

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Высшей математики, информатики, экономики и естествознания
кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

 Л.Н. Храмова
подпись инициалы, фамилия

« 13 » 06 2023 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

код-наименование направления

РАЗРАБОТКА ИГРЫ НА ПЛАТФОРМЕ UNITY В ЖАНРЕ
ГОЛОВОЛОМКА

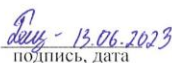
Руководитель


подпись, дата

доцент, канд. пед. наук
должность, ученая степень

Е. В. Киргизова
инициалы, фамилия

Выпускник


подпись, дата

Р. В. Бацeko
инициалы, фамилия

Нормоконтролер


подпись, дата

Е. В. Киргизова
инициалы, фамилия

Лесосибирск 2023

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка игры на платформе Unity в жанре головоломка» содержит 70 страниц текстового документа, 42 использованных источника, 4 таблицы, 27 иллюстраций.

КОМПЬЮТЕРНАЯ ИГРА, ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, UNITY, ПРОГРАММА, ЖАНР, ГОЛОВЛОМКА.

Компьютерные игры причисляются к актуальным задачам современной науки, так как представляют собой исключительный продукт развития техники и прогрессивного индивида. Игры помогают смоделировать всевозможные жизненные ситуации, трудности и предлагают вероятные пути их решения.

Цель исследования – теоретически обосновать и разработать компьютерную игру в жанре головоломка на платформе Unity.

Объект исследования – процесс разработки компьютерных игр.

Предмет исследования – процесс разработки компьютерной игры в жанре головоломка на платформе Unity.

Основные задачи исследования:

– на основе анализа учебной и научно технической литературы по теме исследования определить понятийно категориальный аппарат, сущность жанра головоломка;

– рассмотреть этапы разработки и планирования компьютерной игры в жанре головоломка;

– разработать и протестировать компьютерную игру в жанре головоломка.

В ходе выполнения выпускной квалификационной работы разработана и протестирована компьютерная игра в жанре головоломка на платформе Unity.

СОДЕРЖАНИЕ

Введение.....	4
1 Теоретические аспекты создания компьютерной игры	7
1.1 Сущность понятия «компьютерная игра». Классификация компьютерных игр	7
1.1.1 Развитие жанра «головоломка» в компьютерных играх	10
1.2 Сравнительный анализ и выбор инструментальных средств разработки компьютерных игр	12
1.3 Общий алгоритм реализации проекта	18
2 Проектирование и реализация компьютерной игры в жанре головоломка на платформе Unity	20
2.1 Планирование разработки проекта.....	20
2.2 Реализация компьютерной игры в жанре головоломка	23
2.2.1 Установка программного обеспечения	23
2.2.2 Разработка графического оформления.....	25
2.2.2 Создание пустого игрового проекта.....	27
2.2.4 Создание анимаций персонажа.....	33
2.2.5 Создание игрового уровня.....	36
2.2.6 Создание диалоговой системы.....	39
2.2.7 Создание сцены главного меню.....	40
2.3 Функциональное тестирование.....	41
Список использованных источников	45
Приложение А Листинг C# кода, отвечающего за работу персонажа	49
Приложение Б Листинг C# кода, отвечающего за взаимодействие игрового мира с персонажем	55
Приложение Г Листинг XML кода, отвечающего содержанию в диалоговой системе	67
Приложение Д Руководство пользователя	70

ВВЕДЕНИЕ

Одним из важных аспектов информатизации общества и образования является применение и создание информационных технологий, включая компьютерные игры. В настоящее время компьютерные игры имеют огромную популярность и распространены по всему миру. Каждый год разрабатываются новые технологии, которые открывают новые возможности в области компьютерных игр.

Компьютерные игры причисляются к актуальным задачам современной науки, так как представляют собой исключительный продукт развития техники и прогрессивного индивида. Игры помогают смоделировать всевозможные жизненные ситуации, трудности и предлагают вероятные пути их решения. Игра заключает внутри себя все необходимые основы для естественного формирования индивида и культуры общества.

С развитием компьютеров улучшаются и компьютерные игры, привлекая все больше людей. Сегодня компьютерная техника достигла такого уровня эволюции, что позволяет разработчикам создавать игры с реалистичной графикой и звуковым оформлением.

Головоломки могут помочь развивать навыки, такие как логическое мышление, пространственное мышление, решение проблем, память и внимание. Создание игры в жанре головоломка может быть ориентировано на развитие этих навыков у игроков и помочь им улучшить свои умственные способности.

Для достижения поставленной цели необходимо решить следующие задачи:

– на основе анализа учебной и научно технической литературы по теме исследования определить понятийно категориальный аппарат, сущность жанра головоломка;

– рассмотреть этапы разработки и планирования компьютерной игры в жанре головоломка;

– разработать и протестировать компьютерную игру в жанре головоломка.

Методы исследования:

– теоретические: анализ учебной и научно технической литературы по теме исследования; обобщение и сравнительный анализ;

– эмпирические: беседа; моделирование; тестирование программного продукта.

Практическая значимость исследования заключается в том, что разработанная компьютерная игра в жанре головоломка на платформе Unity может быть использована педагогами для проведения внеклассных мероприятий, материалы исследования могут быть использованы студентам при написании статей, рефератов, курсовых и выпускных квалификационных работ.

Результаты исследований представлены на следующих научных мероприятиях:

1. VI Всероссийская научно-практическая конференция преподавателей, учителей, студентов и молодых ученых «Актуальные проблемы преподавателей дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 7-12 ноября 2022 г., диплом 2 степени).

2. Всероссийский молодёжный научный форум «Современное педагогическое образование: теоретические и прикладные аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 11 апреля 2023 г., диплом 3 степени).

3. Международная научно-практическая конференция «Исследования в современной науке» (Краснодар, КЦНТИ – филиал ФГБУ 30 марта 2023 г.)

По результатам исследования опубликованы статьи:

1. Бацеко, Р.В. Платформа Unity как средство разработки игр в жанре головоломка / Р.В. Бацеко // Исследования в современной науке: сборник

научных статей международной научно-практической конференции. – Краснодар, 2023 – 30 стр.

Структура работы – работа состоит из введения, двух глав, заключения, приложения и списка использованных источников, включающего 42 источника. Результаты работы представлены в 4 таблицах, 27 рисунках. Общий объем работы – 70 печатных листов.

1 Теоретические аспекты создания компьютерной игры

1.1 Сущность понятия «компьютерная игра». Классификация компьютерных игр

История игровой индустрии простирается на протяжении более полувека, начиная с момента её зарождения. Изначально, игры появились не с целью предоставить развлечение, а в качестве научного эксперимента. Это объясняется тем, что ранние электронно-вычислительные машины (ЭВМ) были огромными, дорогостоящими и в основном использовались в образовательных и научных учреждениях [24]. В 1961 году – программисты Массачусетского Технологического Института (MIT) на своих суперкомпьютерах создали игру под названием «SpaceWar». В 2007 году «SpaceWar» вошла в десятку самых важных компьютерных игр всех времён, положивших начало создания игровой индустрии. С момента выхода этой игры, игровая индустрия активно развивалась и прошла много стадий своего развития.

В настоящее время компьютерные игры стали неотъемлемой частью современной культуры и развлечений. Они привлекают миллионы людей по всему миру и предоставляют уникальную возможность погрузиться в виртуальную реальность. Представляют собой источник не только развлечения, но и образования, способствуя развитию навыков, таких как логическое мышление, реакция, сотрудничество и принятие решений в условиях быстрого прогресса.

Компьютерные игры – это вид интерактивных развлечений, созданных с помощью компьютерной технологии. Они представляют собой программы, которые выполняются на компьютере или другом устройстве и позволяют пользователю взаимодействовать с виртуальным миром, созданным внутри игры [23].

Компьютерные игры представлены множеством жанров, каждый из которых обладает уникальными особенностями. Жанры компьютерных игр –

это категории, которые отличаются основной механикой и характеристикой. Жанры помогают организовать игры в группы схожих по стилю и игровому опыту игроков, что позволяет лучше ориентироваться в мире компьютерных игр и выбирать игры, в соответствии с интересами.

Компьютерные игры классифицируют по нескольким основным признакам:

- жанр;
- количество игроков;
- визуальное представление;
- платформа.

Жанр игры определяется целью и основной механикой игры. Рассмотрим наиболее популярные жанры, представленные в таблице 1.

Представленные в таблице 1 жанры нельзя назвать полными, так как в последнее время стали появляться игры собственных жанров, которые можно отнести как к одному из представленных жанров, так и к самостоятельному.

В зависимости от количества игроков существуют два типа игр:

- Игры для одного игрока
- Многопользовательские игры

Компьютерные игры могут быть классифицированы по визуальному представлению следующим образом [26]:

- Текстовые игры: минимальная графика, взаимодействие с игроком осуществляется с помощью текста.
- 2D игры: все элементы отображаются в двухмерной графике (спрайты).
- 3D игры: все элементы представлены в трехмерной графике (3D-модели).

По типу платформы игры также могут быть классифицированы:

- Персональные компьютеры
- Игровые приставки/консоли
- Мобильные телефоны

Таблица 1 – Жанры компьютерных игры

Жанр	Описание	Основные поджанры	Примеры
Шутеры (Shooter)	Игры, в которых основной акцент делается на боевых действиях и стрельбе	<ul style="list-style-type: none"> – От 1-го лица – От 3-го лица 	
Головоломки (Puzzle)	Игры, в которых игроки решают различные задачи и головоломки, чтобы продвигаться дальше в игре	<ul style="list-style-type: none"> – Загадки – Логические – Исследование 	
Стратегии (Strategy)	Игры, в которых игроки управляют своими войсками или другими ресурсами, строят базы и развивают экономику	<ul style="list-style-type: none"> – Пошаговая – Реального времени – Глобальная – Экономические 	
Симуляторы (Simulation)	Игры, имитирующие различные элементы реальной жизни, например, авиасимуляторы, симуляторы управления городом или управления железной дорогой	<ul style="list-style-type: none"> – Технические – Симуляторы жизни 	
Ролевые игры (RPG)	Игры, в которых игрок управляет персонажем, развивает его навыки и способности, выполняет задания и сражается с врагами	<ul style="list-style-type: none"> – Тактическая – Экшен RPG – Японская 	
Приключения (Adventure)	Игры, в которых игрок управляет персонажем, исследует мир, решает головоломки и выполняет задания в рамках своей истории	<ul style="list-style-type: none"> – Квест – Визуальная новелла 	

Представленная классификация не является полной. Например, можно добавить музыкальные игры в качестве специфического жанра, где акцент сделан на музыке и звуках, а многопользовательские игры можно разделить на несколько подкатегорий. Однако стоит отметить, что эта классификация достаточна для определения большинства существующих игр, поскольку на данный момент не существует однозначной и полной классификации компьютерных игр. Это объясняется тем, что многие игры не относятся к конкретным критериям. Например, игра может сочетать несколько жанров,

выпускаться на разных платформах или иметь как однопользовательский, так и многопользовательский режимы. Это обуславливается тем, что игровая индустрия появилась относительно недавно и отличается от других сфер развлечений.

1.1.1 Развитие жанра головоломка в компьютерных играх

Головоломки имеют древнее происхождение, встречаясь на стенах египетских пирамид, в древнегреческих манускриптах и других исторических памятниках [9].

Однако их наибольшая популярность пришла в IX веке, когда образование стало доступным для большего числа людей. В этот период было выпущено первое собрание головоломок в Европе под названием «Задачи для развития молодого ума» [4], созданное ирландским просветителем Алкуином.

В конце XIX и начале XX веков американец Сэм Лойд и англичанин Генри Дьюдени сделали значительный вклад в распространение головоломок, публикуя их в периодических изданиях и привлекая широкую аудиторию.

Затем, в 1974 году, Эрнэ Рубик изобрел кубик Рубика, который помогал развивать умение решать сложные задачи.

Когда ЭВМ начали активно развиваться, начали появляться первые видеоигры, в 1980-х годах одной из первых игр этого жанра стала Sokoban, созданная Хироюки Имабаяси. В этой игре игрок должен был передвигать ящики по лабиринту, чтобы поставить их на определенные места. Но самой известной компьютерной головоломкой стал Тетрис, появившийся в 1985 году и ставший простым, но захватывающим опытом.

В 1987 году была создана еще одна легендарная головоломка, Сапёр, которая полюбилась многим поколениям своей простотой и интересным геймплеем.

Головоломка – это игровой элемент или задание, которое требует от игрока решения логической задачи. Цель головоломок в играх может быть разной: от преодоления препятствия или открытия нового уровня до получения ценных предметов или достижения важного события в сюжете игры [21].

Головоломки в видеоиграх могут иметь разнообразные формы и механики. Некоторые из них требуют от игрока правильного расположения предметов или разгадывания шифров, а другие управления определенными элементами игрового мира. Нередко головоломки представляют собой серии логически связанных действий, которые нужно выполнить в определенном порядке для достижения результата.

Рассмотрим самые известные виды головоломок:

Загадки – это головоломки, включающие в себя различные элементы, такие как шифры, кроссворды, ребусы и т.д.

Исследование – игроку предстоит найти путь, через запутанные проходы лабиринтов, избегая препятствий.

Логические – задачи, основанные на логическом рассуждении и выводе. Игроку могут быть предоставлены условия или инструкции, и он должен определить правильное решение на основе представленной информации.

Таким образом, компьютерные игры представляют собой интерактивные развлечения, созданные с помощью компьютерной технологии. Они разнообразны по жанрам, включая шутеры, ролевые игры, симуляторы, стратегии, головоломки и приключения. Жанры помогают классифицировать игры по их основной механике и характеристикам, что облегчает выбор игр, соответствующих интересам игрока. Головоломки, в свою очередь, могут включать различные виды, такие как загадки, исследования и логические задачи.

1.2 Сравнительный анализ и выбор инструментальных средств разработки компьютерных игр

В этом параграфе проведем сравнительный анализ различных инструментальных средств разработки.

Game engine (далее «движок») – это комплекс программ, объединяющий различные функции для создания полноценного игрового опыта. Он обеспечивает графическую визуализацию, звуковое сопровождение, управление движением персонажей в игре и их взаимодействие в соответствии с заранее заданными скриптами. Благодаря этому игра становится более увлекательной и захватывающей [25].

В разработке компьютерных игр они используются для упрощения процесса создания игры.

Unity [38] – это кроссплатформенный движок, который позволяет разрабатывать 2D и 3D приложения и игры, логотип представлен на рисунке 1. Unity представлен в двух версиях: бесплатная и платная, отличающихся набором возможностей, важных при создании игр:

Бесплатная версия Unity поддерживает Android, Web Player и PC платформы.

Полная версия предоставляет разработчикам возможность публикации своих проектов на популярных платформах, таких как PC, Linux, Mac, Windows Store, iOS, Android, Windows Phone 10 Store, Blackberry 10, Wii U, PS3, Xbox 360, PS4 и Xbox One. Кроме того, с Unity можно разрабатывать приложения для виртуальной реальности (VR), такие как Hololens, Oculus Rift, StarVR и другие.

Unity можно настроить под свои нужды разработки компьютерной игры, можно изменить интерфейс, убрав некоторые элементы меню, и добавить собственные настройки, которые упростят процесс разработки. Unity обладает интуитивным Drag and Drop интерфейсом и поддерживает два языка программирования: C# и JavaScript [26].

К достоинствам относят:

- мультиплатформенность;
- удобный интерфейс;
- большая библиотека ассетов и плагинов.

К недостаткам относят:

- Плохая оптимизация больших пространств;
- Большие затраты памяти.



Рисунок 1 – Логотип Unity

Godot Engine – это универсальный игровой движок, который поддерживает разработку как 2D, так и 3D-игр на различных платформах. Он предоставляет полный набор инструментов, которые помогают разработчикам сосредоточиться на создании игр, вместо того чтобы заниматься повторной разработкой базовых функций, логотип представлен на рисунке 2. С помощью Godot Engine можно легко импортировать игры на множество платформ всего лишь в один клик. Поддерживаются основные настольные платформы, такие как Linux, macOS и Windows, а также мобильные платформы, включая Android и iOS, и веб-платформу HTML5 [31].

К достоинствам относят:

- Большая документация;

- Небольшой размер игр;
- Поддержка нескольких языков программирования.

К недостаткам относят:

- Отсутствие большого количества функций в связи с новизной движка;
- Мало обучающего контента.



Рисунок 2 – Логотип Godot engine

Unreal Engine – это игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Впервые появился в 1998 году с выходом игры «Unreal». С тех пор он стал основой для создания более сотни игр и других проектов, логотип представлен на рисунке 3.

Движок написан на языке C++ и предоставляет возможность разработки игр для широкого спектра операционных систем и платформ.

Unreal Engine использует модульную систему зависимых компонентов, что упрощает процесс портирования игр на разные платформы. Он поддерживает различные системы рендеринга, такие как Direct3D, OpenGL, Pixomatic, а в старых версиях также Glide, S3, PowerVR. Для воспроизведения звука используются EAX, OpenAL, DirectSound3D, а ранее была поддержка A3D. Также включены возможности голосового воспроизведения текста и распознавания речи. Движок предоставляет модули для работы с сетью и поддержку различных периферийных устройств.

При разработке игровой логики будем использовать язык программирования C++ или визуальную систему программирования Blueprint, что делает процесс создания игр более доступным [30].

К достоинствам относят:

- Возможность разработки без знаний языков программирования;
- Широкий функционал.

К недостаткам относят:

- Малое количество библиотеки ассетов;
- Сложный интерфейс
- Требования к системе.



Рисунок 3 – Логотип Unreal Engine

После анализа выше перечисленных движков определены субъективные критерии оценивания по пятибалльной шкале, где 1 – низкий уровень оценки и 5 самый высокий. Субъективная оценка, основанная на работе с каждым движком, представлена в таблице 3.

Таблица 3 – Сравнение игровых движков.

Критерии	Игровые движки		
	Unity	Godot Engine	Unreal Engine
Бесплатное использование	5	5	5
Средний порог вхождения	4	5	3
Количество платных и бесплатных дополнительных ресурсов	5	2	4
Уроки, курсы, документация	5	2	4
Среднее значение	4,75	3,5	4

Из сравнительного анализа субъективных критериев оценивания игровых движков представленных в таблице 3 следует, что Unity является оптимальным выбором. Он получил наивысший рейтинг и наилучшим образом соответствует выделенным критериям, таким как количество обучающего контента и уровень доступности для новичков.

Для написания скриптов в Unity используется язык программирования C# – это язык программирования, разработанный компанией Microsoft. C# является одним из основных языков разработки в платформе Microsoft .NET и используется для создания различных типов приложений, включая настольные приложения, веб-приложения, игры, мобильные приложения и многое другое.

C# является объектно-ориентированным языком программирования. Он предлагает широкий набор функций и инструментов для разработчиков, позволяя создавать мощные и эффективные программы. C# поддерживает основные принципы объектно-ориентированного программирования, такие как наследование, полиморфизм и инкапсуляцию.

C# включает в себя богатую стандартную библиотеку классов, которая предоставляет множество функций для работы с файлами, сетями, базами данных, графикой и другими аспектами разработки приложений.

Особенностью C# является его интеграция с платформой .NET, которая обеспечивает среду выполнения и другие сервисы для работы приложений, написанных на этом языке. C# используется для разработки программных решений под разные операционные системы, такие как Windows, macOS и Linux.

Среди популярных инструментов разработки на C# можно назвать Microsoft Visual Studio, Visual Studio Cod. C# позволяет разработчикам использовать множество сторонних библиотек и фреймворков для ускорения разработки и расширения функциональности приложений [40].

Поскольку C# является языком программирования, используемым в Unity, рекомендуется установить Visual Studio для более удобной работы.

Visual Studio – представляет собой инструментарий разработчика, который обеспечивает полный цикл разработки приложений. В качестве интегрированной среды разработки (IDE) предоставляет широкий набор функций для написания, редактирования, отладки и сборки кода, а также для развертывания готовых приложений.

Visual Studio не ограничивается только редактированием и отладкой кода. В функционал входят компиляторы, инструменты автозавершения кода, системы управления версиями и множество расширений и дополнительных возможностей, направленных на улучшение этапа процесса разработки программного обеспечения [32].

Для работы с графикой и звуком выбраны следующие инструменты:

Adobe Photoshop [29] – это растровый графический редактор, разработанный компанией Adobe Systems, обладающий широким набором функций для работы с изображениями;

Asperite [27] – это удобный редактор для создания анимации в стиле пиксельной графики;

Audacity – это комплексный инструмент для работы со звуком, который позволяет создавать, редактировать и восстанавливать аудиоконтент с помощью спектрального анализа и других функций;

FL Studio – это цифровая звуковая рабочая станция, которая предназначена для написания и производства музыки с помощью встроенных инструментов и эффектов.

В этом параграфе проведён сравнительный анализ различных инструментальных средств разработки игр, с целью определить наиболее подходящий для проектирования компьютерной игры в жанре головоломка. Изучены основные движки Unity, Godot Engine, Unreal Engine и рассмотрены их достоинства и недостатки. Кроме выбора движка, также определены инструменты для работы с графикой и звуком.

1.3 Общий алгоритм реализации проекта

Процесс разработки игры можно разделить на три условных этапа:

1. Формирование концепции:

- определение жанра игры, целевой аудитории, геймплея и пользовательского интерфейса;
- проработка сюжета, визуального и звукового оформления;
- выбор технических средств для реализации проекта.

2. Разработка игры:

- создание первого уровня игры и тестирование его работоспособности;
- разработка основного контента игры, включая графические ресурсы, геймплейные возможности и баланс;
- создание альфа-версии, которая содержит большую часть запланированного контента.

3. Выпуск и поддержка готового продукта:

- релиз игры в популярных цифровых магазинах для максимального охвата аудитории;
- поддержка игры, включая исправление ошибок с помощью патчей и добавление дополнительного контента через загружаемые материалы (DLC).

Формирование концепции – определяет жанр, целевую аудиторию, геймплей, пользовательский интерфейс, визуальное и звуковое сопровождение, сюжет и технические средства. Целевая аудитория важна для определения визуального стиля и особенностей геймплея. Жанр игры и возможное использование элементов других жанров определяются исходя из общей идеи проекта.

Разработка пользовательского интерфейса – учитывает требования геймплея и потребности пользователей. Интерфейс должен отображать необходимую информацию, не отвлекая от игрового процесса и не перегружая его.

Выбор технических средств – основывается на анализе доступных программ и инструментальных средств разработки. Выбор игрового движка важен, так как он влияет на весь процесс разработки. Он должен соответствовать требованиям проекта и возможностям разработчиков, учитывая языки программирования и финансовые аспекты.

После формирования концепции можно приступить к разработке игры. Разработка начинается с создания первого уровня, включающего основные элементы кода и графики. На этом этапе могут быть внесены изменения в концепцию игры. Появление первой игровой части позволяет начать тестирование, которое продолжается до выпуска игры. Затем создается альфа-версия, содержащая около 90% запланированного контента. На этом этапе разработчик занимается наполнением игры контентом.

Следующий этап – бета-версия, когда игра почти готова и полностью работоспособна. На этой стадии игра демонстрируется широкой публике для поиска и исправления ошибок, проблем логики и багов. Бета-версия содержит все ключевые особенности игры, достаточный контент для игры продолжительное время и настроенный сбор, и анализ статистики. Тестирование проводится по тест-плану функционального тестирования, включающего в себя проверку всех функций компьютерной игры. После окончательного тестирования и исправления ошибок, проект готов к выпуску.

На последнем этапе происходит выпуск игры и последующая поддержка. Поддержка игры включает исправление ошибок после обратной связи и добавление контента для поддержания интереса публики. Дополнительный загружаемый контент (DLC) может быть как бесплатным, так и платным, продлевая жизненный цикл игры и принося дополнительный доход.

Таким образом, можно сделать вывод о том, что этапы разработки игр мало отличаются, от этапов разработки любого другого продукта.

2 Проектирование и реализация компьютерной игры в жанре головоломка на платформе Unity

Разработка игр – это творческий и многогранный процесс, включающий в себя множество этапов, от зарождения идеи до создания полноценного продукта. Каждый этап требует тщательного планирования, учета технических аспектов и внимания к деталям. В этом параграфе рассмотрим разработку компьютерной игры в жанре головоломка на платформе Unity:

2.1 Планирование разработки проекта

Планирование является неотъемлемой частью разработки компьютерных игр, играющей важную роль в создании уникального и увлекательного игрового опыта. Этап планирования – это инструмент, который позволяет разработчикам определить целевую аудиторию, цели, задачи и основные элементы игры еще до того, как первый пиксель будет нарисован или первая строчка кода будет написана. В этом параграфе рассмотрим процесс планирования компьютерной игры и ключевые аспекты, которые необходимо учесть, чтобы создать качественный игровой продукт [32].

1 этап: Описание целевой аудитории

Для разработки компьютерной игры, как и для любого другого продукта, одну из самых важных ролей играет определение целевой аудитории. Когда определена целевая аудитория, разработчику намного легче определить многие аспекты разрабатываемого продукта. Для данного разрабатываемого проекта была выделена достаточно большая целевая аудитория, а именно категория лиц от 12 лет. Выбор такой категории обуславливается тем, что дети в 12 лет уже спокойно умеют читать и анализировать информацию.

2 этап: Основные функциональные требования к игре

Проект, представляет собой компьютерную игру в жанре головоломка. Основная цель игры – развитие логического и пространственного мышления, а также сообразительность игрока.

Проект должен отвечать следующим требованиям:

1. геймплей игры должен быть понятным и простым для игроков;
2. основная механика игры заключается в прохождении лабиринтов и решении загадок;
3. управление главным героем не должно быть слишком сложным;
4. меню игры не должно быть перегружено и избыточным;
5. игра должна быть оптимизирована для работы на различных компьютерах и не требовать высоких технических характеристик.

Данные требования к функциональной части являются основными, должны быть выполнены в первую очередь.

3 этап: Моделирование и описание игрового процесса

Для моделирования и описания игрового процесса UML является современным стандартом для визуализации и описания различных аспектов, предоставляющий набор графических элементов и нотаций для создания диаграмм, которые помогают визуализировать структуру и поведение системы.

При запуске компьютерной игры в жанре головоломка игроку предлагается на вкладке «Обучение» познакомиться с правилами взаимодействия с игрой, после этого приступает к прохождению уровня. После решения поставленной задачи игроку предлагается выбрать следующий уровень или выйти из игры. Диаграмма последовательности взаимодействия игрока с системой представлена на рисунке 4.

Диаграмма последовательности – вид диаграмм, используемый в разработке программного обеспечения для визуализации взаимодействия между объектами или компонентами системы.

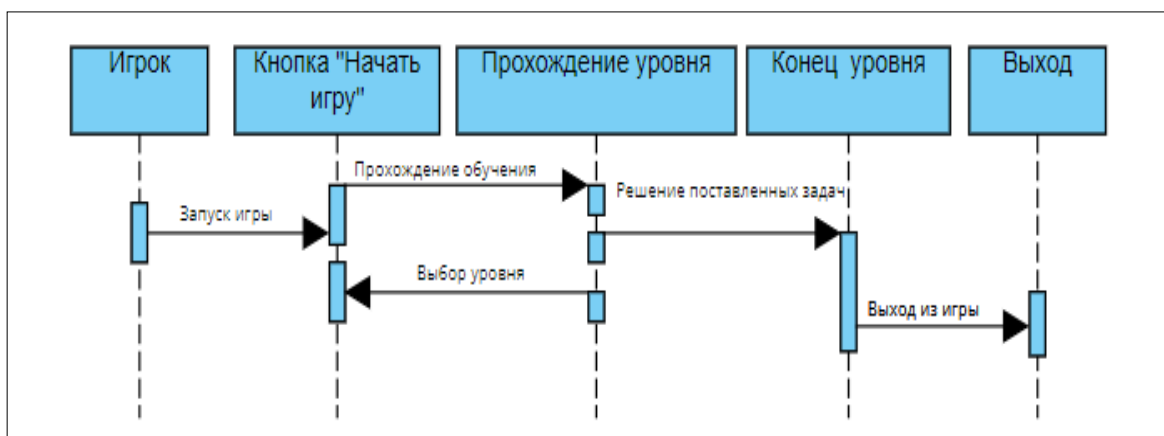


Рисунок – 4 Диаграмма последовательности компьютерной игры в жанре
ГОЛОВОЛОМКА

Таким образом, диаграммы последовательности являются полезными для анализа и проектирования системы, позволяют лучше понять и визуализировать последовательность взаимодействия между объектами и компонентами.

4 этап: Характеристика оборудования для разработки компьютерной игры

Так как для разработки игры выбран игровой движок Unity, а для написания и редактирования программного кода среда Microsoft Visual Studio 2019, то для обеспечения корректной работы этих инструментов требуется компьютер с определенными техническими характеристиками. Поэтому для разработки компьютерной игры в жанре головоломка использовали компьютер со следующими характеристиками:

- Процессор: Ryzen 5 2600X
- Видеокарта: 1660 TI с 6 Гб видеопамяти
- Оперативная память: 32 Гб
- Операционная система: Windows 10

Эти характеристики необходимы для обеспечения эффективной работы выбранных средств разработки.

Таким образом, планирование компьютерной игры в жанре головоломка играет решающую роль в создании качественного и увлекательного игрового продукта.

2.2 Реализация компьютерной игры в жанре головоломка

Разработка проекта – это фаза, когда идеи, планы и концепции превращаются в реальность. Этап, где создатели компьютерных игр приступают к созданию игрового мира. В этом параграфе рассмотрим процесс разработки проекта и важные аспекты, которые помогут превратить идею в полноценную компьютерную игру.

2.2.1 Установка программного обеспечения

Перед началом разработки компьютерной игры установлено специальное программное обеспечение – движок Unity с официального веб-сайта. Для этого перешли на официальный сайт Unity и выбрали тарифный план для частных лиц. Загрузка движка представлена на рисунке 5.

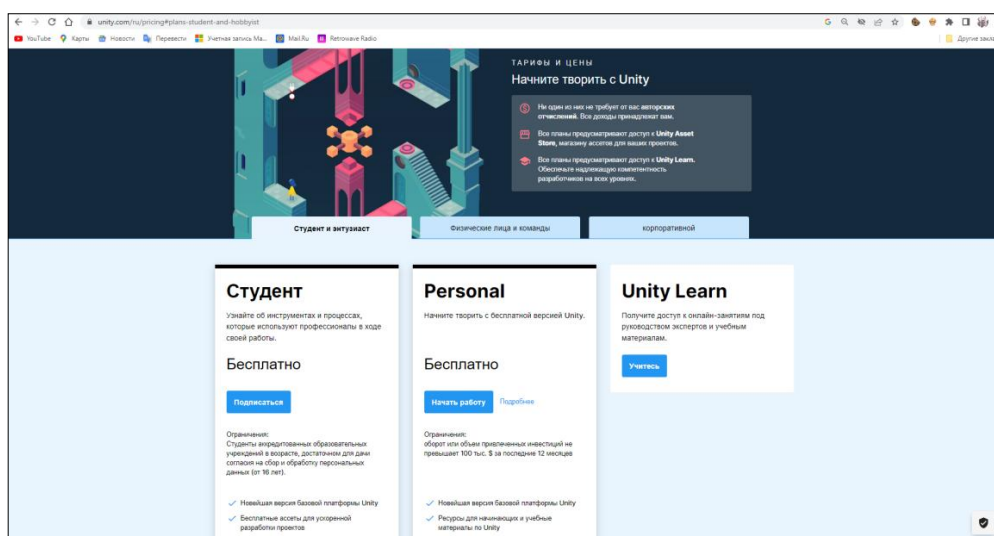


Рисунок 5 – Загрузка Unity

После того как Unity установлен, зарегистрировали UnityID, как показано на рисунке 6.

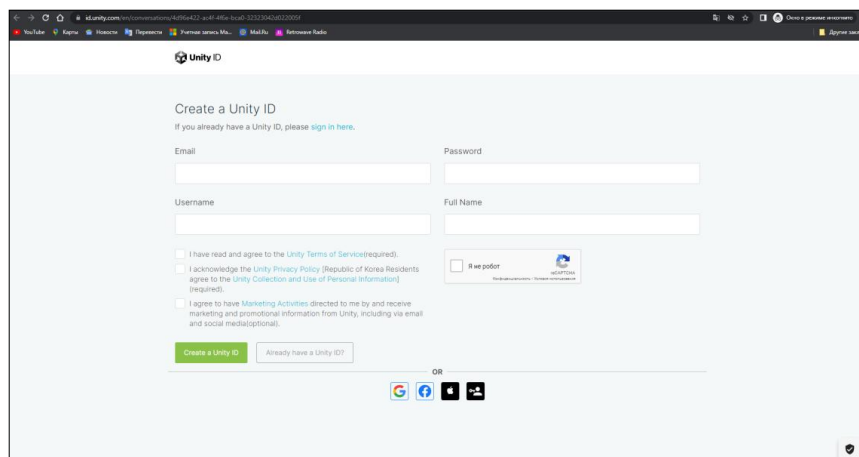


Рисунок 6 – Регистрация в UnityID

Аккаунт в UnityID откроет доступ к обучающим материалам, а также позволит использовать бесплатные ассеты из внутреннего магазина Unity.

Для рисования графики установили графический редактор Asperite, так как он приспособлен для рисования в стиле «PixelArt», интерфейс которого представлен на рисунке 7.

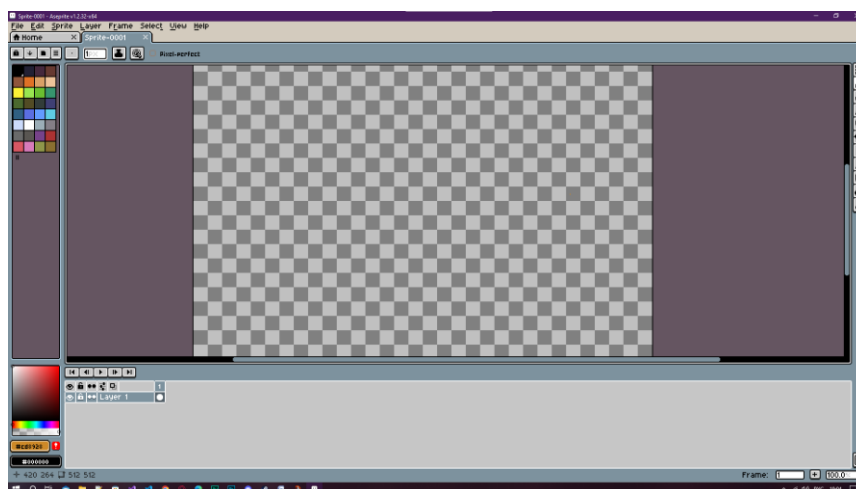


Рисунок 7 – Asperite

Для написания программного кода выбрал Microsoft Visual Studio 2019. Для его загрузки перешли на официальный сайт и выбрали тарифный план для пользователей [17], загрузка MVS 2019 представлена на рисунке 8.

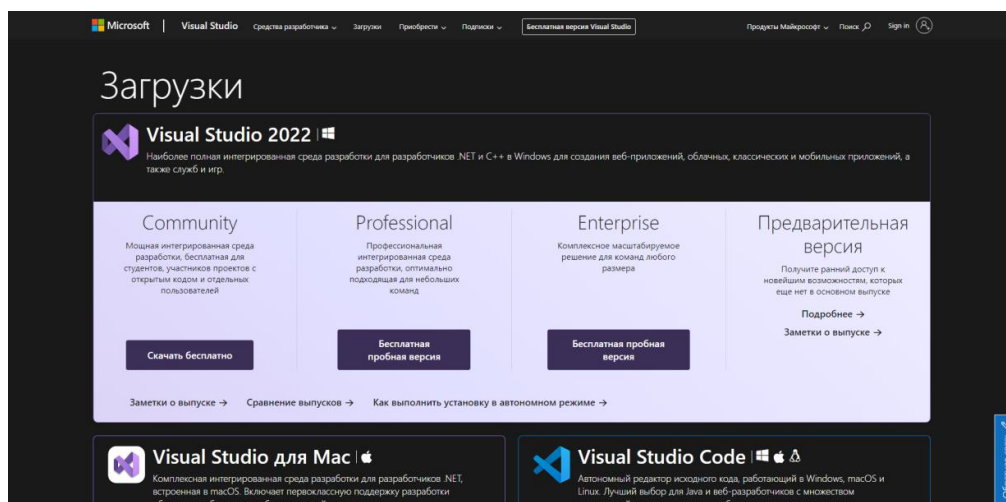


Рисунок 8 – Загрузка MVS 2019

2.2.2 Разработка графического оформления

Графическое оформление игры имеет несколько важных функций:

– *визуальное представление* – графическое оформление позволяет создать привлекательную и узнаваемую атмосферу игры. Оно включает в себя дизайн персонажей, фонов, объектов и интерфейса пользователя, что помогает создать уникальный стиль, который может захватить внимание и увлечь игроков [7].

– *информационная передача* – графический дизайн игры может использоваться для передачи важной информации игрокам. Графика может использоваться для передачи подсказок, инструкций или истории [7].

Выделим основные понятия, тайл (Tile) и спрайт (Sprite), применяемые в графическом оформлении игры.

Тайл (Tile) – небольших размеров повторяющийся фрагмент, который служит для постройки изображений больших размеров (Тайловая графика). Часто используется для создания уровней для двумерных игр.

Спрайт (Sprite) – графический объект, представляющий собой растровое изображение. Используется в компьютерной графике как основная единица для анимаций двумерных объектов.

Для компьютерной игры в жанре головоломка нарисуем персонажа:

1. создадим спрайт размером 256 на 256 пикселей, в качестве фона необходимо выбрать прозрачный фон;
2. нарисуем все состояния персонажа;
3. сохраним в формате .png.

Далее необходимо нарисовать тайлы для локации. Для этого:

1. Создадим спрайт размером 512 на 512 пикселей, в качестве фона необходимо выбрать прозрачный фон;
2. Заливаем квадрат размером 16 на 16 пикселей нужным цветом, а потом детализируем.

Атласы спрайтов и тайлов представлены на рисунках 9 – 10.



Рисунок 9 – Атлас спрайтов персонажа

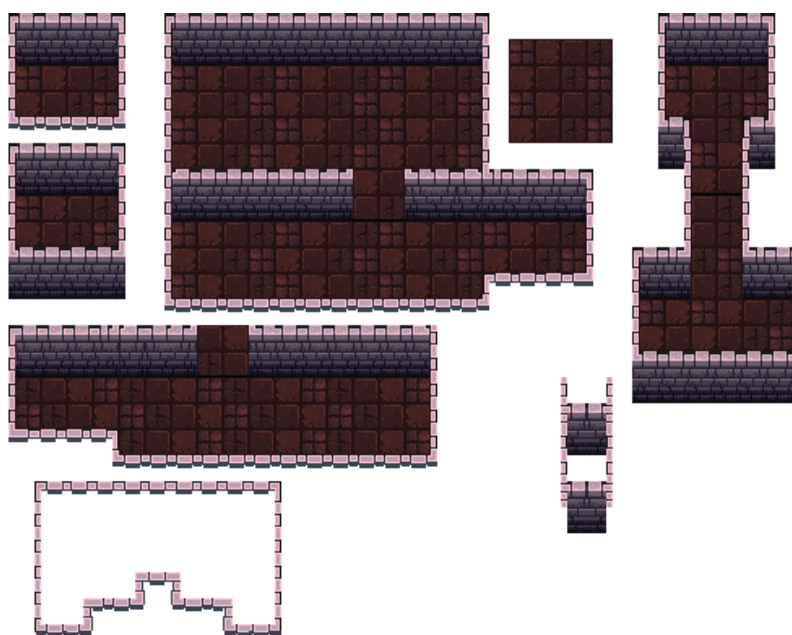


Рисунок 10 – Атлас тайлов

2.2.2 Создание пустого игрового проекта

Для реализации компьютерной игры выбрали игровой движок Unity и Microsoft Visual Studio 2019 для написания программного кода.

Этап разработки компьютерной игры начинается с запуска UnityHub. Появляется окно создания проекта, представленное на рисунке 11, в котором пользователю предлагается выбрать шаблон приложения. Доступны два варианта шаблонов:

1. *Шаблон 2D* – используется для создания игр с плоской графикой, известной как спрайты. В этом шаблоне игр отсутствует трехмерная геометрия. Спрайты отображаются на экране в виде плоских изображений.

2. *Шаблон 3D* – предназначен для создания трехмерных игр. В таких играх используется трехмерное пространство, где материалы и текстуры отрисовываются на поверхности игровых объектов, создавая полноценное окружение, персонажей и объекты игрового мира.

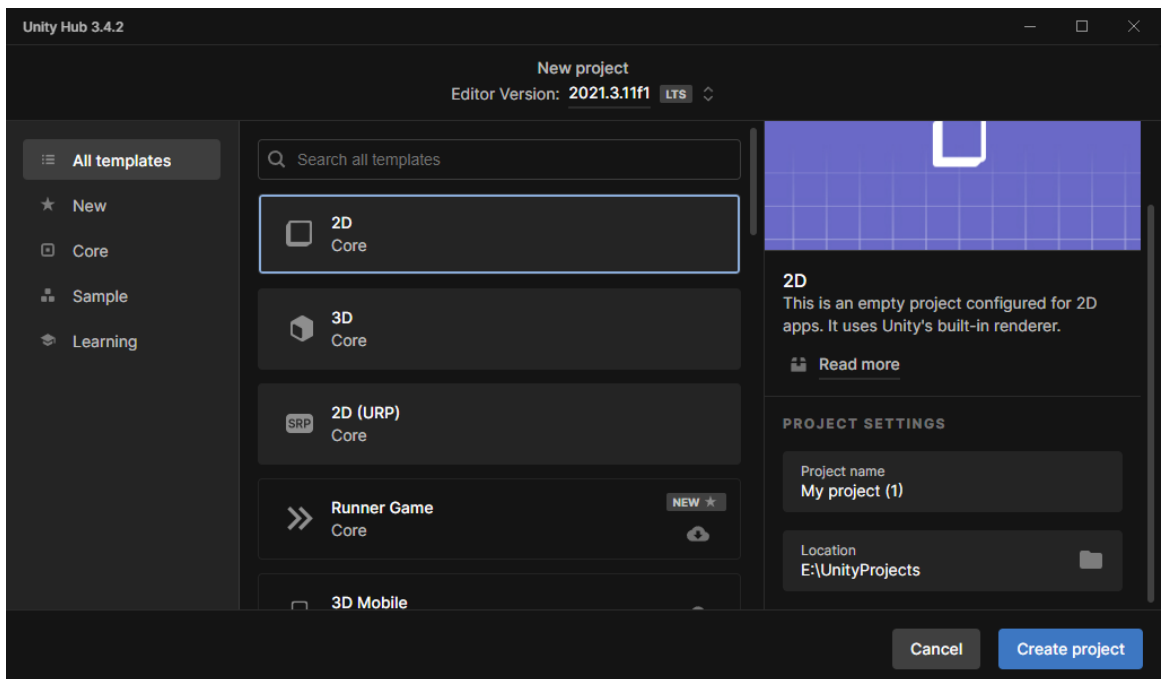


Рисунок 11 – Создание проекта

После выбора шаблона нажали кнопку «Create», после чего появляется пустой проект в Unity, как показано на рисунке 12.

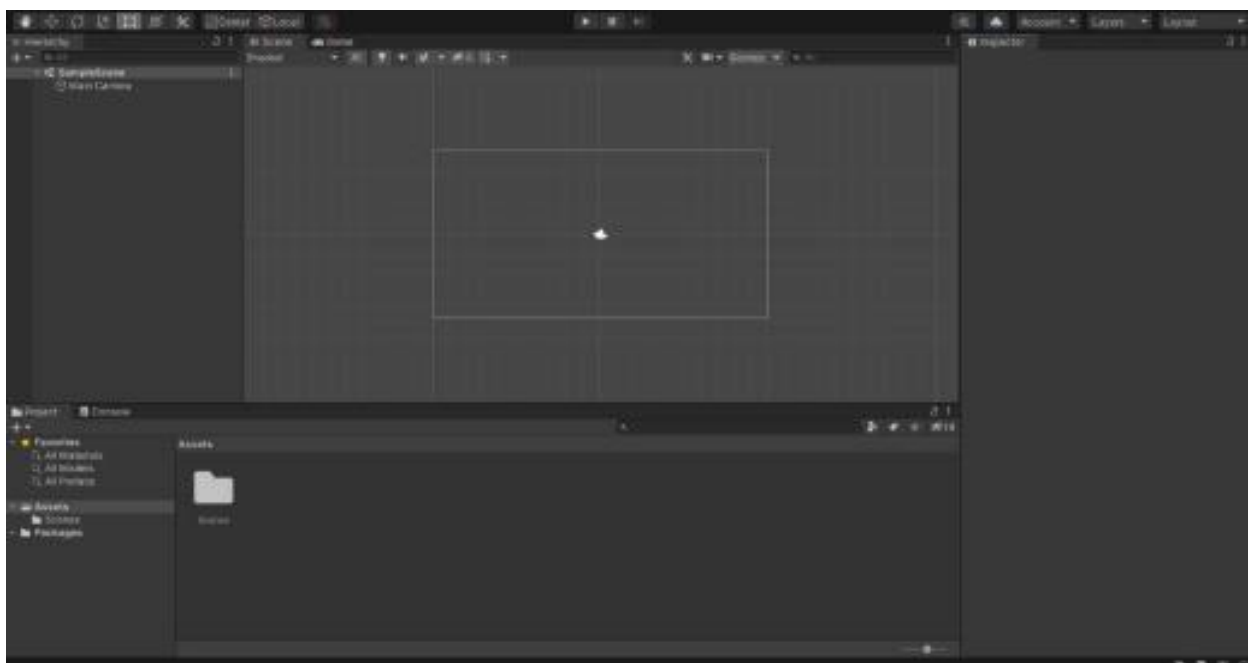


Рисунок 12 – Пустой проект в Unity

В центре окна находятся три панели: «Сцена», «Аниматор» и «Игра». «Сцена» используется для создания общей композиции уровня и добавления новых объектов. «Аниматор» предоставляет возможность создавать и редактировать анимации объектов. «Игра» показывает текущий вид игры, отображая изображение из камеры.

Слева находится панель «Иерархия», где отображаются все объекты, присутствующие на сцене. На данном этапе присутствует только камера.

Справа расположена панель «Инспектор», где отображаются текущие свойства выбранного объекта. Внизу окна расположены три панели: «Проект», и «Консоль». «Проект» отображает все объекты, добавленные в текущую игру, включая скрипты, анимации и другие элементы. «Консоль» служит для отображения ошибок и исключений, возникающих во время работы игры.

Прежде всего, необходимо настроить Unity. Для этого нужно найти настройку «ExternalScriptEditor», которая находится в разделе «Edit» → «Preferences» → «ExternalTools» и выбрать предварительно установленный Visual Studio. Это позволит открывать создаваемые скрипты непосредственно в Visual Studio.

2.2.3 Создание игровых объектов

Рассмотрим процесс создания объектов на примере создания персонажа. В корневом каталоге создадим две папки: «Tiles» и «Player» и добавим заранее подготовленные спрайты и тайлы. В папку «Player» размещаем атлас спрайтов персонажа изображенный на рисунке 9. В окне «Инспектор» в параметре «Sprite Mode» выбираем «Multiple», это нужно для того, чтобы Unity понимал, что на данном файле несколько спрайтов, для того чтобы персонаж не был размытым в параметре «Filter Mode» выбираем «Point (no filter)». Настройка атласа персонажа в Unity показана на рисунке 13.

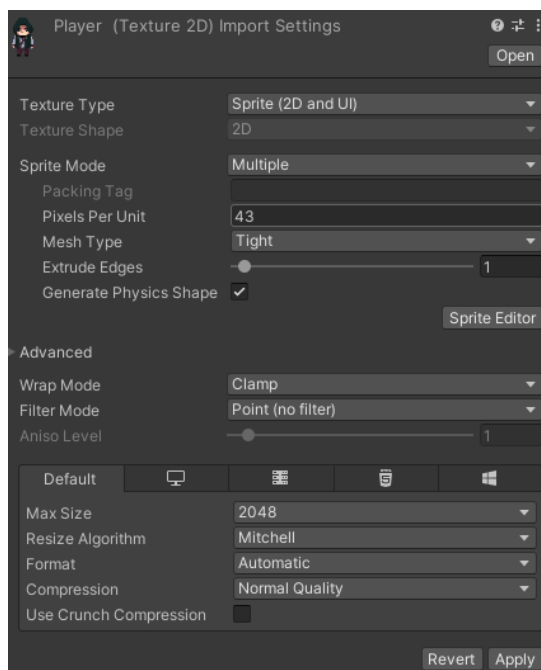


Рисунок 13 – Настройка атласа персонажа в Unity

После того как спрайт настроен необходимо его «порезать», в окне «Инспектор» нажмем на кнопку «Sprite Editor», в открывшемся окне в подменю «Slice» нажимаем на кнопку «Slice», применяем. В папку «Tiles» размещаем атлас тайлов и проделываем такие же действия, как с персонажем.

Разместим на игровой сцене первый объект, представляющий персонажа, управляемого игроком. Для этого необходимо добавить спрайт персонажа в текущую сцену, сначала создаём отдельный пустой объект в окне «Иерархия», а затем применяем к нему спрайт персонажа, представленный на рисунке 14.



Рисунок 14 – Персонаж

Для определения границ объекта необходимо добавить коллайдер. Компоненты коллайдера определяют форму объекта для физических столкновений. Чтобы добавить коллайдер к объекту, нужно выделить его в окне «Иерархия» и затем добавить компонент в окне «Инспектора». Для настройки размеров коллайдера используется кнопка «Edit Collider» показанная на рисунке 15.

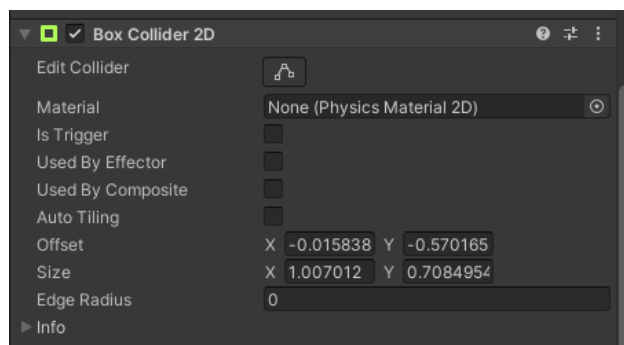


Рисунок 15 – Настройка Box Collider 2D

Unity позволяет сохранять объекты со всеми добавленными свойствами и скриптами для повторного использования. Такой объект называется префабом, его можно просто переместить на сцену, не создавая новый объект. Для создания префабов рекомендуется создать отдельную папку, чтобы они не затерялись среди других элементов игры. Также необходимо добавить компонент RigidBody2D к персонажу. Поскольку игра будет иметь вид «Top Down» (изометрическая проекция сверху), в свойствах RigidBody2D нужно установить флажок «Freeze Rotation» по оси Z и установить значение 0 в параметре «Gravity Scale», иначе персонаж будет постоянно падать вниз.

Так как главный герой должен двигаться и выполнять различные действия, нам нужно создать скрипт на языке C#.

Скрипт – это последовательность команд, выполняющих определённую задачу.

Скрипт можно создать несколькими способами: создать его в окне «Проект» или создать непосредственно на необходимом объекте в окне «Инспектора».

При создании нового скрипта Unity сам создает его структуру и некоторые из методов, представленных в таблице 3.

Таблица 3 – Методы Unity

Методы	Способ вызова
Awake	Вызывается 1 раз в момент появления объекта на сцене
Start	Вызывается 1 раз при активации скрипта
Update	Вызывается постоянно при каждом обновлении экрана
FixedUpdate	Вызывается постоянно через определённый промежуток времени
OnEnable	Вызывается 1 раз при активации объекта
OnDisable	Вызывается 1 раз при деактивации объекта
OnDestroy	Выполняется 1 раз при уничтожении объекта

После создания скрипта открывается Microsoft Visual Studio, где уже подключены основные библиотеки. Для начала создадим переменные для аниматора и Rigidbody2D, а также создадим публичную переменную для параметра скорости как показано на рисунке 16.

```
Animator animator;  
Rigidbody2D rb;
```

Рисунок 16 – Переменные для PlayerController

Далее используем метод «Start» и в нем инициализируем компоненты: Rigidbody2D и аниматор в их переменные как представлено на рисунке 17.

```
private void Start()  
{  
    rb = GetComponent<Rigidbody2D>();  
    animator = GetComponent<Animator>();  
}
```

Рисунок 17 – Инициализация компонентов

Для передвижения персонажа используем метод «FixedUpdate» в нем создаём условие: если код нажатой клавиши «D», то Rigidbody2D толкает персонажа на вектор со значениями по оси X равно переменной speed, по оси Y значение равно 0, иначе, если код нажатой клавиши «A», то Rigidbody2D толкает персонажа на вектор со значениями по оси X равно переменной «-speed», по оси Y значение равно 0. Аналогичным образом задаём движение вверх и вниз как показано на рисунке 18.

```
if (Input.GetKey(KeyCode.D))
{
    rb.velocity = new Vector2(speed, 0);
}
else if (Input.GetKey(KeyCode.A))
{
    rb.velocity = new Vector2(-speed, 0);
}
else if (Input.GetKey(KeyCode.W))
{
    rb.velocity = new Vector2(0, speed);
}
else if (Input.GetKey(KeyCode.S))
{
    rb.velocity = new Vector2(0, -speed);
}
```

Рисунок 18 – PlayerController

2.2.4 Создание анимаций персонажа

Теперь персонаж умеет ходить, однако визуально его спрайт будет просто перемещаться по сцене, чтобы такого не было нужно добавить анимации:

- анимацию покоя (idle);
- анимацию передвижения (walk);
- анимацию поднятия предметов (hold).

Для этой задачи в Unity есть компонент «Аниматор» – это система, предназначенная для управления анимацией. Она работает подобно плееру, который воспроизводит музыку. Аниматор предоставляет возможность гибкого создания дерева анимаций с бесчисленными вариантами разветвлений [16].

Для создания анимации выберем нужный объект в «Инспекторе» и во вкладке «Window» выберем пункт «Animation». Появляется окно создания анимации представленное на рисунке 19.

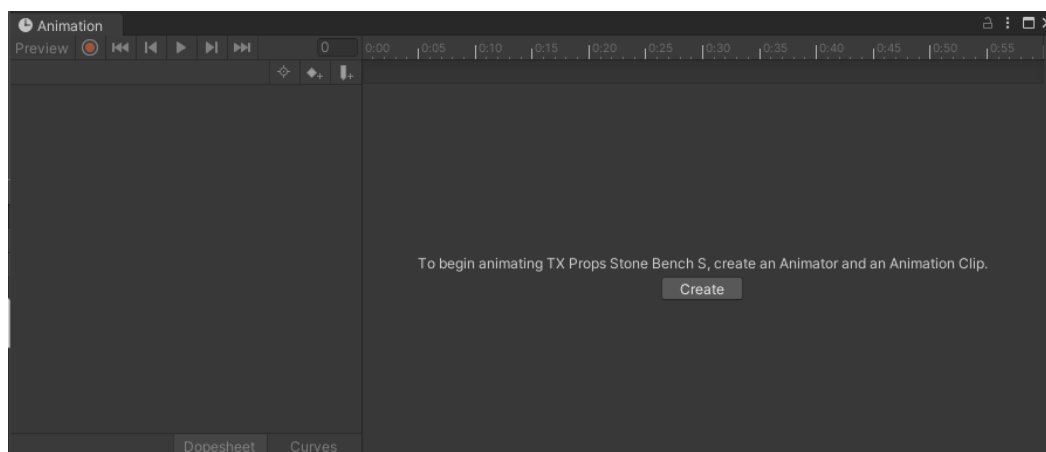


Рисунок 19 – Окно создания анимации

С помощью этого окна можно превратить кадры в анимацию. Для этого нажимаем на кнопку «Create», изображенную на рисунке 19. После этого появится новое окно, в котором должны указать название первой анимации. Назовем ее «idle». Теперь анимация создана, но она пока пуста и не содержит никаких кадров.

Перенесём спрайты для анимации покоя из атласа спрайтов персонажа изображенные на рисунке 9, в пустое пространство окна «Animation». Анимация состояния покоя «idle» готова. Аналогичным образом создаём остальные анимации.

После создания первой анимации автоматически формируется контроллер анимаций с именем, совпадающим с именем объекта, для которого она создана. В данном случае контроллер анимаций будет называться «Player». Контроллер анимаций содержит все состояния анимаций данного объекта и обеспечивает переключение между ними.

Так же, после создания первой анимации, в «Инспекторе» объекта создаётся компонент «Аниматор», который несёт в себе ссылку на контроллер

анимаций. Благодаря этому компоненту в дальнейшем через скрипт C# будут меняться состояния анимации с одной на другую.

Зададим условия для переключения анимации, для этого изменим скрипт «PlayerController», как показано на рисунке 20.

```
private void FixedUpdate()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        SceneManager.LoadScene(0);
    }
    if (takeright.hold == true || takeleft.hold == true)
    {
        if (Input.GetKey(KeyCode.D))
        {
            rb.velocity = new Vector2(speed, 0);
            animator.Play("holdright");
        }
        else if (Input.GetKey(KeyCode.A))
        {
            rb.velocity = new Vector2(-speed, 0);
            animator.Play("holdleft");
        }
        else if (Input.GetKey(KeyCode.W))
        {
            rb.velocity = new Vector2(0, speed);
            animator.Play("holdup");
        }
        else if (Input.GetKey(KeyCode.S))
        {
            rb.velocity = new Vector2(0, -speed);
            animator.Play("holddown");
        }
        else
        {
            rb.velocity = new Vector2(0, 0);
            animator.Play("holdidle");
        }
    }
}
```

Рисунок 20 – PlayerController

В методе FixedUpdate добавим условие, которое будет проверять, держит персонаж какой-либо объект или нет, это нужно для того, чтобы реализовать переключение анимации «walk» и «hold». В условия нажатия клавиши внесём переключение анимации через «аниматор». Когда игрок будет нажимать на кнопку передвижения, то будет проигрываться нужная анимация. Условия переключения анимаций в аниматоре показаны на рисунке 21.

Все скрипты, отвечающие за управление персонажем, представлены в приложении А.

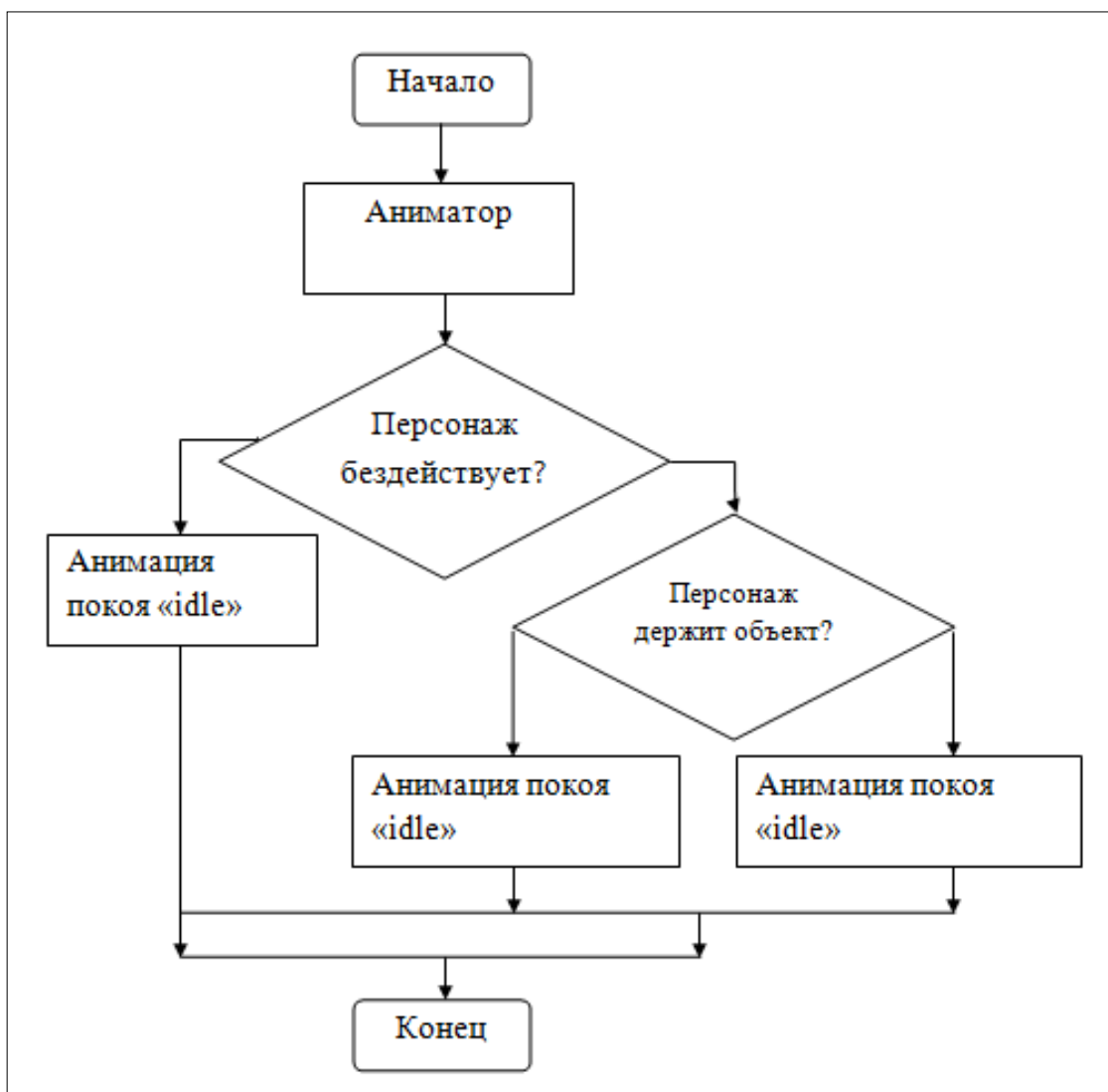


Рисунок 21 – Условия переключения анимации в аниматоре

2.2.5 Создание игрового уровня

Уровни в игре представлены в виде лабиринтов с логическими головоломками и загадками. Для их создания используется система тайловых карт (Tilemap). Tilemap – компонент, который позволяет собирать или наносить тайлы на большой сетчатой области. Tilemap состоит из коллекции тайлов – Tile Palette, благодаря этому инструменту можно создавать палитры, только вместо различных цветов будут тайлы.

Для создания Tilemap нужно выбрать вкладку GameObject → 2D Object → Tilemap → Rectangular. После этого в окно иерархии сцены добавится новый

объект – Tilemap. В инспекторе у этого объекта уже будут добавлены 2 компонента:

1. Tilemap – используется Unity для хранения тайлов на сцене;
2. Tilemap Renderer – используется для назначения материала, который будет использоваться для рендеринга тайлов в Tilemap.

Для работы с Tilemap необходимо создать Tile Palette, это можно сделать через вкладку Window → 2D → Tile Palette. После выполнения этих действий появится окно палитры, изображенное на рисунке 22.

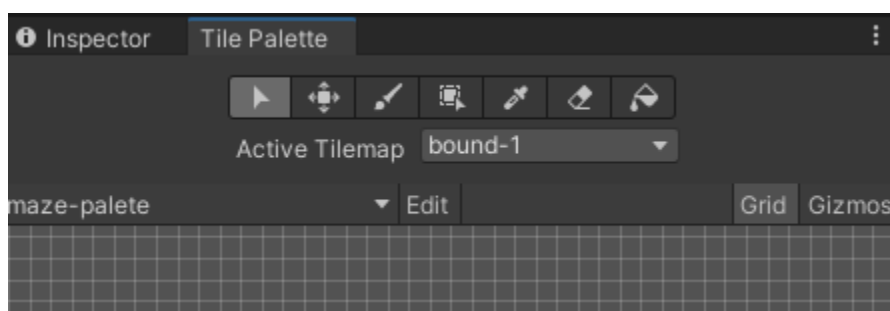


Рисунок 22 – Tile Palette

Для использования Tile Palette нужно загрузить в неё тайлы. Перенесем атлас тайлов, изображенный на рисунке 10 из папки Tiles в пустое пространство на палитре тайлов. После этого палитра будет готова к использованию.

Создадим 2 тайловых карты и назовём их:

1. Ground – отвечает за напольные плитки и носит исключительно декоративный характер;
2. Bound – отвечает за стены. Основная задача создать границы карты.

В качестве основы для создания уровней были взяты лабиринты Я. И. Перельмана [17]. Построим локацию, используя кисть в Tile Palette, как показано на рисунке 23.

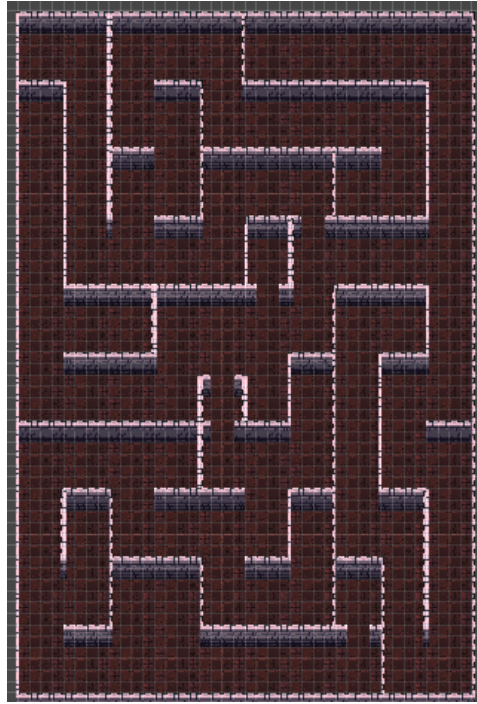


Рисунок 23 – Первый уровень

Для того, чтобы персонаж не проходил сквозь стены, выделим Tilemap: Bound в иерархии сцены и в инспекторе добавим компонент Tilemap Collider 2D. Теперь стены получают границы, через которые нельзя пройти, как показано на рисунке 24.

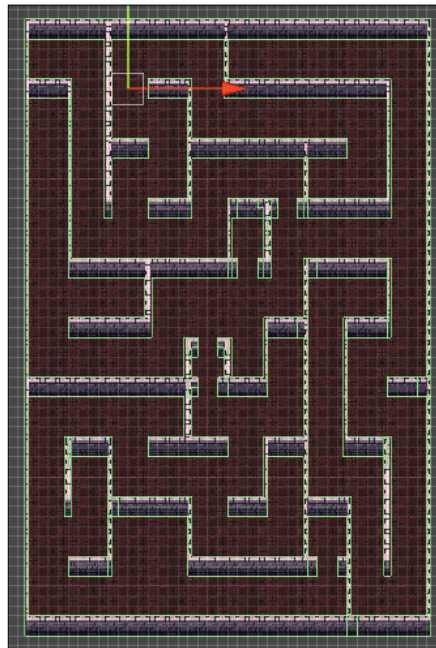


Рисунок 24 – Границы первого уровня

2.2.6 Создание диалоговой системы

Для реализации загадок в игре необходимо реализовать диалоговую систему – система, созданная для общения с пользователем.

Для создания диалоговой системы напишем скрипт, который отвечает за сериализацию строк в XML коде, представленный на рисунке 25.

```
[XmlAttribute("dialogue")]
Ссылка: 8
public class Dialogue
{
    [XmlElement("text")]
    public string text;

    [XmlElement("node")]
    public Node[] nodes;

    Ссылка: 2
    public static Dialogue Load(TextAsset _xml)
    {
        XmlSerializer serializer = new XmlSerializer(typeof(Dialogue));
        StringReader reader = new StringReader(_xml.text);
        Dialogue dial = serializer.Deserialize(reader) as Dialogue;
        return dial;
    }
}

[System.Serializable]
Ссылка: 3
public class Node
{
    [XmlElement("npctext")]
    public string Npctext;

    [XmlArray("answers")]
    [XmlArrayItem("answer")]
    public Answer[] answers;
}

ссылка: 1
public class Answer
{
    [XmlAttribute("tonode")]
    public int nextNode;
    [XmlElement("text")]
    public string text;
    [XmlElement("dialend")]
    public string end;
}
```

Рисунок 25 – Скрипт DialogueManager

Скрипт позволит создавать диалоги на языке XML, что упростит работу с ними в дальнейшем и позволит создавать различные диалоговые ветки.

Для взаимодействия с диалоговой системой используются компоненты Unity UI:

«Canvas» – представляет собой пространство, в котором производится настройка и отрисовка UI [30];

– «Image» – компонент, использующийся для отображения спрайтов;

– «Text» – компонент, использующийся для отображения текста;

– «Button» – реагирует на нажатие пользователя и используется для инициализации или подтверждения действия.

Схема расположения элементов диалогового окна представлена на рисунке 26.

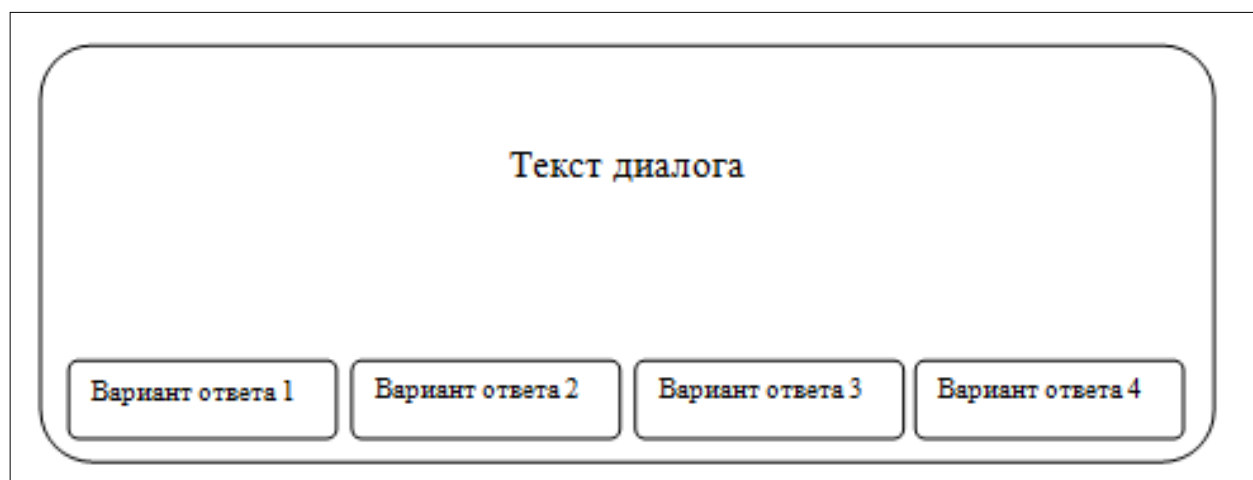


Рисунок 26 – Схема расположения диалогового окна

2.2.7 Создание сцены главного меню

Главное меню содержит несколько кнопок:

«Начать игру» – при нажатии на кнопку, сменится сцена, на которой будет кратко рассказано об игре, какие кнопки, за что отвечают и что в ней нужно делать, после чего запустится первый уровень.

«Загрузить уровень» – при нажатии на кнопку сменится сцена, на которой можно выбрать любой доступный уровень.

«Выход» – завершает работу приложения.

Модель диаграммы выбора в главном меню представлена на рисунке 27.

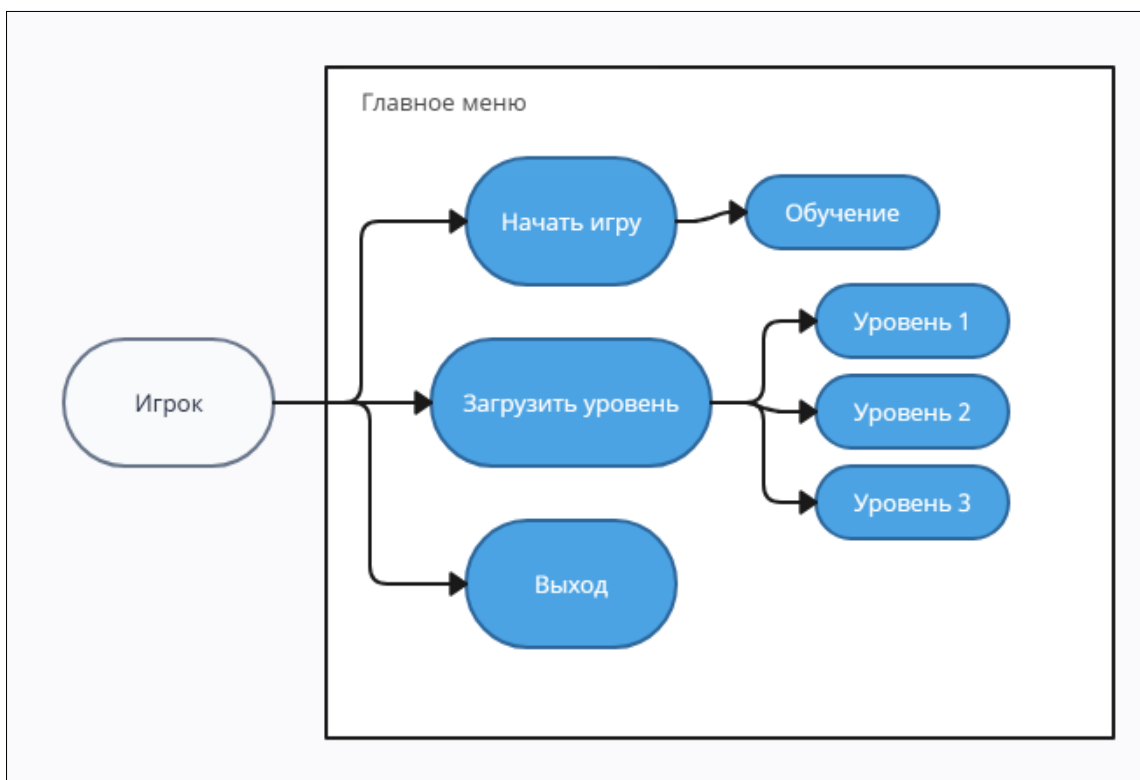


Рисунок 27 Модель UML диаграммы выбора

2.3 Функциональное тестирование

Функциональное тестирование направлено на проверку способности программного обеспечения, решать задачи, необходимые пользователям и удовлетворять функциональным требованиям. Оно фокусируется на анализе спецификаций функциональности компонента или системы в целом и предварительно определенных сценариев поведения. Такое тестирование позволяет убедиться в соответствии реализации ПО функциональным требованиям и его способности выполнять задачи в различных условиях [18].

Результаты тестирования представлены в таблице 4.

Таблица 4 – Функциональное тестирование

№	Название теста	Действия тестировщика	Ожидаемый результат	Прохождение теста
1	Проверка функциональности кнопки «Начать игру»	При нажатии на кнопку, открывается окно с обучением	При нажатии на кнопку, сцена должна смениться на сцену с обучением	Да
2	Проверка функциональности кнопки «Загрузить уровень»	При нажатии на кнопку, игра предлагает выбрать уровень	При нажатии на кнопку, сцена должна смениться на сцену выбором уровня	Да
3	Проверка функциональности кнопки «Выход»	При нажатии на кнопку, окно игры закрывается, процесс игры не отображается в диспетчере задач	При нажатии на кнопку, приложение закрывается	Да
4	Правильные действия при обучении	При нажатии клавиш «WASD» персонаж перемещался в нужном направлении	Игрок перемещается в правильном направлении	Да
5	Нажатие нескольких клавиш при передвижении	При одновременном нажатии клавиш «WD» персонаж двигался либо вправо, либо вверх	Персонаж не должен перемещаться по диагонали	Да
6	Остановка игры	При нажатии клавиши «ESC» появляется меню	Открытие меню паузы, остановка игры	Да
7	Продолжение игры	При нажатии на кнопку «продолжить» игра возобновляется	При нажатии на кнопку «продолжить» Игра должна продолжиться	Да
8	Прохождение уровня	При прохождении уровня сменяются на следующий	При прохождении уровня должен произойти переход на следующий уровень	Да
9	Сбор монет	При сборе количество монет изменяется	Изменение количества монет при сборе	Да

ЗАКЛЮЧЕНИЕ

На основе анализа учебной и научно-технической литературы по теме исследования определен понятийно-категориальный аппарат по теме исследования. В рамках исследования использовали понятие компьютерной игры автора А.А. Васильева. Выделили основные жанры компьютерных игр: шутеры; головоломки; стратегии; симуляторы; ролевые игры; приключения. Рассмотрели историю жанра головоломка.

Проанализированы популярные инструментальные средства разработки компьютерных игр, проведено их сравнение и выбрано наиболее подходящее средство разработки по выделенным субъективным критериям.

Рассмотрены основные этапы планирования компьютерной игры в жанре головоломка такие как: описание целевой аудитории; основные функциональные требования к игре, моделирование и описание игрового процесса, характеристика оборудования для разработки компьютерной игры.

В ходе разработки компьютерной игры в жанре головоломка изучены следующие возможности движка Unity:

- создание проекта;
- создание сцен;
- создание анимации;
- настройка объектов;
- создание UI компонентов.

Таким образом, цель бакалаврской работы – теоретически обосновать и разработать компьютерную игру в жанре головоломка на платформе Unity, была достигнута, поставленные задачи решены.

Результаты исследований представлены на следующих научных мероприятиях:

1. VI Всероссийская научно-практическая конференция преподавателей, учителей, студентов и молодых ученых «Актуальные проблемы преподавателей

дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 7-12 ноября 2022 г., диплом 2 степени).

2. Всероссийский молодёжный научный форум «Современное педагогическое образование: теоретические и прикладные аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 11 апреля 2023 г., диплом 3 степени).

3. Международная научно-практическая конференция «Исследования в современной науке» (Краснодар, КЦНТИ – филиал ФГБУ 30 марта 2023 г.)

По результатам исследования опубликованы статьи:

2. Бацеко, Р.В. Платформа Unity как средство разработки игр в жанре головоломка / Р.В. Бацеко // Исследования в современной науке: сборник научных статей международной научно-практической конференции. – Краснодар, 2023 – 30 стр.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бонд, Д.Г. Unity и С# Геймдев от идеи до реализации / Д.Г. Бонд. – Санкт-Петербург: Питер, 2019.
2. Бацеко, Р.В. Платформа Unity как средство разработки игр в жанре головоломка / Р.В. Бацеко – Краснодар, 2023 – 30 стр.
3. Виды компьютерных игр / URL: <https://wobla.ru/infomat/games.aspx#> (дата обращения: 13.04.2023)
4. Головоломка (жанр компьютерных игр) / URL: [https://ru.wikipedia.org/wiki/Головоломка_\(жанр_компьютерных_игр\)](https://ru.wikipedia.org/wiki/Головоломка_(жанр_компьютерных_игр)) (дата обращения: 10.02.2023).
5. Гейг, М. Разработка игр на Unity за 24 часа / М. Гейг. – Москва : Бомбора, 2020. – 466 с.
6. Денисов, Д. В. Разработка игры в Unity. С нуля до реализации / Д. В. Денисов. – ЛитРес:Самиздат, 2021.
7. Использование Unity для разработки приложений / URL: <https://android-tools.ru/coding/ispolzovanie-unity-dlya-razrabotki-prilozhenij/> (дата обращения: 11.01.2023).
8. Ильин, В. Основы создания 2D персонажа в Unity 3D / URL: <https://habrahabr.ru/post/211472/> (дата обращения: 15.03.2023).
9. Коситова, Я. С. История создания жанра игр «головоломка» и перспективы развития на медиа рынке 21 века/ Я. С. Коситова. – Ростов-На-Дону, 2019.
10. Мэннинг, Д. Unity и для разработчика / Д. Мэннинг. – Санкт-Петербург: Питер, 2018.
11. Майерс, Г. Искусство тестирования программ/ М. Майерс: ЛитРес:Самиздат, 2018.
12. Официальный сайт Unity. / URL: <https://unity3d.com/ru> (дата обращения: 20.03.2023).

13. Официальный сайт Visual Studio. / URL: <https://www.visualstudio.com/ru/vs/> (дата обращения: 20.03.2023).
14. О Godot Engine / URL: <https://docs.godotengine.org/ru/stable/about/introduction.html#> (дата обращения: 10.04.2023).
15. Официальный сайт компании Unreal Engine / URL: <https://www.unrealengine.com/en-US/download> (дата обращения: 25.03.2023).
16. Официальный сайт компании Godot / URL: <https://godotengine.org/download/windows/> (дата обращения: 10.02.2023).
17. Основы анимации в Unity: пер. с англ. Р.Рагимова. – М.: ДМК Пресс, 2016.– 176 с.: ил / А.Торн ISBN 978-5-97060-377-2
18. Перельман, Я.И. Дом занимательной науки: Лабиринты / Я. И. Перельман. – Проспект, 2022 г.
19. Платов, В. Я. Деловые игры: разработка, организация, проведение / В. Я. Платов. – Москва: Профиздат, 1991. – 192 с. – ISBN 5-255-00129-5.
20. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование разработка/ Дж. Рамбо. – СПб.: Питер, 2007. – 544 с.
21. Создание головоломок. Часть 1: Какие бывают головоломки / URL: <https://ifhub.club/2015/09/20/sozдание-golovolomok-chast-1-kakie-byvayut-golovolomki.html> (дата обращения: 09.02.2023).
22. Сергеев, Е. С. Разработка профориентационной vr-игры на платформе unity / Е. С. Сергеев, А.Е. Сухова, И.С Максимов, Н.А. Сенаторов // Научное обозрение. Технические Науки: сборник статей II Международной научно-практической конференции. – 2021. – С. 38-42. – URL: <https://www.elibrary.ru/item.asp?id=45691799> (дата обращения: 09.11.2022)
23. Торн, А. Основы анимации в Unity: учебное пособие/ А. Торн – ред.: Д. Мовчан, переводчик: Р. Рагимов. – Москва: ДМК, 2016 – 176с.

24. Термин «компьютерная игра»: опыт междисциплинарного анализа / URL: <https://cyberleninka.ru/article/n/termin-kompyuternaya-igra-opyt-mezhdistsiplinarnogo-analiza/viewer> (дата обращения: 15.02.2023).
25. Хокинг Дж. Unity в действии. Мультиплатформенная разработка на C#: учебное пособие/ Джозеф Хокинг – Санкт-Петербург: Питер, 2016 – 336с.
26. Что такое игровой движок? / URL: <https://club.dns-shop.ru/blog/t-64-videoigryi/34701-chto-takoe-igrovoi-dvijok/> (дата обращения: 20.11.2022).
27. Что такое Visual Studio? / URL: <https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022> (дата обращения: 10.02.2023).
28. Asprite / URL: <https://www.asprite.org/> (дата обращения: 29.01.2023).
29. Adobe Photoshop / URL: <https://www.adobe.com/products/photoshop/> (дата обращения: 29.01.2023).
30. C# Programming Guide / URL: <https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx> (дата обращения: 10.02.2023).
31. The evolution of video gaming and content consumption / URL: <https://www.pwc.com/us/en/industry/entertainmentmedia/publications/assets/pwc-video-gaming-and-content-consumption.pdf> (дата обращения: 20.11.2022).
32. Unity Manual. / URL: <http://docs.unity3d.com/Manual/index.html> (дата обращения: 10.12.2022).
33. Godot / URL: <https://godotengine.org> (дата обращения: 20.11.2022).
34. HeadHunter: Разработчик на Unity / URL: https://abakan.hh.ru/search/vacancy?clusters=true&no_magic=true&ored_clusters=true&enable_snippets=true&salary=&text=Unity+c%23&from=suggest_post (дата обращения: 25.12.2022).
35. Unity стандарт мобильной разработки / URL: <https://ue4daily.com/blog/growfall-games> (дата обращения: 26.02.2023).

36. Unreal Engine. Внутренние курсы / URL: <https://www.unrealengine.com/en-US/onlinelearning-courses> (дата обращения: 20.01.2023).

37. Unity / URL: <https://ru.wikipedia.org/wiki/Unity> (дата обращения: 20.01.2023).

38. Unity 5 tutorial for beginners: 2D Platformer – Moving Platform, YouTube / URL: https://www.youtube.com/watch?v=4R_AdDK25kQ (дата обращения: 15.12.2022).

39. Unreal Engine как средство разработки игр / URL: https://gamegod.fandom.com/ru/wiki/Unreal_Engine (дата обращения: 10.02.2023).

40. Unity : официальный сайт / Unity Technologies, 2023 – . – URL: <https://unity.com/download> (дата обращения: 15.09.2022) (дата обращения: 10.02.2023).

41. Unity в действии. Мультиплатформенная разработка на C#. Издательский дом. – Санкт-Петербург / Д.Хокинг; под редакцией С. М. Черников. – Санкт-Петербург, 2019. - 351 с. – ISBN 978-5-4461-0816-9.

42. VisualStudio / URL: <https://visualstudio.microsoft.com/ru/vs/> (дата обращения: 20.11.2022).

ПРИЛОЖЕНИЕ А
Листинг C# кода, отвечающего за работу персонажа

1. Скрипт отвечающий за передвижение персонажа:

```
public class PlayerController : MonoBehaviour
{
    Animator animator;
    Rigidbody2D rb;
    SpriteRenderer sprite;
    TakeAndDropRight takeright;
    TakeAndDropLeft takeleft;

    [SerializeField]
    int speed;

    private void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
        sprite = GetComponent<SpriteRenderer>();
        takeright = GetComponent<TakeAndDropRight>();
        takeleft = GetComponent<TakeAndDropLeft>();
    }

    private void FixedUpdate()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            SceneManager.LoadScene(0);
        }
    }
}
```

```

}
if (takeright.hold == true || takeleft.hold == true)
{
    if (Input.GetKey(KeyCode.D))
    {
        rb.velocity = new Vector2(speed, 0);
        animator.Play("holdright");
    }
    else if (Input.GetKey(KeyCode.A))
    {
        rb.velocity = new Vector2(-speed, 0);
        animator.Play("holdleft");
    }
    else if (Input.GetKey(KeyCode.W))
    {
        rb.velocity = new Vector2(0, speed);
        animator.Play("holdup");
    }
    else if (Input.GetKey(KeyCode.S))
    {
        rb.velocity = new Vector2(0, -speed);
        animator.Play("holddown");
    }
    else
    {
        rb.velocity = new Vector2(0, 0);
        animator.Play("holdidle");
    }
}

```

```
else
{
    if (Input.GetKey(KeyCode.D))
    {
        rb.velocity = new Vector2(speed, 0);
        animator.Play("walkR");
    }
    else if (Input.GetKey(KeyCode.A))
    {
        rb.velocity = new Vector2(-speed, 0);
        animator.Play("walkL");
    }
    else if (Input.GetKey(KeyCode.W))
    {
        rb.velocity = new Vector2(0, speed);
        animator.Play("walkU");
    }
    else if (Input.GetKey(KeyCode.S))
    {
        rb.velocity = new Vector2(0, -speed);
        animator.Play("walkD");
    }
    else
    {
        rb.velocity = new Vector2(0, 0);
        animator.Play("idle");
    }
}
}
```

```
}
```

2. Скрипт отвечающий за поднятие персонажем объектов:

```
public class TakeAndDropRight : MonoBehaviour
```

```
{
```

```
    public bool hold;
```

```
    public float distance = 5f;
```

```
    RaycastHit2D hit;
```

```
    public Transform holdPoint;
```

```
    public float throwObject = 5;
```

```
    // Start is called before the first frame update
```

```
    void Start()
```

```
    {
```

```
    }
```

```
    // Update is called once per frame
```

```
    void Update()
```

```
    {
```

```
        if (Input.GetKeyDown(KeyCode.E))
```

```
        {
```

```
            if (!hold)
```

```
            {
```

```
                Physics2D.queriesStartInColliders = false;
```

```
                hit = Physics2D.Raycast(transform.position, Vector2.right * *
```

```
transform.localScale.x, distance);
```

```
                if (hit.collider != null && hit.collider.tag == "throwObject")
```

```

        {
            hold = true;
        }
    }
else
{
    hold = false;

    if (hit.collider.gameObject.GetComponent<Rigidbody2D>() != null)
    {
        hit.collider.gameObject.GetComponent<Rigidbody2D>().velocity = new
Vector2(transform.localScale.x, -0.2f) * throwObject;
    }
}

if (hold)
{
    hit.collider.gameObject.transform.position = holdPoint.position;

    if (holdPoint.position.x > transform.position.x && hold == true)
    {
        hit.collider.gameObject.transform.localScale = new
Vector2(transform.localScale.x, transform.localScale.y);
    }
    else if (holdPoint.position.x < transform.position.x && hold == true)
    {
        hit.collider.gameObject.transform.localScale = new
Vector2(transform.localScale.x * 1f, transform.localScale.y * 1f);

```

```

    }
}

private void OnDrawGizmos()
{
    Gizmos.color = Color.red;
    Gizmos.DrawLine(transform.position, transform.position + Vector3.right *
transform.localScale.x * distance);
}
}

```

3. Скрипт отвечающий за слежение камера за персонажем:

```

public class CameraFollow : MonoBehaviour
{
    private Transform player;

    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player").transform;
    }

    void LateUpdate()
    {
        Vector3 temp = transform.position;
        temp.x = player.position.x;
        temp.y = player.position.y;

        transform.position = temp;
    }
}

```

ПРИЛОЖЕНИЕ Б

Листинг C# кода, отвечающего за взаимодействие игрового мира с персонажем

1. Скрипт отвечающий за сериализацию строк в xml коде:

```
[XmlRoot("dialogue")]
public class Dialogue
{
    [XmlElement("text")]
    public string text;

    [XmlElement("node")]
    public Node[] nodes;

    public static Dialogue Load(TextAsset _xml)
    {
        XmlSerializer serializer = new XmlSerializer(typeof(Dialogue));
        StringReader reader = new StringReader(_xml.text);
        Dialogue dial = serializer.Deserialize(reader) as Dialogue;
        return dial;
    }
}

[System.Serializable]
public class Node
{
    [XmlElement("npctext")]
    public string Npctext;
```

```
[XmlAttribute("answers")]
[XmlElement("answer")]
public Answer[] answers;
}
```

```
public class Answer
{
    [XmlAttribute("tonode")]
    public int nextNode;
    [XmlElement("text")]
    public string text;
    [XmlElement("dialend")]
    public string end;
}
```

2. Скрипт отвечающий за работу диалоговой системы:

```
public class InstantiateDialogue : MonoBehaviour
{
    public GameObject Window;

    public Text text;
    public Text firstAnswer;
    public Text secondAnswer;
    public Text thirdAnswer;
    public Text fourAnswer;
    public Button firstButton;
    public Button secondButton;
    public Button thirdButton;
    public Button fourButton;
```



```

public int index;

bool dialogueEnded = false;

GameObject player;
public TextAsset ta;

[SerializeField]
public int currentNode = 0;
public int butClicked;
bool textSet = false;
Node[] nd;
Dialogue dialogue;

private void Start()
{
    secondButton.enabled = false;
    thirdButton.enabled = false;
    fourButton.enabled = false;
    Window.SetActive(false);
    player = GameObject.Find("Character");
    dialogue = Dialogue.Load(ta);
    nd = dialogue.nodes;

    text.text = nd[currentNode].Npctext;
    firstAnswer.text = nd[currentNode].answers[currentNode].text;

    firstButton.onClick.AddListener(but1);
    secondButton.onClick.AddListener(but2);

```

```
thirdButton.onClick.AddListener(but3);
```

```
fourButton.onClick.AddListener(but4);
```

```
    AnswerClicked(14);
```

```
}
```

```
private void Update()
```

```
{
```

```
    if (Vector3.Distance(player.transform.position, transform.position) < 1.5f &&  
dialogueEnded == false)
```

```
{
```

```
    Window.SetActive(true);
```

```
}
```

```
else
```

```
{
```

```
    Window.SetActive(false);
```

```
}
```

```
}
```

```
private void but1()
```

```
{
```

```
    butClicked = 0;
```

```
    AnswerClicked(0);
```

```
}
```

```
private void but2()
```

```
{
```

```
    butClicked = 1;
```

```
    AnswerClicked(1);
```

```
}
```

```
private void but3()
```

```
{
```

```
    butClicked = 2;
```

```
    AnswerClicked(2);
```

```
}
```

```
private void but4()
```

```
{
```

```
    butClicked = 3;
```

```
    AnswerClicked(3);
```

```
}
```

```
public void AnswerClicked(int numberOfButton)
```

```
{
```

```
    if (numberOfButton == 14)
```

```
        currentNode = 0;
```

```
    else
```

```
    {
```

```
        if (dialogue.nodes[currentNode].answers[numberOfButton].end == "true")
```

```
        {
```

```
            dialogueEnded = true;
```

```
            SceneManager.LoadScene(index);
```

```
        }
```

```
        currentNode
```

```
        dialogue.nodes[currentNode].answers[numberOfButton].nextNode;
```

```
=
```

```

}

text.text = dialogue.nodes[currentNode].Npctext;

firstAnswer.text = dialogue.nodes[currentNode].answers[0].text;
if (dialogue.nodes[currentNode].answers.Length == 4)
{
    secondButton.enabled = true;
    secondAnswer.text = dialogue.nodes[currentNode].answers[1].text;
}
else
{
    secondButton.enabled = false;
    secondAnswer.text = "";
}

if (dialogue.nodes[currentNode].answers.Length == 4)
{
    thirdButton.enabled = true;
    thirdAnswer.text = dialogue.nodes[currentNode].answers[2].text;
}
else
{
    thirdButton.enabled = false;
    thirdAnswer.text = "";
}

if (dialogue.nodes[currentNode].answers.Length == 4)
{

```

```

        fourButton.enabled = true;
        fourAnswer.text = dialogue.nodes[currentNode].answers[3].text;
    }
    else
    {
        fourButton.enabled = false;
        fourAnswer.text = "";
    }

}
}

```

Скрипт отвечающий за работу «дверей»:

```

public class Door : MonoBehaviour
{
    Animator anim;
    Collider2D col;

    // Start is called before the first frame update
    void Start()
    {
        anim = GetComponent<Animator>();
        col = GetComponent<Collider2D>();
    }

    // Update is called once per frame
    void Update()
    {

```

```

    }

    public void Open()
    {
        anim.SetBool("Open", true);
    }

    public void Close()
    {
        anim.SetBool("Open", false);
    }

    public void Enable()
    {
        col.enabled = true;
    }

    public void Disable()
    {
        col.enabled = false;
    }
}

```

3. Скрипт отвечающий работу «нажимной плиты»

```

public class TouchPanel : MonoBehaviour
{

    public Door door;

    // Start is called before the first frame update
    void Start()

```

```

{

}

// Update is called once per frame
void Update()
{

}

public void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.tag == "throwObject")
    {
        door.Open();
        Debug.Log("Дверь открыта");
    }
}

public void OnTriggerExit2D(Collider2D collision)
{
    if (collision.tag == "throwObject")
    {
        door.Close();
        Debug.Log("Дверь закрыта");
    }
}
}

```

4. Скрипт отвечающий за сбор монет:

```

public class CoinsManager : MonoBehaviour
{
    public static int coin;
    Text coinCount;

    void Start()
    {
        coinCount = GetComponent<Text>();
    }

    void Update()
    {
        coinCount.text = coin.ToString();
    }
}

```

5. Скрипт отвечающий за поведение монет в игровом мире:

```

public class Coins : MonoBehaviour
{

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            CoinsManager.coin += 1;
            Destroy(gameObject);
        }
    }
}

```


6. Скрипт отвечающий за сортировку объектов на сцене:

```
public class PositionRenderSorter : MonoBehaviour
{
    [SerializeField]
    private int sortingOrderBase = 5000;
    [SerializeField]
    public float offset = 0;
    [SerializeField]
    private bool isStatic = false;

    private float timer;
    private float timerMax = .1f;
    private Renderer myRenderer;

    private void Awake()
    {
        myRenderer = gameObject.GetComponent<Renderer>();
    }

    private void LateUpdate()
    {
        timer -= Time.deltaTime;
        if (timer <= 0f)
        {
            timer = timerMax;
            myRenderer.sortingOrder = (int)(sortingOrderBase - transform.position.y -
offset);

            if (isStatic)
```

```
        Destroy(this);  
    }  
}  
}
```

7. Скрипт отвечающий за перехват системы событий:

```
public class KostilEventTrigger : MonoBehaviour  
{  
    public UnityEvent triggerEnter;  
  
    private void OnTriggerEnter2D(Collider2D collision)  
    {  
        if (!Application.isPlaying)  
            return;  
        if (triggerEnter != null)  
            triggerEnter.Invoke();  
    }  
}
```

ПРИЛОЖЕНИЕ Г

Листинг XML кода, отвечающего содержанию в диалоговой системе

```
<dialogue>
  <node>
    <npctext> Для прохождения уровня вам необходимо ответить на
загадку </npctext>
    <answers>
      <answer> <text></text> </answer>
      <answer> <text></text> </answer>
      <answer> <text></text> </answer>
    <answer tonode = "1"><text> Продолжить > </text></answer>
  </answers>
</node>
<node>
  <npctext> У ваших родителей шесть сыновей, включая вас, и у
каждого сына по одной сестре. Сколько человек в семье? </npctext>
  <answers>
    <answer tonode = "2"> <text> 12 </text> </answer>
    <answer tonode = "3"> <text> 9 </text> </answer>
    <answer tonode = "2"> <text> 20 </text> </answer>
    <answer tonode = "2"> <text> 14 </text> </answer>
  </answers>
</node>
<node>
  <npctext> Неверно, попробуйте ещё раз </npctext>
  <answers>
    <answer tonode = "1"> <text> Попробовать ещё раз </text>
  </answer>
```

```

        </answers>
    </node>
    <node>
        <npctext> Верный ответ! Двое родителей, шесть сыновей и одна
дочь. Вы можете пройти в следующую комнату для завершения уровня
</npctext>
        <answers>
            <answer><text></text></answer>
            <answer><text></text></answer>
            <answer><text></text></answer>
                <answer tonode = "0"> <text> Продолжить > </text>
                <dialend>true</dialend></answer>
        </answers>
    </node>
</dialogue>
<dialogue>
    <node>
        <npctext> Для прохождения уровня вам необходимо ответить на
загадку </npctext>
        <answers>
            <answer tonode = "1"> <text> Приступим </text> </answer>
            <answer> <text></text> </answer>
        </answers>
    </node>
    <node>
        <npctext> Человек стоит на одном берегу реки, его собака — на
другом. Мужчина зовет свою собаку, которая сразу же пересекает реку, не
промокнув и не используя мост или лодку. Как собака это сделала? </npctext>
        <answers>

```

```

        <answer tonode = "2"> <text> Река высохла. </text> </answer>
        <answer tonode = "2"> <text> Собака зацепилась за ногу птицы
и перелетела реку. </text> </answer>
        <answer tonode = "3"> <text> Река замерзла. </text> </answer>
        <answer tonode = "2"> <text> Собака прокопала ров под рекой.
</text> </answer>
    </answers>
</node>
<node>
    <prctext> Неверно, попробуйте ещё раз </prctext>
    <answers>
        <answer tonode = "1"> <text> Попробовать ещё раз </text>
</answer>
    </answers>
</node>
<node>
    <prctext> Верный ответ! Действие происходило зимой и река
замерзла. Вы можете переходить на следующий уровень </prctext>
    <answers>
        <answer tonode = "0"> <text> Перейти на следующий уровень
</text>
        <dialend>true</dialend></answer>
    </answers>
</node>
</dialogue>

```

ПРИЛОЖЕНИЕ Д

Руководство пользователя

1. Минимальные системные требования:

- Видеокарта – объем памяти от 256 Мб;
- Операционная система – Windows 7/8/10;
- Процессор – 2.2 GHz или выше;
- Оперативная память – 4 гб;
- 100 Мб свободного места на жестком диске.

2. Поставляемый дистрибутив содержит архив с компьютерной игрой в жанре головоломка.

Порядок загрузки данных и проверка работоспособности:

1) Распаковать архив с компьютерной игрой в жанре головоломка на Персональный компьютер;

2) Перейти в каталог Adventures in the maze;

3) Запустить исполняемый файл Adventures in the maze.exe.

Описание пунктов меню:

- Начать игру
- Загрузить уровень
- Выход

Пункт «Начать игру»

После нажатия на кнопку «Начать игру» появится окно с обучением, для его закрытия и начала игры нужно нажать на кнопку «Продолжить»

Пункт «Загрузить уровень»

Используется для загрузки доступных уровней

Пункт «Выход»

Используется для выхода из приложения.