# Введение

Игроки стремятся к новым впечатлениям, а комбинация жанра Adventure с элементами процедурной генерации и нелинейного игрового процесса жанра RogueLite позволяет добиться глубины и непредсказуемости игрового опыта. Это делает проект интересным для широкой аудитории, которая ценит как сюжетную составляющую, так и механики, побуждающие к повторному прохождению.

Современные игровые технологии позволяют значительно расширить границы интерактивного опыта. Использование процедурной генерации и нелинейных сюжетов даёт разработчикам инструменты для создания миров, способных меняться в зависимости от действий игрока. Это способствует более глубокой вовлечённости и формированию индивидуального игрового опыта, где каждое прохождение может существенно отличаться от предыдущего.

Видеоигры давно вышли за рамки простого развлечения, становясь мощным инструментом для исследования человеческой психологии, внутренних конфликтов и поиска смысла. Благодаря интерактивности игрок получает возможность не только переживать историю, но и активно участвовать в её развитии, принимая решения, которые могут влиять на развитие персонажей и окружающий мир. Такой подход способствует самопознанию, развитию эмпатии и осмысленному восприятию собственных поступков.

Цель дипломного проекта – создание компьютерного игрового приложения в жанре Adventure с элементами RogueLite. Таким образом для реализации поставленной цели необходимо выполнить следующие задачи:

– провести сравнительный анализ игровых приложений в жанре Adventure с элементами RogueLite;

– разработать системное проектирование;

– выполнить функциональное поректирование;

– спроектировать UI/UX приложения;

– провести тестирование и анализ работы прототипа;

– разработать технико-экономическое обоснование разработки компьютерного игрового приложения в жанре Adventure с элементами RogueLite.

Выполнение дипломного проекта предусматривает практическое применение знаний, полученных в ходе изучения профильных дисциплин, а также приобретение и совершенствование практических навыков в области использования современных технологий, методов разработки и языков программирования. В рамках проекта осуществляется проектирование интерфейсов игрового приложения, а также углубляется и закрепляется теоретическая база и практический опыт, полученные в процессе обучения.

# 1 С**равнительный анализ игровых приложений**

## 1.1 Характеристика жанров

Игры жанра Adventure предлагают игрокам возможность погружаться в миры, наполненные исследованием и решением головоломок, где ключевыми элементами являются поиск, изучение и взаимодействие с окружающим миром. Основное внимание уделяется истории, сюжету, диалогам и развитию персонажей.

Сюжетные линии в играх этого жанра разнообразны: от эпических приключений до глубоких личных историй персонажей, предлагая широкий спектр эмоций и переживаний. Игроки исследуют мир, решают головоломки и взаимодействуют с персонажами и предметами, что делает мировоззрение и взаимодействие с окружением центральными элементами игрового процесса.

Adventure-игры часто включают головоломки различной сложности, требующие логического мышления для продвижения по сюжету. Важной частью головоломок являются механики сбора и использования предметов, которые помогают продвигаться дальше в игре.

Повествование в таких играх обычно линейное, но в некоторых играх присутствуют нелинейные элементы, дающие игрокам большую свободу выбора и возможность влиять на исход истории.

Боевые элементы в этом жанре минимальны или отсутствуют, хотя могут присутствовать в смешанных жанрах.

Важной особенностью игр жанра Adventure является их художественный дизайн, музыка и атмосфера, которые дополняют захватывающий и увлекательный игровой опыт.

Основные механики жанра Adventure включают:

– исследование игрового мира и взаимодействие с окружением;

– решение головоломок и задач;

– развитие сюжета через диалоги и выполнение квестов.

Таким образом, к играм в жанре Adventure можно отнести игры, которые создают уникальные и запоминающиеся миры для исследования, предлагают игроку увлекательные сюжетные линии и квесты, делают акцент на взаимодействии с персонажами и окружающим миром.

Жанр Roguelite является поджанром Roguelike. Главной особенностью этого поджанра является система прогрессии, которая позволяет игроку сохранять свой прогресс, а не начинать сначала. То есть смерть персонажа означает потерю прогресса в уровне или конкретной попытке, но сохраняются некоторые элементы, такие как улучшения, ресурсы или навыки, которые переносятся в следующие прохождения.

Прогресс в игре включает постоянное улучшение персонажа или его возможностей, что делает каждое последующее прохождение немного проще или предоставляет новые возможности для изучения и победы.

Одно прохождение может быть довольно коротким, часто от нескольких минут до часа, что побуждает игроков делать несколько попыток за раз.

Геймплей предполагает получение разнообразных случайных бонусов, предметов и апгрейдов, которые могут значительно влиять на ход игры и требуют адаптации игрока к текущей ситуации в игре.

Игры жанра Roguelite часто предлагают множество различных способностей, оружия и стилей игры, позволяя каждому прохождению быть неповторимым.

Многое в игре также зависит от удачи – какие предметы, враги или боссы появятся на пути, что добавляет азарт и непредсказуемость в каждую игру.

Основные механики жанра Roguelite включают:

– процедурную генерацию уровней, делая каждую попытку уникальной;

– постоянные улучшения или бонусы, сохраняющиеся между забегами;

– сложные бои и необходимость адаптироваться к различным ситуациям.

Можно сделать вывод, что к играм в жанре Roguelite относятся игры, которые предлагают уникальные испытания и адаптивный игровой процесс, обеспечивают высокую реиграбельность благодаря процедурной генерации уровней, стимулируют игрока постоянно совершенствоваться и находить новые способы прохождения.

## 1.2 Анализ на примерах

«Hades» представляет собой экшен Roguelite с изометрическим видом, сочетающий динамичную боевую систему с глубокой нарративной составляющей. Игрок управляет Загреем, сыном Аида, пытаясь покинуть подземный мир. Геймплей построен на последовательных забегах через процедурно-генерируемые комнаты, где игрок сражается с врагами, выбирает улучшения и взаимодействует с NPC.

Основной игровой процесс включает быстрые бои с возможностью выбора различных стихийных благословений, получаемых от богов Олимпа. Каждое новое прохождение отличается случайным набором бонусов, что формирует уникальные билды и вариативность боевых тактик. Метапрогрессия представлена системой постоянных улучшений, включая прокачку оружия, характеристик персонажа и открытие новых сюжетных линий.

Игра использует модульную генерацию уровней, где порядок комнат изменяется в каждом забеге. Это обеспечивает разнообразие игрового процесса и снижает предсказуемость прохождений. Однако глобальное изменение окружения и влияние решений на мир ограничены.

Геймплей «Hades» представлен на рисунке 1.1.



Рисунок 1.1. – Скриншот геймплея в «Hades»

Одна из ключевых особенностей «Hades» – динамическое развитие сюжета через взаимодействие с персонажами. Каждый забег открывает новые диалоги, влияя на развитие отношений с NPC. Прогрессия истории связана с повторяющимися попытками побега, что органично вплетает нарратив в Roguelite-механику.

«Hades» выделяется выразительным художественным стилем, вдохновлённым античной мифологией. Музыкальное сопровождение динамически адаптируется к боевым сценам, усиливая погружение.

Ключевыми достоинствами «Hades» являются интеграция нарратива в Roguelite-механику, вариативность билдов и кинематографичная подача сюжета.

«Omori» – это психологическая RPG с уникальной визуальной стилистикой и глубоким повествовательным слоем. Игра исследует темы тревожности, травмы и самоидентификации через чередование двух миров: реальности и сна.

Геймплей включает в себя классические пошаговые бои с механикой «эмоциональных состояний», влияющих на характеристики персонажей. Исследование мира сопровождается решением головоломок и социальными взаимодействиями с NPC. Взаимодействие с окружающей средой раскрывает скрытые аспекты сюжета.

Игра разделена на два пространственных слоя: реальный мир, где игрок взаимодействует с окружающими персонажами, и альтернативную реальность, наполненную символизмом. Контраст между этими мирами усиливает эмоциональное воздействие игры.

Сюжет раскрывается нелинейно, в зависимости от выбора игрока и его действий. В игре присутствуют альтернативные концовки, зависящие от решений в ходе прохождения.

Геймплей «Omori» представлен на рисунке 1.2.

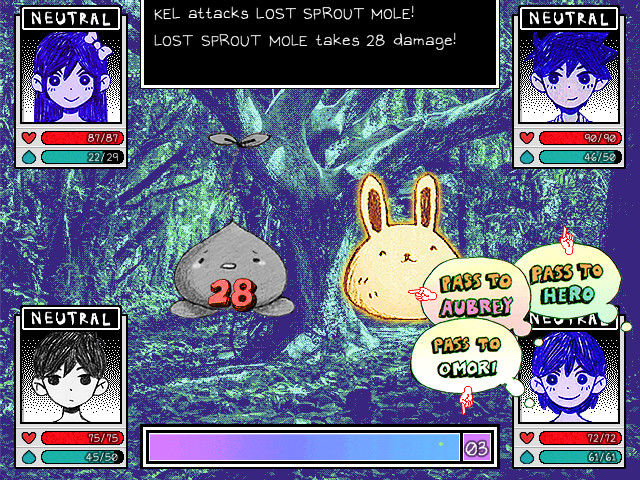


Рисунок 1.2. – Скриншот геймплея в «Omori»

«Omori» использует пиксельный стиль с ручной анимацией, что подчёркивает эмоциональную глубину сцен. Саундтрек адаптируется к происходящему, усиливая атмосферу.

Игра «Omori» выделяется за счёт эмоционально насыщенного сюжета, уникальной механики эмоций в боевой системе и визуального контраста между мирами. Однако основной фокус на нарративе ограничивает глубину игровой механики по сравнению с классическими RPG.

## 

## 1.3 Выводы

На основе анализа жанров Adventure и RogueLite в разрабатываемом игровом приложении планируется объединить ключевые механики, обеспечивающие глубокий нарратив и высокую реиграбельность.

Из жанра Adventure предполагается реализовать системы исследования, взаимодействия с окружением и развитие сюжета посредством диалоговых структур. Предусмотрена вариативность пользовательского опыта за счёт нелинейных диалогов.

В проектируемом приложении особое внимание уделяется архитектуре окружения: планируется реализация двух пространств – фиксированной хаб-локации («Реальность») и процедурно-генерируемого мира («Сознание»). Окружение будет оформлено таким образом, чтобы передавать соответствующую тематику и атмосферу каждой комнаты. Для усиления атмосферы и эмоционального воздействия будет внедрено адаптивное звуковое сопровождение, включающее фоновую музыку и звуковые эффекты, изменяющиеся в зависимости от состояния и событий в приложении.

От жанра Roguelite в проектируемом приложении планируется внедрить модульную процедурную генерацию уровней в пространстве «Сознания», что позволит формировать уникальную структуру комнат и событий при каждом запуске. Также будет реализована система метапрогрессии, обеспечивающая сохранение определённых улучшений между сессиями, что отражает постепенное развитие персонажа.

Таким образом, в игровом приложении планируется интеграция исследовательского геймплея и нарративных элементов Adventure с реиграбельностью, сложностью и адаптивностью RogueLite. Это обеспечит уникальный пользовательский опыт, сочетающий эмоциональную вовлечённость и стратегическую гибкость.

В качестве референсов используются такие проекты, как «Hades» (модульная процедурная генерация уровней, система метапрогрессии) и «Omori» (разделение пространства на фиксированную хаб-локацию и процедурно-генерируемый мир, система эмоций). Комплексное применение этих механик позволит создать архитектуру, сочетающую глубокий нарратив, вариативность пользовательского опыта, тематическое окружение и звуковое оформление.

# 2 Системное проектирование

## 2.1 Нарратив

В разрабатываемом игровом приложении нарративная система проектируется как один из ключевых компонентов пользовательского опыта. Основная сюжетная линия строится вокруг внутренней трансформации главного персонажа, преодоления личных страхов и поиска выхода из замкнутого жизненного цикла. Тематика нарратива включает саморефлексию, принятие себя, влияние прошлого на настоящее и возможности для изменений.

Подача нарратива реализуется через несколько взаимосвязанных механизмов:

1 Пользователь взаимодействует с окружением и NPC-проводником посредством диалогов, которые раскрывают детали сюжета, внутренние переживания и мотивацию главного героя. Система диалогов динамически обновляется: новые диалоги становятся доступны после посещения определённых комнат или выполнения определённых условий. Некоторые реплики в диалогах могут инициировать особые действия, такие как мгновенный возврат пользователя в «Реальность» или отключение появления врагов в текущей сессии.

2 После взаимодействия с определёнными предметами или объектами пользователь слышит внутренние монологи персонажа, что позволяет глубже понять его состояние и отношение к происходящему.

3 Элементы окружения (определенные предметы в комнатах) содержат дополнительную информацию о прошлом и текущем положении персонажа, а также могут запускать новые сюжетные события через монологи или диалоги.

4 На протяжении всего прохождения в пространстве «Сознания» пользователя сопровождает NPC-проводник, который выполняет функцию навигации, предоставляет ключевую информацию и поддерживает раскрытие нарратива.

Нарративная система проектируется с учётом вариативности: пользовательские решения и стиль прохождения влияют на доступность определённых сюжетных элементов, развитие диалогов и возможные действия в приложении. Предусмотрена возможность альтернативных событий и нелинейного раскрытия отдельных аспектов истории за счёт выбора реплик и последовательности исследования локаций.

Взаимосвязь нарратива с другими системами приложения реализуется через запуск монологов после взаимодействия с объектами, обновление доступных диалогов в зависимости от прогресса пользователя и выполнение особых действий на основе выбранных реплик. Таким образом, нарративная составляющая интегрируется в общую архитектуру приложения и поддерживает его эмоциональную и смысловую глубину.

## 2.2 Геймплей

Геймплей разрабатываемого игрового приложения строится на сочетании исследования, взаимодействия с окружением, принятия решений и сражений с врагами в рамках двух пространств: «Реальность» и «Сознание». Пользовательский опыт организован в виде циклического прохождения, где чередуются этапы исследования, взаимодействия с объектами, диалоговые взаимодействия, сражения и возвращение в исходную локацию.

В пространстве «Реальность» реализован режим управления от первого лица. Пользователь может свободно перемещаться по фиксированной локации, взаимодействовать с предметами, а также инициировать переход в пространство «Сознания». Некоторые объекты доступны для осмотра через специальное окно, предоставляющее дополнительную информацию. Взаимодействие с определёнными объектами может запускать внутренние монологи персонажа, раскрывающие детали сюжета и состояния главного героя.

В пространстве «Сознание» реализован режим изометрического управления. Пользователь исследует процедурно-генерируемые уровни, что включает перемещение между комнатами, взаимодействие с окружением и NPC-проводником. В процессе исследования пользователь может находить объекты, инициировать диалоги, а также принимать решения, влияющие на дальнейшее развитие событий. Важной частью геймплея в пространстве «Сознания» являются сражения с врагами, которые возникают в определённых комнатах.

Перед началом сражения пользователь выбирает набор активных эмоций из доступного перечня. Выбранные эмоции оказывают непосредственное влияние на геймплей: они модифицируют характеристики персонажа и определяют доступные действия в бою с врагами.

Некоторые действия, такие как выбор определённых реплик в диалоге, могут приводить к особым сценариям, например, мгновенному возврату в «Реальность» или изменению условий появления врагов. Прогресс пользователя фиксируется по мере прохождения комнат и взаимодействия с сюжетно значимыми объектами.

Система диалогов обновляется в зависимости от прогресса: новые диалоги становятся доступны после посещения определённых комнат или взаимодействия с объектами, что обеспечивает вариативность пользовательского опыта и адаптацию геймплея под индивидуальный стиль прохождения.

Таким образом, геймплей приложения строится на чередовании режимов исследования, принятия решений, взаимодействия с окружением, использования эмоций и сражений с врагами, что обеспечивает динамичный и насыщенный пользовательский опыт.

## 2.3 Сеттинг

Сеттинг разрабатываемого игрового приложения основан на разделении мира на два ключевых пространства: «Реальность» и «Сознание». Такое деление позволяет реализовать различные подходы к визуальному и атмосферному оформлению, а также поддерживать тематическую глубину пользовательского опыта.

Пространство «Сознание» реализовано как процедурно-генерируемый набор комнат с абстрактным и символическим оформлением. Каждая комната имеет индивидуальную цветовую палитру, световые акценты и набор символических элементов, соответствующих определённой теме.

Комнаты связаны с ключевыми внутренними конфликтами персонажа:

1 Комната буллинга в школе оформлена в виде школьного класса с отдельно стоящей партой, что символизирует изоляцию. Враги представлены силуэтами людей с насмешливыми улыбками, что подчёркивает атмосферу отчуждённости и давления со стороны окружающих.

2 Комната одиночества содержит два стула: один – из пространства «Реальность» и отражает настоящее, второй – старый и потрёпанный, ассоциируется с человеком из прошлого персонажа. В качестве врага выступает тёмная прозрачная тень, являющаяся копией главного героя, что визуализирует внутреннюю борьбу с самим собой.

3 Комната неопределённости будущего оформлена с использованием множества дверей, размещённых на стенах и полу, а также текстур с вопросами о будущем (например, «Кто ты?», «Это твой выбор или чужой?»). Враги представлены в виде знаков вопроса, что подчёркивает тему тревоги и неуверенности в завтрашнем дне.

4 Базовые комнаты имеют абстрактное оформление с использованием нейтральных цветовых решений. Враги в этих комнатах выполнены в форме стеклянных осколков, которые наносят урон при контакте. Такой образ подчёркивает, что даже на первый взгляд незначительные и безобидные мелочи способны причинять вред и вызывать внутреннюю боль.

5 Стартовая комната в форме гексагона служит отправной точкой для пользователя, содержит NPC-проводника и двух стражников в виде котиков, объект для активации выбора эмоций, а также два портала: один ведёт к дальнейшему исследованию, второй возвращает пользователя в «Реальность».

6 Босс-комната отличается увеличенным размером и формой гексагона с трещинами на поверхности. В этой локации находится основной противник: видоизменённый NPC-проводник, что подчёркивает кульминационный характер данной комнаты.

Динамика сеттинга выражается в смене пространств и тематик комнат, однако внутри каждой локации оформление остаётся фиксированным и соответствует заранее определённой теме. Это позволяет пользователю быстро ориентироваться в пространстве и ассоциировать визуальные элементы с определёнными этапами или состояниями персонажа.

Пространство «Реальность» оформлено как современная жилая комната с приглушённой цветовой гаммой и минимальным освещением, где единственным источником света выступает светильник в виде котика. Окружение наполнено предметами, отражающими повседневность и внутреннее состояние персонажа: незастеленная кровать, гора одежды, таблетки, фоторамка, личный дневник, книга по психологии, письмо будущему себе из школьных лет и другие личные вещи. Такое наполнение формирует атмосферу замкнутости, рутины и подчёркивает психологическую тематику.

Звуковое оформление приложения включает адаптивную фоновую музыку и звуковые эффекты, которые различаются для «Реальности» и «Сознания». В «Реальности» используются спокойные, монотонные звуки, подчёркивающие атмосферу рутины, тогда как в «Сознании» звуковое сопровождение становится более напряжённым, динамичным и изменяется в зависимости от ситуации, поддерживая эмоциональный фон соответствующей локации.

Таким образом, сеттинг игрового приложения обеспечивает не только визуальное и звуковое погружение пользователя, но и поддерживает тематическую и эмоциональную направленность всего пользовательского опыта.

## 2.4 Механики

**Механика перемещения в «Реальности».** Управление персонажем реализовано в режиме от первого лица. Перемещение осуществляется с помощью клавиш «WASD»: «W» – движение вперёд, «S» – назад, «A» – влево, «D» – вправо. Вращение камеры реализованы с помощью движения мыши.

**Механика перемещения в «Сознании».** Управление персонажем реализовано в изометрической проекции. Движение осуществляется с использованием клавиш «WASD».

**Перемещение между мирами.** Переход между пространствами «Реальность» и «Сознание» осуществляется через нажатие клавиши «R» в пределах области специального объекта. Возврат в «Реальность» возможен из стартовой комнаты, в результате смерти персонажа и через последний портал в комнате босса.

**Система эмоций.** В стартовой комнате пространства «Сознание» пользователь выбирает в окне набор эмоций из доступного перечня. Эмоции делятся на несколько видов, каждая из которых обладает уникальными характеристиками и влияет на параметры персонажа:

1 Решимость – увеличение наносимого урона на 10% при уровне здоровья персонажа ниже 50%, а также повышение скорости в бою с элитными противниками на 10%.

2 Сострадание – восстановление 5% максимального запаса здоровья после победы над врагом.

3 Страх – увеличение шанса избежать входящего урона на 20% и одновременное снижение наносимого урона на 10%.

4 Принятие – однократное восстановление 20% максимального запаса здоровья при смерти персонажа в каждой комнате.

5 Гнев – увеличение наносимого урона на 15% и одновременное снижение максимального запаса здоровья на 10%.

6 Отрицание – игнорирование одного входящего удара в каждой комнате.

Каждая эмоция имеет 3 уровня улучшений, которые открываются за счет использования специальной валюты (фрагментов), получаемых за устранение врага.

После выбора активных эмоций в стартовой комнате изменить их в текущем цикле сессии в пространстве «Сознание» невозможно.

**Боевая система.** В боевых комнатах пространства «Сознание» реализованы сражения с врагами. Пользователь атакует врагов с помощью «ЛКМ», для рывка используется клавиша «Space». Урон и дополнительные эффекты зависят от выбранных эмоций. Враги преследуют и атакуют игрока при приближении. Победа достигается при уничтожении всех врагов в комнате, поражение – при исчерпании здоровья персонажа. После победы игрок получает внутриигровую валюту, а бой завершается очисткой комнаты и открытием портала для перехода дальше.

**Система диалогов.** Диалоговые взаимодействия реализованы с NPC-проводником. Новые диалоги становятся доступны после посещения комнат в «Сознании». Взаимодействие с NPC инициируется при нахождении персонажа в области взаимодействия и нажатии клавиши «F». Переключение между вариантами реплик осуществляется с помощью клавиш «W/S» или стрелок вверх/вниз, а выбор реплики и переход к следующему слайду диалога выполняется нажатием левой кнопки мыши или клавиши «Enter».

Некоторые реплики в диалоге могут приводить к особым сценариям, например, мгновенному возврату в «Реальность» или изменению условий появления врагов.

**Система монологов.** Взаимодействие с определенными объектами окружения инициирует внутренние монологи персонажа. Монологи раскрывают детали нарратива и состояние главного героя.

**Система осмотра объектов.** Пользователь может осматривать отдельные объекты, получая дополнительную информацию через специальное окно. Для активации режима осмотра используется клавиша «E» при наведении на объект, доступный для детального изучения, и нахождении в пределах области взаимодействия. В режиме осмотра реализовано вращение объекта с помощью правкой кнопки мыши для более подробного изучения, зума на колесико мыши и выход из окна осмотра на «ESC».

**Система взаимодействия с объектами.** Взаимодействие осуществляется по нажатию клавиши «F» при наведении на соответствующий объект и нахождении в пределах области взаимодействия. Запускает монологи персонажа.  
 **Система порталов.** Порталы используются для перехода между комнатами в пространстве «Сознание» и возвращении игрока в пространство «Реальность».

В стартовой комнате размещаются два портала: один ведёт к следующей комнате, другой – возвращает пользователя в «Реальность». В остальных комнатах порталы обеспечивают продвижение по маршруту, определённому процедурной генерацией. Взаимодействие с порталом осуществляется по нажатию клавиши «F» в зоне взаимодействия, после чего происходит загрузка следующей комнаты и перемещение персонажа к новой точке спавна.

**Система модульной процедурной генерации уровней.** В пространстве «Сознания» при каждом запуске сессии формируется набор комнат на основе заданных параметров конфигурации. Генерация начинается с создания стартовой комнаты в фиксированной позиции, после чего последовательно размещаются остальные комнаты на свободных позициях сетки. Для каждой комнаты определяется тип (стартовая, базовая, элитная, босс-комната) с учётом удалённости от стартовой точки и случайного выбора из пула доступных вариантов.

Алгоритм гарантирует наличие всех ключевых типов комнат: стартовая комната всегда создаётся первой, босс-комната размещается в самой удалённой позиции от старта, одна из оставшихся комнат назначается элитной, остальные получают статус базовых. Для каждой комнаты подбирается соответствующий шаблон (префаб) на основе типа, после чего комната инстанцируется в игровом мире с учётом заданного расстояния между комнатами.

Перемещение игрока между комнатами осуществляется через систему порталов, а активация следующей комнаты и уничтожение предыдущей происходит динамически по мере продвижения пользователя.

Размещение врагов в каждой комнате определяется на этапе инициализации комнаты с помощью отдельного менеджера. Для каждой точки спавна врагов случайным образом выбирается тип из доступных и количество противников из конфигурации врагов, соответствующей комнаты.

**Система мета-прогрессии.** В игре реализована внутриигровая валюта (фрагменты), которая выпадает при победе над врагами и боссами. Валюта автоматически сохраняется между сессиями через PlayerPrefs. Игрок тратит фрагменты на улучшение эмоций и увеличение максимального числа одновременно активных эмоций. Все улучшения и прогресс эмоций сохраняются между сессиями.

## 2.5 Левел-дизайн

Левел-дизайн игрового приложения построен на сочетании фиксированной хаб-локации и процедурно-генерируемого набора комнат. Центральной точкой выступает пространство «Реальность» – неизменяемая локация, служащая отправной и возвратной точкой между сессиями исследования.

Пространство «Сознание» формируется динамически при каждом переходе в пространство. Алгоритм построения уровня включает обязательную стартовую комнату, одну комнату босса, одну элитную комнату, а также переменное количество базовых комнат.

Связь между комнатами реализована через систему порталов. В стартовой комнате всегда присутствует портал для возврата в «Реальность» и портал для перехода к следующей части уровня. В остальных комнатах порталы обеспечивают продвижение пользователя по маршруту, определённому процедурной генерацией.

Влияние расположения объектов и врагов на геймплей проявляется в необходимости тактического обхода препятствий и принятия решений в условиях ограниченного пространства. Например, группировка врагов-осколков в базовых комнатах может затруднять прямой доступ к порталу, а расстановка противников и препятствий в элитных комнатах требует адаптации стратегии пользователя.

Вариативность левел-дизайна достигается за счёт случайной последовательности комнат, их наполнения и тематического оформления. Такая структура поддерживает интерес пользователя и позволяет реализовать различные сценарии взаимодействия с окружением и противниками.

## 2.6 USP

В разрабатываемом игровом приложении уникальное торговое предложение (USP) формируется за счёт сочетания архитектурных и геймплейных решений, которые непосредственно влияют на глубину, вариативность и индивидуализацию пользовательского опыта.

Ниже приведены ключевые особенности, определяющие уникальность проекта:

1 Модульная процедурная генерация пространства «Сознание», обеспечивающая уникальную структуру уровней и событий в каждом прохождении, что исключает повторяемость маршрутов и сценариев.

2 Система эмоций, позволяющая пользователю формировать индивидуальный набор боевых и защитных модификаторов перед началом каждой игровой сессии, а также развивать их между забегами за счёт внутриигровой валюты;

3 Сохранение прогресса и мета-прогрессия, реализованные через накопление и улучшение эмоций, что обеспечивает долгосрочную мотивацию и постепенное открытие новых стратегических возможностей.

В совокупности эти решения обеспечивают проекту высокую степень реиграбельности, персонализации и стратегической глубины, делая каждую игровую сессию уникальной и значимой для пользователя.

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

**3.1 Разработка графической части**

**3.1.1 Выбор графического пакета**

**Blender** – это бесплатный и открытый программный пакет для создания 3D-графики, широко применяемый в геймдеве для моделирования, скульптинга, текстурирования, анимации, риггинга, визуализации и подготовки игровых ассетов. Blender поддерживает полный цикл 3D-производства, включая экспорт моделей в игровые движки (Unity, Unreal Engine и др.), а также интеграцию с пайплайнами разработки игр.

Преимущества:

– бесплатная лицензия и открытый исходный код;

– поддержка моделирования, скульптинга, UV-развёртки, текстурирования, анимации, риггинга, симуляций, рендеринга и композитинга в одном пакете;

– высокая скорость работы и современный интерфейс;

– прямая интеграция с игровыми движками через экспорт в форматы FBX, OBJ, glTF и другие;

– активное сообщество и регулярные обновления, большое количество бесплатных плагинов и обучающих материалов.

Недостатки:

– быстрые обновления могут приводить к появлению багов и несовместимости плагинов;

– интерфейс и логика работы могут быть непривычны для пользователей других 3D-пакетов.

**Autodesk 3ds Max** –это профессиональное программное обеспечение для 3D-моделирования, анимации и визуализации, широко используемое в игровой индустрии для создания высокополигональных и низкополигональных моделей, а также подготовки ассетов для игровых движков. 3ds Max поддерживает экспорт в форматы, совместимые с Unity, Unreal Engine и другими платформами.

Преимущества:

– высокая точность и развитый инструментарий для полигонального моделирования;

– мощные инструменты для анимации, риггинга и симуляции физических эффектов;

– поддержка большого количества плагинов и интеграция с другими продуктами Autodesk;

– признанный стандарт в крупных студиях.

Минусы:

– высокая стоимость лицензии и необходимость регулярной оплаты;

– для новичков интерфейс может показаться перегруженным;

– медленный запуск и обновления по сравнению с другими решениями.

**Adobe Photoshop** – это профессиональный графический редактор, применяемый в геймдеве для создания и обработки текстур, концепт-артов, спрайтов, UI-элементов и других 2D-ассетов. Photoshop поддерживает работу с многослойными изображениями, масками, фильтрами и пакетной обработкой.

Преимущества:

– широкий набор инструментов для рисования, ретуши, цветокоррекции и работы с текстурами;

– большое количество обучающих материалов и плагинов;

– интеграция с другими продуктами Adobe (например, Illustrator, After Effects);

– стандарт индустрии для 2D-графики и текстурирования.

Недостатки:

– платная подписка, отсутствие бесплатной версии.

**Figma** – это облачный инструмент для проектирования интерфейсов и прототипирования, активно используемый в геймдеве для разработки UI/UX, создания интерактивных прототипов, макетов меню, HUD и других элементов пользовательского интерфейса.

Преимущества:

– поддержка плагинов, библиотек компонентов и автоматизации рутинных задач;

– возможность экспорта ассетов в форматы PNG, SVG и др., пригодные для интеграции в игровые движки;

– совместная работа в реальном времени, что удобно для командной разработки;

– интуитивный интерфейс и быстрый обмен макетами между дизайнерами и программистами.

Недостатки:

– бесплатная версия ограничена по количеству проектов и функций;

– ограниченные возможности для работы с растровой графикой и сложными эффектами по сравнению с Photoshop.

Исходя из анализа современных 3D-графических редакторов, оптимальным выбором был признан Blender. Инструмент выделяется широкой распространённостью в сообществе, бесплатной лицензией, а также обширной базой обучающих материалов и ассетов. Кроме того, Blender предоставляет интегрированные средства для моделирования, анимации и визуализации, что позволяет реализовывать полный цикл создания 3D-контента в рамках одной программной среды.

В результате анализа 2D-графических редакторов наилучшим решением был выбран Adobe Photoshop. Ключевыми факторами стали высокая функциональная насыщенность, включающая продвинутые инструменты для обработки растровой графики, работы с текстом, а также поддержка большого количества плагинов. Дополнительным преимуществом является наличие обширной базы обучающих материалов и активное сообщество пользователей.

**3.1.2 Этапы разработки графической части**

На этапе постановки задач для графической части прототипа были определены все необходимые 3D-ассеты, разделённые по двум основным пространствам – «Реальность» и «Сознание».

Для пространства «Реальность» требовалось создать модель главного персонажа для режима от первого лица, а также окружение жилой комнаты:

– кровать;

– гора одежды;

– таблетки;

– фоторамка;

– личный дневник;

– книга по психологии;

– письмо;

– светильник в виде котика;

– другие предметы комнаты, отражающие повседневность и внутреннее состояние персонажа.

Результат выполненной работы по разработке графической части для пространства «Реальность» представлен на рисунке 3.1.



Рисунок 3.1 – Внешний вид комнаты в пространстве «Реальность»

Для пространства «Сознание» предусматривалась разработка low poly модели главного персонажа для изометрического режима, а также различных типов врагов:

– силуэты людей с насмешливыми улыбками (для школьной комнаты);

– тёмная прозрачная тень, которая выглядит как копия главного героя (для комнаты одиночества);

– враги в виде знаков вопроса (для комнаты неопределённости будущего);

– стеклянные осколки (для базовых комнат);

– видоизменённый NPC-проводник (для босс-комнаты).

Помимо этого, требовались low poly модели NPC-проводника, стражников в виде котиков, а также окружение для различных комнат:

– школьная парта (комната буллинга);

– множество дверей и текстуры с вопросами (комната неопределённости);

– гексагоны для стартовой и босс-комнаты;

– порталы для перехода между комнатами;

– объект для активации выбора эмоций;

– объект для активации области погружения в «Сознание».

Визуальный стиль ассетов разделён по пространствам:

1 Для пространства «Реальности» использовался реализм с детализированными моделями и текстурами, акцентом на атмосферу замкнутости и рутины

2 Для пространства «Сознания» использовался low poly стиль с упрощённой геометрией, минималистичными текстурами.

К техническим требованиям относились:

– оптимизация всех моделей для работы на ПК, экспорт в форматы FBX или OBJ для моделей, PNG для текстур;

– разрешение текстур преимущественно 1024x1024 и 2048x2048;

– поддержка риггинга и анимации для персонажей и врагов;

– обеспечение баланса между качеством визуализации и производительностью прототипа.

Рассмотрим процесс создания моделей на основе low poly женской и мужской фигуры человека. Для ускорения и симметричности работы применялся модификатор Mirror, что позволяло моделировать только одну половину тела, автоматически отражая изменения на другой стороне. На первом этапе создавалась базовая заготовка – половина туловища, начиная с простых примитивов, которые постепенно редактировались в режиме редактирования (Edit Mode) с помощью инструментов Extrude, Move, Scale и Insert Edge Loop. Для придания модели характерных пропорций мужской и женской фигуры корректировались ширина плеч, талии, бедер и груди.

После завершения моделирования применялся модификатор Mirror, чтобы получить окончательную симметричную модель.

Для подготовки модели к текстурированию выполнялась UV-развертка. В режиме редактирования выделялись все полигоны модели, после чего применялся инструмент Smart UV Project, который автоматически создавал развертку для всех элементов. После использовалась функция Pack Islands для оптимального размещения всех островков развертки на UV-карте. Полученная UV-развертка экспортировалась в виде изображения (UV Layout), которое открывалось в Adobe Photoshop. В Photoshop на основе UV-шаблона создавалась текстура, где добавлялись необходимые детали. Готовая текстура импортировалась обратно в Blender и назначалась на материал модели.

На рисунке 3.1 изображены полученные модели с наложенной чистой UV текстурой.

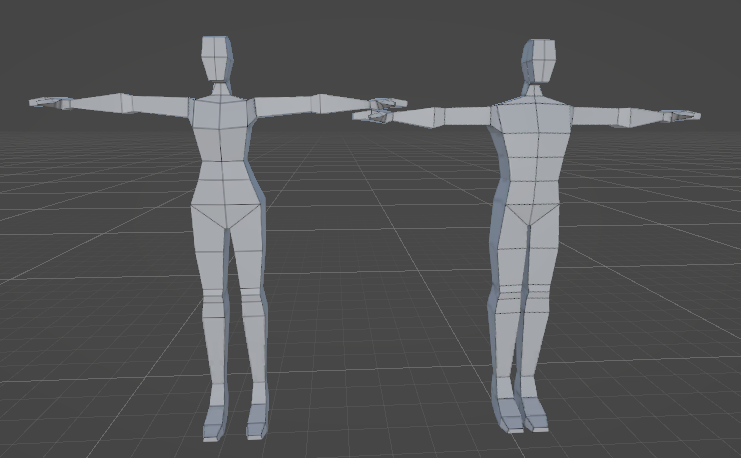


Рисунок 3.1 – Скриншот женской и мужской моделей

Анимация игровых персонажей и врагов осуществлялась в несколько этапов.

Для автоматизации процесса риггинга простых low poly моделей использовалось специализированное программное обеспечение AccuRIG, позволяющее быстро создавать скелетную структуру для 3D моделей.

Модель экспортировалась из Blender в формате FBX и загружалась в AccuRIG, где с помощью автоматических инструментов выставлялись контрольные точки, которые перемещались на основные суставы, выставлялись точки для костей пальцев рук, после чего приложение генерировало полноценный скелет и осуществляло скиннинг.

После риггинга модель загружалась на платформу Mixamo, где осуществлялся подбор подходящих анимаций (бег, атака, idle, получение урона, смерть). На сайте Mixamo можно было просмотреть, как выбранная анимация ложится на модель, и при необходимости скорректировать параметры (например, скорость и амплитуда движений).

Импорт скачанных отдельных анимаций осуществлялся в Unity. Для управления анимациями использовался компонент Animator и система состояний (Animator Controller). В Animator настраивались переходы между различными анимационными состояниями (бег, атака, idle, получение урона, смерть), а также условия их переключения на основе параметров (isRunning, isAttacking, isHit, isDead). Такой подход обеспечил плавное и логичное воспроизведение анимаций в зависимости от состояния врага, управляемого NavMesh Agent.

В процессе создания игрового приложения использовались как самостоятельно созданные модели, так и ассеты, найденные в открытых источниках. При подборе сторонних моделей особое внимание уделялось их соответствию требуемому визуальному стилю для каждого из пространств и количеству полигонов, чтобы выбирать только оптимизированные модели с сохранением детализации.

Скачанные модели проходили дополнительную обработку: выполнялась настройка материалов, создание или корректировка UV-развертки (с использованием функции Pack Islands в Blender), а также нанесение собственных или доработанных текстур в Photoshop на основе экспортированных UV-шаблонов.

Пример доработанного объекта изображен на рисунке 3.2.



Рисунок 3.2 – Пример объекта с измененной текстурой

Такой подход позволил обеспечить стилистическую целостность ассетов и их техническую совместимость с игровым движком.

В процессе разработки графической части особое внимание уделялось настройке освещения и материалов для усиления атмосферы и визуального восприятия игровых объектов.

Для имитации локального источника света в пространстве «Реальность» использовался объект Point Light, размещённый внутри модели светильника. Параметры источника света (интенсивность, радиус, цвет) подбирались таким образом, чтобы создать мягкое, рассеянное освещение, соответствующее приглушённой и уютной атмосфере комнаты.

Результат всех настроек изображен на рисунке 3.3.



Рисунок 3.3 – Светильник в комнате главной героини

Для усиления эффекта свечения самого светильника применялась настройка материала с использованием параметра Emission. В инспекторе у материала для соответствующего объекта активировался канал Emission, задавался цвет и интенсивность свечения, а также использовалась отдельная эмиссионная текстура. Это позволяло добиться эффекта, при котором светильник выглядел как самостоятельный источник света даже при выключенном глобальном освещении.

Для создания визуального эффекта стеклянных осколков в пространстве «Сознание» в Unity был настроен специальный материал с прозрачностью. В качестве типа материала был выбран стандартный шейдер.

В настройках материала в параметре Surface Type устанавливалось значение Transparent, что позволяло объекту корректно отображать прозрачные области текстуры.

Для управления степенью прозрачности использовался режим смешивания (Blending Mode) Alpha. Для корректного отображения осколков с обеих сторон в параметре Render Face выбиралось значение Both. Это позволяло рендерить полигоны материала как с лицевой, так и с обратной стороны, что особенно важно для тонких или плоских объектов, таких как стеклянные осколки.

**3.2 Разработка программной части**

**3.2.1 Выбор игрового движка**

**Unity 3D** – это кроссплатформенный игровой движок, предназначенный для разработки 2D и 3D приложений, широко используемый в индустрии для создания игр, интерактивных симуляций и визуализаций. Движок поддерживает множество платформ, включая Windows, macOS, Android, iOS, WebGL и другие. Основной язык программирования – C#.

Преимущества:

– простота освоения и интуитивно понятный интерфейс редактора;

– большое количество обучающих материалов и активное сообщество;

– поддержка широкого спектра платформ (PC, мобильные устройства, VR/AR, WebGL);

– гибкая система компонентов и удобная интеграция пользовательских скриптов на C#;

– большой ассортимент ассетов и плагинов в Unity Asset Store;

– быстрая итерация и предпросмотр изменений в редакторе;

Недостатки:

– ограниченные возможности по работе с высокополигональной графикой и фотореализмом по сравнению с Unreal Engine;

– менее продвинутые инструменты визуальных эффектов и освещения «из коробки»;

– ограничения бесплатной версии по функционалу и лицензированию для коммерческих проектов.

**Unreal Engine 5** – это мощный игровой движок, разработанный компанией Epic Games, ориентированный на создание высококачественных 3D-игр и интерактивных приложений с фотореалистичной графикой. Основной язык программирования – C++ и визуальный скриптовый язык Blueprints. Движок активно используется в AAA-разработке, архитектурной визуализации и киноиндустрии.

– передовые технологии рендеринга и освещения, поддержка фотореалистичной графики;

– мощная система визуального программирования Blueprints;

– бесплатное использование всех функций движка до достижения определённого дохода;

– встроенные инструменты для создания сложных анимаций, VFX и пост-обработки;

– поддержка современных технологий (Ray Tracing, Nanite, Lumen и другие).

Недостатки:

– более высокая сложность освоения для начинающих разработчиков;

– требовательность к аппаратным ресурсам, как для разработки, так и для конечных пользователей;

– больший размер финальных сборок и более длительное время компиляции;

– меньше готовых ассетов и плагинов для простых 2D/мобильных проектов по сравнению с Unity.

После рассмотрения сильных и слабых сторон каждого движка наилучшим вариантом для создания игрового приложения определен движок Unity, за его гибкость, доступность, графическую составляющую и бесплатную лицензию для начинающих. Также репутация этого движка и его ориентированность на похожие проекты открывают большой потенциал для разработки.

Для реализации дипломного проекта выбран игровой движок Unity, поскольку он обеспечивает оптимальное соотношение простоты освоения, гибкости и поддержки множества платформ. Unity позволяет быстро прототипировать и реализовывать игровые механики, обладает развитой экосистемой ассетов и плагинов, а также предоставляет удобные инструменты для работы с 2D и 3D графикой. Несмотря на более ограниченные возможности по созданию фотореалистичной графики по сравнению с Unreal Engine, Unity является более подходящим выбором для небольших и средних проектов, а также для индивидуальной разработки.

**3.2.2 Этапы разработки программной части**

На начальном этапе разработки программной части проекта были сформулированы основные функциональные требования на основе результатов системного проектирования.

В качестве ключевых задач были определены: реализация двух режимов управления (от первого лица в «Реальности» и изометрического в «Сознании»), поддержка процедурной генерации комнат, создание системы эмоций, боевой системы, диалоговой системы и системы метапрогрессии.

**GameManager** – центральный управляющий компонент проекта, реализующий паттерн Singleton (статическое свойство Instance). Он отвечает за глобальное управление состояниями игры, переходами между пространствами «Реальность» и «Сознание», а также за инициализацию и координацию других менеджеров.

GameManager осуществляет асинхронную загрузку и выгрузку сцен с помощью корутин, управляет плавными переходами через компонент ScreenFader, а также обеспечивает автосохранение и восстановление состояния игрока и ключевых объектов с использованием PlayerPrefs.

Взаимодействие с валютой реализовано через инициализацию и вызовы методов CurrencyManager. Для UI предусмотрена динамическая активация и настройка панелей в зависимости от текущего состояния (UpdateUIState(), SetupCurrencyUI()).

GameManager также реализует обработку смерти игрока (HandlePlayerDeath()), запуск новой игры (StartNewGame()), продолжение сохранённой сессии (ContinueGame()) и выход из приложения (ExitGame()).

GameManager обеспечивает целостность игрового процесса, централизованное управление состояниями и координацию между основными системами проекта.

В проекте реализованы два отдельных контроллера для управления персонажем в разных игровых режимах.

**ConsciousnessController** отвечает за поведение игрока в пространстве «Сознание» (изометрический режим). В этом скрипте реализованы методы обработки ввода для перемещения (HandleMovement()), рывка (HandleDash()), взаимодействия с объектами (HandleInteraction()), а также управления камерой и выбора эмоций. Контроллер интегрирован с системой эмоций через взаимодействие с компонентом PlayerEmotionHandler, что позволяет динамически изменять характеристики персонажа в зависимости от выбранных эмоций.

**RealityPlayerController** реализует управление персонажем в пространстве «Реальность» (режим от первого лица). В этом контроллере обрабатывается базовое перемещение с помощью клавиш WASD, вращение камеры мышью, а также взаимодействие с объектами окружения через методы типа TryInteract().

Оба контроллера поддерживают централизованную блокировку управления через менеджер PlayerControlManager, что позволяет временно отключать ввод игрока, например, при открытии меню или в режиме осмотра объектов.

Архитектура управления персонажем построена на разделении логики по игровым режимам, что обеспечивает гибкость и расширяемость системы, а также позволяет реализовать уникальные механики для каждого пространства.

**DungeonGenerator** – ключевая система процедурной генерации уровней в пространстве «Сознание». Она реализована в виде отдельного компонента, который отвечает за создание структуры подземелья, размещение комнат, врагов и порталов при каждом запуске сессии.

Основная логика генерации запускается методом Start() и асинхронно выполняется через корутину GenerateDungeonAsync(). Для конфигурирования параметров генерации используется ScriptableObject-конфиг (DungeonConfig), в котором задаются типы комнат, количество, параметры связей, а также ссылки на префабы комнат и врагов.

В процессе генерации вызывается метод ValidateConfiguration(), который проверяет корректность настроек. Далее формируется граф комнат с помощью структуры данных RoomNode, где для каждой комнаты определяется тип (стартовая, базовая, элитная, босс-комната) и размещается соответствующий префаб.

Связи между комнатами реализуются через систему порталов (InteractablePortal), которые автоматически инстанцируются и настраиваются для перехода между локациями. Размещение врагов и объектов в комнатах происходит на этапе инициализации комнаты с помощью менеджера RoomManager и соответствующих ScriptableObject-конфигураций врагов.

DungeonGenerator обеспечивает гарантированное наличие всех ключевых типов комнат, а также вариативность структуры уровня за счёт случайного выбора и последовательного соединения комнат из пула, определённого в ScriptableObject.

Система процедурной генерации построена на модульной архитектуре с активным использованием ScriptableObject для параметризации, что обеспечивает гибкость, расширяемость и повторяемость генерации уровней без необходимости изменения кода.

**EmotionSystem** реализует механику выбора, активации и улучшения эмоций, которые напрямую влияют на характеристики и боевые возможности игрока в пространстве «Сознание». Система построена на компоненте EmotionSystem, который содержит список доступных эмоций (enum EmotionType), их уровни и состояния активности. Для хранения и обработки данных об эмоциях используется внутренняя структура Emotion, включающая поля типа, уровня и статуса активности.

Основные методы системы:

1 ActivateEmotion(EmotionType type) – активирует выбранную эмоцию, если не превышен лимит одновременных активных эмоций.

2 UpgradeEmotion(EmotionType type) – повышает уровень эмоции за счёт игровой валюты, интеграция с CurrencyManager.

3 ApplyEmotionEffects() – применяет эффекты активных эмоций к характеристикам игрока через компонент PlayerEmotionHandler.

4 ModifyIncomingDamage(float damage) и аналогичные методы — модифицируют параметры урона, здоровья, скорости и других характеристик в зависимости от активных эмоций.

Система поддерживает интеграцию с UI через компоненты EmotionUI и EmotionButtonUI, обеспечивая отображение текущих эмоций, их эффектов и уровней. Для параметризации эффектов и стоимости улучшений используются публичные поля и словари внутри скрипта, а также взаимодействие с ScriptableObject-конфигами эмоций (если требуется расширение набора эмоций или их свойств).

Сброс активных эмоций происходит при смерти персонажа или переходе между мирами, что реализовано через вызовы соответствующих методов в жизненном цикле игрока. Все изменения состояния эмоций автоматически отражаются в UI и влияют на игровой процесс в реальном времени.

EmotionSystem обеспечивает гибкую и расширяемую механику модификации геймплея через выбор и развитие эмоций, тесно интегрированную с боевой системой, UI и менеджером валюты.

**CurrencyManager** реализует систему игровой валюты (фрагментов), которые игрок получает за победу над врагами в пространстве «Сознание» и тратит на улучшение эмоций. Компонент построен по паттерну Singleton и обеспечивает централизованное управление валютой во всех сценах.

Текущее количество валюты хранится в приватном поле currentCurrency и синхронизируется с PlayerPrefs для сохранения прогресса между сессиями.

Основные методы:

1 AddCurrency(int amount) – добавляет валюту игроку, обновляет PlayerPrefs и вызывает событие OnCurrencyChanged.

2 SpendCurrency(int amount) – списывает валюту, если её достаточно, и возвращает результат операции.

3 GetCurrentCurrency() – возвращает текущее количество валюты.

CurrencyManager интегрирован с UI через событие OnCurrencyChanged, на которое подписывается компонент CurrencyUI для обновления отображения валюты на экране.

Инициализация CurrencyManager происходит автоматически при запуске игры через вызов из GameManager (InitializeCurrencyManager()), что гарантирует его присутствие в любой игровой сцене.

В проекте реализована модульная система интерактивных объектов, обеспечивающая взаимодействие игрока с элементами окружения, NPC, панелей эмоций и порталами. Базовым элементом системы является абстрактный класс BaseInteractable, который реализует интерфейс IInteractable и содержит общую логику для всех интерактивных объектов.

В классе BaseInteractable определены основные параметры:

1 interactionRadius – радиус, в пределах которого игрок может взаимодействовать с объектом.

2 isInteractable – флаг доступности взаимодействия.

3 onInteractEvent – UnityEvent, вызываемый при успешном взаимодействии.

Ключевые методы:

1 OnInteract() – основной метод, вызываемый при взаимодействии игрока с объектом. По умолчанию вызывает событие onInteractEvent, может быть переопределён в наследниках.

2 OnPlayerEnter() и OnPlayerExit() – методы для обработки входа и выхода игрока из зоны взаимодействия, используются для отображения UI-подсказок или активации визуальных эффектов.

Для конкретных типов интерактивных объектов реализованы наследники:

1 InteractableObject представляет собой базовую реализацию для простых объектов.

2 EmotionPanelInteractable отвечает за открытие панели выбора эмоций (emotionPanel) и взаимодействует с системой эмоций (EmotionSystem).

3 InteractablePortal реализует переход между комнатами или мирами, а также активирует загрузку новой сцены или перемещение игрока.

Взаимодействие с объектами инициируется из контроллеров игрока (ConsciousnessController, RealityPlayerController) через проверку наличия компонента, реализующего интерфейс IInteractable, и вызов метода OnInteract() при нажатии соответствующей клавиши.

Система поддерживает расширяемость за счёт наследования от BaseInteractable и реализации интерфейса IInteractable, что позволяет легко добавлять новые типы интерактивных объектов с уникальным поведением, не нарушая архитектуру проекта.

Система **ObjectInspector** реализует функциональность осмотра объектов в отдельном режиме. Основной компонент – скрипт ObjectInspector, который управляет процессом осмотра. При активации осмотра вызывается метод, который переключает камеру на специальную (inspectionCamera), отображает UI-окно с информацией об объекте (inspectionUI.ShowWindow(obj)) и блокирует управление игроком через вызов PlayerControlManager.Instance.SetControlsEnabled(false).

В процессе осмотра игрок может вращать объект и использовать зум. После завершения осмотра вызывается метод StopInspection(), который возвращает управление основной камерой, скрывает UI-окно (inspectionUI.HideWindow()) и вновь включает управление игроком.

Система интегрирована с компонентами UI и поддерживает взаимодействие с различными типами инспектируемых объектов, реализуя удобный и наглядный режим детального изучения предметов в игре.

В проекте реализована система монологов и диалогов, обеспечивающая взаимодействие игрока с NPC и окружением. Основные компоненты – скрипты MonologueManager, MonologueTrigger, NPCDialogueInteract и интеграция с внешней диалоговой системой VIDE.

**MonologueManager** управляет показом монологов на экране. При активации триггера (MonologueTrigger) вызывается метод, который отображает текст монолога в UI и блокирует управление игроком на время воспроизведения. После завершения монолога управление возвращается игроку.

Диалоговая система с NPC реализована через компонент **NPCDialogueInteract**. При входе в зону взаимодействия и нажатии клавиши вызывается метод, инициирующий диалог с использованием системы VIDE. Диалоговые данные подключаются через компонент VIDE\_Assign, что позволяет использовать заранее подготовленные диалоговые ветки и реплики. В процессе диалога игрок может выбирать варианты ответов, а система отслеживает прогресс и обновляет доступные диалоги в зависимости от состояния игры.

Система поддерживает запуск особых сценариев (например, мгновенный возврат в «Реальность» или изменение условий появления врагов) через обработку выбранных реплик и событий в диалоге.

В проекте также реализована модульная система пользовательского интерфейса, обеспечивающая отображение и управление всеми основными UI-элементами игры. Управление отдельными частями интерфейса осуществляется через специализированные компоненты, каждый из которых отвечает за свою область.

Для отображения и управления эмоциями используется компонент EmotionUI, который синхронизируется с системой эмоций (EmotionSystem) и отображает текущие активные эмоции, их уровни и эффекты. Кнопки выбора и активации эмоций реализованы через компонент EmotionButtonUI, который обрабатывает пользовательский ввод и визуальную обратную связь.

Всплывающие подсказки и дополнительная информация при наведении на эмоцию в системе эмоций реализованы через компоненты ToolTip и ToolTipManager, которые обеспечивают отображение информации о предметах, эмоциях и других элементах интерфейса.

Интерфейс валюты реализован через компонент CurrencyUI, который подписывается на событие изменения валюты в CurrencyManager и обновляет отображение количества фрагментов на экране. Создание и настройка панели валюты может происходить динамически через вспомогательный класс CurrencyPrefab.

Для отображения подсказок при взаимодействии с объектами используется компонент InteractionPromptUI, который активируется при входе игрока в зону взаимодействия с интерактивным объектом.

Меню паузы реализовано через компонент PauseMenuUI, который управляет состоянием панели паузы, обработкой событий (продолжить игру, выйти в меню, сохранить прогресс) и блокировкой управления игроком во время паузы.

Таким образом, система UI построена на принципах разделения ответственности и событийной архитектуры, что обеспечивает гибкость, расширяемость и удобство взаимодействия пользователя с игровым процессом.

**4 UI/UX ПРИЛОЖЕНИЯ**

В проекте выбран минималистичный стиль UI/UX, ориентированный на простоту, лаконичность и отсутствие избыточных визуальных элементов. Интерфейс не должен отвлекать пользователя от игрового процесса, а дополнять стиль игрового приложения, обеспечивая при этом быструю доступность информации и интуитивное взаимодействие.

Все элементы UI выполнены в единой цветовой гамме, с акцентом на читаемость и функциональность, что соответствует современным принципам проектирования пользовательских интерфейсов для игр.

Главное меню игрового приложения, представленное на рисунке 4.1, реализовано в стиле минимализма, что соответствует общей концепции UI/UX проекта.

Рисунок 4.1 – Главное меню игры

На рисунке 4.1 видно, что меню состоит из заголовка и четырёх кнопок, расположенных по центру экрана. Такой подход обеспечивает максимальную читаемость и простоту навигации для пользователя.

В Unity главное меню выполнено с использованием Canvas, настроенного в режиме Screen Space – Overlay для корректного отображения на любом разрешении экрана. В иерархии Canvas размещён объект MainMenuPanel, содержащий текстовый элемент заголовка игры, группы кнопок («Проодлжить», «Новая игра», «Настройки», «Выйти»), каждая из которых отвечает за определенную функцию в игровом приложении.

При нажатии на клавишу «ESC» в игровом пространстве открывается панель паузы с кнопками: «Продолжить», «Сохранить», «Главное меню». Панель паузы содержит: фон (image), кнопки управления, текстовые элементы на них.

Внешний вид окна паузы представлен на рисунке 4.2.

Рисунок 4.2 – Панель паузы

Панель валюты распологается в левом нижнем углу экрана и состоит из: image (фон) на котором отображается валюта, иконка валюты и текст для отображения текущего количества фрагментов.

Полоска здоровья игрока реализована в нижнем левом углу экрана с помощью отдельного объекта HealthBarUI, который содержит панель, на которой распологается image для визуализации заполнения и уменьшения полосы и еще один элемент image для цвета фона.

Внешний вид полоски здоровья игрока и фрагментов игрока представлены на рисунке 4.3.

Рисунок 4.3 – Полоска здоровья игрока и количество фрагментов

Панель эмоций в Unity реализована с использованием отдельного Canvas, настроенного в режиме Screen Space – Overlay для обеспечения корректного отображения на различных разрешениях экрана.

В иерархии Canvas размещён объект EmotionPanel, который содержит визуальные элементы для управления и отображения состояния эмоций игрока. Внутри EmotionPanel располагаются отдельные кнопки для каждой эмоции и текстовое поле (TextMeshProUGUI) для отображения уровня игрока.

Рисунок 4.4 – Панель эмоций

Для каждой эмоции реализована всплывающая подсказка (ToolTip), отображающая описание эффекта при наведении курсора на эмоцию.

ToolTip используется для отображения дополнительной информации о эмоции (ее эффектах и текущем уровне). Всплывающая подсказка состоит из панели и текстового поля с описанием.

Рисунок 4.5 – Всплывающая подсказка (ToolTip)

Панель монолога состоит из фона панели (image), обеспечивающей читаемость текста на любом фоне и визуально отделяющая монолог от остального интерфейса и текстового поля для отображения текста монолога.

Рисунок 4.6 – Панель монолога

Внутри панели диалога располагаются следующие основные элементы: текстовое поле для вывода реплик, текстовое поле для имени NPC, отдельный контейнер для реплик игрока, включающий текстовое поле для имени игрока и список кнопок для выбора вариантов ответа в диалоге. Текст располагается на фоне специальной панели (image), обеспечивающей читаемость текста на любом фоне.

Рисунок 4.7 – Панель диалога

Окно осмотра предмета состоит из панели, содержащей несколько текстовых элементов: поле для названия предмета, поле для описания предмета, а также текстовая подсказка по управлению в окне осмотра в левом нижнем углу экрана.

Рисунок 4.8 – Окно осмотра предмета

В результате проведённой работы по проектированию и реализации пользовательского интерфейса и пользовательского опыта (UI/UX) игрового приложения удалось создать интуитивно понятную, минималистичную и функциональную систему взаимодействия с пользователем.

Все элементы интерфейса – главное меню, HUD, панели эмоций, диалоговые и монологовые окна, а также вспомогательные подсказки выполнены в едином стиле, что обеспечивает целостность визуального восприятия и не отвлекает от основного игрового процесса.

Структура UI построена с учётом адаптивности к различным разрешениям экрана и удобства навигации.

**5 ТЕСТИРОВАНИЕ**

Для проверки корректности функционирования всех ключевых компонентов использовалось ручное тестирование, при котором выполнялись различные сценарии использования приложения и фиксировались возможные возникающие ошибки, проверяя тем самым соответствие поведения игрового приложения заявленным требованиям.

Ручное тестирование включало последовательную проверку всех пользовательских сценариев: запуск и завершение игры, переходы между сценами, взаимодействие с объектами, использование эмоций, сражения с врагами, работу систем монологов и диалогов, а также корректность отображения и обновления элементов интерфейса.

Для каждого сценария составлялись отдельные тест-кейсы, которые позволяли систематизировать процесс проверки и обеспечить полноту охвата функционала.

В таблице 5.1 приведен набор случаев, использовавшихся при тестировании компьютерного игрового приложения.

Таблица 5.1 – Тест-кейсы проверки игрового приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Название | Предусловие | Промежуточные шаги | Ожидаемый результат |
| QA –1 | Запуск новой игры | Приложение установлено | 1 Открыть игровое приложение  2 Нажать кнопку «Новая игра» в главном меню | Загружается стартовая сцена «Реальность», управление персонажем активно, интерфейс отображается корректно |
| QA –2 | Переход между мирами | Игрок находится в пространстве «Реальность»,Объект для перехода активен | 1 Войти в область действия объекта перехода  2 Нажать клавишу R | Происходит плавный переход в пространство «Сознание», загружается соответствующая сцена, интерфейс обновляется |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Название | Предусловие | Промежуточные шаги | Ожидаемый результат |
| QA –3 | Взаимодействие с предметом | Игрок находится рядом с интерактивным объектом | 1 Находиться в пределах области взаимодействия объекта  2 Навести курсор на объект  3 Нажать клавишу «F» | Внизу экрана появляются реплики игрока (система монологов) |
| QA –4 | Открытие панели эмоций | Игрок находится в стартовой комнате пространства «Сознание» | 1 Находиться в пределах области взаимодействия объекта  2 Нажать клавишу «F» | Открывается панель эмоций, отображаются доступные эмоции, их уровни и эффекты, управление персонажем блокируется |
| QA –5 | Улучшение эмоции | Открыта панель эмоций, у игрока достаточно валюты | 1 Удерживать кнопку выбора эмоции (ЛКМ) до визуального подсвечивания | Происходит визуальный эффект, уровень выбранной эмоции увеличивается, количество валюты уменьшается, отображается обновлённая информация |
| QA –6 | Открытие панели паузы | Игрок находится в любом игровом пространстве | 1 Нажать клавишу ESC | Панель паузы появляется, управление блокируется |

Важным инструментом при ручном тестировании в Unity выступал механизм вывода сообщений в консоль с помощью метода Debug.Log. В проекте Debug.Log активно использовался для отслеживания хода выполнения кода, фиксации значимых событий, проверки значений переменных и параметров на различных этапах работы приложения.

В каждом ключевом скрипте были добавлены сообщения Debug.Log, информирующие о запуске методов, изменении состояния объектов, срабатывании событий и обработке пользовательского ввода. Это позволяло оперативно выявлять ошибки, некорректные переходы между состояниями, а также отслеживать логику работы сложных систем (например, процедурной генерации, системы эмоций).

Использование Debug.Log существенно ускорило процесс поиска и устранения багов, позволило быстро локализовать проблемные участки кода и подтвердить корректность работы отдельных функций без необходимости постоянного использования внешних средств отладки.

При тестировании механик игрового прототипа выявлены некоторые ошибки и недочеты:

– в некоторых случаях управление передвижением игрока в пространстве «Реальность» блокировалось;

– камера осмотра предмета некорректно переключала управление, в результате чего происходило одновременное влияние на основную камеру игрока;

– система диалогов с NPC позволяла повторно запускать диалог даже при уже открытом окне диалога.

Все выявленные в ходе тестирования ошибки были успешно устранены.

В результате проведённой работы над ошибками стабильность и качество игрового прототипа были выведены на новый уровень, что позволило обеспечить максимально комфортный и безошибочный пользовательский опыт.