Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехника и комплексная автоматизация»

КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Визуализация природных ландшафтов в трёхмерном движке Unreal Engine 5»

Студент РК6-83Б _____ **Козлов В.В.**
(группа) (подпись) (инициалы, фамилия)

Руководитель ВКР _____ **Витюков Ф.А.**
(подпись) (инициалы, фамилия)

Нормоконтролер _____ **Грошев С.В.**
(подпись) (инициалы, фамилия)

2025 год

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой РК6
(Индекс)
_____ Карпенко А.П.
(Фамилия И.О.)
«12» февраля 2025 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

Студент группы РК6-83Б

Козлов Вадим Владимирович
(фамилия, имя, отчество)

Тема выпускной квалификационной работы: «Визуализация природных ландшафтов в трёхмерном движке Unreal Engine 5»

При выполнении ВКР:

	Используются / Не используются	Да/Нет
1)	Литературные источники и документы, имеющие гриф секретности	Нет
2)	Литературные источники и документы, имеющие пометку «Для служебного пользования», иных пометок, запрещающих открытое опубликование	Нет
3)	Служебные материалы других организаций	Нет
4)	Результаты НИР (ОКР), выполняемой в МГТУ им. Н.Э. Баумана	Нет
5)	Материалы по незавершенным исследованиям или материалы по завершенным исследованиям, но ещё не опубликованные в открытой печати	Нет

Тема квалификационной работы утверждена распоряжением по факультету:	
Название факультета:	РК
Дата и рег. номер распоряжения:	03.07-03/07 от 13.02.2025

Часть 1. Аналитическая часть

Изучить создание материала ландшафта. Исследовать настройки материалов различных элементов. Проанализировать влияние различных объектов на производительность сцены. Исследовать отличия различных методов отражения.

Часть 2. Практическая часть 1. Визуализация сцены

Создать ландшафт и импортировать его полигональную сетку. Применить необходимые настройки материалов различных элементов. Сгенерировать элементы растительности.

Настроить отражения и создать объемный туман. Создать визуализацию палаточного лагеря и места крушения самолёта.

Часть 3. Практическая часть 2. Разработка кода пролета самолета над сценой.
Разработать программный код, реализующий пролет самолета над сценой. Учесть все необходимые условия для корректной работы. Предусмотреть удобное изменение параметров объекта.

Оформление квалификационной работы

Расчетно-пояснительная записка на 57 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

Приложение А:
Лист 1. Создание поверхности в TerreSculptor и ее импортирование в Unreal Engine;
Лист 2. Создание озера на поверхности и настройка его материала;
Лист 3. Поверхность воды без отражений и с отражениями Ray Tracing;
Лист 4. Визуальный вид плоского и объемного туманов;
Лист 5. Вид текстур поверхности после их замены, размещение деревьев и камней;
Лист 6. Редактирование текстур деревьев и граф RTV-текстур камней;
Лист 7. Графы материала плоского и объемного туманов;
Лист 8. Сцена палаточного лагеря и места крушения самолета.

Дата выдачи задания: «10» февраля 2025 г.

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до «2» июня 2025 г.

Руководитель квалификационной работы	_____	Витюков Ф.А.	10.02.2025
	(подпись)	(инициалы, фамилия)	(дата)
Студент	_____	Козлов В.В.	10.02.2025
	(подпись)	(инициалы, фамилия)	(дата)

Примечание: Задание оформляется в двух экземплярах: один выдается обучающемуся, второй хранится на кафедре.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ РК

КАФЕДРА РК6

ГРУППА РК6-83Б

УТВЕРЖДАЮ

Заведующий кафедрой РК6
(Индекс)

_____ Карпенко А.П.
(Фамилия И.О.)

«12» февраля 2025 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения выпускной квалификационной работы

Студента

Козлова Вадима Владимировича
(фамилия, имя, отчество)

Тема квалификационной работы: «Визуализация природных ландшафтов в трёхмерном движке Unreal Engine 5»

№ п/п	Наименование этапов выпускной квалификационной работы	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Должность	ФИО, подпись
1.	Задание на выполнение работы. Формулирование проблемы, цели и задач работы	11.02.2025		Руководитель ВКР	Витюков Ф.А.
2.	1 часть. Аналитическая часть	25.02.2025		Руководитель ВКР	Витюков Ф.А.
3.	Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач	02.03.2025		Заведующий кафедрой	Карпенко А.П.
4.	2 часть. Практическая часть 1	25.03.2025		Руководитель ВКР	Витюков Ф.А.
5.	3 часть. Практическая часть 2	16.05.2025		Руководитель ВКР	Витюков Ф.А.
6.	1-я редакция работы	22.05.2025		Руководитель ВКР	Витюков Ф.А.
7.	Подготовка доклада и презентации	08.06.2025		Руководитель ВКР	Витюков Ф.А.
8.	Отзыв руководителя	16.06.2025		Руководитель ВКР	Витюков Ф.А.
9.	Нормоконтроль	09.06.2025		Нормоконтролер	Грошев С.В.

10.	Защита работы на ГЭК	17.06.2025		Секретарь ГЭК	Кузьмина И.А.

Руководитель квалификационной работы _____ Витюков Ф.А. 10.02.2025
(подпись) (инициалы, фамилия) (дата)

Студент _____ Козлов В.В. 10.02.2025
(подпись) (инициалы, фамилия) (дата)

АННОТАЦИЯ

В данной работе рассматривается процесс создания интерактивной среды в трехмерном движке Unreal Engine 5. Целью выпускной квалификационной работы является разработка высококачественной виртуальной среды, которая способна обеспечить пользователю увлекательный и интерактивный опыт. В практической части ВКР была проведена разработка концепции и дизайна интерактивной среды, включающей в себя моделирование объектов, создание текстур и материалов, а также настройку освещения и атмосферы.

В ходе работы были использованы инструменты и возможности Unreal Engine 5, такие как Blueprints для визуального программирования, системы частиц для создания эффектов, а также технологии рендеринга для достижения высокого качества графики. Результатом проекта стала полноценная интерактивная среда, демонстрирующая возможности современного игрового движка и предоставляющая пользователю уникальный опыт взаимодействия с виртуальным миром. Данная работа подчеркивает значимость использования современных технологий в разработке интерактивных приложений и открывает перспективы для дальнейших исследований в области создания виртуальных сред.

Тип работы: выпускная квалификационная работа.

Тема работы: «Визуализация природных ландшафтов в трёхмерном движке Unreal Engine 5».

Объекты исследований: 3d-моделирование, разработка программного кода.

Расчетно-пояснительная записка включает 57 машинописных страницы, 18 рисунков, 1 приложение, выполнена с использованием 15 источников.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Game engine (игровой движок) – набор ключевых компонентов программного обеспечения, используемых для разработки игр и иных 3d-приложений. Как правило, инструменты движка абстрагированы от специфики конкретной игры, но могут учитывать некоторые особенности жанра – они предоставляют «базис» для разработки, «надстройку» над которым создает его пользователь.

UE5 (Unreal Engine 5) – игровой движок, разрабатываемый и поддерживаемый компанией Epic Games.

Terresculptor – редактор 3D-ландшафта.

Epic Games – американская компания, занимающаяся разработкой компьютерных игр и программного обеспечения.

Nanite – технология, позволяющая создавать сцены с высокой детализацией без перегрузки системы.

Blueprint – система визуального программирования в UE5 на основе нодов с данными (события и функции).

Quixel Bridge (QB) – торговая площадка для приобретения и скачивания различных моделей.

PFV (Procedural Foliage Volume) – инструмент, который позволяет автоматически генерировать и размещать растительность (листву, деревья, кустарники и т.д.) в заданной области уровня.

RVT (Runtime Virtual Textures) – технология, которая позволяет эффективно управлять текстурами и материалами в реальном времени, обеспечивая более высокое качество визуализации и производительность.

Actor (актор, актер) – в рамках движка UE5 любой объект, который может быть размещен на уровне. Базовый класс всех actor’ов – AActor.

Actor Component – специальный тип объекта, который может быть присоединен к выбранному actor’у как подобъект (subobject). Как правило, используется для внедрения функциональности, общей для различных actor’ов. Базовый класс – UActorComponent.

Level (уровень) – в рамках движка UE5 3d-сцена. Обычно состоит из мешей, акторов, источников освещения и т.д.

Blueprint Visual Scripting – система визуального скриптинга в UE5. Классы, созданные с помощью этой системы, называются *Blueprints (блюпринты)*.

Rendering (рендеринг, отрисовка) – процесс создания 2d-изображения исходя из имеющейся 3d-модели.

Текстура – изображение, накладываемое на поверхность 3d-модели. Может содержать одно или несколько свойств поверхности, например: цвет, жёсткость (roughness), смещение (displacement), направление нормалей (normal map), и т.д.

Текстурирование – процесс создания текстур объекта.

Материал – набор свойств, определяющих поведение света при отражении от поверхности. Материалы также могут использовать одну или несколько текстур.

Material (UMaterial) – класс, позволяющий хранить и модифицировать информацию о материале.

Static Mesh Component (UStaticMeshComponent) – компонент, который может иметь статический меш и набор материалов, применимых к нему.

Коллизия (collision, столкновение) – пересечение между двумя или более объектами сцены.

Geometry instancing (инстансинг, дублирование геометрии) – техника, позволяющая отрисовывать множество однотипных элементов за один проход. Элементы называются инстансами.

Instanced Static Mesh Component (UInstancedStaticMeshComponent, ISMC) – класс, производный от UStaticMeshComponent, позволяющий использовать механизм инстансинга геометрии.

Hierarchical Instanced Static Mesh Component (HISMC, UHierarchicalInstancedStaticMeshComponent) – класс, производный от UInstancedStaticMeshComponent, позволяющий использовать LOD.

LOD (Level of Detail, уровни детализации) – техника, позволяющая подменять разные по детализации версии модели в зависимости от дистанции между камерой и объектом, либо в зависимости от процента площади экрана, занимаемой моделью.

CPU (Central Processing Unit) – центральный процессор.

GPU (Graphics Processing Unit) – графический процессор (видеокарта).

FPS (Frames per second) – количество кадров в секунду.

Asset (ассет) – в контексте компьютерных игр, объект, представляющий собой единицу контента. Игровыми ассетами являются 3d-модели, текстуры, материалы, аудиофайлы, и т.д. Как правило, созданием ассетов занимаются художники, дизайнеры, музыканты.

3d-model (3d-модель) – математическое представление объекта в трехмерном пространстве.

3d-modeling (3d-моделирование) – процесс создания 3d-модели объекта.

3d hard-surface modeling (3d-моделирование твердых поверхностей) – процесс создания 3d-моделей объектов, имеющих гладкие поверхности и острые углы. Как правило, в данную категорию входят неживые объекты, например: механизмы, машины, здания, мебель.

3d organic modeling (3d-моделирование органических объектов) – процесс создания 3d-моделей объектов живой природы и объектов, имеющих похожие на них формы. В данную категорию входят такие объекты моделирования, как люди, животные, гуманоиды, растения.

High-poly модель (высокополигональная модель) – максимально детализированная версия 3d-модели. Имеет большое количество полигонов (от нескольких сотен тысяч до миллионов), в основном используется для дальнейшего запекания текстур. Как правило, высокополигональные модели не используются при отрисовке в реальном времени, поскольку для их обработки требуется слишком много вычислительных ресурсов.

Low-poly модель (низкополигональная модель) – модель, содержащая относительно низкое количество полигонов (несколько тысяч), при этом

сохраняющая основные геометрические свойства объекта. Низкополигональные модели широко используются при отрисовке в реальном времени, поскольку затраты на их обработку приемлемы для получения достаточно плавного изображения.

Transform (трансформ) – данные о местоположении, повороте и масштабе объекта. Представляются матрицей преобразований.

Scene Component (USceneComponent) – класс, производный от UActorComponent. Представляет собой компонент, который может иметь свой трансформ на сцене. Данный компонент используется для внедрения функциональности, не требующей геометрического представления.

Полигональная сетка – набор вершин, рёбер и граней, определяющий внешний вид многогранного объекта.

Полигон – многоугольник, являющийся гранью или набором граней 3d-сетки. Основные типы: треугольник (tri), четырёхугольник (quad) и n-gon (5 или более вершин).

Sculpting (скульптинг) – процесс создания и детализации 3d-моделей с помощью 3d-кистей.

Polycount – количество полигонов модели.

Ретопология – процесс изменения топологии полигональной сетки модели с целью ее упрощения/улучшения.

UV-unwrapping (UV-развертка) – процесс создания развертки 3d-модели. Представляет собой процесс получения соответствия между координатами поверхности 3d-модели и координатами на текстуре (U по горизонтали, V по вертикали).

Rendering Hardware Interface (RHI) – в UE5 надстройка над множеством графических API (например: DirectX, OpenGL, Vulkan), позволяющая писать независимый от платформы код.

Reference (референс) – изображение, используемое художником в процессе 3d-моделирования для получения дополнительной информации о моделируемом объекте.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Unreal Engine 5 был разработан компанией Epic Games и официально представлен в мае 2020 года. Одним из ключевых аспектов разработки является создание интерактивных сред, которые могут реагировать на действия пользователя, изменяя своё состояние в зависимости от взаимодействия. В данной работе рассматривается разработка ландшафта и программного кода для интерактивных элементов сцены. Работа делится на две основные части:

1. Визуализация ландшафта сцены.
2. Разработка программного кода для полета самолета над сценой.

Часть первая. Визуализация ландшафта сцены

В этой части подробно рассматриваются:

- Инструменты ландшафта: UE5 предоставляет мощные инструменты для создания ландшафта, которые позволяют разработчикам формировать рельеф с помощью различных кистей.
- Размеры и масштаб: при создании ландшафта важно учитывать масштаб сцены. UE5 позволяет задавать размеры ландшафта, что дает возможность создавать как небольшие, так и обширные игровые миры.
- Материалы: UE5 поддерживает создание сложных материалов с использованием системы Material Editor. В движке есть возможность комбинировать текстуры, добавлять карты нормалей и использовать различные шейдеры для достижения реалистичного внешнего вида.
- Текстурные слои: с помощью системы слоев вы можете накладывать несколько текстур на один участок ландшафта, что позволяет создавать разнообразные поверхности, такие как трава, камни и песок.
- Инструменты для размещения растительности: UE5 предлагает инструменты для автоматического размещения растительности, такие как Foliage Tool.

- Динамические объекты: также можно добавлять статические и динамические объекты, такие как здания, дороги и другие элементы, которые обогащают игровую среду и делают её более интерактивной.
- Система Lumen: UE5 включает в себя Lumen — новую систему глобального освещения, которая позволяет создавать реалистичное освещение и тени в реальном времени. Это значительно улучшает атмосферу сцены, делая её более живой и динамичной.
- Эффекты погоды и времени суток: имеется возможность настройки эффекты погоды, такие как дождь или туман, а также изменять время суток, что добавляет дополнительный уровень реализма и погружения.
- Nanite: UE5 использует технологию Nanite, которая позволяет загружать и отображать высокодетализированные модели без значительного влияния на производительность. Это означает возможность использовать более сложные геометрические формы и текстуры, не беспокоясь о снижении FPS.
- Оптимизация ландшафта: важно оптимизировать ландшафт для обеспечения плавной работы игры. Это включает в себя использование LOD (уровней детализации) для объектов и текстур, а также правильное распределение ресурсов.

Часть вторая. Разработка программного кода для полета самолета над сценой.

В этой части рассматривается разработка программного кода для создания самолета, который состоит из:

- Разработки класса самолета.
- Определения начальной и конечной позиций.
- Реализации логики движения.
- Обеспечения плавного удаления объекта из поля зрения.
- Тестирования.

Во второй части рассматриваются все файлы с кодом, требуемые для реализации поставленной задачи, а также их взаимосвязь между собой. Затем представлены результаты тестирования с заданными параметрами – начальной и конечной позициями, а также временем полета. В конце были проанализированы результаты: найдем ключевые преимущества и недостатки реализации данного класса, а также предложения по возможному дальнейшему улучшению.

Цели работы:

1. создать и оптимизировать ландшафт сцены и его компоненты;

2. написать программный код для реализации полета самолета в сцене.

Для выполнения работы были использованы средства следующих программ:

- Epic Games Launcher – лаунчер для запуска Unreal Engine.
- Unreal Engine 5 – разработка внутриигровых систем.
- Terresculptor – создание базовой поверхности.
- Visual Studio – написание кода на C++ для материалов и самолета.
- Nvidia App – замер кадров в секунду в сцене.
- MSI Afterburner – тестирование оптимизации.
- Blender – создание моделей некоторых объектов.

1 Основные этапы и методы создания интерактивной среды

Создание интерактивной среды в Unreal Engine 5 — это увлекательный и многогранный процесс, который сочетает в себе элементы дизайна, программирования и художественного творчества. На первом этапе важно определить концепцию и цели проекта. Это включает в себя выбор жанра, стиля и целевой аудитории, что поможет сформировать общее видение интерактивной среды [1].

Следующим шагом является создание базовой геометрии и ландшафта. Unreal Engine 5 предлагает мощные инструменты для моделирования, такие как Nanite, которые позволяют создавать высокодетализированные объекты без значительных затрат на производительность. Важно уделить внимание не только внешнему виду, но и функциональности среды, чтобы она была не только красивой, но и удобной для взаимодействия. После этого следует этап текстурирования и освещения. Использование Lumen для глобального освещения позволяет добиться реалистичных эффектов освещения и теней, что значительно улучшает атмосферу среды. Текстуры и материалы также играют ключевую роль в создании погружающего опыта, поэтому стоит уделить внимание их качеству и разнообразию.

Затем необходимо добавить интерактивные элементы, такие как объекты. С ними а также механиками, которые будут определять поведение среды, может взаимодействовать игрок. Это может включать в себя программирование логики игры с помощью Blueprints или C++, что позволяет создавать уникальные сценарии и взаимодействия.

Наконец, тестирование и оптимизация являются критически важными этапами. Необходимо убедиться, что среда работает плавно на различных устройствах и что все интерактивные элементы функционируют корректно. Этот процесс может включать в себя сбор отзывов от тестировщиков и внесение изменений на основе их рекомендаций.

1.1 Начальные этапы: планирование и импортирование

Перед началом разработки важно иметь четкое представление о том, какую среду нужно создать. Это может быть концептуальный арт, сценарий или даже прототип. Необходимо определение цели, атмосферы и основных элементов, которые должны присутствовать в интерактивной среде. Графы для создания сцены в Unreal Engine представляют собой мощный инструмент, который позволяет разработчикам визуально проектировать и настраивать игровые уровни и сцены. Эти графы используют узловую систему, что делает процесс создания более интуитивным и доступным, особенно для тех, кто не имеет глубоких знаний в программировании (рисунок 1).

Основные компоненты графов для создания сцены:

Узлы: каждый узел в графе представляет собой определённую функцию или действие, которое можно выполнить. Узлы могут быть связаны между собой, создавая цепочки логики, что позволяет разработчикам настраивать поведение объектов, взаимодействия и другие аспекты сцены.

События: графы позволяют обрабатывать различные события, такие как нажатия клавиш, столкновения объектов или изменения состояния игры. Это позволяет создавать динамичные и интерактивные элементы в сцене.

Переменные: разработчики могут создавать и использовать переменные для хранения данных, таких как позиции объектов, состояния игры или параметры анимации. Это позволяет легко управлять данными и изменять их в зависимости от логики игры.

Компоненты: графы позволяют добавлять различные компоненты к объектам, такие как коллайдеры, материалы и анимации. Это упрощает процесс настройки объектов и их взаимодействий в сцене.

Дебаггинг: Unreal Engine предоставляет инструменты для отладки графов, что позволяет разработчикам отслеживать выполнение логики и находить ошибки. Это особенно полезно при создании сложных взаимодействий и механик.

Применение графов для создания сцены:

Графы для создания сцены используются для разработки различных аспектов игры, включая управление персонажами, взаимодействие с окружением, анимацию и многое другое. Они позволяют разработчикам быстро вносить изменения и тестировать новые идеи, что значительно ускоряет процесс разработки.

Преимущества:

Интуитивность: узловая система делает графы доступными для пользователей с разным уровнем подготовки, позволяя сосредоточиться на творческом процессе.

Гибкость: разработчики могут легко изменять и настраивать логику, добавляя новые узлы или изменяя существующие.

Визуализация: графы предоставляют наглядное представление логики, что упрощает понимание и работу с проектом.

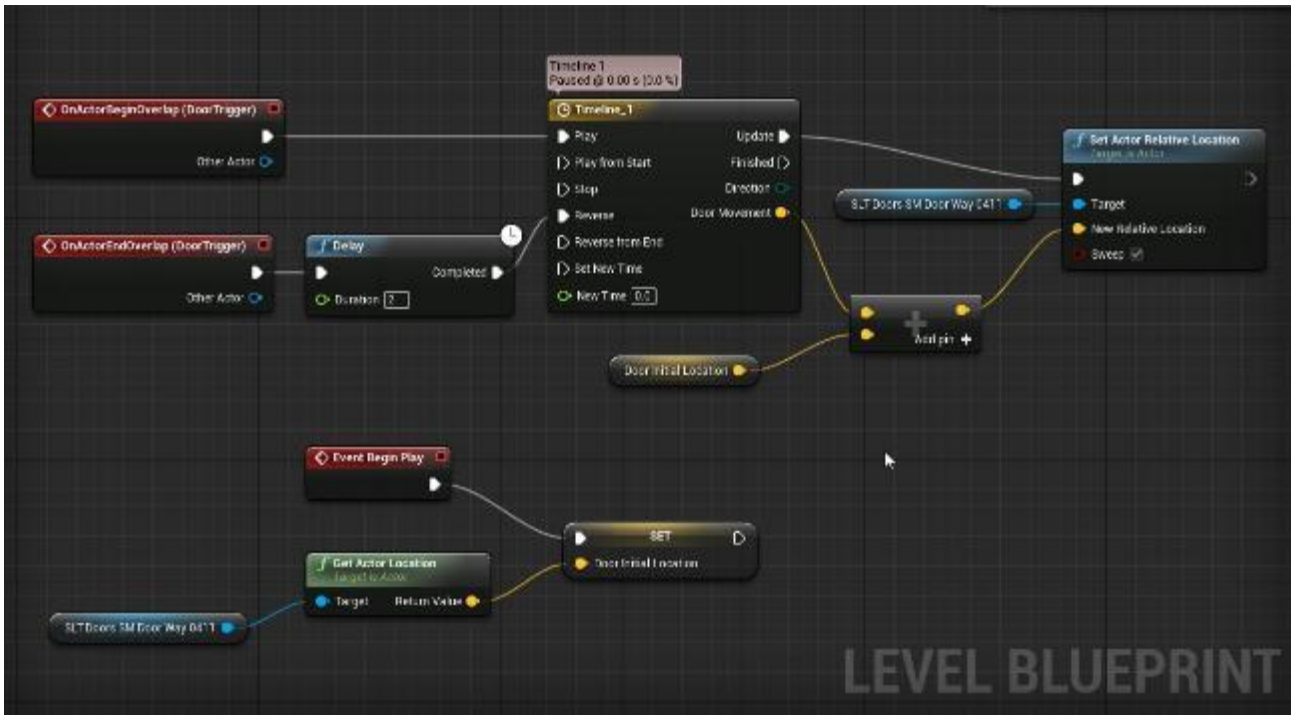

Рисунок 1. Граф для создания сцены в Unreal Engine

UE5 предоставляет мощные инструменты для создания уровней:

Редактор уровней в Unreal Engine — это мощный инструмент, который позволяет разработчикам создавать, редактировать и управлять игровыми

мирами. Он предоставляет интуитивно понятный интерфейс для работы с различными элементами сцены, такими как ландшафты, объекты, освещение и эффекты. Редактор уровней включает в себя множество функций, которые упрощают процесс проектирования и позволяют быстро вносить изменения.

Основные функции редактора уровней:

Инструменты для размещения объектов: разработчики могут легко добавлять и перемещать объекты в сцене, используя инструменты для трансформации, такие как перемещение, вращение и масштабирование. Это позволяет точно настраивать расположение элементов в игровом мире.

Работа с ландшафтами: редактор уровней предоставляет инструменты для создания и редактирования ландшафтов, включая возможность рисовать рельеф, добавлять текстуры и настраивать материалы. Это позволяет создавать разнообразные и реалистичные природные окружения.

Система освещения: разработчики могут настраивать освещение в сцене, используя различные источники света, такие как точечные, направленные и окружные источники. Это позволяет создавать атмосферу и улучшать визуальное восприятие игры.

Управление уровнями: редактор уровней позволяет работать с несколькими уровнями одновременно, что упрощает организацию и управление контентом. Разработчики могут создавать подуровни и использовать их для разделения различных частей игрового мира.

World Partition:

World Partition — это новая система управления уровнями, представленная в Unreal Engine 5, которая значительно упрощает создание и управление большими открытыми мирами. Она позволяет разбивать игровые уровни на более мелкие секции, которые загружаются и выгружаются по мере необходимости, что обеспечивает более эффективное использование ресурсов (рисунок 2).

Основные особенности World Partition:

Динамическая загрузка: World Partition автоматически управляет загрузкой и выгрузкой секций уровня в зависимости от положения игрока и других факторов. Это позволяет поддерживать высокую производительность даже в больших мирах.

Упрощенное проектирование: разработчики могут работать с большими мирами без необходимости вручную управлять уровнями и их загрузкой. Это упрощает процесс проектирования и позволяет сосредоточиться на создании контента.

Интеграция с редактором уровней: World Partition интегрирован с редактором уровней, что позволяет разработчикам легко переключаться между различными секциями и управлять ими. Это делает процесс работы более интуитивным и удобным.

Поддержка многопользовательских игр: система World Partition также оптимизирована для многопользовательских игр, позволяя нескольким игрокам взаимодействовать с одним и тем же миром без потери производительности.

Преимущества:

Использование редактора уровней и системы World Partition в Unreal Engine 5 позволяет разработчикам создавать масштабные и детализированные игровые миры, упрощая процесс разработки и улучшая производительность. Эти инструменты обеспечивают гибкость и удобство, что делает их важными для создания современных игр.

World Partition — это новая система управления уровнями, которая позволяет разбивать большие игровые миры на более мелкие секции, которые загружаются и выгружаются по мере необходимости. Это обеспечивает более эффективное использование ресурсов и позволяет разработчикам создавать обширные открытые миры без потери производительности. World Partition автоматически управляет загрузкой и выгрузкой секций, что позволяет сосредоточиться на создании контента, а не на технических аспектах оптимизации.

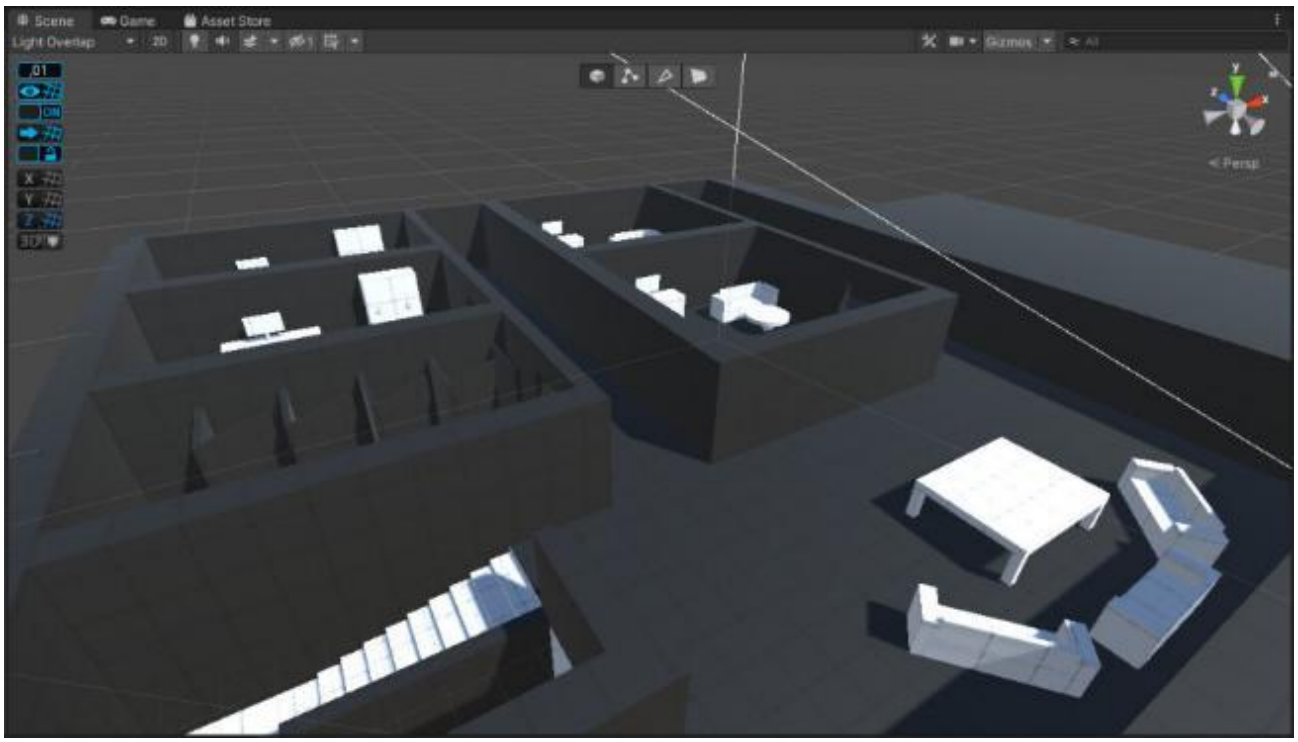

Рисунок 2. Пример визуального уровня в Unreal Engine

В Unreal Engine 5 (UE5) имеется возможность импортировать 3D-модели, текстуры и анимации из различных сторонних программ, таких как Blender, Maya или 3ds Max. Это значительно упрощает процесс интеграции контента в игровые проекты. Это позволяет разработчикам использовать свои любимые инструменты для создания и редактирования 3D-ресурсов, а затем легко переносить их в UE5 для дальнейшей работы.

Импорт 3D-моделей:

Разработчики могут экспортировать 3D-модели из таких программ, как Blender, Maya или 3ds Max в форматах, поддерживаемых Unreal Engine, например, FBX или OBJ. Эти форматы позволяют сохранить геометрию, текстуры и материалы модели. При импорте в UE5 пользователи могут настраивать параметры импорта, такие как масштаб, ориентация и настройки материалов, что обеспечивает гибкость в процессе интеграции (рисунок 3).

Импорт текстур:

Текстуры, созданные в графических редакторах, таких как Photoshop или Substance Painter, также могут быть импортированы в UE5. Поддерживаются различные форматы изображений, включая PNG, JPEG и TGA. После импорта

текстуры могут быть использованы для создания материалов, которые придают объектам реалистичный внешний вид. UE5 позволяет настраивать параметры текстур, такие как масштаб, повторение и фильтрация, что дает возможность добиться желаемого визуального эффекта.

Импорт анимаций:

Анимации, созданные в 3D-редакторах, могут быть экспортированы вместе с моделями в формате FBX. UE5 поддерживает импорт анимаций для персонажей и объектов, что позволяет разработчикам легко интегрировать анимационные последовательности в свои проекты. При импорте анимаций можно настроить параметры, такие как количество кадров в секунду и типы анимаций (например, скелетные или морфинг).

Создание контента непосредственно в движке:

Кроме импорта, Unreal Engine 5 также предоставляет возможность создавать 3D-модели, текстуры и анимации непосредственно в движке. Разработчики могут использовать встроенные инструменты, такие как Geometry Editing для создания и редактирования геометрии, Material Editor для создания материалов и Animation Blueprint для настройки анимаций. Это позволяет быстро прототипировать идеи и вносить изменения в реальном времени, что значительно ускоряет процесс разработки.

Nanite — это революционная виртуализированная геометрическая система в Unreal Engine 5, которая позволяет разработчикам использовать высокодетализированные модели без необходимости в ручной оптимизации. Эта система предназначена для работы с миллионами полигонов в реальном времени, что значительно упрощает процесс создания контента и позволяет достигать невероятного уровня детализации в играх и приложениях.

Основные особенности Nanite:

Виртуализация геометрии: Nanite автоматически управляет подгрузкой и отображением геометрии, что позволяет разработчикам использовать модели с высоким количеством полигонов, не беспокоясь о производительности. Это

означает, что можно использовать детали, которые ранее были невозможны для рендеринга в реальном времени.

Динамическое управление уровнем детализации: система автоматически подстраивает уровень детализации в зависимости от расстояния до камеры. Это позволяет сохранять высокое качество изображения на близком расстоянии, в то время как более удаленные объекты отображаются с меньшей детализацией, что оптимизирует производительность.

Поддержка различных форматов: Nanite поддерживает импорт моделей из различных 3D-редакторов, что позволяет разработчикам легко интегрировать свои активы в Unreal Engine. Это упрощает процесс работы с контентом и позволяет использовать существующие ресурсы.

Интеграция с другими системами: Nanite работает в связке с другими функциями Unreal Engine, такими как Lumen для освещения, что позволяет создавать реалистичные сцены с высоким качеством графики.

Система материалов:

Система материалов в Unreal Engine 5 также претерпела значительные улучшения, предоставляя разработчикам более мощные инструменты для создания реалистичных материалов. Она основана на узловой системе, что позволяет легко настраивать внешний вид объектов и их взаимодействие с освещением.

Основные особенности системы материалов:

Узловая система: разработчики могут создавать материалы, используя узлы, которые представляют различные функции и параметры. Это позволяет легко комбинировать и настраивать материалы, создавая уникальные визуальные эффекты.

Поддержка виртуальных текстур: система материалов поддерживает виртуальные текстуры, что позволяет оптимизировать использование текстур и улучшить производительность. Это особенно полезно для больших проектов с множеством активов.

Анимация материалов: разработчики могут анимировать параметры материалов, что позволяет создавать динамичные эффекты, такие как изменение цвета, прозрачности или текстуры в зависимости от условий игры.

Интеграция с Nanite: система материалов работает в связке с Nanite, что позволяет использовать высококачественные текстуры и детали на моделях без потери производительности.

Преимущества:

Использование Nanite и улучшенной системы материалов в Unreal Engine 5 позволяет разработчикам создавать визуально впечатляющие и детализированные игровые миры. Эти инструменты упрощают процесс разработки, позволяя сосредоточиться на творческих аспектах, и обеспечивают высокое качество графики, что значительно улучшает пользовательский опыт.

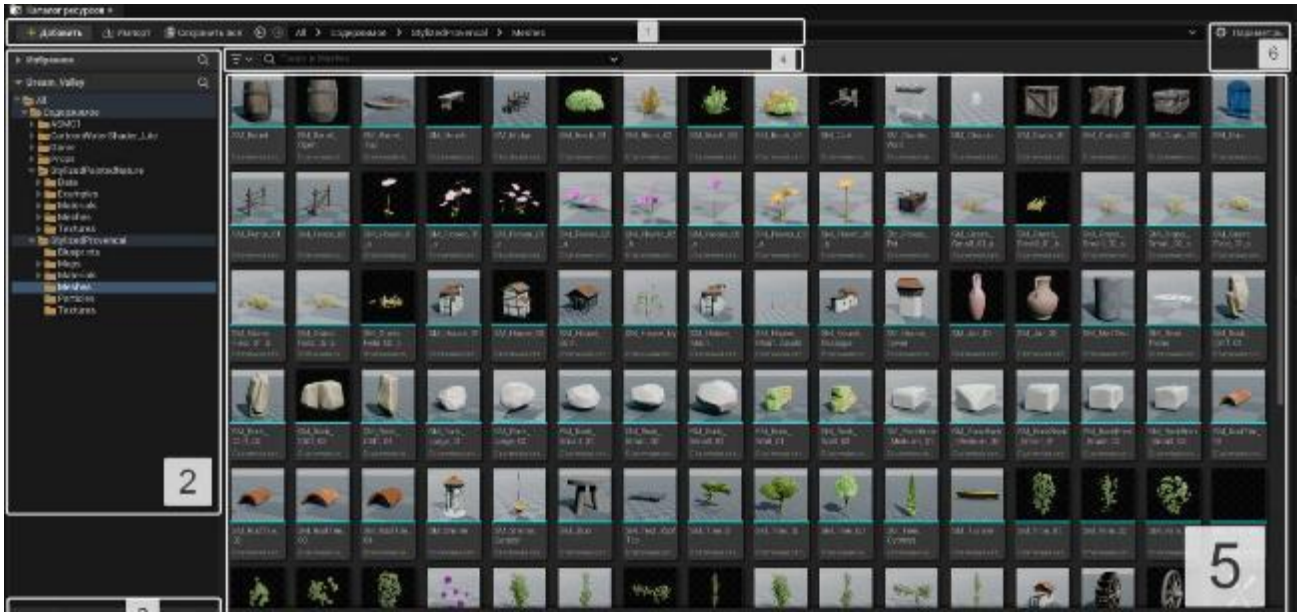

Рисунок 3. Каталог ресурсов, встроенный в движок

1.2 Атмосфера и интерактивность

Для создания динамического освещения и реалистичных отражений используется система Lumen. Это новая система глобального освещения и отражений, представленная в Unreal Engine 5, которая обеспечивает реалистичное освещение в реальном времени. Она предназначена для упрощения процесса разработки, позволяя разработчикам сосредоточиться на творческих аспектах, а не на сложной настройке освещения. Lumen работает на основе динамического глобального освещения, что позволяет свету взаимодействовать с объектами в сцене, создавая естественные тени и отражения (рисунок 4).

Рисунок 4. Пример сцены в Unreal Engine с освещением Lumen

Основные особенности Lumen:

Динамическое глобальное освещение: Lumen использует алгоритмы, которые позволяют свету отражаться от поверхностей и взаимодействовать с окружающей средой в реальном времени. Это означает, что изменения в освещении, такие как перемещение источников света или изменение материалов, мгновенно отражаются в сцене. Lumen поддерживает высококачественные отражения, которые учитывают как статические, так и динамические объекты. Это позволяет создавать реалистичные водные поверхности, зеркала и другие

отражающие материалы, что значительно улучшает визуальное восприятие. Система интегрирована в Unreal Engine 5, что делает ее доступной для разработчиков без необходимости в сложной настройке. Это позволяет быстрее достигать высококачественных визуальных эффектов и сосредоточиться на создании контента. Lumen оптимизирован для работы на различных платформах, включая консоли и ПК, что делает его универсальным инструментом для разработчиков.

Другие системы освещения в Unreal Engine:

Помимо Lumen, Unreal Engine предлагает и другие системы освещения, которые могут быть использованы в зависимости от потребностей проекта.

Static Lighting: это система, которая использует предварительно рассчитанные карты освещения (lightmaps) для статических объектов. Она обеспечивает высокое качество освещения, но требует больше времени на предварительную настройку и не поддерживает динамические изменения в освещении. Stationary Lighting: эта система позволяет использовать как статическое, так и динамическое освещение. Она поддерживает динамические тени для движущихся объектов, в то время как статические объекты используют lightmaps. Это обеспечивает баланс между качеством и производительностью. Dynamic Lighting: полностью динамическое освещение, которое позволяет источникам света изменять свое положение и параметры в реальном времени. Это обеспечивает максимальную гибкость, но может негативно сказаться на производительности, особенно в больших сценах с множеством источников света. Light Propagation Volumes (LPV): это система, которая использует объемы для распространения света в сцене. Она позволяет создавать динамическое глобальное освещение, но может быть менее точной по сравнению с Lumen.

Применение систем освещения:

Выбор системы освещения зависит от требований проекта и желаемого уровня реализма. Lumen идеально подходит для современных игр и приложений, где важна высокая степень реализма и динамичность. В то время как статическое

освещение может быть предпочтительным для проектов с фиксированными сценами, где производительность является критически важной.

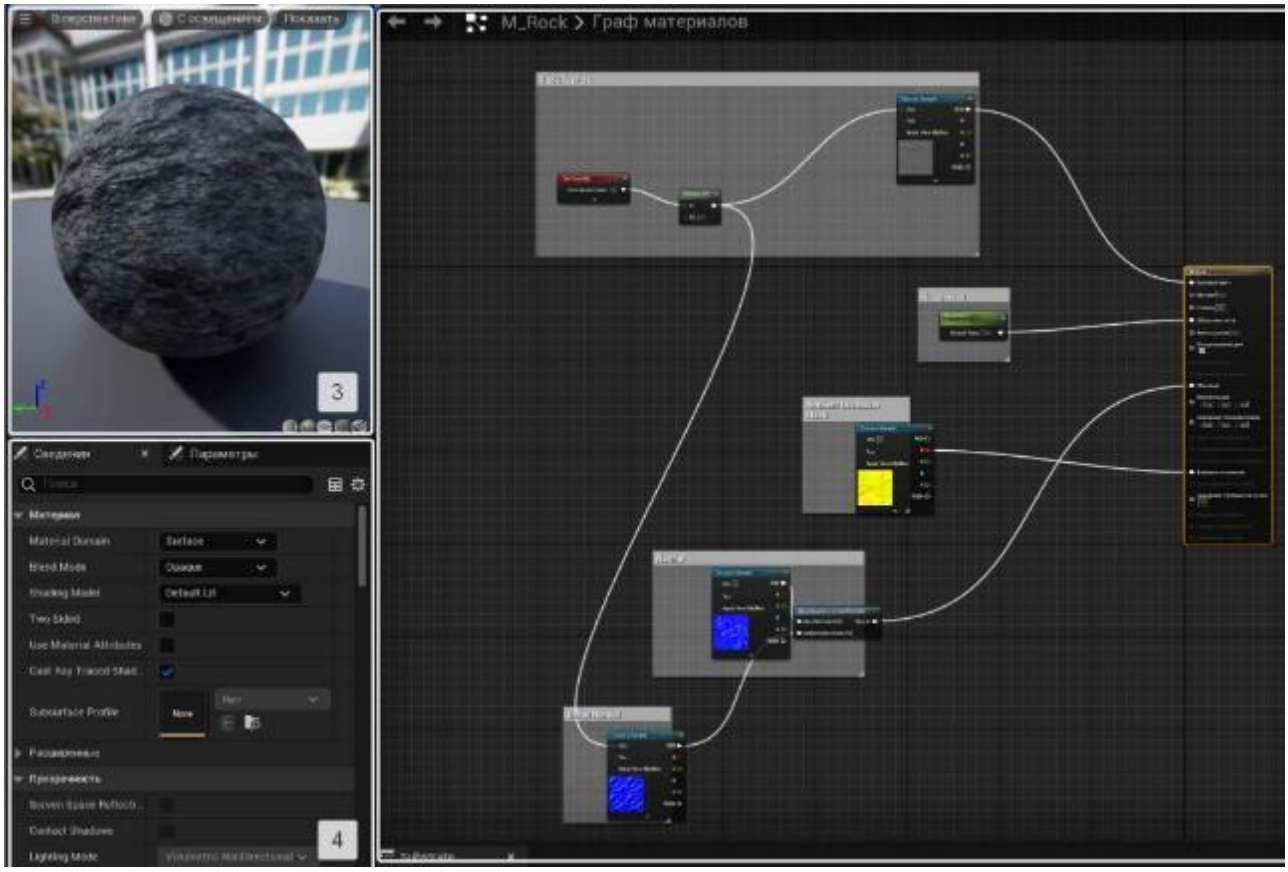

Рисунок 5. Настройка материала с помощью графа материалов

Настройка материалов в Unreal Engine с помощью графа материалов — это мощный и интуитивно понятный процесс. Он позволяет разработчикам создавать сложные и реалистичные материалы для объектов в игре. Граф материалов использует узловую систему, где каждый узел представляет собой определённую функцию или параметр, что позволяет легко комбинировать и настраивать различные аспекты материалов (рисунок 5).

Основные компоненты графа материалов:

- Узлы: каждый узел в графе выполняет определённую задачу, такую как создание текстуры, настройка цвета, управление прозрачностью или добавление эффектов. Узлы могут быть связаны между собой, создавая цепочки логики, что позволяет комбинировать различные функции.

- Входные и выходные параметры: узлы имеют входные и выходные параметры, которые позволяют передавать данные между узлами. Например, выходной параметр одного узла может быть входным параметром для другого, что позволяет создавать сложные связи и эффекты.
- Текстуры: в графе материалов можно использовать текстуры, которые добавляют детали и реалистичность материалам. Текстуры могут быть загружены из внешних файлов или созданы непосредственно в Unreal Engine.
- Параметры: разработчики могут создавать параметры, которые позволяют настраивать свойства материалов в реальном времени. Это может быть полезно для создания материалов, которые изменяются в зависимости от условий игры, таких как цвет, прозрачность или отражение.

Процесс создания материала:

- Создание нового материала: для начала необходимо создать новый материал в контент-браузере Unreal Engine. Это делается с помощью правого клика и выбора опции "Material". После этого откроется граф материалов.
- Добавление узлов: в графе материалов можно добавлять узлы, перетаскивая их из панели узлов или используя контекстное меню. Например, можно добавить узел "Texture Sample" для загрузки текстуры, узел "Multiply" для умножения значений или узел "Lerp" для смешивания цветов.
- Настройка узлов: каждый узел можно настраивать, изменяя его параметры в панели деталей. Например, для узла текстуры можно выбрать конкретную текстуру, а для узла цвета — задать RGB-значения.
- Соединение узлов: узлы соединяются между собой с помощью линий, которые представляют собой потоки данных. Например,

выходной параметр узла текстуры может быть подключен к входному параметру узла цвета, что позволяет использовать текстуру в качестве цвета материала.

- Вывод материала: в конце графа материалов должен быть узел "Material Output", который определяет, как материал будет выглядеть на объекте. Этот узел принимает входные параметры, такие как базовый цвет, металлическость, шероховатость и другие свойства.
- Сохранение и применение: после завершения настройки материала его необходимо сохранить. Затем материал можно применить к объектам в сцене, перетаскивая его на нужный объект или назначая в панели деталей.

Примеры использования графа материалов:

- Создание реалистичных поверхностей: с помощью графа материалов можно создавать различные типы поверхностей, такие как металл, дерево, камень и другие, комбинируя текстуры и настраивая параметры отражения и шероховатости.
- Динамические материалы: разработчики могут создавать материалы, которые изменяются в зависимости от условий игры, например, меняя цвет в зависимости от времени суток или состояния объекта.
- Эффекты: Граф материалов позволяет добавлять различные визуальные эффекты, такие как прозрачность, свечение или анимацию, что значительно обогащает визуальное восприятие игры.
- Граф материалов в Unreal Engine предоставляет разработчикам мощные инструменты для создания и настройки материалов, позволяя достигать высокого уровня детализации и реализма. Интуитивно понятный интерфейс и узловая система делают процесс создания материалов доступным даже для новичков, что способствует более творческому подходу к разработке игр.

1.3 Тестирование и оптимизация

Интерактивная среда подвергается постоянным тестам, чтобы убедиться, что все работает как задумано. Налаживается оптимизация производительности, чтобы обеспечить плавный игровой процесс. Тестирование и оптимизация интерактивной среды в Unreal Engine 5 — это ключевые этапы, которые обеспечивают не только стабильность и производительность проекта, но и общее качество игрового опыта. На начальном этапе тестирования важно выявить и устранить ошибки, которые могут повлиять на взаимодействие игрока с окружением. Это включает в себя проверку всех интерактивных элементов, механик и сценариев, чтобы убедиться, что они работают так, как задумано.

Оптимизация, в свою очередь, направлена на улучшение производительности проекта. В условиях современных игровых требований важно, чтобы среда работала плавно на различных устройствах. Использование инструментов, таких как Profiler и Stat Commands, позволяет разработчикам анализировать производительность и выявлять узкие места, которые могут замедлять игру. Это может включать в себя оптимизацию текстур, моделей и освещения, а также LOD для объектов.

Кроме того, тестирование должно охватывать различные аспекты пользовательского опыта, включая управление, интерфейс и общую атмосферу. Сбор отзывов от тестировщиков и игроков помогает выявить проблемные области и улучшить взаимодействие с игрой. Важно учитывать, что оптимизация не должна негативно сказываться на визуальном качестве, поэтому разработчики должны находить баланс между производительностью и эстетикой.

2 Разработка ландшафта сцены

Разработка ландшафта сцены в Unreal Engine 5 включает в себя использование мощных инструментов для создания реалистичных и детализированных природных окружений. С помощью системы ландшафта можно легко формировать рельеф, добавлять текстуры и настраивать материалы, чтобы добиться максимальной реалистичности. Также доступны инструменты для работы с растительностью, позволяющие размещать деревья, кустарники и другие элементы флоры, что значительно обогащает визуальную составляющую сцены. Кроме того, Unreal Engine 5 предлагает возможности для использования Nanite и Lumen, что позволяет создавать высокодетализированные объекты и динамическое освещение, улучшая общую атмосферу и погружение в игровую среду.

2.1 Визуализация ландшафта сцены

Создание ландшафта в Unreal Engine 5 — это увлекательный процесс, который позволяет разработчикам и художникам создавать реалистичные и впечатляющие игровые миры. Первым шагом является создание нового уровня. После создания уровня идет настройка параметров ландшафта, таких как размер, количество компонентов и разрешение. Это позволяет определить, насколько детализированным будет ландшафт. Затем используются инструменты редактирования для формирования рельефа. Имеются возможности поднимать, опускать, сглаживать и изменять текстуру поверхности, используя различные кисти, которые помогут создать горы, долины и другие природные формы.

После создания базовой формы ландшафта, следующим шагом будет текстурирование. Для этого можно использовать материалы, которые уже есть в библиотеке UE5, или создать свои собственные. Важной деталью сцены является свет. Правильное освещение может значительно улучшить внешний вид ландшафта сцены. В UE имеются встроенные утилиты Directional Light для создания солнечного света и Sky Light для добавления глобального освещения.

Также можно настроить атмосферные эффекты, такие как туман и облака, чтобы добавить глубину и реализм.

2.1.1 Создание ландшафта в TerreSculptor и его импорт в Unreal Engine

TerreSculptor 2.0 — мощный инструмент для создания ландшафтов, освоив возможности которого, можно создавать впечатляющие и реалистичные поверхности. Создание поверхности в программе TerreSculptor 2.0 — это многоэтапный процесс, который включает в себя несколько ключевых шагов. Если имеются существующие данные (например, высотные карты или текстуры), есть возможность импортировать их в проект. TerreSculptor поддерживает различные форматы файлов.

Использование инструментов для создания базовой поверхности (рисунок 6). Поверхность сделана с помощью генераторов ландшафта, таких как:

Noise (шум) — для создания случайных форм.

Perlin Noise — для более естественных и органических форм.

Fractal — для создания фрактальных ландшафтов.

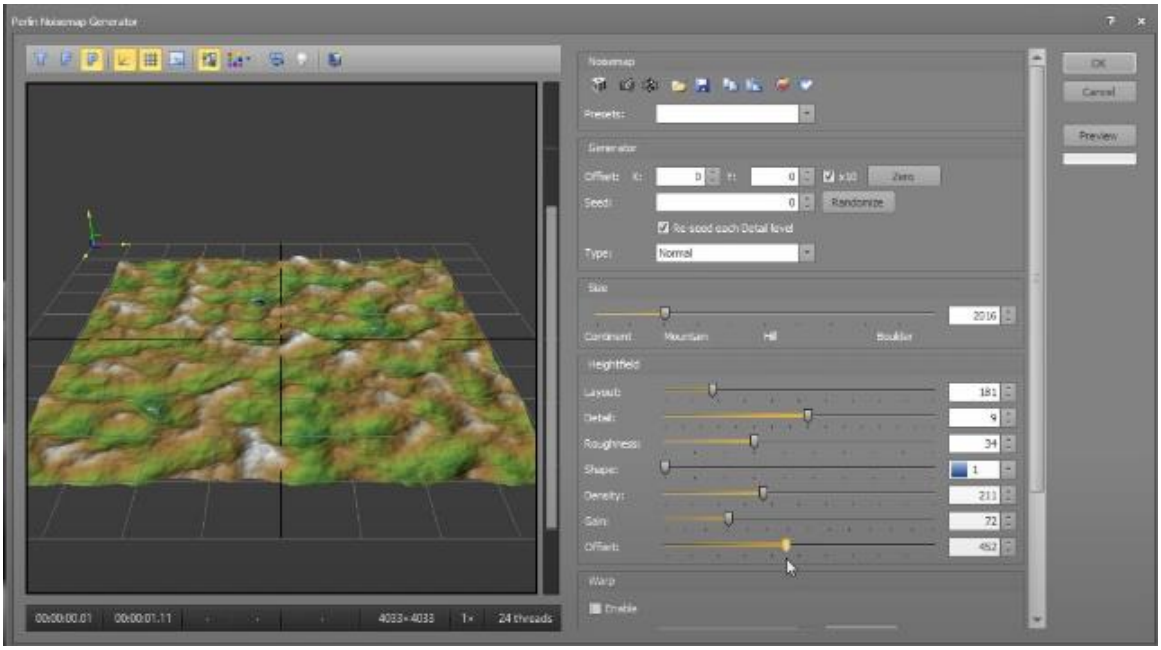

Рисунок 6. Создание поверхности в TerreSculptor с помощью Noise Map Generator

Инструменты для редактирования:

Sculpting Tools — для ручного изменения высот.

Erosion — для симуляции эрозии и создания более реалистичных форм.

Smoothing — для сглаживания поверхности.

После окончания работы над проектом, выполняется экспорт созданной поверхности в нужном формате для использования в Unreal Engine 5 (рисунок 7).

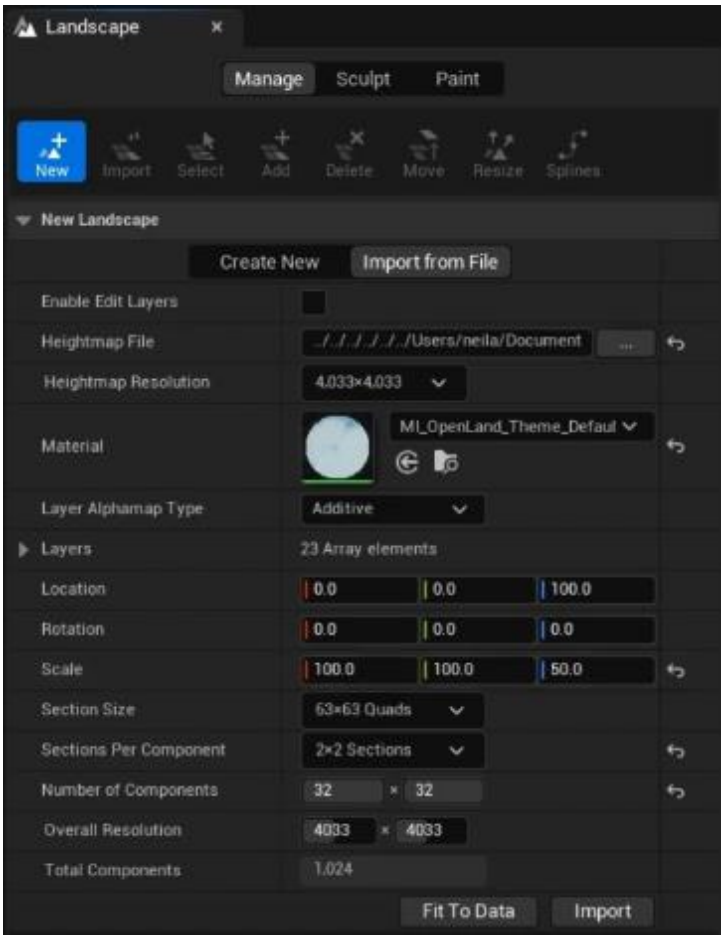

Рисунок 7. Импортирование материала в Unreal Engine

Особенности работы:

- Параметры генерации: проводились эксперименты с параметрами генерации, которые значительно влияли на конечный результат.
- Производительность: работа с высокими разрешениями потребовала значительных ресурсов. Пришлось искать компромисс между качеством и производительностью.

2.1.2 Доработка ландшафта в Unreal Engine и создание новых моделей и материалов

После импортирования базовой поверхности из TerreSculptor, необходимо доработать ландшафт, добавив несколько элементов. Одним из таких элементов является озеро (рисунок 8). Это – неотъемлемая часть арктического пейзажа. Инструменты TerreSculptor не позволяют его сделать, поэтому создание воды ведется непосредственно в UE5.

Рисунок 8. Озеро в Unreal Engine с материалом по умолчанию

Использовав актера класса Water body custom, можно создать блок воды в любом удобном месте поверхности. Озеро создается с материалом по умолчанию, поэтому цвет воды необходимо настроить таким образом, чтобы он соответствовал общей атмосфере сцены (рисунок 9). Для этого используется изменение параметров материала, что позволяет его настроить без каких-либо проблем.

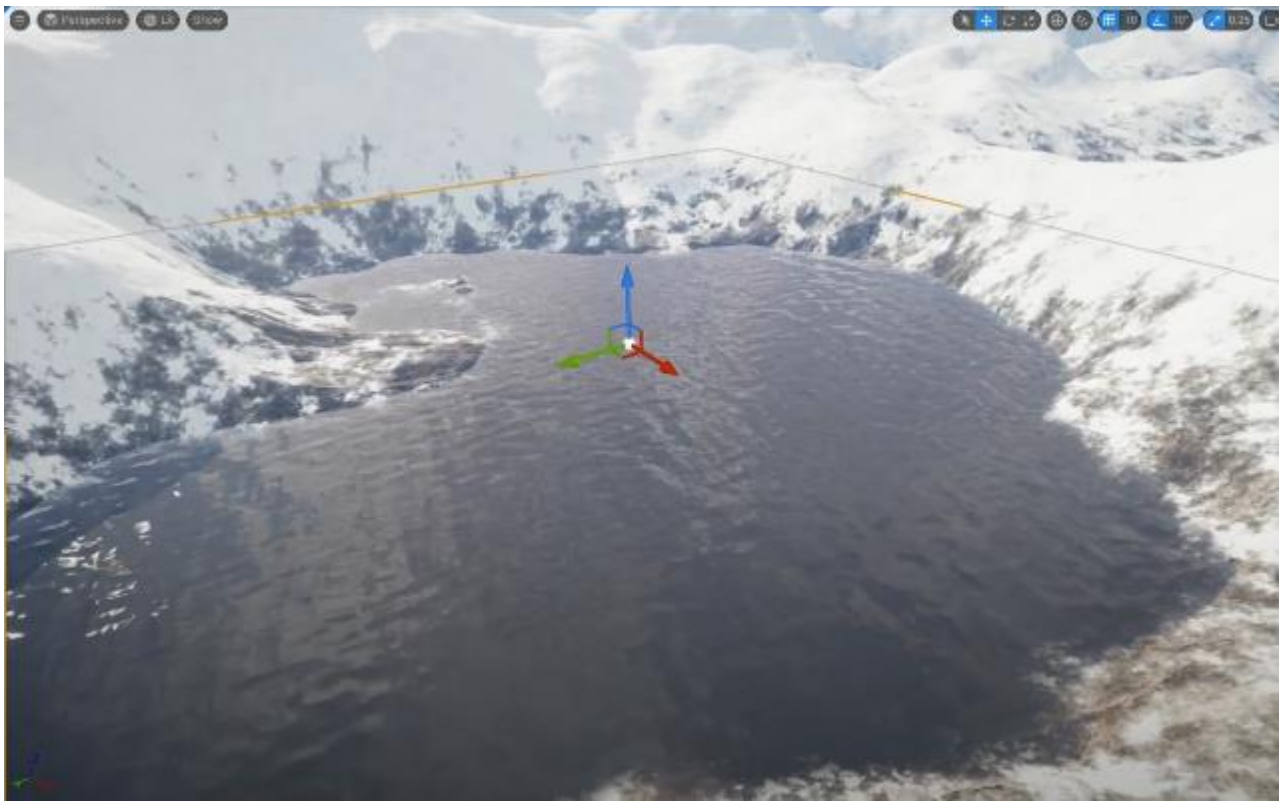

Рисунок 9. Озеро в Unreal Engine с настроенным материалом

После нанесения правильного материала и выбора его нужных параметров, озеро принимает более тусклый оттенок, а также становится менее прозрачным, что прекрасно вписывается в концепцию арктического пейзажа. Стоит заменить слишком резкие текстуры поверхности на более мягкие (рисунок 10). Для этого используется платформа Quixel Bridge – торговая площадка, которая позволяет купить или скачать нужные модели или текстуры. В сцене необходимо заменить два сета текстур: вершины гор заменяются на более мягкий белый цвет, а их подножия – на рыхлый, менее резкий.

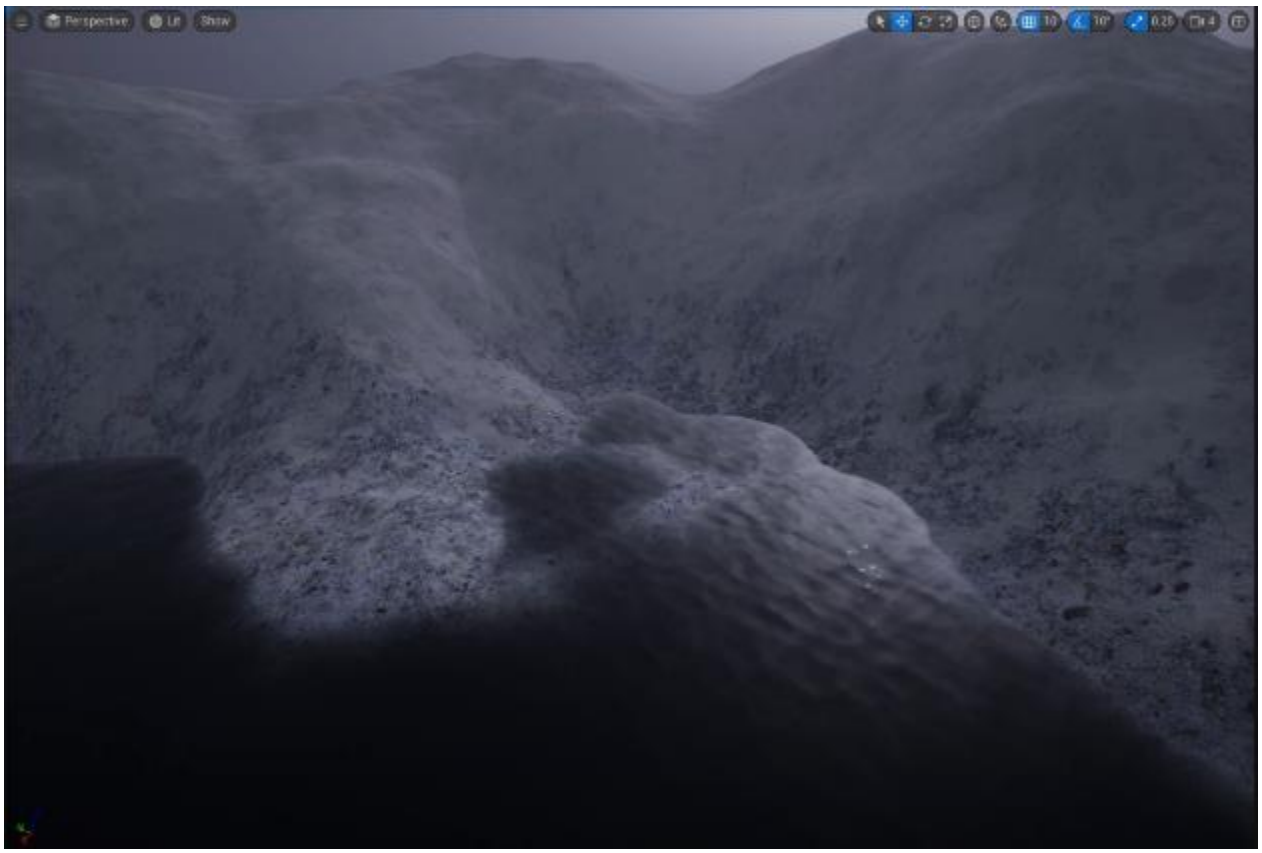

Рисунок 10. Замененные текстуры снега на вершинах гор и у их подножий

Ассеты для деревьев и камней можно также найти в каталоге Quixel Bridge. В параметрах моделей необходимо настроить несколько функций, главными из которых являются коллизия (collision) и прирост (growth). Чтобы не создавать каждое дерево вручную, уместно воспользоваться утилитой PFV (ProceduralFoliageVolume), которая позволяет создать большое количество моделей в выбранной области [3] (рисунок 11).

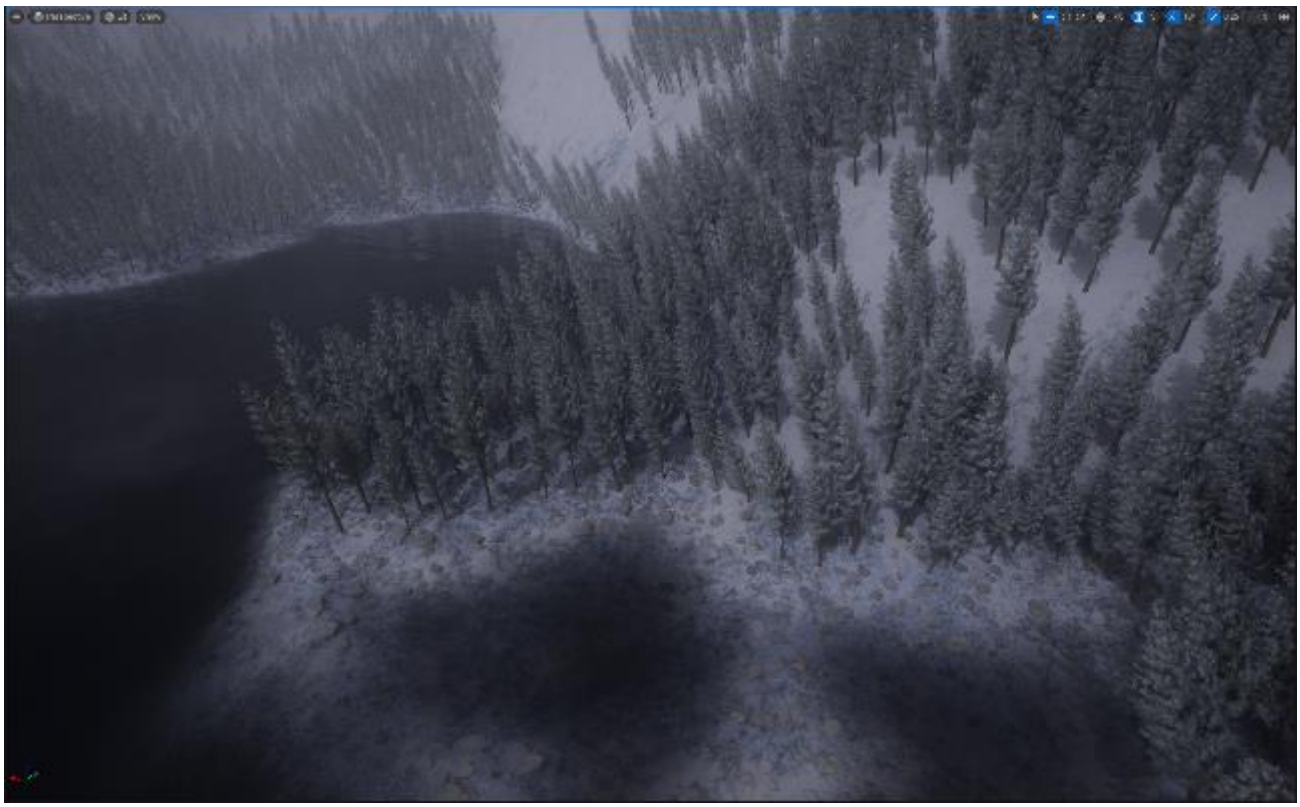

Рисунок 11. Деревья и камни, размещенные с помощью PFV

Моделям деревьев и камней также нужно присвоить заснеженные текстуры, чтобы они вписывались в общую арктическую атмосферу. В Unreal Engine система материалов организована с использованием концепции родительских и дочерних материалов, что позволяет эффективно управлять и переиспользовать материалы в проекте (рисунок 12). Здесь используется Geometry Instancing – техника, позволяющая отрисовывать большое количество объектов, для которой не требуется большое количество итераций [5]. Что касается деревьев – необходимо подняться на уровень родительского материала, изменив при этом летние текстуры на зимние.

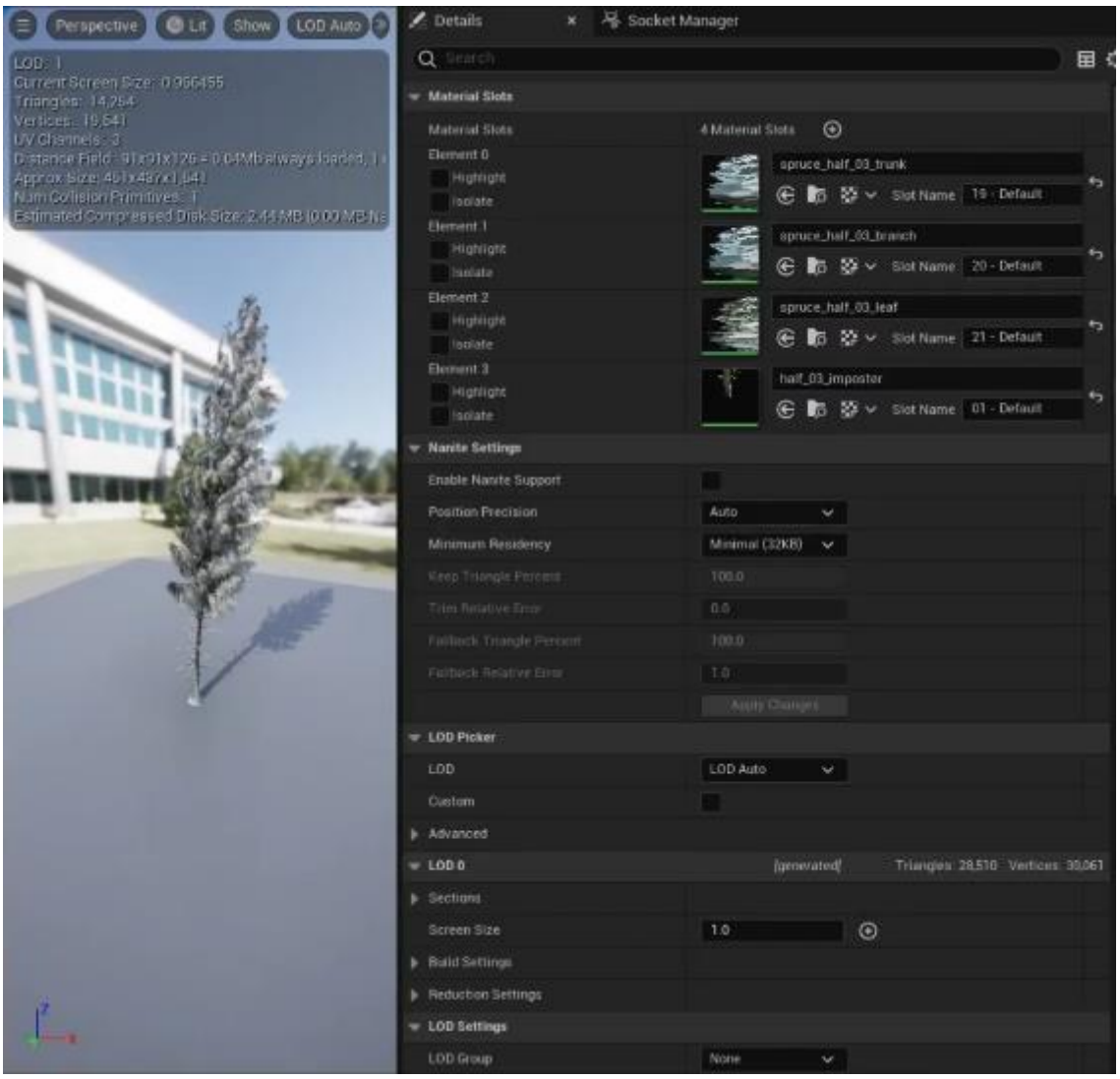

Рисунок 12. Окно деталей материала деревьев

Использование родительских и дочерних материалов — это мощный инструмент для художников и разработчиков, позволяющий создавать сложные и разнообразные визуальные эффекты в Unreal Engine. RVT (Runtime Virtual Textures) в Unreal Engine — это система, которая позволяет эффективно управлять текстурами и их отображением в реальном времени, улучшая производительность и качество графики в играх и приложениях (рисунок 13). Основная идея RVT заключается в том, чтобы объединить текстуры в виртуальные текстуры, которые могут динамически обновляться и использоваться для рендеринга [8]. Для изменения текстур камней как раз используется Runtime Virtual Textures, которая позволяет настроить внешний вид текстур с помощью графа.

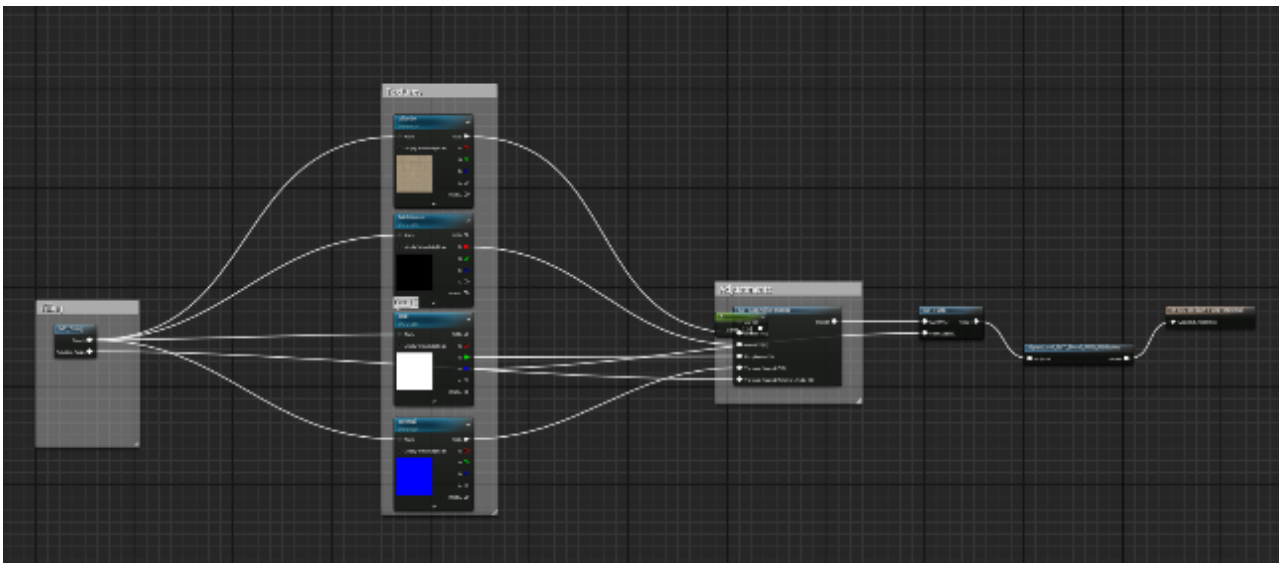

Рисунок 13. Граф RVT для изменения текстур камней

Для использования RVT в Unreal Engine необходимо создать RVT-активы, настроить их в материалах и применить к объектам в сцене. В итоге это позволило значительно улучшить визуальное качество и общую производительность сцены. RVT-граф представляет собой систему, предназначенную для управления рендерингом и визуализацией в реальном времени. Он позволяет разработчикам настраивать и оптимизировать процесс рендеринга, обеспечивая высокое качество изображения и производительность. RVT-граф использует узловую структуру, что делает его интуитивно понятным и доступным для пользователей, позволяя легко связывать различные элементы рендеринга, такие как освещение, материалы и эффекты. С помощью RVT-графа разработчики могут создавать сложные визуальные эффекты и настраивать параметры рендеринга в реальном времени, что позволяет быстро видеть результаты изменений. Это упрощает процесс создания контента и позволяет более эффективно управлять ресурсами.

2.1.3 Настройка отражений и тумана

В Unreal Engine существует несколько методов отражения, каждый из которых имеет свои особенности, преимущества и недостатки. Можно выделить три основных метода: Lumen, Screen Space Reflections (SSR) и Ray Tracing.

a) Lumen

Lumen — это система глобального освещения и отражений, представленная в Unreal Engine 5. Она предназначена для работы в реальном времени и обеспечивает высокое качество освещения и отражений.

Преимущества метода Lumen:

- **Динамическое освещение:** Lumen поддерживает динамическое освещение и может адаптироваться к изменениям в сцене, что делает его идеальным для игр с открытым миром.
- **Качество:** обеспечивает реалистичные отражения и освещение, включая сложные взаимодействия света с поверхностями.
- **Простота использования:** Lumen интегрирован в движок и не требует сложной настройки, что упрощает процесс разработки.

Недостатки метода Lumen:

- **Производительность:** может быть более требовательным к ресурсам по сравнению с другими методами, особенно на менее мощных системах.
- **Ограничения:** в некоторых случаях может не обеспечивать такое же качество отражений, как трассировка лучей.

b) Screen Space Reflections (SSR)

SSR — это метод, который использует информацию о пикселях, уже отрендеренных на экране, для создания отражений. Он работает только с теми объектами, которые видны в текущем кадре.

Преимущества метода SSR:

- **Производительность**: метод обычно менее требователен к ресурсам по сравнению с трассировкой лучей, что делает его подходящим для игр с высокими требованиями к производительности.

- **Простота**: легко интегрируется в существующие материалы и не требует сложной настройки.

Недостатки метода SSR:

- **Ограничения по видимости**: SSR не может отражать объекты, которые не находятся в поле зрения камеры, что может привести к артефактам и недостаткам в отражениях.
- **Качество**: Качество отражений может быть ниже, чем у других методов, особенно в сложных сценах.

c) Ray Tracing

Трассировка лучей — это метод, который использует физические модели света для создания реалистичных отражений, теней и освещения. Этот метод требует поддержки аппаратного обеспечения, такого как NVIDIA RTX (рисунок 14).

Преимущества метода трассировки лучей:

- **Высокое качество**: обеспечивает реалистичные отражения, включая отражения от прозрачных и полупрозрачных материалов, а также сложные эффекты, такие как глобальное освещение.
- **Точная симуляция**: позволяет точно моделировать поведение света, что делает его идеальным для фотореалистичных сцен.

Недостатки метода трассировки лучей:

- **Производительность**: трассировка лучей требует значительных вычислительных ресурсов и может заметно снизить производительность, особенно на старых или менее мощных системах.
- **Сложность настройки**: настройка трассировки лучей может быть более сложной и требовать дополнительных знаний.

Рисунок 14. Отражения в воде, созданные методом Ray Tracing

Для создания отражений в воде был использован метод Ray Tracing, так как он делает отражения максимально реалистичными и саму сцену более кинематографичной. В Unreal Engine существуют два основных типа тумана: плоский и объемный. Каждый из этих типов тумана имеет свои особенности, применения и визуальные эффекты.

а) Плоский туман (2D Fog)

Плоский туман представляет собой эффект, который накладывается на сцену и создает иллюзию тумана или дыма, но не взаимодействует с геометрией в 3D-пространстве. Он обычно используется для создания атмосферы и улучшения визуального восприятия (рисунок 15).

Преимущества плоского тумана:

- **Производительность**: плоский туман менее требователен к ресурсам, так как он не требует сложных расчетов для взаимодействия с геометрией.
- **Простота настройки**: легко настраивается и может быть быстро добавлен в сцену для создания эффекта тумана.

Недостатки плоского тумана:

- **Ограниченная реалистичность**: плоский туман не взаимодействует с объектами в сцене, что может привести к менее реалистичному восприятию, особенно в сложных сценах.
- **Отсутствие глубины:** плоский туман не создает эффекта глубины, который можно было бы ожидать от настоящего тумана.

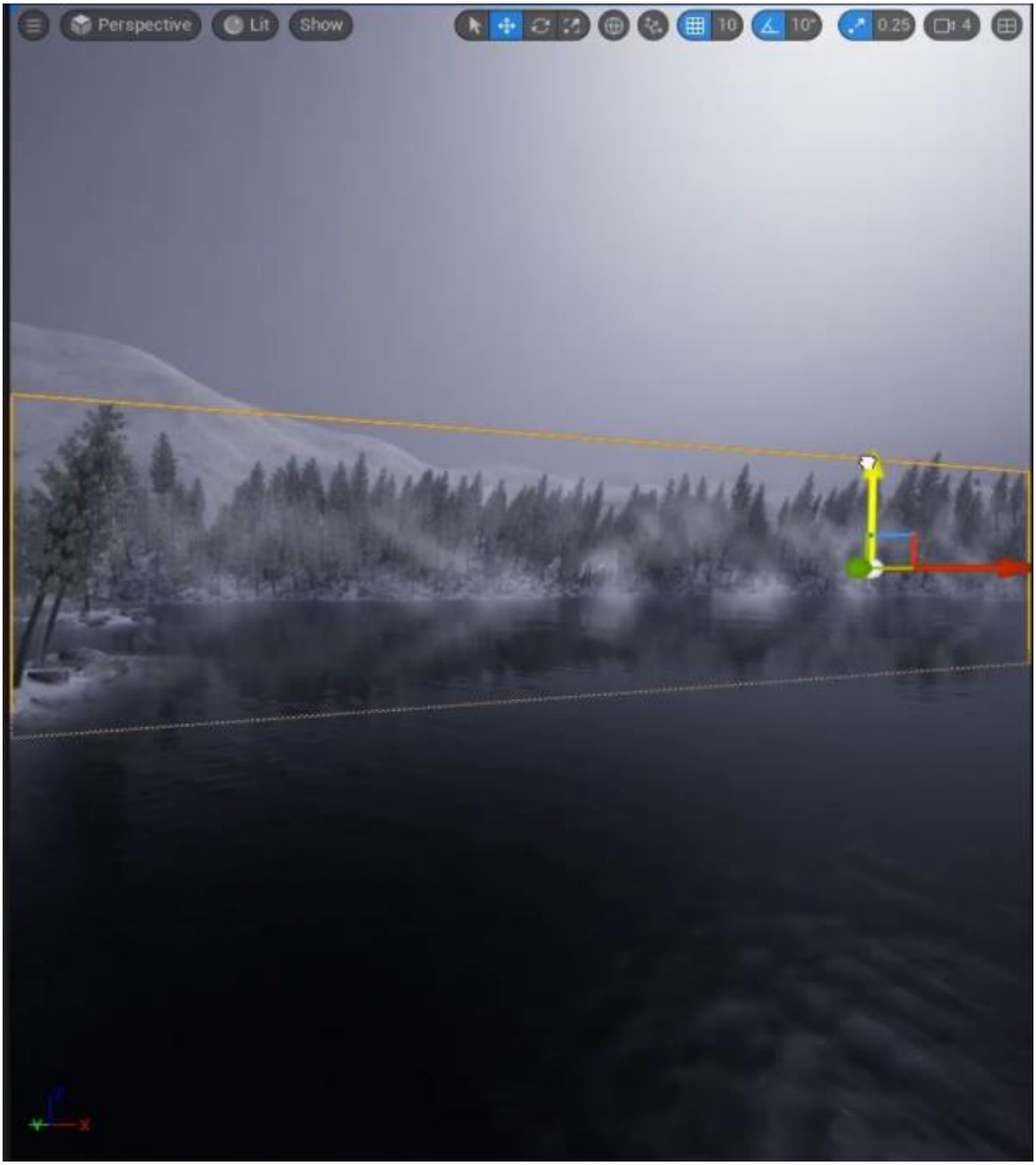

Рисунок 15. Плоский туман в Unreal Engine

b) Объемный туман (3D Fog). Объемный туман создает эффект тумана, который взаимодействует с геометрией в 3D-пространстве. Он может заполнять пространство между объектами и создавать более реалистичное восприятие глубины и атмосферы (рисунок 16).

Преимущества объемного тумана:

- **Реалистичность**: объемный туман создает более правдоподобные эффекты, так как он взаимодействует с освещением и геометрией, создавая ощущение глубины.
- **Динамика**: объемный туман может изменяться в зависимости от положения камеры и объектов в сцене, что добавляет динамичности.

Недостатки объемного тумана:

- **Производительность**: объемный туман более требователен к ресурсам, так как требует сложных расчетов для рендеринга.
- **Сложность настройки**: настройка объемного тумана может быть более сложной и требовать больше времени для достижения желаемого эффекта.

Рисунок 16. Объемный туман в Unreal Engine

2.1.4 Создание палаточного лагеря и места крушения

В сцене необходимо разбавить естественную природу несколькими искусственными объектами. Для создания палаточного лагеря использовались материалы из Quixel Bridge, такие как палета, фургон, бочка, коробка и брезент [6]. Большинство моделей огня сильно нагружают сцену вследствие огромного количества частиц самого пламени и дыма (рисунок 17). Чтобы число кадров не сильно падало, был использован оптимизированный огонь из UE Starter Kit.

UE Starter Kit — это набор инструментов и ресурсов, предназначенный для упрощения процесса начала работы с Unreal Engine 5. Он включает в себя различные шаблоны, примеры проектов и обучающие материалы, которые помогают новичкам быстро освоить основные функции и возможности движка. Starter Kit предоставляет готовые элементы, такие как модели, текстуры и анимации, что позволяет разработчикам сосредоточиться на создании контента, а не на поиске и создании базовых ресурсов. Кроме того, UE Starter Kit включает в себя документацию и руководства, которые объясняют, как использовать различные функции Unreal Engine, что делает его идеальным для тех, кто только начинает свой путь в разработке игр. Набор инструментов помогает ускорить процесс обучения и позволяет быстрее перейти к созданию собственных проектов.

Рисунок 17. Сцена палаточного лагеря

Quixel Bridge — это инструмент, интегрированный в Unreal Engine, который предоставляет доступ к обширной библиотеке 3D-ресурсов, текстур и материалов от Quixel Megascans. Он позволяет разработчикам легко находить, загружать и импортировать высококачественные активы прямо в свои проекты. Quixel Bridge упрощает процесс работы с ресурсами, предлагая интуитивно понятный интерфейс для поиска и управления контентом.

С помощью Quixel Bridge пользователи могут быстро добавлять реалистичные модели, текстуры и материалы, что значительно ускоряет процесс разработки и улучшает визуальное качество проектов. Инструмент также поддерживает автоматическую настройку материалов для Unreal Engine, что позволяет избежать дополнительных шагов при интеграции активов.

Для создания места крушения использовалось все из вышеописанного: Quixel Bridge для модели самолета, ProceduralFoliageVolume для камней под самолетом, огонь из UE Starter Kit (рисунок 18). Новое в этой сцене – снежная буря. Она была создана с помощью плагина Niagara, а также отредактирована в разделе графа материала [4].

Niagara — это система визуальных эффектов в Unreal Engine 5, предназначенная для создания сложных и реалистичных эффектов частиц. Она предоставляет разработчикам мощные инструменты для создания динамичных визуальных эффектов, таких как огонь, дым, взрывы и другие природные явления. Niagara использует узловую систему, что позволяет легко настраивать и комбинировать различные элементы, создавая уникальные эффекты без необходимости в написании кода. Система поддерживает работу с различными типами данных и позволяет интегрировать эффекты с другими системами Unreal Engine, такими как освещение и физика.

Рисунок 18. Сцена крушения самолета и снежной бури

2.2 Разработка программного кода для реализации полета самолета над сценой

Создание объектов в Unreal Engine 5 (UE5) с использованием программирования на C++ открывает широкие возможности для разработчиков игр и симуляций. UE5 предоставляет мощные инструменты для визуализации и взаимодействия с 3D-объектами, что позволяет создавать уникальные игровые миры и механики. Одной из ключевых задач в разработке является создание динамичных объектов, которые могут взаимодействовать с окружающей средой и игроками. Это включает в себя не только статические элементы, но и движущиеся объекты, такие как транспортные средства, персонажи и, в данном случае, самолеты.

В последние годы наблюдается рост интереса к созданию реалистичных симуляций полета, что находит применение в различных областях, от игр до образовательных программ и военных симуляторов. Например, в играх, таких как Microsoft Flight Simulator, реализованы сложные модели полета, которые учитывают физику, аэродинамику и управление. Однако создание таких объектов требует глубокого понимания как физики, так и программирования.

Существуют различные подходы к созданию летательных аппаратов в игровых движках. Например, в Unreal Engine можно использовать как визуальные скрипты (Blueprints), так и программирование на C++, что позволяет разработчикам выбирать наиболее удобный для них способ реализации. В рамках данной работы рассматривается создание простого самолета, который будет пролетать заданное расстояние в небе, используя код на C++. Это позволит не только продемонстрировать основные принципы работы с объектами в UE5, но и углубить понимание взаимодействия между программным кодом и игровым движком.

Актуальность проекта обусловлена потребностью в создании динамичных и интерактивных объектов в Unreal Engine 5, что является важным аспектом

разработки современных игр и симуляций. Цель данной работы – разработать класс самолета, который будет перемещаться по заданной траектории в небе, продемонстрировав основные принципы программирования и работы с 3D-объектами в UE5. Для достижения этой цели необходимо реализовать логику движения самолета, а также протестировать его поведение в игровом мире.

2.2.1 Постановка задач

Целью данной работы является создание класса самолета в Unreal Engine 5, который будет перемещаться по заданной траектории в небе. Это позволит продемонстрировать основные принципы программирования на C++ и взаимодействия с игровым движком, а также создать динамичный объект, который может быть использован в различных игровых сценариях [7]. Для успешной реализации проекта были достигнуты следующие задачи:

1. Разработан класс самолета: создать класс ArcticFlyingPlane, который наследуется от базового класса AActor и реализована логика движения самолета.
2. Определены начальные и конечные позиции: установлена начальная и конечная точки полета, а также задана продолжительность полета, чтобы обеспечить плавное движение самолета.
3. Реализована логика движения: использована интерполяцию для плавного перемещения самолета от начальной позиции к конечной, учитывая время полета и обновление позиции в каждом кадре.
4. Обеспечено удаление объекта: реализован механизм удаления самолета из игрового мира по завершении полета, чтобы избежать накопления неактивных объектов в сцене.
5. Проведено тестирование: проверена работоспособность созданного класса в игровом мире, корректность движения и удаления самолета, а также оценено его взаимодействие с окружающей средой.

Таким образом, итогом данной работы стало создание функционального класса самолета. Этот класс демонстрирует основные принципы работы с

объектами в Unreal Engine 5. Он также может послужить основой для дальнейших разработок в области симуляции полета и создания динамичных игровых объектов [9].

2.2.2 Разработка программного кода самолета

В ходе выполнения практики был разработан класс самолета для Unreal Engine 5, который демонстрирует основные принципы программирования на C++ и взаимодействия с игровым движком. Тестирование проводилось в игровом мире, что позволило проверить корректность работы класса и его поведение в реальных условиях [10].

1) Класс ArcticFlyingPlane.h

В этом файле был создан заголовок класса AArcticFlyingPlane, который наследуется от базового класса AActor. Заголовочный файл содержит объявления переменных и методов, необходимых для реализации логики движения самолета. Важно отметить, что использование макроса UCLASS() позволяет Unreal Engine автоматически генерировать необходимый код для работы с классом в редакторе и во время выполнения. Также в этом файле определяются переменные, которые будут использоваться для хранения начальной и конечной позиций, продолжительности полета и времени, прошедшего с начала полета.

Листинг 1. Код файла ArcticFlyingPlane.h

```
#pragma once

#include "CoreMinimal.h"
#include "GameFramework/Actor.h"
#include "ArcticFlyingPlane.generated.h"

// Определение класса самолета, который будет наследоваться от AActor
UCLASS()
class ARCTIC_API AArcticFlyingPlane : public AActor
{
    GENERATED_BODY()

public:
```

Продолжение листинга 1. Код файла ArcticFlyingPlane.h

```
    // Конструктор класса
    AArcticFlyingPlane();

protected:
    // Метод, вызываемый при начале игры
    virtual void BeginPlay() override;

public:
    // Метод, вызываемый каждый кадр
    virtual void Tick(float DeltaTime) override;

private:
    // Начальная позиция самолета
    FVector StartLocation;
    // Конечная позиция самолета
    FVector EndLocation;
    // Продолжительность полета в секундах
    float FlightDuration;
    // Время, прошедшее с начала полета
    float ElapsedTime;
};
```

2) Класс ArcticFlyingPlane.cpp

В этом файле была реализована логика движения самолета. Код отвечает за интерполяцию между начальной и конечной позициями, а также за удаление объекта по завершении полета. Конструктор класса инициализирует начальные значения переменных, устанавливает начальную позицию самолета и включает возможность обновления в каждом кадре. Метод Tick отвечает за обновление позиции самолета, используя линейную интерполяцию для плавного движения. Важно отметить, что использование функции FMath::Lerp позволяет добиться плавного перехода между двумя точками, что делает движение более реалистичным.

Листинг 2. Код файла ArcticFlyingPlane.cpp

```
#include "ArcticFlyingPlane.h"

// Конструктор класса
AArcticFlyingPlane::AArcticFlyingPlane()
{
    // Включение возможности обновления в каждом кадре
    PrimaryActorTick.bCanEverTick = true;

    // Установка начальной и конечной позиций
```

Продолжение листинга 2. Код файла ArcticFlyingPlane.cpp

```
    StartLocation = FVector(0.0f, 0.0f, 1000.0f); // Начальная позиция в небе
    EndLocation = FVector(5000.0f, 0.0f, 1000.0f); // Конечная позиция
    FlightDuration = 5.0f; // Время полета в секундах
    ElapsedTime = 0.0f; // Инициализация времени

    // Установка начальной позиции самолета в игровом мире
    SetActorLocation(StartLocation);
}

// Метод, вызываемый при начале игры
void AArcticFlyingPlane::BeginPlay()
{
    Super::BeginPlay(); // Вызов метода базового класса
}

// Метод, вызываемый каждый кадр
void AArcticFlyingPlane::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime); // Вызов метода базового класса

    // Увеличение времени, прошедшего с начала полета
    ElapsedTime += DeltaTime;
    // Вычисление коэффициента интерполяции
    float Alpha = FMath::Clamp(ElapsedTime / FlightDuration, 0.0f, 1.0f);
    // Вычисление новой позиции самолета
    FVector NewLocation = FMath::Lerp(StartLocation, EndLocation, Alpha);
    // Установка новой позиции самолета в игровом мире
    SetActorLocation(NewLocation);

    // Удаление самолета из игрового мира по завершении полета
    if (Alpha >= 1.0f)
    {
        Destroy(); // Удаление объекта
    }
}
```

3) Определение макроса в файле Arctic.h

Нижеприведенный макрос ARCTIC_API используется для управления экспортом и импортом функций и классов в динамических библиотеках (DLL) в Unreal Engine. Если определённый символ ARCTIC_API уже задан (например, при компиляции библиотеки), макрос будет определён как __declspec(dllexport), что позволяет экспортировать функции и классы из библиотеки. В противном случае, если символ не задан, макрос будет определён как __declspec(dllimport), что позволяет импортировать функции и классы из библиотеки в другие модули или приложения.

Листинг 3. Код файла Arctic.h

```
#pragma once

#ifdef ARCTIC_API
#define ARCTIC_API __declspec(dllexport)
#else
#define ARCTIC_API __declspec(dllimport)
#endif
```

4) Результаты тестирования

Тестирование класса AArcticFlyingPlane проводилось в игровом мире Unreal Engine 5. Основная задача заключалась в проверке корректности движения самолета от начальной до конечной позиции за заданное время.

Фактический результат:

- Самолет плавно перемещается от начальной позиции (0, 0, 1000) до конечной позиции (5000, 0, 1000) за 5 секунд.
- Объект корректно удаляется из игрового мира по завершении полета.

Таким образом, разработанный класс успешно выполняет поставленные задачи, демонстрируя основные принципы работы с объектами в Unreal Engine. Результаты тестирования подтвердили его работоспособность и готовность к дальнейшему использованию в более сложных игровых сценариях.

2.2.3 Анализ результатов

Результаты, полученные в ходе разработки класса самолета для Unreal Engine 5, продемонстрировали высокую эффективность созданного решения в контексте программирования и взаимодействия с игровым движком. Класс AArcticFlyingPlane успешно реализует логику движения, обеспечивая плавное перемещение объекта от начальной до конечной позиции в заданный промежуток времени. Все тесты подтвердили корректность работы класса, что позволяет использовать его в различных игровых сценариях. Класс AArcticFlyingPlane продемонстрировал свою способность к динамическому перемещению в игровом мире, что открывает возможности для создания более сложных механик, таких как взаимодействие с другими объектами, анимация и

управление полетом. Плавное движение самолета было достигнуто благодаря использованию линейной интерполяции, что делает его поведение более реалистичным и приятным для восприятия игроками. Это также позволяет разработчикам легко настраивать параметры полета, такие как скорость и траектория, что значительно расширяет возможности для дальнейших разработок. Ключевым преимуществом реализации данного класса является возможность его дальнейшего расширения. Также можно интегрировать физические эффекты, такие как ветер и аэродинамика, что сделает симуляцию полета еще более реалистичной.

Перспективы данного решения включают:

- Добавление анимации: реализация анимаций для самолета, таких как взлет и посадка, что повысит уровень погружения в игровой процесс.
- Интеграция с системами управления: внедрение системы управления полетом, позволяющей игрокам управлять самолетом с помощью клавиатуры или контроллера.
- Расширение функционала: возможность добавления различных типов самолетов с уникальными характеристиками и поведением в воздухе.
- Создание системы взаимодействия: разработка механик взаимодействия самолета с другими объектами, такими как здания, другие самолеты или природные элементы (например, облака и ветер).
- Оптимизация производительности: улучшение производительности класса для работы с большим количеством объектов в сцене, что особенно важно для многопользовательских игр или сложных симуляций.

Таким образом, полученные результаты являются надежной основой для дальнейшего развития проекта в области создания динамичных объектов в Unreal Engine 5. Они демонстрируют потенциал для интеграции в более сложные игровые механики и открывают новые горизонты для научно-исследовательской деятельности в области разработки игр и симуляций. Результаты позволяют использовать разработанный класс в более сложных игровых сценариях.

ЗАКЛЮЧЕНИЕ

Создание интерактивной среды в Unreal Engine 5 — это многогранный процесс, который требует сочетания художественных и технических навыков. На каждом этапе, от проектирования до программирования, разработчики сталкиваются с уникальными вызовами, которые требуют креативного подхода и глубокого понимания инструментов движка. Unreal Engine 5 предоставляет разработчикам мощные инструменты, такие как Nanite и Lumen, которые позволяют создавать высококачественные и реалистичные миры. Использование Blueprints упрощает процесс программирования, позволяя сосредоточиться на создании увлекательного игрового опыта.

В результате практики был разработан класс самолета для Unreal Engine 5, который демонстрирует устойчивую и функциональную логику движения в игровом мире. Класс AArcticFlyingPlane прошел тестирование и подтвердил свою надежность при взаимодействии с игровым движком, обеспечивая плавное перемещение от начальной до конечной позиции. Собранные данные о поведении самолета в игровом мире обладают высоким качеством и могут быть использованы для дальнейшего анализа и разработки более сложных игровых механик. Работа позволила не только продемонстрировать основные принципы программирования на C++, но и заложить основу для будущих проектов в области создания динамичных объектов и симуляций в Unreal Engine.

Разработанное решение обладает потенциалом для масштабирования и регулярного использования в различных игровых сценариях. В дальнейшем возможно расширение функционала класса, добавление новых механик и интеграция с другими системами, что откроет новые горизонты для разработки игр и симуляций. Таким образом, результаты работы могут стать основой для дальнейших исследований и разработок в области игрового дизайна и программирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Unreal Engine 5 Documentation // Unreal Engine Documentation [Электронный ресурс] — URL: https://dev.epicgames.com/community/learning/tutorials/DYE1/unreal-engine-5-1-unreal-engine-5 (дата обращения: 15.02.2025).
2. Introduction to Materials // Unreal Engine Documentation [Электронный ресурс] — URL: https://dev.epicgames.com/community/learning/tutorials/9d0a/unreal-engine-introduction-to-materials (дата обращения: 24.02.2025).
3. Procedural Foliage Tool // Unreal Engine Documentation [Электронный ресурс] — URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/procedural-foliage-tool-in-unreal-engine (дата обращения: 27.02.2025).
4. Visual Effects in Niagara for Unreal Engine 5 // Unreal Engine Documentation [Электронный ресурс] — URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/creating-visual-effects-in-niagara-for-unreal-engine (дата обращения: 22.03.2025).
5. Geometry instancing // Wikipedia, the free encyclopedia [Электронный ресурс] — URL: https://en.wikipedia.org/wiki/Geometry_instancing (дата обращения: 20.09.2024).
6. Quixel Bridge // Quixel Bridge market [Электронный ресурс] — URL: https://quixel.com/bridge (дата обращения: 02.04.2025).
7. Procedural generation // Wikipedia, the free encyclopedia [Электронный ресурс] — URL: https://en.wikipedia.org/wiki/Procedural_generation (дата обращения: 10.04.2025).
8. Device Reference // World Machine Help [Электронный ресурс] – URL: https://help.world-machine.com/topics/reference/ (дата обращения: 21.04.2025).
9. Unreal Engine YouTube channel / [Электронный ресурс] // : [сайт]. — URL: https://www.youtube.com/user/UnrealDevelopmentKit (дата обращения: 25.04.2025).

10. Unreal Engine Форум / [Электронный ресурс] // : [сайт]. — URL: https://forums.unrealengine.com/categories?tag=unreal-engine (дата обращения: 05.05.2025).

11. К. С. Бенжамин, А. М. Костин

Unreal Engine 5: Полное руководство по разработке игр. – М.: Издательство "БХВ-Петербург", 2021. – 480 с.: ил. - ISBN 978-5-9775-0665-4 (дата обращения: 11.05.2025).

12. М. Р. Смит, Д. А. Джонсон

Разработка игр на Unreal Engine 5: от идеи до реализации. – М.: Издательство "Вильямс", 2022. – 512 с.: ил. - ISBN 978-5-8459-2020-3 (дата обращения: 18.05.2025).

13. Т. Л. Хантер, С. П. Грей

Unreal Engine 5 для начинающих: пошаговое руководство. – М.: Издательство "Питер", 2022. – 320 с.: ил. - ISBN 978-5-4461-1234-5 (дата обращения: 26.05.2025).

14. Д. К. Мартин, Э. Р. Фостер

Создание 3D-игр с Unreal Engine 5: практическое руководство. – М.: Издательство "Диалектика", 2023. – 400 с.: ил. - ISBN 978-5-9901234-5-6 (дата обращения: 30.05.2025).

15. А. В. Сидоров, Н. И. Петров

Unreal Engine 5: Современные подходы к разработке игр. – М.: Издательство "Книга", 2023. – 550 с.: ил. - ISBN 978-5-98765-432-1 (дата обращения: 02.06.2025).

ПРИЛОЖЕНИЕ А

Графическая часть выпускной квалификационной работы

В графическую часть выпускной квалификационной работы входят 8 чертежей.

1. Создание поверхности в TerreSculptor и ее импортирование в Unreal Engine;
2. Создание озера на поверхности и настройка его материала;
3. Поверхность воды без отражений и с отражениями Ray Tracing;
4. Визуальный вид плоского и объемного туманов;
5. Вид текстур поверхности после их замены, размещение деревьев и камней;
6. Редактирование текстур деревьев и граф RTV-текстур камней;
7. Графы материала плоского и объемного туманов;
8. Сцена палаточного лагеря и места крушения самолета.