МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Доцент кафедры экологии
и химической технологии ИЕТН
_____ О.Ю. Ленская
“13” июня 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор
_____ Л.Б. Соколинский
“___”_____ 2020 г.

Разработка компьютерной 3D – игры в жанре «Action» на Unreal Engine 4

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2020.308-545.ВКР

Научный руководитель,
д.г.н., к.ф.-м.н.
профессор кафедры СП,
_____ С.М. Абдуллаев

Автор работы,
студент группы КЭ-402
_____ Е.О. Савченко

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
“___”_____ 2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____Л.Б. Соколинский
09.02.2020

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-402 Савченко Евгению Олеговичу,
обучающемуся по направлению 02.03.02
«Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 24.04.2020 № 627)

Разработка компьютерной 3D - игры в жанре «Action» на Unreal Engine 4.

2. **Срок сдачи студентом законченной работы:** 05.06.2020.
3. **Исходные данные к работе**

3.1. Blueprint https://uengine.ru/site-content/docs/blueprints-docs/blueprint.

3.2. World Composition https://uengine.ru/site-content/docs/landscape/world-composition.

3.3. Интернет-портал Unreal Engine 4.Documentation // https://docs.unrealengine.com/latest/INT (дата посещения 07.05.2020).

4. **Перечень подлежащих разработке вопросов**

4.1. Анализ предметной области.

4.2. Провести сравнительный анализ существующих аналогов.

4.3. Выбор средств реализации.

4.4. Написание технического предложения.

4.5. Создание 3D моделей персонажей и элементов окружения.

4.6. Написание визуального скриптинга, сборка Block Out.

4.7. Провести функциональное тестирование.

5. Дата выдачи задания: 08.02.2020.

Научный руководитель
д.г.н., к.ф.-м.н.,
профессор кафедры СП С.М. Абдуллаев

Задание принял к исполнению Е.О. Савченко

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

Видеоигры сейчас очень популярны. Многие люди погружаются в видеоигровой мир, чтобы избежать скучной реальности примеряя роль героя игры, сражаясь с ужасными монстрами или спасая мир от враждебных инопланетных форм жизни. Видеоигры дают возможность окунуться в захватывающие приключения: исследовать сложные лабиринты, преодолевать препятствия, сражаться со страшными драконами, управлять различными транспортными средствами, пилотировать космический корабль или управлять самолетом, строить города, решать различные головоломки, играть в спортивные игры и даже ухаживать за виртуальным питомцем.

Видеоигры постоянно становятся более живыми и сложными. Появляются новые стили и жанры. Разработка компьютерной игры требует работы и навыков довольно большой группы людей, включая программистов, графических дизайнеров, звукорежиссеров, музыкантов и других техников. Современные видеоигры содержат уникальный синтез 3D-искусства, звуковых эффектов, реальной среды, людей, похожих на персонажей, архитектуры, искусственного интеллекта, драматических представлений, музыки, повествования историй и интерактивности.

Разработчики видеоигр обвиняются в изображении графического насилия, сексуальных тем, употребления наркотиков, алкоголя или табака, плохого языка, пропаганды или ненормативной лексики в некоторых играх. Многие игры поощряют агрессивное поведение, стирают разницу между правильным и неправильным и вызывают зависимость. Некоторые люди утверждают, что видеоигры делают детей необщительными и пассивными. Они поглощены играми, не разговаривая друг с другом в течение длительного времени. Дети и подростки, увлекающиеся компьютерными играми, не заинтересованы в чтении. Некоторые средства массовой информации заявляют, что компьютерные игры тормозят развитие головного мозга. Дети могут часами играть без еды, сна или обучения.

Но видеоигры также имеют свои преимущества. Большинство игр требуют большого терпения и сосредоточенности от игрока, поэтому видеоигры могут даже увеличить возможности внимания игроков. Они также могут улучшить зрительные навыки ребенка и развить координацию. Определенные типы видеоигр могут улучшить ловкость игроков, а также их способность решать проблемы. [6]

Видеоигры предназначены не только для развлечений. Некоторые видеоигры сделаны по другим причинам. Развивающие игры пытаются научить игроков использовать игру как средство самообразования. Они помогают детям развивать математику, навыки чтения и правописания посредством игры. Компьютерные игры могут способствовать развитию навыков стратегического мышления и планирования. Было замечено, что геймеры не понимают, что они учатся. Так что, если в школе можно будет использовать развивающие игры, образование получит значительные преимущества.

В данной работе реализуется одно из возможных типов повествования сюжета, игровой механикой.

Цель и задачи

Разработка компьютерной 3D-игры в жанре action на Unreal Engine 4.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- осуществить анализ предметной области;
- определить требования к системе, проведя сравнительный анализ существующих аналогов;
- написание технического предложения;
- создание 3D моделей персонажей и элементов окружения;
- реализовать игровое приложение на основе проанализированных требований;
- провести функциональное тестирование полученного приложения.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения и библиографического списка. Объем работы составляет 54 страницы, объем библиографии – 15 источников.

В главе «Теоретическая часть» описана постановка задачи, произведен обзор существующих аналогов приложения и оптимальных способов реализации.

Глава «Проектирование приложения» посвящена определению требований к разрабатываемому приложению. В этой же главе описываются: диаграмма вариантов использования, диаграмма классов, проектирование интерфейса приложения.

Третья глава посвящена «Реализации приложения», в ней описаны компоненты и структура реализованного приложения, реализация базовой логики приложения при помощи BluePrint, реализация интерфейса.

В главе «Тестирование» описаны результаты функционального тестирования и тестирование интерфейса.

В заключении сделаны выводы о проделанной работе и сформулированы перспективы дальнейшей разработки.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Анализ предметной области

Перед началом проектирования этапов разработки игры надлежит провести анализ предметной области, тем самым акцентировать основные моменты во избежание недочётов при составлении требований к проектируемому проекту.

Для начала необходимо дать определение жанру action и определить его отличие от action-adventure акцент который делается в данном проекте.

В action-играх обычно игрок управляет протагонистом цель которого найти из уровня выход, собрать необходимые предметы, избежать опасных стычек или сразиться с врагами разными способами. В конце уровней игрок обычно сражается с боссом, где битва более требовательна к игроку, а сам босс, часто, крупнее обычных врагов. Действие игр подобного жанра проходят динамично и требует внимательности, а также быстрой реакции на происходящие в игре повороты сюжета. К жанру может быть отнесена любая игра, в которой победа над соперником обеспечивается благодаря физическому превосходству, лучшим прицеливанием или меньшем временем реакции. [7]

Action-adventure является смешанным жанром видеоигр, переплетением элементов квеста и экшена который предлагают игроку преодолевать препятствия как к примеру интеллектуального рода, так и физического. Определение того, когда такая игра перестает быть квестом и превращается в чистую action-игру, является лишь вопросом интерпретации.

К элементам квеста могут принадлежать головоломки сюжет, многочисленные персонажи, диалоги между ними, инвентарь для собранных предметов и другие приметы жанра квестов, по сравнению с квестами action-adventure в большинстве своем опираются на перемещение персонажа в мире игры — это относится как игровому процессу, в котором могут преобладать сражения, так и к сюжету: перемещение заставляет запускаться сюжетные сцены, в соответствии с которыми меняется темп игры и

контекст действий игрока. Вид от третьего лица является довольно распространенным в играх жанра action-adventure; по сравнению с чистыми action-играми, квестовые элементы в action-adventure требуют более сложного поведения камеры и более сложного устройства игровых уровней. Это одна из тех разновидностей видеоигр, которая обязательно нуждается в присутствии сюжета, а не только геймплея.

1.2. Анализ аналогичных проектов

Рассмотрим аналогичные проекты.

Silent Hill 4: The Room (2004 г.) [12].

Мультиплатформенная игра в жанрах survival horror и action-adventure, действие игры разворачивается в городе Эшфилд, расположенным неподалеку от Сайлент Хилла, происходит все в реальном мире и параллельном мире сна, и сосредоточено на главном герое Генри, пытающемся сбежать из запертой квартиры. Скриншот игры представлен на рисунке 1.

Рис. 1. Скриншот игры Silent Hill 4 The Room

Сюжет игры строится вокруг посещений параллельного мира, в который Генри попадает через туннель в стене, появившийся в ночь после кошмара, в ванной его квартиры. Погрузившись в потусторонний мир квартиры, герой пытается осмыслить что происходит, спасаясь от призраков параллельного мира попутно следуя по следам серийного убийцы.

Игровой процесс состоит из исследования миров, поиска предметов, решений головоломок, противостоянию Генри кошмарам другого мира.

Subnautica (2018 г.) [14].

Компьютерная приключенческая игра, симулятор выживания с открытым миром, игрок может свободно исследовать океан другой планеты, управляя единственным выжившим при крушении космического корабля Аврора. Скриншот игры представлен на рисунке 2.

Рис. 2. Скриншот игры Subnautica

Death Stranding (2019 г.) [13].

Компьютерная игра в жанре action - adventure с открытым миром. Сюжет игры разворачивается в будущем на территории США, разрушенных вторжением призрачных существ из другого мира. Игрок управляет курьером Сэмом — своего рода постапокалиптическим почтальоном; пе-

ред ним стоит задача пересечь континент и связать друг с другом изолированные поселения. Скриншот игры представлен на рисунке 3.

Рис. 3. Скриншот игры Death Stranding

Игра совмещает в себе свободное путешествие по открытому миру, где игрок должен самостоятельно находить путь к следующей цели, и продолжительные кинематографические катсцены, созданные на движке игры с помощью технологии захвата движения.

Основной игровой процесс связан с переноской ценных посылок из одного поселения в другое, причём игрок должен избегать потери и повреждения груза, игра содержит необычайно проработанную механику перемещения по пересечённой местности и элементы асинхронной многопользовательской игры.

Вывод

В результате обзора аналогов можно сделать вывод, что данное направление жанра динамично развивается, вкладывая все больше повествования сюжета и динамики не только на уровне поисков документации по игровому миру и уничтожением противников но и повествует стилистиче-

ским подходом дизайна игрового мира где окружающая сцена может передать больше атмосферы и сюжета происходящего чем прочитанный документ так и динамичным переходом боя где противники уже не ждут своей очереди чтобы нанести урон протагонисту.

1.3. Обзор средств реализации приложения

В настоящее момент для разработки игр существует множество различных, а главное легко доступных средств разработки видеоигр со своими плюсами. Для реализации видеоигры необходимо выбрать наиболее сбалансированную и оптимальную платформу с удобным интерфейсом, на сегодняшний день не существует проблем с поиском документации по эксплуатации и получению бесплатной лицензии игрового движка.

В ходе выполнения работы было рассмотрено несколько примеров программного обеспечения для создания видеоигр:

Unity

Кроссплатформенная среда разработки игровых приложений от компании Unity Technologies. На Unity написано огромное количество игровых приложений, работающих под более чем 20 различными ОС. При создании приложений используется компонентно-ориентированный подход. Хорошо подходит для создания средних по сложности проектов. Основной язык разработки – C#. Выпуск Unity состоялся в 2005 году и активно развивается до сих пор.

Достоинства:

- низкий порог вхождения;
- бесплатная лицензия;
- модульная система компонентов.

Unreal Engine 4

Кроссплатформенная среда разработки от компании Epic Games. Изначально предназначался для разработки игр в жанре «шутер от первого лица», но в дальнейшем был адаптирован под всевозможные виды жанров.

Бесплатный для всех разработчиков, начиная с 2015 года. В Unreal Engine входят: управление сетевой и файловой системой, искусственный интеллект, движок обеспечения физики и графики.

Основной язык разработки – C++, также имеет встроенный редактор визуального программирования удобный для реализации базовой логики – BluePrint.

Достоинства:

- высокая производительность;
- бесплатная лицензия;
- визуализация элементов в реальном времени;
- открытый исходный код.

Godot Engine

Открытый кроссплатформенный игровой движок. Движок предоставляет разработчикам возможность создавать игры, не пользуясь никакими инструментами, кроме тех, что необходимы для разработки игрового контента, музыкальных треков, элементов графики и т.д. Процесс разработки и программирования не требует внешних инструментов, но при необходимости можно использовать внешний редактор, это можно легко сделать.

Вывод

Анализ предметной области позволил определиться с использованием методов. Таким образом, детальный анализ аналогичных игровых движков помог выявить наиболее благоприятные элементы для реализации проекта.

На основе полученных данных был произведён обзор существующих средств реализации игры, в результате которого была выбрана самая оптимальная платформа: **Unreal Engine 4.** Все приоритеты и недостатки игровых движков указаны в таблице 1.

Табл. 1. Результаты анализа аналогов

№	Характеристики	Среды разработки		
		Unity	**Unreal Engine**	**Godot Engine**
1	Мультиплатформенность	да	да	да
2	Бесплатная лицензия	да	да	да
3	Наличие обучающих материалов	да	да	нет
4	Большое количество игровых ресурсов	да	да	нет
5	Удобный визуальный скриптинг	нет	да	да
6	Проблемы с производительностью	да	нет	да
7	Удобство пользовательского интерфейса	да	да	нет

2. ПРОЕКТИРОВАНИЕ

2.1. Техническое предложение

Общие сведения

Доктору Айзексу приходит приглашение на участие в научном форуме «Регенерация» где должна решится судьба нового метода восстановления потерянных конечностей человеком, которая будет проходить на научной станции REDFIELD.

Основная концепция

REDFIELD — компьютерная игра в жанре action - adventure с элементами выживания, в котором игроку предстоит исследовать игровой мир и выполнять сюжетные задания для достижения конечной цели. При выполнении заданий он может получить сюжетные вещи и игровые ресурсы которые могут быть использованы в игровом мире, а главная роль в игровом процессе состоит в решение головоломок, которая требует от игрока как внимательности, так и умственных усилий.

Основную часть карты охватывают леса, можно так же наткнуться на разваленные поселения, большую опасность так же представляет погодные условия, которая влияет на показатели жажды и голода. Конечной целью является выполнение главного задания, а именно главная задача игрока добраться до ближайшего поселения, и связаться с научной станцией.

Основные особенности:

- головоломки и задачи;
- открытый мир.

Игровой процесс

Игрок выступает в роли доктора Айзекса, управление главным героем происходит от первого лица. Доктор носит с собой записную книгу, в которую он фиксирует в течение игры происходящее в качестве заметок, и видеокамеру. Так как доктор обычный человек, не владеющий оружием, он также не может драться, с собой у него видеокамера, которой он фикси-

рует события, чтобы потом их сохранить в качестве заметок в своей книге. [2]

Доктор может обследовать локации с помощью видеокамеры, которая оснащена ночным видением, без которого плохо освещенную местность просто невозможно разглядеть. При использовании прибора ночного видения камера начинает быстро разряжаться, что заставляет игрока постоянно искать разбросанные повсюду батарейки.

Игровой процесс включает в себя стэлс, который представляет собой скрытное перемещение от врагов, решение головоломок, а бегство от своих преследователей, преодолевая различные препятствия, и скрываться.

2.2. Определение требований к проекту

Задача заключается в выполнении работы по планированию функционала и навигации, структурировать идеи в качестве конкретных требований, во избежание ошибок усложняя работу на стадии ранней разработки. На этапе выполнения работы по проектированию были определены функциональные и нефункциональные требования к системе. [4]

Функциональные требования к проектируемой системе

Данный этап определений требования как функциональность программного обеспечения, описывают пункты, которые прежде всего необходимо реализовать в системе, о том какие возможности функциональность может предоставить.

Таким образом, можно перечислить следующий набор функциональных требований к системе.

Для разрабатываемой системы были выдвинуты следующие функциональные требования.

1. Система должна предоставлять пользователю возможность сохранять и загружать игровой прогресс.

2. Система должна предоставлять пользователю возможность взаимодействовать с предметами в инвентаре.

3. Система должна предоставлять пользователю возможность взаимодействовать с предметами на игровой карте.

4. Система должна предоставлять возможность активации видео камеры при нажатии клавиши.

5. Система должна иметь отображение шкалы здоровья, голода и жажды.

6. Система должна предоставлять интерфейс в случае проигрыша или выполнения задания.

7. Система должна выводить на экран пользователя текущее количество свободного места в инвентаре.

На основание этих требований была составлена диаграмма вариантов использования представлена на рисунке 4 и рисунке 5.

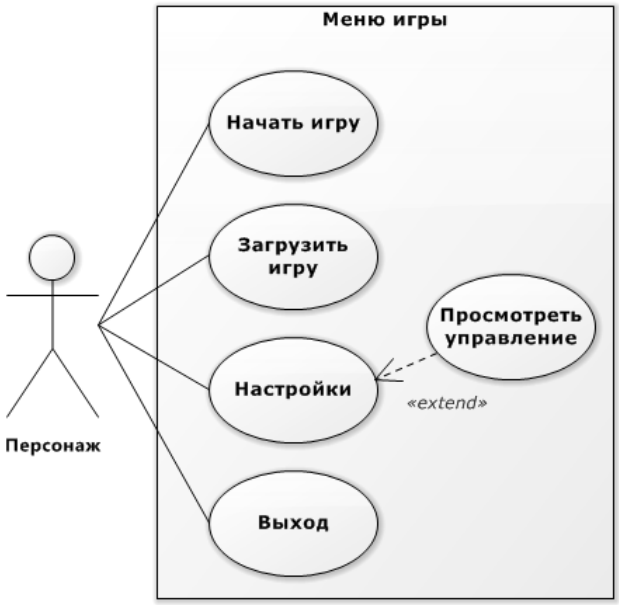

Рис.4. Диаграмма вариантов использования меню игры

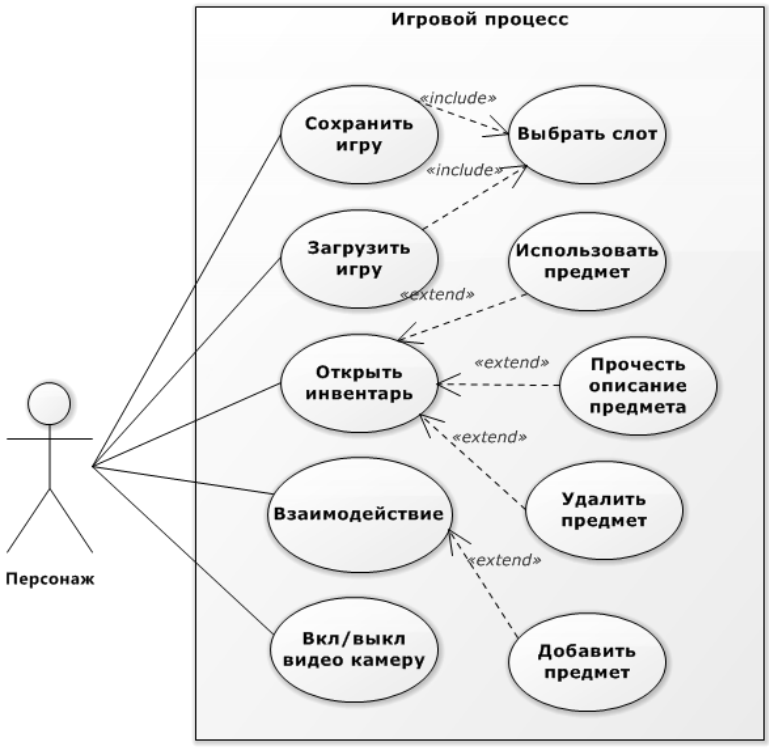

Рис.5. Диаграмма вариантов использования игрового процесса

Нефункциональные требования к проектируемой системе

Описание нефункциональных требований представляют собой этапы проектирования, которые в свою очередь предстоит выявить и описать как свойства, так и ограничения, накладываемые на систему. По итогу для реализации приложения сформулированы следующие нефункциональные требования.

1. Игра должна быть разработана на игровом движке Unreal Engine 4.

2. Скульптурирование органических объектов должно быть выполнено в ZBrush.

3. Ретопология а так же моделирование должно быть осуществлено в Autodesk Maya.

4. Система должна иметь открытый бесшовный мир.

5. Система должна быть написана на BluePrint.

2.3. Диаграмма классов

Была создана диаграмма класса для персонажа игры, на диаграмме присутствует основной класс персонажа Person Character, для взаимодействия с объектами на игровой карте данный класс использует метод Interect интерфейса InteractObjectInterface.

Так же этот класс имеет два компонента, Storage и HealthStats.

Класс Storage является коллекцией объектов типа StorageItem и реализует интерфейс StorageInterface для определения методов по редактированию этой коллекции. Storage является классом реализующий инвентарь. HealthStats содержит параметры игрока такие как жизни, голод и жажду.

От класса Person Character наследуется класс Enemy Character представляющий собой врагов данной игры (рисунок 6).

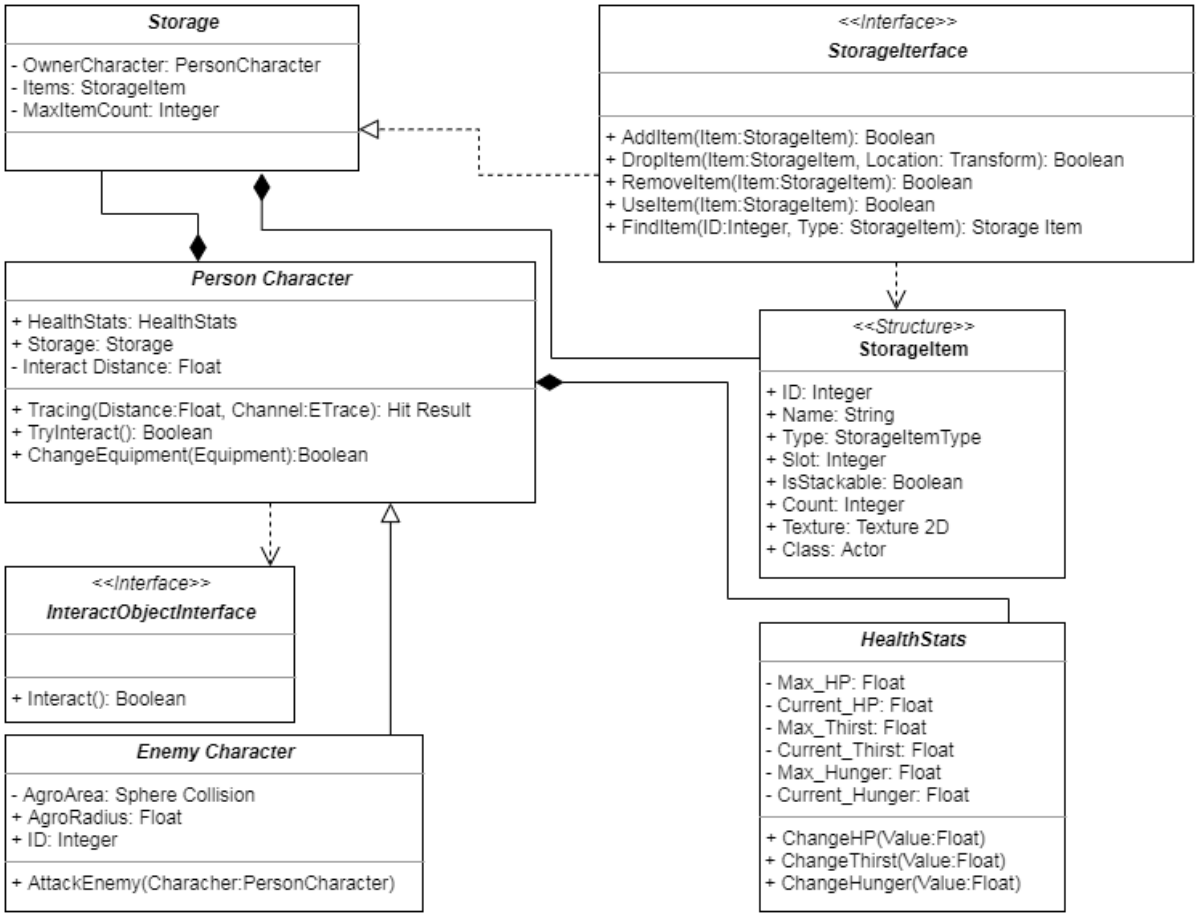

Рис. 6. Диаграмма класса для Person Character

StorageItem – это структура содержащая информацию об отдельном предмете инвентаря. ID – это уникальный номер предмета, Name содержит название объекта, Type имеет тип StorageitemType, Slot – номер слота инвентаря в каком находится предмет, isStackable – это булевское поле отвечающее за возможность предмета складываться в стек. То есть если это поле true, то в одной ячейке инвентаря может содержаться несколько таких предметов. Если isStackable равно true, то поле count хранит количество предметов в стеке, Texture содержит 2D изображение предмета, отображаемое в инвентаре. Class хранит класс объекта игрового мира соответствующего предмету из инвентаря.

Клавиши для управления персонажем

В настройках игры можно ознакомится с управлением персонажа. Перечисление действий в таблице 2.

Табл. 2. Управление

Действие	Клавиши
Движение вперед	W
Движение влево	A
Движение вправо	D
Движение назад	S
Поворот камерой	Mouse
Бег	Shift + W
Прыжок	Space
Взаимодействие с предметами	E
Меню паузы	P
Открыть инвентарь	Tab
Вкл/выкл видеокамеры	V

Влияние компонентов

Перечислен необходимый список компонентов которой оказывает как положительное, так и отрицательное воздействие на показатели игрока в таблице 3.

Табл. 3. Жизненные показатели

Компоненты	Жизни	Жажда	Голод
Бутылка воды		+ 15	
Еда готовая		- 5	+ 25
Еда сырая		- 15	+ 15
Медикаменты	+ 50	- 10	
Сон	+ 40	- 20	- 10

2.4. Проектирование интерфейса

На этапе проектирования необходимо организовать и скомпоновать интерфейс, сделать его понятным для пользователя.

Проектирование интерфейса осуществлялось на основе анализа, проведённого в пункте 1.1 «Анализ аналогичных проектов». В интерфейс были включены элементы с положительной оценкой от пользователей.

Ниже представлены интерфейс: Главное меню (рисунок 7), Инвентарь (рисунок 8), Видео камера (рисунок 9), Пауза (рисунок 10).

Рис. 7. Интерфейс «Главное меню»

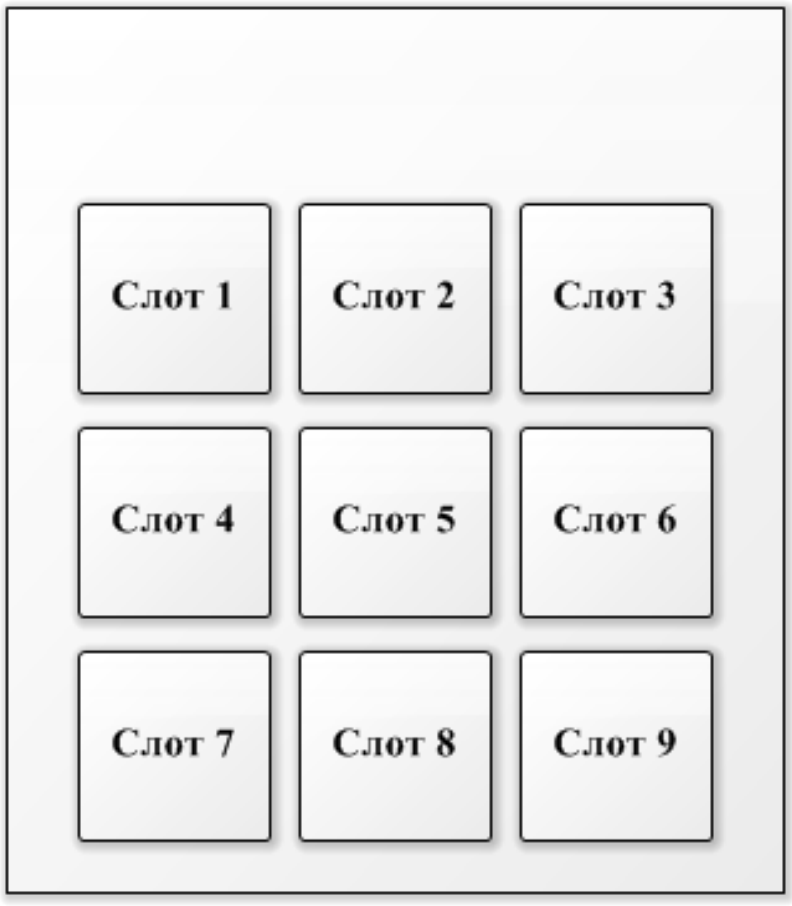

Рис. 8. Интерфейс «Инвентарь»

Рис. 9. Интерфейс «Видео камера»

Рис. 10. Интерфейс «Пауза»

Вывод

На основании теоретической части в ходе проектирования определены функциональные и нефункциональные требования к разрабатываемому проекту.

Для наглядного отображения функциональных требований была разработана диаграмма вариантов использования, после чего для детального представления внутренней части проекта была построена диаграмма классов. Окончательным этапом проектирования стала разработка интерфейса игры.

3. РЕАЛИЗАЦИЯ

3.1. Компоненты и структура проекта

Для разработки данного проекта была выбрана кроссплатформенная среда разработки Unreal Engine 4. Преимущества данного игрового движка перечислены на странице 13. Но все же одно из важнейших преимуществ – это технология Blueprint, предоставляющая возможность значительно упростить и ускорить разработку проекта.

Для графической составляющей игры, в качестве объектов наполнения, были использованы компоненты, ранее смоделированные мною на курсе основ разработки компьютерных игр.

Для создания приложения использовался пустой шаблон. На рисунке 11, представлена файловая структура игры. Проект состоит из набора каталогов, в которых содержится анимации, текстуры, модели, скрипты, аудио.

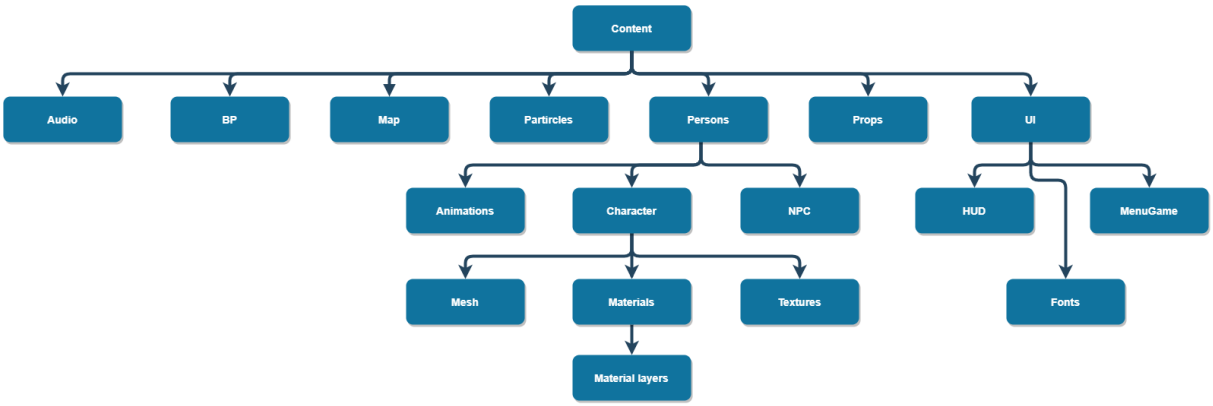

Рис. 11. Файловая диаграмма структуры приложения

В папке Audio хранятся звуковые дорожки, такие как шаги, бег, гроза, ветер, музыкальное сопровождение.

В папке BP содержатся все скрипты данного проекта: логика не игровых персонажей, смена времени суток.

В папке Map содержится карта бесшовного мира и папка с прилагающим материалом.

В папке Particles содержаться визуальные эффекты и система частиц, такие как огонь, дым, электрические искры.

В папке Persons содержаться модели NPC, а также модель протагониста и анимации. В том числе их текстуры, материалы, а также слои материалов.

В папке Props содержатся модели для наполнения игрового пространства, такие как деревья, столбы, дороги.

В папке UI содержится пользовательский интерфейс меню игры, шрифты, используемые в проекте, а также визуальный интерфейс игрока, отображающий на фоне игрового пространства видеоигры, такие показатели как шкала жажды, голода и здоровья.

3.2. Реализация базовой логики видеоигры средствами BluePrint

Blueprint - это визуальный скриптинг в Unreal Engine 4, представляет собой в свою очередь визуальный интерфейс для создания элементов игрового процесса. Система движка гибкая и мощная, позволяет программистам и дизайнерам использовать концепцию и полный потенциал программирования. [15]

В начале была осуществлена реализация Главного меню, был создан класс MainMenuGameMode, это класс, определяющий правила текущей игры, в том числе: как игрок подключаются к игре, может ли игра быть поставлена на паузу, порядок и правила смены уровней, условия победы.

После чего необходимо было создать MainGameInstance класс в папке BP, служит сохранению значения переменных при загрузке следующего Level, к тому же туда легко кастомизировать что делает его одним из посредников, для некоторого взаимодействия между blueprint.оступен на протяжении всей запущенной игры, а именно, если мы заходим в игру, не важно на каком бы уровне мы не находились, он всегда доступен, это такой специфический класс, затем в папке Map создан Level и назван как новый уровень MainMenu.

В настройках проекта Maps & Modes, по дефолту устанавливается MainMenuGameMode, после по дефолту в Editor Startup Map и Game Default Map установлен Level MainMenu. И в завершении устанавливается свой класс MainGameInstance.

В подпапке MenuGame были созданы следующие Widget – Blueprint, сокращенно WBP, назовем их MainMenuWBP, OptionMenuWBP и LoadingScreenWBP. Они служат для визуализации интерфейса главного меню.

Далее размещаются шрифты в под папке Fronts, которые будут использоваться в проекте.

В папке Audio были размещены две аудио дорожки:

Button_Click – воспроизводится в момент когда игрок кликает на кнопку в меню.

Button_Hover – воспроизводится в момент когда игрок при наведении курсора мыши по кнопку.

Аудио файлы должны быть в формате WAV от англ. waveform — «в форме волны», формат файла-контейнера для хранения записи оцифрованного аудиопотока, подвид RIFF. Этот контейнер, как правило, используется для хранения несжатого звука в импульсно-кодовой модуляции которая используется для оцифровки аналоговых сигналов.

В MainGameInstance создан Custom Event и именован как Show Main Menu (рисунок 12).

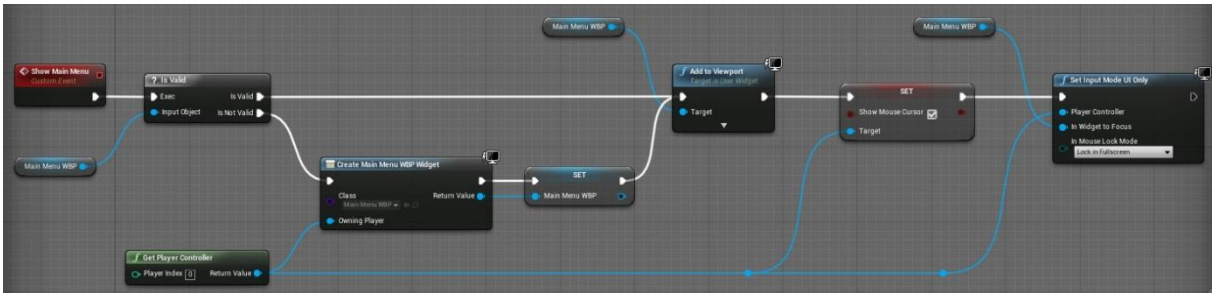

Рис. 12. Реализация класса MainGameInstance

Show Main Menu является пользовательским событием. Рассмотрим реализацию класса MainGameInstance в виде блок-схемы (рисунок 13).

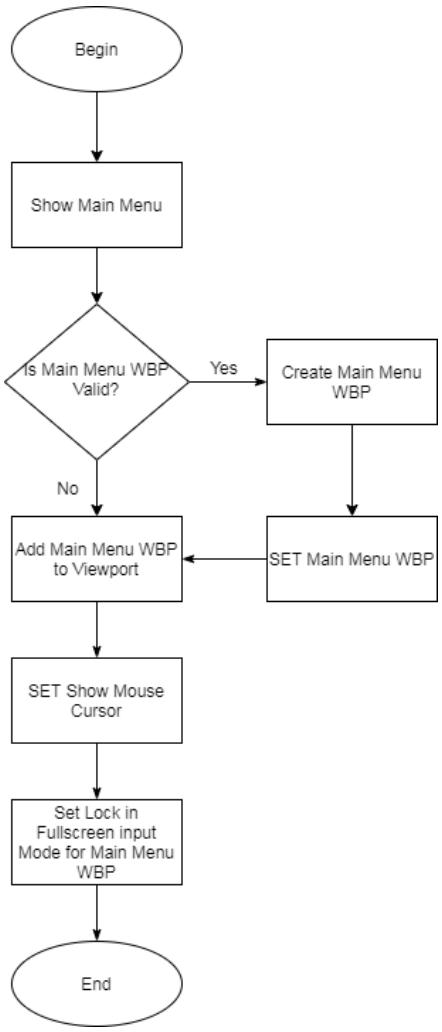

Рис. 13. Блок-схема запуска главного меню игры

На запуске игры мы создаем функцию widget меню, назначаем класс Main Menu WBP. Затем мы с Get Player Controller подаем в Owning Player контроллер в widget, после мы объявляем и записываем все в переменную MainMenuWBP. Все переменные, которые будут отвечать за widget, обозначим категорией widgets.

После того как мы объявили переменную, мы создадим функцию Add to Viewport, которая принимает значение переменной. Но в том случае, если игрок будет несколько раз заходить в главное меню, widget будет каждый раз создаваться по-новому.

Проведена оптимизация. Для того чтобы каждый раз не создавать и не удалять заново widget, мы берем ссылку MainMenuWBP на этот widget, а дальше вызываем конструкцию Is Valid?, Затем мы подключаем её к со-

бытию Show Main Menu. В том случае, если мы в первый раз заходим в игру, наша ссылка не валидна, то есть по умолчанию widget еще не создан, первый раз, когда мы зашли в игру, то мы создаем widget и добавляем его таким образом во Viewport, но в случае если мы второй раз за игру зашли в главное меню и у нас этот widget уже был создан мы будем его скрывать в памяти и он будет оставаться. Соответственно если оно валидно, то мы просто выводим widget на экран.

Курсор по умолчанию скрытый, для этого добавляем переменную Show Mosu Cursor типа Boolean, которая установит видимость курсора и в завершении, чтобы добавить фокус на widget меню игры, мы добавляем функцию Set Input Mode UI Only, где устанавливаем Lock in Fullscreen это действие необходимо чтобы курсор мыши не выезжал за пределы окна.

После того как была создана функция в MainGameInstance, необходимо вызвать событие Show Main Menu в Level MainMenu. Для этого вызываем функцию Get Game Instance и в Cast указываем событие Show Main Menu и все это связываем с Event BeginPlay (рисунок. 14).

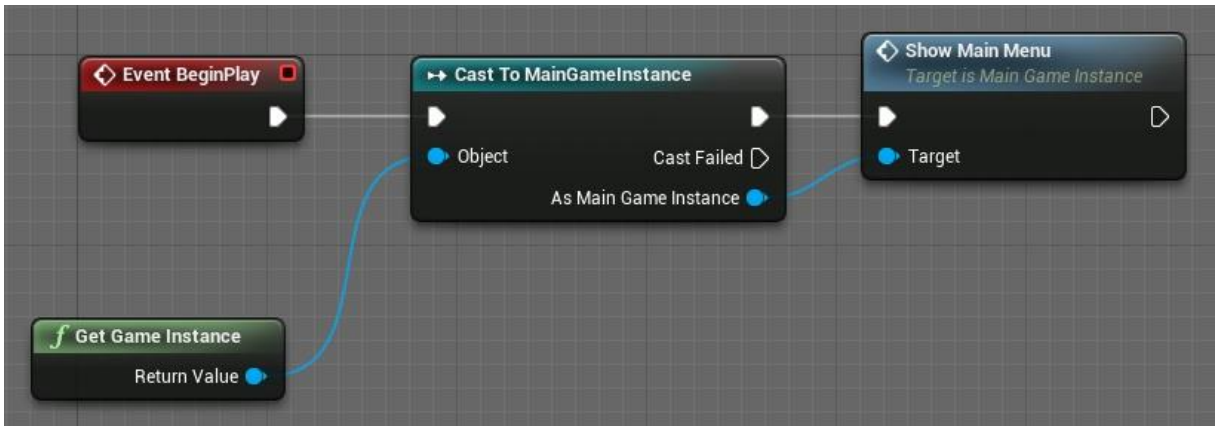

Рис. 14. Событие BeginPlay в Level MainMenu

Widget, расположенный в под папке MenuGame [11], визуализируем интерфейс (рисунок 15).

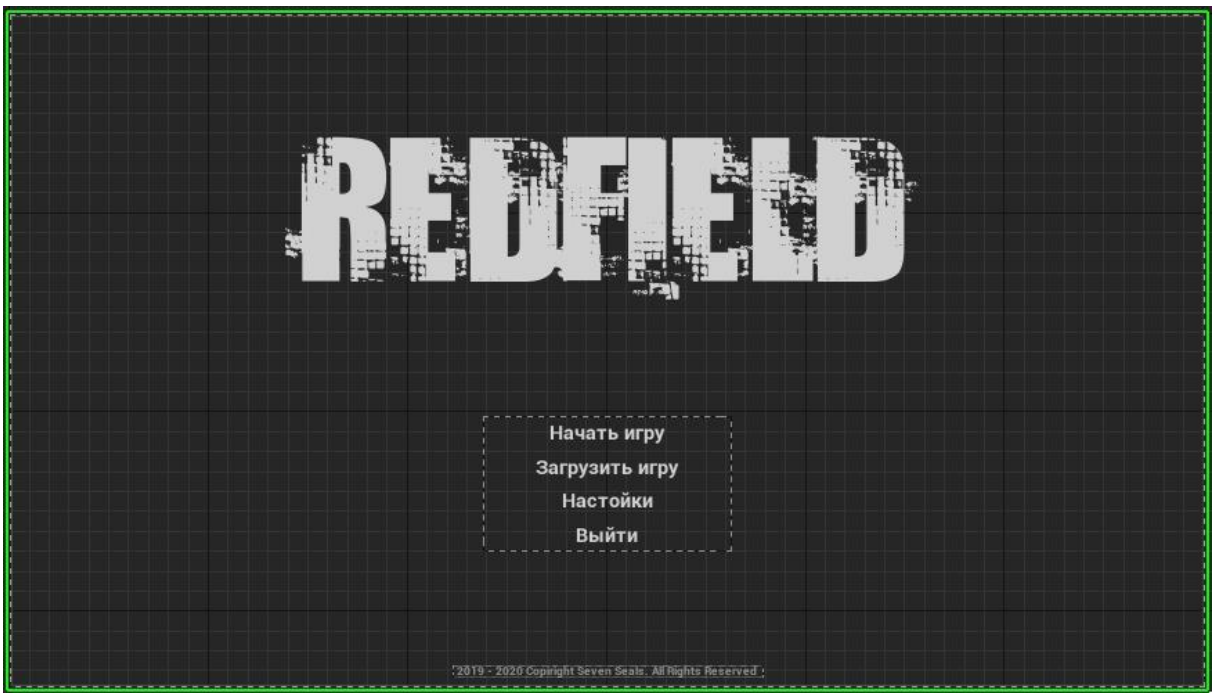

Рис. 15. Widget – Blueprint главного меню игры

Canvas Panel

Панель для свободного размещения элементов внутри себя. Так же поддерживает якоря для выравнивания элементов и Z-Order, который позволяет установить какие элементы будут поверх других.

Vertical Box

Упорядочивает элементы в вертикальном порядке.

Button

Кнопка с понятным функционалом нажатия. Поддерживает один дочерний элемент, в который вы можете поместить дополнительный элемент, чтобы придать кнопке особый вид.

Text

Простой текст, который можно разместить на пользовательском интерфейсе. Может использоваться как для однострочных названий, так и для многострочных текстов.

В [Canvas Panel], выводим название игры через [Text] стилизованным шрифтом Docteur Atomic, затем располагаем [Vertical Box] ниже центра экрана по координатам X 0,5 и Y -0,35 и устанавливаем [Button] под

каждую кнопку в [Vertical Box]. Закрепив на каждую кнопку [Text], даем кнопке название.

У всех [Button] в параметре Pressed Sound назначаем Button_Click и в Hovered Sound назначаем Butten_Hover. Теперь фокус курсора на кнопке и клик курсора по кнопке будет сопровождаться аудио дорожкой, расположенной в папке Audio.

Для [Text] и [Vertical Box] необходимо сделать привязку к разрешению экрана по центру, указав параметр Anchors. Эта мера необходима для того, чтобы у пользователя не съезжалось меню куда-нибудь в бок. Таким образом мы зафиксировали привязку, и значение экрана будет пропорционально увеличиваться и уменьшаться.

В Graph MainMenuWBP под каждый [Button] создадим соответствующие названию события On Clicked (рисунок 16).

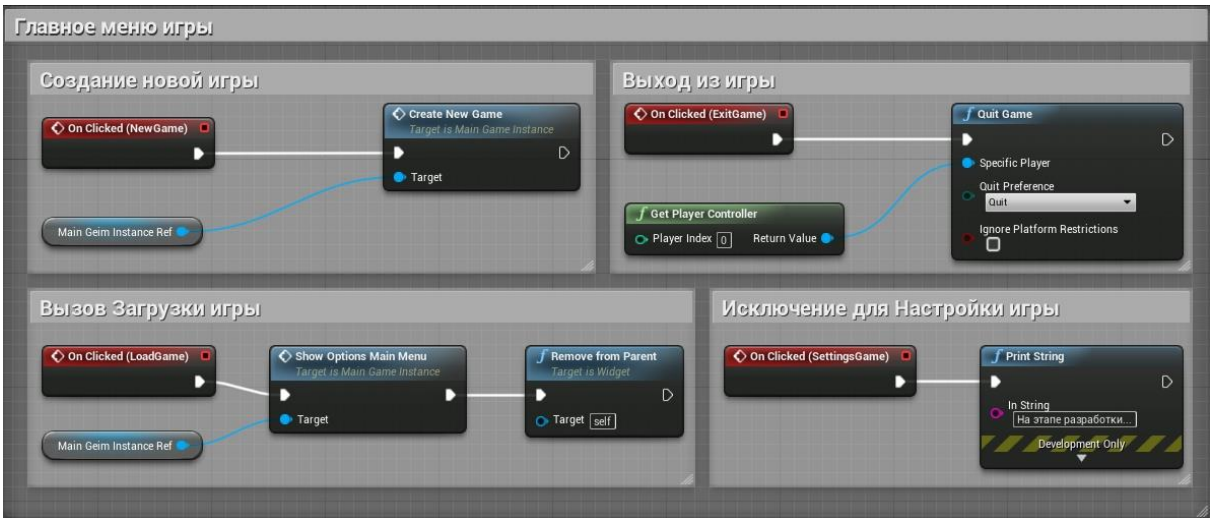

Рис. 16. On Clicked для всех [Button] Меню

Видеокамера

Создадим VideoCameraHUD в котором пропишем событие на клавишу V которая при нажатии будет активировать widget камеры. После активации текстура шероховатости белого шума будет накладываться на экран с настройкой освещения изображения в результате в темных местах включая камеру мы будет просто увеличивать интенсивность освещения, которая в свою очередь будет смотреться как аналог ночного видения.

Добавим переменную типа float Energy которая будет уменьшать объем энергии камеры, восполнить которую можно будет найдя на локации, батарейку которая будет содержать параметр Battery и обновлять переменную Energy (рисунок 17).

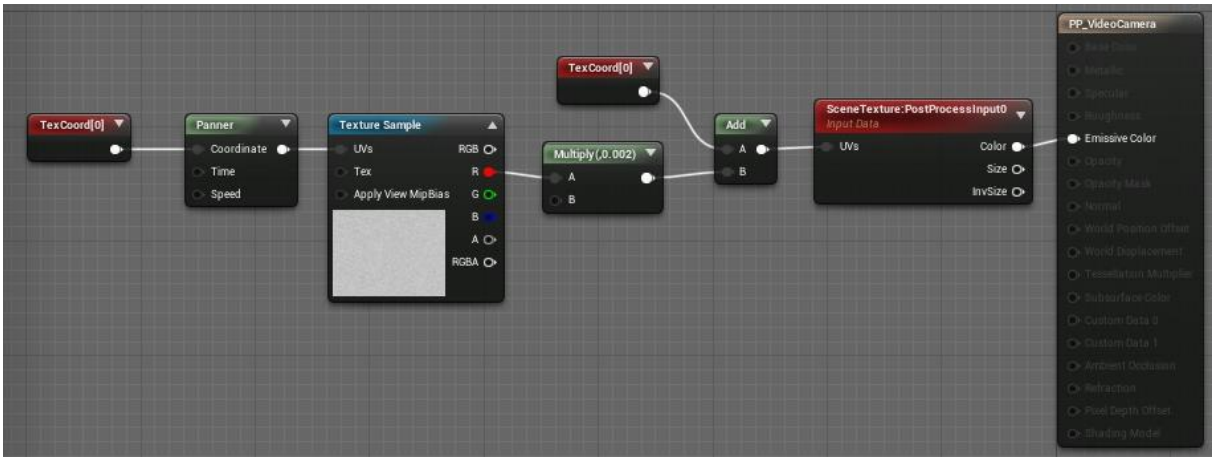

Рис. 17. Класс VideoCamera

Создадим SetVisibleDot для того чтобы в правом верхнем углу мигала красная точка, периодичностью в 1 секунду сигнализируя о работе камеры, а также с левого верхнего угла будет показатель батарейки с 4 делениями каждое из которых хватает на 5 минут. Следовательно, Battery при обновлении значения переменной Energy выставит 20 минут (рисунок 18).

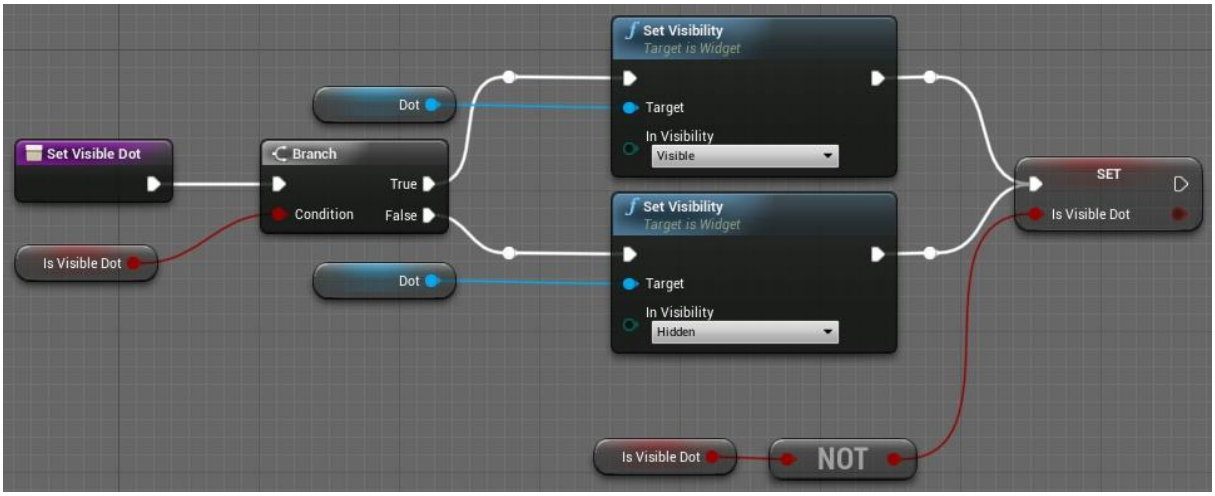

Рис. 18. Функция SetVisibleDot

В Photoshop рисуем один угл, батарейку и её деления, а также значок zoom, здесь он играет исключительно визуальную часть и не функционирует. REC вставляем через [Text]. Затем просто разместим все в widget который активируется при нажатии клавиши V (рисунок 19).

Рис. 19. Включенная камера

Инвентарь

Для инвентаря был создан класс Storage, который должен предоставлять возможности пользователю, такие как добавлять объект в инвентарь, можно использовать (рисунок 20) а так же хранить его там (рисунок 21), и удалять объект из инвентаря (рисунок 22).

Класс Use Item

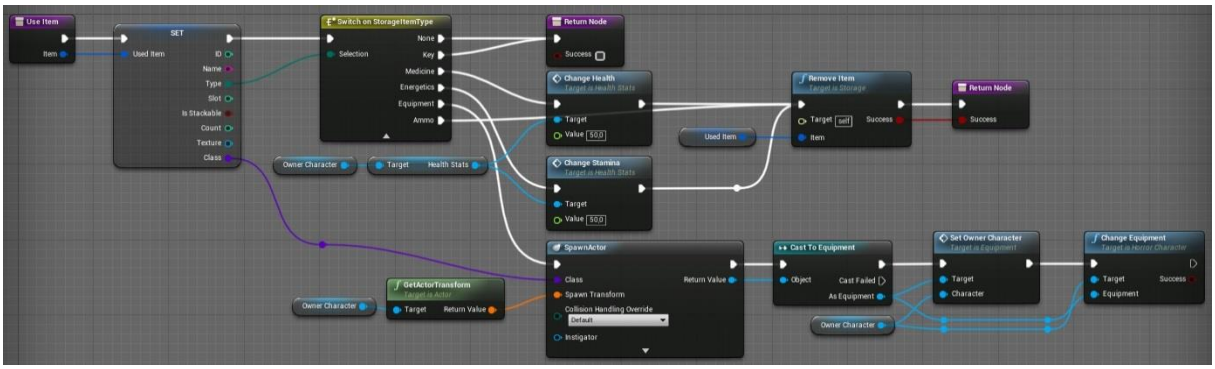

Рис. 20. Использование предмета в инвентаре

Класс Add Item

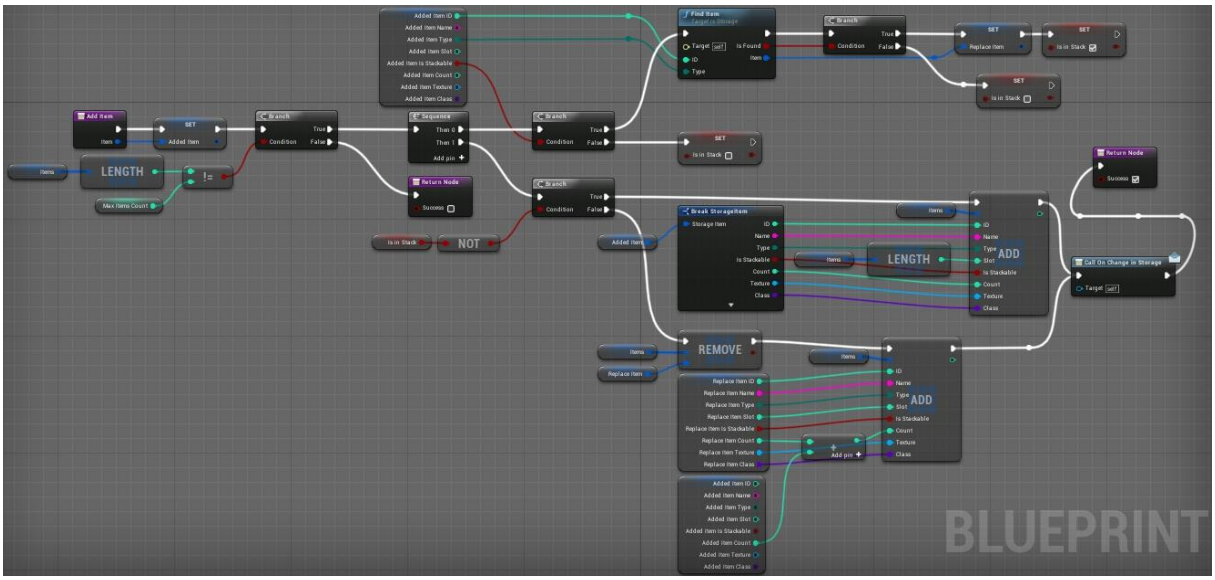

Рис. 21. Добавление предмета в инвентарь

Класс Remove Item

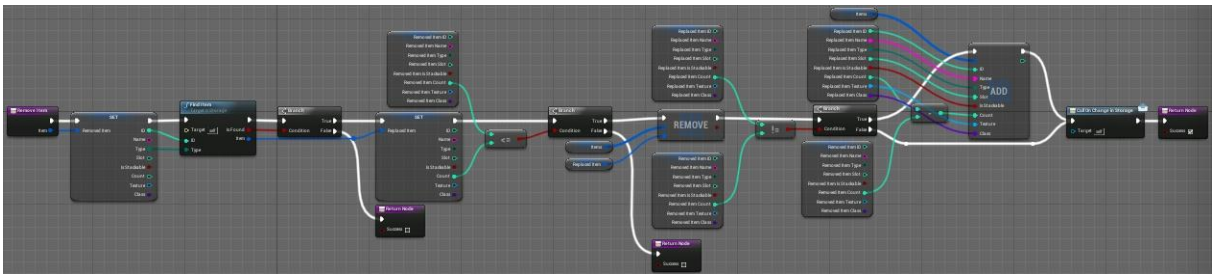

Рис. 22. Удаление предмета из инвентаря

Виды предметов

Предметы делятся на три типа представленные в таблице 4, это сюжетные, такие как ключи, вспомогательные такие как бутылка чтобы отвлечь внимание противника и повествующие, такие как фотография экспедиции, которая обнаружила захоронения, раскрывающие характер второстепенных персонажей игры как сюжетную составляющую.

Табл. 4. Игровые типы предметов

Сюжетные	Вспомогательные	Повествующие
Ключ	Бутылка	Фотография
Пропускная карточка	Батарейка	Дневник

3.3. Разработка конвейера 3D моделей

На данном этапе была проделана огромная работа. Помимо создания второстепенных моделей окружения, было проведено создание и разработка главного протагониста игры. Из простых примитивов до детализированного героя. Поэтапное прохождение по пайплайну на примере главного протагониста игры.

Pipline (Пайплайн) в переводе с английского можно перевести как «Конвейер», дословный перевод «путепровод».

Пайплайн – это несколько последовательных этапов разработки модели, связанные друг с другом, которые на выходе дают оптимизированного персонажа для игры с максимальным качеством и выразительностью. В него входят и художественные этапы, создание арта, обеспечивая необходимый уровень художественной выразительности, а не просто замоделить что-либо.

3.3.1. Этапы Pipline

1. Скульптинг.
2. Ретопология.
3. Развертка модели.
4. Запекание UV текстур.
5. Текстурирование модели.
6. Риггинг и скининг.
7. Анимация.
8. Сборка в игровом движке.

Скульптинг

Для обеспечения работы со скульптом [9], этапы его производства делятся на 4 этапа, а точнее 4 рида:

Первый рид – базовая форма и силуэт. Лепим форму черепа на рисунке 23 создание главного протогониста. В Z-Brush можно слепить персонажа из любого куска «цифровой глины». На старте, чтобы голова стала похож на голову, нам достаточно создать обычную сферу.

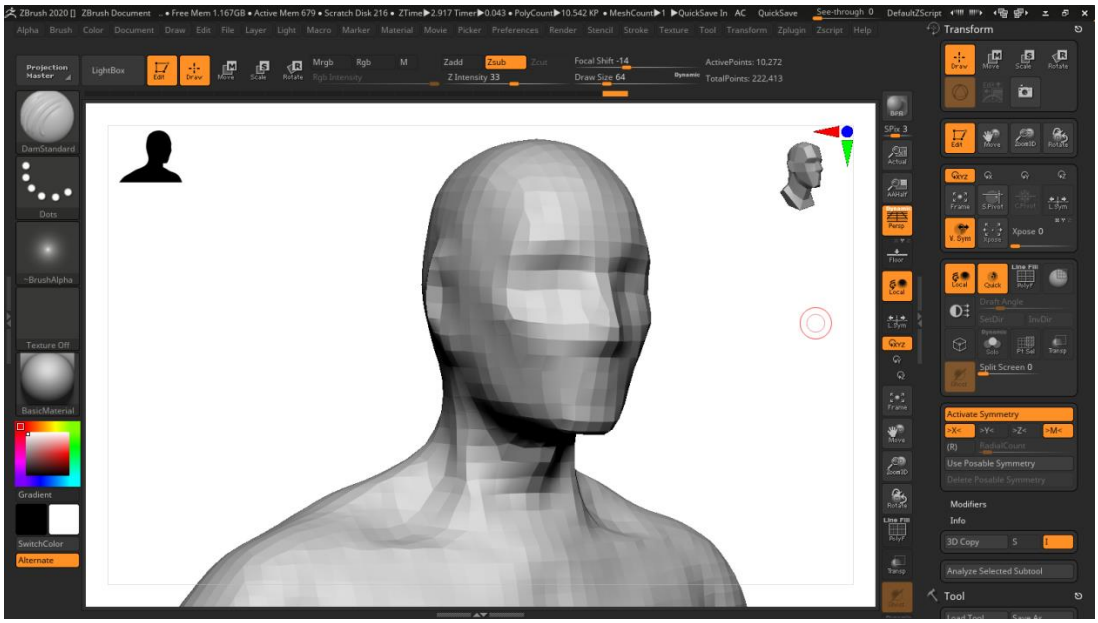

Рис. 23. Базовая форма и силуэт

Второй рид – внутреннее наполнение и большие детали. Нам нужно уточнить уже созданные формы. Разделить их на отдельные элементы (рисунок 24).

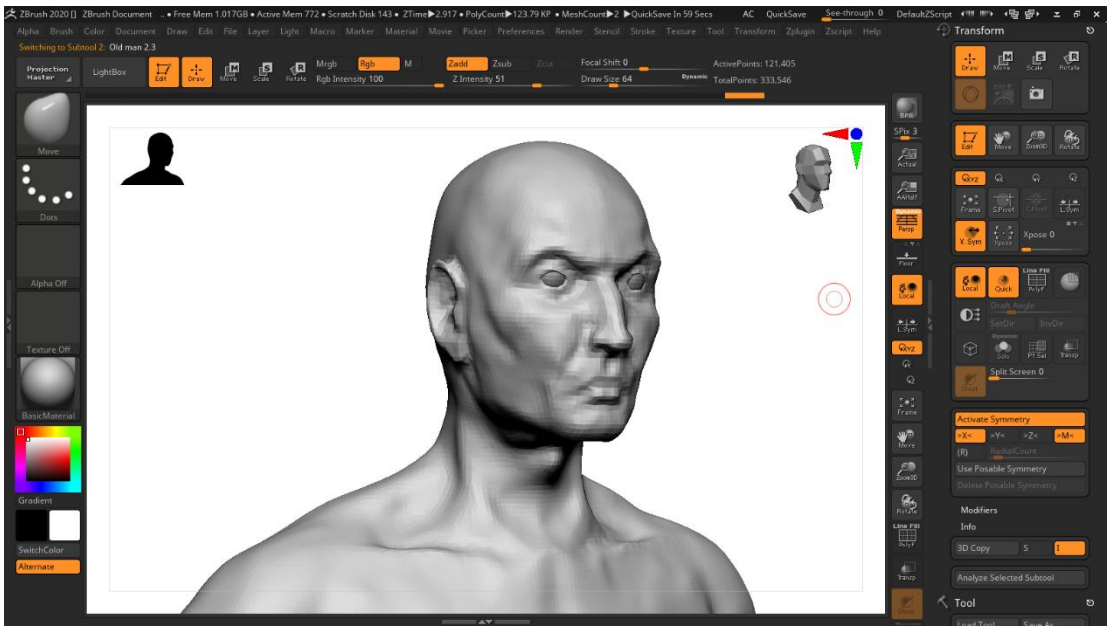

Рис. 24. Внутреннее наполнение и большие детали

Третий рид – маленькие детали и нюансы анатомии. Цель третьего рида – работа над крупными морщинами, складками и прочими деталями, а также с нюансами анатомии, подготовка к высоко детализированной проработке (рисунок 25).

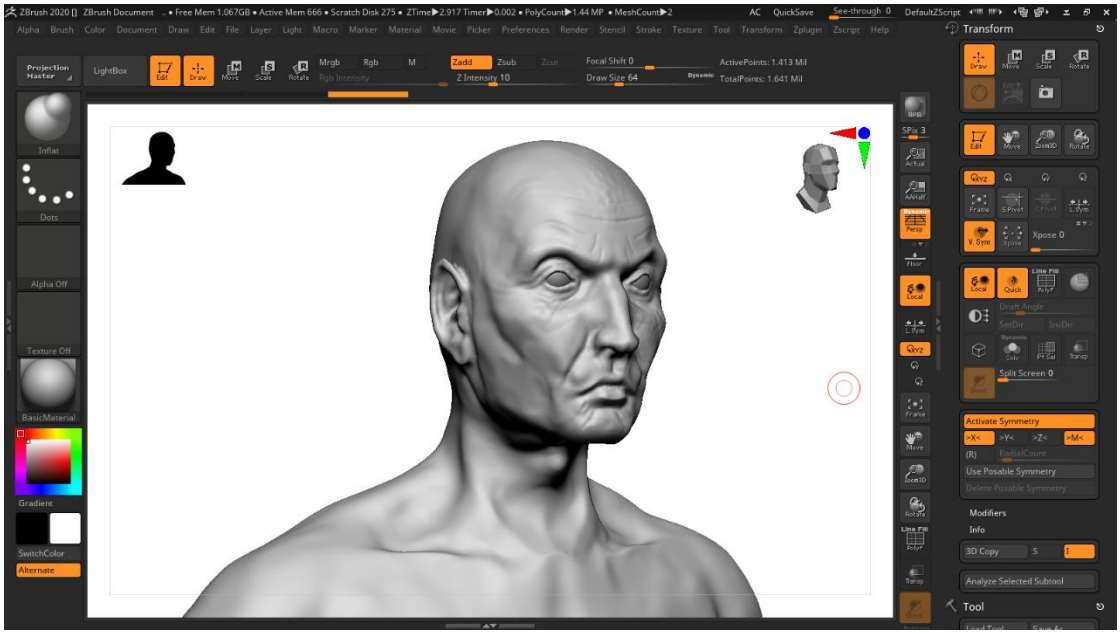

Рис. 25. Маленькие детали и нюансы анатомии

Четвертый рид – финальная детализация материалов и поверхности. Включает в себя поры, морщины, мелкие трещины. Он используется в основном для реалистичных персонажей (рисунок 26).

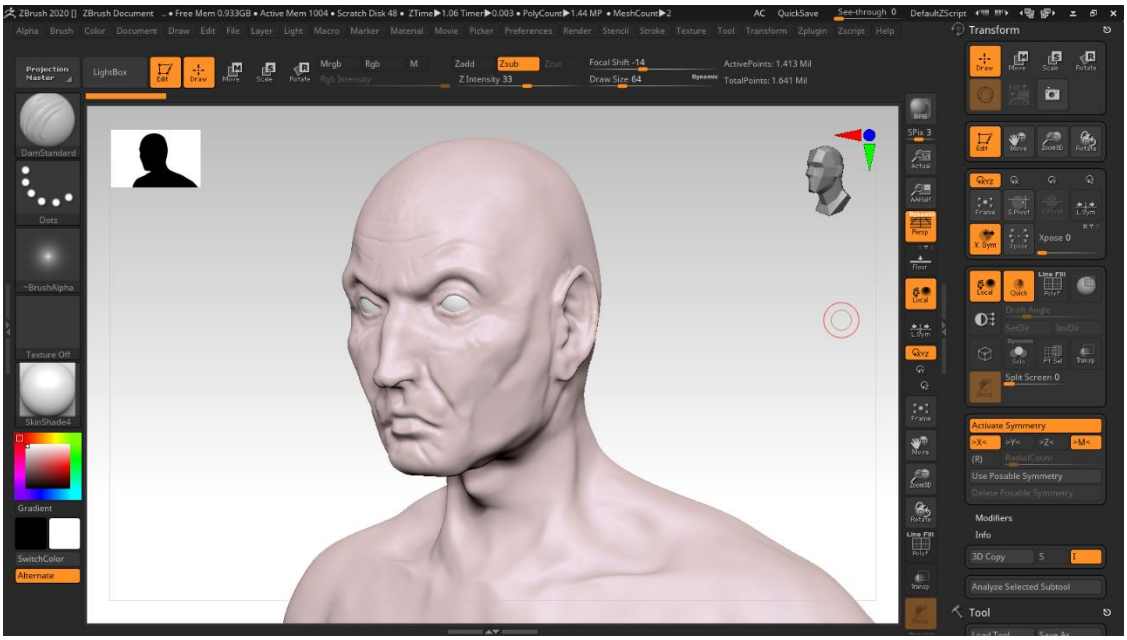

Рис. 26. Финальная детализация материалов

Ретопология

Создаём новую модель, которая соответствует требованиям игрового движка, сокращая количество полигонов в сотни и тысячи раз. Если скульпт модели называется highpoly, то после этапа ретопологии модель трансформируется в lowpoly модель.

Развёртка

3D модель в UV пространстве, развёрнутая на плоскости. Она может напоминать выкройку под одежду, как по функции, так и по производству. После того как модель развернута в двухмерной плоскости, на нее можно будет печь, проецировать и вырисовывать необходимые текстурные карты. UV-преобразование, другими словами развёртка в трёхмерном пространстве. Соответствие трёхмерного объекта, между координатами на поверхности расположенного по координатам [x, y, z] и координатами на текстуре U, V. Значения U и V изменяются от 0 до 1. Развёртка может строиться как автоматически, так и вручную.

Запекание

Это достаточно сложный этап. Это процесс проецирования (переноса) детализации с высокополигональной модели на низкополигональную.

Текстурирование

На этом этапе необходимо настроить базовые материалы и маски. А дальше — время текстур. Для органики и оружия есть свои правила правильного текстурирования.

Риггинг и Скининг

Это не просто создание скелета персонажа, это также создание различных систем управления персонажами, техникой, объектами, создание вспомогательных инструментов и инструментов автоматизации. Без рига персонаж выглядит для программы как куча полигонов. Для анимации персонажа нужно понимать на что он будет опираться. Создать скелет персонажа — это называется ригом. Затем нужно привязать этот скелет к твоей модели — этот этап называется скининг. Риг и скининг можно делать в любом большом 3д редакторе, но для этой работы предпочтительнее 3Ds Max.

Анимация

Есть несколько самых распространённых видов анимации персонажей: стойка (idle), походка, атака, смерть.

Анимация — финальный этап пайплайна, но это не всё. Потому что модель необходимо экспортировать в игровой движок.

3.4. Сборка модели Crow

Одним из самых опасных врагов в игре являются стаи ворон. Пример сборки в Unreal Engine 4 на рисунке 27 готовая собранная модель.

Рис. 27. Crow

Модель состоит из 4698 треугольников, 4058 вершин, 1 канал UV развертки, размеры 155x96x53. Для её сборки на этапе запекания были созданы карты нормалей для имитации неровностей поверхности на объекте. Они применяются, чтобы сделать вашу финальную модель похожей на высоко полигональной версию. С их помощью можно добить различные детали, которые нельзя передать через геометрию и заставить вашу модель выглядеть более скругленной для лучшей передачи освещенности и большей реалистичности.

Карты нормалей или текстурные карты создаются во время этапа запекия – это RGB изображения, где каждый из каналов (красный, зелёный, синий) интерпретируется в X, Y и Z координаты нормалей поверхности соответственно. Красный канал пространства касательных карты нормалей отвечает за ось X (нормали направленны влево или вправо), зелёный канал за ось Y (нормали направлены вверх или вниз) и синий канал за ось Z (нормали направлены прямо от поверхности).

Текстурные карты тела

Нормаль - это вектор перпендикулярный поверхности с направлением определяемым правилом буравчика. Карты нормалей или рельефное текстурирование позволяет значительно улучшить графику поверхностей в игре, сделать их более реальными и насыщенными. Сама карта хранит информацию в пикселах о поверхности объекта.

На рисунке 28 карта цвета, 29 рисунок является картой нормали не содержат информации о высоте. Вместо этого они содержат информацию о углах. Они цветные, значения RGB сообщают рендеру, в каком направлении идет наклон, используя информацию об угле для того, чтобы искусственно сгибать края соседних граней друг к другу, тем самым имитируя эффект фаски. Это невозможно сделать только с информацией о высоте, потому что рендер не может знать, в каком направлении должны быть согнуты края.

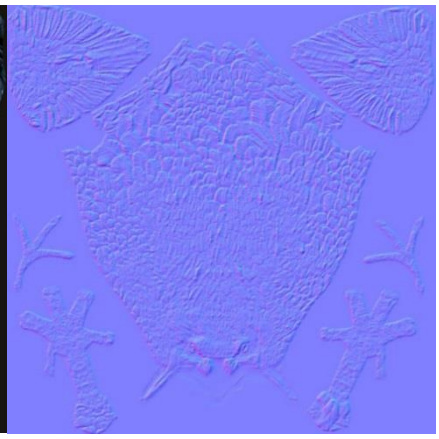

Рис. 28. Body_Color **Рис. 29.** Body_Normal

На рисунке 30 изображена карта шероховатости, которая имитирует неровность поверхности методом варьирования полутонов. Позволяет сэкономить затраты вычислительных ресурсов на непосредственное моделирование неровной поверхности. И наконец карта отражения на рисунке 31 текстура, которая показывает способность отражения материала. В отличие от reflection map, specular map не показывает отражения сцены, в которой находится объект, а показывает отражения света, падающего на не-

го. Specular Map содержит в себе пиксели в черно-белой цветовой гамме. Чем светлее пиксель, тем больше способность материала отбивать свет и тем ярче на нём блики от света. Соответственно, чем темнее пиксель, тем матовее становится материал и теряет своё свойство отражать свет. Для материалов керамической плитки и полированного металла могут использоваться светлые тоны, а для тканей и дерева тёмные.

Рис. 30. Body_Roughness **Рис. 31.** Body_Specular

Для полноценной модели тела вороны теперь в Unreal Engine 4 нужно создать материал, который будет объединять слои, накладывая их с параметрами, которые мы можем сами настроить.

Создадим материал и назовем его Body_MAT на рисунке 32, соединим между собой карты текстур затем. Можно использовать нод Multiply для регулировки яркости текстуры, он умножает один вход на другой вход. С помощью умножения можно изменять яркость пикселя, не затрагивая его оттенок или насыщенность.

Ниже представлен пример уменьшения яркости наполовину умножением каждого канала. Выполнив эту операцию для каждого пикселя, мы можем изменить яркость всей текстуры. Fresnel используется для описания того, как свет, который вы видите, отражается в разных интенсивностях, исходя из угла, с которого вы смотрите.

Например, если вы стоите над бассейном, глядя прямо на бассейн, вы не увидите много отражения. Metalic буквально управляет тем, насколько

металлическая ваша поверхность. Значение 0 для неметаллов и значение 1 для металлов. Для чистых поверхностей, таких как чистый металл, натуральный камень, чистый пластик и т.д. Это значение будет 0 или 1, а не что-то между ними.

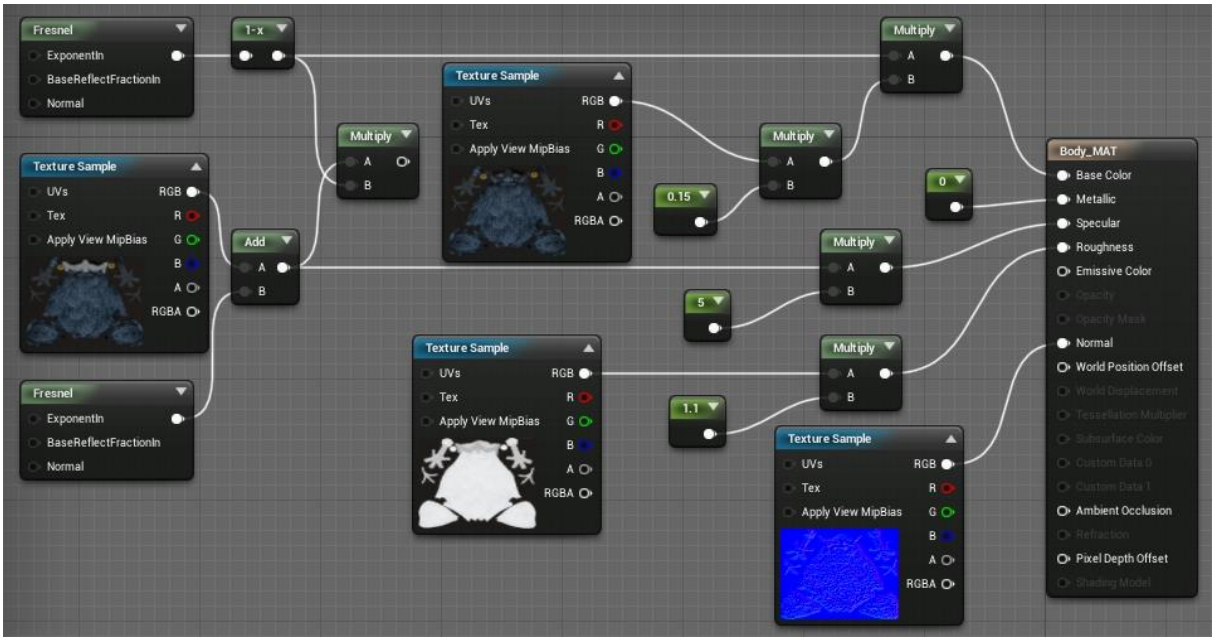

Рис. 32. Реализация Body_MAT

Для полного завершения реалистичности модели необходимо сделать перья, для этого создадим аналогичный материал и назовем его Feathers_MAT, соединение и параметры точно такие же как на рисунке 26. Ниже на рисунке 33 карта цвета, рисунок 34 карта нормалей пера, карта шероховатости пера на рисунке 35 и на рисунке 36 зеркальная карта показывает отражения света, падающего на перо.

Текстурные карты перьев.

Рис.33. Feathers_Color

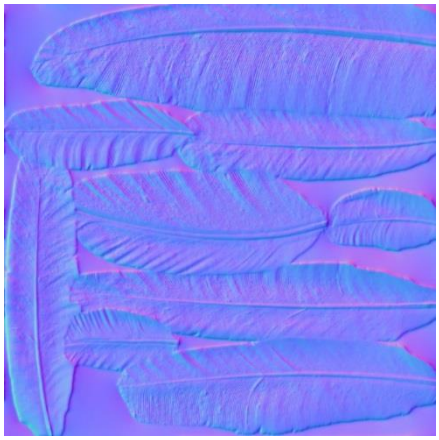

Рис. 34. Feathers_Normal

Рис. 35. Feathers_Roughness

Рис. 36. Feathers_Specular

3.4.1. Blueprint Enemy Character

Базовый искусственный интеллект, который может видеть преследовать и атаковать игрока, как на ближних, так и на дальних дистанциях. Присутствуют несколько примеров такие, как стаи ворон, гиены и жители поселения. Персонажи полностью про анимированы, настроены и озвучены. [10]

Event Graph

В Blueprint Enemy Character создадим метод Event Graph и опишем следующие действия (рисунок 37).

Event BeginPlay – событие запускаемое в начале игры.

Get Game Instance – передача GameInstance. Описание в главе 3.2. Реализация базовой логики видеоигры средствами BluePrint на странице 24.

Cast To MainGameInstance – преобразования к типу.

Target Deated Enemies – переменная типа integer содержащая информацию о смерти объекта.

ID – переменная типа integer содержащая идентификационный номер объекта.

CONTAINTS – проверка на содержание в списке объекта с данными значениями.

Branch – данный узел служит простым способом выбора одного из двух действия на основе логического условия. Если условие вернет истину (true) то будет выполняться одноименная ветка либо вернется ложь (false) то будет выполняться другая ветка.

DestroyActor – удаляет с уровня объект класса Actor.

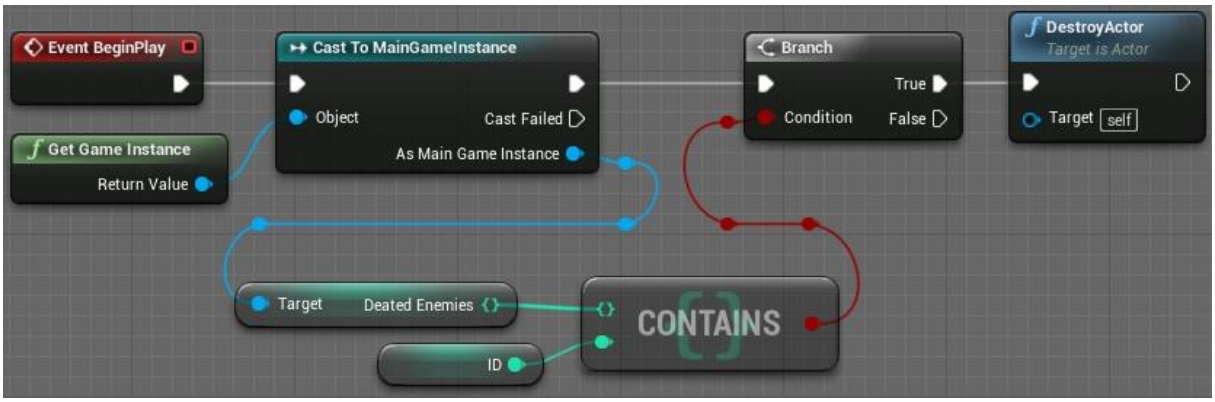

Рис. 37. Реализация удаление с уровня ранее умерших врагов

При прогрузки уровня данный метод проверяет существует ли объект в данной секции либо умер, если объект мертв, то он должен быть удален с карты в противном случае он будет появляться каждый раз на одной и той же секции в одном и том же месте после прогрузки уровня.

Construction Script

В Blueprint Enemy Character создадим метод Construction Script и опишем следующие действия Construction Script – запуск скрипта конструктора (рисунок 38).

Parent: Construction Script – вызов конструктора базового класса объекта.

Set Sphere Radius – устанавливаем сферу радиуса видимости.

Agro Area – переменная содержащая сферу видимости.

Agro Radius – переменная типа float радиус сферы видимости.

Рис. 38. Реализация конструктора объекта

Attack Enemy

В Blueprint Enemy Character создадим метод Attack Enemy и описание следующих действий Attack Enemy – запуск скрипта атаки (рисунок 39).

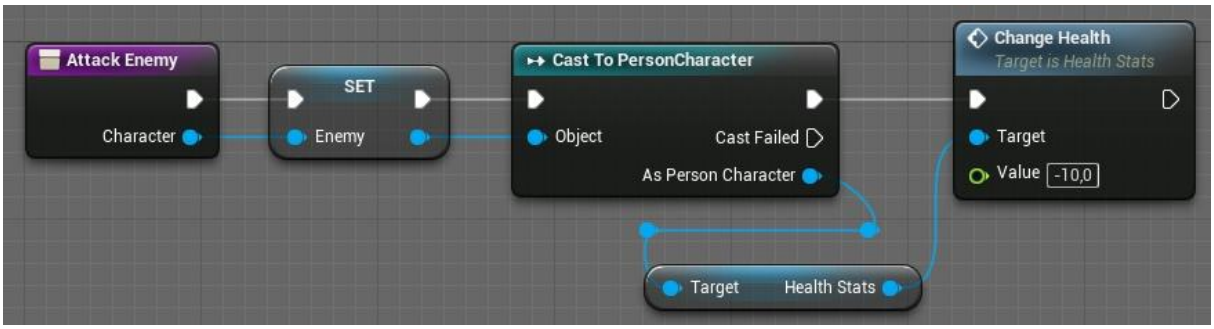

Рис. 39. Атака врага

Attack Enemy – запуск метода.

SET – установка значения Enemy.

Enemy – переменная содержащая игрока.

Cast To PersonCharacter – преобразование типа переменной Enemy.

Target Health Stats – жизненные показатели.

Change Health – изменение жизненных показателей.

3.5. Сборка Block Out

Блокаут это трехмерный макет уровня, делится он на два типа: Grey Box и White Box.

Grey Box

Это макет, который используют для проверки геймплея, в нем есть только то, что не посредственно влияет на игровой процесс. Серые кубы, цилиндры и плоскости схематично отображают будущую локацию, хотя и создаются с учетом всех масштабов и пропорций. На данном этапе не важно на сколько красиво нарисована пропасть с кислотой, важно чтобы игрок мог через неё перепрыгнуть.

На данном этапе тестируют механики, такие как можно ли забраться на отступ или спрятаться за ним, изучают как ощущается геймплей и сложность, в случаях если что-то не функционирует, проблемный элемент переделывают или удаляют (рисунок 40).

После проведения всех правок по механикам, прототип обрастает деталями и превращается в White Box.

White Box

Именно на этом этапе художники прорабатывают визуальную составляющую часть уровня, то есть накладывают модели на серый эскиз, а также устанавливают полноценные модели вместо примитивов (рисунок 41).

Создание локации начинается с планирования, необходимо определить, какое место конкретная локация займет в финальной игре. После чего нужно проработать напряженность уровня и его структуру.

Напряженность событий регулирует накал эмоций на каждом этапе прохождения. На уровне необходимы слабые доли – чтобы игрок мог отдохнуть после сложного этапа, а также сильные доли, чтобы игрок не заскучал.

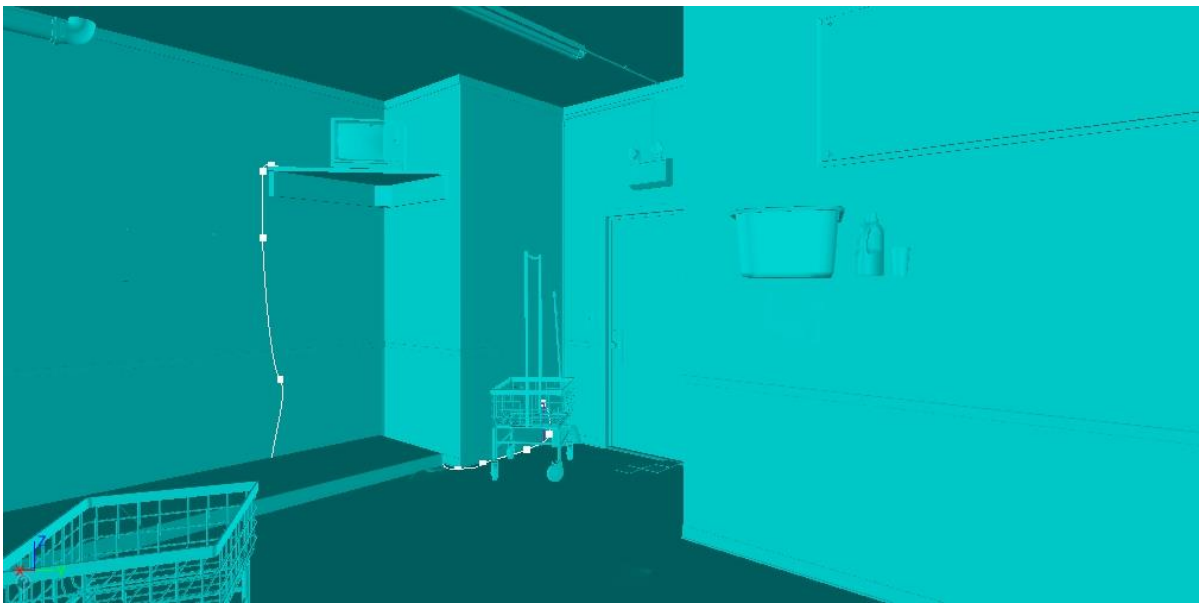

Рис. 40. Уровень из примитивов для теста механик Grey Box

Рис. 41. Уровень, текстурированный White Box

Основные типы уровня

Линейный – ответвления локации возвращают героя на основные рельсы. Пример: Call of Duty Modern Warfare.

Кластерный – цепочка центральных хабов от каждого из которых отходит несколько дополнительных путей. Пример: Resident Evil 2.

Паутина – Соединение множества кластерных уровней, по которым игрок может свободно перемещаться и проходить уровень нелинейно. Пример: Watch Dogs 2.

3.6. Технология World Composition

Не смотря на то что в игре будет открытый мир, тип уровня у него линейный. Отличия открытого мира с прогрузкой отдельных частей локации от бесшовного открытого мира отличается следующим. Можно создать огромный ландшафт и строить на нем все свои локации. Но чем больше будет по размеру локация, тем больше оперативной памяти она будет занимать и тем выше будут требования к системе игрока. Размер мира будет упираться в среднестатистического пользователя.

Unreal Engine 4 на вооружении имеет такую технологию как World Composition [8]. Технология позволяет разбивать одну огромную карту мира на много небольших секций, расстояние можно задать от секции до игрока на котором эта секция будет автоматически подгружаться в память.

Открытый мир

На карте открытого мира расположен матрица уровней 5 на 5. Где каждый уровень занимает 4 км^2, суммарно вся карта имеет площадь в 100 км^2 (рисунок 42).

1. Зеленая зона указывает расположение игрока на уровне.

2. Желтая зона указывает на разграничения между уровнями.

3. Красная зона указывает на прогруженные уровни вокруг радиуса расположения игрока.

В случаи превышения этого расстояния от игрока до секции, секция от которой игрок отдалился, автоматически будет выгружена из памяти. Расстояние настраивается так чтобы была загружена только секция, на которой игрок присутствует, а также соседние секции к ней. Таким образом размер игрового мира не ограничен ничем, и может быть сколько угодно большим. Ни каких логов и швов при этом не существует.

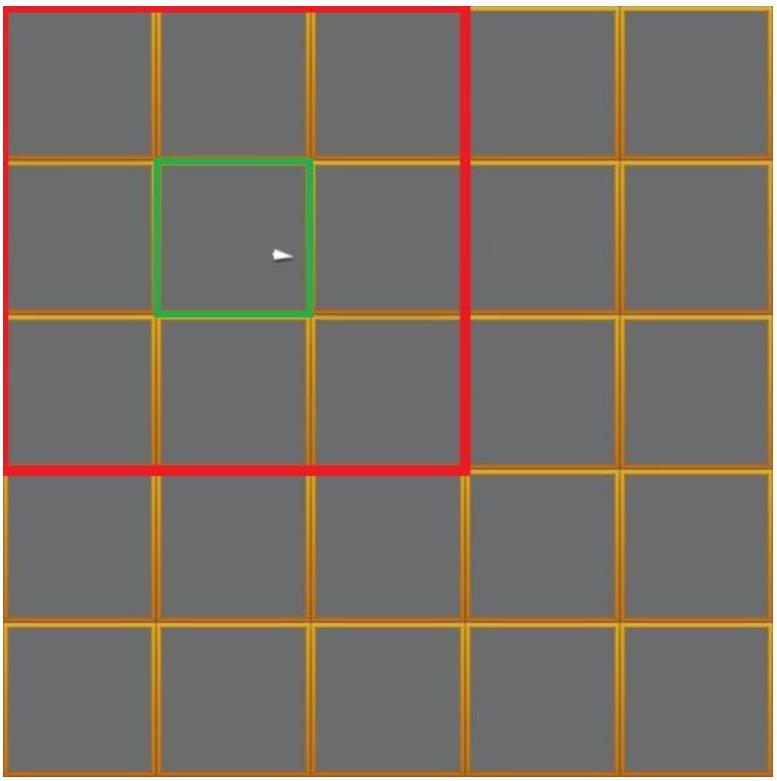

Рис. 42. Открытый мир

Шов своего рода является системой перехода между уровнями, где во время прогрузки уровня, разработчики всячески ухитрялись его замазывать во избежание нарушения атмосферы игры. В игре Resident Evil 2 1998 года во время прогрузки следующей комнаты на темном фоне от первого лица проигрывалась медленная анимация открывания двери которая сопровождалась скрипом, что придавало игре атмосферы.

4. ТЕСТИРОВАНИЕ

Тестирование производилось при помощи конечного набора тестов. [3] Функциональное тестирование реализованных классов на соответствие требованиям, составленным в пункте проектирования и тестирование интерфейса.

Функциональное тестирование

Ниже будут представлены результаты заранее отобранных тестов, соответствующих функциональным требованиям.

Тест № 1. Отображение экрана главного меню приложения.

Входные данные: - Запуск игры.

Цель: Проверить корректность отображения Главного меню.

Результат: тест пройден.

Тест № 2. Корректная работоспособность прогрузки уровня.

Входные данные: Игра открыта на экране главное меню, начать игру.

Цель: Проверить запуск после нажатия кнопки на экране.

Результат: тест пройден.

Тест № 3. Корректная работоспособность заглушки для Настройки.

Входные данные: Игра открыта на экране главное меню, Настройки.

Цель: Проверить исключение после нажатия кнопки на экране.

Результат: тест пройден.

Тест № 4. Корректный выход с игры.

Входные данные: Игра открыта на экране главное меню.

Цель: Проверить корректный выход из игры.

Результат: тест пройден.

Тест № 5. Корректная прогрузка секций карты игры.

Входные данные: Переход с секции станции в секцию пустыня.

Цель: Проверить корректность подгрузки локаций.

Результат: тест пройден.

Тест № 6. Корректное отображение предметов в инвентаре.

Входные данные: Подобрать ключ.

Цель: Получить отображение иконки ключа в инвентаре.

Результат: тест пройден.

Тест № 7. Корректное добавление предмета в инвентарь.

Входные данные: Подобрать содовую.

Цель: Проверить при переходе на другую секцию присутствие содовой.

Результат: тест пройден.

Тест № 8. Корректное удаление предмета с инвентаря.

Входные данные: Правой кнопкой мыши кликнуть по предмету.

Цель: Проверить удаляется ли предмет из инвентаря.

Результат: тест пройден.

Тест № 9. Корректное использование предмета с инвентаря.

Входные данные: Использовать медикаменты.

Цель: Проверить пополнение жизни персонажа.

Результат: тест пройден.

Тест № 10. Корректная работа радиуса видимости Enemy Character.

Входные данные: Подбежать к воронам.

Цель: Проверить дальность видимости персонажа воронами.

Результат: тест пройден.

Тест № 11. Корректная работа нанесения урона игроку.

Входные данные: Провоцировать ворон.

Цель: При приближении происходит налет и атака персонажа.

Результат: тест пройден.

Тест № 12. Корректная работа видео камеры.

Входные данные: Нажать на клавишу V вытащить камеру в темноте.

Цель: Проверить корректное освещение, а также работу батарейки.

Результат: тест пройден.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы бакалавра была разработана компьютерная 3D игра в жанре «Action» на Unreal Engine 4.

Также были выполнены поставленные задачи.

1. Произведен анализ предметной области.
2. Произведен анализ существующих аналогов.
3. Выбраны средства реализации.
4. Разработано техническое предложение.
5. Созданы 3D модели персонажей и элементы окружения.
6. Написаны скрипты, произведена сборка Block Out.
7. Произведено тестирование.

В данной главе были описаны виды тестирования, проведенные для проверки корректности работы разрабатываемого интерфейса игры. В ходе тестирования была проверена работоспособность, а также общего функционала системы. По результатам тестирования можно сделать выводы о том, что разработанная система работает корректно и функциональность соответствует заявленной.

Планируется дальнейшее развитие проекта, включающее в себя пункты, представленные ниже.

1. Разработка системы смены дня и ночи.
2. Разработка системы крафта.
3. Разработка новых моделей NPC.

ЛИТЕРАТУРА

1. Интернет-портал Unreal Engine 4. Documentation, [Электронный ресурс] URL: https://docs.unrealengine.com/latest/INT (дата обращения: 20.03.2020).

2. DeAnda M. A., Kocurek C. A. Game design as technical communication: Articulating game design through textbooks //Technical Communication Quarterly. – 2016. – Т. 25. – №. 3. – С. 202-210.

3. Kaner C., Falk J., Nguyen H. Q. Testing Computer Software. – USA: Wiley Computer Publishing, 1999. – 479 p. (дата обращения: 23.03.2020).

4. Дин Л., Дон У. Принципы работы с требованиями к программному обеспечению. Унифицированный подход. – М.: Издательский дом, «Вильямс», 2002. – 448 с. (дата обращения: 20.04.2020).

5. ГОСТ 19.701-90 (ИСО 5807-85). Единая система программной документации. 1992 – 24 с. (дата обращения: 20.03.2020).

6. Romero C. Handbook of Educational Data Mining. / C.Romero, S.Ventura, Pechenizkiy, R.S. Baker. // Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Taylor and Francis Group, LLC, 2011.– 503 p. (дата обращения: 20.03.2020).

7. Ветеранова Д. С. Основные тенденции разработки видеоигр //Информационно-компьютерные технологии в экономике, образовании и социальной сфере. – 2019. – №. 2. – С. 44-50 с.

8. Unreal Engine 4. Docs World Composition [Электронный ресурс] URL: https://docs.unrealengine.com/en-US/Engine/LevelStreaming/WorldBrowser/index.html. (дата обращения: 15.03.2020).

9. Spencer S. ZBrush Character Creation: Advanced Digital Sculpting. – John Wiley & Sons, 2011. – 216 с.

10. Белоногов Е. В., Икс Л. А. Использование искусственного интеллекта в игровой индустрии //Аллея науки. – 2019. – Т. 2. – №. 1. – С. -142.

11. Satheesh P. V. Unreal Engine 4 Game Development Essentials. – Packt Publishing Ltd, 2016. – 158 c.

12. Kirkland E. Restless dreams in Silent Hill: approaches to video game analysis //International Journal of Technology Management & Sustainable Development. – 2007. – Т. 6. – №. 3. – С. 167-178.

13. Kagen M. The Worries of a Patriarchy: Death Stranding, Hideo Kojima (2019) //Journal of Gaming & Virtual Worlds. – 2020. – Т. 12. – №. 1.

14. Melnic D. P., Melnic V. Liberty and limitation: the multifaceted representation of the sea in video games //Analele Universității „Ovidius" din Constanța. Seria Filologie.

15. Valcasara N. Unreal engine game development blueprints. – Packt Publishing Ltd, 2015.