Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки 09.04.04 Программная инженерия
Направленность (профиль) образовательной программы Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«_____»_____ 2023 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Разработка игрового 3D-приложения на платформе Unity

Исполнитель
студент группы 157-ом _____ К.И. Кузеванов
(подпись, дата)

Руководитель
доцент, канд.техн. наук _____ Т. А. Галаган
(подпись, дата)

Руководитель научного
содержания программы
магистратуры
профессор, доктор техн. наук _____ И.Е. Еремин
(подпись, дата)

Нормоконтроль
доцент, канд.техн. наук _____ Л.В. Никифорова
(подпись, дата)

Рецензент _____ О.Г. Какаулин
(подпись, дата)

Благовещенск, 2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
подпись
«_____»_____2023 г.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

К магистерской диссертаии студента группы 157-ом _____

Кузеванова Кирилла Ивановича

1.Тема выпускной квалификационной работы: Разработка игрового
(утверждена приказом 21.02.2023 № 442-уч)
3D-приложения на платформе Unity

2. Срок сдачи студентом законченной работы (проекта): 20.06.2023

3. Исходные данные к магистерской диссертации: предметная область, отчеты по практической подготовке, результаты выступления на научной конференции, свидетельство о регистрации игрового приложения

4. Содержание магистерской диссертации: анализ предметной области, разбор существующих методов процедурной генерации, взаимодействие функциональных модулей, разработка прототипа видеоигры.

5. Перечень материалов приложения: (наличие таблиц, графиков, схем, программных продуктов и т.п.): Техническое задание на создание видеоигры

6. Рецензент магистерской диссертации:_____

7. Дата выдачи задания: 30.01.2023

Руководитель выпускной квалификационной работы: Галаган Т.А. доцент кафедры ИиУС, канд. тех. наук, доцент

Задание принял к исполнению (30.01.2023): _____

РЕФЕРАТ

Магистерская диссертация работа содержит 74с., 53 рисунка, 2 таблицы, 1 приложение, 50 источников.

ПРОГРАММИРОВАНИЕ, UNITY, АРХИТЕКТУРА ВИДЕОИГРЫ, ПРОЦЕДУРНАЯ ГЕНЕРАЦИЯ УРОВНЕЙ, РЕАЛИЗАЦИЯ ПРОТОТИПА ИГРОВОГО ПРИЛОЖЕНИЯ

Целью работы является проектирование и реализация прототипа игрового приложения жанра выживания на платформе Unity, разработка прототипа игрового 3D-приложения с процедурной генерацией окружения.

В данной работе подробно описываются все этапы разработки игрового приложения. Создание прототипа игрового приложения выполняется в несколько этапов с помощью современных технологий, предоставляемых платформой Unity

Задачи выпускной квалификационной работы:

– анализ предметной области в сфере применения процедурной генерации в игровой индустрии;

– определение типовых решений задач, выбор оптимального решения;

– разработка структурного и функционального проекта, определение структур входных и выходных данных;

– обоснование выбора программного обеспечения программное обеспечение необходимого при разработке игрового приложения;

– описание этапов реализации прототипа видеоигры.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Игровая индустрия заслужено заняла свою нишу в сфере развлечений. Уже никто не отрицает того факта, что игровая индустрия имеет такое же влияние, как кинематограф или другие виды развлечений. Также рост популярности видеоигр подтолкнула эпоха COVID-19. Во время пандемии у множества людей появилось большое количество свободного времени дома. И внимание пользователей заняли виртуальные развлечения как видеоигры, а также сервисы стриминговых развлечений такие, как платформа Netflix, которые с энтузиазмом экранизировали игровые серии и, таким образом еще больше повышая популярность компьютерных игр.

С распространением информационных технологий во всех сферах деятельности также появилась новая профессия – специалист по информационным технологиям. И в ближайшем будущем спрос на специалистов в этой области будет только возрастать. Также существует много многообещающих проектов по разработке игр. GameDev ("разработка игр") означает разработку игр, а процесс разработки игровых приложений включает в себя игровой дизайн, программирование на игровом движке или использование готовых решений, а также разработку визуальной концепции и компонентов. Многие крупные компании, такие как Amazon и Apple, извлекают выгоду из потенциала игрового рынка и инвестируют сотни миллионов долларов в игровые проекты. Киноиндустрия включает в себя такие адаптации видеоигр как "смертельную битву" или "Кредо убийцы". Актуальность темы квалификационной работы заключается в разработке видеоигры и внедрение в нее системы игрового окружения игрока методом процедурной генерации.

Процедурная генерация позволяет получить большое количество достаточно разнообразных игровых объектов, которые позволяют снизить нагрузку на разработчиков видеоигр. Скорость работы алгоритмов процедурной генерации

тоже является немаловажным факторам, потому что процесс процедурной генерации часто связан с массивными вычислениями, где оптимизация каждого этапа может существенно увеличить производительность всего процесса. Разработка данной системы может значительно ускорить цикл создания видеоигры, а также снизить стоимость конечного продукта.

В данной работе будут рассмотрены и описаны все этапы разработки прототипа игрового приложения жанра выживания, который в совокупности с другими технологиями платформы Unity можно легко внедрить и в другие проекты, связанные с разработкой видеоигр.

1 АНАЛИЗ И ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Тенденции мирового рынка видеоигр

У игровой индустрии появилось большое влияние на мировой рынок. То, что было невообразимо 30 лет назад, теперь стало реальностью. Странами, которые являются лидерами игрового рынка, являются Китай, Соединенные Штаты Америки и Япония [1]. Игровой рынок вырос на 19% в 2020 году из-за пандемии коронавируса, что составляет 23,1%. Однако в 2021 году прибыль уменьшилась на 1,1 процента и составила 175,8 млрд долларов. По сравнению с прогнозом на 2020 год, причиной снижения рынка, как полагают, является глобальный дефицит поставок полупроводников и стабилизация пандемии, и, несмотря на эти негативные факторы, тенденция роста игрового рынка очень предсказуема, и ожидается, что выручка компаний, связанных с игровым контентом, достигнет в 2023 году 204,7 млрд долларов [1].

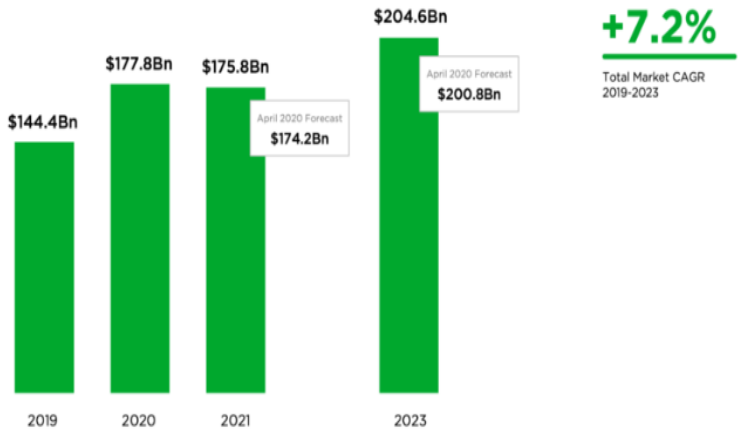

Рисунок 1 – Прибыль мирового рынка видеоигр

В игровой индустрии стратегия распространения игровых приложений также меняется. Эта тенденция особенно заметна в онлайн-играх, которые ориентированы на прибыль от покупок различных виртуальных предметов. Напри-

мер, базовая выручка Take-to-Interactive в четвертом квартале 2020 года составила 760,5 миллионов долларов, что на 41% больше по сравнению с аналогичным периодом прошлого года, большая часть которой основывалась на потребительских расходах на долю которых приходится 54% от общей выручки. Наибольший вклад внесли NBA 2K20, NBA 2K19 и GTA Online. Стоит отметить и то, что такая компания как Epic Games бесплатно распространяет цифровые версии игры, чтобы расширить свою целевую аудиторию, и тратит на это более 115 миллионов долларов. По данным Epic Games, за 10 месяцев у нее появилось 18,5 миллионов новых пользователей, около 130 миллионов человек, 7 процентов из которых приобрели игру. На рынке мобильных устройств самыми скачиваемыми играми являются аркадные игры "Among Us", "Join Clash 3D" и платформер "Subway Surfers"[2]. Также у крупной компании Tencent общий доход подразделения мобильной разработки вырос на 26% в годовом исчислении до 20,4 миллиардов долларов. Из них прибыль от внутриигровых покупок выросла на 1% до 5,598 миллиардов долларов, а игрой с наибольшим количеством активных пользователей в день среди мобильных игр, выпущенных в 2020 году, стала "Call of Duty Mobile" [2]. " Honor of Kings " в настоящее время является самой продаваемой мобильной игрой в мире. Таким образом, согласно тенденциям мирового рынка, политика крупных игровых компаний, заключается в расширении и поддержании активных пользователей, способных оплачивать покупки в играх.

1.2 Применение процедурной генерации в игровой индустрии

При разработке игровых приложений часто применяется алгоритмы процедурной генерируемый контент. Процедурная генерация позволяет иначе взглянуть на дизайн и подчеркивает элегантность основных показателей системы, а также позволяет экономить время. Для создания почвы, деревьев и кустарников разработчики видеоигры «The Witcher 3: Wild Hunt» использовали инструмент – кисти для процедурной растительности [3].

С помощью этого инструмента у разработчиков «ведьмака» появилась возможность распознавать несколько типов поверхностей. Например, система соотносила почву и песок к природным поверхностям, а кирпич и брусчатку к рукотворным. Благодаря этому создатели видеоигры смогли наложить на разные типы материалов различные эффекты. Таким образом были получены плавные переходы у краев дороги – система накладывала материал брусчатки и текстуру земли так, что оставались различимы контуры каждого камня (Рисунок 2).

Рисунок 2 – Видеоигра The Witcher 3: Wild Hunt

При этом в «The Witcher 3: Wild Hunt» созданная с помощью этой системы растительность не перекрывает детали исходной текстуры, а выделяет её.

В видеоигре Skyrim для создания пещер разработчики из компании Bethesda использовали «модульный дизайн» – уровни собирались из так называемых «китов» (Рисунок 3) [3]. Это наборы ассетов (игровых файлов), предназначенных для определённых типов локаций. Из этих ассетов дизайнеры уровней в дальнейшем собирали игровые сцены, как из кубиков конструктора. Идея «кита» заключается в том, чтобы такой набор был чем-то большим, чем просто сумма его частей. Например, художник сделал всего лишь четыре 3D-модели кусочков

трубы –но из них можно собрать бесконечное число разных трубопроводов для бесконечного числа уровней.

Рисунок 3 – Генерация подземелий в видеоигре Skyrim

В видеоигре «Horizon Zero Dawn», разработчики Guerrilla Games тоже использовали процедурную генерацию. Особенность их системы заключалась в достаточно тонкой настройке размещения игровых объектов. (Рисунок 4).

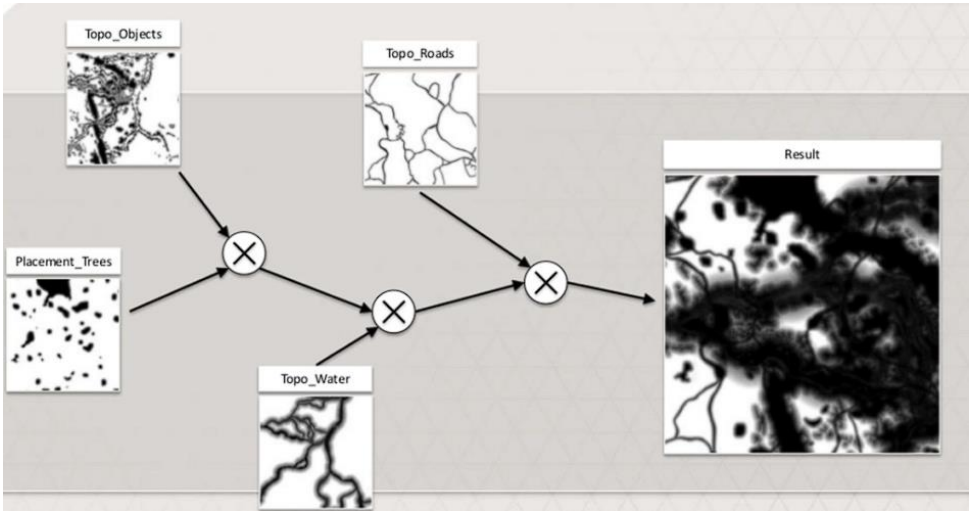

Рисунок 4 – Генерация игрового уровня в видеоигре Horizon Zero Dawn

Генератор разделял игровой уровень на слои. Главным слоем служила карта растительности, на которой разработчики отмечали, где должна располагаться растительность и степень ее насыщенности [3]. После наложения этой

карты на остальную местность, генератор получал возможность сравнивать значения в определенных местах. Благодаря этому даже если через очень густой лес проходила дорога или река, система сама изменяла параметры так, чтобы под водой не росла трава, а кусты плавно исчезали на подходах к тропинке. После настройки параметров генератора, разработчики легко могли создавать определенные биомы на игровом уровне.

Предыдущие описанный примеры применялись разработчиками для создания частичной генерации контента. Также разработчики применяют и полную генерацию мира. Minecraft, самая продаваемая игра в мире, наиболее известная своими пикселизированными блоками и бесконечными мирами, содержит потрясающий процедурный генератор ландшафта — с пещерами, водоёмами, и даже различными биомами (Рисунок 5).

Рисунок 5 – Видеоигра Minecraft

Также стоит упоминания No Man’s Sky – это видеоигра про исследование космоса, в которой используется технология процедурной генерации игрового окружения и ресурсов (текстур, моделей, рельефа и т.д). Видеоигра позволяет игрокам исследовать бесконечные просторы космоса. Игроки могут взять на себя

роль путешественника-космонавта, которому предстоит изучать различные планеты, жизненные формы и галактики. Приступая к игровой сессии, пользователь попадает в случайную точку галактики, где ему предстоит исследовать неизведанные миры и планеты. В игре есть возможность создавать и улучшать свой космический корабль, снаряжение и оружие для защиты от опасных созданий и природных явлений.

Принцип процедурной генерации No Man’s Sky построен на работе с определенным набором ресурсов и с помощью процедурной генерации создавать на их основе тысячи вариаций разных планет с разными живыми существами (Рисунок 6).

Рисунок 6 – Генерация планет в видеоигре No Man’s Sky

При этом на каждой планете создаются уникальные биомы, которые не повторяются. Таким образом, рассмотрев примеры использования процедурной генерации в игровой индустрии, можно сделать вывод о актуальности ее использования как инструмента для упрощения процесса разработки видеоигр.

1.3 Обзор существующих методов решения задачи

Перед тем как переходить к созданию системы генерации игровых уровней, нужно рассмотреть существующие методы решения данной задачи. В своей

основе игровая локация должна включать несколько типов игровых объектов такие как ландшафт, растительность и водную поверхность. Такой минимальный набор начальных условий для генератора позволяет создать оптимальный и незатратный по производству игровой проект.

Из наиболее используемых алгоритмов генерации выделяют три группы методов, с помощью которых можно построить игровой уровень:

– генерация карт высот,

– процедурный шум,

– генерация плиточной структуры.

В основе генерации карты высот лежит двухмерный массив, элементами которого являются значения высот для определенных точек ландшафта. С помощью таких алгоритмов оптимально строить 3D-локации. Одни из основных представителей данной группы являются алгоритм фрактальной генерации и алгоритм ромб-квадрат (Diamond-Square) [3].

Преимуществами группы данных алгоритмов является их простота реализации, скорость работы, для их выполнения не требуются больших вычислительных мощностей, легко масштабировать. Недостатком является проблема сглаживания, которую можно решить, усовершенствовав алгоритм.

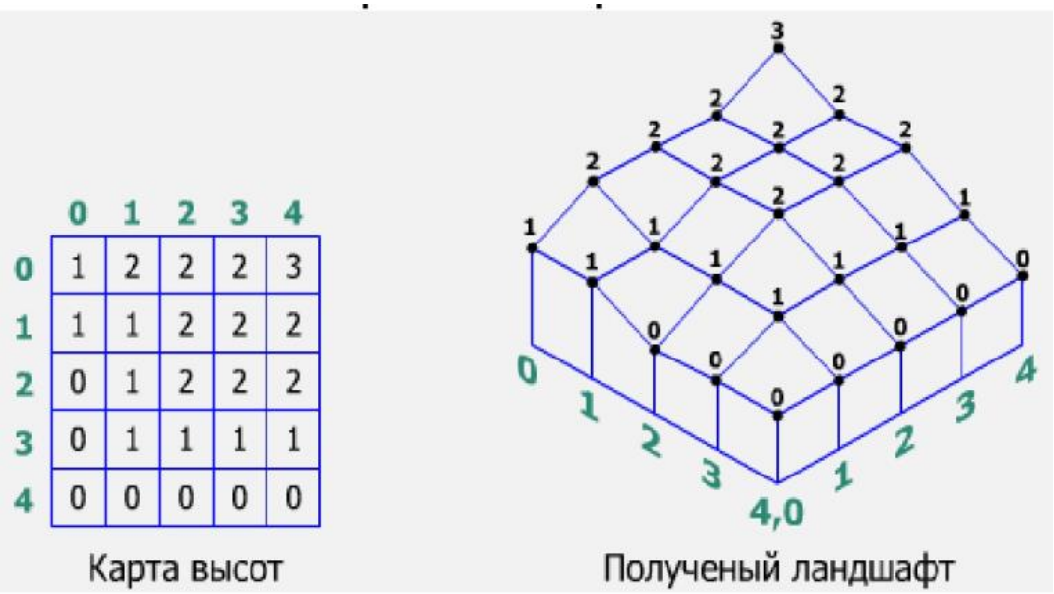

Рисунок 7 – Карта высот

Понятие «шум» подразумевает собой набор случайных чисел из заданного промежутка значений, без закономерностей и корреляции между двумя соседними числами. Элементарный пример шума – изображение, построенное путем присвоения каждому пикселю случайным образом значения 0 (черный цвет) или 1 (белый цвет) (Рисунок 8).

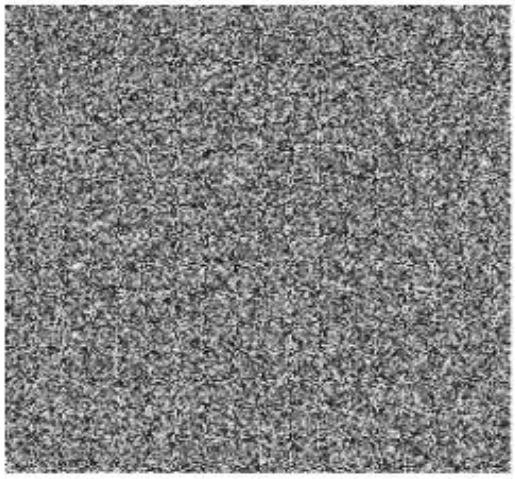

Рисунок 8 – Изображение шума

Процедурный шум создается посредством программного кода. Например, при генерации двухмерной текстуры методом процедурного шума, цвет каждого пикселя изображения вычисляется при помощи алгоритмов и математических функций. Наиболее известным из таких алгоритмов является шум Перлина (Рисунок 9).

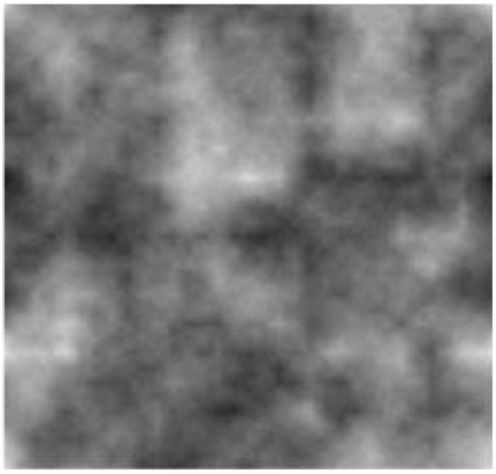

Рисунок 9 – Шум Перлина

Алгоритмы шума имеют широкое применение в области компьютерной графики и чаще всего используются при генерации текстур и создании сложных графических объектов, таких как огонь, облака, туман. Также, шум Перлина применяется при генерации карт местности. К преимуществам алгоритма также можно отнести относительную простоту реализации, масштабируемость. Недостатком у данного алгоритма является то, что подобные изображения крайне трудно параметризировать. Также существенным недостатком шумов, является тот факт, что при частом использовании функций случайных чисел, становится трудно контролировать окончательный результат, а также не самая быстрая скорость.

Третья группа методов. Под мозаикой Вороного конечного множества S понимается разбиение плоскости, при котором каждая область этого разбиения образует множество точек, более близких к одному из элементов множества S, чем к любому другому элементу множества (Рисунок 10).

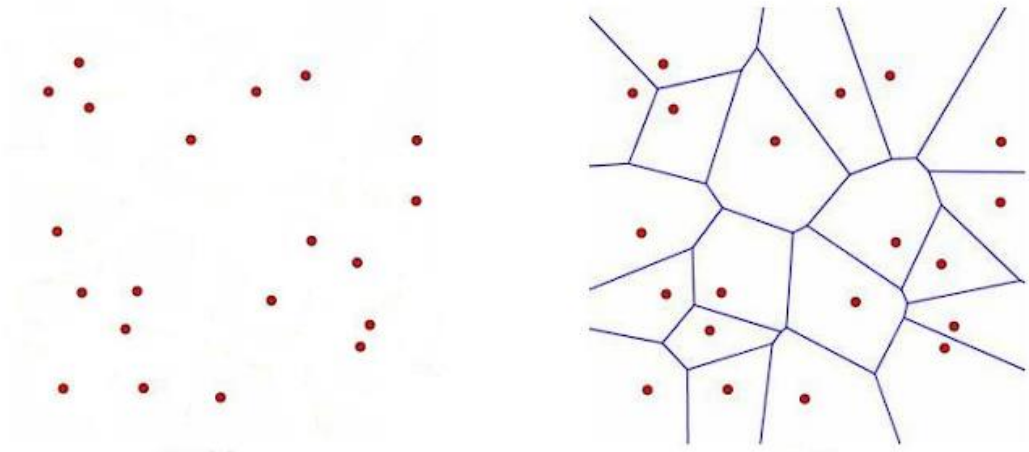

Рисунок 10 – Мозаика Вороного

Пусть на всей плоскости задано некое ограниченное и не содержащее одинаковых элементов множество точек (узлов). Порождаемая этим множеством мозаика состоит из полигонов (или плиток), соответствующих узлам и определяемых по формуле 1.

$$V_i = \{x:\ d(x, C_i) < d(x, C_m) \quad | \quad \forall\, m \neq i\} \tag{1}$$

где: $\mathrm{d}(x, C_i), \mathrm{d}(x, C_m)$ – эвклидово расстояние между точкой x на плоскости и точкой множества C_s.

Другими словами, к данному полигону мозаики относятся все точки плоскости, расстояние которых до соответствующего полигону узла меньше расстояния до любого другого узла. Преимуществом данной группы алгоритмов является то, что легко параметризировать получившееся изображение [3]. Существенными недостатками группы данных алгоритмов является ее сложность реализации, проблема масштабирования, а также необходимость в высоких вычислительные мощности из-за специфики разбиения сеток. В таблице 1 представлено сравнение результатов перечисленных групп алгоритмов генерации.

Таблица 1 – Сравнение алгоритмов по параметрам

Алгоритм/параметр	Генерация карты высот	Процедурный шум	Генерация плиточной структуры
Масштабируемость	Масштабирование требует модификации алгоритма	Масштабирование не требует модификации алгоритма	Трудно масштабировать
Сложность реализации	Легко	легко	Сложно
Возможность модификации	Легко	Легко	Сложно
Требования к вычислительным мощностям	низкие	низкие	высокие
Гибкость параметров результата генерации	Средняя	низкая	высокая
Скорость работы	быстрая	быстрая	медленная

Таким образом исходя из сравнения групп алгоритмов по параметрам для выполнения задач выпускной квалификационной работы оптимальны группы алгоритмов генерации карт высот или процедурный шум.

1.4 Описание этапов разработки видеоигры

Процесс разработки игры является трудоемкий творческим процессом. Он не является жестко регламентированным. Однако существует общий подход, упрощающий управление жизненным циклом проекта. Обычно он включает следующие этапы:

– проектирование;

– разработка;

– тестирование;

– реализация и поддержка.

На этапе проектирования определяются цель игры и средства её разработки (рисунок 11).

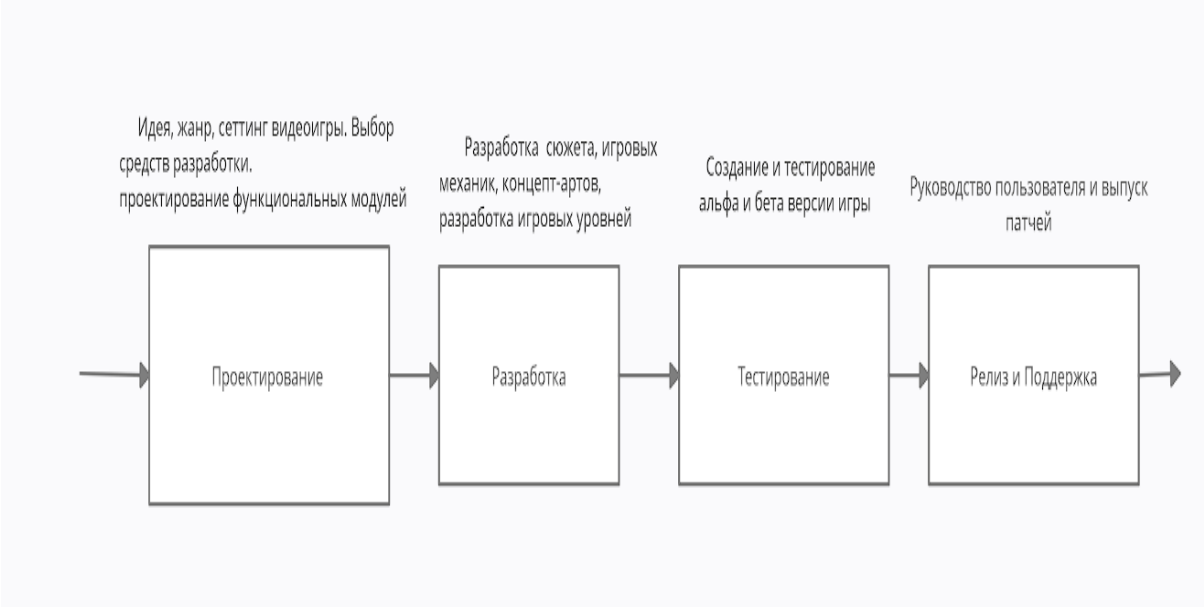

Рисунок 11 – Этапы разработки видеоигры

Идея игрового приложения тесно связана с жанром, поскольку оно побуждает пользователей играть в разработанные видеоигры. Например, суть ролевой игры заключается в том, что "игрок может создавать и воплощать персонажа в соответствии со своими представлениями", в то время как суть игры экшена заключается в том, что "игрок может участвовать в реальных или воображаемых военных действиях". Так что жанр определяется практически сразу после определения сути игры.

Следующий шагом по проектированию видеоигры – выбор сеттинга. Сеттинг – это среда, в которой разворачивается основной сюжет игры, и она определяет местоположение и время действия. Поскольку выбор сеттинга может как усложнить, так и ускорить разработку сценария игрового приложения, то его следует обговаривать на ранней стадии создания игры и при этом учитывать вкусы целевой группы пользователей.

Выбор средств разработки также является важным шагом. От выбора игрового движка и языка программирования зависит скорость создания игрового приложения и его работоспособность. Язык программирования во многом определяется платформой, на которой разрабатывается игра.

Игровой движок отвечает за низкоуровневое описание логики видеоигры (физика, рендеринг и т.д.) [4]. При выборе движка в первую очередь важны условия его использования и язык программирования, на котором пишут код разработчики. Как только цель игры и средства производства определены, начинается вторая фаза проекта – разработка.

Разработка является наиболее амбициозным и трудоемким этапом проекта, и для создания жизнеспособного продукта требуется выполнить несколько шагов. На первом шаге определяется сюжет и основные игровые механики. Механики видеоигры определяют объекты и правила, с помощью которых пользователь будет взаимодействовать с игровым приложением [4].

Затем идет разработка игровых уровней. При разработке игровых уровней изначально создаётся его упрощённый план, на котором схематично изображён сам уровень, а также изображены предметы, с которыми будет впоследствии взаимодействовать игрок.

В дальнейшем создаётся первая версия уровня с минимумом необходимых для прохождения предметов. Эта версия уровня служит для его тестирования на проходимость. После теста уровень начинают постепенно заполняют объектами.

Вскоре после создания первых уровней идет сборка первого прототипа игры, который называют альфа-версией игры. Альфа-версия необходима для того, чтобы разработчик мог провести тестирование основной механики игры, а также проверить, насколько она соответствует заявленным требованиям. Часто в альфа-версии игры у объектов даже нет текстур или они вообще представлены в виде абстрактных объектов.

Если альфа-версия игры успешно проходит тестирование, наступает следующий этап разработки – проработка механик и объектов игры. На этом этапе

завершается разработка уровней и игровых механик, добавляются первые сюжетные события, такие как видеоклипы и сюжетные разговоры, а также исправляются ошибки в игровом коде, обнаруженные во время альфа-тестирования [4].

После этого наступает этап создания второго прототипа игры – бета-версии. С помощью бета-версии игру тестируют на неисправности. Фактически, она представляет собой предфинальную версию игры. В ней могут отсутствовать незначительные элементы, которые не влияют на игровой процесс. При тестировании бета-версии проверяются все игровые механики [4].

После прохождения бета-тестирования игра отправляется на окончательную доработку и исправление критических ошибок, после чего окончательная версия игры выпускается в продажу. После выпуска оказывается дополнительная поддержка путем выпуска исправлений (файлов для устранения неполадок, включенных в конечный продукт). Чтобы продлить срок службы игры, разработчики выпускают DLC (загружаемый контент), в который добавляют различные предметы и функции для пользователей.

2 АЛГОРИТМИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕШЕНИЯ ЗАДАЧИ

2.1 Предлагаемый алгоритм решения задачи

Алгоритм решения задач выпускного квалификационного исследования состоит из описания функционала видеоигры, его программных модулей и архитектуры. А также самого алгоритма, который будет использован для генерации игрового окружения игрока.

2.1.1 Описание функционала программного обеспечения

В разрабатываемой видеоигре от первого лица игроку предстоит исследовать и выживать в окрестностях острова. В игровом приложении предполагается реализовать следующие функции:

– создание персонажа «авантюриста» и добавление ему 3D-модели, реализация его анимации передвижения и взаимодействия, отслеживание уровня здоровья (health bar), взаимодействие с другими объектами с помощью Rigitbody3D;

– генерация игровых уровней острова и его окрестностей;

– расстановка источников освещения;

– реализация нескольких типов осколков, разбросанных по игровой карте. (ледяной, огненный и пустынный). От типа осколка, вставленного в алтарь, зависит в какой именно игровой уровень перейдет персонаж;

– реализация системы алтарей, необходимых для перемещения между игровыми уровнями;

– генерация подбираемых предметов (сферы здоровья, оружие, артефакты), которые необходимы для выживания игрока;

– отрисовка пользовательского интерфейса видеоигры, который должен обеспечивать передачу игроку основной информации о персонаже;

– создание меню игрового приложения, из которого можно запустить игру, изменять настройки, посмотреть раздел помощи или выйти из нее;

– реализация меню паузы, для остановки игры. С возможностью продолжения;

– создание звуковых эффектов игрового приложения.

Подробнее изложение основных составляющих видеоигры описано в техническом задании приложения А.

2.1.2 Описание функциональных модулей видеоигры

Для того чтобы приступить к разработке видеоигры необходимо выполнить проект концептуальной структуры видеоигры. Его можно представить в виде функциональной модели. Функциональная модель представляет собой абстрактное описание, не зависящее от принципов ее физической реализации. Она отображает взаимосвязь функций, выполняемых в сети, которые в данном случае рассматриваются в качестве элементов модели. На входе в видеоигру поступают действия пользователя и данные, которые он вводит (Рисунок 12).

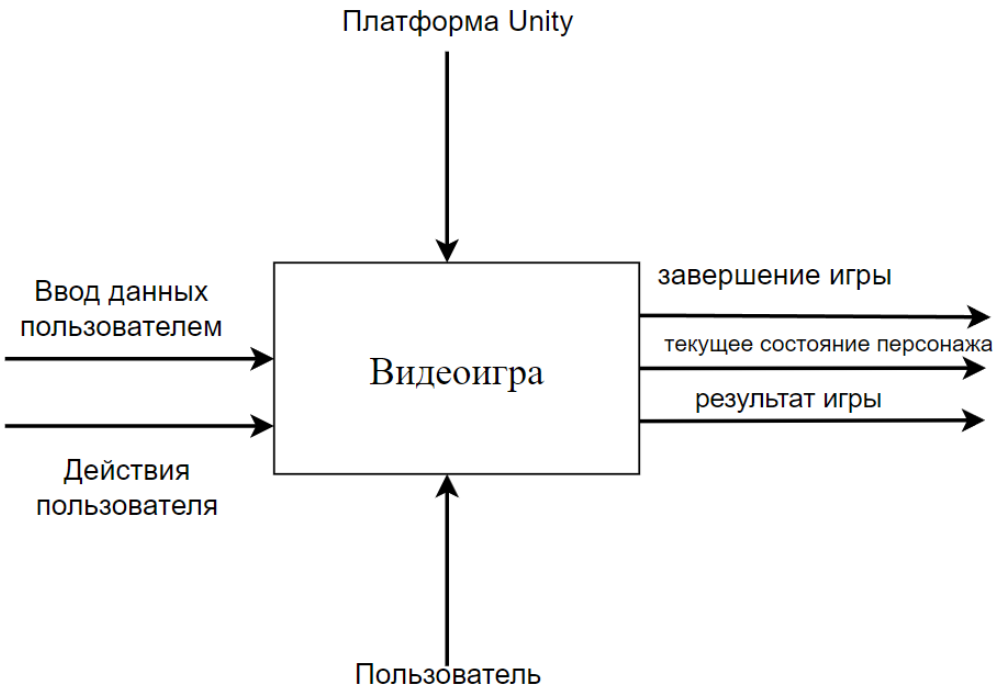

Рисунок 12 – Диаграмма взаимодействия видеоигры

На выходе выводятся в пользовательском интерфейсе игроку выводится вся необходимая ему информация в визуальной форме и результат в конце игровой сессии. Также необходимо провести декомпозицию блока видеоигры на отдельные модули, отвечающие за определенные задачи видеоигры. В результате

были спроектированы функциональные модули игрового приложения (Рисунок 13).

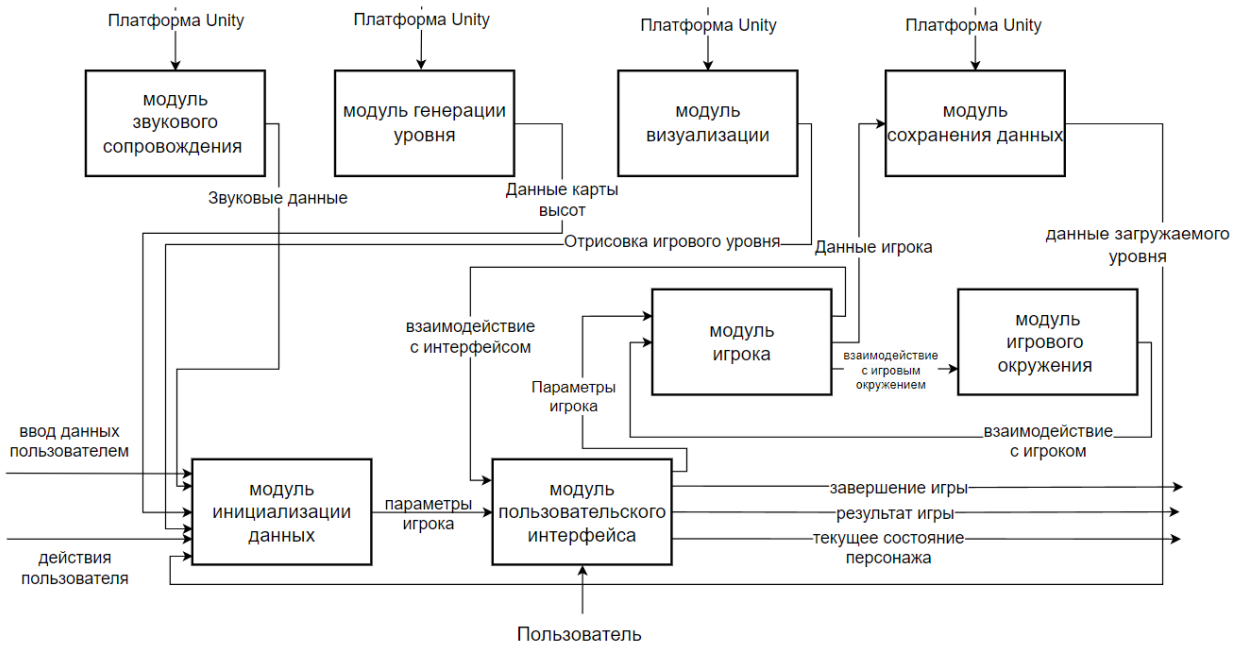

Рисунок 13 – Схема декомпозиции видеоигры

Функциональные модули включают:

– модуль инициализации данных отвечает за инициализацию данных о игроке, игровых карте;

– модуль сохранения служит для передачи данных игрока между игровыми уровнями;

– модуль игрока непосредственно управляется игроком;

– модуль игрового окружения включает в себя все игровые объекты (сундуки, противник, ловушки, подбираемые предметы и т.д.);

– модуль визуализации служит для отрисовки игрового уровня и рендеринга освещения, а также за управление различными графическими шейдерами.

– модуль звукового сопровождения, который необходим для передачи всех звуковых данных;

– модуль генерации уровня, который служит для генерации игрового уровня.

– модуль пользовательского интерфейса выводит всю необходимую информацию игроку.

2.1.3 Архитектурный проект видеоигры

Архитектурный проект игрового приложения представлен несколькими UML-диаграммами графически отображающих компоненты системы. На рисунках 14 изображена диаграмма вариантов использования главного меню.

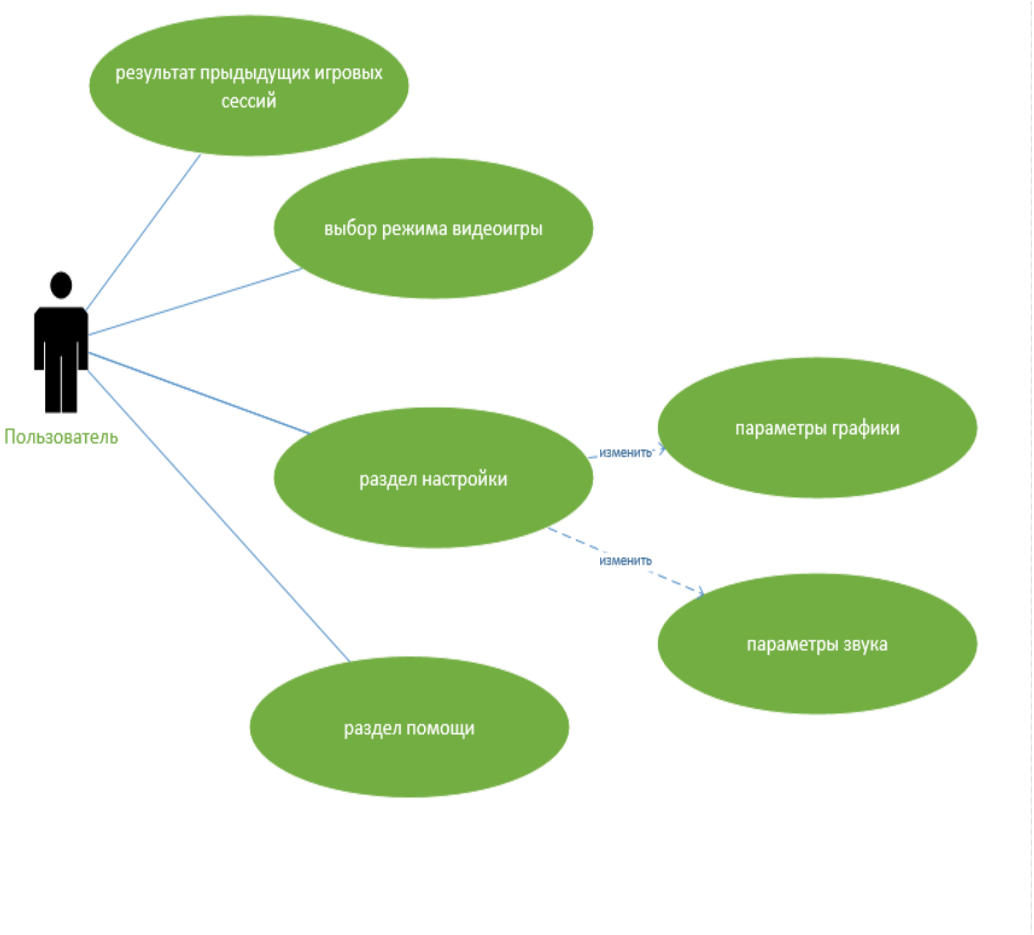

Рисунок 14 – Диаграмма прецедентов главного меню

Диаграмма главного меню представляет собой способы использования функционала игрового для обычного пользователя [5]. Например, пользователь может изменять настройки графики такие как качество текстур, ambient occlusion, контролировать качество теней. Игрок может менять громкость звуковых эффектов видеоигры, получить справочную информацию (цели видеоигры и правила управления игровым персонажем и другая полезная информация).

Также пользователь может выбирать режим игры и смотреть результаты предыдущих сессий. При начале и выходе из игры пользователю выведется визуальное окно, которое попросит подтвердить действие пользователя. Диаграмма прецедентов, представленная на рисунке 15 описывает возможности пользователя непосредственно в игровой сессии.

Рисунок 15 – Диаграмма прецедентов игровой сессии

Представленные диаграммы в полной мере описывает возможности использования игрового приложения пользователем. Диаграмма последовательности описывает логику сценариев использования системы пользователем во времени [5]. Схема последовательности построена таким образом, что она представляет собой временную шкалу, которая начинается сверху и постепенно опускается, чтобы отметить последовательность взаимодействий. Каждый объект имеет колонку, а сообщения, которыми обмениваются между собой, представ-

лены стрелками. Актёр (пользователь) изображён, как и на UML-диаграмме вариантов использования. Пользователь может из главного меню пройти в игровую сессию. Из игровой сессии пользователь заходит в меню паузы, а также переходит на следующий уровень выполняя определенные условия такие как сбор артефактов и активация алтаря (Рисунок 16).

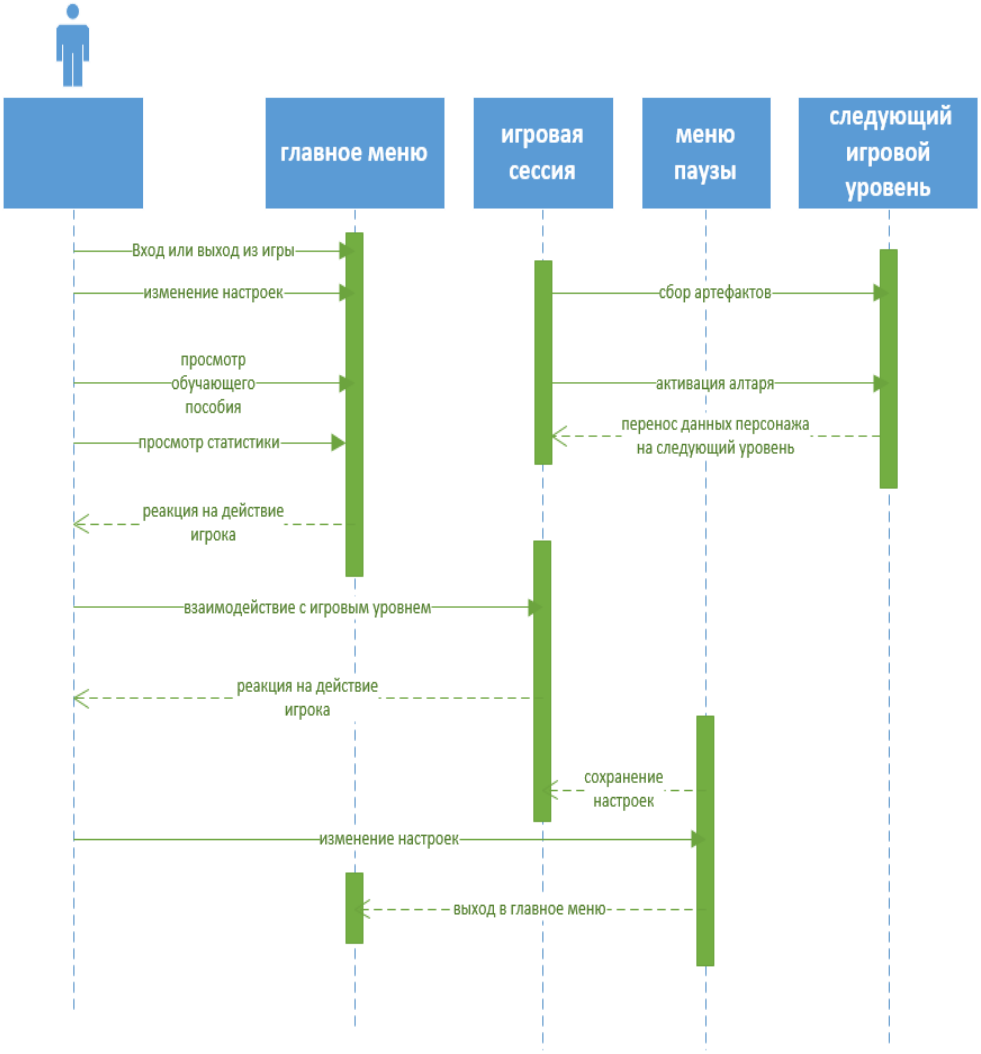

Рисунок 16 – Диаграмма последовательности

Таким образом из представленной диаграммы видно, каким образом и в какой последовательности может взаимодействовать с игровым приложением пользователь. Также в архитектурном проекте игрового приложения представлены диаграммы деятельности и состояний. Диаграмма деятельности – один из

доступных видов диаграмм, поддерживаемых Flexberry Designer. Она, как и диаграмма состояний, отражает динамические аспекты поведения системы. По существу, эта диаграмма позволяет более детально визуализировать конкретный случай использования. Это поведенческая диаграмма, которая иллюстрирует поток деятельности через систему [5].

Представленная на рисунке 17, диаграмма деятельности отображает поведение видеоигры при переходе на следующий игровой уровень.

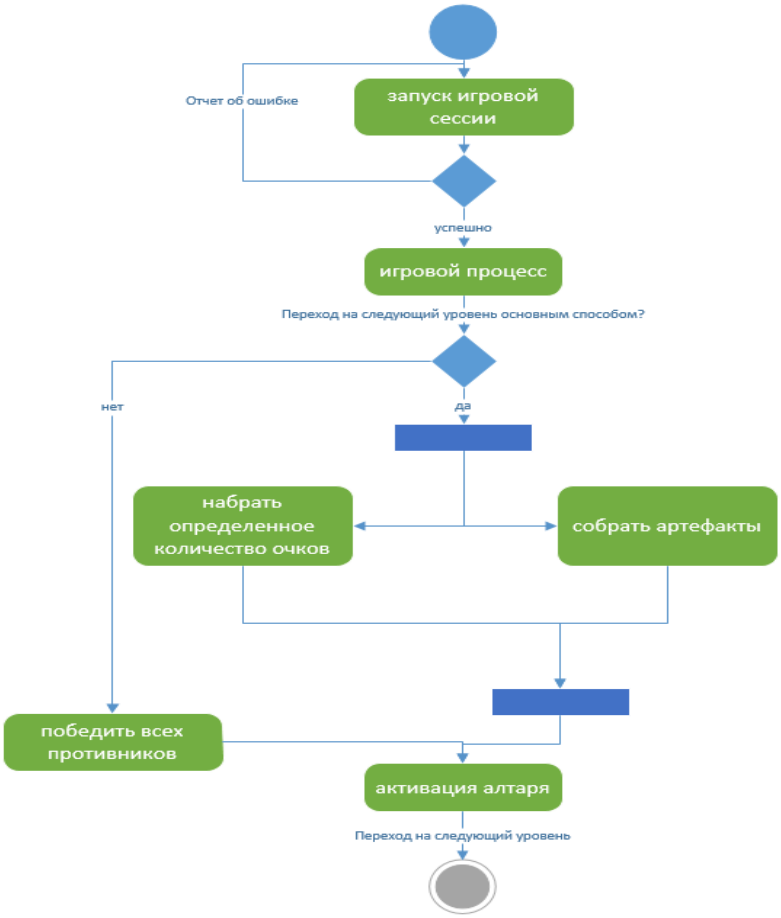

Рисунок 17 – Диаграмма деятельности

Диаграмма состояний, изображённая на рисунке 18 отображает поведение объектов разрабатываемого игрового приложения в динамике, а также связь состояний объектов с событиями. Главное предназначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей,

на основе спецификации их реакции на восприятие некоторых конкретных событий [5]. Системы, которые реагируют на внешние действия от других систем или от пользователей, иногда называют реактивными. Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Хотя диаграммы состояний чаще всего используются для описания поведения отдельных экземпляров классов (объектов), но они также могут быть применены для спецификации функциональности других компонентов моделей.

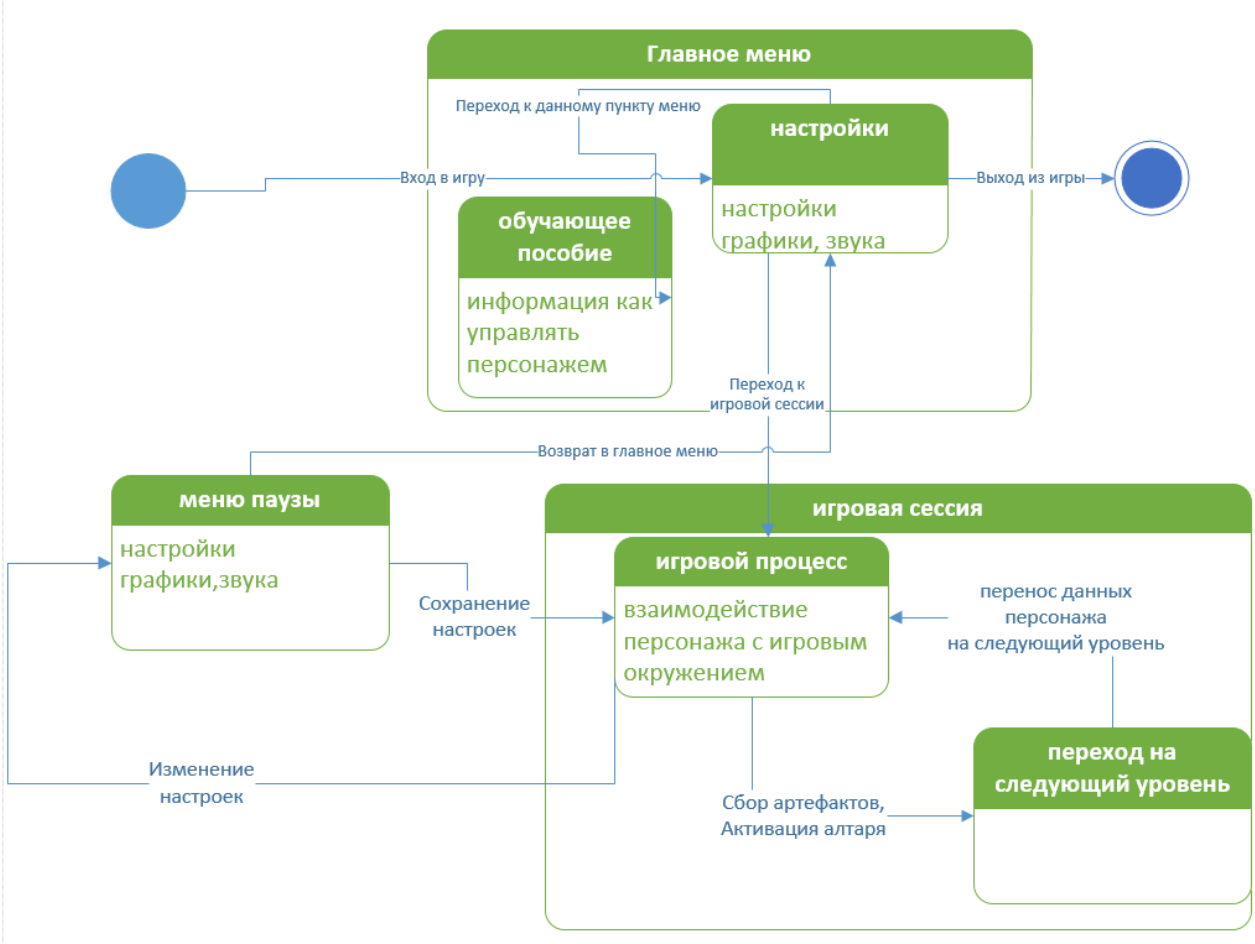

Рисунок 18 – Диаграмма состояний видеоигры

Диаграмма компонентов используются для визуализации организации компонентов видеоигры и зависимостей между ними. Она позволяют получить высокоуровневое представление о компонентах игрового приложения. Компонентами могут быть программные компоненты, такие как база данных или поль-

зовательский интерфейс; или аппаратные компоненты, такие как схема, микросхема или устройств [5]. Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. На рисунке 19 представлена диаграмма компонентов разрабатываемой видеоигры.

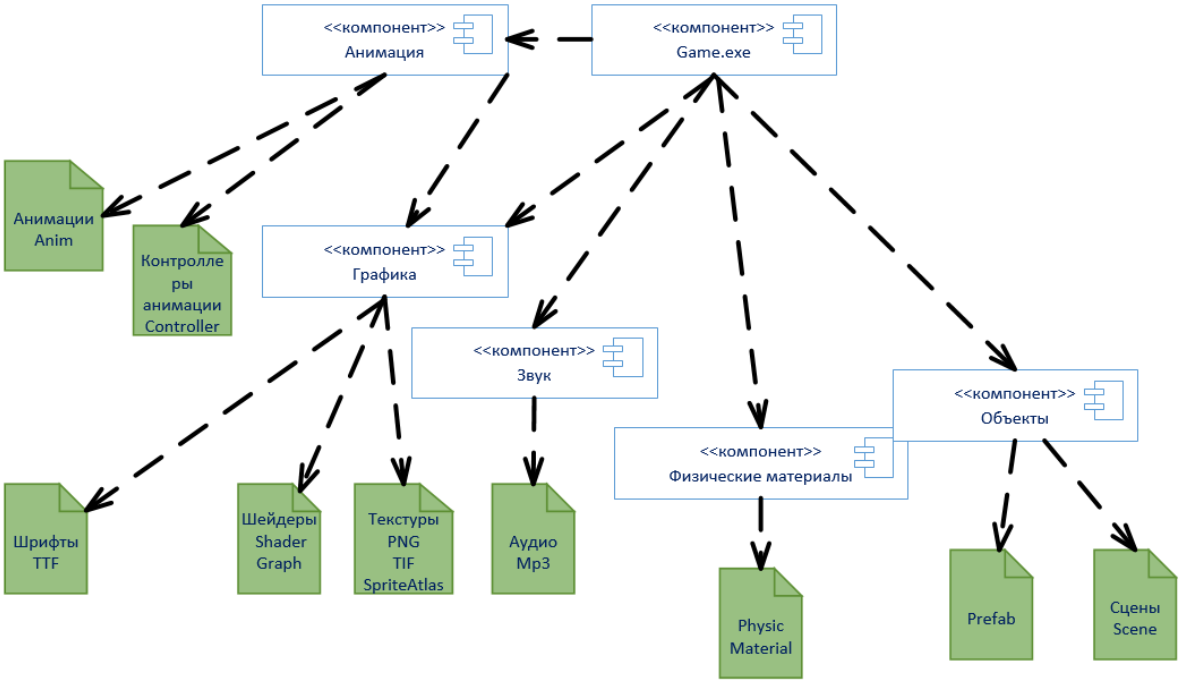

Рисунок 19 – Диаграмма компонентов

У игрового приложения главным является компонент Game от которого зависят другие компоненты игрового приложения. Компонент анимация включает в себя объекты anim и контроллеры анимации. Компонент графика включает в сель элементы – шрифты, текстуры, шейдеры. Компонент звук содержит звуковые файлы видеоигры. Компонент физические материалы включает в себя файлы материалов моделей игрового приложения. Компонент объекты содержит элементы prefab и сцены.

Таким образом архитектурный проект создаваемого игрового приложения был полностью описан.

2.1.4 Описание алгоритма для генерации игровых уровней

Алгоритм Diamond-Square позволяет легко создавать разнообразные ландшафты. Есть возможность сложить или умножить эти значения рельефа на другие коэффициенты. Например, можно возвести нормализованные значения в квадрат, чтобы сделать равнину более плоской, а горный склон более крутым. Алгоритм ромбического квадрата, который генерирует реальный "фрактальный ландшафт", состоит из двух шагов, в отличие от двумерного перемещения средней точки (Рисунок 20).

На первом шаге "квадрат" получается среднее значение углов и аналогичным образом добавляется точный сдвиг (случайный сдвиг), чтобы определить центральную точку квадрата.

На шаге номер два, "Ромб", предназначен для определения высоты точки в центре стороны. Здесь усредняются не только "верхние" и "нижние" 2 точки, но также "левые" и "правые", то есть еще 2 центральные точки, полученные на шаге "квадрат". Следовательно, сначала нужно сделать шаг "квадрат" для каждого квадрата, затем шаг "ромб" для каждого ромба, а затем "слой" с меньшим квадратом.

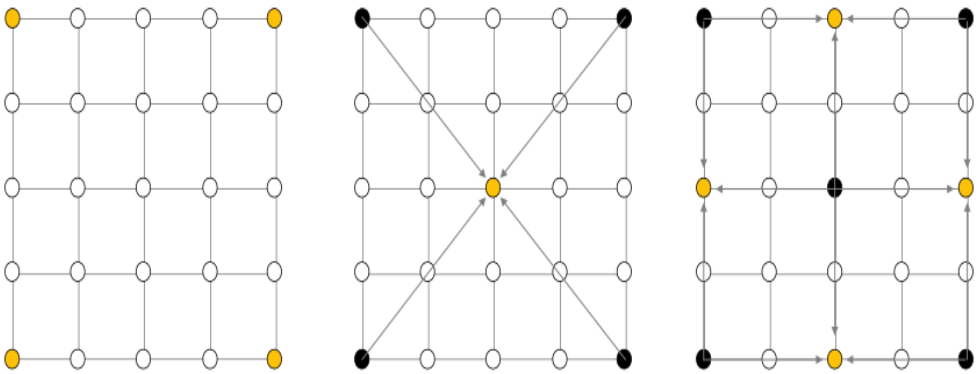

Рисунок 20 – Алгоритм diamond-square

В дополнение к необходимости использовать ширину обхода, а не глубину, есть еще одна тонкость. Это крайняя ситуация в ландшафте. Дело в том, что на

этапе "ромб" алгоритм использует высоту точки, которая находится за пределами текущей сетки или, в некоторых случаях, за пределами всей карты. Есть два решения этой проблемы, можно установить высоту равной 0 (или 1, или любой другой константе). Однако, если нужно определить высоту точки из 64 пикселей на левом краю карты, свернув плоскость в тор (дискообразную планету), можно представить, что получаете высоту точки из 64 точек на правом краю. Это очень легко реализовать (аналогично первому варианту). Необходимо умножить координату по модулю так, чтобы она равнялась размеру карты. Алгоритм diamond-square дает наиболее реалистичные результаты, но он не лишен некоторых недостатков. Например, даже если создать великолепно выглядящую карту при большом увеличении, можно увидеть множество маленьких островов (или шумных отправных точек), а не большие континенты и океаны. Самоподобия не выходит. Эту проблему можно исправить, используя комбинацию фрактальных ландшафтов различных масштабов. Можно умножать значения, складывать их, использовать разные коэффициенты и вводить их в разные алгоритмы.

Алгоритм ромбического квадрата также может быть модифицирован для соответствия картам большего размера. Используя стандартную реализацию этого алгоритма, можно легко увеличить детализацию (переместить "внутрь"), но при этом не увеличить размер (переместить "наружу").

Этот вопрос решается следующим образом: то есть, во-первых, представить ландшафт в больших размерах. Важно то, что не нужно знать высоту каждой точки, вместо этого с есть "окно" небольшого размера (например, 128x128 пикселей), и его можно перемещать по карте высот по мере необходимости.

2.2 Обоснование выбора программно-технического обеспечения

Выбор среды для разработки игр является важным фактором, который влияет на спецификацию разработки и скорость реализации проекта. Существуют различные продукты для разработки программного обеспечения для видеоигр. Они классифицируются следующим образом:

– общие среды для разработки. Unity и Unreal Engine 4 в настоящее время являются наиболее распространенными платформами для разработки работают над этим. Они подходят для разработки любого типа игрового приложения;

– среды разработки, разработанные для конкретных случаев использования. Примерами таких движков являются Frostbite, CryEngine, Creation Engine.

Unreal Engine 4 Является бесплатной программой с открытым исходным кодом. Выпущена 19 марта 2014 года компанией Epic Games. Рабочая область Unreal Engine4 включает в себя следующие элементы:

– панель обозревателя контента: отображает ресурсы проекта для создания всех игровых элементов, включая чертежи и сценарии;

– окно модификации : Содержит множество объектов, упорядоченных по категориям, таких как "Вспышка" и "Куб", а также различные инструменты, такие как раскраска объектов, создание и редактирование пейзажей;

– окно «ViewPort», отображающее текущий уровень проекта, является основной сценой, на которой размещаются игровые объекты, которые можно разместить, перетащив их из браузера содержимого или окна мода;

– окно «World Outliner», которое отображает объекты, созданные на игровой сцене;

– панель инструментов со всеми основными инструментами, включая кнопку "Играть", необходимую для начала игровой сессии.

Преимущества Unreal Engine 4:

– среда разработки имеет встроенный язык программирования C++, который позволяет работать с реестрами и сторонними файлами;

– для программирования сценариев можно использовать C++, а также в платформу интегрирован язык визуального программирования BluePrint для упрощения разработки игр;

– возможность создавать игровые проекты с использованием фотореалистичной графики;

– никаких специальных знаний не требуется для настройки графических компонентов проекта.

Плохие стороны Unreal Engine 4:

– изучение C++ требует обширных навыков в объектно-ориентированном программирования;

– если игра разрабатывается на движке Unreal Engine 4 и ее доход составляет более 3000 долларов в квартал, вы должны будете уплатить лицензионный сбор в размере 5% от своего дохода.

Unity – это инструмент для разработки 2D и 3D-игр и одна из самых популярных систем разработки на сегодняшний день. Есть возможность создавать приложения, которые работают в большинстве современных операционных систем (Windows, OS X, Android, Apple iOS), а также на игровых консолях, таких как PlayStation 4 и Nintendo Switch [6].

Интерфейс Unity состоит из следующих элементов:

– окно сцены –это окно содержит все игровые объекты, которые видит пользователь.

– игровое окно – это окно используется для тестирования игры;

– окно инспектора – панель свойств игрового объекта;

– окно иерархии со всеми игровыми объектами;

– панель инструментов: различные инструменты, например, для перемещения и поворота объектов;

– вкладка "Проекты" для просмотра файловой системы игры.

Все окна и панели в пользовательском интерфейсе Unity можно перетаскивать по своему усмотрению. Преимущества Unity:

– сценарии для игровых проектов разрабатываются на C#, самом простом языке программирования для начинающих разработчиков видеоигр.

– кроссплатформенная поддержка означает, что можно использовать не только место работы (ПК, мобильное устройство, консоль и т. Д.), но и инструменты разработки (встроенная среда может использоваться как для Windows, так и для Mac OS) [6].

– имеется обширный учебный материал и обширная документация.

– низкий порог для пользователей.

– доступ к встроенному "ресурсному магазину", где можно приобрести различные материалы для своего игрового проекта.

– модульная система компонентов Unity используется для создания игровых объектов, состоящих из комбинации функциональных элементов. В отличие от наследования, объекты Unity создаются путем объединения функциональных блоков, а не размещения их на узлах в дереве наследования [6]. Такой подход упрощает создание важных прототипов при разработке игр. Недостатки:

– очень сложно оптимизировать крупные проекты;

– создание сложной графики в Unity возможно, но требует больше усилий по настройке, чем Unreal Engine 4;

– визуальный редактор ограничен при работе с многокомпонентными схемами, что затрудняет визуальную работу в сложных сценах.

Сотни игр, приложений и симуляторов были написаны на Unity, и Unity используется как крупными разработчиками (например, Blizzard), так и независимыми создателями игр.

CryEngine – это игровой движок, разработанный немецкой компанией Crytek в 2002 году и лицензированный другими компаниями. С 30.3.2006 все права на движок принадлежат Ubisoft. Движок обладает расширенными возможностями разработки видеоигр и поддерживает передовые технологии, такие как DirectX 12, Vulcan API, VR, сценарии на C#, пиксельное освещение в реальном времени, карты отражения, детализированные текстуры, туман, блики, реалистичная физика и продвинутая анимация.

CryEngine способен создавать игры с почти фотореалистичной графикой, и при наличии надлежащих навыков проекты, разработанные с использованием CryEngine, превосходят по качеству игры, разработанные с использованием Unreal Engine 4 и Unity. Компания Crytek разработала собственную технологию трассировки лучей с использованием этого движка, которая работает с видеокартами AMD и Nvidia и не требует производительности графического чипа RTX.

Наконец, перейдем к инструменту GameSDK, который позволяет быстро создавать пользовательские игры, в том числе игры, размещенные на официальном сайте Crytek.

У этого движка есть и недостатки. Например, многие разработчики указывают на трудности в работе с этим движком. Это обусловлено сложностью построений, наличием ошибок в редакторе, скромным выбором ресурсов, Ограничения при разработке онлайн-игр,

Game Maker: Studio – игровой движок разработанный и поддерживаемый компанией YoYo Games, написанный на языке C#. При разработке игры используется его собственный упрощенный язык программирования GML (Game Maker Language).

Представленный игровой движок имеет несколько версий распространения:

– Trial;
– Desktop;
– Web.

Каждая версия имеет свою платформу, на которой разрабатывается игра, за исключением Thrial. Территориальная – это бесплатная пробная версия, предназначенная для тех, кто хочет попробовать игровой движок в первый раз. В пробной версии количество объектов, появляющихся в игре ограничено и может компилировать проект только для теста на операционной системе Windows.

Таким образом для реализации игрового приложения была выбрана среда разработки Unity, исходя из преимуществ таких, как возможность быстрого освоения редактора платформы и возможности компилировать игровые проекты на таких современных операционных системах таких как Windows 11, Mac OS и различных дистрибутивов Linux. Также есть возможность использования визуального языка программирования для разработчиков мало знакомых с программированием.

2.3 Обзор возможностей программной среды Unity

Разбор возможностей программной среды Unity начинается с разбора интерфейса редактора, предоставляемого компанией разработчиков. Во-первых, проект в Unity делится на сцены (уровни) – отдельные файлы, содержащие свои игровые локации со своим набором объектов. (рисунок 21).

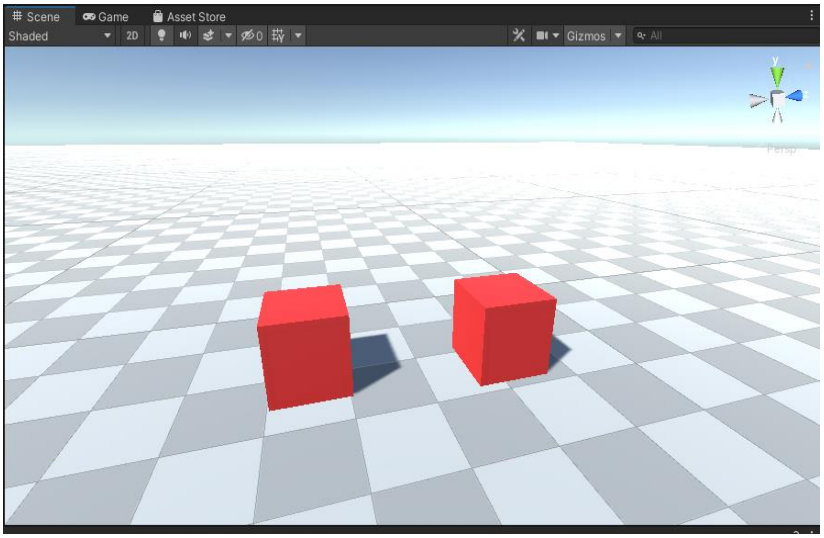

Рисунок 21 – Окно сцены

Редактор Unity имеет простой интерфейс, который легко настраивать, состоящий из различных окон, благодаря чему можно производить отладку игры прямо в редакторе [7]. Движок поддерживает два скриптовых языка: C#, JavaScript. Сцены могут содержать в себе как объекты (модели), так и пустые игровые объекты – объекты, которые не имеют модели («пустышки»). Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют

скрипты. Также у объектов есть название, может быть тег (метка) и слой, на котором он должен отображаться. Так с помощью панели инструментов можно управлять элементами сцены (рисунок 22).

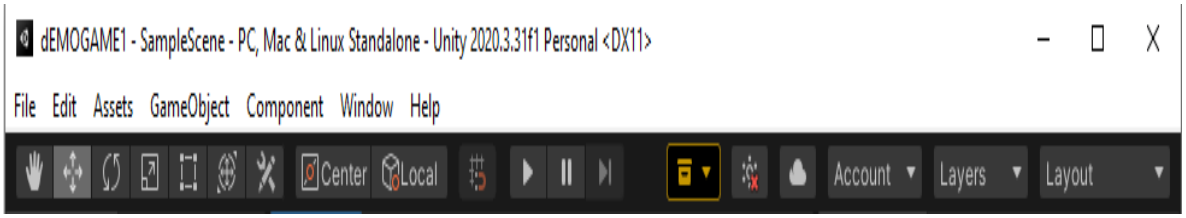

Рисунок 22 – Панель инструментов

Например, с помощью компонента Transform tool можно влиять на содержимое сцены, т. е. перемещать, изменять размеры объектов. С помощью другого элемента панели – Кнопки Play/Pause/Step используются для запуска, паузы текущей версии видеоигры [7]. Помимо этого, с помощью панели инспектора в программной среде Unity отслеживаются свойства игровых объектов на сцене и добавляются новые (рисунок 23).

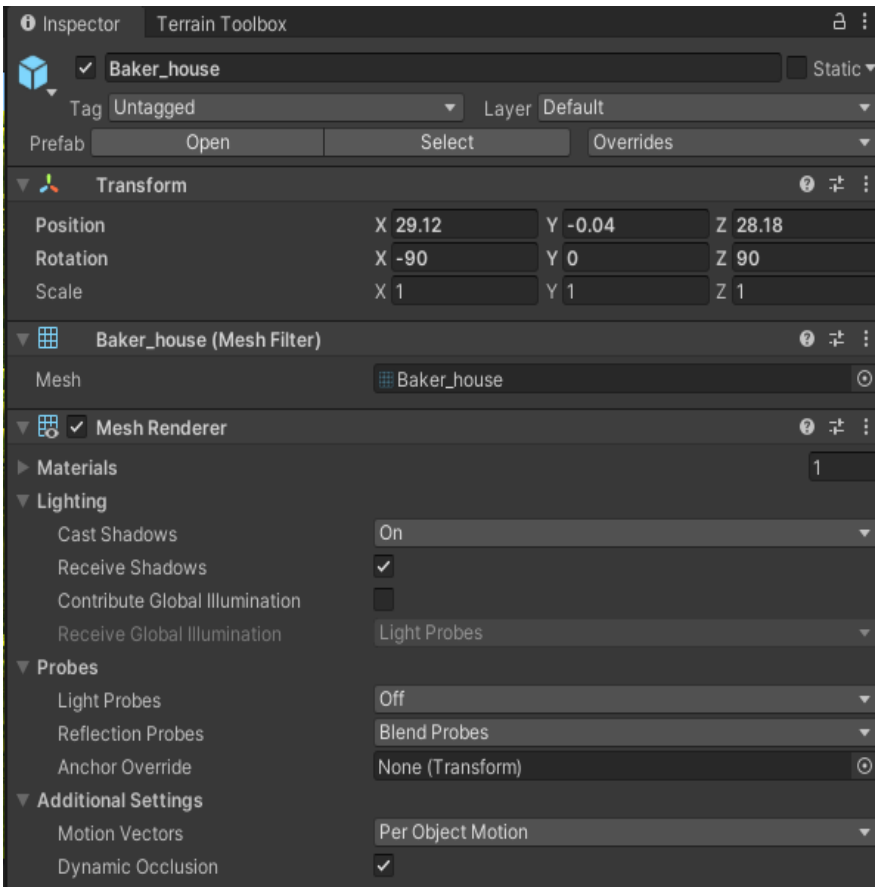

Рисунок 23 – Окно инспектора

К таким свойствам можно отнести компонент Mesh Renderer объекты, делающий модель объекта видимой (по умолчанию присутствует) [7]. К объектам можно применять коллизии (в Unity т. н. коллайдеры – collider), которых существует несколько типов и другие типы компонентов. Также Unity поддерживает физику твёрдых тел и ткани, а также физику типа Ragdoll (тряпичная кукла). В редакторе имеется система наследования объектов; дочерние объекты будут повторять все изменения позиции, поворота и масштаба родительского объекта. Скрипты в редакторе прикрепляются к объектам в виде отдельных компонентов. Все элементы выбранной сцены хранятся в окне иерархии (Рисунок 24).

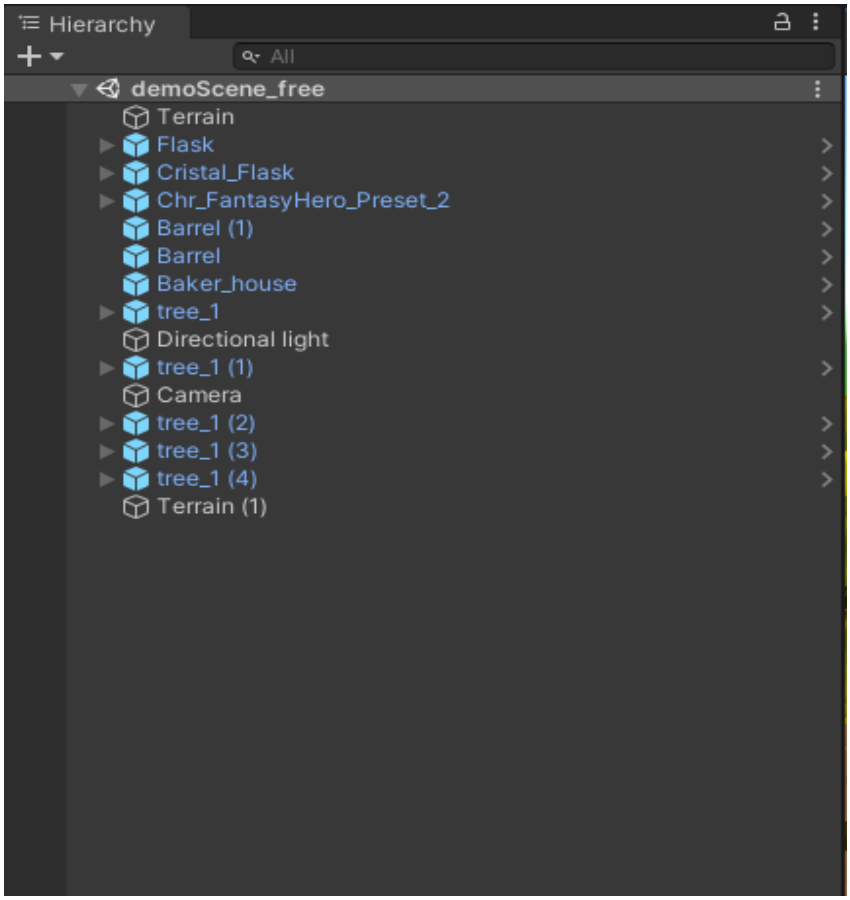

Рисунок 24 – Окно иерархии

Некоторые из них являются прямыми экземплярами файлов ассетов, таких как 3D-модели, а другие – экземплярами префабов, пользовательских объектов, из которых состоит большая часть игры. Есть возможность выбрать объекты в иерархии, и перетаскивать один объект на другой, для создания родительской

связи. При добавлении и удалении объектов в сцене, они также будут появляться и исчезать из иерархии. Unity 3D поддерживает систему Level Of Detail (сокр. LOD), суть которой заключается в том, что на дальнем расстоянии от игрока высоко детализированные модели заменяются на менее детализированные, и наоборот, а также систему Occlusion culling, суть которой в том, что у объектов, не попадающих в поле зрения камеры не визуализируется геометрия и коллизия, что снижает нагрузку на центральный процессор и позволяет оптимизировать проект [7]. При компиляции проекта создается исполняемый (.exe) файл игры (для Windows), а в отдельной папке – данные игры.

Все файловая система проекта находится в обозревателе объектов (рисунок 25).

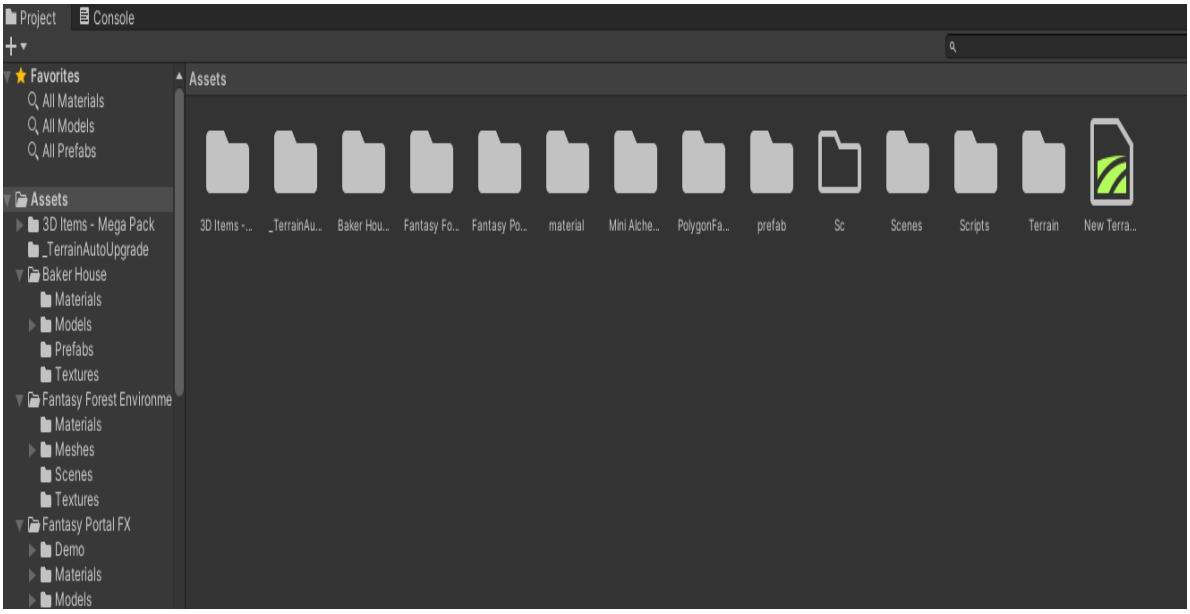

Рисунок 25 – Обозреватель объектов

Левая панель обозревателя служит для отображения общей структуры файловой системы. Содержимое каждой папки можно посмотреть просто если навестись на нее. Иконками представлены различные типы файлов. Иконки можно изменять по усмотрению разработчика или представить их в виде дерева иерархии. Левее от слайдера выводится имя выбранного ассета и полный путь до него. Над списком структуры проекта находится раздел Favorites (избранное), в котором хранятся ссылки на часто используемые ассеты для быстрого доступа к ним.

Прямо над панелью находится навигационная цепочка, отображающая путь до просматриваемой папки. В секции General окна настроек Unity можно отключить отображение количество найденных в Store ассетов, если они не требуются. Обозреватель имеет мощные возможности поиска, которые особенно полезны при поиске ассетов в большом или незнакомом проекте. Обычный поиск будет фильтровать ассеты согласно тексту, введённому в поле для поиска.

Имеется возможность импорта. При импорте текстуры в Unity можно сгенерировать alpha-канал, mip-уровни, normal-map, light-map, карту отражений, однако непосредственно на модель текстуру прикрепить нельзя – будет создан материал, которому будет назначен шейдер, и затем материал прикрепится к модели.

Модели, звуки, текстуры, материалы, скрипты можно запаковывать в формат. unityassets и передавать другим разработчикам, или выкладывать в свободный доступ. Этот же формат используется во внутреннем магазине Unity Asset Store [7].

Редактор Unity поддерживает создание шейдеров с помощью технологии ShaderGraph. С помощью компонента Unity Animator в среде легко создавать скелетные анимации с нуля или же просто импортировать уже готовую анимацию в редактор и связывать ее с игровым объектом. Магазин Unity Asset Store позволяет разработчикам выкладывать различные материалы для создания видеоигры такие как звуковые фалы, 3D-модели и т.д. (рисунок 26).

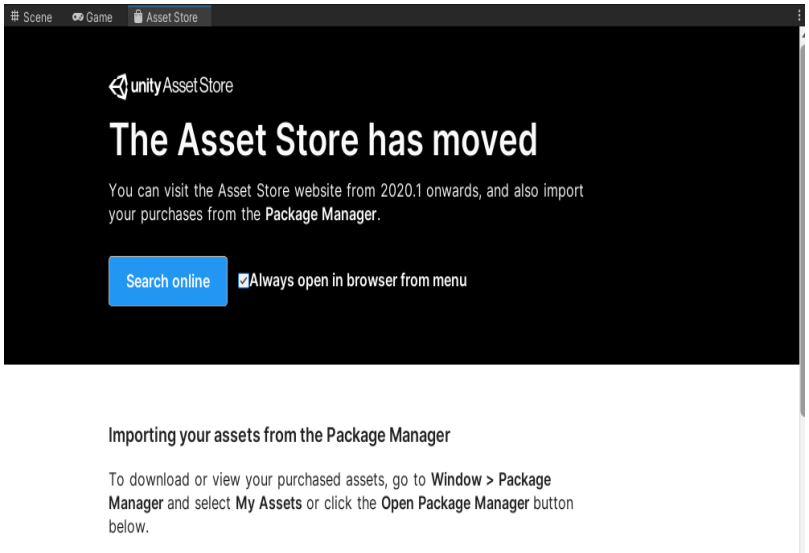

Рисунок 26 – Unity asset store

За каждый отдельный ассет может быть установлена цена, которую создатель посчитает нужной или же вести политику бесплатного распространения контента. Для использования функционала данного магазина необходимо зарегистрировать аккаунт Unity.

Таким образом, были перечислены основные возможности программной среды Unity.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРЕДЛАГАЕМОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ

3.1 Основные этапы практической разработки

Для программного решения существующих задач, было выявлено несколько этапов работы:

– реализация системы генерации игровых предметов;

– создание материалов для видеоигры;

– система генерации игрового уровня;

– игровой персонаж и его анимации взаимодействия;

– реализация главного меню видеоигры.

Описание архитектуры видеоигры было изложено выше. Далее будет описан каждый этап реализации игрового приложения.

3.1.1 Реализация системы генерации игровых предметов

Принцип работы алгоритма основан на создании определенных игровых объектах – spawner (спавнеров), которые порождают необходимые предметы. На рисунке 27 представлена блок схема алгоритма.

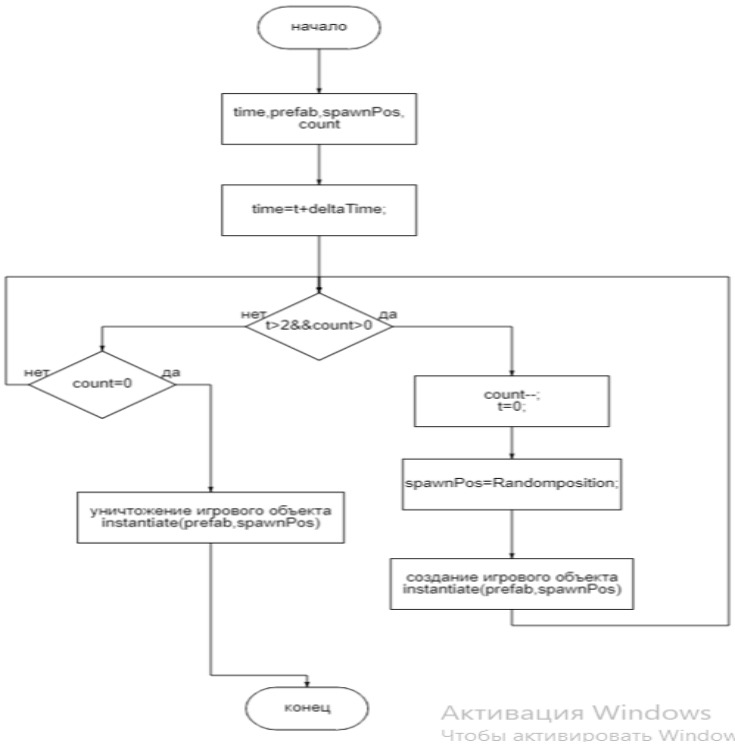

Рисунок 27 – Блок-схема алгоритма генерации игровых предметов

После того как данные игровые объекты создадут необходимое количество предметов они автоматически уничтожаются, для того чтобы не допустить появление лишних игровых предметов (Рисунок 28).

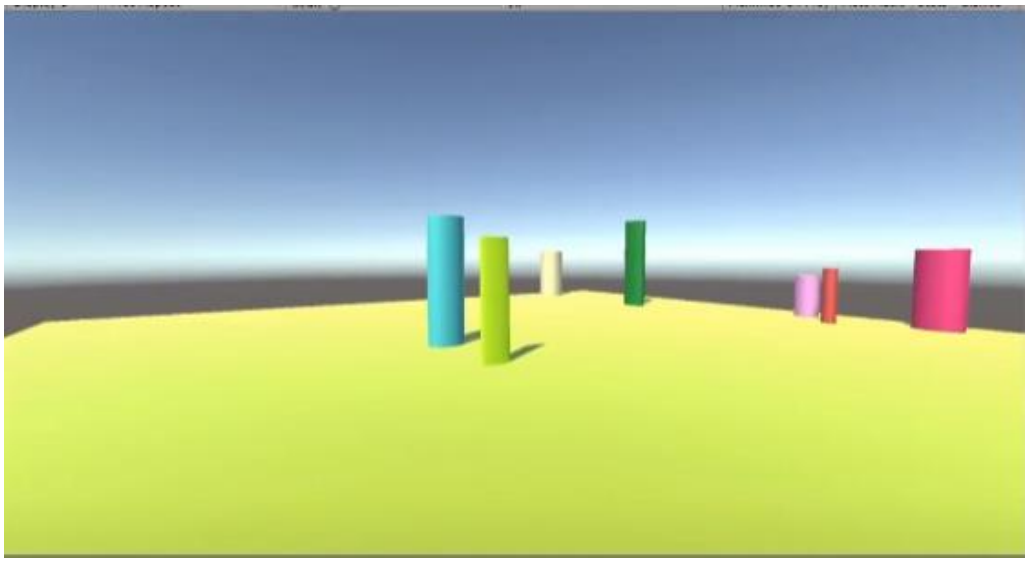

Рисунок 28 – Результат работы алгоритма

Данный алгоритм можно использовать как основу для системы генерации игровых предметов (сферы здоровья, оружие, предметы усиления, артефакты), так и для точек появления противников.

3.1.2 Реализация материалов для видеоигры

Прежде чем приступать к работе с материалами для ландшафта, воды и растительности необходимые для реализации системы игрового окружения была выполнена настройка технологии Universal Render Pipeline. Это необходимо для корректной работы шейдеров и текстур. Технологию URP можно использовать практически на любой крупной платформе, и она обеспечивает красивую графику и высокую производительность [8]. Первое, что нужно сделать – создать файл настройки «URP» – Create – Rendering – Universal Render Pipeline, а затем перейти к редактированию настройка проекта. Автоматически создается два файла Universal Render Pipeline.

Далее в разделе Scriptable Render Pipeline Settings выбрать «URP» который создали. После этого материалы были преобразованы для дальнейшей работы с

ними. Обновление материалов проекта произведено с помощью «Upgrade Project Materials to UniversalRP» (Рисунок 29).

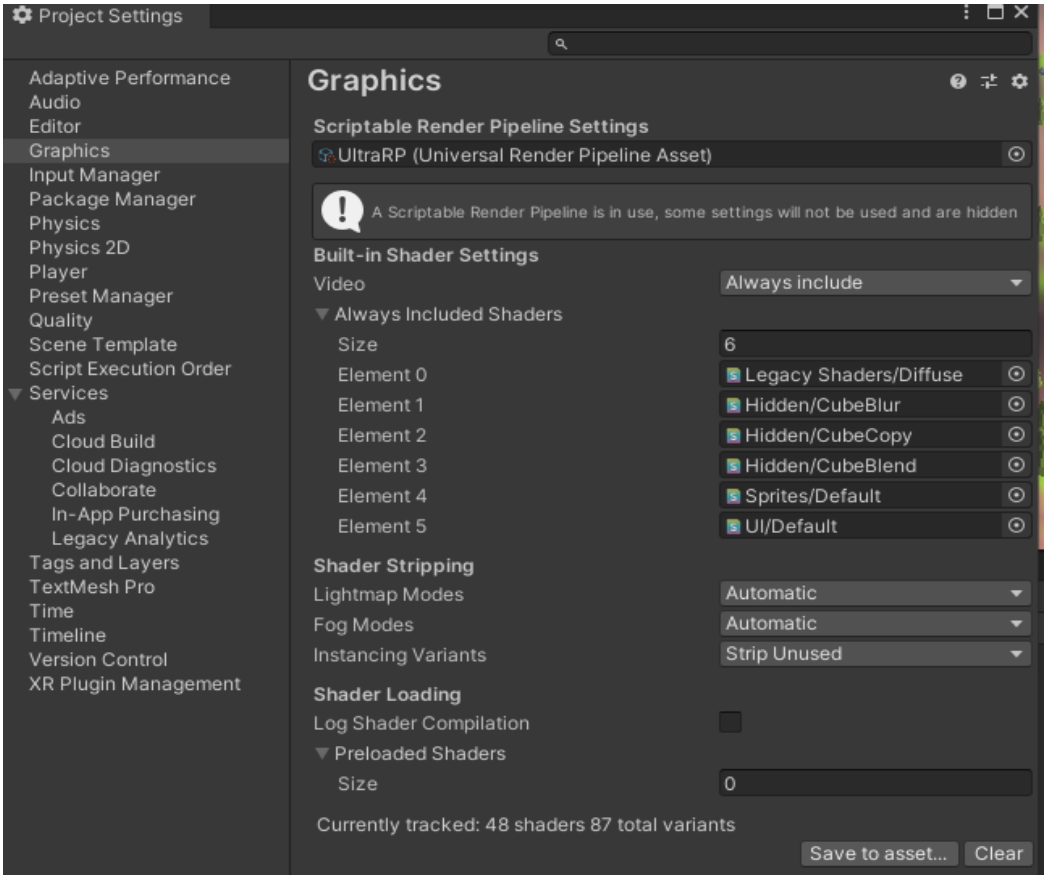

Рисунок 29 – Настройки Universal Render Pipeline

После настройки проекта игрового приложения преобразуется каждая текстура отдельно. Например, сцена состоит в основном из деревьев и травы, которые нужно преобразовать. Если материал использует пользовательский шейдер он не будет преобразован. С помощью инспектора для каждого конкретного материала были применены свойства «Universal Renderpipeline/Lit», а также, включен Alpha Clipping, а значение Smoothness установлено 0.

После этого был создан новый материл для текстуры ландшафта с помощью Create – Material, и установлен такой же тип шейдера «Universal Render Pipeline/Terrain/Lit» что и для других материалов. Таким образом, настройка графической составляющей почти завершена. Остается только в инспекторе исправить параметры теней объектов (surface type), выбрав прозрачный (transparent) и заменить на непрозрачный (opaque) и установить Threshold на 0,5.

Следующим шагом был установлен плагин Shader Graph необходимый для создания пользовательских текстур (Рисунок 30).

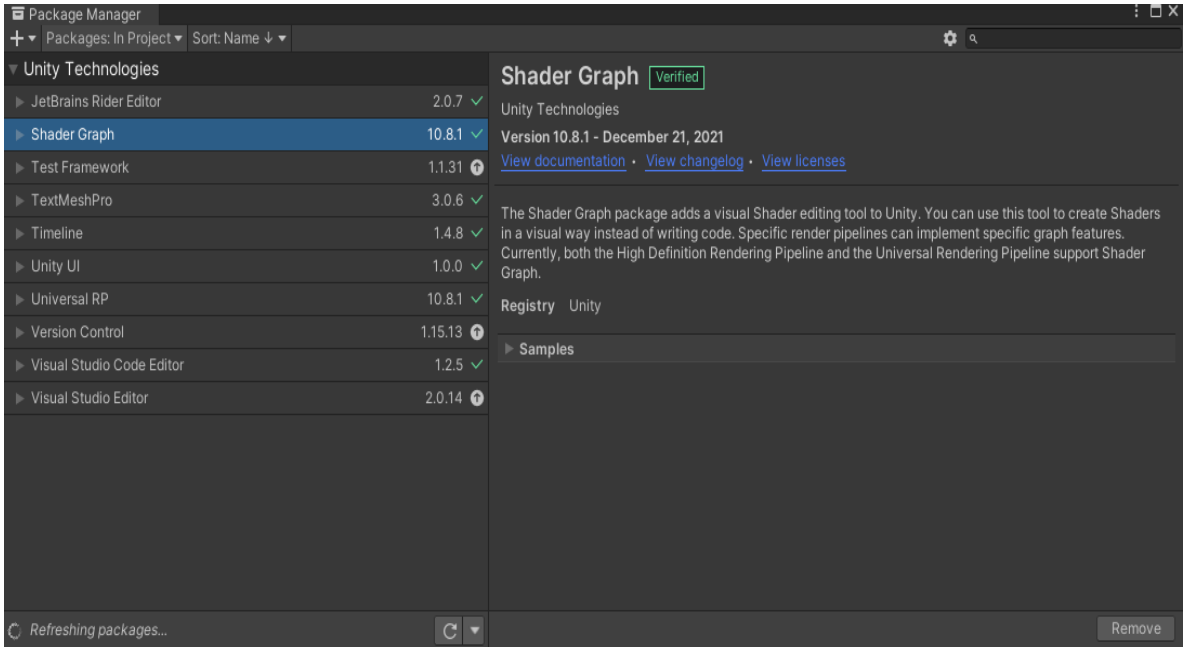

Рисунок 30 – Окно установки ShaderGraph

Материалы используются совместно с Mesh Renderers, Particle Systems и другими компонентами рендеринга, используемыми в Unity. Они играют важную роль в определении того, как отображается ваш объект. На рисунке 31 изображены созданные материалы.

Рисунок 31 – Материалы текстур

Основные свойства, отображаемые инспектором материала, определяются шейдером, который использует материал.

Шейдер – это специализированная графическая программа, которая определяет, как информация о текстуре и освещении комбинируется для создания пикселей визуализируемого объекта на экране. Shader Graph – это решение Unity для работы с шейдерами [8]. Вместо того, чтобы писать код с нуля, Unity предоставляет удобный холст («график»), который позволяет добавлять к нему определенные узлы шейдера. Каждый из этих узлов шейдера предназначен для выполнения чего-то другого, например, для изменения общего цвета, добавления шума или изменения внешнего вида зеркального освещения. С помощью Shadergraph были созданы текстуры (материал) ландшафта и воды. На рисунке 32 представлен шейдер для ландшафта.

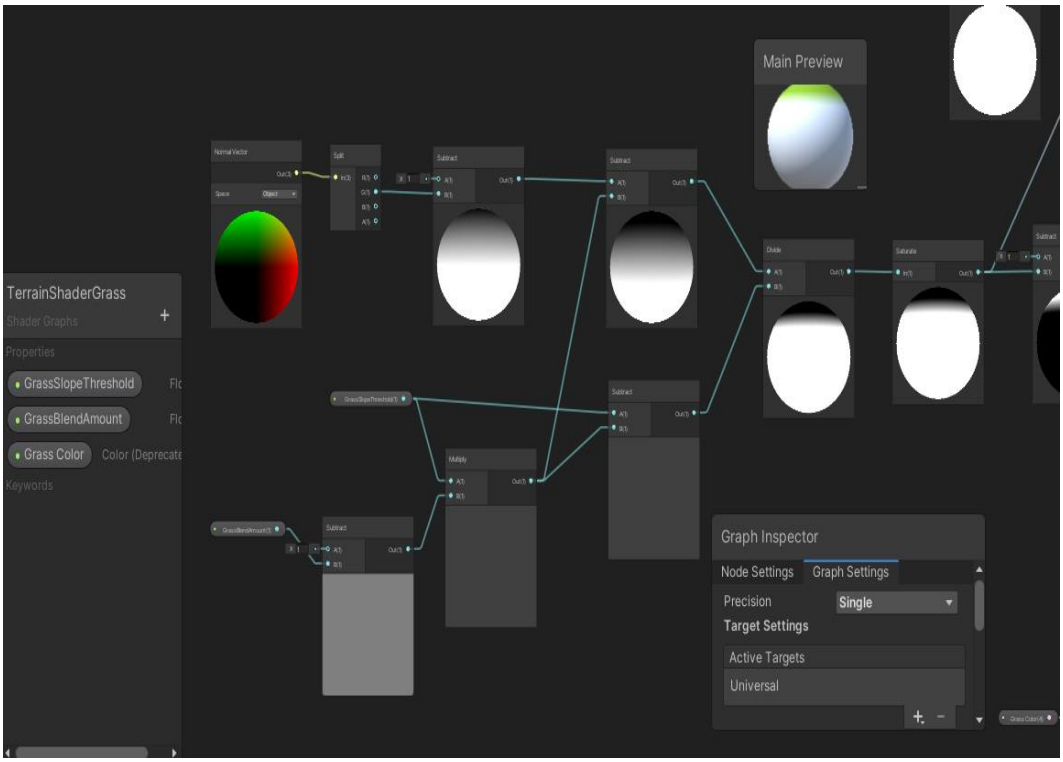

Рисунок 32 – Текстура (шейдер) ландшафта

На выходе каждого шейдера контролируются следующие свойства:

– Position – позицию вершины;

– Color – цвет каждого пикселя поверхности;

– Alpha – его прозрачность;

– AlphaClipThreshold – порог прозрачности.

Также был создан шейдер воды, изображенный на рисунке 33. Возможно усовершенствование шейдера воды, которое можно учесть при рендеринге эффекта воды. Например, воспроизведением преломления, симуляцией пены, и т.д.

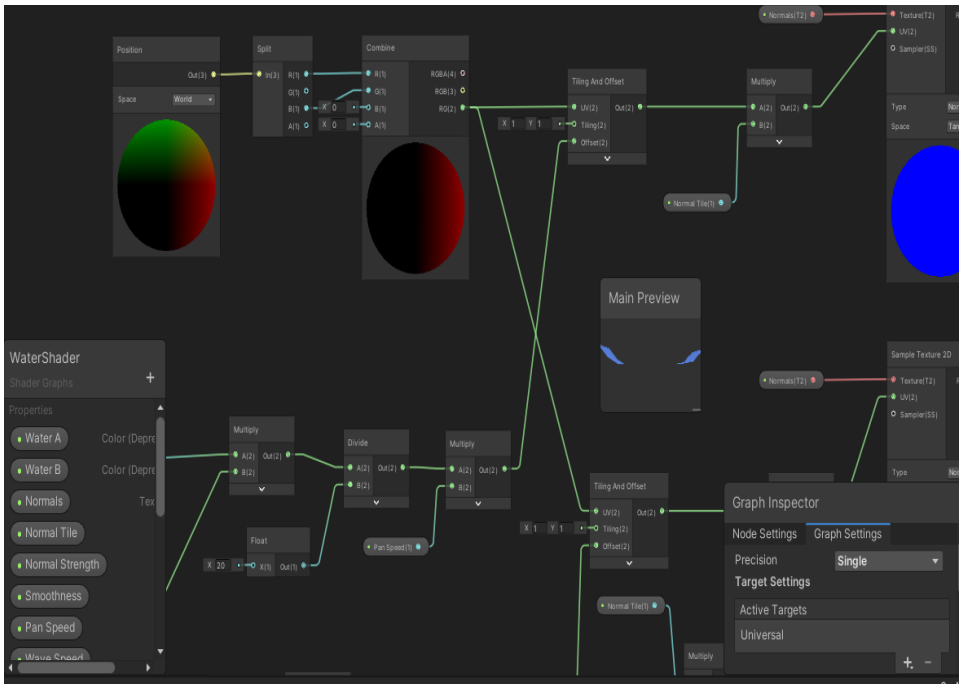

Рисунок 3 – Текстура (шейдер) воды

Таким образом, были реализованы все необходимые материалы для создания игрового приложения.

3.1.3 Реализация системы создания игрового ландшафта

В ходе реализации системы генерации игрового окружение была создана иерархия файловой системы, изображенной на рисунке 34.

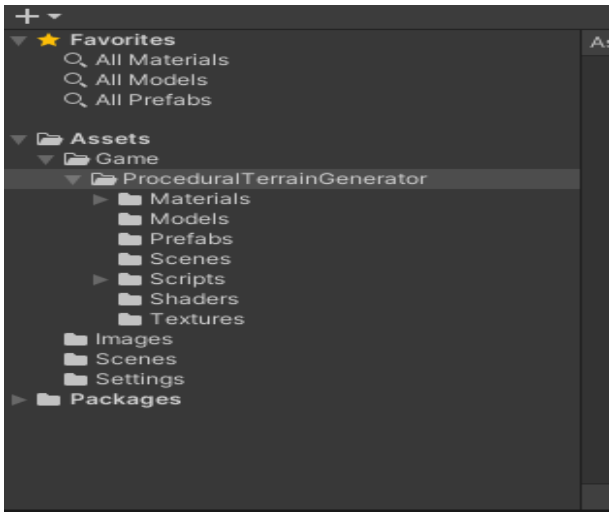

Рисунок 34 – Файловая система генератора

Реализация системы игрового окружения начинается с написания скриптов генератора, который содержит два вида алгоритмов – алгоритм, основанный на функции шума Перлина и diamond-square. Чтобы сделать равнины более пологими, а склоны гор более крутыми нормализованные значения возводятся в квадрат. Для того чтобы создать процедурно сгенерированную текстуру существует несколько шагов. В основе генерации лежит вышеописанный алгоритм, который поможет нам полущить карту высот. Сначала необходим двумерный массив размерностью 2^n+1. Далее необходимо реализовать два основных шага – square и diamond. Square будет принимать две противоположные точки квадрата – левую нижнюю и правую верхнюю и потом сохранять значение и центр квадрата. Следующий метод – diamond – будет принимать значение точки, у которой нужно вычислять центральную точку квадрата. Далее вычисляет точку на базисе значений перекрёстных углов квадрата, центра и центральной точки соседнего квадрата. Ниже показана реализация одного из шагов DS. Далее необходимо проверить координаты всех точек не входят ли они за пределы массива. Когда это все же случается то точке необходимо присвоить обратное ей значение. Результат работы алгоритма представлен на рисунке 35.

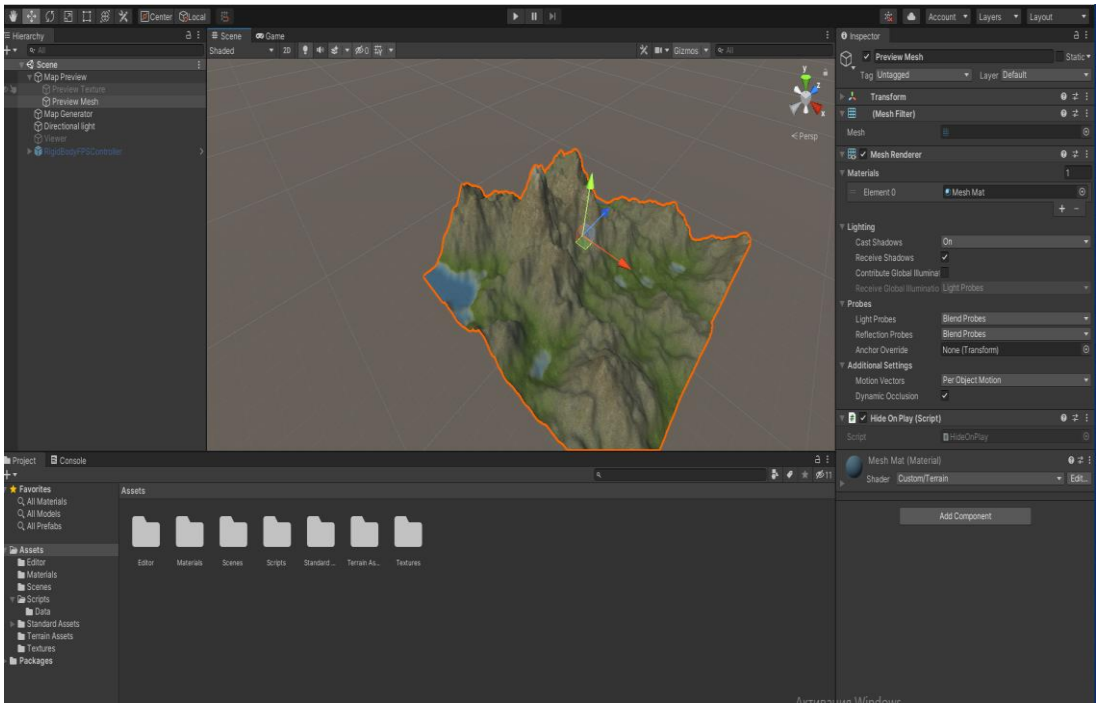

Рисунок 35 – Результат работы алгоритма

В следующем шаге необходимо перейти к цвету. Его генерация происходить при помощи midpoint displacement. В текущем этапе он будет использоваться напрямую, т.е. он будет содержать список координат по оси y, которые сопоставлены координате по оси x. Ось x является границей поясов. Для придания поясу замкнутости необходимо сделать края равными. Для пущего реализма необходимо делать различные значения roughness, таким образом, чтобы пояса могли двигаться по неожиданным траекториям. Выходная сгенерированная текстура должна быть приближена к реальной. Для того чтобы сделать текстуру более реалистичной можно возвести значения в степень, что придаст ей более пологий и гладкий вид.

В свойствах объекта генератора созданы настройки генератора, ландшафта и карты высот (Рисунок 36).

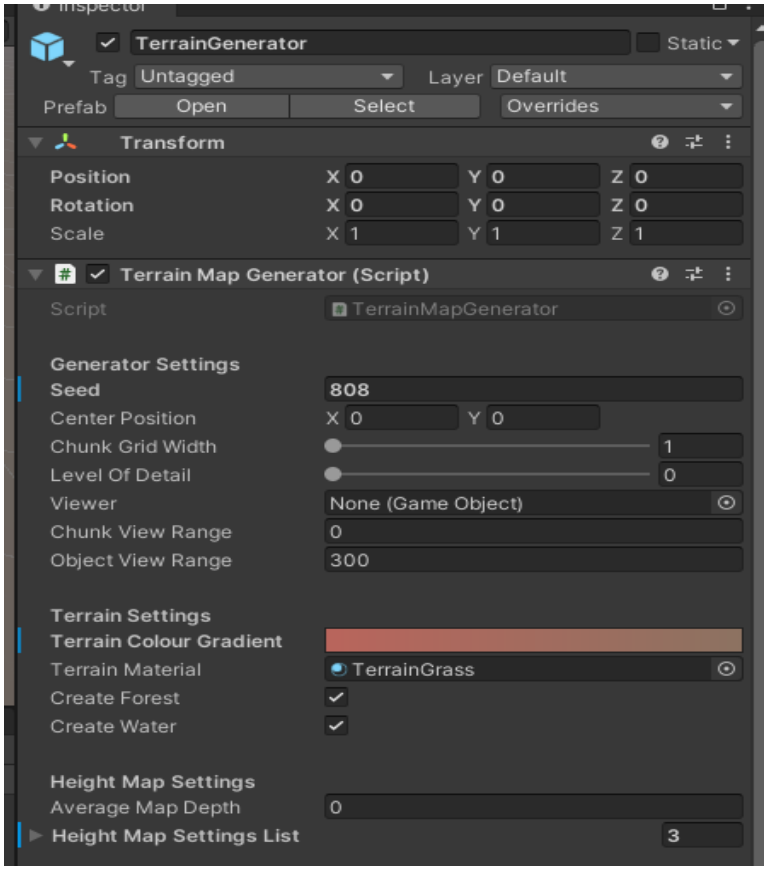

Рисунок 36 – Свойства объекта генератора

При добавлении в алгоритм деревьев и травы можно получить более проработанный игровой уровень (рисунок 37).

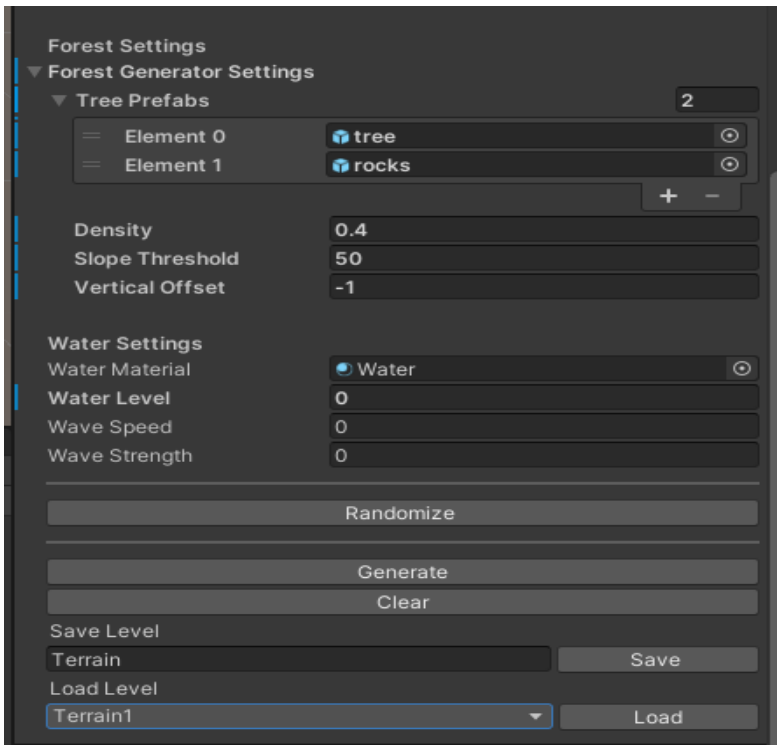

Рисунок 37 – Свойства воды и леса

Таким образом, генерируя случайные условия ландшафта, можно получать различные игровые уровни (Рисунок 38).

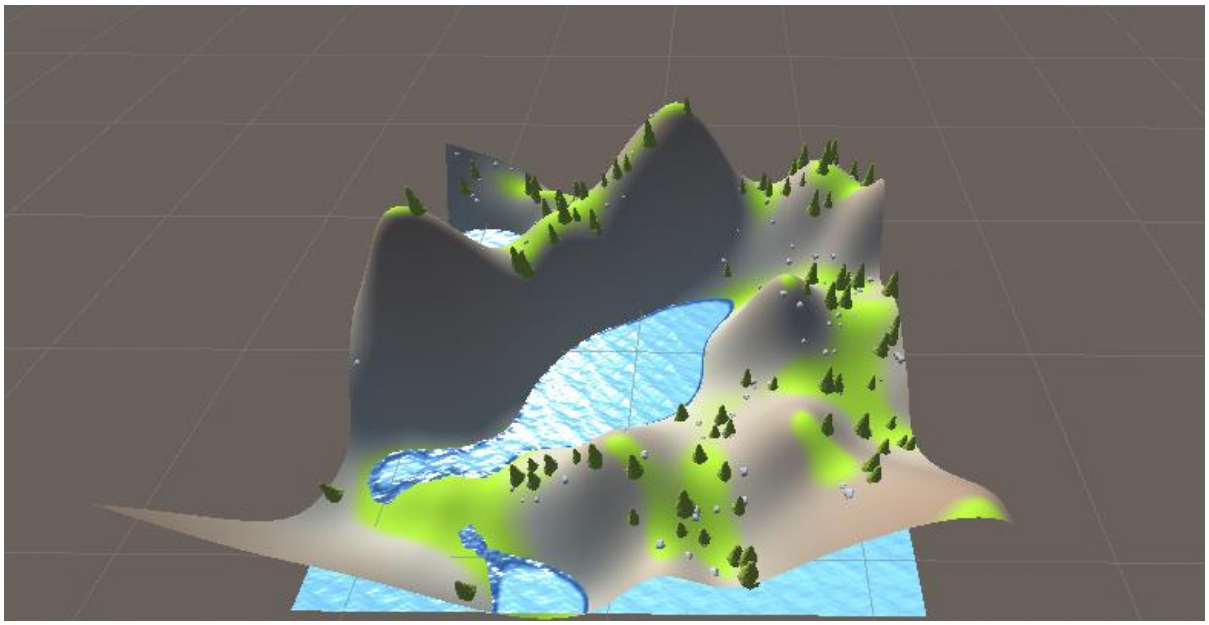

Рисунок 38 – Сгенерированный ландшафт

Также, добавив в свойства леса необходимые игровые объекты, можно контролировать количество растительности на уровне.

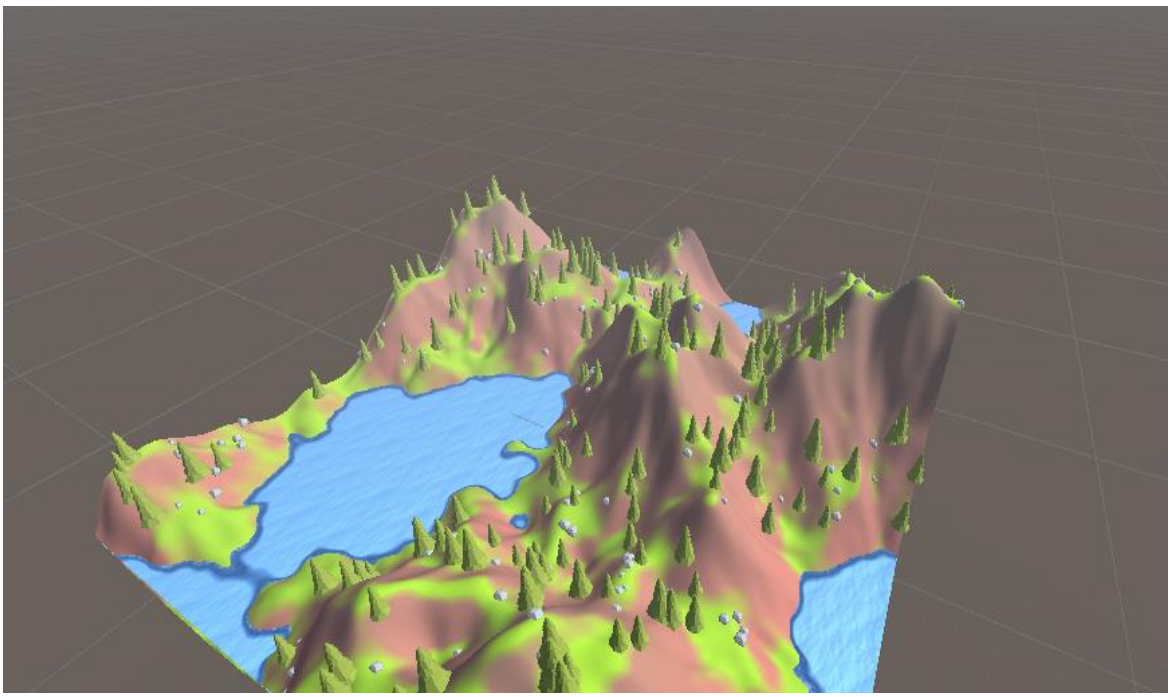

Рисунок 39 – Предварительный шаблон игрового уровня

Таким образом, в итоге был реализован предварительный шаблон игрового уровня (Рисунок 39).

3.1.4 Создание и анимация персонажа

Для того чтобы создать персонажа нужна его 3D-модель. С помощью asset store была загружена 3D-модель персонажа (рисунок 40). Также к нему был добавлен capsule collider для того чтобы персонаж мог физически взаимодействовать с окружением.

Рисунок 40 – Игровой персонаж

Для персонажа были созданы анимации бездействия бега и прыжка (Рисунок 41).

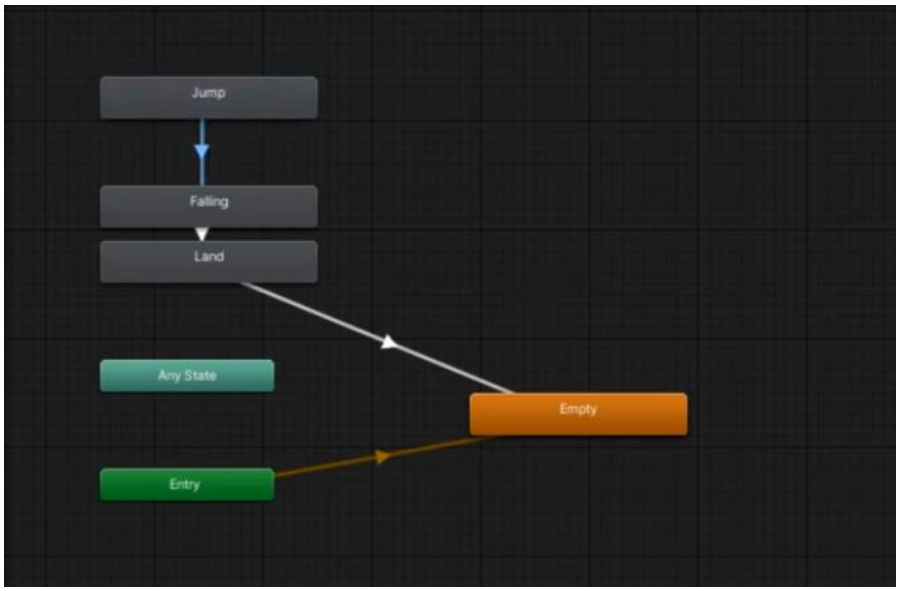

Рисунок 41 – Аниматор персонажа

В окне аниматора Animator были созданы 3 состояния (Create State): Idle, Land, Jump. Выделив одно из состояний (например, idle), в окне Inspector в поле Motion помещается анимационный клип (у которого настроены ключи Start-End, Loop pose, и т.д.). Также устанавливаются клипы в остальные состояния (walk, jump). Таким образом, все необходимые анимации для персонажа были реализованы

3.1.5 Реализация главного меню

Пользовательский интерфейс программы, приложения или игры включает в себя все элементы, которые позволяют пользователю вводить и получать обратную связь от программного обеспечения. GUI включает в себя такие элементы, как код, поддерживаемые устройства ввода и вывода, а также аудио- и видео отчеты и находится между пользователем и внутренними компонентами программного обеспечения. Дизайн HUD (часть визуального интерфейса игрока, отображающаяся на переднем плане виртуального игрового пространства в видеоигре) и пользовательского интерфейса – один из самых недооценённых, но

крайне важных элементов разработки. Он определяет, как пользователь будет взаимодействовать с основными системами игры, и не только передаёт игроку информацию о персонаже и об игровом мире, но и формирует определённую модель его поведения.

Среда Unity предоставляет разнообразный список технологий и инструментов для создания GUI. Основным инструментом разработчиков для создания визуальных объектов является Unity UI. Unity UI включает в себя н инструменты для создания пользовательского интерфейса такие как Canvas и других интерактивных объектов разрабатываемого игрового приложения.

Еще одним инструментом для создания интерфейса является UI Toolkit, который включает специальные инструменты для разработки, такие как UI Builder и UI Debugger [9]. Они создавались по примеру аналогичных веб-технологий. Инструмент UI Builder подходит для визуального проектирования и редактирования пользовательских интерфейсов игр и приложений прямо в Unity, а UI Debugger помогает быстро устранять проблемы. В отличие от Unity UI, в данном инструменте можно создавать элементы интерфейса без использования текстур непосредственно в UI Toolkit. Такой подход значительно сокращает время, затрачиваемое на переключение между инструментами, а также уменьшает общее потребление памяти.

При разработке пользовательского интерфейса также использовалась технология Cinemachine. Cinemachine – это набор модулей для работы с камерой Unity. Cinemachine решает сложную математику и логику отслеживания целей, компоновки, смешивания и разделения кадров [9]. Он предназначен для значительного сокращения количества трудоемких ручных манипуляций и ревизий скриптов, которые происходят во время разработки. Процедурный характер этих модулей делает Cinemachine устойчивым к ошибкам. При внесении изменений – например, изменение анимации, скорости автомобиля, ландшафта или других игровых объектов в сцене – Cinemamachine динамически регулирует свое поведение, чтобы сделать лучший кадр. Нет необходимости переписывать сценарии камеры

только потому, что персонаж поворачивается налево, а не направо. Данный инструмент хорошо работает с другими инструментами Unity, выступая в качестве мощного дополнения [9]. Особенностью пользовательского интерфейса для создаваемой видеоигры является применение 3D-элемента для разработки интерфейса. Решение создать такой интерфейс принято исходя из того, что, что человеку проще погрузиться в игровой процесс, если элементы интерфейса внедрены в него.

Для реализации главного меню создается отдельная сцена, на которой располагаться игровые объекты с функционалом перехода в различные пункты игрового меню. К каждому элементу привязывается игровой объект камеры cinemachine. Также к игровым объектам привязана анимация, которая воспроизводится при интерактивном взаимодействии с пользователем. С помощью созданного скрипта на языке программирования C# происходит плавный переход между различными элементами сцены главного меню. Иерархия игровых объектов главного меню изображена на рисунке 42.

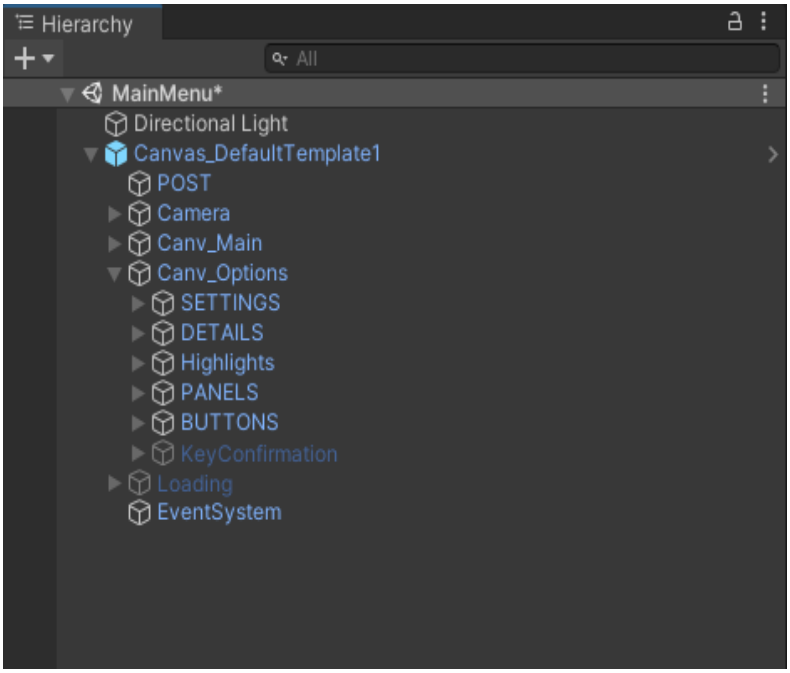

Рисунок 42 – Иерархия объектов главного меню

Для реализации пользовательского интерфейса во время игровой сессии пользователя были использованы UI-элемент холст (Canvas) и на нем размещены

объекты slider(слайдер), textMeshPro, Image [10]. С помощью компонента холста – Canvas Scaler было установлено управление общим масштабом и плотностью пикселей элементов пользовательского интерфейса вместе с размером используемого экрана. В сцене главного меню был установлен skybox (Рисунок 43).

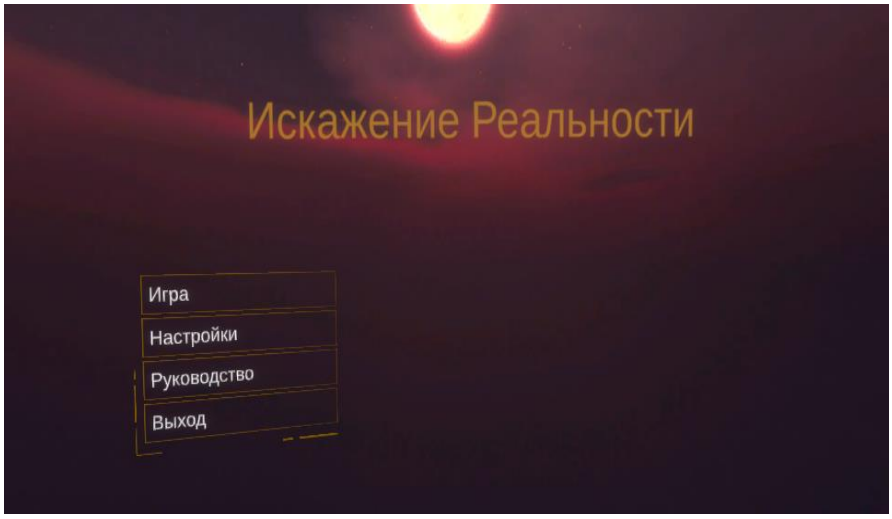

Рисунок 43 – результат работы алгоритма

В панели главного меню были помещены 4 кнопки (играть, настройки, руководство, выход. При нажатии каждой из них появляется дополнительное диалоговое окно с необходимыми инструкциями для игрока. Например, при нажатии кнопки настройки камера перемеситься на другую панель меню изображенной на рисунке 44.

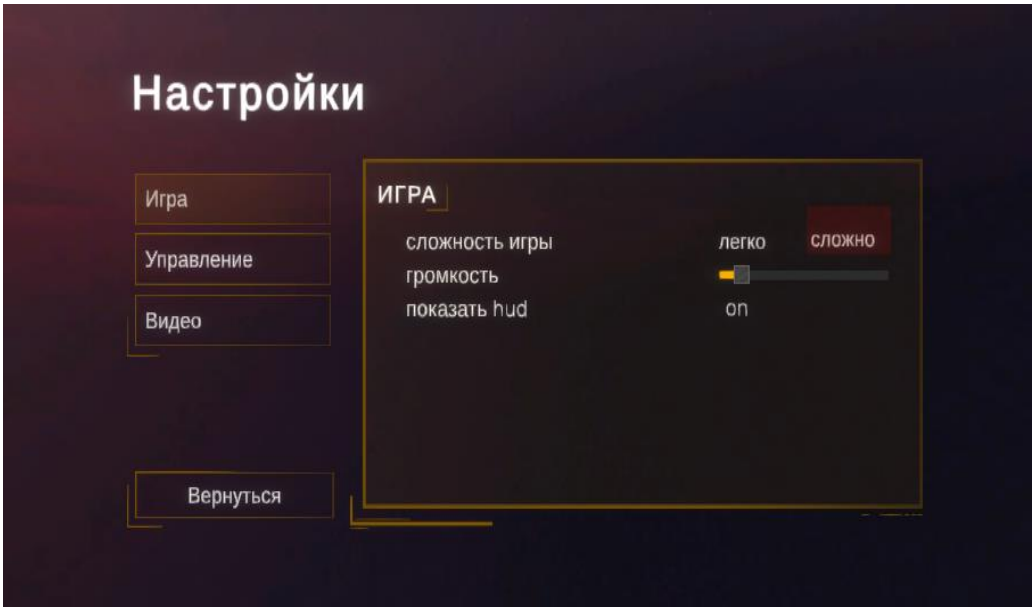

Рисунок 44 – Подпункт игра

Панель настроек содержит три кнопки, необходимые для отображения подпунктов настроек. В подпункте игра можно выбирать уровень сложности, менять громкость и делать видимость Hud персонажа. В подпункте управление изображенном на рисунке 45 находятся настройки компьютерной мышки игрока.

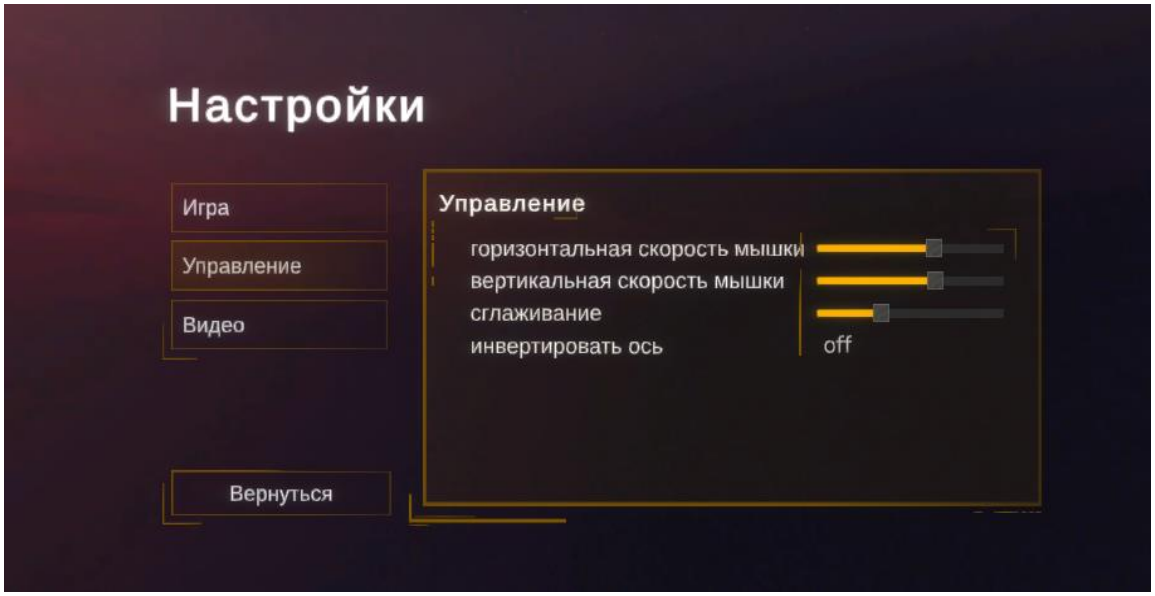

Рисунок 45 – Подпункт управление

В продпункте находятся такие параметры как сглаживание и инвертирование оси. В подпункте видео изображенном на рисунке 46 находятся основные настройки графической составляющей видеоигры.

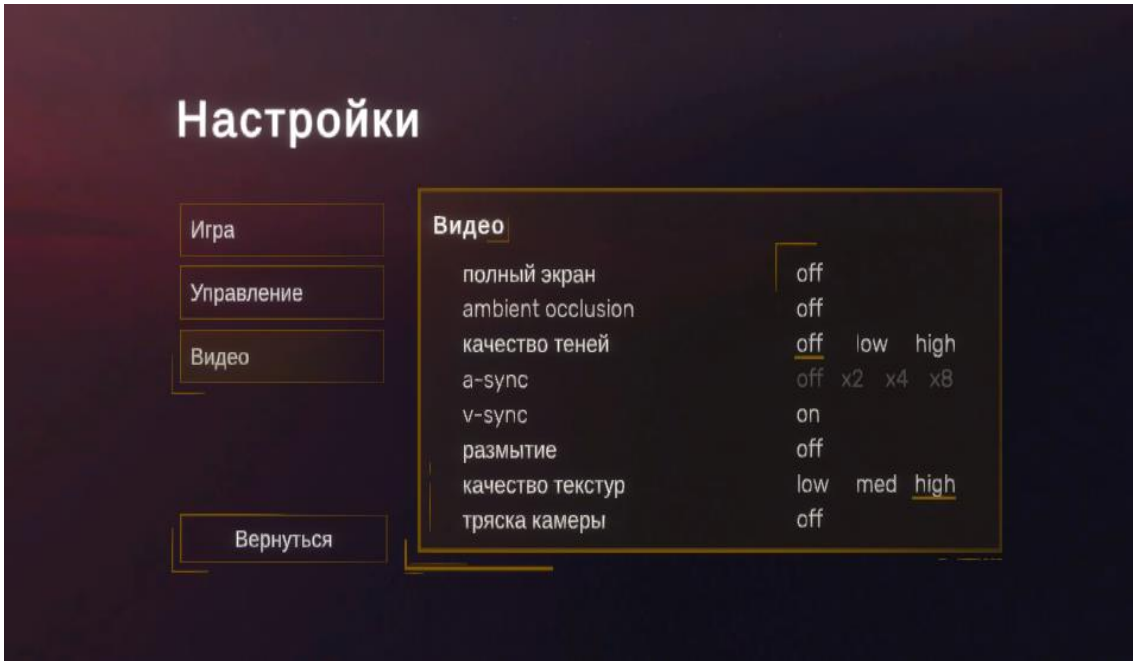

Рисунок 46 – Подпункт видео

Например, можно изменять качество текстур и теней, делать размытие при движении персонажа, тряску камеры. На рисунке 47 показано диалоговое окно, появляющееся после нажатия кнопки игра.

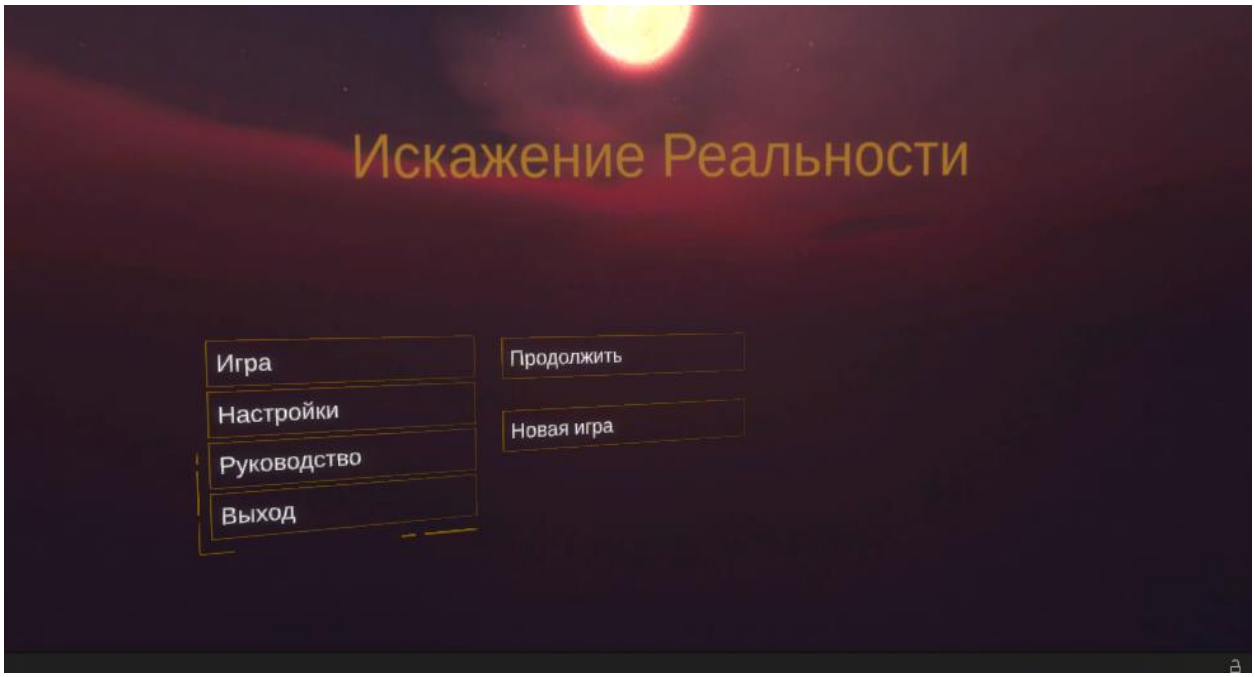

Рисунок 47 – Диалоговое окно пункта игра

В этом диалогом окно находятся кнопки, позволяющие начать новую игру или продолжить ее с того места, на котором игрок остановился при предыдущем прохождении. При нажатии кнопки руководство откроется подпункт с основными правилами видеоигры (рисунке 48).

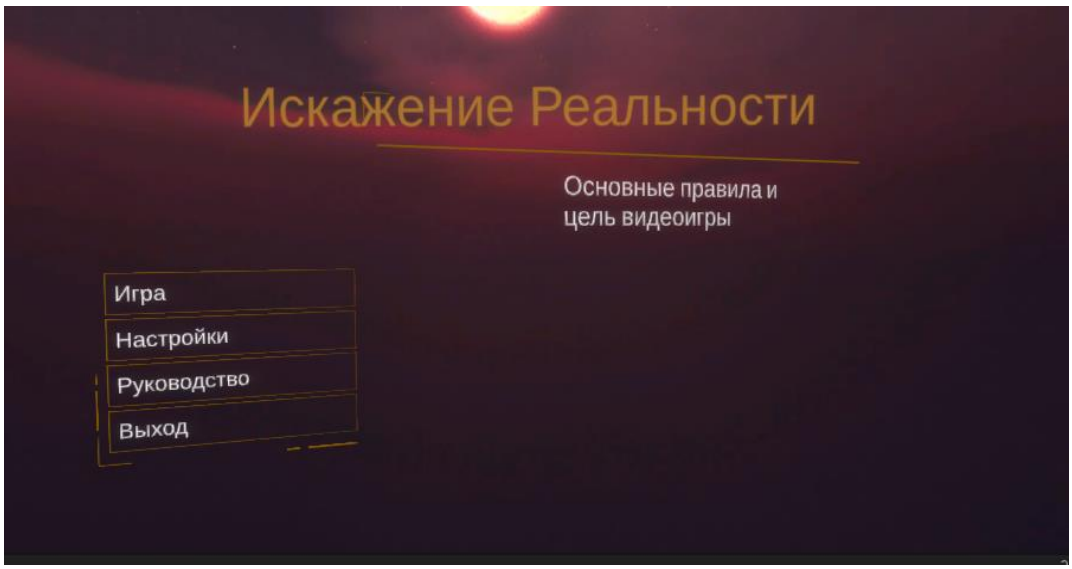

Рисунок 48 – Подпункт руководство

На рисунке 49 изображены подпункт при нажатии кнопки выхода. Диалоговое окно запрашивает подтверждение пользователя на выход из игры. И если ответ утвердительный, то игровое приложение прекращает свою работу.

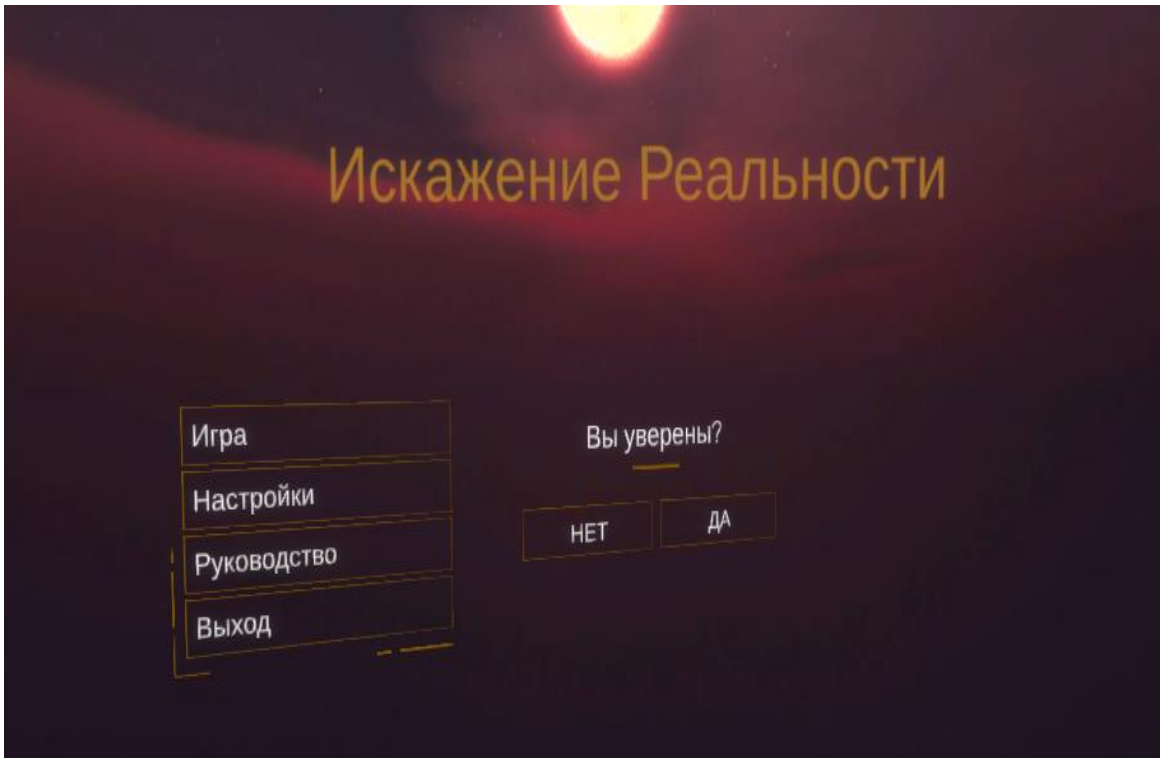

Рисунок 49 – Подпункт выход

Таким образом, с помощью технологий среды Unity был создан прототип функционального интерфейса видеоигры, который не содержит лишних элементов, мешающих пользователю.

3.2 Результаты фактического тестирования программного продукта

Тестирование разработанного прототипа игрового приложения было проведено с помощью специализированной технологии Unity юнит-тестов. С помощью данного типа тестирования можно отследить ошибку в конкретном участке программного кода и убедиться, что исследуемая функция работает правильно.

Работоспособность юнит-теста возможна, если он находится в файле класса проверяемого метода. Test Runner позволяет запускать тестирование как из игрового режима, так и из редактора, следовательно, у разработчика появляется больше возможностей по поиску ошибок. Также система Unity содержит

комплект тестов, который включает в себя файл группы классов в котором находятся юнит-тесты.

Для того чтобы приготовить к отладке Test Runner, в панели инструментов Window была выбрана вкладка General (Рисунок 50).

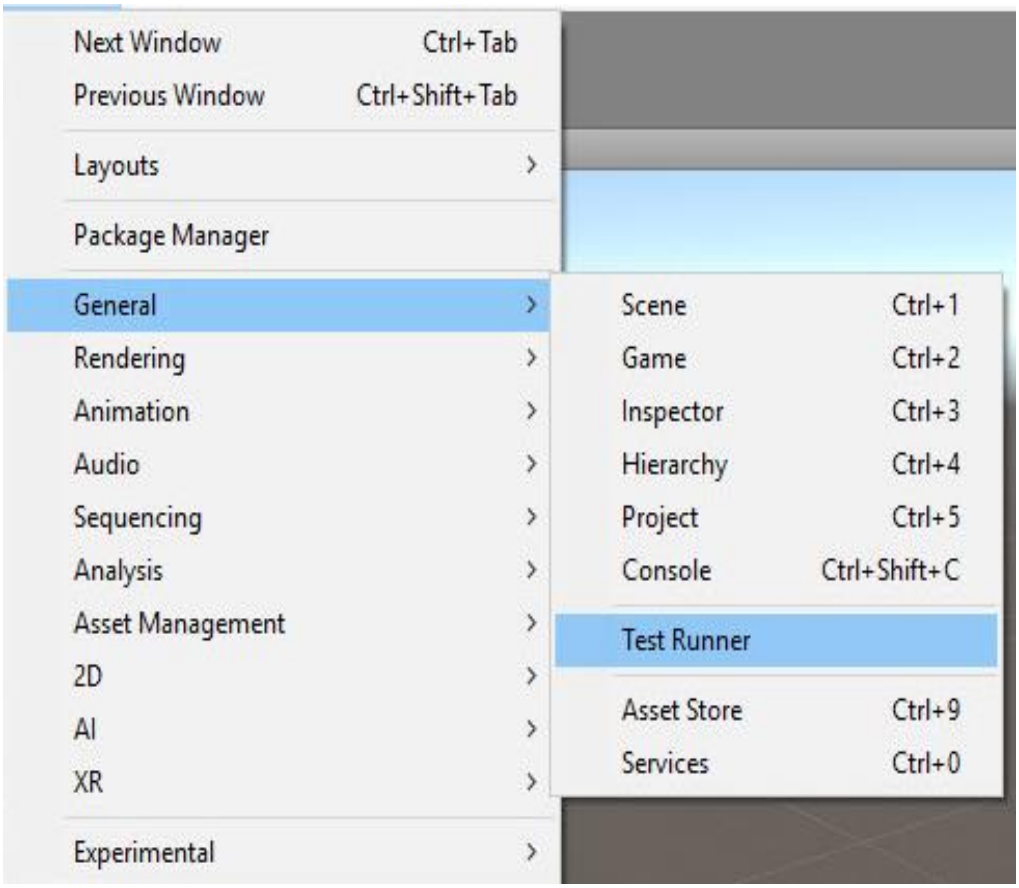

Рисунок 50 – Панель инструментов

Для написания Unit-тестов был создан экземпляр объекта Unity Test Runner, с помощью которого были выполнены тесты и проведена проверка на их успешность.

Тесты запускаются автоматически, любым скриптом, с помощью Test Runner API. Интерфейс позволяет получать список тестов, которые запускаются в режиме редактирования, в режиме игры или в обоих режимах без необходимости их запуска [10]. Есть возможность подключить обратные вызовы регистрации и дерегистрации в начало и конец каждого теста на каждом уровне в рамках цикла тестирования, то есть для каждой тестовой сборки, тестового устройства, тесто-

вого класса и отдельного теста. В начале приходит информация о ходе запускаемого теста. После завершения теста легко увидеть полученные результаты (Рисунок 51).

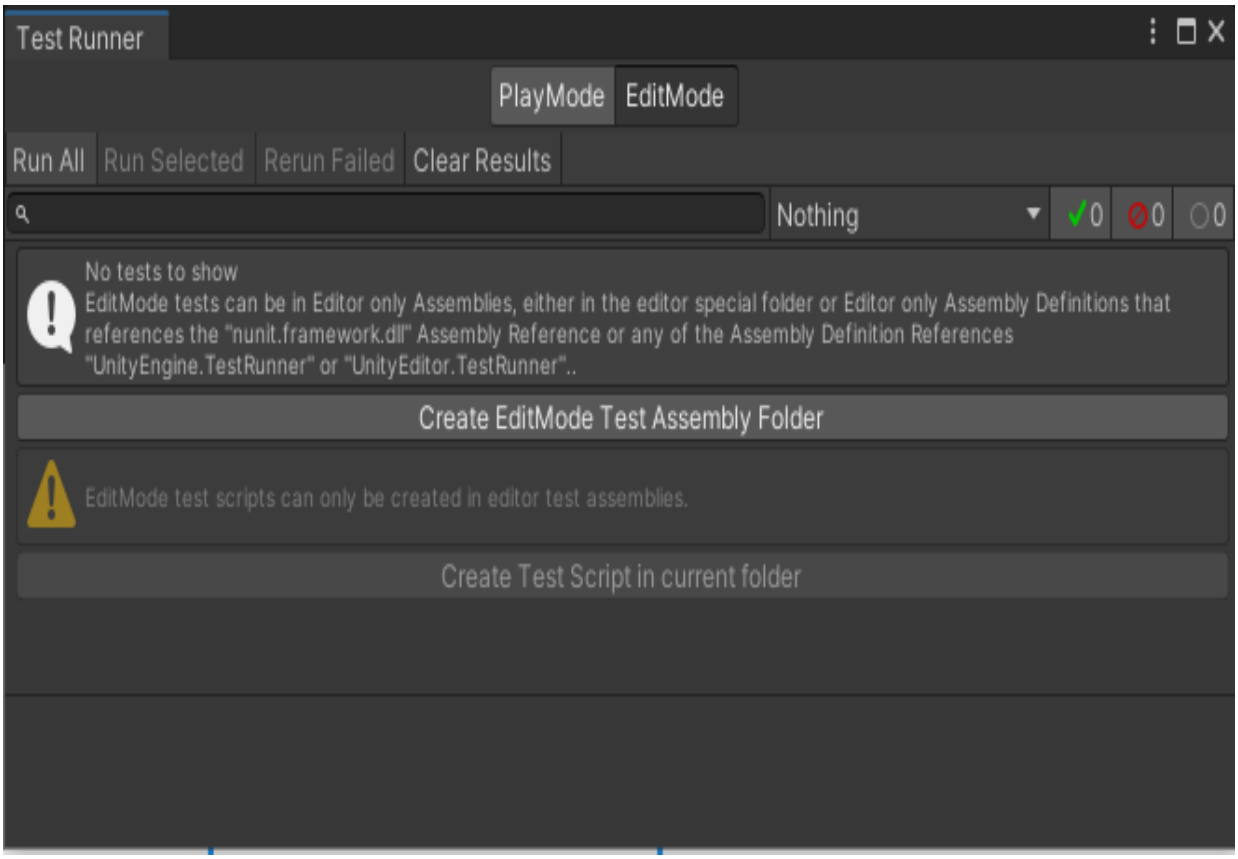

Рисунок 51 – Окно Test Runner

Помимо запуска UTF в режиме игры в редакторе Unity, новая точка настройки позволяет запускать тест на целевых устройствах. Она вызывается до сборки и позволяет изменить её настройки. Например, изменить настройки запуска тестов и указать расположения сборок.

В магистерской диссертации было использовано модульное (блочное) тестирование. В ходе тестирования были проведены 5 тестов различных механик игрового приложения. Во-первых, была проверена правильность заполнения карты высот для генерации игрового уровня. Во-вторых, выполнено тестирование функции очищения ландшафта представленной сцены для того, чтобы не появилась ошибка переполнения буфера массива данных или возникла вероятность

наложения друг на друга игровых уровней. Далее была проведена проверка корректности работы функции волн для корректного отображения водной поверхности.

Также дополнительно были написаны юнит тесты для проверки изменения звуковых настроек и возможность перехода и запуска игровой сессии из главного меню, что позволяет избежать критических ошибок при запуске игрового 3D-приложения на платформе Unity.Успешность проведенного тестирования игровых механик подтверждена результатами работы системы Test Runner (Рисунок 52).

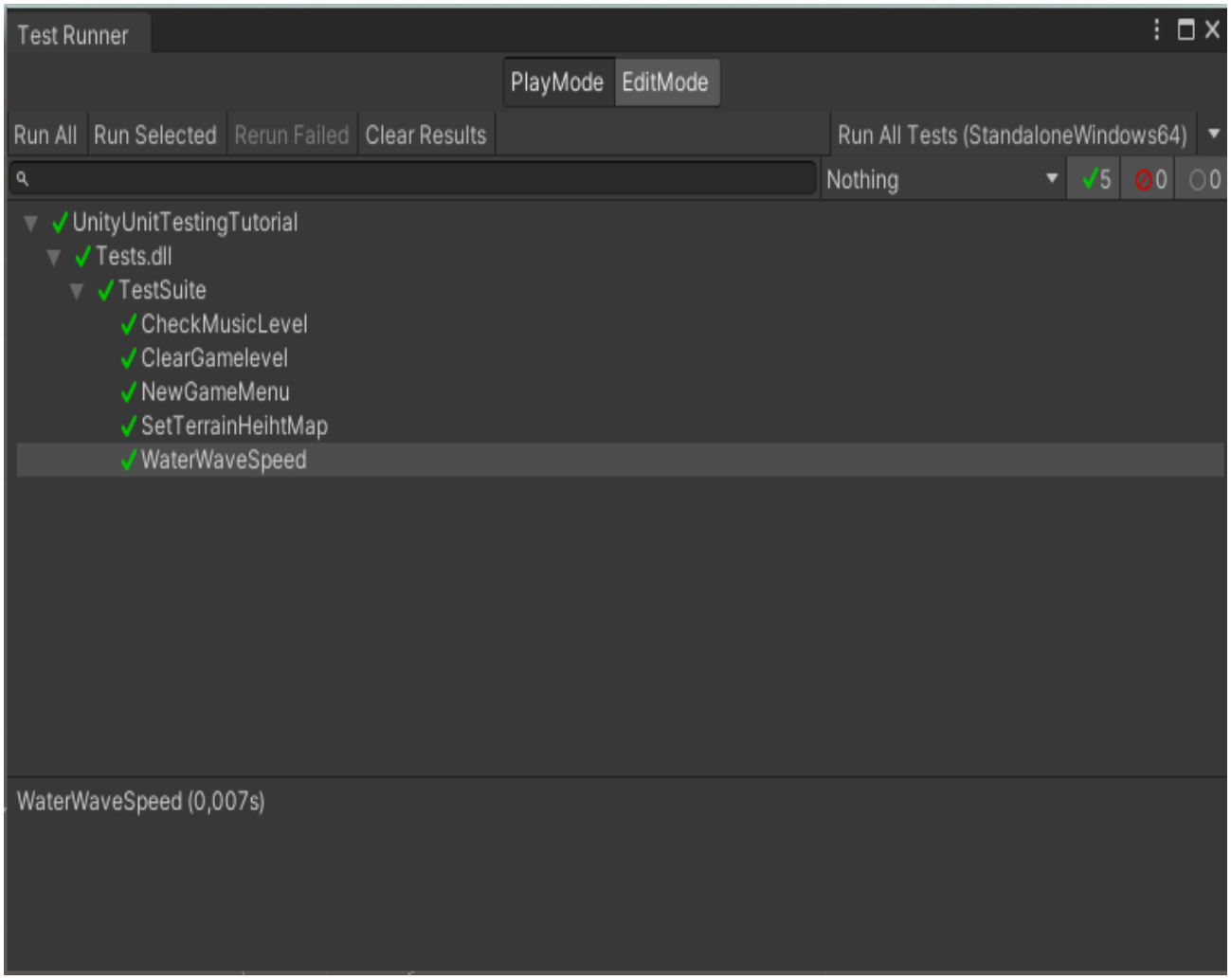

Рисунок 52 – Окно протестированных механик видеоигры

Проведенное тестирование показало, что система тестов Unity имеет достаточно большую функциональность и обладает кроссплатформенностью.

Также есть возможность формирования автоматических ссылок и преимущества использования GUIDs. Файлы тестов ссылаются с помощью ассета Assembly Definiton находящегося в директории с тестируемыми скриптами (Рисунок 53).

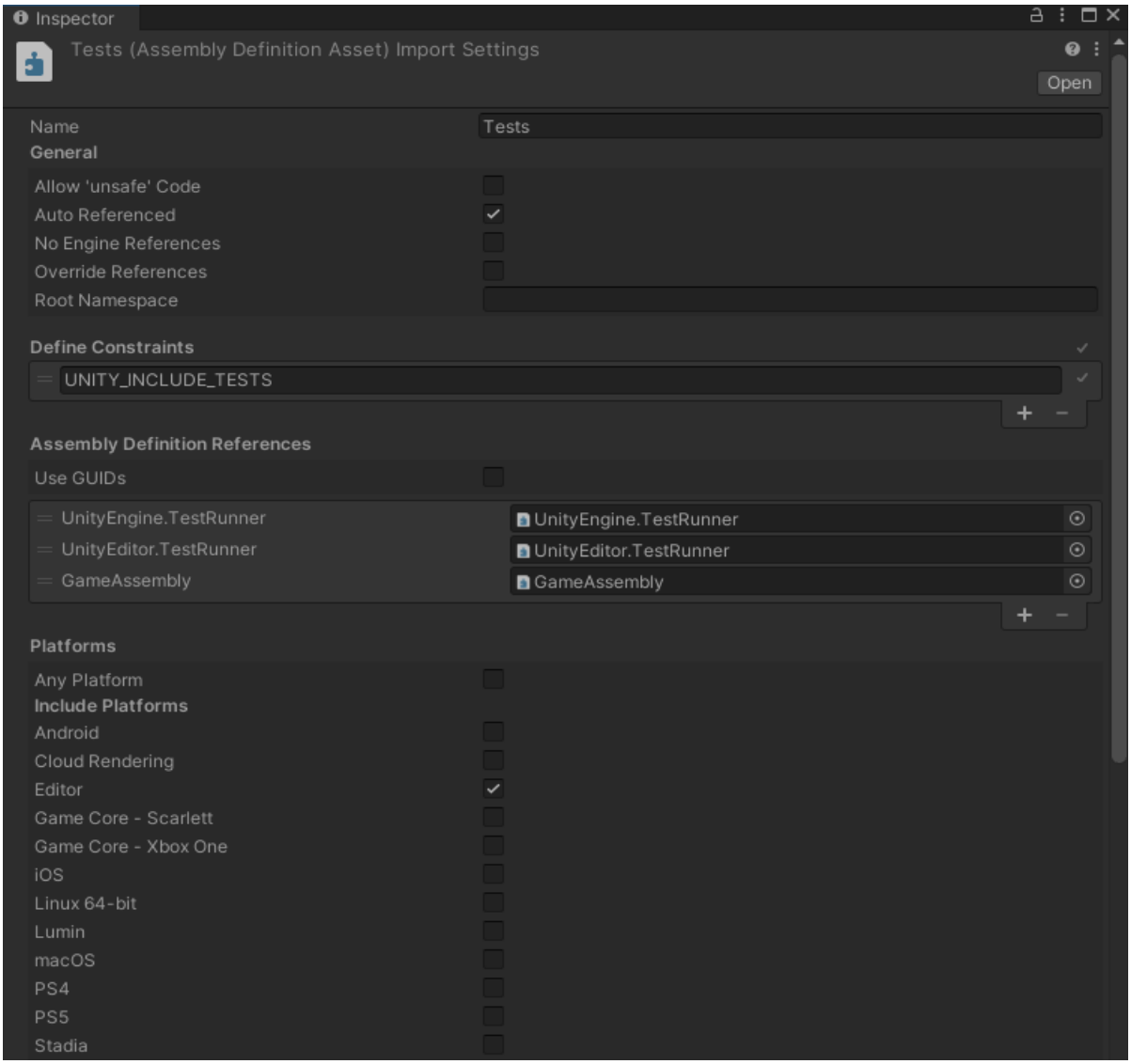

Рисунок 53 – Окно инспектора Assembly Definition

Таким образом, проведено частичное тестирование игровых механик генератора игрового уровня и главного меню видеоигры. И поэтому действия разработчика или пользователя не станут причиной критических ошибок и выхода видеоигры из рабочего состояния.

3.3 Анализ достоверности и практическая значимость результатов

Достоверность результатов, полученных в диссертационной работе обусловлена приведенными теоретическими сведениями об процедурной генерации

и корректностью математических преобразований. Дополнительно подтверждена практическими результатами разработанной системы генерации ландшафтов. Достоверность полученных результатов также обусловлена их обсуждениями на научных конференциях.

Практическая значимость результатов выпускной квалификационной работы заключается в том, что в ходе исследования будут применены новые подходы по разработке игровых приложений. В ходе реализации игрового приложения была разработана система генерации окружения, которую можно усовершенствовать, добавив другие методы генерации. Таким образом с помощью разработанного инструмента можно легко и быстро создавать игровые уровни и легко внедрять его в другие Unity проекты и это сильно упрощает и ускоряет процесс разработки. В магистерской диссертации также разработаны уникальные игровые механики построения различных игровых сценариев, а также описаны алгоритмы и реализация их работы.

ЗАКЛЮЧЕНИЕ

В рамках выполнения магистерской диссертации был создан прототип игрового приложения в жанре выживание. В ходе выполнения работы были получены следующие результаты:

− выявлены тенденции развития игровой индустрии в области применения процедурной генерации и разобраны примеры ее применения в современных популярных видеоиграх;

− определены 3 существующие группы методов процедурной генерации, разобраны преимущества и их недостатки и выбраны оптимальные для реализации создания системы окружения игрока –это алгоритмы генерации карты высот и процедурный шум;

− получены и применены алгоритмы diamond-square и шум Перлина для решения задачи генерации системы окружения игрока. Также были спроектированы восемь функциональных модулей и выполнен архитектурный проект игрового приложения;

− проведено обоснование выбора программного средства разработки игрового приложения − среда Unity версии 2021.3.7f1;

− реализован прототип игрового приложения, а именно система генерации предметов и игровых уровней. Создан игровой персонаж авантюрист, а также разработано главное меню игрового приложения;

− проведено частичное модульное тестирование отдельных механик игрового уровня на корректность заполнения карты высот и выполнения функции очистки игровой локации. Проверен на работоспособность шейдер волн водной поверхности, а также возможность запуска игровой сессии из главного меню.

В ходе реализации игрового приложения была разработана система генерации окружения, которую можно усовершенствовать, добавив другие методы генерации. Таким образом с помощью разработанного инструмента можно легко

и быстро создавать игровые уровни и легко внедрять его в другие Unity проекты и это сильно упрощает и ускоряет процесс разработки.

В настоящее время работа над прототипом игрового приложения завершена, но в дальнейшем можно расширить функционал приложения.

На основе исследовательской работы сделаны 3 публикации.

Подана заявка на официальную регистрацию программы для ЭВМ.

Результаты работы прошли апробацию на XXIII региональной научно-практической конференции Молодежь XXI века: шаг в будущее (г. Благовещенск, 24 мая 2022 года).

БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1 Global Games Market Report [Электронный ресурс] / URL: https://newzoo.com/products/reports/global-games-market-report (дата обращения 18.06.2023)

2 Annual Report [Электронный ресурс] / URL: https://www.tencent.com/en-us/investors/financial-reports.html (дата обращения 16.06.2023)

3 Short, T. X. Procedural Generation in Game Design / T. X. Short, T. Adams. – A. K. Peters, Ltd. 63 South Avenue Natick, MA United States, 2017. – 336 p.

4 Гибсон, Дж. Б. Unity и C#. Геймдев от идеи до реализации / Дж. Б. Гибсон – СПб: Питер, 2019. – 928 с.

5 Вейцман, В. М. Проектирование информационных систем: учебное пособие / В. М. Вейцман. – Санкт-Петербург: Лань, 2019. – 316 с. – ISBN 978-5-8114-3713-9. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/122172. (дата обращения 14.06.2023)

6 Ламот, А. Основы разработки игр на Unity 3D / А. Ламот, Д. Ратклифф, М. Семинаторе, Д. Тайлер – СПб: Питер, 2018. – 716 с.

7 Гейг, М. Разработка игр на Unity 2018 за 24 часа/ М. Гейг – М: Бомбора, 2020. – 464 с.

8 Шейдеры и эффекты в Unity. Книга рецептов – Кенни Ламмерс [Электронный ресурс]. – РЕЖИМ ДОСТУПА: https://www.litres.ru/book/kenni-lammers/sheydery-i-effekty-v-unity-kniga-receptov-22998602/ – 30.05.2023

9 Cinemachine [Электронный ресурс]. URL: https://learn.unity.com/tutorial/cinemachine# (дата обращения: 14.06.2023).

10 Unity UI: Unity User Interface [Электронный ресурс]. URL: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/index.html (дата обращения: 20.06.2023).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Адамс, Т. Процедурная генерация в гейм-дизайне / Т. Адамс, Т. Шорт; под редакцией Тани Х. Шорт и Тарна Адамса; перевод с английского М. С. Рыжиковой. – Москва: ДМК Пресс, 2020. – 344 с. – ISBN 978-5-97060-860-9. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/190739 (дата обращения: 01.06.2023)

2 Берг, Д.Б. Модели жизненного цикла: учебное пособие / Д.Б. Берг, Е. А. Ульянова, П. В. Добряк. – Екатеринбург: Изд-во Урал. ун-та, 2014. – 74 с.

3 Брауде, Э. Технология разработки программного обеспечения: пер. с англ. / Э. Брауде – СПб.: ПИТЕР, 2014. – 655 с.

4 Буч, Г. UML. Классика CS / Г. Буч, Д. Рамбо, И. Якобсон. – 2-е изд. – СПб.: «Питер», 2016. – 736с.

5 Васильев, А.Н. Программирование на C# для начинающих. Основные сведения / А.Н. Васильев – М. : Бомбора, 2022. – 592 с.

6 Вейцман, В. М. Проектирование информационных систем: учебное пособие / В. М. Вейцман. – Санкт-Петербург: Лань, 2019. – 316 с. – ISBN 978-5-8114-3713-9. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/122172. (дата обращения 14.06.2023)

7 Гейг, М. Разработка игр на Unity 2018 за 24 часа/ М. Гейг – М: Бомбора, 2020. – 464 с.

8 Гибсон, Дж. Б. Unity и C#. Геймдев от идеи до реализации / Дж. Б. Гибсон – СПб: Питер, 2019. – 928 с.

9 Грекул, В. И. Проектирование информационных систем: учебное пособие / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. – 3-е изд. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. – 299 с.

10 Джейсон, Г. Игровой движок. Программирование и внутреннее устройство / Г. Джейсон – СПб: Питер, 2018. – 1136 с.

11 Дикинсон, К. Оптимизация игр в Unity 5. Советы и методы оптимизации игровых приложений / К. Дикинсон Google-Books-ID: 5c1SEAAAQBAJ. – Litres, 2022. – 308 с.

12 Котов, О. М. Язык C#. Краткое описание и введение в технологии программирования: учебное пособие / О. М. Котов. – Екатеринбург: Уральский федеральный университет, ЭБС АСВ, 2014. – 208 с

13 Кузеванов, К.И., Научный руководитель – Галаган Т.А. Разработка игрового 3D-приложения с процедурной генерацией окружения // Молодежь XXI века: шаг в будущее: материалы XXII региональной научно-практической конференции (24 мая 2022 года). Материалы отправлены.

14 Кузеванов, К. И. Свидетельство о государственной регистрации программы для ЭВМ №2921661323 игровое приложение “Исследователь подземелий” дата поступления 06.07.2021

15 Кузеванов, К.И., Научный руководитель – Галаган Т.А. Применение технологий среды Unity для разработки пользовательского интерфейса игрового приложения // «Студенческий вестник»: научный журнал. – №13(252). Часть7. Москва, Изд. «Интернаука», 2023. – 72с.

16 Леоненков, А. Самоучитель UML / А. Леоненков – 2-е изд., перераб и доп. – СПб.: БХВ-Петербург, 2014. – 432 с.

17 Ламот, А. Основы разработки игр на Unity 3D / А. Ламот, Д. Ратклифф, М. Семинаторе, Д. Тайлер – СПб: Питер, 2018. – 716 с.

18 Нистром, Р. Шаблоны игрового программирования. / Р. Нистром – М: ДМК, 2018. – 72 с.

19 Никулин, Е. А. Компьютерная графика. Модели и алгоритмы: учебное пособие / Е. А. Никулин. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 708 с. – ISBN 978-5-8114-2505-1. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/213038 (дата обращения: 01.06.2023).

20 Орлов, С.А. Технологии разработки программного обеспечения /

С.А. Орлов. – СПб.: ПИТЕР, 2012. – 464 с.

21 Советов, Б. Я. Информационные технологии: теоретические основы: учебное пособие / Б. Я. Советов, В. В. Цехановский. – 2-е изд., стер. – Санкт-Петербург: Лань, 2021. – 444 с.

22 Сиротина, И.К. Методы Оптимизации Графических Файлов Видеоигр / И.К. Сиротина, А.А. Мышковский, К.Э. Остапчук // Актуальные Научные Исследования В Современном Мире. – 2021. – № 2 – 2 (70).

23 Токманцев, Т. Б. Алгоритмические языки и программирование: учебное пособие для СПО / Т. Б. Токманцев; под редакцией В. Б. Костоусова. – 2-е изд. – Саратов, Екатеринбург: Профобразование, Уральский федеральный университет, 2019. – 102 с.

24 Троелсен, Э. Язык программирования C# 7 и платформы .NET и .NET Core : пер. с англ. / Э. Троелсен, Ф. Джепикс. – 8-е изд. – СПб.: ООО “Диалектика”, 2018. – 1328 с.

25 Торн, А. Искусство создания сценариев в Unity. / А. Торн – М: ДМК, 2016. – 360с.

26 Торн, А. Основы анимации в Unity: учебное пособие / А. Торн – М: ДМК, 2016. – 176 с.

27 Хокинг, Дж. Unity в действии. Мультиплатформенная разработка на C# учебное пособие/ Дж. Хокинг - СПб: Питер, 2016. – 336с.

28 Шампандар, А.ДЖ. Искусственный интеллект в компьютерных играх / А.ДЖ. Шампандар – М: Вильямс, 2018. – 383 с.

29 Шейдеры и эффекты в Unity. Книга рецептов – Кенни Ламмерс [Электронный ресурс]. – РЕЖИМ ДОСТУПА: https://www.litres.ru/book/kenni-lammers/sheydery-i-effekty-v-unity-kniga-receptov-22998602/ – 30.05.2023

30 Шарыгин, А.Ю. Основные Процессы Оптимизации Мобильной Игры На Unity / А.Ю. Шарыгин, С.П. Кайгордов, С.П. Кайгородов // Аллея Науки. – 2018. – Т. 5. – № 5 (21).

31 Annual Report [Электронный ресурс] / URL: https://www.tencent.com/en-us/investors/financial-reports.html (дата обращения 16.06.2023)

32 Animation and Virtual Worlds. – 2018. – Vol. 29. – № 2. – P. e1801.

33 Bosman, F. G. There Is No Solution!: “Wicked Problems” in Digital Games. // Games and Culture. – 2019. – №5. – С. 543-559.

34 Costello, B. M. The Rhythm of Game Interactions: Player Experience and Rhythm in Minecraft and Don’t Starve // Games and Culture. - 2018. - №8. - С. 807-824.

35 Fluid-flow modeling and stability analysis of communication networks: 20th IFAC World Congress / N. Espitia [et al.] // IFAC-PapersOnLine. – 2017. – Vol. 50. – № 1. – P. 4534-4539.

36 Goldstone, W. Unity Game Development Essentials/ W. Goldstone – UK, Birmingham: Packt, 2017. – 266 с.

37 Galin E., Guérin E., Peytavie A., Cordonnier G., Cani M. – P., Benes B., Gain J. A Review of Digital Terrain Modeling // Computer Graphics Forum. – 2019. – №2. – С. 553-577.

38 Global Games Market Report [Электронный ресурс] / URL: https://newzoo.com/products/reports/global-games-market-report (дата обращения 18.06.2023)

39 Journal I. IRJET- A GPU Parallel Implementation of Bitonic Sort using CUDA / I. Journal // IRJET. – 2021.

40 Lavoue G., Dupont F., Baskurt A. Toward a near optimal quad/triangle subdivision surface fitting // Fifth International Conference on 3-D Digital Imaging and Modeling. – 2015. – С. 402-409

41 OMG Unified Modeling Language (OMG UML), Infrastructure Version 2.2 [Электронный ресурс]: офиц. сайт. – 2009 – Режим доступа: https://www.omg.org/spec/UML/2.2/Superstructure/PDF – 20.02.2023

42 Price M.J. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core,

and ML.NET using Visual Studio Code, 4th Edition. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development / M.J. Price Google-Books-ID: Qzm8DwAAQBAJ. – Packt Publishing Ltd, 2019. – 819 p.

43 Short, T. X. Procedural Generation in Game Design / T. X. Short, T. Adams. – A. K. Peters, Ltd. 63 South Avenue Natick, MA United States, 2017. – 336 p.

44 Selby, R.W. Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research / R.W. Selby – Нью-Йорк: John Wiley & Sons, 2017. – 834 с.

45 Technologies U. Unity - Manual: Physics [Электронный ресурс]. – РЕЖИМ ДОСТУПА: https://docs.unity3d.com/Manual/PhysicsSection.html – 30.05.2023.

46 Thalmann D., Musse S. R. Crowd Simulation. – Berlin, Heidelberg: Springer-Verlag, 2007. – 254 с.

47 Technologies U. Батчинг вызовов отрисовки (Draw Call Batching) – Unity Manual [Электронный ресурс]. – РЕЖИМ ДОСТУПА: https://docs.unity3d.com/ru/2019.3/Manual/DrawCallBatching.html – 21.05.2023.

48 Technologies U. Unity - Manual: Shaders [Электронный ресурс]. – РЕЖИМ ДОСТУПА: https://docs.unity3d.com/Manual/Shaders.html – 30.05.2023).

49 Unity–Scripting API [Электронный ресурс] / Unity Technologies.– Режим доступа:https://docs.unity3d.com/ScriptReference (дата обращения 11.06.2023)

50 Unity UI: Unity User Interface [Электронный ресурс]. URL: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/index.html (дата обращения: 11.06.2021).

ПРИЛОЖЕНИЕ А

Техническое задание

1.Введение.

1.1 Тема: Разработка игрового приложения “Искажение реальности” на платформе Unity.

1.2 Жанр видеоигры, перспектива: Приключение, боевик, выживание. Вид от третьего лица.

1.3 Режим видеоигры:

– Однопользовательский.

– Кооперативный (опционально).

1.4 Целевая Аудитория:

Видеоигра рассчитана на аудиторию 16-25 лет, предпочитающую активный игровой процесс и любителей видеоигр с уклоном на выживание с максимально возможным игровым временем. Возможность сравнить свои результаты с другими игроками.

1.5 Описание основной идеи Видеоигры:

В видеоигре авантюристу предстоит исследовать загадки и другие тайны различных локаций с помощью алтарей, разбросанных по игровому миру.

1.6 Цель видеоигры:

Игроку необходимо пройти как можно больше игровых уровней и собрать наибольшее количество артефактов.

1.7 Описание главного героя:

Игроку предстоит взять на себя роль храброго авантюриста, который исследует различные загадочные места мира.

2 Основание для разработки

2.1. Основанием для разработки данной работы является потребность разработчиков на платформе Unity в довольно гибком инструменте создания внешнего наполнения игрового приложения (ландшафт, растительность, здания и т.д.) и внутреннего (игровые предметы, персонажи, противники и другие игровые объекты)

3. Исполнители: студент группы 157-ом Кузеванов Кирилл Иванович

4. Соисполнители: нет

3 Назначение разработки

3.1 Создание модуля генерации игрового окружения для игрового приложения “Искажение реальности” на платформе Unity

4 Технические требования

4.1 Требования к функционалу видеоигры.

4.1.1 Цели игрока:

– Ориентироваться в игровой локации.

– Защитить себя от игровых противников.

– Подбирать различные игровые предметы или оружие для усиления игрока.

– Собирать различные осколки для открытия доступа к следующей локации.

– Собирать сферы здоровья для лечения персонажа.

4.1.2 Требования к наполнению игровой карты

Игровое наполнение игрового уровня разделяется на внешнее и внутреннее.

Внешнее наполнение карты должно состоять из:

– Ландшафт.

– Растительность (деревья, камни)

– Водная поверхность.

Внутреннее наполнение карты состоит из: (сферы здоровья оружие, артефакты)

Игровой персонаж управляемой игроком.

Противники (дикие животные, одержимые, демоны).

Игровые предметы, влияющие на игрока (сферы здоровья, оружие, предметы усиления, артефакты).

Игровые предметы необходимые для продвижения к следующему уровню (осколки алтаря, алтарь).

4.1.2.1 Логика персонажа

4.1.2.3 Логика предметов необходимая для продвижения к следующему уровню. Игрок может подобрать один из нескольких типов осколков, разбросанных по игровой карте и вставить их в портал. Существует несколько типов осколков. Ледяной, огненный и пустынный. От типа осколка, вставленного в алтарь, зависит в какой именно игровой уровень перейдет персонаж.

4.6 Требования к системным параметрам и программной совместимости

– Компьютер должен быть оснащен двухъядерным процессором не менее 2GHs частоты.

– 2 Гигабайта оперативной памяти

– Графический ускоритель не менее чем с 512 мегабайтами видеопамяти с поддержкой DirectX 11 и выше.

– Места на жестком диске или твердотельном накопителе не менее чем с 5 гигабайтами памяти.

– Операционная система Windows 10,11

4.7 Требования к интерфейсу

Игровой интерфейс должен состоять из главного меню и меню паузы. Интерфейс должен быть максимально дружественен пользователем любой квалификации (в плане компьютерной грамотности).

Главное меню содержит:

– Раздел играть в котором непосредственно можно начать или продолжить игру.

– Раздел настроек (громкость, настройки графической составляющей, настройки управления персонажем.)

– Раздел Помощи (раздел справки).

– Кнопка выход.

Меню паузы содержит:

– Кнопка возобновить игру.

управления персонажем.).

– Кнопка выход (возвращает к главному меню видеоигры).

4.8 Требования к звуковой составляющей:

У каждого персонажа, противника и к игровым предметам (сферы здоровья, оружие, предметы усиления) должен быть прикреплён соответствующий звуковой файл формата mp4.

4.9 Требования к визуальной составляющей:

В разрабатываемой видеоигре используется система URP (Universal Render Pipline)

4.10 Требования к элементам управления ввода-вывода в видеоигре:

– Для управления игроку необходима клавиатура и компьютерная мышь.

– Назначения клавиш для движения (Стрелки на клавиатуре и кнопки WASD).

Специальный Клавиши:

– Прыжок (кнопка space).

– Взаимодействие с предметами (клавиша E).

– Правая кнопка мыши.

– Левая кнопка мыши.

5 Требования к программной документации:

Основными документами, регламентирующими разработку, должны быть документы Единой Системы Программной Документации, а также документация программного комплекса Unity.

6 Технико-экономические показатели

Эффективность игрового приложения можно определить числом пользователей использующими данное ПО, а также экономической выгодой, полученной после выпуска видеоигры на игривых площадках Steam, Epic Games и др.

7 Порядок Релиза и поддержки видеоигры

После релиза разработчиком игрового приложения, пользователи имеют право имеют право сообщать об ошибках, возникших в ходе использования игрового приложения, а также выдвигать предложения по улучшению данного программного обеспечения. Заявки пользователем следует оформлять в электронном виде и посылать на электронную почту kub203@mail.ru. Разработчик обязан в течение всего срока поддержки игрового приложения выпускать патчи, дорабатывающие программное обеспечение.

Наименование этапа	Сроки этапа	Результат выполнения этапа
1. Изучение предметной области	01.01.2023-30.01.2023	Выбраны идея, сеттинг жанр видеоигры, а также средства разработки. Выполнено проектирование функциональных модулей и составлены схемы вариантов использования системы.
2. Разработка ПО игрового приложения	01.02.2023-01.07.2023	Завершено создание игровых механик, визуальной составляющие а также игровых уровней игрового приложения
3. Тестирование и отладка игрового приложения	02.05.2023-30.08.2023	Созданы и отлажены альфа и бета версии игрового приложения
4. Релиз и поддержка видеоигры	01.09.2023-30.12.2023	Готовое игровое приложение. Руководство пользователя. Выпуск патчей для дальнейшей поддержки видеоигры