i = 0; i < resolutions.Length; i++)
    {
       string option = resolutions[i].width + "x" + resolutions[i].height +
" " +
resolutions[i].refreshRate + "Hz";
       options.Add(option);
       if (resolutions[i].width == Screen.currentResolution.width && reso-
lutions[i].height == Screen.currentResolution.height)
           currentResolutionIndex = i;
     }

     resolutionDropdown.AddOptions(options);
     resolutionDropdown.RefreshShownValue();
     LoadSettings(currentResolutionIndex);
     LoadTotalCoinsText();
}

void LoadTotalCoinsText()
{
    int totalCoins = PlayerPrefs.GetInt("TotalCoins", 0);
    totalCoinsText.text = totalCoins.ToString() + "/30";
}

public void SetFullscreen(bool isFullscreen)
{
     Screen.fullScreen = isFullscreen;
}

public void SetResolution(int resolutionIndex)
{
     Resolution resolution = resolutions[resolutionIndex];
     Screen.SetResolution(resolution.width, resolution.height, Screen.full-
Screen);
}
public void SetQuality(int qualityIndex)
{
     QualitySettings.SetQualityLevel(qualityIndex);
}

public void ExitSettings()
{
     SceneManager.LoadScene("Menu");
}

public void SaveSettings()
```

```
{
     PlayerPrefs.SetInt("QualitySettingPreference", qualityDropdown.value);
     PlayerPrefs.SetInt("ResolutionPreference", resolutionDropdown.value);
     PlayerPrefs.SetInt("FullscreenPreference", System.Con-
vert.ToInt32(Screen.fullScreen));
}

public void LoadSettings(int currentResolutionIndex)
{
     if (PlayerPrefs.HasKey("QualitySettingPreference"))
        qualityDropdown.value = PlayerPrefs.GetInt("QualitySettingPrefer-
ence");
     else
        qualityDropdown.value = 3;
     if (PlayerPrefs.HasKey("ResolutionPreference"))
        resolutionDropdown.value = PlayerPrefs.GetInt("ResolutionPrefer-
ence");
     else
        resolutionDropdown.value = currentResolutionIndex;
     if (PlayerPrefs.HasKey("FullscreenPreference"))
       Screen.fullScreen = System.Convert.ToBoolean(PlayerPrefs.Get-
Int("FullscreenPreference"));
     else
       Screen.fullScreen = true;
   }
}
```

Приложение В. Скриншоты итоговой версии игры

На рисунках 1–5 представлены скриншоты итоговой версии игры.

Рисунок 1 – Главное меню

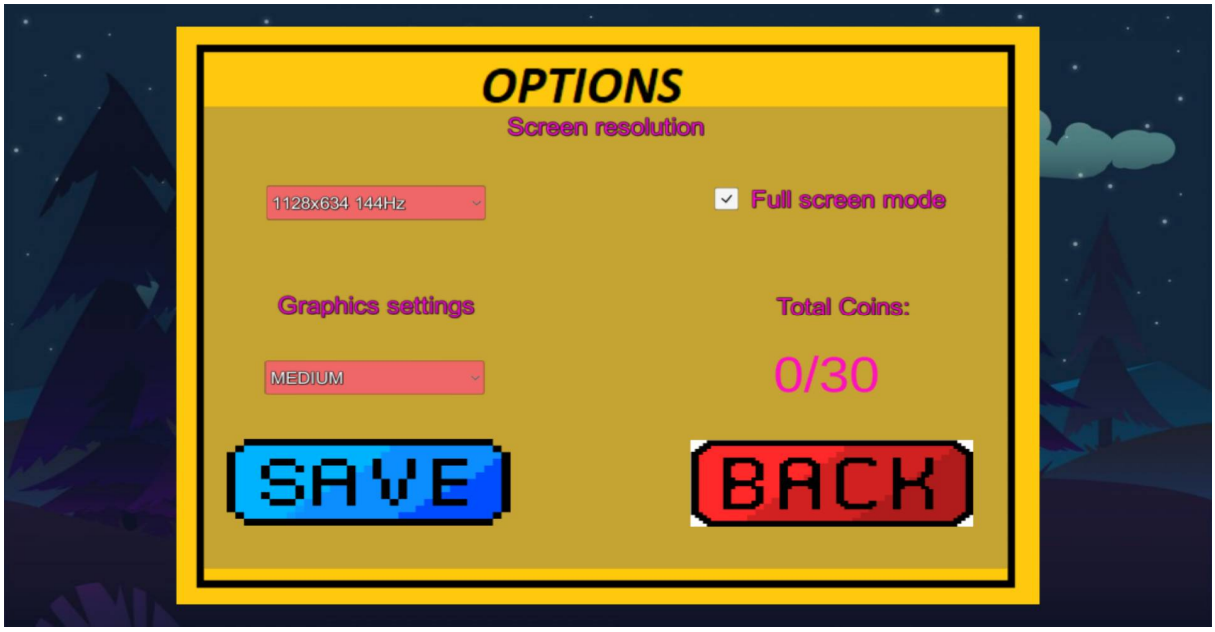

Рисунок 2 – Меню настроек

Рисунок 3 – Меню паузы

Рисунок 4 – Меню конца уровня

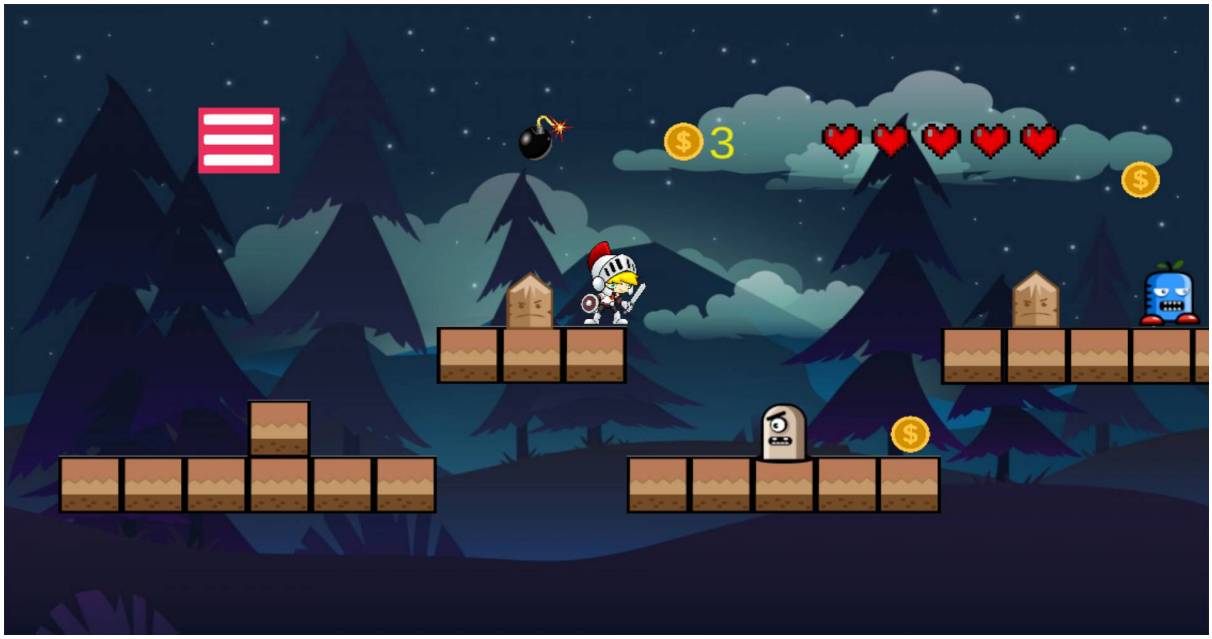

Рисунок 5 – Игровой процесс