|  |  |
| --- | --- |
|  | **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 Unreal Engine 5» | |
| При выполнении ВКР:   |  |  |  | | --- | --- | --- | |  | Используются / Не используются | Да/Нет | |  | Литературные источники и документы, имеющие гриф секретности | Нет | |  | Литературные источники и документы, имеющие пометку «Для служебного пользования», иных пометок, запрещающих открытое опубликование | Нет | |  | Служебные материалы других организаций | Нет | |  | Результаты НИР (ОКР), выполняемой в МГТУ им. Н.Э. Баумана | Нет | |  | 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: |  | | |
| **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 55 листах формата A4. | |
| Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.): | |
| |  | | --- | | *Работа содержит 8 графических листов формата A4.* | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | |
| 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. |
|  | (Подпись, дата) | (Фамилия И.О.) |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Козлов В.В. |
|  | (Подпись, дата) | (Фамилия И.О.) |

|  |
| --- |
| 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 |  | УТВЕРЖДАЮ | | |
| КАФЕДРА | РК6 | Заведующий кафедрой | | | РК6 |
| (Индекс) |
| ГРУППА | РК6-83Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | 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 Unreal Engine 5» | | | | | | | | |
| **№ п/п** | **Наименование этапов выпускной квалификационной работы** | | **Сроки выполнения этапов** | | **Отметка о выполнении** | |
| **план** | **факт** | **Должность** | **ФИО, подпись** |
|  | Задание на выполнение работы. Формулирование проблемы, цели и задач работы | | 11.02.2025 |  | Руководитель ВКР | Витюков Ф.А. |
|  | 1 часть. Аналитическая часть | | 25.02.2025 |  | Руководитель ВКР | Витюков Ф.А. |
|  | Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач | | 02.03.2025 |  | Заведующий кафедрой | Карпенко А.П. |
|  | 2 часть. Практическая часть 1 | | 25.03.2025 |  | Руководитель ВКР | Витюков Ф.А. |
|  | 3 часть. Практическая часть 2 | | 16.05.2025 |  | Руководитель ВКР | Витюков Ф.А. |
|  | 1-я редакция работы | | 22.05.2025 |  | Руководитель ВКР | Витюков Ф.А. |
|  | Подготовка доклада и презентации | | 08.06.2025 |  | Руководитель ВКР | Витюков Ф.А. |
|  | Отзыв руководителя | |  |  | Руководитель ВКР | Витюков Ф.А. |
|  | Нормоконтроль | |  |  | Нормоконтролер | Грошев С.В. |
|  | Внешняя рецензия | |  |  | Секретарь ГЭК | Кузьмина И.А. |
|  | Защита работы на ГЭК | | 17.06.2025 |  | Секретарь ГЭК | Кузьмина И.А. |

|  |  |  |
| --- | --- | --- |
| Руководитель квалификационной работы | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Витюков Ф.А. |
|  | (Подпись, дата) | (Фамилия И.О.) |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Козлов В.В. |
|  | (Подпись, дата) | (Фамилия И.О.) |

**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 Unreal Engine 5».

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

**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.

**Unreal Engine 5 (UE5)** – 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.

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

**Runtime Virtual Textures (RVT)** – 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.

**Central Processing Unit (CPU)** – центральный процессор.

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

**Frames per second (FPS) –** 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 11](#_Toc200365703)

[1. Основные этапы и методы создания интерактивной среды 16](#_Toc200365704)

[1.1 Начальные этапы: планирование и импортирование 17](#_Toc200365705)

[1.2 Атмосфера и интерактивность 20](#_Toc200365706)

[1.3 Тестирование и оптимизация 21](#_Toc200365707)

[2. Разработка ландшафта сцены 23](#_Toc200365708)

[2.1 Визуализация ландшафта сцены 24](#_Toc200365709)

[2.1.1 Создание ландшафта в TerreSculptor и его импорт в Unreal Engine 25](#_Toc200365710)

[2.1.2 Доработка ландшафта в Unreal Engine и создание новых моделей и материалов 27](#_Toc200365711)

[2.1.3 Настройка отражений и тумана 33](#_Toc200365712)

[2.1.4 Создание палаточного лагеря и места крушения 39](#_Toc200365713)

[2.2 Разработка программного кода для реализации полета самолета над сценой 41](#_Toc200365714)

[2.2.1 Постановка задач 43](#_Toc200365715)

[2.2.2 Отчет о выполненных работах 44](#_Toc200365716)

[2.2.3 Анализ результатов 50](#_Toc200365717)

[ЗАКЛЮЧЕНИЕ 52](#_Toc200365718)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 54](#_Toc200365719)

ВВЕДЕНИЕ

Unreal Engine 5 (UE5) — это один из самых мощных и популярных движков для разработки игр и интерактивных приложений, предлагающий инновационные технологии и инструменты для создания высококачественных трехмерных миров. Он 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.

Здесь 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 которыми игрок может взаимодействовать, и механики, которые будут определять поведение среды. 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. **Начальные этапы: планирование и импортирование**

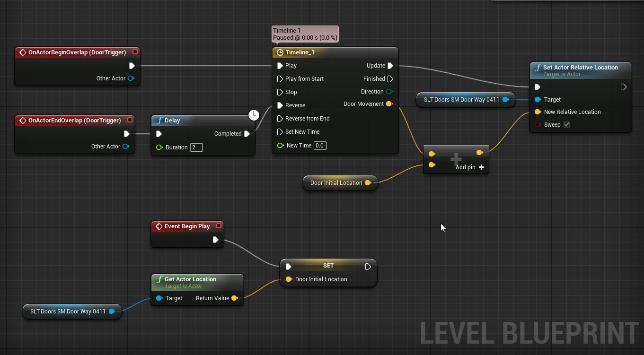


Рисунок 1 – Граф для создания сцены в Unreal Engine

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.

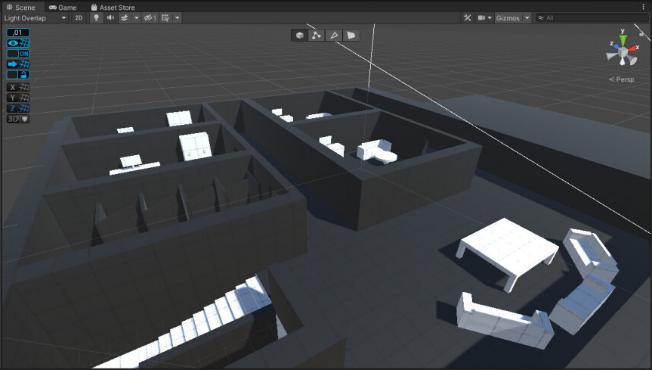


Рис. 2 «Пример визуального уровня в Unreal Engine»

UE5 предоставляет мощные инструменты для создания уровней:

Редактор уровней: позволяет размещать объекты, настраивать ландшафт и создавать архитектурные элементы;

World Partition: упрощает работу с большими мирами, автоматически загружая и выгружая части уровня по мере необходимости.

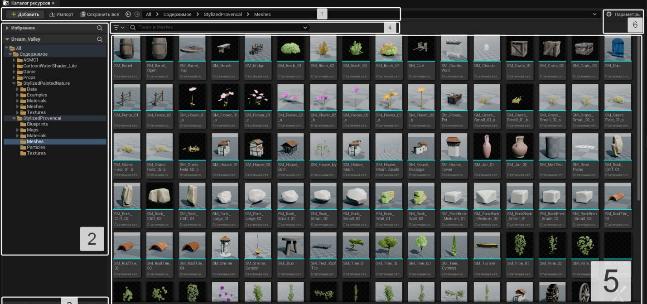


Рис. 3 «Каталог ресурсов, встроенный в движок»

В UE5 имеется возможность импортировать 3D-модели, текстуры и анимации из других программ (например, Blender, Maya или 3ds Max) или создавать их непосредственно в движке. Для этого можно использовать:

Nanite: для работы с высокодетализированными моделями без необходимости оптимизации;

Систему материалов: для создания реалистичных поверхностей с помощью нодового редактора.

* 1. **Атмосфера и интерактивность**



Рис. 4 «Пример сцены в Unreal Engine с освещением Lumen»

Для создания динамического освещения и реалистичных отражений используется система Lumen. В ней можно настроить атмосферные эффекты, такие как туман, облака и погодные условия, чтобы добавить глубину и реализм.

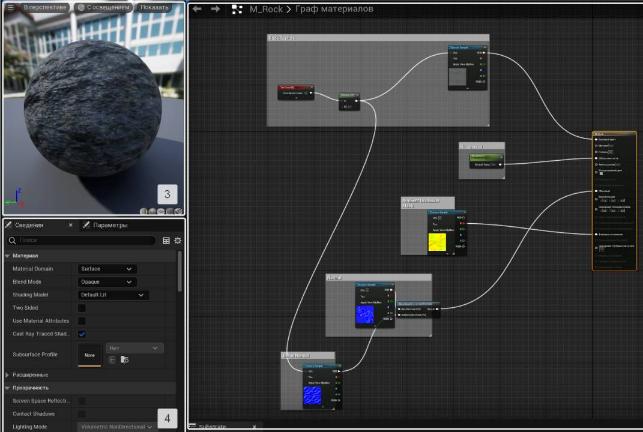


Рис. 5 «Настройка материала с помощью графа материалов»

Интерактивные элементы добавляются с помощью системы Blueprints или C++. Это может включать в себя:

Взаимодействие с объектами (открытие дверей, сбор предметов и т.д.);

Системы триггеров (активация событий или изменений в среде) [2].

* 1. **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 настройку уровней детализации (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 опредеить, 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. 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.

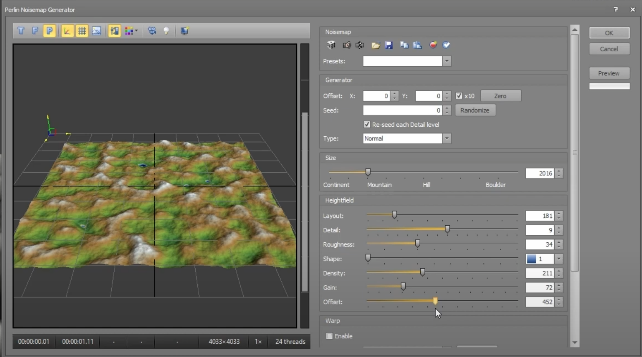


Рисунок 1 – Создание поверхности в TerreSculptor с помощью Noise Map Generator

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 2 – 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.

Озеро – неотъемлемая часть арктического пейзажа. Инструменты TerreSculptor не позволяют его сделать, поэтому создание воды ведется непосредственно в UE5.



Рисунок 3 – 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. Для этого используется изменение параметров материала, что позволяет его настроить без каких-либо проблем.

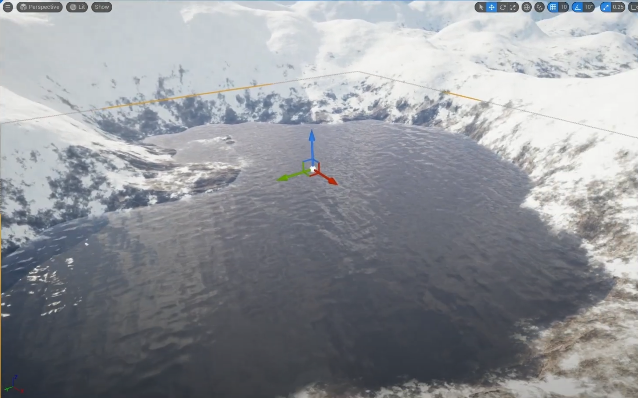


Рисунок 4 – Озеро в Unreal Engine с настроенным материалом

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. 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.

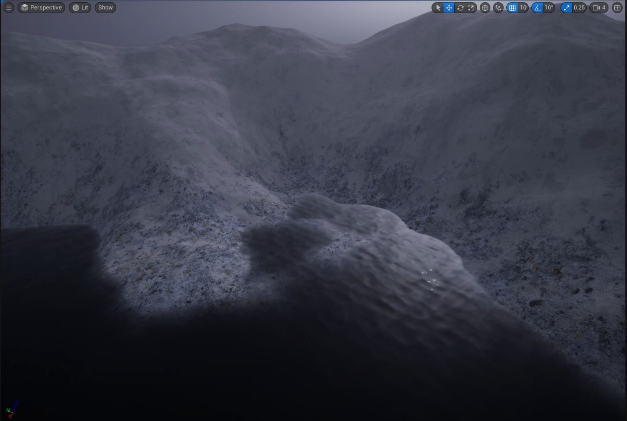


Рисунок 5 – Замененные текстуры снега на вершинах гор и у их подножий

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].

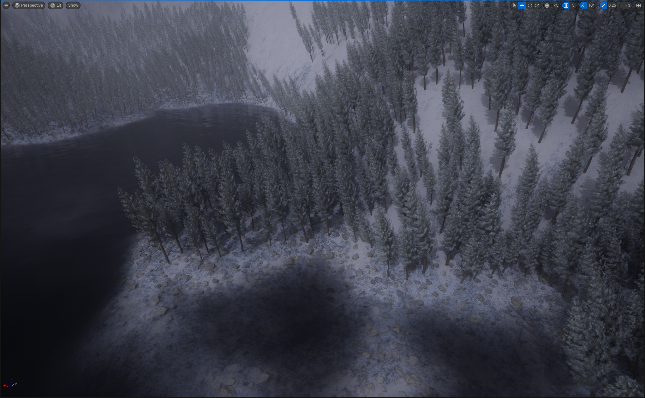


Рисунок 6 – Деревья и камни, размещенные с помощью PFV

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.

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.

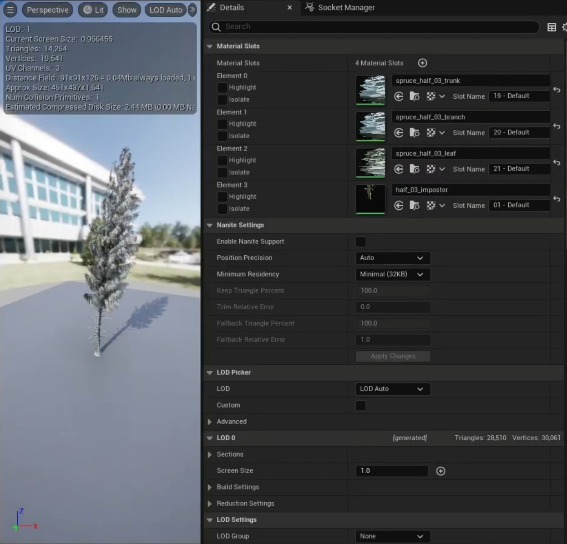


Рисунок 7 – Окно деталей материала деревьев

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. 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.

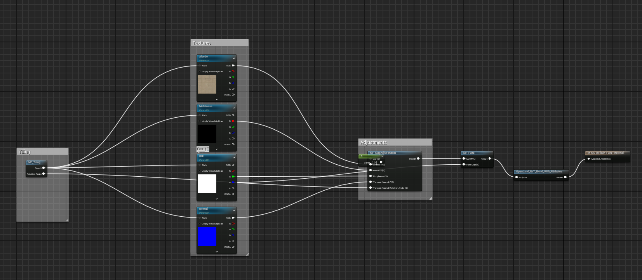


Рисунок 8 – Граф RVT для изменения текстур камней

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.

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.

**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 9 – 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.

a) 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.

**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.

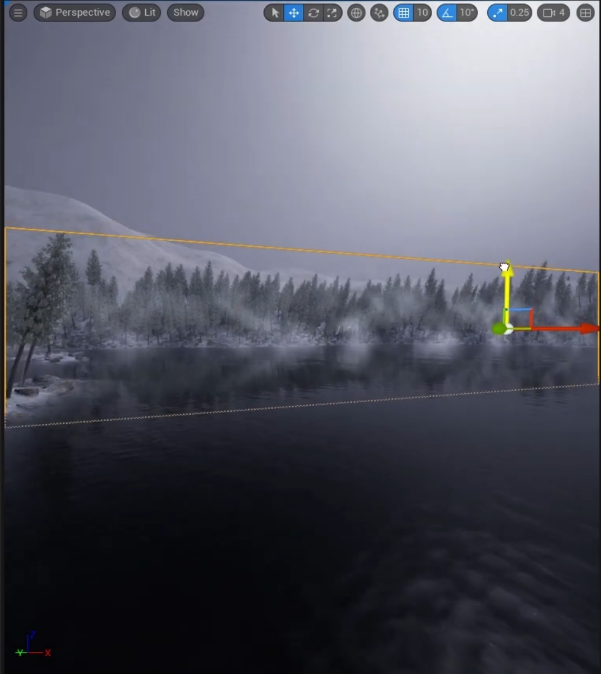


Рисунок 10 – Плоский туман в Unreal Engine

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.

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 11 – 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. Чтобы число кадров не сильно падало, был использован оптимизированный огонь из UE Starter Kit.



Рисунок 12 – Сцена палаточного лагеря

Для создания места крушения использовалось все из вышеописанного: Quixel Bridge для модели самолета, ProceduralFoliageVolumeдля камней под самолетом, огонь из UE Starter Kit.

Новое в этой сцене – снежная буря. Она была создана с помощью плагина Niagara, а также отредактирована в разделе графа материала [4].



Рисунок 13 – 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 сформулированы следующие задачи:

1. Разработать класс самолета: создать класс ArcticFlyingPlane, который будет наследоваться от базового класса AActor и реализовывать логику движения самолета.
2. Определить начальные и конечные позиции: установить начальную и конечную точки полета, а также задать продолжительность полета, чтобы обеспечить плавное движение самолета.
3. Реализовать логику движения: использовать интерполяцию для плавного перемещения самолета от начальной позиции к конечной, учитывая время полета и обновление позиции в каждом кадре.
4. Обеспечить удаление объекта: реализовать механизм удаления самолета из игрового мира по завершении полета, чтобы избежать накопления неактивных объектов в сцене.
5. Провести тестирование: проверить работоспособность созданного класса в игровом мире, убедиться в корректности движения и удалении самолета, а также оценить его взаимодействие с окружающей средой.

Таким образом, итогом данной работы должно стать создание функционального класса самолета, который будет демонстрировать основные принципы работы с объектами в Unreal Engine 5 и служить основой для дальнейших разработок в области симуляции полета и создания динамичных игровых объектов [9].

2.2.2 Отчет о выполненных работах

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.

#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:

// 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;

};

Листинг 1. Код файла «ArcticFlyingPlane.h»

**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.

#include "ArcticFlyingPlane.h"

// Конструктор класса

AArcticFlyingPlane::AArcticFlyingPlane()

{

// Включение возможности обновления в каждом кадре

PrimaryActorTick.bCanEverTick = true;

// Установка начальной и конечной позиций

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

}

}

Листинг 2. Код файла ArcticFlyingPlane.cpp»

**3) Определение макроса в файле Arctic.h**

В этом файле был определен макрос ARCTIC\_API, который используется для управления экспортом и импортом классов и функций в проекте. Этот макрос позволяет компилятору правильно обрабатывать вызовы функций.

#pragma once

#ifdef ARCTIC\_API

#define ARCTIC\_API \_\_declspec(dllexport)

#else

#define ARCTIC\_API \_\_declspec(dllimport)

#endif

Листинг 3. Код файла «Arctic.h»

**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 5. Результаты тестирования подтвердили его работоспособность и готовность к дальнейшему использованию в более сложных игровых сценариях.

* + 1. 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. Например, можно добавить функционал для управления самолетом с помощью пользовательского ввода, что позволит игрокам управлять полетом и взаимодействовать с окружающей средой. Также можно интегрировать физические эффекты, такие как ветер и аэродинамика, что сделает симуляцию полета еще более реалистичной.

**Перспективы развития 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.

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.

Был разработан метод повышения производительности при рендеринге групп 3d-объектов. Также был изучен полный цикл создания статических 3d-моделей. Изучены средства разработки на языке C++, предоставляемые движком Unreal Engine 5. Приобретены знания об организации проектов.

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: 19.09.2024).
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: 01.11.2024).
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: 12.10.2024).
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: 20.11.2024).
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: 12.12.2024).
7. Procedural generation // Wikipedia, the free encyclopedia [Электронный ресурс] — URL: <https://en.wikipedia.org/wiki/Procedural_generation> (дата обращения: 12.12.2024).
8. Device Reference // World Machine Help [Электронный ресурс] – URL: <https://help.world-machine.com/topics/reference/> (дата обращения: 09.10.2024).
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).