



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное 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:**

**«Разработка реалистичных природных ландшафтов на**  
**Unreal Engine 4»**

Студент    РК6-81Б  
(Группа)

\_\_\_\_\_

**Фёдоров А.В.**  
(Фамилия И.О.)

Руководитель ВКР

\_\_\_\_\_

**Витюков Ф.А.**  
(Фамилия И.О.)

Нормоконтролёр

\_\_\_\_\_

**[REDACTED]**  
(Фамилия И.О.)

2024 г.

**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное 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.)

**«15» февраля 2024 г.**

**ЗАДАНИЕ  
на выполнение выпускной квалификационной работы бакалавра**

Студент группы РК6-81Б

Фёдоров Артемий Владиславович  
(фамилия, имя, отчество)

Тема квалификационной работы: «Разработка реалистичных природных ландшафтов на Unreal Engine 4»

Источник тематики (НИР кафедры, заказ организаций и т.п.): кафедра.

Тема квалификационной работы утверждена распоряжением по факультету РК №\_\_ от «\_\_»  
2024 г.

**Техническое задание**

**Часть 1. Аналитическая часть**

*Изучить полный цикл создания природных ландшафтов. Изучить средства работы с ландшафтами большого размера, предоставляемые трёхмерным движком Unreal Engine 4. Провести анализ различных методов рельефного текстурирования.*

**Часть 2. Практическая часть 1. Создание ландшафтов**

*Создать природную сцену небольшого размера с использованием готовых ассетов. Создать ландшафт большого размера, выполнить импорт в движок Unreal Engine 4, провести оптимизацию отображения.*

**Часть 3. Практическая часть 2. Разработка водоёмов**

*Разработать шейдеры речной воды. Разработать инструмент для создания рек произвольной формы. С помощью полученных инструментов создать и встроить реку в природный ландшафт.*

## **Оформление выпускной квалификационной работы**

Расчетно-пояснительная записка на **76** листах формата А4.

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

*Работа содержит 8 графических листов формата А4.*

Дата выдачи задания: **«10» февраля 2022 г.**

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до **«11» июня 2022 г.**

**Руководитель квалификационной работы**

---

(Подпись, дата)

**Витюков Ф.А.**

(Фамилия И.О.)

**Студент**

---

(Подпись, дата)

**Фёдоров А.В.**

(Фамилия И.О.)

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

**Министерство науки и высшего образования Российской Федерации**  
**Федеральное государственное бюджетное 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 РК6-81Б

Карпенко А.П.  
(Фамилия И.О.)

**«15» февраля 2024 г.**

**КАЛЕНДАРНЫЙ ПЛАН**  
**выполнения выпускной квалификационной работы**

Студент группы РК6-81Б

Фёдоров Артемий Владиславович

(фамилия, имя, отчество)

Тема квалификационной работы: «Разработка реалистичных природных ландшафтов на Unreal Engine 4»

№ 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	10.02.2024	_____	Руководитель ВКР	Витюков Ф.А.
2.	1 часть. Аналитическая часть	18.02.2024	_____	Руководитель ВКР	Витюков Ф.А.
3.	Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач	28.02.2024	_____	Заведующий кафедрой	Карпенко А.П.
4.	2 часть. Практическая часть 1	21.04.2024	_____	Руководитель ВКР	Витюков Ф.А.
5.	3 часть. Практическая часть 2	23.05.2024	_____	Руководитель ВКР	Витюков Ф.А.
6.	1-я редакция работы	28.05.2024	_____	Руководитель ВКР	Витюков Ф.А.
7.	Подготовка доклада и презентации	04.06.2024	_____	Студент	Фёдоров А.В.
8.	Заключение руководителя	10.06.2024	_____	Руководитель ВКР	Витюков Ф.А.
9.	Допуск работы к защите на ГЭК (нормоконтроль)	17.06.2024	_____	Нормоконтролер	Грошев С.В.
10.	Внешняя рецензия	17.06.2024	_____		
11.	Защита работы на ГЭК	21.06.2024	_____		

Студент \_\_\_\_\_  
(подпись, дата)

Фёдоров А.В.  
(Фамилия И.О.)

Руководитель ВКР \_\_\_\_\_  
(подпись, дата)

Витюков Ф.А.  
(Фамилия И.О.)

## **АННОТАЦИЯ**

Работа посвящена различным способам и инструментам разработки реалистичных пейзажей и ландшафтов в Unreal Engine 4.

В данной работе рассмотрены и проиллюстрированы следующие этапы: прохождения полного цикла создания ландшафта, импорт в движок Unreal Engine 4, оптимизация отображения больших ландшафтов с помощью механизма тайлинга, создание материалов ландшафта, добавление растительности в сцены, настройка освещения и погодных условий, реализация интерактивного управления погодными условиями, создание водоёмов и интеграция их в природные сцены.

Тип работы: выпускная квалификационная работа.

Тема работы: «Разработка реалистичных природных ландшафтов на Unreal Engine 4».

Объекты исследований: 3d-моделирование, создание шейдеров, повышение производительности при рендеринге.

## **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 4 (UE4)** – игровой движок, разрабатываемый и поддерживаемый компанией Epic Games.

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

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

**Текстурирование** – процесс создания текстур объекта.

**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.

**Actor (актёр)** – в рамках движка UE4 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.

**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** (скульптуинг) – процесс создания и детализации 3d-моделей с помощью 3d-кистей.

**Rendering** (рендеринг, отрисовка) – процесс получения 2d-изображения по имеющейся 3d-модели.

**Polycount** – количество полигонов модели.

**Static Mesh (UStaticMesh)** – класс, представляющий собой статический геометрический объект. Хранит данные о полигональной сетке 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.

**Шейдер (Shader)** – компьютерная программа, выполняющаяся параллельно на графическом процессоре, и служащая для различных графических вычислений, таких как отрисовка 3д моделей, расчёт освещения и так далее.

**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.

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

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

**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.

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

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

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

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

**Rendering Hardware Interface (RHI)** – в UE4 надстройка над множеством графических API (например: DirectX, OpenGL, Vulkan), позволяющая писать независимый от платформы код.

**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.

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

# **СОДЕРЖАНИЕ**

<b>ВВЕДЕНИЕ .....</b>	<b>11</b>
1. Создание природной сцены со статичной камерой.....	14
1.1. Создание ландшафта .....	15
1.2. Заполнение сцены объектами .....	16
1.3. Настройка освещения и погодных условий .....	17
2. Создание большой сцены с природным ландшафтом.....	20
2.1 Создание карты местности.....	21
2.2 Создание материала ландшафта .....	22
2.3 Добавление растительности в сцену.....	24
2.4 Добавление деталей .....	26
2.5 Обзор техник рельефного текстурирования.....	27
2.5.1 Измерение производительности различных методов рельефного текстурирования .....	32
2.6 Измерение производительности сцен с ландшафтом .....	34
3. Разработка инструментов для создания водоёмов .....	36
3.1 Разработка шейдера речной воды .....	37
3.1.1 Предварительное создание материала воды .....	37
3.1.2 Реализация ряби на поверхности воды .....	39
3.1.3 Реализация изменения цвета в зависимости от глубины.....	40
3.1.4 Реализация эффекта полного отражения света при низких углах падения.....	42
3.1.5 Реализация пены на поверхности воды .....	43
3.2 Разработка инструмента для создания рек .....	45
3.3 Внедрение водоёмов в готовую природную сцену .....	47
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>48</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....</b>	<b>48</b>

## **ВВЕДЕНИЕ**

В современном мире трехмерные ландшафты находят применение во многих сферах. Большие сцены с участием природных пейзажей, созданных с помощью компьютерной графики, повсеместно используются в телевидении, кинематографе, видеоиграх, в качестве демонстрационного материала для различных архитектурных проектов.

В данной работе рассматривается создание природных ландшафтов с использованием трёхмерного движка Unreal Engine 4. Из всех инструментов Unreal Engine выделяет ориентированность на реалистичность получаемого изображения и возможность работы «в реальном времени», то есть не требуются длительные расчёты как при рендре видео с помощью таких инструментов как 3ds Max, Blender и др. Это позволяет быстро привносить изменения в проекты и создавать интерактивные сцены, способные меняться в зависимости, например, от ввода пользователя.

Работа делится на **три** основные части:

1. Создание сцены малого размера со статичной камерой;
2. Создание большой ландшафтной сцены;
3. Разработка инструментов для создания водоёмов;

### *Часть первая. Создание сцены со статичной камерой*

Статичные природные сцены могут использоваться как фоны для ведущих телевидения, интерьерные интерактивные картины, рекламные фоны. В таких сценах большую роль играет уровень детализации, ведь в случае, если зритель будет приглядываться к изображению, которое не меняется, он быстрее заметит отсутствие мелких деталей и других погрешностей.

Другим важным аспектом является освещение, зачастую именно оно определяет уровень реалистичности изображения. Грамотное использование прямого и отраженного света, атмосферной перспективы в виде тумана и других природных эффектов может позволить получить картинку, местами почти неотличимую от фотографии.

При создании таких сцен важно ориентироваться на реальные фотографии природы и анализировать их составляющие.

В рамках первой части данной работы рассмотрено создание небольшой природной сцены, а также настроено интерактивное управление погодой в сцене.

### *Часть вторая. Создание большой ландшафтной сцены*

Большие ландшафтные сцены могут использоваться для кинематографа и видеоигр. При создании сцен, по которым будет перемещаться камера, помимо детализации и освещения важную роль играют «наполненность сцены» и оптимизация.

Под «наполненностью» подразумевается равномерное распределение объектов, привлекающих внимание, таких как деревья, растения и камни. Потенциальный зритель или игрок не должен натыкаться на куски локации, выглядящие пустыми или переполненными, так как это сильно скажется на уровне реализма.

Оптимизация – достижение приемлемой производительности на всех целевых платформах при разработке видеоигры. Для сохранения стабильно высокого уровня производительности важно использовать качественные ассеты: 3д модели, использующие оптимальное количество полигонов для нужного уровня детализации. Существует набор способов оптимизации, достигаемых программно:

*Использование LOD (Level Of Detail)* - Для этого необходимо на стадии моделирования создать одну или несколько дополнительных версий модели с пониженной детализацией, а затем, после импорта в движок, настроить зависимость уровня детализации от дистанции между объектом и камерой, либо от процента площади экрана, занимаемой объектом.

*Тайлинг (Tiling) ландшафта* – Техника, похожая на LOD, но относящаяся к 3д моделям ландшафта. Ландшафт делится на квадратные «куски», для каждого из которых создается набор LOD-ов. Правильный LOD отображается в

зависимости от расстояния между камерой и куском ландшафта. Такая техника позволяет отображать огромные ландшафты с сохранением производительности.

*Использование технологий микрорельефного текстурирования* – создание рельефа поверхностей без создания большого количества полигонов, а с помощью текстур. Повсеместное использование таких техник сильно повышает детализированность сцены без большого ущерба производительности.

В рамках второй части данной работы рассмотрено создание большой ландшафтной сцены с использованием технологий оптимизации.

Цели работы:

1. создать природную сцену со статичной камерой;
2. разработать систему управлению погодой по нажатию клавиши;
3. создать и оптимизировать природную сцену большого размера;
4. разработать инструменты для создания водоёмов;

Для выполнения работы были использованы средства следующих программ:

- Unreal Engine 4 – создание сцен и разработка внутриигровых систем;
- Blender – создание 3d-модели ландшафта небольшой сцены;
- Adobe Substance 3D Designer – создание текстур;
- World Machine – создание ландшафтов больших размеров.

## **1. Создание природной сцены со статичной камерой**

Процесс создания статичных сцен начинается с изучения референсов для определения общей композиции и конкретных элементов, которые будут заполнять сцену.



Рисунок 1. Фотографии-референсы, взятые за основу при создании сцены.

Для поддержания хорошего баланса детализации сцена должна содержать большие, средние и малые элементы. Основными элементами композиции будут являться деревья, валуны и лесная тропа. В качестве средних элементов будут выступать ветки и камешки, лежащие на земле, а также различные кусты, а в качестве мелких элементов будет выступать трава.

## 1.1. Создание ландшафта

Ландшафт сцены был создан с помощью программы blender 3d и импортирован в проект виде Static Mesh.

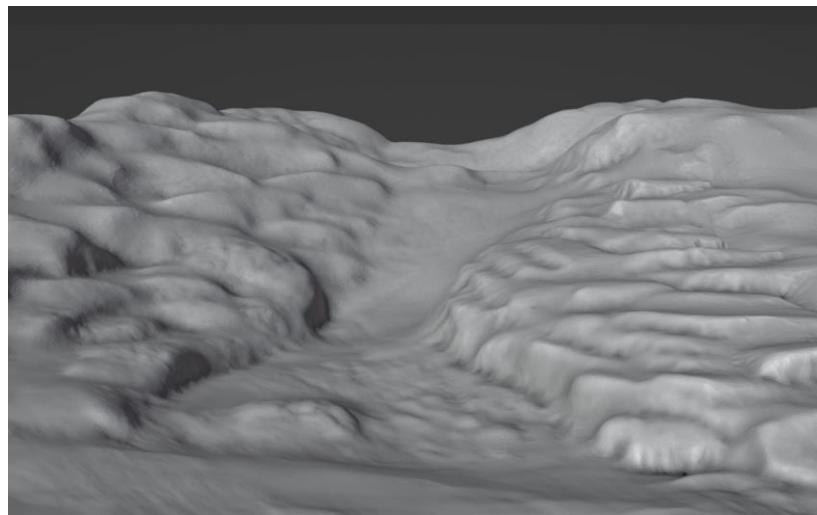


Рисунок 2. Базовая 3д модель лесной тропы

Был создан материал, состоящий из трёх «слоёв»: травы, камня и песка. Распределение слоёв было реализовано с помощью Vertex Colors: материал каждой вершины модели выбирается в соответствии с её цветом, состоящим из трёх каналов: красного, зеленого и синего. Высокие значения в красном канале отвечает за материал травы, Высокие значения в зеленом канале отвечает за материал камней, значения, близкие к нулю в зеленом и красном канале отвечают за материал песка. Благодаря этому возможно распределение материалов ландшафта с помощью “раскрашивания” модели внутри редактора Unreal Engine.

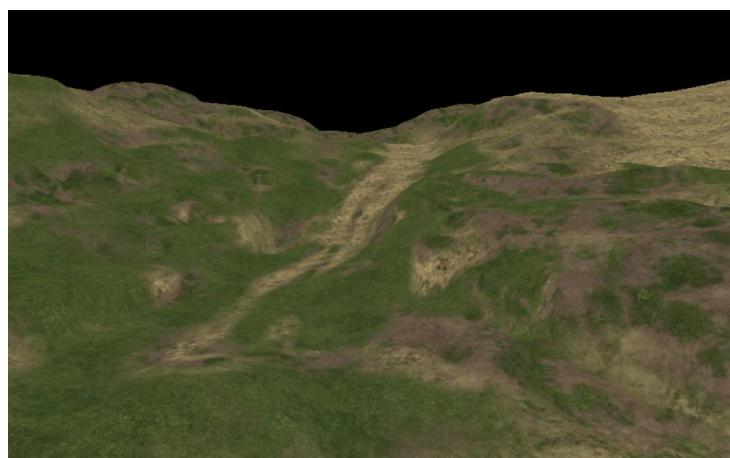


Рисунок 3. Многослойный материал примененный к модели

## 1.2. Заполнение сцены объектами

Основные элементы композиции – камни и деревья – были расставлены вручную. Были использованы Megascans – высококачественные модели, созданные с помощью сканирования реальных объектов.



Рисунок 4. Основные элементы композиции

Инструмент Foliage Tool позволяет размещать в сцене большое количество одинаковых моделей, таких как пучки травы, деревья и камни, при этом производительность сцены гораздо выше, чем в случае, если бы все модели были размещены вручную. Это достигается за счёт технологии Static Mesh Instancing. С помощью инструмента Foliage Tool были расставлены модели травы, упавших веток, маленьких камней и различных ростков.



Рисунок 5. Озеленение сцены

### 1.3. Настройка освещения и погодных условий

Освещение сцены состоит из Directional Light – источника прямого света, имитирующего солнце, Sky Light – источника рассеянного света, и Exponential Height Fog – объекта, создающего эффект тумана.



Рисунок 6. Сцена при ясной погоде

Эффект дождя был создан при помощи повышения плотности тумана, увеличения отражающей способности материалов (Specular) для создания эффекта влажной поверхности. Частицы дождя были созданы с помощью системы частиц Niagara FX. Визуальный эффект капель, стекающих по стеклу, был реализован с помощью Post-Process Material.

Для возможности переключения погоды по нажатию клавиши был создан класс *AWeatherHandler*. Функции *AWeatherHandler::SetClearWeather()* и *AWeatherHandler::SetRainWeather()* отвечают за установку ясной и дождливой погоды соответственно. Данные функции представлены в листингах 1 и 2.



Рисунок 7. Сцена при дождливой погоде

Листинг 1. Функция *SetClearWeather()*, отвечающая за переключение погоды на ясную.

```
void AWeatherHandler::SetClearWeather()
{
    UE_LOG(LogTemp, Warning, TEXT("Set Clear Weather"));
    for (int Index = 0; Index < DirectionalLights.Num(); ++Index)
    {
        ULightComponent* DirectionalLightComponent = DirectionalLights[Index]-
>FindComponentByClass<ULightComponent>();
        DirectionalLightComponent->SetIntensity(DirectinalLightIntensityClear);
        DirectionalLights[Index]->SetActorRotation(DirectinalLightAngleClear);
    }
    for (UNiagaraComponent* Component : RainFX)
    {
        Component->Deactivate();
    }
    SkyLight->SetIntensity(SkyLightIntensityClear);

    for (const AActor* it : WetActors)
    {
        UStaticMeshComponent* SMC = it->FindComponentByClass<UStaticMeshComponent>();
        UMaterialInstanceDynamic* Material = (UMaterialInstanceDynamic*)SMC-
>GetMaterial(0);
        Material->SetScalarParameterValue(FName(TEXT("Wetness")), 0);
    }
    Fog->SetFogDensity(FogDensityClear);
    FPostProcessSettings& PostProcessSettings = PPV->Settings;
    if (TestMatIns) {
        TestMatInsDyna = UKismetMaterialLibrary::CreateDynamicMaterialInstance(this,
TestMatIns);
        FWeightedBlendable WeightedBlendable;
        WeightedBlendable.Object = TestMatInsDyna;
        WeightedBlendable.Weight = 0;
        PostProcessSettings.WeightedBlendables.Array.Empty();
        PostProcessSettings.WeightedBlendables.Array.Add(WeightedBlendable);
    }
}
```

Листинг 2. Функция *SetRainWeather()*, отвечающая за переключение погоды на дождливую.

```
void AWeatherHandler::SetRainWeather()
{
    UE_LOG(LogTemp, Warning, TEXT("Set Rain Weather"));
    for (int Index = 0; Index < DirectionalLights.Num(); Index++) {
        ULightComponent* DirectionalLightComponent = DirectionalLights[Index]-
>FindComponentByClass<ULightComponent>();
        DirectionalLightComponent->SetIntensity(DirectinalLightIntensityRain);
        DirectionalLights[Index]->SetActorRotation(DirectinalLightAngleRain);

        for (AActor* it : WetActors) {
            UStaticMeshComponent* SMC = it-
>FindComponentByClass<UStaticMeshComponent>();
            UMateralInstanceDynamic* Material = (UMaterialInstanceDynamic*)SMC-
>GetMaterial(0);
            Material->SetScalarParameterValue(FName(TEXT("Wetness")), 1);
        }
    }

    for (UNiagaraComponent* Component : RainFX) {
        Component->Activate();
    }

    SkyLight->SetIntensity(SkyLightIntensityRain);
    Fog->SetFogDensity(FogDensityRain);

    FPostProcessSettings& PostProcessSettings = PPV->Settings;

    if (TestMatIns) {
        TestMatInsDyna =
UKismetMaterialLibrary::CreateDynamicMaterialInstance(this, TestMatIns);
        FWeightedBlendable WeightedBlendable;
        WeightedBlendable.Object = TestMatInsDyna;
        WeightedBlendable.Weight = 1;
        PostProcessSettings.WeightedBlendables.Array.Empty();
        PostProcessSettings.WeightedBlendables.Array.Add(WeightedBlendable);
    }
}
```

## **2. Создание большой сцены с природным ландшафтом**

Сцена будет состоять из скалистого острова, окруженного водой. Перед созданием ландшафта были изучены фото-референсы.



Рисунок 8. Фотографии-референсы, взятые за основу при создании сцены.

## 2.1 Создание карты местности

Ландшафт сцены с островом был создан в программе World Machine, позволяющей создавать реалистичные пейзажи с помощью таких техник как наложение различных шумов (шум Перлина, Симплексный шум) друг на друга и эрозия грунта ветром и водой.

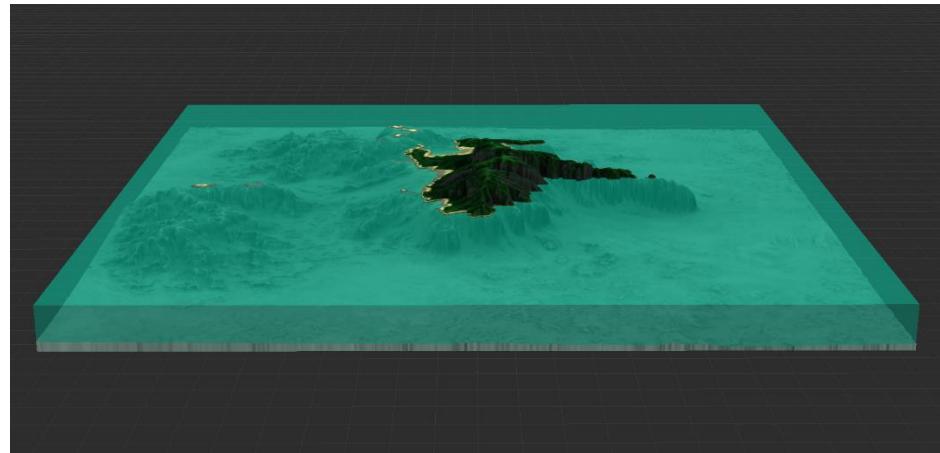


Рисунок 9. Ландшафт, созданный в World Machine

Ландшафт был разбит на 16x16 карт высот 253x253 пикселей каждая для представления в виде тайлов (квадратных “кусков” ландшафта) в World Composition. World Composition позволяет работать с большими сценами, показывая близкие к наблюдателю тайлы в высоком качестве и далёкие от него в низком качестве. Это позволяет увеличивать размеры сцены без потери производительности.

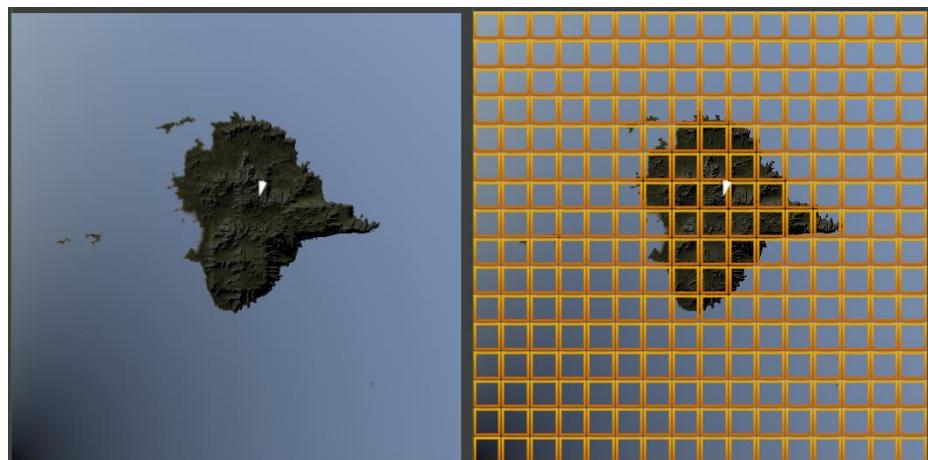


Рисунок 10. Представление ландшафта в World Composition

## 2.2 Создание материала ландшафта

Материал ландшафта был создан с использованием технологий Landscape Layers и Triplanar Mapping (box mapping).

Landscape Layers – технология, позволяющая хранить информацию о различных «слоях» ландшафта, что даёт возможность «раскрашивать» местность разными типами поверхности. В данном случае были использованы следующие слои:

- Скалы
- Песок
- Зеленая трава
- Выцветшая трава
- Каменистое дно водоёмов

Техника Triplanar Mapping позволяет избегать «растягивания» текстур на отвесных гранях и заключается в интерполяции между проекциями текстуры в зависимости от направления поверхности. Разработанная для этого Material Function представлена на Рисунке 9.

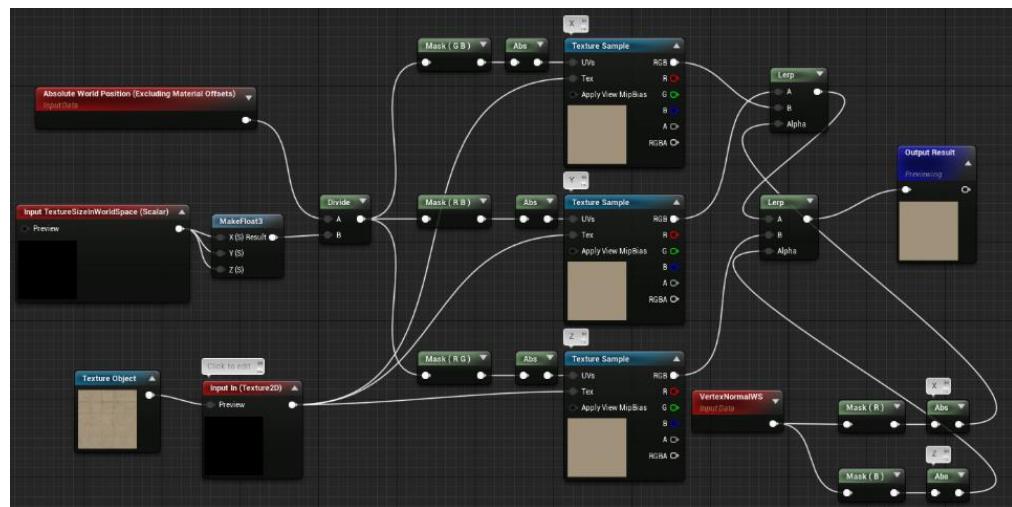


Рисунок 11. Material function для triplanar mapping



a)

б)

Рисунок 12. Материал ландшафта а) без использования triplanar mapping  
б) с использованием triplanar mapping



Рисунок 13. Ландшафт, импортированный в Unreal Engine

## **2.3 Добавление растительности в сцену**

Технология Landscape Grass позволяет размещать различные типы травы на пейзаж и управлять ими из материала ландшафта. На сцену были добавлены несколько видов полевой травы с помощью Landscape Grass. Использованные материалы травы используют Vertex Animation, анимации, заключающейся в изменении позиций индивидуальных вершин модели внутри шейдера. Это позволяет получить эффект растительности, раскачивающейся на ветру. Скорость, турбулентность и другие параметры ветра настраиваются в материалах растений.



Рисунок 14. Трава, созданная с помощью Landscape Grass

Технология Procedural Foliage Spawner позволяет управлять «посадкой» растительности: для каждого вида растений задается плотность размещения, время роста, допустимая близость к другим объектам и другие параметры, позволяющие создать натуральный вид. С помощью Procedural Foliage Spawner по всей карте были размещены четыре разновидности елей. Важно использовать отличающиеся модели деревьев для создания эффекта разнообразия растительности и избежать однообразности ландшафта.



Рисунок 15. Различные модели деревьев, использованные в сцене



Рисунок 16. Результат работы Procedural Foliage Spawner

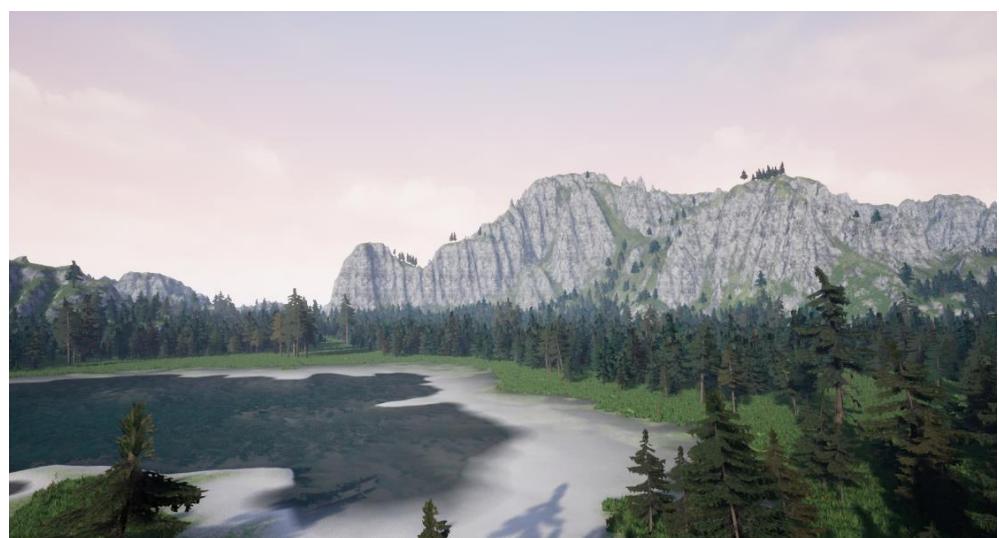


Рисунок 17. Сцена в финальном вид

## 2.4 Добавление деталей

На горе была создана поляна с цветами и тропинкой, размещена модель деревянного дома. Эффект «вытоптанной» тропинки был реализован с помощью раскрашивания травы в цвет грунта и уменьшения её размеров. Для передачи информации о положении тропинки использована технология Virtual Texture. Она заключается в создании текстуры, доступ к которой имеют различные объекты сцены: материал ландшафта записывает в текстуру данные о расположении тропинки, а материал растительности считывает их для определения высоты и цвета травы.

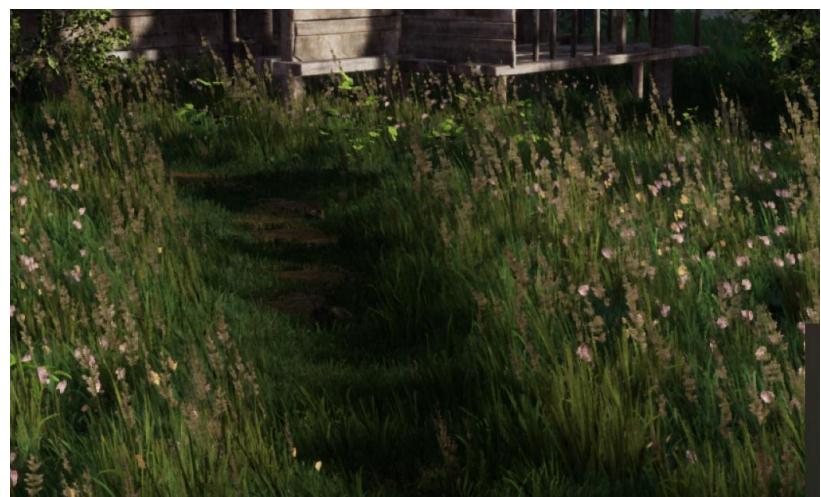


Рисунок 18. «Вытоптанная» тропинка, ведущая к дому

Конечный вариант сцены с настроенным освещением представлен на рисунке 17.



Рисунок 19. Сцена с домом на горе.

## **2.5 Обзор техник рельефного текстурирования**

Большие элементы 3д моделей отображаются с помощью набора полигонов, однако использовать полигоны для отображения рельефа может быть очень невыгодно с точки зрения производительности программы.

Мелкие детали такие как морщины на коже или камни на земле могут быть изображены с помощью 2д текстуры, наложенной на модель. Но такой подход может выглядеть недостаточно убедительно, при приближении к поверхности будет заметно что все детали на самом деле плоские, а при изменении угла освещения тени на деталях не будут изменяться.

Для достижения лучшего результата существует ряд технологий **рельефного текстурирования**.

Для демонстрации различных технологий в программе Substance 3D Designer был разработан материал речных камней. Данный материал содержит большое количество мелких деталей, которые необходимо убедительно отобразить.

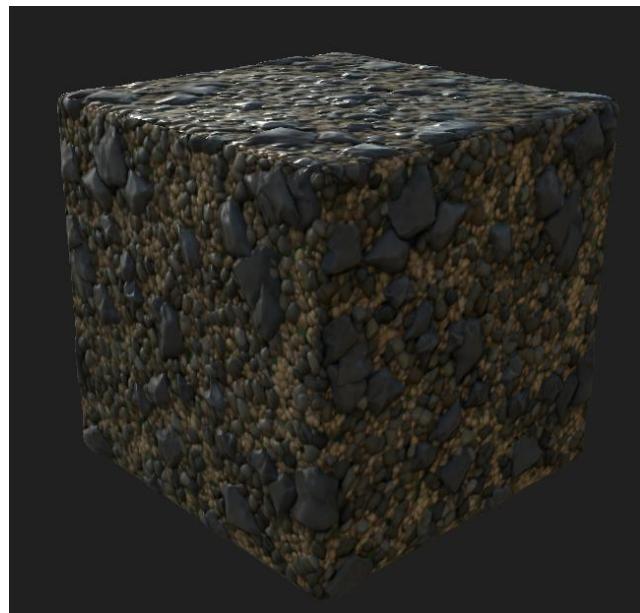


Рисунок 1. Демонстрация материала, разработанного в Substance 3D Designer

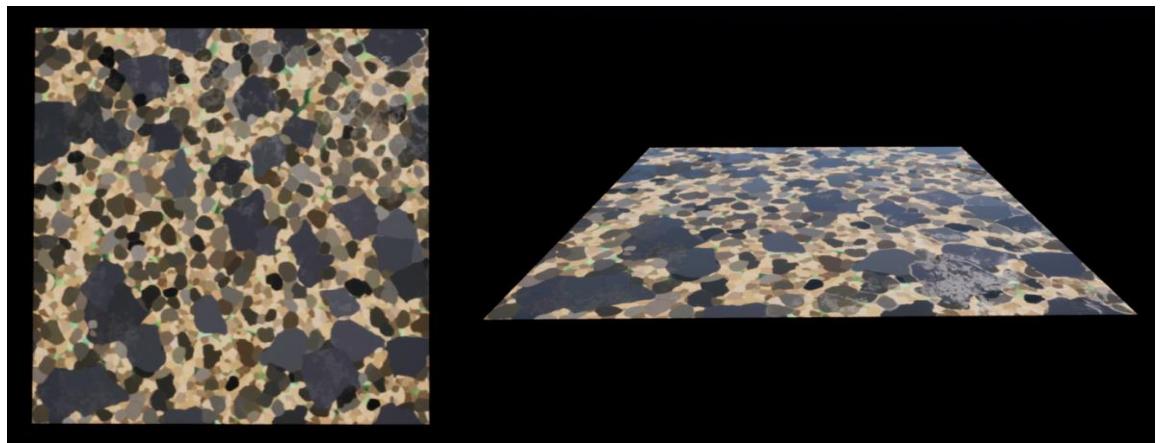


Рисунок 2. Базовое наложение текстуры без использования технологий рельефного текстурирования

### Bump Mapping

Технология bump mapping заключается в использовании текстуры, в которую закодирована карта высот микрорельефа, для затенения определенных элементов модели, таких как трещины и щели.

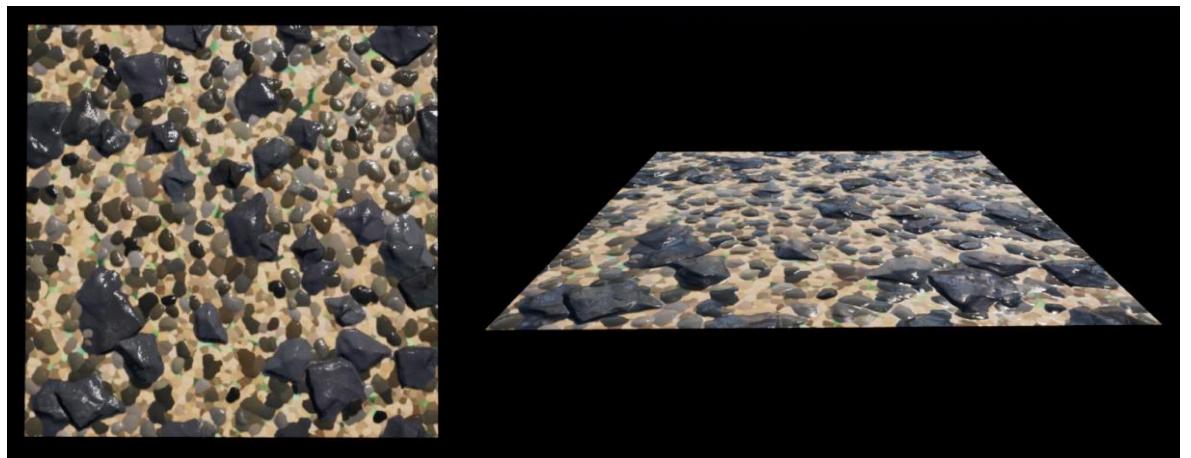


Рисунок 3. Материал, использующий bump mapping.

### Normal Mapping

Normal Mapping – технология, дающая результат похожий на bump mapping, но гораздо более распространённая. Вместо информации о высоте рельефа используется информации об нормалях поверхности, позволяющая управлять отражением падающего света. Normal mapping имеет заметный эффект на блестящих материалах с сильным рефлексом.

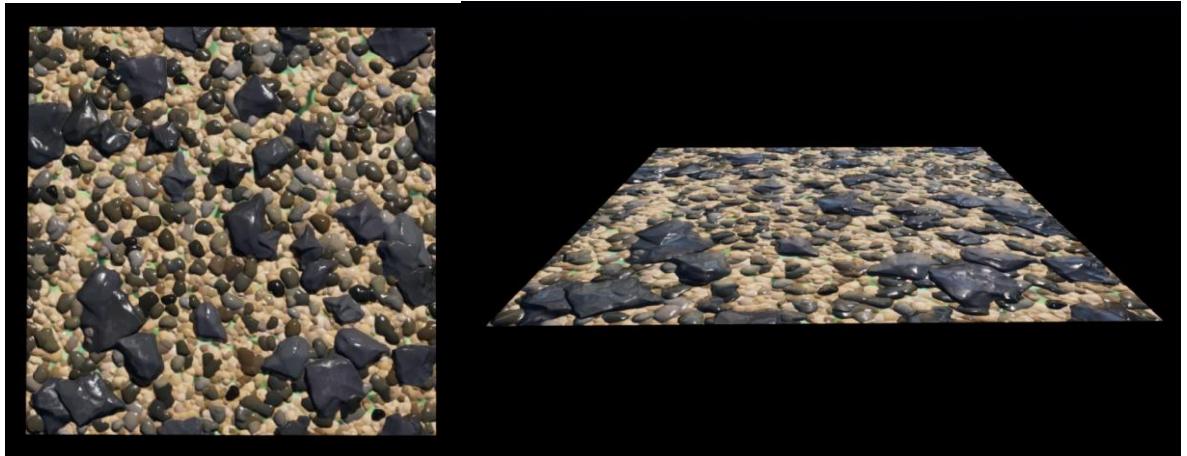


Рисунок 4. Материал, использующий normal mapping

### Bump offset

Bump offset позволяет в некоторой степени передать перекрытие выступающими элементами материала более плоских участков, добавляя глубины.

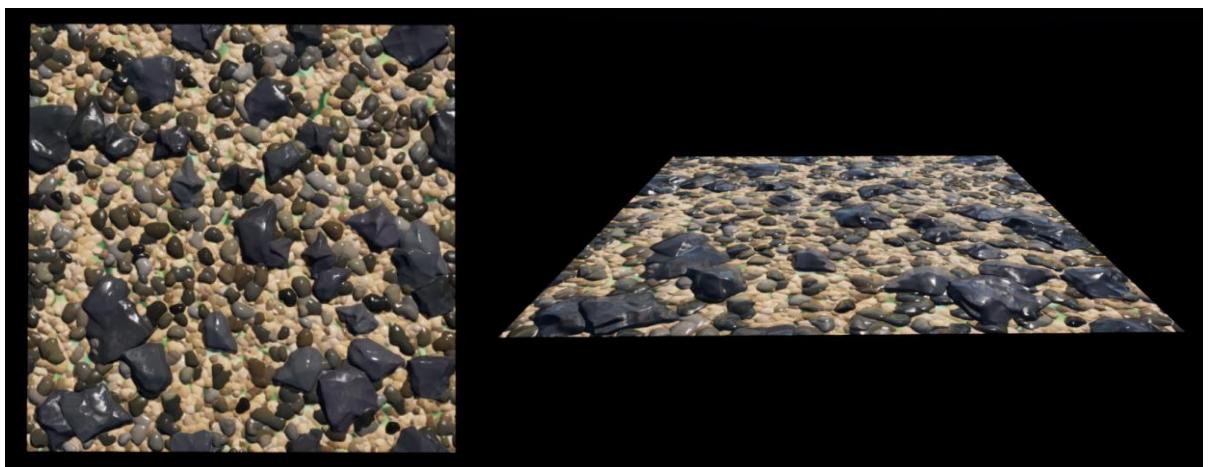


Рисунок 5. Материал, использующий normal mapping и bump offset

### Parallax occlusion mapping

Фактически представляет собой форму локальной трассировки лучей в пиксельном шейдере. Трассировка лучей используется для определения высот и

учёта видимости текселей. Иными словами, данный метод позволяет создавать ещё большую глубину рельефа при небольших затратах полигонов и применении сложной геометрии. Недостаток метода — невысокая детализация силуэтов и граней.

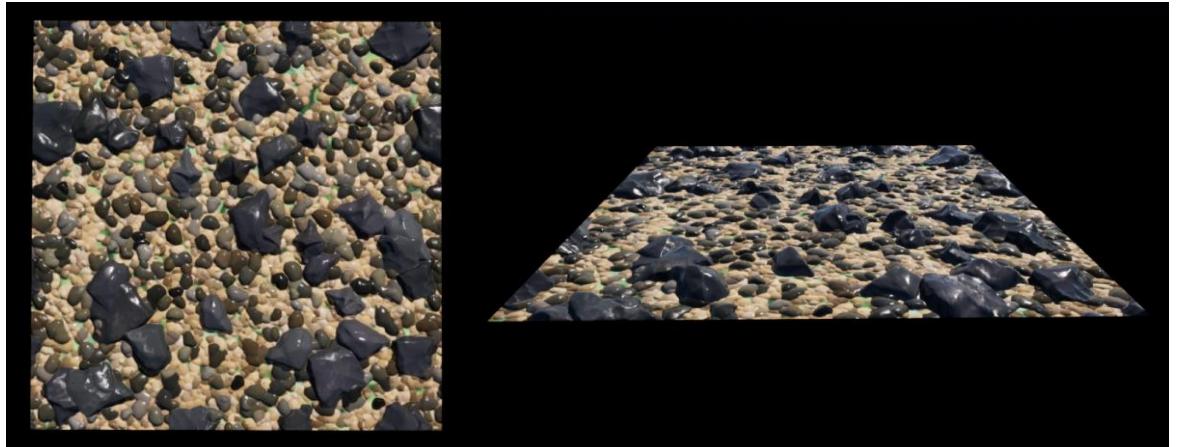


Рисунок 5. Материал, использующий normal mapping и parallax occlusion mapping

### **Displacement mapping**

Displacement mapping заключается в изменении геометрии поверхности по карте высот. В отличии от вышеупомянутых технологий результат не «плоский» и микрорельеф имеет реальный объёмный вид.

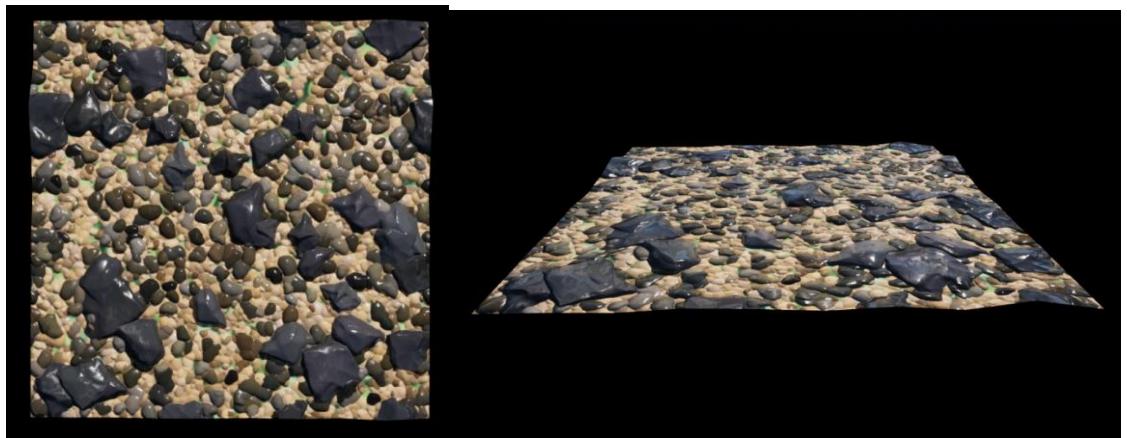
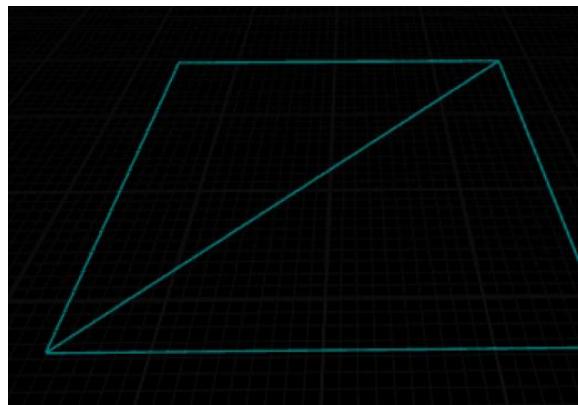


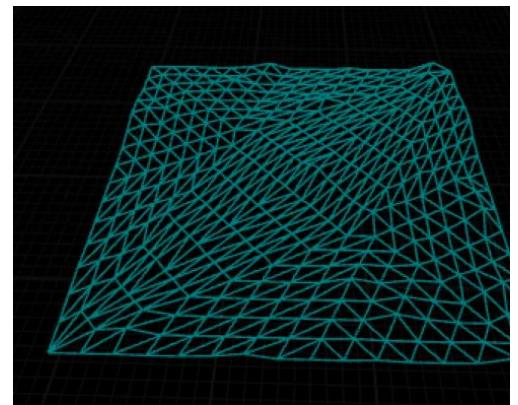
Рисунок 5. Материал, использующий normal mapping и displacement mapping.

### **Tessellation**

Tessellation (замощение, тесселяция) — техника автоматизированного добавления новой геометрии с целью повышения детализации 3д модели. На рисунке 5 демонстрируется тесселяция модели, состоящей из двух полигонов.



А)



Б)

Рисунок 5. А) Материал, не использующий Tessellation

Б) Материал, использующий Tessellation

Тесселяция позволяет использовать displacement mapping с более заметным эффектом: при приближении камеры к объекту увеличивать детализацию сетки, тем самым создавая «реальный» объёмный рельеф.

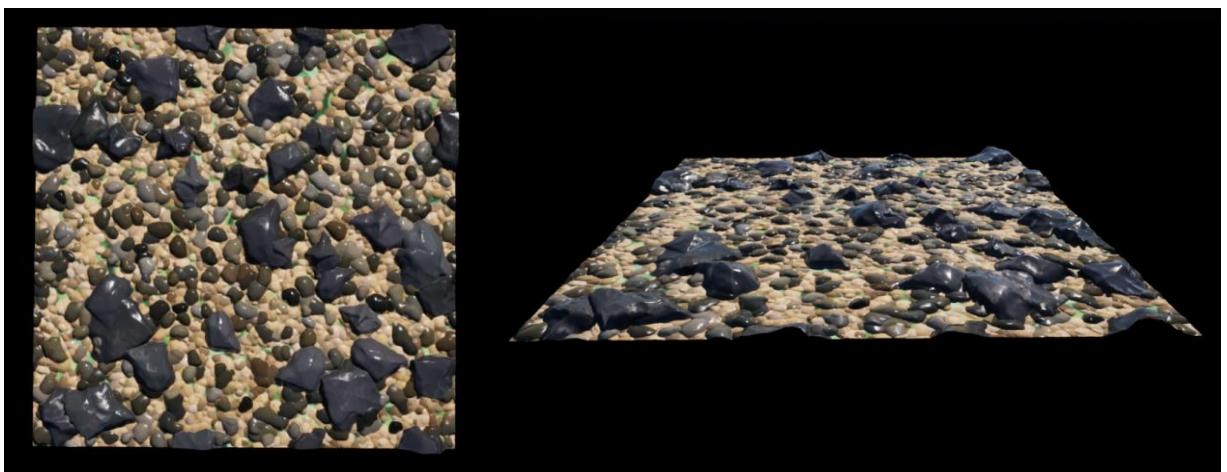


Рисунок 6. Материал, использующий normal mapping, displacement mapping и tessellation.

## 2.5.1 Измерение производительности различных методов рельефного текстурирования

Был проведен тест по измерению зависимости производительности программы от выбранного метода рельефного текстурирования.

В рамках теста была создана пустая сцена содержащая в себе модель дизайнерского стула, состоящую из около 200 тысяч полигонов.

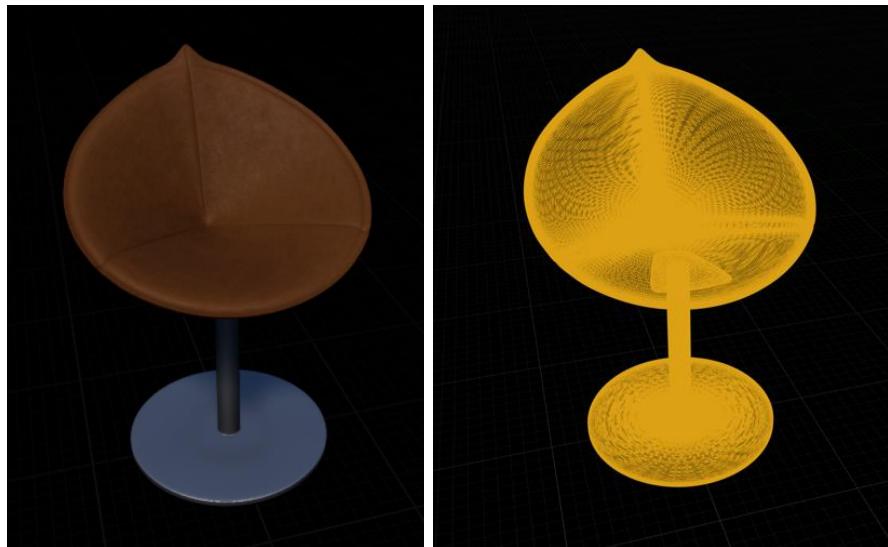


Рисунок 20. Высокополигональная модель стула

Для каждого из методов рельефного текстурирования был создан соответствующий материал обивки кресла и был произведен замер производительности. На рисунке () представлен график зависимости производительности сцены от плотности сетки.

Метод	FPS	FPS, %
Без метода	146	100
Bump map	140	95,89041
Normal map	140	95,89041
Bump offset	133	91,09589
Parallax occlusion	130	89,0411
Displacement	125	85,61644
Displacement+Tessellation	60	41,09589

Таблица 1. Измерения производительности

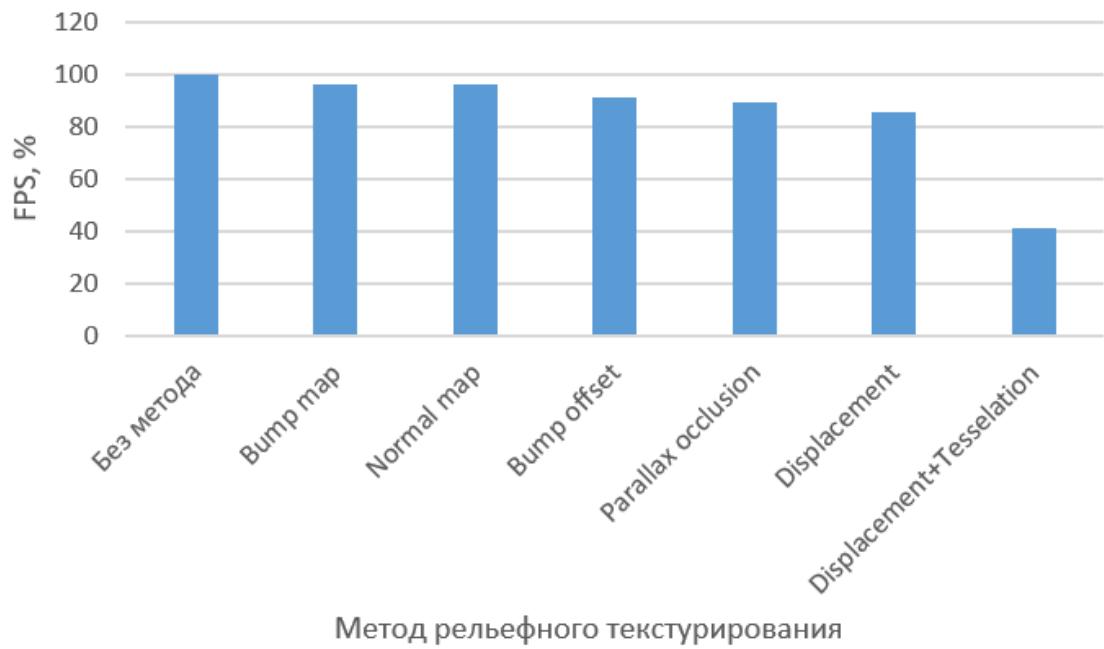


Рисунок 21. Зависимость количества кадров в секунду от выбранного метода рельефного текстурирования.

По итогам теста был сделан вывод: при использовании методов Bump Mapping и Normal Mapping нет значительной потери производительности. Bump Offset и Parallax Occlusion Mapping, так же приводят к небольшим потерям производительности. При сравнивать потерей производительности и визуальных результатов, которые достигается за их счёт, можно прийти к выводу что использование методов рельефного текстурирования эффективно.

Характеристики компьютера, на котором проводились замеры:

Процессор 12<sup>th</sup> Gen Intel Core i5-12600;

Графический процессор NVIDIA GeForce GTX 1050 Ti, 32 Гб ОЗУ.

## 2.6 Измерение производительности сцен с ландшафтом

Был проведен тест по измерению зависимости производительности программы от плотности сетки ландшафта.

В рамках теста в программе World Machine был создан несложный ландшафт размером 4x4 км и экспортирован в виде карты высот в следующих разрешениях: 8129x8129, 4033x4033, 2017x2017, 1009x1009, 505x505, 253x253 и 127x127 пикселей.

Для каждой из плотностей сетки был произведен замер производительности. На рисунке 20 представлен график зависимости производительности сцены от плотности сетки.

Количество вершин	Плотность сетки, пиксель/метр	FPS
66 080 641 (8129)	2.03	68
16 265 089	1.01	126
4 068 289	0.50	134
1 1018 081	0.25	135
255 025	0.06	140
16 129	0.03	141

Таблица 1. Измерения производительности

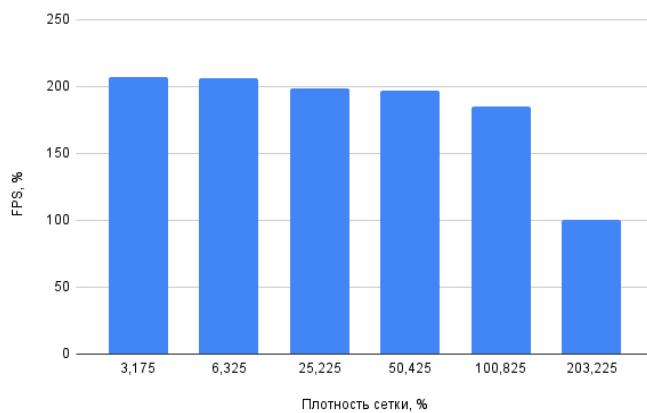


Рисунок 20. Зависимость количества кадров в секунду от плотности сетки ландшафта.

Внешний вид ландшафтов с различными плотностями карты высот представлен на рисунке 21.

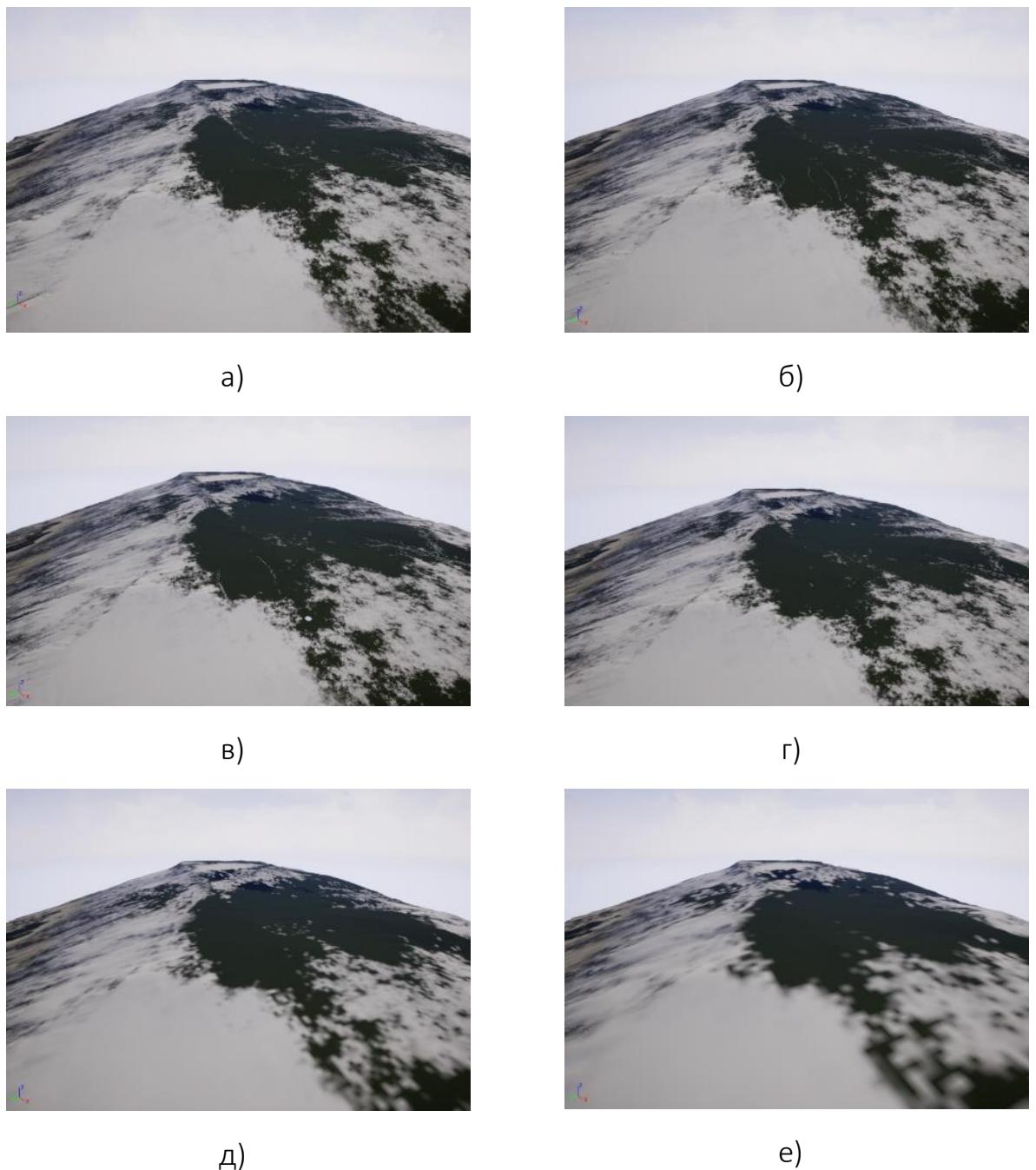


Рисунок 21. Внешний вид ландшафта при плотностях карты высот  
 а) 2.03225 б) 1.00825 в) 0.50425 г) 0.25225 д) 0.06325 е) 0.03175  
 пикселей на метр.

По итогам теста был сделан вывод: при использовании ландшафта с плотностью карты высот 0.5 пикселя на метр нет значительной потери производительности ни качества изображения.

Характеристики компьютера, на котором проводились замеры:

Процессор 12<sup>th</sup> Gen Intel Core i5-12600;

Графический процессор NVIDIA GeForce GTX 1050 Ti, 32 ГБ ОЗУ.

### **3. Разработка инструментов для создания водоёмов**

Различные водоёмы, такие как реки, озёра, моря, являются неотъемлемой частью многих пейзажей, поэтому важно понимать техники создания и внедрения водоёмов в природные сцены.



Рисунок 22. Фотографии-референсы, взятые за основу при создании материалов воды.

### **3.1 Разработка шейдера речной воды**

После анализа некоторого количества референсов водоёмов был выделен ряд важных деталей:

- Двигающаяся рябь на поверхности воды
- Рефракция света
- Поглощение света, зависящее от глубины
- Отражение света в зависимости от угла, под которым наблюдается вода
- Пена, появляющаяся на границах воды и в местах резкого изменения направления течения

#### **3.1.1 Предварительное создание материала воды**

Был создан материал с параметрами *Blend Mode: Translucent* и *Refraction Mode: Pixel Normal Offset*. Первый параметр позволяет управлять прозрачностью материала с помощью свойства *Opacity*, второй отвечает за тип рефракции.

В Unreal Engine 4 присутствуют два типа рефракции в прозрачных материалах: *Index Of Refraction* и *Pixel Normal Offset*.

*Index Of Refraction* вычисляет отклонение лучей по физическим законам преломления в зависимости от индекса отражения материала (IOR), эта техника подходит для небольших объектов, таких как посуды, стекол и других изогнутых поверхностей. Однако при использовании с большими объектами, такими как водоёмы, может иметь непредсказуемые результаты и вызывать визуальные артефакты.

*Pixel Normal Offset* создаёт иллюзию физического преломления, используя разницу между реальной нормалью поверхности, и нормалью поверхности, вычисленной, например, с помощью карт нормалей (см. *Normal Mapping*). Результат не обладает большой реалистичностью, но хорошо подходит для больших объектов, и объектов, чьи нормали быстро изменяются, таких как водоёмы с волнами. Именно этот тип рефракции использован в данном материале.

Для создания отражений на воде был использован параметр *Screen Space Reflections: On* и *Lighting Mode: Surface Translucency Volume*. *Screen Space Reflections* (SSR) – это метод повторного использования данных экрана для расчета отражений. SSR — дорогостоящий метод с точки зрения вычислений, но при правильном использовании он может дать отличные результаты.

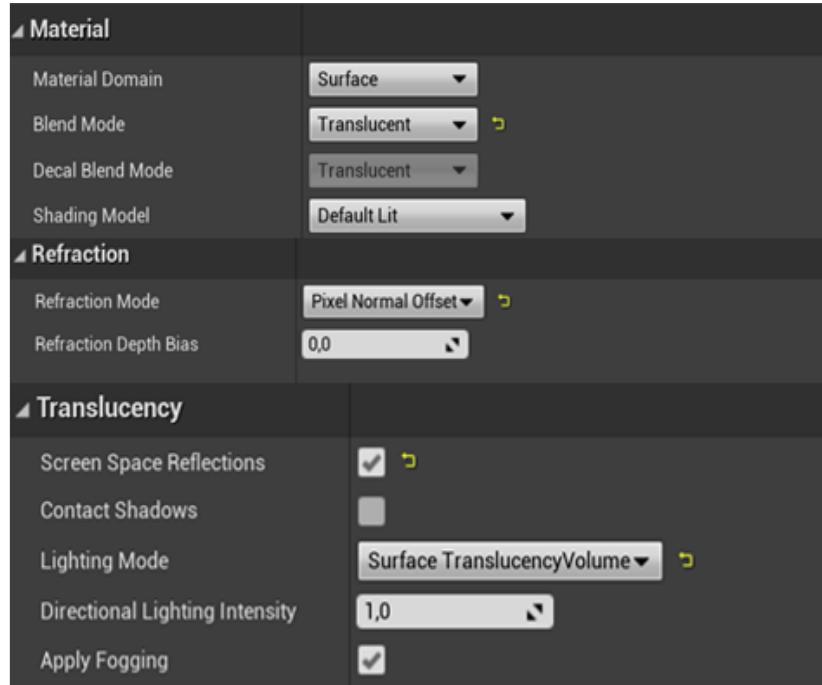


Рисунок 23. Параметры материала воды.

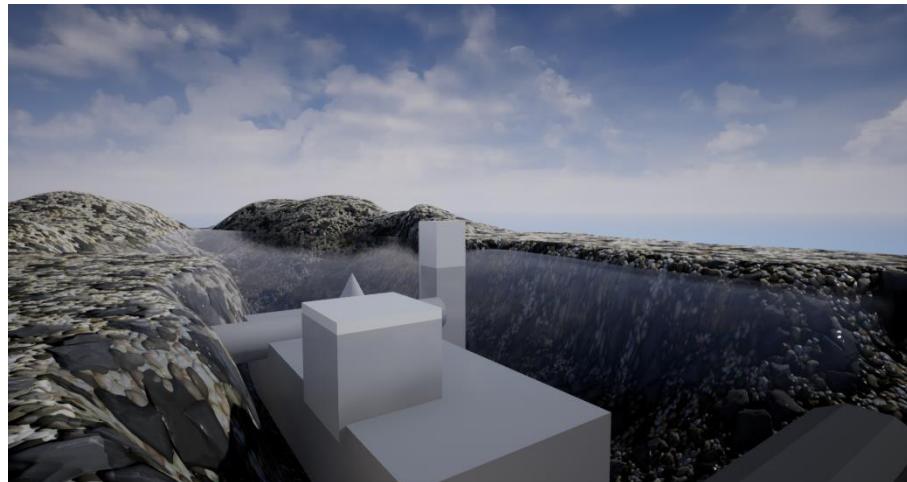


Рисунок 24. Заготовка для материала воды в тестовой сцене

### 3.1.2 Реализация ряби на поверхности воды

Для создания ряби была выбрана техника Scrolling Textures – «прокрутка текстур», заключающаяся, в данном случае, в использовании движущихся карт нормалей. Для удобного управлением прокруткой текстур была создана функция (material function) MF\_DirectionalPanner.

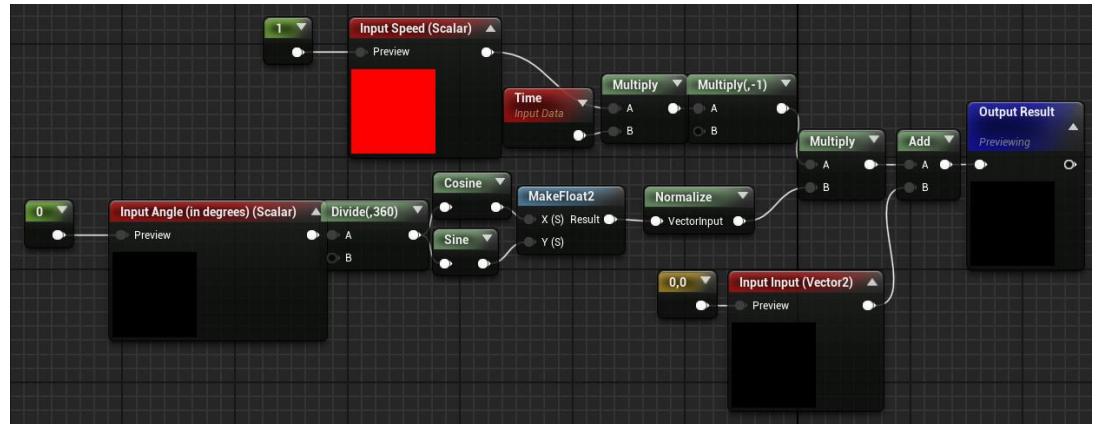


Рисунок 25. Функция MF\_DirectionalPanner.

Для более реалистичного и «хаотичного» внешнего вида ряби были использованы три карты нормалей двигающиеся с разными скоростями в разных направлениях.

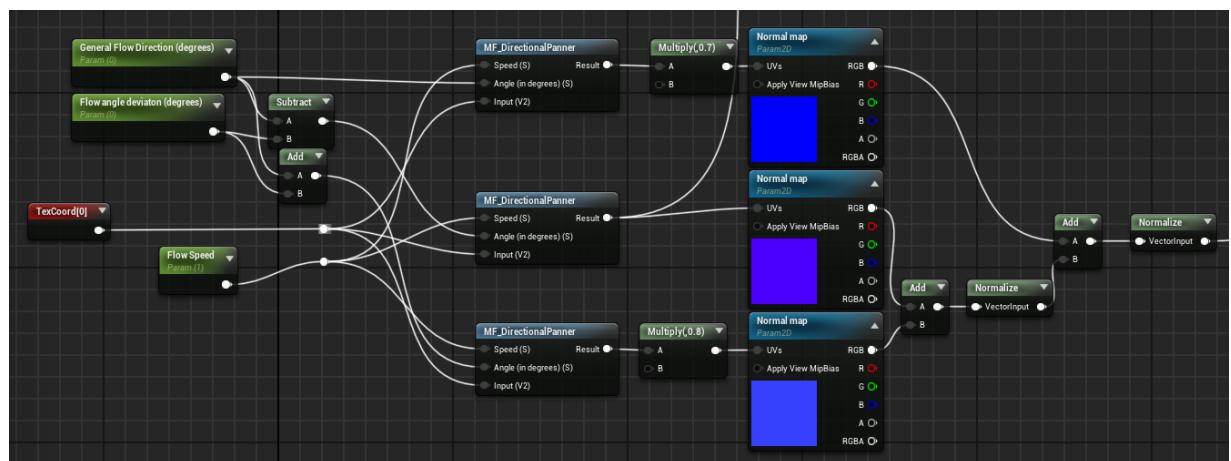


Рисунок 26. Часть шейдера, отвечающая за прокрутку карт нормалей.

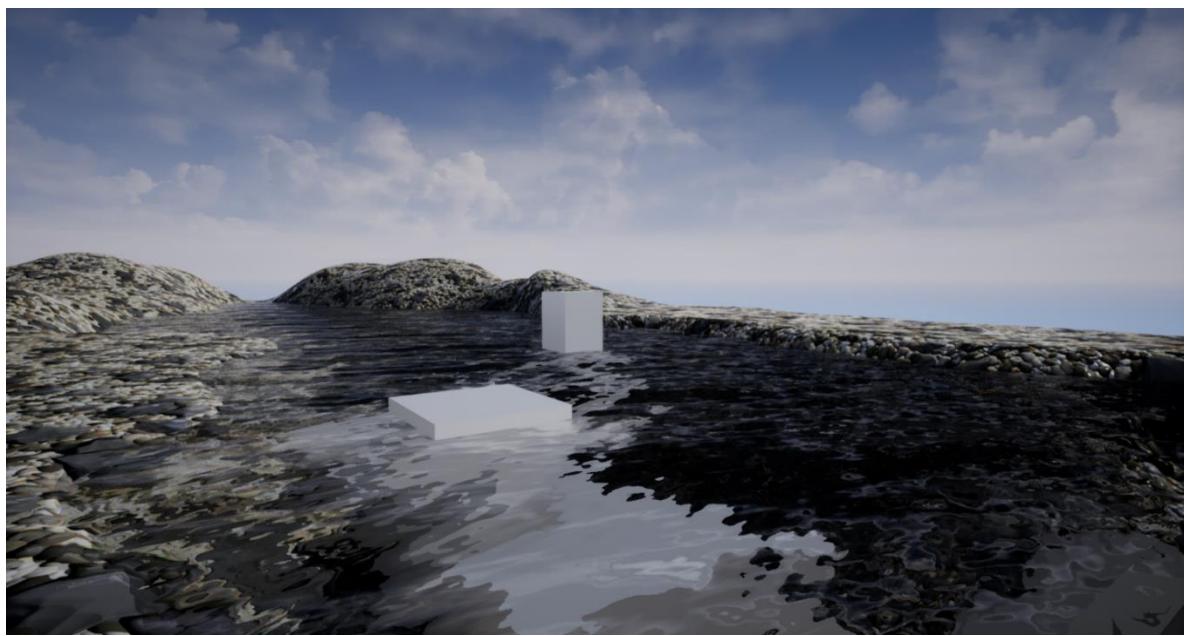


Рисунок 27. Рябь на поверхности воды.

### 3.1.3 Реализация изменения цвета в зависимости от глубины

Одним из способов определения глубины воды может быть сравнение расстояний от камеры до дна с расстоянием от камеры до поверхности воды. В таком случае глубина находится по формуле:

$Depth = Scene\ Depth - Pixel\ Depth$ , где Scene Depth – расстояние от камеры до дна, а Pixel Depth – расстояние от камеры до поверхности.

Это «дешёвая» техника, но не такая реалистичная, ведь как правило освещенность воды меняется в зависимости от расстояния от дна до поверхности воды а не до наблюдателя.

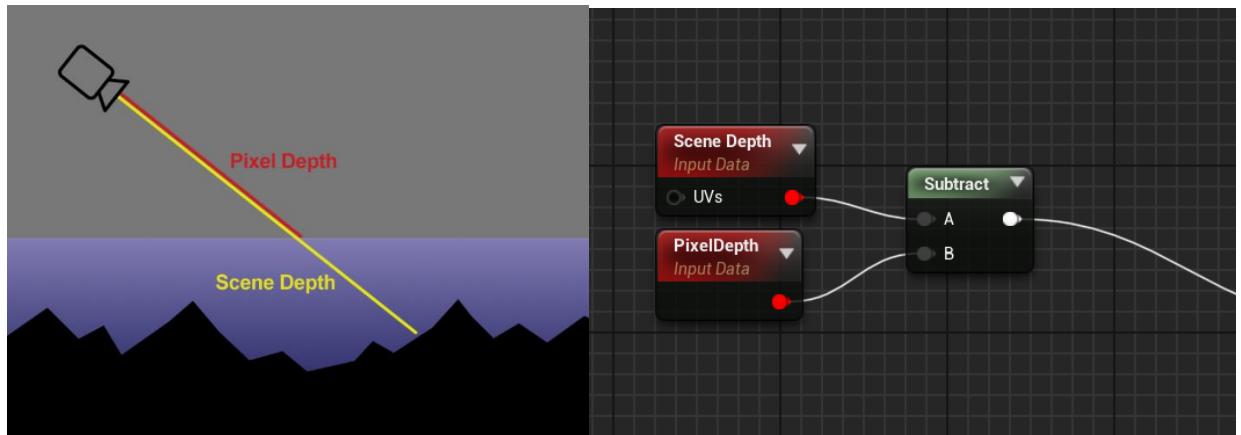


Рисунок 28. Схема и реализация базового определения глубины воды

Для более реалистичного результата можно находить вертикальное расстояние от дна до поверхности воды, тогда формула для глубины будет следующей:

$Depth = (Absolute\ World\ Position - Camera\ Position) * (Scene\ Depth / Pixel\ Depth) - Water\ Level$ , где Absolute World Position – положение отрисовываемого пикселя в абсолютной системе координат, Camera Position – положение камеры в абсолютной системе координат, Scene Depth – расстояние от камеры до дна, Pixel Depth – расстояние от камеры до поверхности, и Water Level – положение поверхности воды в абсолютной системе координат.

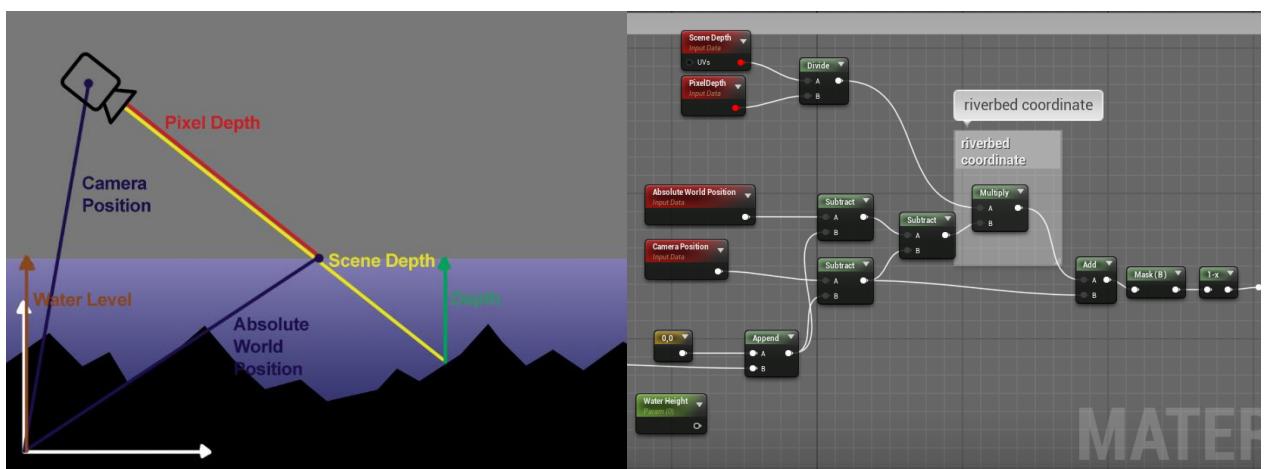


Рисунок 29. Схема определения расстояния от поверхности воды до дна

Такая схема приводит к гораздо более реалистичному результату, но может быть использована только на относительно горизонтальных поверхностях, так как необходимо знать высоту поверхности в каждой её точке.

Теперь, с использованием одной из двух схем можно реализовать изменение цвета и прозрачности в зависимости от глубины. В данном случае цвет будет терять яркость по мере увеличения, вместе с уменьшением прозрачности.

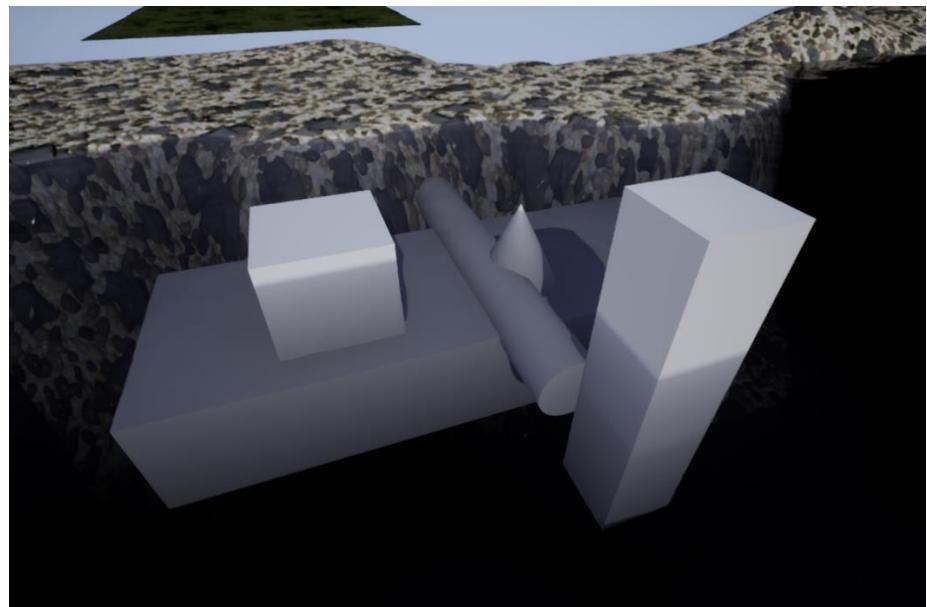


Рисунок 30. Зависимость цвета и прозрачности воды от глубины

### 3.1.4 Реализация эффекта полного отражения света при низких углах падения.

С помощью функции Fresnel реализуется полное отражение света при низких углах падения: материал становится полностью непрозрачным и полностью отражающим.

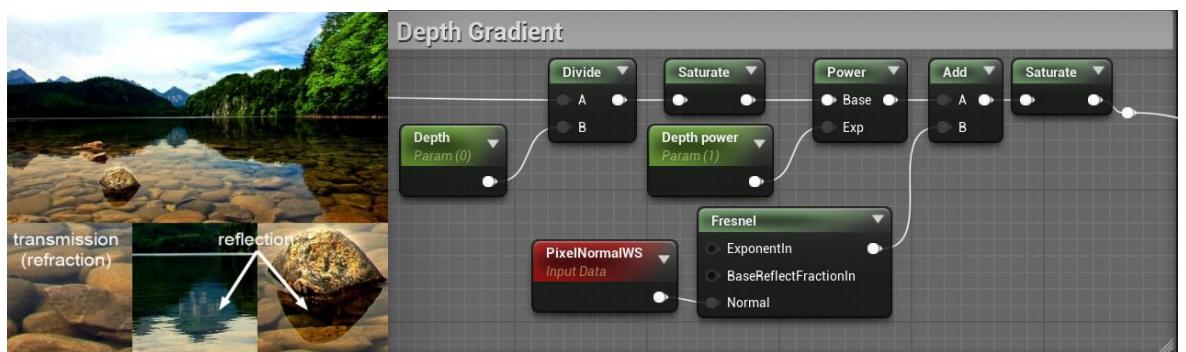


Рисунок 31. Часть шейдера, отвечающая за полное отражение лучей света при низких углах обзора.

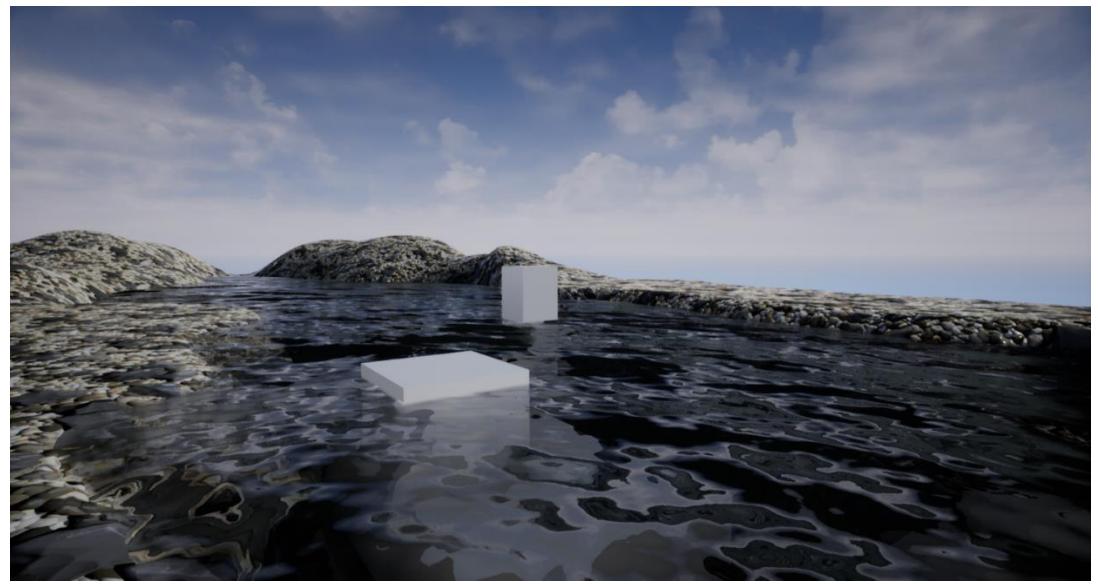


Рисунок 32. Материал воды с поглощением света на глубине и полном отражении при низких углах падения света.

### 3.1.5 Реализация пены на поверхности воды

В потоке пена появляется на контакте на границах воды, поэтому для определения положения пены можно использовать функцию *DistanceToNearestSurface*, возвращающую расстояние до ближайшей поверхности. Более того, для того чтобы иметь некоторый художественный контроль над размещением пены, она также зависит от цвета вершин модели (Vertex Coloring), что позволяет вручную «раскрашивать» водоём пеной.

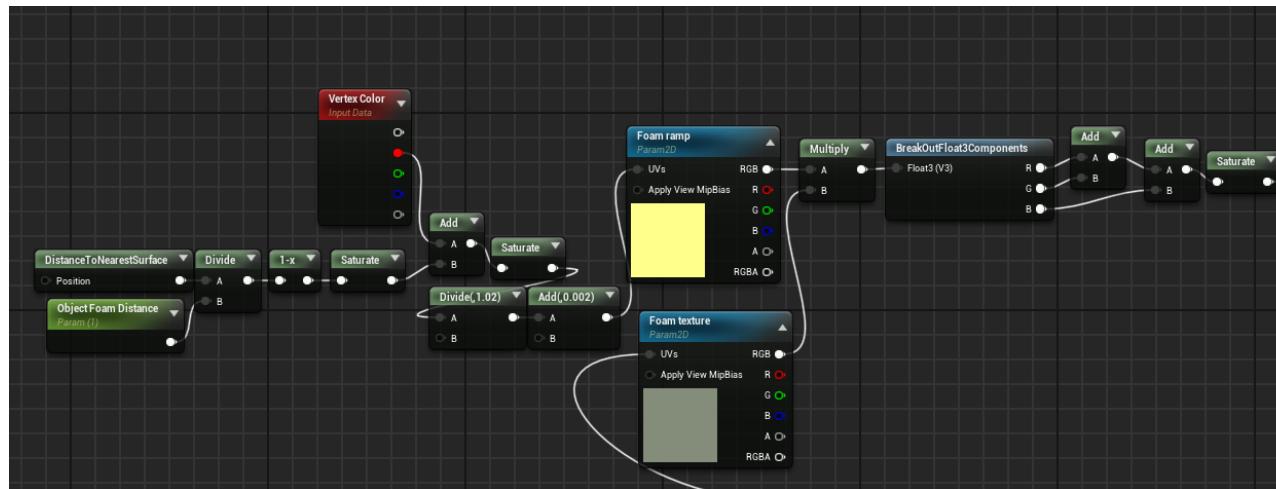


Рисунок 33. Часть шейдера, отвечающая за размещение пены.

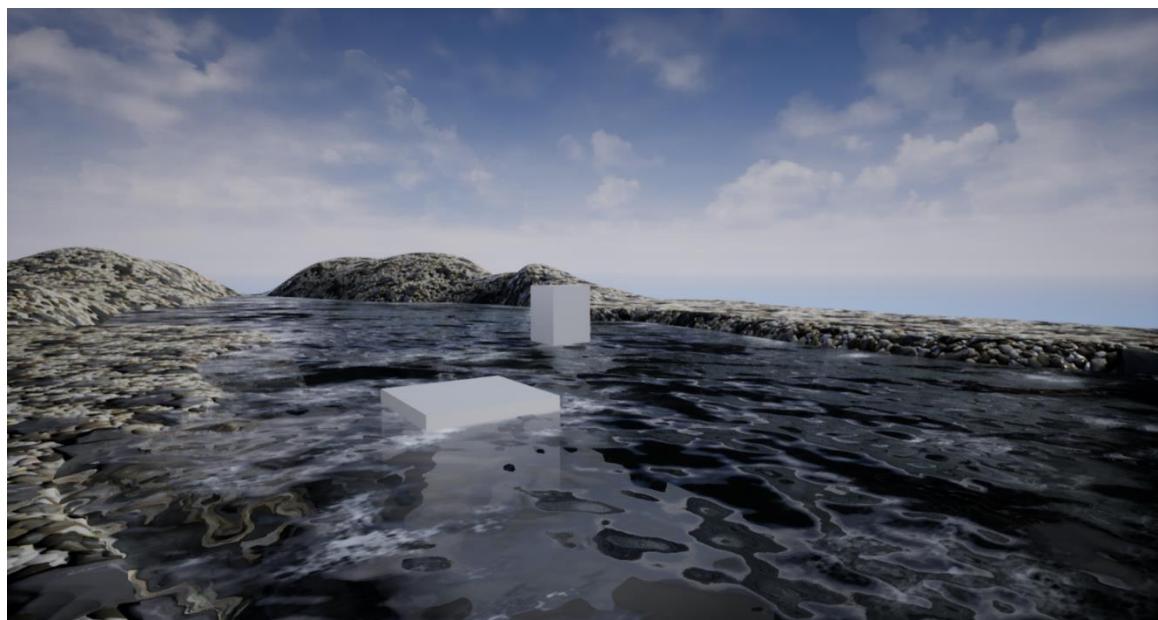


Рисунок 34. Материал воды с пеной на границах с сушей.

### 3.2 Разработка инструмента для создания рек

С помощью системы **Blueprints** был разработан инструмент, позволяющий создавать реки по заданному пользователем объемному сплайну. На рисунке 17 представлена демонстрация работы инструмента. Белым цветом выделен модифицируемый сплайн, по которому создается набор Сплайн-Мешей (Spline-Mesh), 3д моделей, чья геометрия «изогнута» по форме кривой.

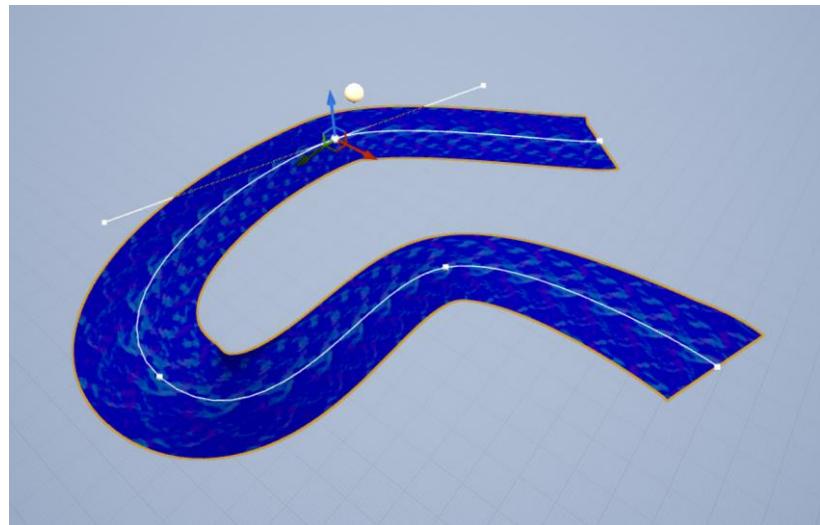


Рисунок 35. Демонстрация инструмента для создания рек.

Для того чтобы направление течения всегда было параллельно руслу реки UV координаты были организованы как на рисунке 18. Таким образом была достигнута видимая неразрывность прокручивающихся текстур волны.

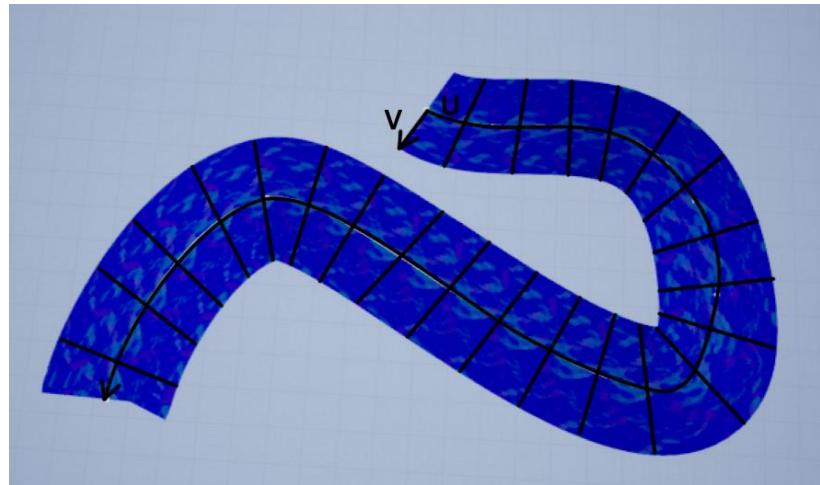


Рисунок 36. Схематичные UV координаты сплайн-меша реки.

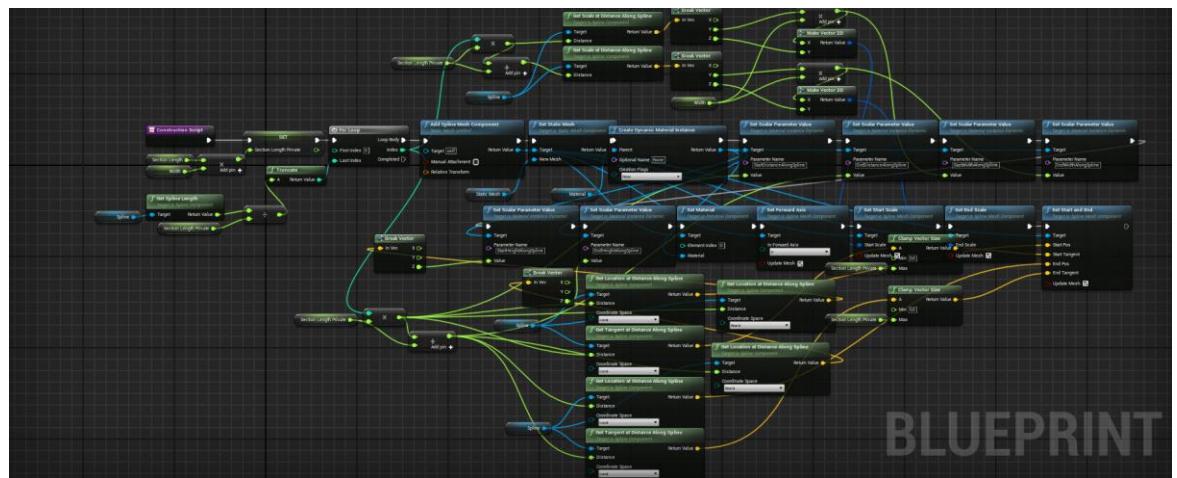


Рисунок 37. Blueprint инструмента для создания рек.

### **3.3 Внедрение водоёмов в готовую природную сцену**

В природную сцену, созданную во время эксплуатационной практики, добавлена река с помощью разработанных материалов воды и инструмента для создания рек.



Рисунок 38. Река в контексте природного ландшафта.

С использованием того-же инструмента для создания рек и пены на воде был реализован водопад. С помощью системы частиц *Niagara* были создан эффект брызг водопада.

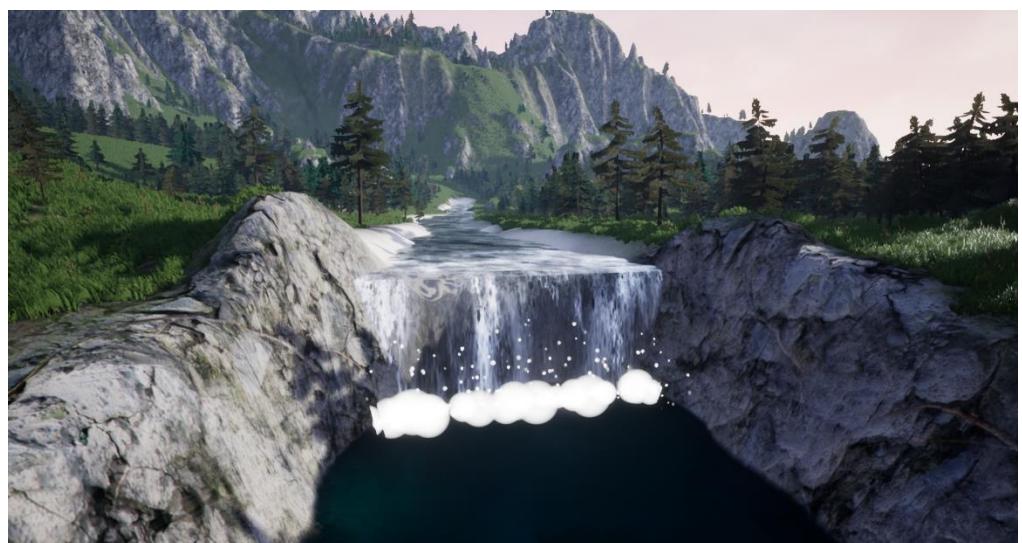


Рисунок 39. Водопад.

## **ЗАКЛЮЧЕНИЕ**

В результате работы были изучены методы создания природных сцен. Изучены средства разработки на языке C++, и технология Blueprints предоставляемые движком Unreal Engine 4. Приобретены знания об организации проектов.

Также в рамках данной работы были изучены различные методы оптимизации природных сцен.

Тут что то ещё нужно

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Unreal Engine 4 Documentation // Unreal Engine Documentation URL: <https://docs.unrealengine.com/>. Дата обращения: 07.04.2022;
2. Display aspect ratio // Wikipedia, the free encyclopedia URL: [https://en.wikipedia.org/wiki/Display\\_aspect\\_ratio/](https://en.wikipedia.org/wiki/Display_aspect_ratio/). Дата обращения: 13.04.2022;
3. Geometry instancing // Wikipedia, the free encyclopedia URL: [https://en.wikipedia.org/wiki/Geometry\\_instancing](https://en.wikipedia.org/wiki/Geometry_instancing). Дата обращения: 21.04.2022;
4. Modeling – Blender Manual // Blender Manual URL: <https://docs.blender.org/manual/en/latest/modeling/index.html>. Дата обращения: 18.02.2022;
5. Мирмап // Wikipedia, the free encyclopedia URL: <https://en.wikipedia.org/wiki/Мирмап>. Дата обращения 05.03.2022;
6. Level of Detail (computer graphics) // Wikipedia, the free encyclopedia URL: [https://en.wikipedia.org/wiki/Level\\_of\\_detail\\_\(computer\\_graphics\)](https://en.wikipedia.org/wiki/Level_of_detail_(computer_graphics)). Дата обращения: 05.04.2022;
7. Creating and Using LODs // Unreal Engine Documentation URL: <https://docs.unrealengine.com/4.27/en-US/WorkingWithContent/Types/StaticMeshes/HowTo/LODs/>. Дата обращения: 05.04.2022;
8. Инстансинг [Электронный ресурс] // Habr: интернет-портал. URL: <https://habr.com/ru/post/352962/>. Дата обращения: 03.04.2022;
9. UE4 Optimization: Instancing // YouTube: видео хостинг. URL: <https://www.youtube.com/watch?v=oM1bV2rQO4k>. Дата обращения: 03.04.2022;
10. Божко А.Н., Жук Д.М., Маничев В.Б. Компьютерная графика. [Электронный ресурс] // Учебное пособие для вузов. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2007. - 389 с., - ISBN 978-5-7038-3015-4, Режим доступа: <http://ebooks.bmstu.ru/catalog/55/book1141.html>. Дата обращения: 10.02.2022;
11. Programming Quick Start // Unreal Engine Documentation URL: <https://docs.unrealengine.com/5.0/en-US/unreal-engine-cpp-quick-start/>. Дата обращения: 29.12.2021.