«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	09.03.04 Программная инженерия
Профиль	Разработка программно-информационных систем
Факультет	КТИ
Кафедра	МО ЭВМ

К защите допустить

Зав. кафедрой Кринкин К.В.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: РАЗРАБОТКА ИГРЫ «COIN GAME» НА БАЗЕ UNITY 3D

Студент		_____ *подпись*	Щука А.А.
Руководитель	к.т.н., доцент *(Уч. степень, уч. звание)*	_____ *подпись*	Романцев В.В.
Консультанты	Старший преподаватель *(Уч. степень, уч. звание)*	_____ *подпись*	Герасимова Т.В.
	к.э.н., доцент *(Уч. степень, уч. звание)*	_____ *подпись*	Антонова А.М.
	к.т.н., доцент *(Уч. степень, уч. звание)*	_____ *подпись*	Заславский М. М.

Санкт-Петербург

2022

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ Кринкин К.В.
«___» _____2022 г.

Студент Щука А.А. Группа 8304
Тема работы: Разработка игры «Coin Game» на базе Unity 3D

Место выполнения ВКР: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)

Исходные данные (технические требования):
ОС Windows, ОС Linux, ОС Mac OS, Unity 3D

Содержание ВКР:
Обзор предметной области, Разработка персонажа, Разработка объекта «монета», Разработка игровой локации, Разработка GUI, Экономическое обоснование ВКР.

Перечень отчетных материалов: пояснительная записка, иллюстративный материал.

Дополнительные разделы: Экономическое обоснование ВКР.

Дата выдачи задания «20» ___апреля____2022 г.

Дата представления ВКР к защите «21» _ __июня_____2022 г.

Студент _____ Щука А.А.
подпись

Руководитель к.т.н., доцент _____ Романцев В.В.
(Уч. степень, уч. звание) *подпись*

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ Кринкин К.В.
«___» _____2022 г.

Студент Щука А.А. Группа 8304

Тема работы: Разработка игры «Coin Game» на базе Unity 3D

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	20.04 – 26.04
2	Обзор предметной области	26.04 – 28.04
3	Выбор технологий и сред разработки	28.04 – 29.04
4	Проектирование игры	29.04 – 03.05
5	Разработка игры	03.05 – 14.05
6	Экономическое обоснование ВКР	01.05 – 02.05
7	Оформление пояснительной записки	26.04 – 16.05
8	Оформление иллюстративного материала	16.05 – 30.05
9	Предзащита	02.06
10	Защита	21.06

Студент _____ Щука А.А.
подпись

Руководитель к.т.н., доцент _____ Романцев В.В.
(Уч. степень, уч. звание) *подпись*

РЕФЕРАТ

Пояснительная записка 59 стр., 22 рис., 16 табл., 21 ист.

3D, АРКАДНЫЕ ИГРЫ, UNITY, WINDOWS, LINUX, MAC OS.

Объектом исследования данной работы являются видеоигры в жанре «аркада».

Предметом исследования работы является разработка игр в жанре «аркада» от третьего лица для персональных компьютеров.

Цель работы – спроектировать и разработать компьютерную игру в жанре «аркада» от третьего лица с возможностью сбора предметов (монет) за определенное количество времени на игровом движке Unity 3D.

В рамках данной работы была разработана аркадная видеоигра для персональных компьютеров с операционной системой Linux, Windows или Mac, направленная на развитие реакции, мелкой моторики и пространственного мышления пользователя, с использованием современных технологий игрового движка Unity. В работе был выполнен обзор предметной области, в котором определены преимущества и недостатки различных игровых движков.

ABSTRACT

Explanatory note 59 p., 22 fig., 16 tables, 21 sources.

3D, ARCADE GAMES, UNITY, WINDOWS, LINUX, MAC.

The object of this research are video games in the arcade genre.

The subject of the research work is the development of third-person arcade games for personal computers.

The goal of the work is to design and develop a third-person arcade computer game with the ability to collect items (coins) in a certain amount of time using the Unity 3D game engine.

As part of this work, an arcade video game was developed for personal computers with the Linux, Windows or Mac OS operating system, aimed at developing the reaction, fine motor skills and spatial thinking of the user, using modern technologies of the Unity game engine. In the work, an overview of the subject area was made, in which the advantages and disadvantages of various game engines were identified.

СПИСОК ИСПОЛЬЗОВАННЫХ ОПРЕДЕЛЕНИЙ, ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

В настоящей пояснительной записке применяются следующие сокращения и термины с соответствующими определениями:

Rigitbody – компонент игрового движка Unity, реализующий симуляцию физики.

3D – трехмерный.

Коллайдер (Collider) - компонент игрового движка Unity, реализующий симуляцию столкновений объектов.

Legacy – устаревший.

IT - Information Technology (информационные технологии).

AAA-игры - класс высокобюджетных компьютерных игр.

Mesh (меш) – объект для моделирования геометрии.

Коллизия – столкновение.

Платформер — жанр компьютерных игр, в которых основу игрового процесса составляет преодоление препятствий.

GUI – Graphic User Interface (Графический пользовательский интерфейс).

ВКР – выпускная квалификационная работа.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире информационные технологии стали неотъемлемой частью жизни людей. Количество используемых компьютеров (в том числе смартфонов и ноутбуков) во всём мире достигло 6,2 млрд в 2021 году [1]. Заказать доставку продуктов, создать электронный документ, отправить письмо, воспользоваться поиском информации в сети Интернет - действия, которые для людей стали повседневными. Цифровые развлечения, в частности игры для персональных компьютеров и игровых приставок, очень популярны. Можно предположить, что игры интересны только детям, но, согласно исследованию Центра развития НИУ «ВШЭ» [2], более 70% игроков старше 18 лет. Это можно объяснить тем, что игры позволяют отвлечься от повседневных дел, отдохнуть или нескучно провести время. Для того чтобы удовлетворить растущие потребности пользователей, разработчики и геймдизайнеры постоянно создают новые игры различных жанров. Один из популярных жанров видеоигр – аркада, особенностью которого является динамичный и простой игровой процесс, в таких играх для успешного прохождения пользователю не требуются особые навыки и знания. Аркады характеризуются незамысловатым сюжетом, длительность сессии у них относительна короткая: несколько минут. Эти особенности позволяют пользователю не тратить много времени и сил: человек легко погружается в игровой процесс, быстро достигает результата и получает удовольствие от игры за короткий промежуток времени. Такой критерий очень важен, учитывая быстрый ритм современной жизни людей. Все вышеперечисленное делает аркадные игры очень востребованными, они охватывают широкую аудиторию.

На рынке приложений для персональных компьютеров и игровых приставок, как и на любом другом рынке, есть конкуренция. Чтобы разработанная игра была популярна среди целевой аудитории, создатели используют всевозможные технологические и визуальные способы

улучшения своего проекта. По мере развития технологий и создания всё более усовершенствованных устройств игры становятся реалистичнее. Облегчить процесс разработки помогают игровые движки.

Игровой движок - это базовое программное обеспечение практически всех компьютерных видеоигр. Понятия «игра» и «игровой движок» довольно близкие по смыслу, но основное различие заключается в том, что термин «игровой движок» используется для программного обеспечения, которое может быть неоднократно использовано и удобно для масштабирования. Таким образом, движок можно считать основой для создания большого количества различных программ без существенных изменений.

В данной работе будут рассмотрены различные игровые движки и создана аркадная 3D-игра с возможностью управления персонажем и сбором ключевых предметов за ограниченное время, она будет направленна на развитие реакции, мелкой моторики и пространственного мышления.

Сценарий игры будет следующим: пользователь управляет персонажем с видом от третьего лица, он может перемещаться по локации, например, бегать и прыгать. Его цель – собрать все монеты за определённый промежуток времени. При этом, на игровой локации будут различные препятствия: движущиеся объекты, вода, поваленные деревья, камни. При падении в воду или при окончании таймера игрок проигрывает. Если же пользователю удалось собрать все монеты, то игра награждает его фейерверками.

1.1. Постановка задачи

Необходимо разработать 3D-игру для ПК, обеспечивающую поддержку следующих возможностей:

1. Управление персонажем с видом от третьего лица.
2. Пользователю необходимо собрать ключевые предметы на игровой локации за ограниченное время.

3. Игровое окружение включает в себя препятствия и области, которые воздействуют на персонажа заданным способом, например, при падении в воду игрок проигрывает.
4. GUI содержит информацию о текущем прогрессы игры и состоянии игрока.

1.2. Цель, задачи, объект и предмет исследования

Цель работы: создать компьютерную игру в жанре «аркада» от третьего лица с возможностью сбора предметов (монет) за определенное количество времени на игровом движке Unity 3D.

Объект исследования: видеоигры в жанре «аркада».

Предмет исследования: разработка игр в жанре «аркада» от третьего лица с возможностью сбора предметов за определенное количество времени, что обуславливает необходимость контроля сразу нескольких показателей.

Для создания игры нужно выполнить следующие **задачи**:

1. Провести обзор предметной области.
2. Обозначить ключевые аспекты и технологии, необходимые для разработки приложения.
3. Создать и настроить игрового персонажа (модель, управление, анимации).
4. Разработать объекты монет (модели, анимации, взаимодействие с другими объектами).
5. Создать игровые локации и набор игровых сценариев.
6. Реализовать графический пользовательский интерфейс игры, включающий в себя неигровой (главное меню, меню помощи, меню паузы) и игровой интерфейсы.
7. Провести экономическое обоснование целесообразности разработки.

1.3. Новизна и практическая значимость работы

Игровая индустрия в эпоху цифровых технологий быстро растет и развивается, поэтому остается актуальным объектом для исследований. Жанр «аркада» является одним из самых популярных игровых жанров, так как такие игры просты в отношении геймплея и графики и не требуют особых навыков от игрока. К жанру «аркада» можно отнести следующие виды игр: приключения, платформеры, гонки, экшн-игры и другие.

Согласно обзору НИУ ВШЭ мирового рынка компьютерных игр за 2019 год, на игры для игровых приставок и ПК приходится 55% продаж, на мобильные игры 45% [2]. Рынок игр для ПК и игровых приставок, следуя тенденциям мобильных игр к простоте геймплея и графике, расширяется. Успехом у геймеров пользуются не только ААА-игры, но и казуальные аркадные игры.

Научная новизна работы состоит в том, что в результате работы будет разработана аркадная 3D-игра с возможностью сбора предметов за определенное количество времени с использованием современных технологий.

Практическая значимость работы состоит в том, что разработанная игра будет не только интересной, но и полезной широкому кругу пользователей, так как она направлена на развитие реакции, мелкой моторики и пространственного мышления игрока, а также, учитывая спрос на рынке игр для персональных компьютеров, она может быть коммерчески успешной.

2. АНАЛИТИЧЕСКИЙ ОБЗОР СОВРЕМЕННОГО СОСТОЯНИЯ ВОПРОСА

2.1. Анализ и общая характеристика предметной области

Компьютерная игра – это компьютерная программа, организующая игровой процесс, реализующая взаимодействие между игроками или самостоятельно выступающая в роли участника (искусственный интеллект). Аркада – жанр компьютерных игр, которые характеризуются коротким по времени, но насыщенным игровым процессом. При этом пользователь может не иметь специальные навыки для того, чтобы разобраться с управлением и правилами игры [19].

По мере роста производительности ЭВМ и их распространённости среди частных лиц появилась необходимость разработки приложений, которые удовлетворяют разнообразные потребности пользователей. Для аркад характерно быстрое обучение и простой процесс игры, а также незамысловатый сюжет. В них имеется подсчёт очков, который позволяет игроку понять, насколько он хорошо действовал. Данные качества, с коммерческой точки зрения, дают возможность аркадам охватить как можно больший спектр игроков.

2.2. Обзор существующих решений

В рамках обзора существующих решений были изучены некоторые игровые движки. Особенности каждого из рассмотренных решений приведены ниже.

2.2.1. CryEngine

Движок, который использовался для создания таких известных игр, как Crysis [3] и Far Cry [4], но большой популярности так и не набрал. Условия использования — 5% при доходе более 5 тысяч евро в год. CryEngine [5] хорошо подходит для разработки шутеров под ПК или игровую приставку, но возможность разработки для мобильных устройств отсутствует.

В момент релиза игру Crysis считали революционной благодаря уровню взаимодействия с окружающей средой. Предметы можно было разрушать, подбирать и перемещать, но на максимальных настройках графики производительность данной игры была на очень низком уровне. В настоящее время оптимизация движка плохая, по сравнению с аналогами.

Из положительных характеристик следует отметить высококачественный объёмный звук и реалистичную проработку перспективы. Например, туман имеет три разновидности: слоистый, объемный и дальний.

2.2.2. Godot

Движок Godot [6], главное достоинство которого - цена. Он полностью бесплатный. Размер не требующего установки исходного файла - менее 70 мегабайт. Несмотря на это, Godot имеет полноценные возможности, привычные современному разработчику.

Godot стремится предложить полностью интегрированную среду разработки игр. Он позволяет разработчикам создавать игры, не нуждаясь в других инструментах, кроме тех, которые используются для создания контента (визуальные ресурсы, музыка и т.д.). Архитектура движка построена на концепции дерева «узлов». Узлы организованы внутри «сцен», которые представляют собой повторно используемые экземпляры, наследуемые и вложенные группы узлов. Все игровые ресурсы, включая сценарии и графические ресурсы, сохраняются как часть файловой системы компьютера, а не в базе данных.

Godot содержит систему анимации с графическим интерфейсом для скелетной анимации, смешивания, анимационных деревьев, морфинга и кат-сцен в реальном времени. Практически любую переменную, определенную или созданную в игровом объекте, можно анимировать.

2.2.3. Unreal Engine

Unreal Engine [8] – это известный движок, с помощью которого были созданы многие известные игры, например, Fortnite [7]. Условия использования — бесплатно для некоммерческого использования или 5% дохода для коммерческого при продажах продукта на сумму более 1 млн. $.

Unreal Engine имеет много положительных качеств: широкий набор стандартных шаблонов (трехмерные шутеры, полеты в космосе, двухмерные пазлы, платформеры, гонки), многие функции, которые в Unity надо непременно описывать в сценариях, встроенных в движок заранее, в Marketplace присутствует много бесплатного качественного контента.

Логика программируется на C++, но простые вещи можно реализовать с помощью Blueprint [20].

Существенный недостаток - проекты получаются достаточно объемными, и для мобильных устройств приходится очень серьезно заниматься оптимизацией.

2.2.4. Unity

Unity [9] – один из самых популярных игровых движков. На нём были созданы очень многие известные игры как на настольные, так и на мобильные устройства. Условия использования – бесплатная лицензия, если доход не превышает 100 000 долларов в год, в противном случае необходимо приобрести подписку.

Значимые преимущества Unity следующие: низкий порог вхождения, простой, удобный, интуитивно понятный визуальный редактор, встроенный редактор ландшафта, базовая физика, в том числе симуляция ткани, кроссплатформенность, AssetStore [10], в котором есть большое множество ресурсов для создания приложения, модульная архитектура проекта.

К недостаткам можно отнести следующие пункты: для реализации любой идеи требуется запрограммировать сценарий на языке программирования C#, большой объем проектов, отсутствие гибкости в настройке фундаментальных параметров движка [11].

2.3. Сравнение игровых движков

Результаты сравнения игровых движков представлены в таблице 2.1.

Таблица 2.1 – Сравнение игровых движков

Название	**Стоимость лицензии при коммерческом использовании**	**Поддерживаемые платформы**	**Дополнительные возможности**
CryEngine	5% при доходе более 5 тысяч евро в год	Microsoft Windows, Linux	1. фотореалистичная графика 2. GameSDK [21] 3. Трассировка лучей без использования RTX
Godot	Бесплатно	macOS, Microsoft Windows, Linux, iOS, Android	1. Независимое 2D и 3D. 2. Открытый исходный код 3. Поддержка сторонних языков.
Unreal Engine	5% при доходе более 1000000 $ от продажи продукта	macOS, Microsoft Windows, Linux, iOS, Android	1. Визуальное создание сценариев [20] 2. Физика твердых и жидких тел
Unity	1 800 $ при доходе более 100000 $ в год	macOS, Microsoft Windows, Linux, iOS, Android	1. AssetStore [10] 2. Удобный редактор 3. Физика твердых тел 4. Система анимации

2.4. Выводы

Был проведен анализ предметной области и рассмотрены варианты существующих решений. Для создания кроссплатформенной, масштабируемой и оптимизированной игры лучше всего подходят Unreal Engine и Unity.

Было установлено, что Unity, в отличие от Unreal Engine, более простой в освоении, обладает собственным магазином готовых решений,

которые зачастую бесплатны или стоят недорого, что с экономической точки зрения выгодно для небольшого проекта. Также он более оптимизирован для разработки приложений для мобильных платформ. Таким образом, для разработки приложения был выбран Unity.

3. РАЗРАБОТКА ИГРОВОГО ПЕРСОНАЖА

Разработка игрового приложения и его особенностей начинается с создания и настройки игрового персонажа – модели, системы управления его передвижениями, соответствующей анимацией, а также взаимодействия с окружающими предметами игрового мира, например, монетами, водой или препятствиями.

3.1. Модель персонажа

3.1.1. Формулировка требований к модели

Персонаж, которым будет управлять пользователь, представляет собой 3D-модель, которая должна обладать анимацией, симуляцией физики, а также корректно взаимодействовать с другими объектами. Таким образом, компоненты модели, которые требуется разработать:

1. Костная структура.
2. Skinned Mesh Renderer.
3. Rigitbody.
4. Коллайдер для обработки столкновений.
5. Контроллер анимаций.

Unity использует компонент `Skinned Mesh Renderer` для рендеринга анимации костей, где форма сетки деформируется анимированными костями. Эта технология полезна для персонажей и других объектов, чьи суставы изгибаются (в отличие от машины, где суставы больше похожи на шарниры).

Кости — это невидимые объекты внутри модели, которые влияют на то, как меш деформируется во время анимации. Кости соединяются вместе, чтобы сформировать иерархический «скелет», вращение суставов скелета определяет анимацию. Каждая кость прикреплена к некоторым вершинам окружающего меша. Когда воспроизводится анимация, вершины перемещаются вместе с костью или костями, с которыми они связаны, поэтому «кожа» следует за движением скелета.

Для реализации взаимодействия с другими объектами модели требуются `Collider` и `Rigitbody`. Коллайдер должен соответствовать геометрии меша, не допускать «прохождения» модели в другие объекты.

3.1.2. Обзор реализованной модели

В 3D-редакторе была создана модель. Она представляет собой скелет, обтянутый мешем. Визуально, она напоминает человека, одетого в форму ниндзя. 3D-модель представлена на рисунке 1.

Рисунок 1 - 3D-модель персонажа

Для обработки коллизий на модель было добавлено несколько цилиндрических коллайдеров с полусферами на каждом конце, которые представлены на рисунке 2 (отмечены зелеными линиями). Их размер немного превосходит размеры видимой геометрии персонажа для того, чтобы симуляция физических столкновений выглядела естественно.

Рисунок 2 - Коллайдеры модели персонажа

3.2. Архитектура сценариев управления персонажем

3.2.1. Формулировка требований к игровому персонажу

Были сформулированы требования к возможностям и характеру управления персонажем:

- Герой может перемещаться по локации. Команды для перемещения задаются с помощью клавиатуры или геймпада.
- Имеется возможность увеличивать скорость перемещения на определенный промежуток, обусловленный параметром *выносливость*, который восстанавливается со временем.
- Для ориентации в пространстве и обзора локации используется камера с видом от третьего лица, изменение ракурса обзора происходит с помощью движения компьютерной мыши.
- При столкновении с монетами, у персонажа увеличивается счетчик собранных монет, после чего монета пропадает.
- При взаимодействии с водой персонаж становится недоступным к дальнейшим командам пользователя.

- При движении по платформам или столкновении с декорациями, игрок не может проходить через объекты. Если объекты перемещаются, то они влияют на положение того объекта, с которым они взаимодействуют.
- Пользователь отслеживает свой прогресс по ходу игрового процесса, а именно: количество собранных монет, необходимое количество монет, оставшееся время, текущий уровень выносливости.

3.2.2. Проектирование системы управления персонажем

При создании иерархии классов был использован шаблон проектирования «мост». Он разделяет абстракцию и реализацию так, чтобы они могли изменяться независимо. Схема шаблона проектирования представлена на рисунке 3.

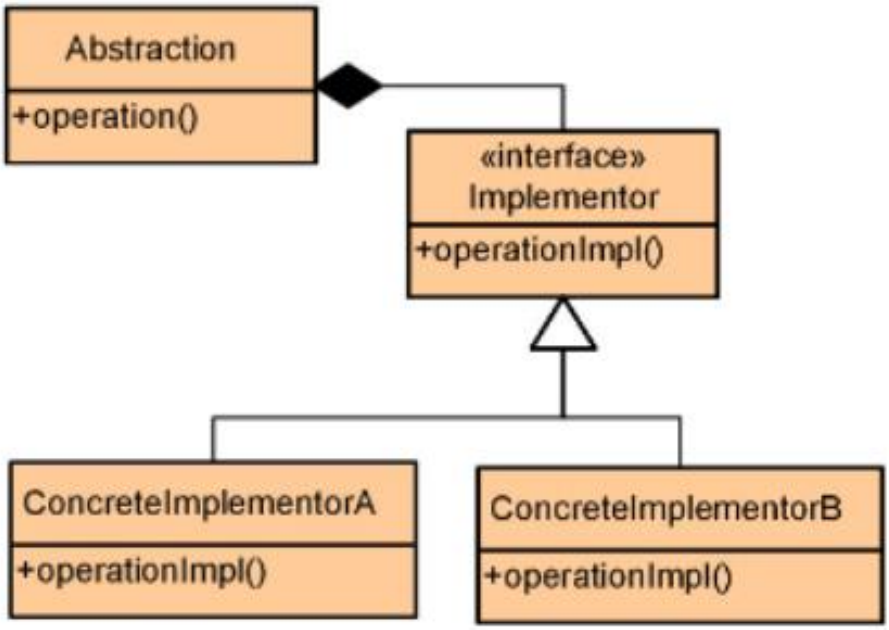

Рисунок 3 - Схема паттерна "мост"

UML-диаграмма, описывающая связи между всеми управляющими сценариями игрового персонажа представлена на рисунке 4.

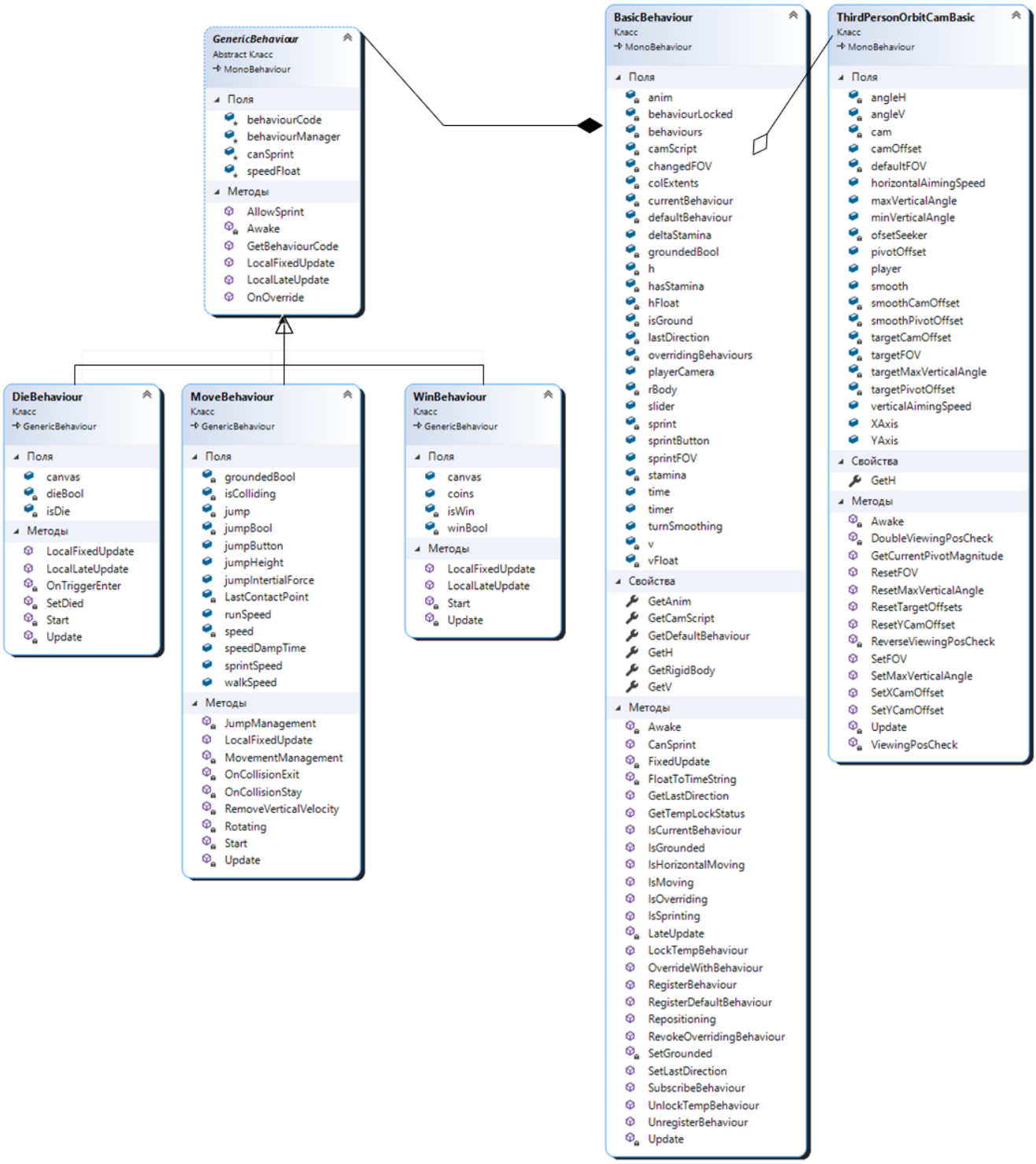

Рисунок 4 - Диаграмма классов управления персонажем

Класс `BasicBehaviour` реализует базовое поведение игрока. Он контролирует корректное переопределение поведения, связывает персонажа с камерой и контроллером анимации, а также отслеживает различные параметры, такие как: положение, выносливость, время до окончания игры, состояние бега.

`GenericBehaviour` – это абстрактный класс, реализующий общий интерфейс переопределяющего поведения.

`MoveBehaviour` – это класс, реализующий передвижение. Он содержит следующие параметры: скорость при ходьбе, беге и быстром беге, высоту и силу прыжка, коллайдер. `MoveBehaviour` обрабатывает столкновения, нажатия клавиш для перемещения и вращения, а также просчитывает логику этих действий.

`WinBehaviour`, `DieBehaviour` – это классы, которые реализуют логику поведений при победе или поражении.

Для позиционирования камеры используется класс `ThirdPersonOrbitCamBasic`. Различные параметры и методы, которые содержит этот класс, позволяют камере следовать за персонажем и менять свое положение, если пользователь перемещает компьютерную мышь или вращает колесо мыши.

3.3. Управление персонажем

3.3.1. Обработка событий нажатий клавиш

В игровом движке Unity 3D существует возможность обработки событий нажатия клавиш напрямую с указанием кода каждой конкретной клавиши.

Также в игровом движке Unity 3D есть реализация системы игрового ввода (`Input`), которая позволит игроку по мере надобности (в ходе игрового процесса) переназначать клавиши управления и обеспечит кроссплатформенность [11].

В таблице 3.1 представлено описание системы игрового ввода. Все методы содержатся в классе `Input`.

Таблица 3.1 – Система игрового ввода

№	Метод	Описание	Клавиша на ПК	Клавиша на геймпаде
1	GetAxis(“Horizontal”)	Движение влево/вправо	A/D/←/→	Joystick
2	GetAxis(“Vertical”)	Движение вперед/назад	W/S/↑/↓	Joystick
3	GetButtonDown(“Jump”)	Прыжок	Space	Button 0
4	GetButton(“Sprint”)	Быстрый бег	L. shift	Button 8
5	GetButton(“Cancel”)	Меню паузы/отмена действия	Esc	Button 1
6	GetAxis(“Mouse X”)	Движение камеры влево/вправо	Mouse	
7	GetAxis(“Mouse Y”)	Движение камеры вверх/вниз	Mouse	
8	GetAxis(“Analog X”)	Движение камеры влево/вправо		Joystick
9	GetAxis(“Analog Y”)	Движение камеры вверх/вниз		Joystick
10	GetAxis(“ScrollWheel“)	Отдаление/приближение камеры	Mouse wheel	Button 5/6

3.3.2. Обзор системы управления персонажем

Была создана система управления персонажем с помощью методов класса `Input`. Игрок с помощью устройств ввода может управлять передвижением и прыжками героя, направлением и положением камеры, а также приостанавливать игру.

Персонаж в игре ведет себя как физический объект, на него действуют силы гравитации и трения. При столкновении с игровыми предметами происходит обработка коллизий, в зависимости от типа предмета. Например, при движении по земле или столкновении со стеной модель не «проваливается» внутрь другого объекта, при столкновении с монетой – она исчезает, при падении в воду герой «умирает», после чего продолжает плавать.

Компоненты управления были добавлены к 3D-модели, результат представлен на рисунке 5.

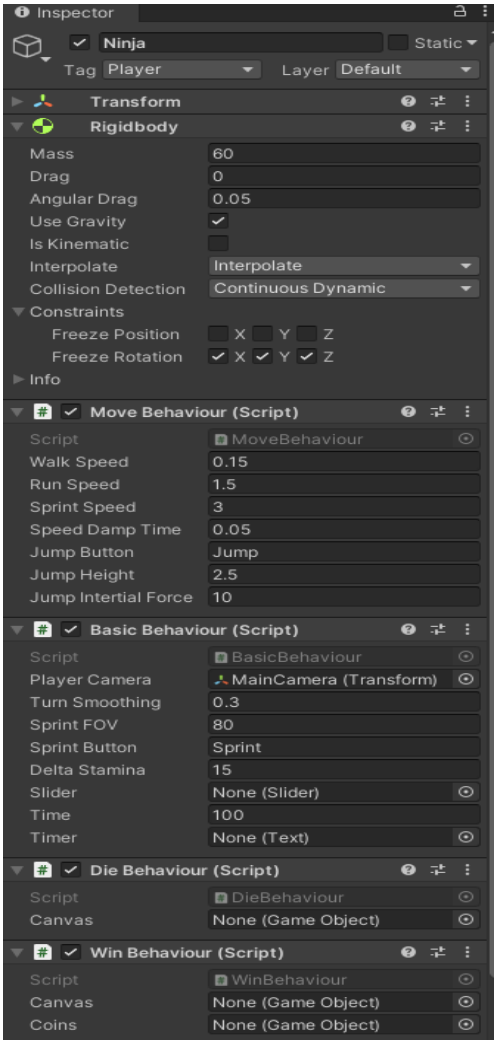

Рисунок 5 - Компоненты управления

3.4. Система анимации персонажа

В соответствии с требованиями к персонажу (см. раздел 3.2.1), требуется реализовать следующие анимации:

- персонаж стоит на месте, слегка покачиваясь (Idle);
- бег (Run);
- быстрый бег (FastRun);
- прыжок (Jump);
- падение (Falling);
- смерть персонажа (Die);
- несколько видов танцев.

В Unity для связи анимаций между собой используется компонент `AnimatorController`, который позволяет задать условия перехода от одной анимации к другой и время этого перехода [11].

Для того, чтобы с помощью сценариев управлять анимацией в компоненте `AnimatorController`, в классе `BasicBehaviour` используется поле `anim` (см. раздел 3.2.2). Листинг участков кода, связанных с анимацией представлен ниже:

```
public class BasicBehaviour : MonoBehaviour
{
    void Awake()
    {
        // ...
         // Установка ссылок
        anim = GetComponent<Animator>();
        hFloat = Animator.StringToHash("H");
        vFloat = Animator.StringToHash("V");

        groundedBool = Animator.StringToHash("Grounded");
      // ...
    }

    void Update()
    {
        // Сохранение осей ввода.
        h = Input.GetAxis("Horizontal");
        v = Input.GetAxis("Vertical");

        // Установка осей ввода на контроллер анимации.
        anim.SetFloat(hFloat, h, 0.1f, Time.deltaTime);
        anim.SetFloat(vFloat, v, 0.1f, Time.deltaTime);

      // ...

        // Установка на контроллер анимации проверки на столкновение
с землёй.
        anim.SetBool(groundedBool, isGround);

      // ...

    }
    // ...
}
```

3.4.1. Анимации движения и поведения персонажа

Были созданы следующие анимации: Idle, Locomotion, Jump, Die, Win, Falling. Locomotion (структура представлена на рисунке 6) содержит в себе ходьбу, бег, быстрый бег. Выбор анимации зависит от текущей скорости персонажа, на рисунке это поле Threshold.

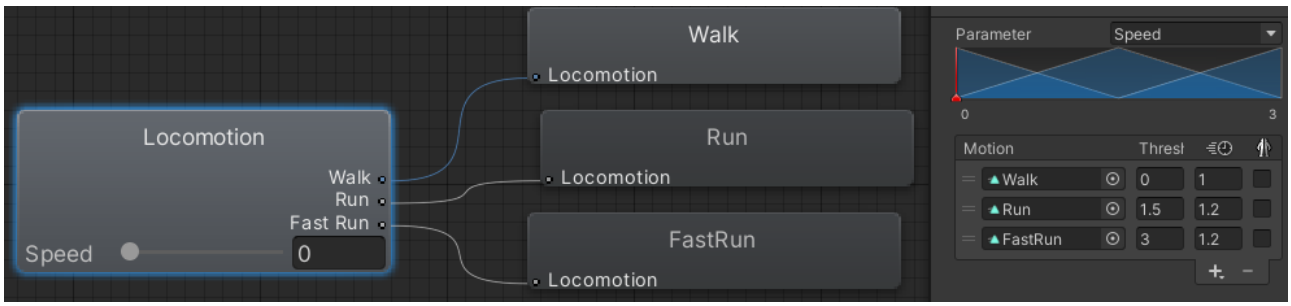

Рисунок 6 - Структура Locomotion

В `AnimatorController` были добавлены созданные анимации и переходы между ними. В таблице 3.2 представлено описание переходов, в столбце «условие» указаны параметры, которые содержатся внутри компонента. Стартовое состояние – Idle.

Таблица 3.2 – Переходы в компоненте `AnimatorController`

№	Начальное состояние	Конечное состояние	Условие перехода
1	Idle	Locomotion	$Speed > 0.1$ $Grounded = true$
2	Locomotion	Idle	$Speed < 0.1$
3	Idle	Jump	$Jump = true$
4	Locomotion	Jump	$Jump = true$
5	Jump	Falling	Переход происходит после завершения анимации
6	Idle	Falling	$Grounded = false$
7	Falling	Idle	$Speed < 0.1$ $Grounded = true$
8	Locomotion	Falling	$Grounded = false$
9	Falling	Locomotion	$Speed > 0.1$ $Grounded = true$
10	Any State	Die	$Die = true$
11	Any State	Win	$Win = true$

3.4.2. Анимации поведения персонажа в главном меню

Для того, чтобы игровая сцена в главном меню не была однообразной и скучной, были придуманы анимации танцев персонажа. Созданные анимации: Dance1, Dance2.

В MenuAnimatorController были добавлены созданные анимации и переходы между ними. В таблице 3.3 представлено описание переходов между анимациями. Стартовое состояние – Dance1.

Таблица 3.3 – Переходы в компоненте MenuAnimatorController

№	Начальное состояние	Конечное состояние	Условие перехода
1	Dance1	Dance2	Переход происходит после завершения анимации во второй раз
2	Dance2	Dance1	Переход происходит после завершения анимации во второй раз

3.4.3. Архитектура системы анимаций персонажа

Была разработана система анимаций, которая соответствует всем требованиям, изложенным в разделе 3.4. Анимации были связаны между собой посредством переходов, созданных в компонентах AnimatorController, и добавлены к системе управления персонажем (см. раздел 3.3).

Схемы AnimatorController и MenuAnimatorController отображены на рисунках 7 и 8 соответственно.

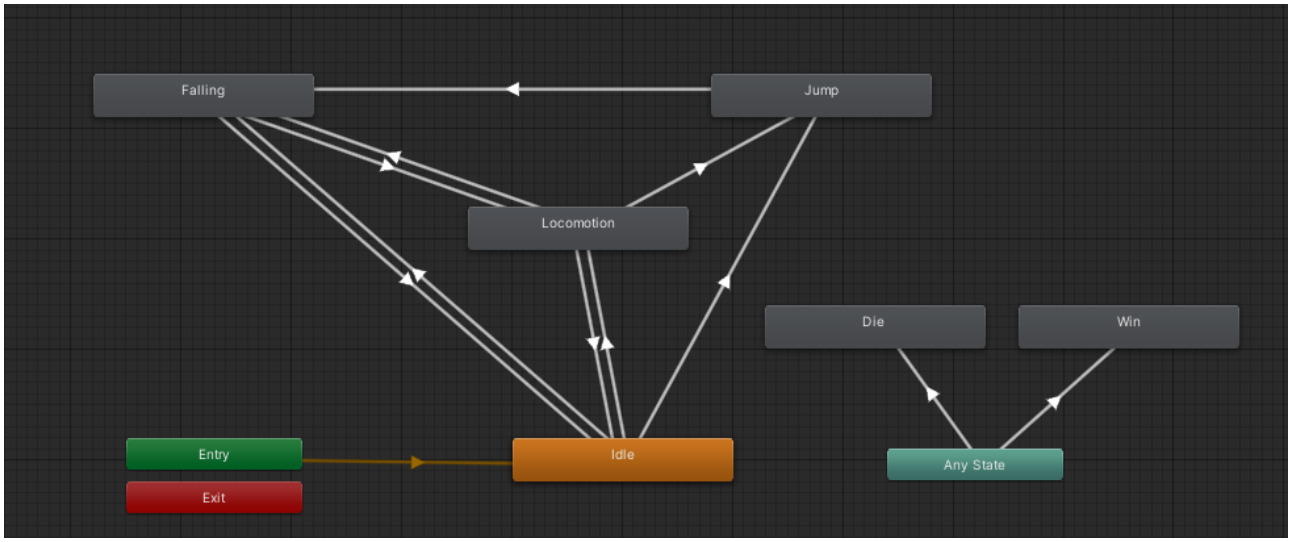

Рисунок 7 - Схема анимаций и их связей во время игры

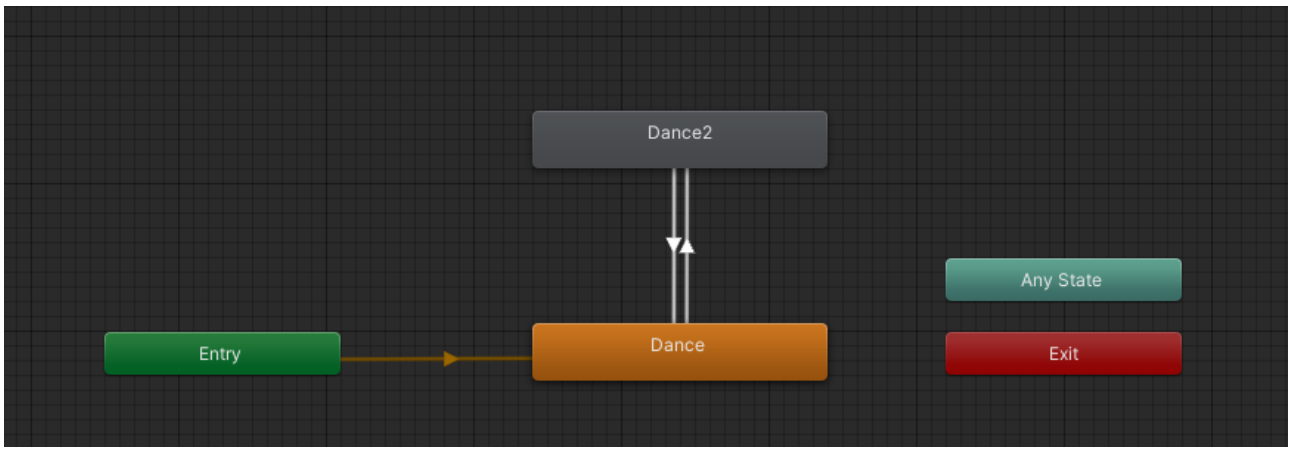

Рисунок 8 - Схема анимаций и их связей в главном меню

3.5. Выводы

В данном разделе представлена разработка игрового персонажа и его компонентов. Модель персонажа создана на основе скелетного каркаса, к ней добавлены: коллайдер, симуляция физики, сценарии управления и система анимаций.

Сценарии управления движением персонажа спроектированы по шаблону «мост», что даёт возможность масштабировать его поведение, добавляя новые сценарии без изменения других. Игровой ввод реализован с помощью системы `Input`, что обеспечивает кроссплатформенность.

Анимации персонажа объединены в компоненте `AnimatorController`, который предоставляет удобный интерфейс контроля анимаций.

4. РАЗРАБОТКА ОБЪЕКТА «МОНЕТА»

Игра в жанре «аркада» требует наличия условия для прохождения уровней. Одно из таких условий – сбор всех монет. Для разнообразия игрового процесса различные монеты должны влиять на игрока по-разному, например, замедлять или ускорять передвижение на определенный промежуток времени, добавлять время, оставшееся до окончания игры.

4.1. Формулировка требований к объекту «монета»

Монета – это 3D-объект игрового окружения. Ниже изложен перечень требований к внешнему виду и характеру взаимодействия с объектом:

- Геометрия предмета должна соответствовать общепринятым представлениям о монетах.
- Материал монеты обладает отражающими и рассеивающими свойствами, так как монеты состоят из металлов.
- Размер монет должен быть таким, чтобы пользователь мог замечать монеты с большого расстояния, но при этом их размер не должен превышать размеры игрового персонажа.
- Необходимо несколько типов монет для разнообразия игрового процесса: золотая, серебряная, медная.
- У объекта присутствует анимация.

Медная монета, являясь наименее ценной, при взаимодействии с персонажем, уменьшает его скорость бега. При этом скорость не может стать меньше определенного значения, подобранного экспериментально.

Серебряная монета, при взаимодействии с персонажем, увеличивает его высоту прыжка. В начале игры высота прыжка будет иметь небольшое значение, недостаточное для достижения некоторых мест на локации.

Золотая монета, при взаимодействии с персонажем, добавляет дополнительное время, оставшееся до окончания игры.

4.2. Система анимации объекта

Для создания анимации в Unity, кроме `AnimatorController`, используется компонент `Animation`. `Animation` – это legacy компонент, который использовался для целей анимации до введения текущей системы анимации Unity. Его недостаток в том, что он не содержит переходы между анимациями, поэтому позволяет воспроизводить только одну [11].

Единственная анимация монеты – это вращение вокруг своей вертикальной оси. Для ее реализации хорошо подходит `Animation`, но в целях дальнейшего масштабирования проекта был использован `AnimatorController`. Его схема представлена на рисунке 9. Стартовое состояние – Rotating.

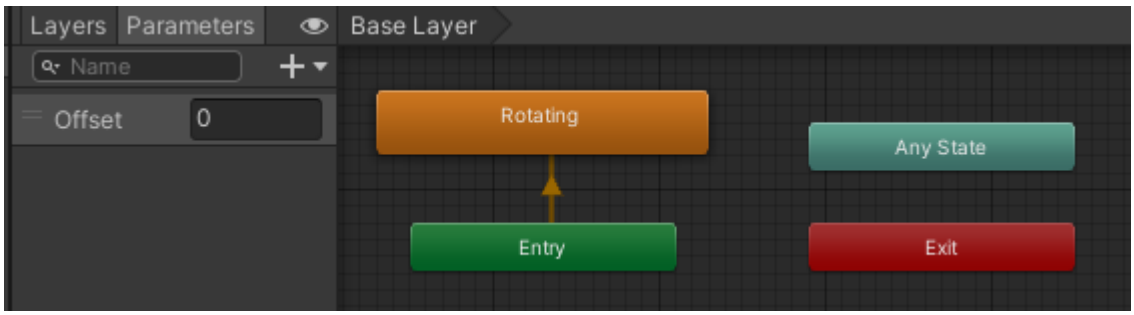

Рисунок 9 - Схема CoinAnimatorController

Для того, чтобы вращение у различных предметов начиналось с разным исходным углом, в `AnimatorController` переменная `offset` была привязана к `Cycle Offset Animation`, а также был создан сценарий, представленный ниже:

```
void Start()
{
    int offsetFloat = Animator.StringToHash("Offset");
    Animator animatior = GetComponent<Animator>();

    animatior.SetFloat(offsetFloat, Random.Range(0f, 0.5f));
}
```

4.3. Реализация взаимодействия с объектом

Для обработки коллизий к объекту был добавлен компонент `Box Collider` и создан сценарий, представленный ниже:

```
public class PickUpCoins : MonoBehaviour
{
    public string coinType;
```

```
    void OnTriggerEnter(Collider collider)
    {
        if (collider.gameObject.tag == "Player")
        {
            ChangeCoinTextUI.nCoins++;

            if (coinType == "Gold")
            {

collider.gameObject.GetComponent<BasicBehaviour>().time += 5f;
            }
            else if (coinType == "Silver")
            {

collider.gameObject.GetComponent<MoveBehaviour>().jumpHeight       +=
0.08f;
            }
            else if (coinType == "Copper")
            {
                if
(collider.gameObject.GetComponent<MoveBehaviour>().runSpeed > 1f)
                {

collider.gameObject.GetComponent<MoveBehaviour>().runSpeed -= 0.03f;
                }
            }

            Destroy(gameObject);
        }
    }
}
```

При столкновении с объектом происходит проверка того, является ли другой предмет игроком, с помощью тега. Далее, в зависимости от типа монеты, у персонажа изменяются некоторые атрибуты.

В итоге, вызывается метод `Destroy()`. Он удаляет `GameObject` сразу после текущего цикла обновления или через заданный промежуток времени. Фактическое уничтожение объекта всегда откладывается до завершения текущего цикла обновления, но всегда выполняется до рендеринга. Таким образом, происходит корректное удаление монет со сцены [12].

4.4. Обзор реализованных типов объекта

Медная монета уменьшает скорость бега игрока, но не ниже единицы. С логической точки зрения это означает, что пользователь должен вначале собирать монеты других номиналов, чтобы успеть закончить игру до срабатывания таймера. Внешний вид представлен на рисунке 10.

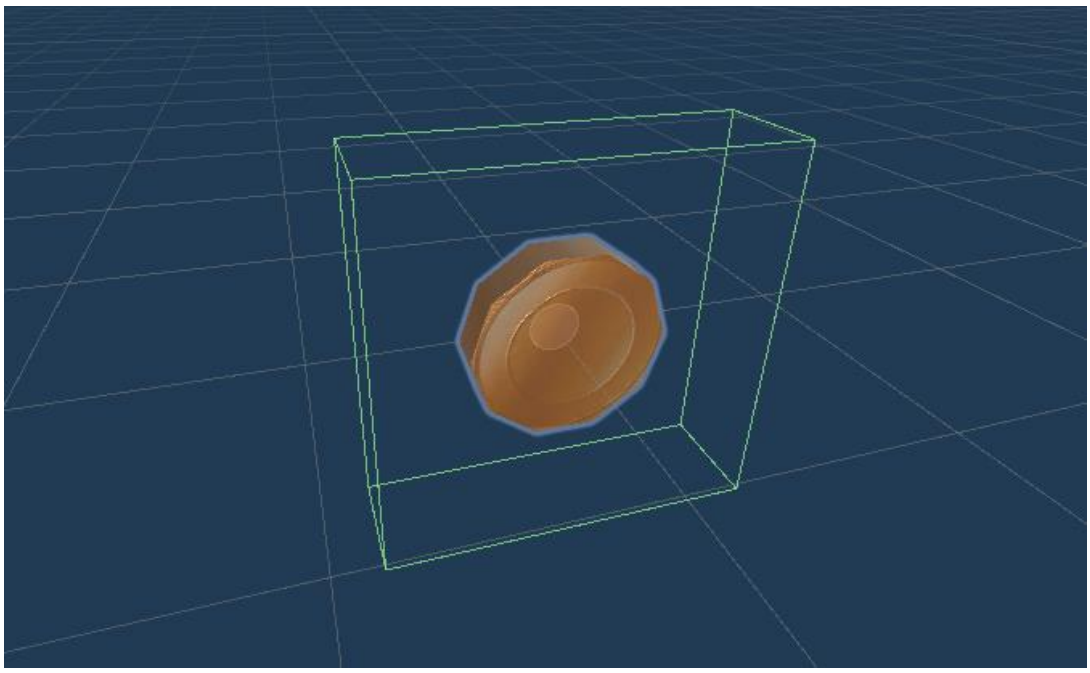

Рисунок 10 - Медная монета

Серебряная монета увеличивает высоту прыжка. При старте игры персонаж прыгает недостаточно высоко, чтобы попасть в определенные места на локации (см. раздел 5), поэтому, для успешного прохождения игры, пользователю нужно строить свой маршрут заранее, что вносит разнообразия в игру. Внешний вид представлен на рисунке 11.

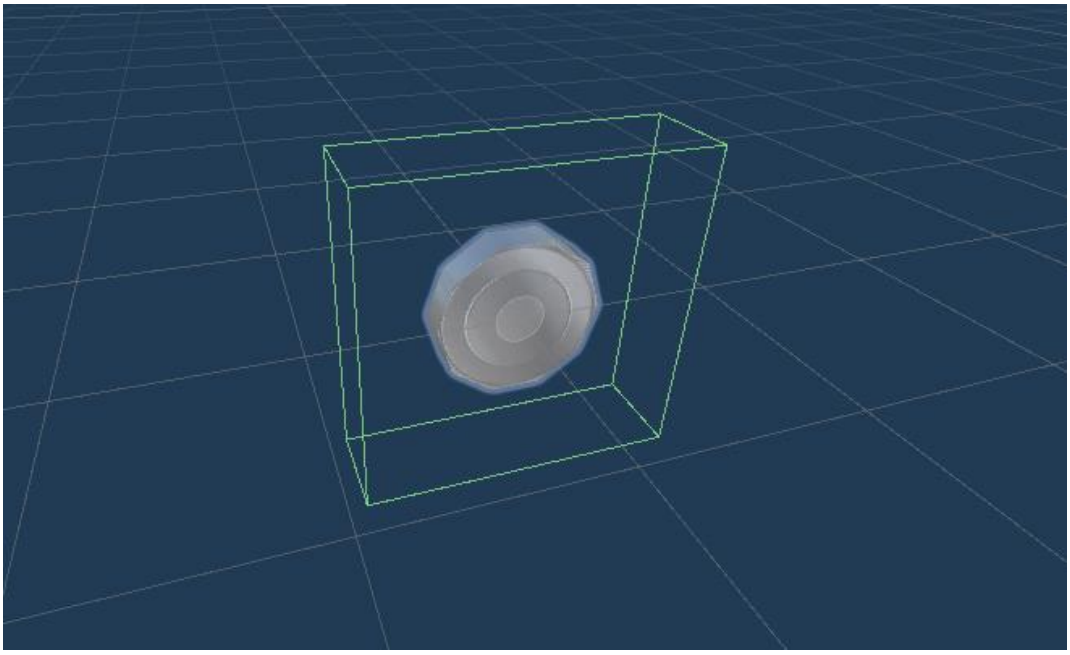

Рисунок 11 - Серебряная монета

Золотая монета увеличивает оставшееся время таймера, добавляя 5 секунд. Таким образом, собирая золотые монеты, игрок получает преимущество. Внешний вид представлен на рисунке 12.

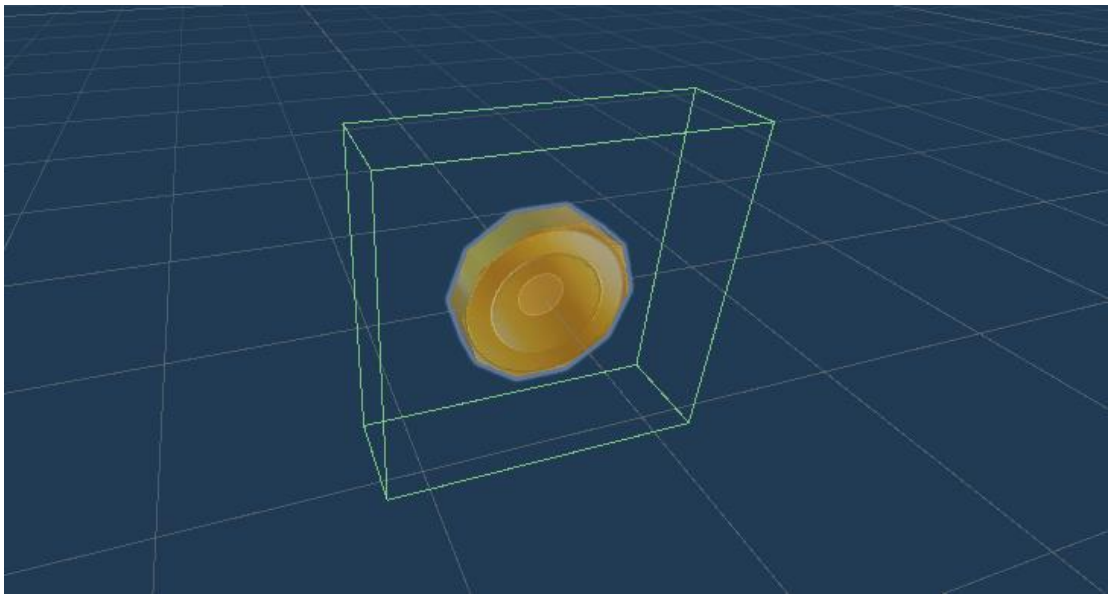

Рисунок 12 - Золотая монета

4.5. Выводы

В данном разделе представлена разработка объекта «монета» в соответствии со всеми заданными требованиями. Взаимодействие с ним реализовано при помощи коллайдеров и сценариев, которые обрабатывают коллизии. Также, была создана система анимации, при помощи которой монета вращается. К каждому типу монет добавлены различные бонусы при поднятии, что позволяет разнообразить игровой процесс.

5. ОПИСАНИЕ ИГРОВОГО ПРОЦЕССА. РАЗРАБОТКА ИГРОВОЙ ЛОКАЦИИ

5.1. Игровые сценарии и их реализация

5.1.1. Обзор игровых сценариев

Игровые сценарии — это долгосрочные тесты, состоящие из нескольких задач. Обычно они прямо сообщаются игроку, будь то в сюжете или в виде формальных условий победы. В некоторых видеоиграх есть только один сценарий, однако часто имеется несколько пересекающихся сценариев

God of War [13] помещает игрока в несколько сеттингов, в каждом из которых есть сценарий, который нужно преодолеть, прежде чем внутриигровой видеоролик перенесет историю в новое место. Выигрыш в теннисном матче или выполнение серии связанных голов в CityVille [14] — другие примеры сценариев.

В играх используется множество названий для описания сценариев, таких как миссия, квест, эпизод, сцена, мир или глава.

Популярные игровые сценарии, используемые в аркадах [18]:

- Игроку необходимо попасть из пункта А в пункт В, преодолевая препятствия на своём пути;
- Сбор ключевых предметов;
- Уничтожение всех противников;
- Поражение, при столкновении с противником или специальными объектами, например шипами или огнём;
- Накопление игровых очков, которые начисляются за определенные действия;
- Развитие навыков игрового персонажа.

Для реализации в разрабатываемой игре были отобраны следующие игровые сценарии, Use-Case UML-диаграмма представлена на рисунке 13:

1. Для победы игроку необходимо собрать все монеты на локации;
2. Время игры ограничено;

3. При падении в воду засчитывается поражение.

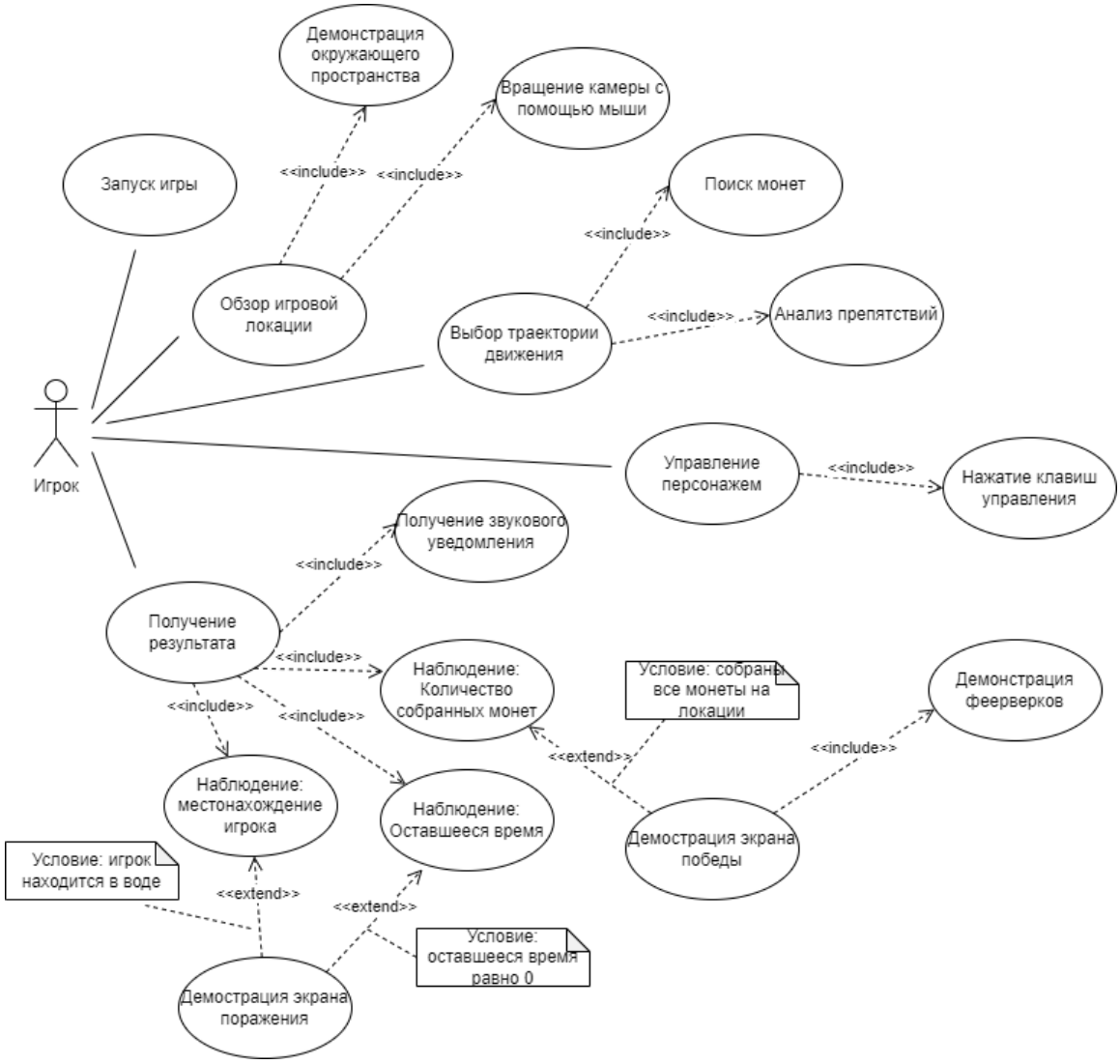

Рисунок 13 - Use-Case UML-диаграмма игрового раунда

5.1.2. Отслеживание прогресса по игровым сценариям

Игровой сценарий: положительный исход игры (победа) при сборе всех монет. Для реализации данного сценария был создан класс `WinBehaviour` (см. раздел 3.2.2). Логика программы:

1. Инициализация переменных;
2. В функции `Update()` проверка количества оставшихся монет на локации;
3. Если монет не осталось, то переход к пункту 4, иначе переход к пункту 2;
4. Начало анимации победы.

Листинг участков кода:

```
public class WinBehaviour : GenericBehaviour
{
    // ...
    void Start()
    {
        winBool = Animator.StringToHash("Win");
        behaviourManager.SubscribeBehaviour(this);
    }

    void Update()
    {
        if (isWin)
        {
            behaviourManager.GetAnim.SetBool(winBool, isWin);
            GetComponent<DieBehaviour>().enabled = false;
            canvas.GetComponent<Menu>().Victory();
        }
        else if (coins.transform.childCount == 0)
        {
            isWin = true;
        }
    }
}
```

Игровой сценарий: отрицательный исход игры (поражение) при падении в воду или по окончанию таймера. Для реализации данного сценария был создан класс `DieBehaviour` (см. раздел 3.2.2). Логика программы:

1. Инициализация переменных;
2. В функции `Update()` происходит проверка оставшегося времени и состояния игрока;
3. Если оставшееся время равно 0 или произошло столкновение с водой, то устанавливается флаг поражения, иначе переход к пункту 2.

Листинг участков кода:

```
public class DieBehaviour : GenericBehaviour
{
   // ...
    void Start()
    {
        dieBool = Animator.StringToHash("Die");
        canSprint = false;
    }

    void Update()
    {
        if (isDie)
        {
```

```
            behaviourManager.GetAnim.SetBool(dieBool, isDie);
            canvas.GetComponent<Menu>().Defeat();
        }
        else if (GetComponent<BasicBehaviour>().time == 0)
        {
            SetDied();
        }
    }

    private void OnTriggerEnter(Collider collider)
    {
        if (collider.tag == "Water")
        {
            SetDied();
        }
    }

    private void SetDied()
    {
        isDie = true;
        GetComponent<MoveBehaviour>().enabled = false;
    }
}
```

5.2. Специфика создания игровой локации

5.2.1. Требования к игровой локации

Сеттинг — это среда, в которой происходит действие в игре. Существуют всевозможные сеттинги. К ним относятся арт-декор, пост апокалипсис, научная фантастика, фэнтези, поле битвы и т.д. Из сеттинга можно получить много информации об игре.

Для игр жанра «аркада» наиболее часто используются локации, оформленные в фэнтези стиле. Фэнтези основывается на использовании мифологических и сказочных мотивов в современном виде. В этом жанре преобладают декорации, имитирующие Землю, но с отличиями от реальности [18].

Разрабатываемый проект – это аркада, в которой игроку необходимо преодолевать препятствия для того, чтобы собрать все монеты за ограниченное время, следовательно, окружение должно соответствовать следующим критериям:

- Сеттинг – фэнтези;
- Присутствуют препятствия, например движущиеся блоки, поваленные деревья.

- Ландшафт и декорации оформлены в ярких цветах;
- Размер локации небольшой, наполнение плотное;
- Существуют несколько путей прохождения, то есть локация нелинейная.

5.2.2. Создание ландшафта и определение ключевых объектов

Игровой движок Unity 3D предоставляет различные инструменты для создания ландшафта. Один из таких инструментов Unity Terrain [11].

Unity Terrain — это встроенный редактор местности, который позволяют добавлять пейзажи в игру. Редактор предоставляет возможность создать несколько слоёв, отрегулировать высоту или внешний вид ландшафта, а также добавить к нему деревья или траву. Во время выполнения Unity оптимизирует встроенный рендеринг Terrain для повышения эффективности.

Несмотря на все преимущества, с помощью данного редактора не получится создать ландшафт, состоящий из отдельных блоков. Для решения этой проблемы можно воспользоваться различными 3D-объектами в форме куба или цилиндра [19].

Для создания локации, удовлетворяющей требованиям, необходимы следующие элементы: наземные блоки, стены, мост, вода, движущиеся блоки. Объекты интерактивны, то есть влияют на поведение персонажа.

Разработанная локация представлена на рисунке 14.

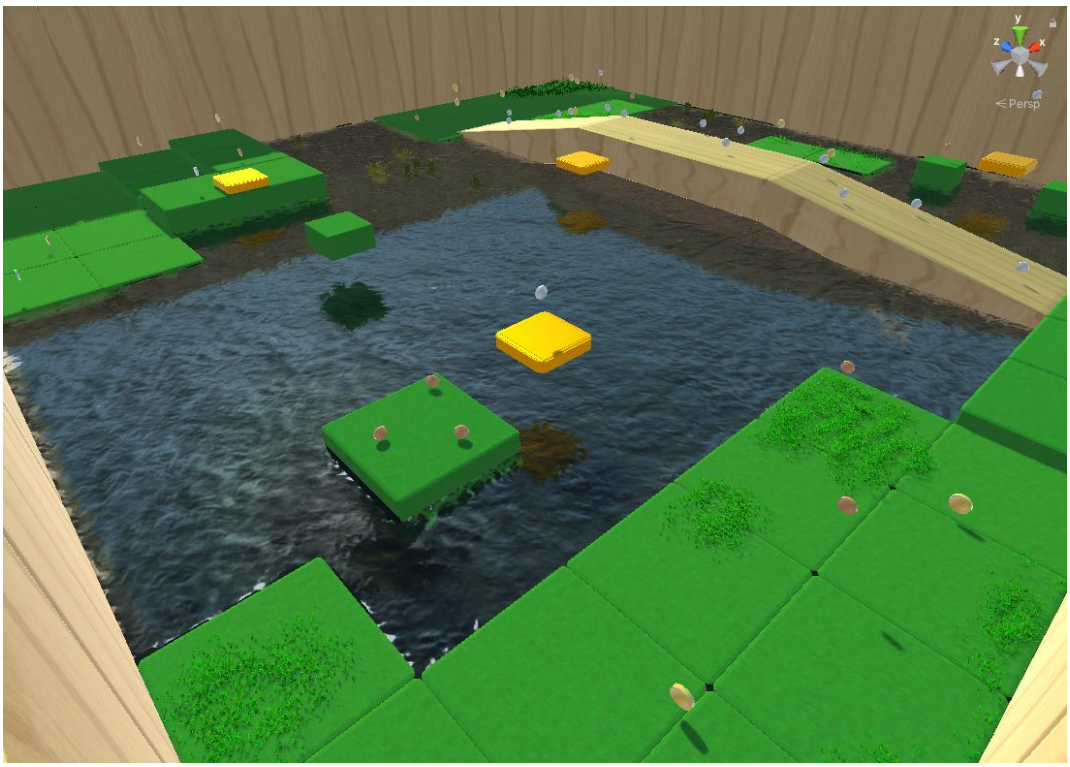

Рисунок 14 - Игровая локация, вид в перспективе

Для того, чтобы наполнить локацию интерактивной травой, стандартный шейдер, взятый из Asset Store, был доработан с помощью создания дополнительных сценариев. В итоге, добавились следующие особенности: интерактивность, удобный интерфейс настройки параметров цвета, толщины и высоты [15].

Вместо обычного вершинного/фрагментного шейдера для геометрического шейдера между ними была добавлена третья часть. Порядок обработки: вершина, геометрия (v2g), затем фрагмент (g2f). Интерактивность достигается путем определения положения игрока и вычисления сферы вокруг него с помощью свойства `_Radius`, которое используется для смещения травинок. Участок кода в сценарии расчёта шейдера представлен ниже:

```
// Основное ядро
[numthreads(128, 1, 1)]
void Main(uint3 id : SV_DispatchThreadID)
{
    // ...
    // Интерактивность
    float3 dis = distance(_PositionMoving, worldPos);
    float3 radius = 1 - saturate(dis / _InteractorRadius);
    // На основе радиуса взаимодействия объектов изменение позиции
```

```
        float3 sphereDisp = worldPos - _PositionMoving;
        sphereDisp *= radius;
        // Учёт силы взаимодействия
        sphereDisp = clamp(sphereDisp.xyz*_InteractorStrength,-0.8,0.8);
        // ...
}
```

Результат взаимодействия персонажа с травой представлен на рисунке 15.

Рисунок 15 - Взаимодействие персонажа с травой

5.2.3. Наполнение игровой локации

Для достижения необходимого уровня наполнения и разнообразия игровой локации она должна включать в себя разнообразные препятствия и декорации, например, деревья и камни. Полный перечень объектов, их количество, а также наличие коллайдера представлены в таблице 5.1.

Таблица 5.1 – Объекты окружения

№	Наименование объекта	Количество, шт.	Коллайдер
1	Дерево	10	Присутствует
2	Цветы	11	Отсутствует
3	Гриб	7	Присутствует
4	Камень	5	Присутствует
5	Ветка/бревно	10	Присутствует
6	Пень	3	Присутствует
7	Кустарник	5	Отсутствует
8	Тропинка	9	Отсутствует

Опытным путем было подобрано расположение на локации объектов из таблицы 5.1 таким образом, чтобы, с одной стороны, придать игровому миру фэнтези-стиль и наполненность, а с другой, чтобы усложнить и разнообразить процесс прохождения игры. Результат представлен на рисунке 16.

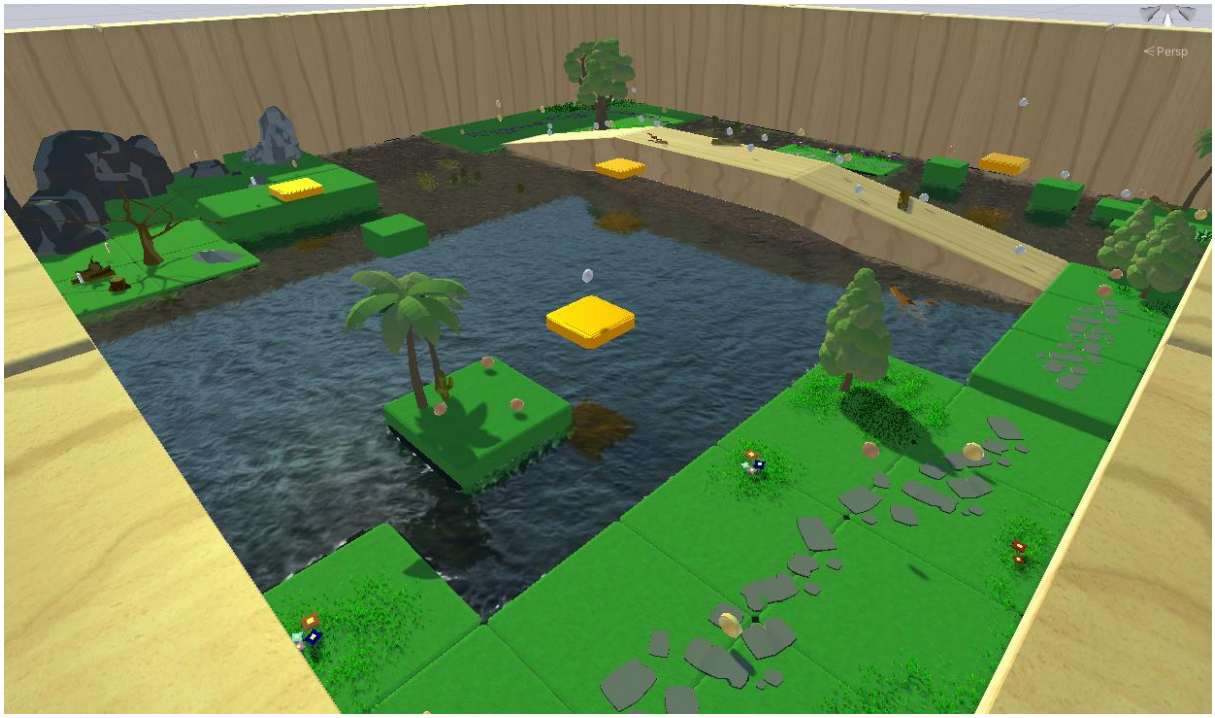

Рисунок 16 - Финальная версия игровой локации, вид в перспективе

5.2.4. Проверка соответствия локации требованиям

Ниже представлен краткий обзор полученной локации на предмет наличия условий для реализации всех заявленных игровых сценариев.

На локации в различных местах расположены монеты. Общее количество – 53 шт. Расположение монет позволяет собрать их все, они расположены таким образом, что игроку необходимо побывать во всех частях локации. Подводя итог, игровой сценарий «сбор монет» реализован полностью.

На рисунках 16 и 17 видны подвижные блоки желтого цвета, препятствия в виде деревьев, брёвен и камней, а также различные декорации. Это позволяет разнообразить визуальную составляющую, и делает игру более интересной.

Игровой сценарий «поражение при падении в воду» также полностью реализован. Об этом свидетельствует большая площадь поверхности воды, расположение монет и блоков и наличие мест, в которых есть потенциальная возможность упасть в воду в процессе прохождения игры.

Оформление локации выполнено в едином стиле фэнтези.

5.3. Выводы

В рамках данного раздела были рассмотрены популярные игровые сценарии, используемые в аркадах. По результатам анализа выбраны некоторые из них: победа при сборе всех монет, поражение при падении в воду, ограниченное время игры. Данные игровые сценарии были реализованы при помощи модификации сценариев управления персонажем, а также с помощью разработки игровой локации.

Созданная игровая локация соответствует всем поставленным требованиям. Она содержит различные препятствия, декорации, движущиеся платформы, монеты, и позволяет воспроизводить требуемые игровые сценарии. Ландшафт и объекты окружения оформлены в едином стиле фэнтези в яркой цветовой гамме.

6. РАЗРАБОТКА GUI

6.1. Формулировка требований к GUI

Выделены следующие элементы пользовательского интерфейса во время игры:

- Панель счетчика времени, на которой будет отображаться оставшееся время до окончания игры.
- Панель показателя выносливости.
- Панель отображения монет, на которой будут отображаться количество собранных и всех имеющихся на локации монет.
- Кнопки управления во время паузы, победы или поражения.
- Панель вывода результата игры.

Макет игрового интерфейса представлен на рисунке 17.

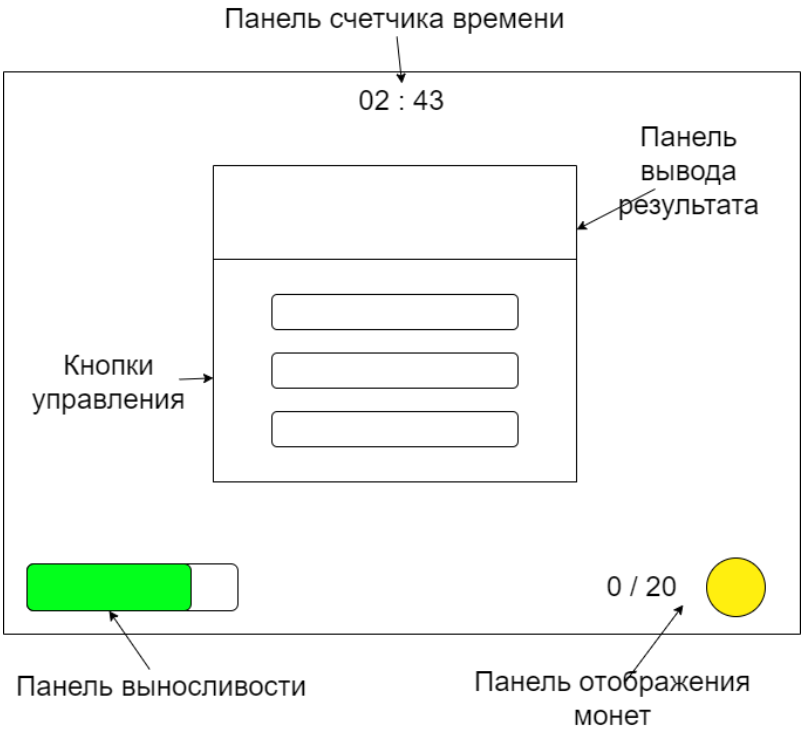

Рисунок 17 - Макет игрового интерфейса

Выделены следующие элементы пользовательского интерфейса в меню:

- Кнопки навигации.
- Панель отображения помощи в управлении.

Макет игрового интерфейса представлен на рисунке 18.

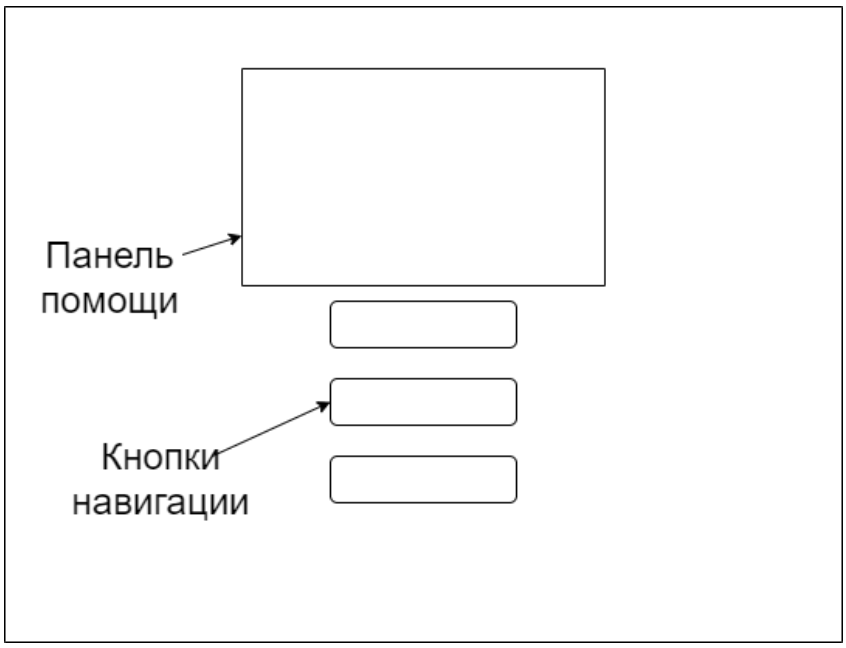

Рисунок 18 - Макет интерфейса меню

6.2. Проектирование и реализация неигровой сцены

Используя макет (рис. 18), был разработан интерфейс главного меню. Для создания кнопок в Unity существует элемент `Button UI`, для создания текста – `Text UI` [11]. Реализованный интерфейс представлен на рисунках 19 и 20.

Рисунок 19 - Разработанный интерфейс меню (1)

Рисунок 20 - Разработанный интерфейс меню (2)

В таблице 6.1 приведено описание действий, которые происходят при нажатии на соответствующие кнопки интерфейса.

Таблица 6.1 – Функции кнопок интерфейса в меню

№	Название кнопки	Действие при нажатии
1	Play	Загрузка игровой сцены, удаление сцены меню
2	Tutorial	Скрытие кнопок Play, Quit, Tutorial, показ панели помощи и кнопки Back
3	Quit	Завершение работы приложения
4	Back	Скрытие панели помощи и кнопки Back, показ кнопок Play, Quit, Tutorial

6.3. Проектирование и реализация игровой сцены

Используя макет (рис. 17), был разработан игровой интерфейс. Для обновления количества собранных монет используется сценарий:

```
public class ChangeCoinTextUI : MonoBehaviour
{
    static public int nCoins = 0;
    private Text cointText;

    void Start(){
        nCoins = 0;
        cointText = GetComponent<Text>();
    }

    void FixedUpdate(){
        cointText.text = nCoins.ToString();
    }
}
```

Для отслеживания текущего уровня выносливости в сценарии управления персонажа добавлен соответствующий код, управляющий поведением компонента Slider.

Slider – это стандартный ползунок, который можно перемещать между минимальным и максимальным значением. Компонент слайдера — это объект Selectable , который управляет заливкой, маркером или и тем, и другим. Заполнение, когда оно используется, простирается от минимального значения до текущего значения, в то время как дескриптор, когда он используется, следует за текущим значением [19].

На рисунке 21 продемонстрировано меню паузы, которое открывается при нажатии клавиши Esc. На этом рисунке также видны таймер, прогресс-бар выносливости и панель монет.

Рисунок 21 - Пауза во время игры

На рисунке 22 изображен вид пользовательского интерфейса при поражении. При победе вид практически не изменяется: надпись «Defeat» заменяется надписью «Victory».

Рисунок 22 - Поражение игрока

В таблице 6.2 приведено описание действий, которые происходят при нажатии на соответствующие кнопки во время игры.

Таблица 6.2 – Функции кнопок интерфейса во время игры

№	**Название кнопки**	**Действие при нажатии**
1	Menu	Загрузка сцены меню, удаление игровой сцены
2	Restart	Перезапуск уровня
3	Quit	Завершение работы приложения
4	Resume	Скрытие панели паузы и продолжение игры

6.4. Выводы

В рамках данного раздела представлено проектирование и разработка графического пользовательского интерфейса, состоящего из игрового и неигрового интерфейсов. Созданы все элементы, требуемые для корректной работы приложения.

7. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

Основной целью данной работы является разработка игры для персональных компьютеров с использованием технологий Unity 3D. Цель игры, в первую очередь, это приятное времяпрепровождение пользователя, с другой стороны, игра тренирует реакцию и стратегическое мышление. Игры жанра «аркада» пользуются большим спросом на рынке программного обеспечения для различных типов устройств, следовательно, высока вероятность коммерческого успеха разработанной игры.

В данном разделе приводится экономическое обоснование целесообразности создания аркадной игры и производится расчет затрат на ее разработку.

7.1. Расчет расходов на оплату труда

Разработкой программного обеспечения занимались два человека: научный руководитель и студент. Для расчета расходов на оплату труда была определена длительность каждого этапа разработки ПО, то есть какое количество рабочих дней исполнитель затратил на его выполнение. Данные о затраченных трудовых ресурсах приведены в таблице 7.1, в которой указаны наименования этапов работы над программным продуктом и соответствующие трудоемкости научного руководителя и исполнителя (студента).

Таблица 7.1 – Перечень и длительность работ

№	Наименование работы	Длительность работы, чел./дней	
		Руководитель	Исполнитель
1	Формирование технического задания	1	2
2	Обзор предметной области	1	3
3	Выбор технологий и среды разработки	1	2
4	Проектирование игры	3	7
5	Разработка игры	2	15
6	Тестирование прототипа игры	1	2

Таблица 7.1 (продолжение) – Перечень и длительность работ

№	Наименование работы	Длительность работы, чел./дней	
		Руководитель	Исполнитель
7	Подготовка экономического обоснования работы	1	3
8	Оформление пояснительной записки	4	15
9	Оформление презентации	1	4
ИТОГО		15	53

Для каждого из участников разработки ПО была рассчитана дневная ставка, которая равна отношению месячной заработной платы к количеству рабочих дней в одном месяце. Согласно Приложению №7 к приказу № ОД/0432 от 15.10.2020 «Положение об оплате труда работников университета (действующая редакция с 01.10.2020) [16], минимальный размер оплаты труда доцента кафедры, кандидата технических наук, составляет 40600 рублей. Средняя месячная заработная плата по профессии инженер-программист, по данным Интернет-ресурса gorodrabot.ru [17], составляет 54 тысячи рублей. Из-за того, что студент не является дипломированным специалистом, его заработная плата была принята 27 тысяч рублей. Количество рабочих дней в одном месяце было принято равным 21.

В результате дневная ставка научного руководителя составляет 1933,33 рублей, а студента - 1285,71 рублей.

Используя данные о дневной ставке и информации о длительности работ из таблицы 7.1, были рассчитаны расходы на выплату основной и дополнительной заработных плат каждого из исполнителей. Для расчёта основной заработной платы была использована формула 7.1:

$$З_{\text{осн.з/пл}} = \sum_{i=1}^{k} T_i \cdot C_i, \qquad (7.1)$$

где $З_{\text{осн.з/пл}}$ - расходы на основную заработную плату исполнителя в рублях; k - количество исполнителей; T_i - время, затраченное i-м

участником на работу в днях; C_i – ставка i-го участника в рублях в день. Данные и результаты расчётов приведены таблице 7.2.

Таблица 7.2 – Основная заработная плата исполнителей

Исполнитель	**Дневная ставка, руб./день**	**Общее время работы, дни**	**Основная заработная плата, руб.**
Студент	1285,71	53	68142,63
Научный руководитель	1933,33	15	28999,95

Для расчёта дополнительной заработной платы была использована формула 7.2:

$$З_{доп.з./пл} = З_{осн.з./пл} \cdot \frac{Н_{доп}}{100}, \quad (7.2)$$

где $З_{доп.з/пл}$ - расходы на дополнительную заработную плату исполнителя в рублях; $З_{осн.з/пл}$ - расходы на основную заработную плату исполнителя в рублях; $Н_{доп}$ – норматив дополнительной заработной платы (для ВКР принят в размере 8,3%). Исходные данные и результаты расчётов представлены в таблице 7.3.

Таблица 7.3 – Дополнительная заработная плата исполнителей

Исполнитель	**Основная заработная плата, руб.**	**Дополнительная заработная плата, руб.**
Студент	68142,63	5655,83
Научный руководитель	28999,95	2406,99

Также, кроме основной и дополнительной заработной платы, были рассчитаны затраты на отчисления на социальные нужды. Для расчёта была использована формула 7.3:

$$З_{соц} = (З_{осн.з/пл} + З_{доп.з/пл}) \cdot \frac{Н_{соц}}{100}, \quad (7.3)$$

где $З_{соц}$ - отчисления на социальные нужды исполнителя в рублях; $З_{осн.з/пл}$ - расходы на основную заработную плату исполнителя в рублях; $З_{доп.з/пл}$ - расходы на дополнительную заработную плату исполнителя в рублях; $Н_{соц}$ - процент социальных отчислений, равный 30,2%. Исходные данные и результаты расчётов представлены в таблице 7.4.

Таблица 7.4 – Отчисления на социальные нужды

Исполнитель	**Основная заработная плата, руб.**	**Дополнительная заработная плата, руб.**	**Социальные отчисления, руб.**
Студент	68142,63	5655,83	20579,07
Научный руководитель	28999,95	2406,99	8757,98
ИТОГО	97142,58	8062,82	29337,05

7.2. Расчет накладных расходов

Накладные расходы – это расходы, связанные с расходами на управление и обслуживание, в том числе коммунальные платежи и затраты на продвижение продукта. У каждой организации свой индивидуальный норматив накладных расходов. В данной работе норматив составляет 20%.

Накладные расходы были рассчитаны по формуле 7.4:

$$Н = \frac{\sum_{i=1}^{2}(З_{доп.з./пл} + З_{осн.з./пл}) \cdot Н_{нр}}{100}. \quad (7.4)$$

Исходные данные и результаты расчётов представлены в таблице 7.5.

Таблица 7.5 – Накладные расходы

Исполнитель	**Основная заработная плата, руб.**	**Дополнительная заработная плата, руб.**	**Накладные расходы, руб.**
Студент	68142,63	5655,83	21041,08
Научный руководитель	28999,95	2406,99	

7.3. Расчет материальных расходов

Материальные расходы – это расходы на сырье, материалы и комплектующие изделия, спецоборудование необходимые для разработки проекта.

Общие материальные расходы были рассчитаны по формуле 7.5:

$$З_м = \sum_{i=1}^{L} G_i \cdot Ц_i, \quad (7.5)$$

где G_i – норма расхода на единицу материала, $Ц_i$– цена единицы материала, L – индекс вида сырья. Исходные данные и расчет затрат на материальные расходы представлены в таблице 7.6.

Таблица 7.6 – Материальные расходы

Наименование материала	**Норма расхода на единицу продукции, шт.**	**Цена, руб./шт.**	**Сумма на единицу продукции, руб.**
Бумага для офисной техники «SvetoCopy»	1	592	592
USB флэш-накопитель «Kingston»	1	442	442
Ручка шариковая «Pilot»	1	60	60
ИТОГО	1094		

7.4. Расчет расходов на амортизационные отчисления

При разработке программного продукта был использован персональный компьютер, стоимостью 40000 рублей.

Нормативный срок полезного использования персонального компьютера – 5 лет. С помощью этих данных была рассчитана годовая норма амортизации по формуле 7.6:

$$\text{Нα} = \frac{100}{\text{НСС}}\%, \quad (7.6)$$

где HCC – нормативный срок полезного использования средства.

Затем, были рассчитаны амортизационные отчисления за год по формуле 7.7:

$$\text{А}_i = \text{Ц}_{\text{п.н.}i} \cdot \frac{H_{ai}}{100}, \quad (7.7)$$

где $\text{Ц}_{\text{п.н.}i}$ – первоначальная стоимость; H_{ai} – годовая норма амортизации.

В итоге, была рассчитана величина амортизационных отчислений по средствам, использованным в процессе разработки. Расчеты были произведены по формуле 7.8:

$$A_{i\text{ВКР}} = A_i \cdot \frac{T_{i\text{ВКР}}}{12}, \quad (7.8)$$

где $T_{i\text{ВКР}}$ – время использования средства, мес. Время работы составляет 53 дня, приблизительно 2 месяца.

Результаты расчетов по амортизационным отчислениям представлены в таблице 7.7.

Таблица 7.7 – Амортизационные отчисления

Оборудование	**Норма амортизации, %**	**Годовые амортизационные отчисления, руб.**	**Амортизационные отчисления, руб.**
ПК	20%	8000	1333,33
Сумма амортизационных отчислений, руб.			1333,33

7.5. Расчет затрат на продукты сторонних организаций

Для разработки ПО требуется среда разработки Unity, а также различные компоненты этой среды. Для установки Unity, доступа к технической документации, а также возможности коммуникации исполнителей требуется прямой доступ к сети Интернет. Расчет затрат на услуги Интернет-провайдера представлен в таблице 7.8.

Таблица 7.8 – Затраты на продукты сторонних организаций

Наименование	**Цена, руб.**	**Время использования, мес.**	**Общая стоимость, руб.**
«Prometei»	800	2	1600
Сумма затрат на продукты сторонних организаций			1600

7.6. Расчет совокупных расходов

Был произведен расчет совокупных расходов. Результат представлен в таблице 7.9.

Таблица 7.9 – Совокупные затраты на разработку

№	**Наименование статьи**	**Сумма, руб.**
1	Расходы на основную заработную плату	97142,58
2	Расходы на дополнительную заработную плату	8062,82
3	Отчисления на социальные нужды	29337,05
4	Накладные расходы	21041,08
5	Материальные расходы	1094
6	Расходы на амортизационные отчисления	1333,33
7	Расходы на продукты сторонних организаций	1600
ИТОГО		159610,9

7.7. Выводы

В данном разделе работы были рассчитаны затраты на разработку ПО, они составили 159610,9 рублей. Суммарная трудоемкость разработки 69 чел./дней.

Жанр разрабатываемой игры – аркада. Аркадные игры пользуются большой популярностью у потребителей благодаря легкости игрового процесса и низкому порогу вхождения.

Разработанная игра является казуальной, что позволяет получить удовольствие от игрового процесса за короткий промежуток времени. Разработка полезна пользователю, в частности, детям и подросткам: развивает их скорость реакции, пространственное мышление и мелкую моторику.

Игровая индустрия постоянно развивается, объем рынка ПК-игр в России в 2021 году достиг оборота в 85,4 млрд руб., что указывает на растущий интерес пользователей и инвесторов к данной области. При стоимости продажи игры, равной 200 руб., потребуется 800 проданных экземпляров для того, чтобы покрыть расходы на разработку. Например, 2D-игра «Ori and the Blind Forest», концептуально похожая на разрабатываемую игру, всего за 9 месяцев с момента выхода была продана в количестве 450 тысяч копий.

Подводя итог, можно предположить, что разработанная игра может быть востребована на игровом рынке, и затраченные на разработку ресурсы могут окупиться за короткий промежуток времени.

ЗАКЛЮЧЕНИЕ

Современные аркадные игры сильно изменились с момента появления, но их отличительной особенностью все еще остается динамичный и простой игровой процесс, незамысловатый сюжет. Благодаря этим качествам, аркадные видеоигры охватывают большой круг пользователей, они актуальны и востребованы в настоящее время.

В ходе выполнения работы разработана игра в жанре «аркада», направленная на развитие реакции, мелкой моторики и пространственного мышления пользователя, с использованием современных технологий игрового движка Unity 3D. Разработанная игра совместима с операционными системами Windows, Linux и Mac.

В процессе выполнения работы решены следующие задачи:

1. Выполнен обзор предметной области.
2. Произведен анализ технологий, необходимых для разработки приложения. По результатам анализа был выбран игровой движок Unity 3D.
3. Создан игровой персонаж и его компоненты: модель, система управления, анимации.
4. Разработаны объекты монет, их модели, анимации, а также реализовано взаимодействие с персонажем.
5. Спроектированы игровые сценарии и созданы игровые локации.
6. Реализован графический пользовательский интерфейс игры, включающий в себя неигровой и игровой интерфейсы.
7. Выполнено экономическое обоснование целесообразности разработки. В результате которого выявлено, что разработанная игра окупаема и в перспективе может иметь коммерческий успех.

Игры аркадного жанра востребованы на рынке игр для различных платформ, в том числе для ПК. Проект в дальнейшем может быть доработан, могут быть добавлены новые игровые сценарии, локации или интерактивные предметы окружения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Исследовательская и консалтинговая компания Gartner [Электронный ресурс]. URL: https://www.gartner.com/en (дата обращения: 26.04.2022).
2. Индустрия компьютерных игр [Электронный ресурс]. URL: https://app2top.ru/wp-content/uploads/2020/07/Industriya-komp-yuterny-h-igr-2020.pdf (дата обращения: 26.04.2022).
3. «Crysis» [Электронный ресурс]. URL: https://www.crysis.com/ (дата обращения: 27.04.2022).
4. «Far Cry» [Электронный ресурс]. URL: https://www.ubisoft.com/ru-ru/game/far-cry/ (дата обращения: 27.04.2022).
5. Cry Engine [Электронный ресурс]. URL: https://www.cryengine.com/ (дата обращения: 27.04.2022).
6. GoDot [Электронный ресурс]. URL: https://godotengine.org/ (дата обращения: 28.04.2022).
7. «Fortnite» [Электронный ресурс]. URL: https://www.epicgames.com/fortnite/ru/home (дата обращения: 28.04.2022).
8. Unreal Engine 4 [Электронный ресурс]. URL: https://www.unrealengine.com/en-US/ (дата обращения: 28.04.2022).
9. Unity 3D [Электронный ресурс]. URL: https://unity.com/ru/ (дата обращения: 28.04.2022).
10. Библиотека ассетов [Электронный ресурс]. URL: https://assetstore.unity.com/ (дата обращения: 28.04.2022).
11. Джозеф Хокинг, Unity в действии. Мультиплатформенная разработка на C# // ООО Издательство «Питер» 2016 г. 336 с. (дата обращения: 30.04.2022).
12. Алан Торн, Искусство создания сценариев в Unity // Издательство ООО «ДМК Пресс» 2019 г. 360 с. (дата обращения: 01.05.2022).

13. «God of War» [Электронный ресурс]. URL: https://store.epicgames.com/ru/p/god-of-war (дата обращения: 05.05.2022).
14. «City Ville» [Электронный ресурс]. URL: https://www.facebook.com/pages/category/Community/City-Ville-172260972824827/ (дата обращения: 05.05.2022).
15. Кенни Ламмерс, Шейдеры и эффекты в Unity. Книга рецептов // Издательство ООО «ДМК Пресс» 2016 г. 274 с. (дата обращения: 06.05.2022).
16. Положение об оплате труда работников университета [Электронный ресурс]. URL: https://etu.ru/assets/files/ru/universitet/normativnye-dokumenty/Prikaz%20OD432/prilozhenie-7-k-prikazu-nod0342-ot-15.10.2020.pdf (дата обращения: 01.05.2022).
17. Система поиска вакансий [Электронный ресурс]. URL: https://gorodrabot.ru/ (дата обращения: 01.05.2022).
18. Джесси Шелл, Геймдизайн. Как создать игру, в которую будут играть все // Издательство «Альпина Диджитал» 2019 г. 756 с. (дата обращения: 18.05.2022).
19. Бонд Джонатан, Unity и C#. Геймдев от идеи до реализации // Издательство «Питер» 2019 г. 928 с. (дата обращения 18.05.2022).
20. Система визуального скриптинга Blueprint [Электронный ресурс]. URL: https://docs.unrealengine.com/5.0/en-US/blueprints-visual-scripting-in-unreal-engine/ (дата обращения 18.05.2022).
21. Шаблон проектов CryEngine [Электронный ресурс]. URL: https://www.cryengine.com/marketplace/product/crytek/cryengine-gamesdk-sample-project (дата обращения 18.05.2022).