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 (РК)

КАФЕДРА Системы автоматизированного проектирования (РК6)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«Методы создания ландшафта и его элементов в трёхмерном движке Unreal Engine 4»

Студент РК6-75Б _____ **Киселев С. А.**
(Подпись, дата) И.О. Фамилия

Руководитель курсового проекта _____ **Витюков Ф.А.**
(Подпись, дата) И.О. Фамилия

2023 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
А.П. Карпенко

«_____» _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине _____Модели и методы анализа проектных решений_____

Студент группы ____РК6-75Б___

_____Киселев Сергей Андреевич_____
(Фамилия, имя, отчество)

Тема курсового проекта Методы создания ландшафта и его элементов в трёхмерном движке Unreal Engine 4

Направленность КП (учебный, исследовательский, практический, производственный, др.) учебный
Источник тематики (кафедра, предприятие, НИР) __кафедра

График выполнения проекта: 25% к 5 нед., 50% к 11 нед., 75% к 14 нед., 100% к 16 нед.

Задание. С помощью движка Unreal Engine 4 воссоздать фотореалистичный пейзаж, используя различные техники создания ландшафта. Добиться максимальной реалистичности и детализации.

Оформление курсового проекта:

Расчетно-пояснительная записка на 27 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):
3 графических листа.

Дата выдачи задания «10» сентября 2022 г.

Руководитель курсовой работы _____ **Витюков Ф.А.**
(Подпись, дата) И.О. Фамилия

Студент _____ **Киселев С. А.**
(Подпись, дата) И.О. Фамилия

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

АННОТАЦИЯ

В данной работе рассмотрены подходы для создания фотореалистичного пейзажа местности. Описаны основные компоненты, а также, рассмотрены принципы создания материалов. Рассмотрена реализация анимации движения солнца по небу и эффекта дождя. Рассмотрен способ создания материала водной поверхности.

В расчётно-пояснительной записке 27 страниц, 20 рисунков, 3 графических листа.

СОДЕРЖАНИЕ

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

UE4 – трёхмерный движок Unreal Engine 4.

Кат-сцена — это эпизод, который прерывает геймплей и используется, чтобы продвинуть сюжет, представить развитие персонажа и обеспечить информацией фона, атмосферой, диалогом и ключами.

Референс - вспомогательное изображение, которое художник или дизайнер изучает перед работой, чтобы точнее передать детали, получить дополнительную информацию, идеи.

3D-модель - это объемная фигура в пространстве, создаваемая в специальной программе. За основу, как правило, принимаются чертежи, фотографии, рисунки и подробные описания, опираясь на которые, специалисты и создают виртуальную модель.

Материал - набор настроек, описывающий свойства поверхности. Материал можно назначить на любой объект, но нельзя использовать как задний фон.

Текстура – изображение, накладываемое на поверхность 3D-модели. Может содержать различные свойства поверхности, например: цвет, жёсткость (roughness), смещение (displacement), направление нормалей (normal map), и т.д.

Полигон – многоугольник, являющийся базовым компонентом 3D-сетки. Основные типы: треугольник (tri), четырёхугольник (quad) и n-gon (5 или более вершин).

ВВЕДЕНИЕ

При создании ландшафта с помощью движка Unreal Engine 4 одной из главных задач является достижение как можно более реалистичной картинки пейзажа. Для достижения фотореалистичности важно правильно понимать масштабы объектов и передачу цветов при взаимодействии с пользователем. Для правильного понимания природы местности перед началом работы в движке необходимо изучить все особенности по референсам из интернета с похожей местностью в реальном мире. При этом важно понимать, какие детали являются ключевыми, а какие нет необходимости реализовывать. В движке UE4 присутствует множество плагинов и ассетов, но, к сожалению, встроенных методов зачастую бывает недостаточно, чтобы реализовать все задумки.

Немаловажной частью разработки является продумывание пользовательского пути его использования. При этом можно выделить такие основные аспекты удобства, как простота управления, интуитивность дизайна, производительность.

При создании уровней разработчику необходимо удерживать тонкую грань между красотой картинки и производительностью. Для достижения комфортной для пользователя производительности важно применять различные способы оптимизации на всех этапах разработки.

В данной работе рассматривается разработка нескольких приемов для достижения максимальной реалистичности пейзажа.

1. Создание ландшафта

Для эффективности Unreal разбивает сетку на секции и компоненты.

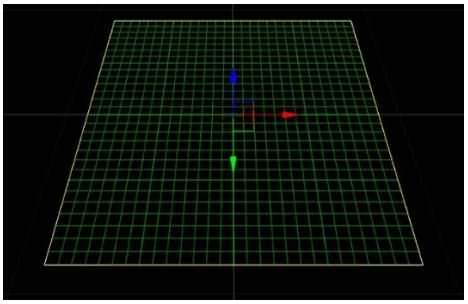

Рисунок 1. Сетка ландшафта

Компонент - самый большой элемент сетки. Каждый компонент требует ресурсов процессора для рендеринга, поэтому можно сделать ландшафт эффективнее за счет уменьшения количества используемых компонентов. При создании ландшафта у нас есть возможность настроить количество компонентов и секций в них. Unreal уменьшает качество отрисовки деталей при отдалении от секции для сохранения производительности на графическом процессоре. Большие секции не уменьшаются так же, как и маленькие, поэтому большая секция будет стоить дороже для графического процессора.

Задача состоит в подборе такой комбинации количества компонентов и размера секции, которая являлась бы оптимальной для баланса между процессором и графическим процессором (CPU и GPU). Хорошее решение состоит в том, чтобы сделать секции больше на больших ландшафтах и меньше на маленьких ландшафтах.

Overall size (vertices)	Quads / section	Sections / component	Component size	Total Components
8129x8129	127	4 (2x2)	254x254	1024 (32x32)
4033x4033	63	4 (2x2)	126x126	1024 (32x32)
2017x2017	63	4 (2x2)	126x126	256 (16x16)
1009x1009	63	4 (2x2)	126x126	64 (8x8)
1009x1009	63	1	63x63	256 (16x16)
505x505	63	4 (2x2)	126x126	16 (4x4)
505x505	63	1	63x63	64 (8x8)
253x253	63	4 (2x2)	126x126	4 (2x2)
253x253	63	1	63x63	16 (4x4)
127x127	63	4 (2x2)	126x126	1
127x127	63	1	63x63	4 (2x2)

Рисунок 2. Эффективное разбиение ландшафта.

Подробнее об инструментах работы с ландшафтом описано в НИРС.

2. Создание воды

При создании любого нового шейдера для начала необходимо найти достаточное количество референсов. Важно понять, как на самом деле выглядит то, что мы собираемся создать. Благодаря эталонным фотографиям можно определить какие свойства объекта являются ключевыми. Незначительные детали можно пропускать, потому что природа очень сложна в повторении мельчайших подробностей.

Рассмотрим создание собственного материала воды в Unreal Engine. Самое заметное в эталонных изображениях – это то, что вода имеет рябь на поверхности, которая имеет очень хаотичное движение. Также вода меняет цвет по мере того, как становится глубже. На мелководье она практически прозрачна, но чем больше высота столба жидкости, тем больше цвета она приобретает. Вода отражает окружающую среду, однако, это отражение зависит от угла, под которым мы смотрим на поверхность. Когда мы смотрим прямо вниз, мы видимо сквозь поверхность воды до дна, а когда горизонтально параллельно – рябь и волны создают очень интересные искажения. Вода преломляет свет и заставляет предметы на дне изменяться. Итак, выделим основные критерии будущего материала и реализуем его.

2.1 Реализация ряби на воде

В первую очередь необходимо найти карту нормалей для воды с достаточно крупным масштабом, чтобы избежать артефактов. Подключив карту нормалей к соответствующему узлу в корневом узлу материала получим близкий к необходимому результат. Чтобы сделать эту рябь подвижной воспользуемся встроенной в движок функцией «Panner», которая прокручивает текстуры. Для того, чтобы создать некоторый хаос в движении ряби, нужно продублировать уже созданное движение несколько раз, задавая различные направления движения. На рисунке 3 показан итоговый материал ряби на воде.

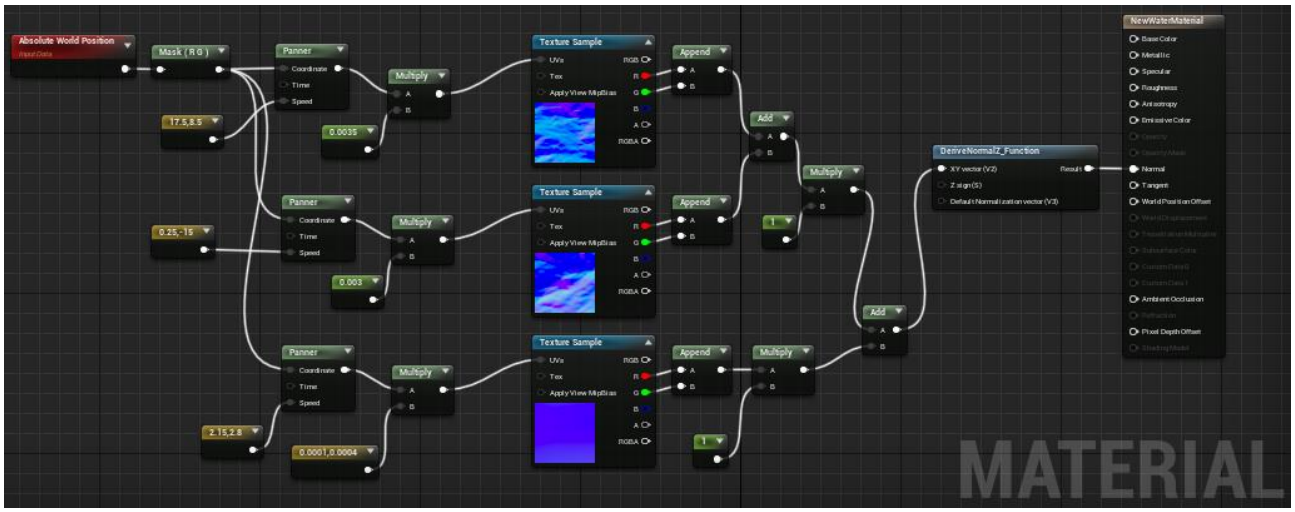

Рисунок 3. Материал ряби на воде

2.2 Реализация волн

Для реализации волн на поверхности воды будем использовать метод волн Герстнера. В компьютерной графике для реализации этого метода используется карта высот для дискретной сетки и на каждой итерации рендеринга для одной волны пересчитываются все три координаты для каждой точки поверхности согласно формулам:

$$\vec{x} = \overrightarrow{x_0} - \frac{\vec{k}}{k} A \sin\left(\vec{k} \cdot \overrightarrow{x_0} - \omega t\right),$$

$$y = A\cos(\vec{k} \cdot \overrightarrow{x_0} - \omega t)$$

Начальное положение точки на моделируемом участке — $\overrightarrow{x_0}$, а величина k обратно пропорциональна длине волны λ согласно формуле:

$$k = \frac{2\pi}{\lambda}$$

Векторная величина $\vec{k}$, называемая также «волновым вектором» имеет размерность R^2 и указывает направление движения воды на плоскости, параллельной плоскости дискретной сетки. Частота ω зависит от характеристики глубины моделируемой поверхности. При больших размерах глубины она выражается формулой параметра k, описанной в формуле:

$$\omega^2(k) = gk$$

Однако для глубины D, сравнимой с длиной волны, частота имеет вид:

$$\omega^2(k) = gkth(kD)$$

Из уравнения видно, что частота движения волны затухает при глубине D близкой к нулю, а при глубине D намного большей λ величина $th(kD)$ в предельном переходе обращается в единицу. Из-за того, что для волны Герстнера пересчитываются сразу все три координаты, такая волна «сгущает» точки к наиболее острым пикам поверхности, давая высокую точность триангуляции возвышенностей.

Перейдем к реализации волн Герстнера в UE. Обычно для изменения геометрии пространства необходимо использовать смещение мировой позиции. Но также можно работать с параметром «Tesselation», который регулирует разбиение пространства на полигоны. Для этого создадим математическую формулу, которая будет перемещать вершины геометрии вверх и вниз.

На рисунке 4 представлена реализация волн методом Герстнера в UE:

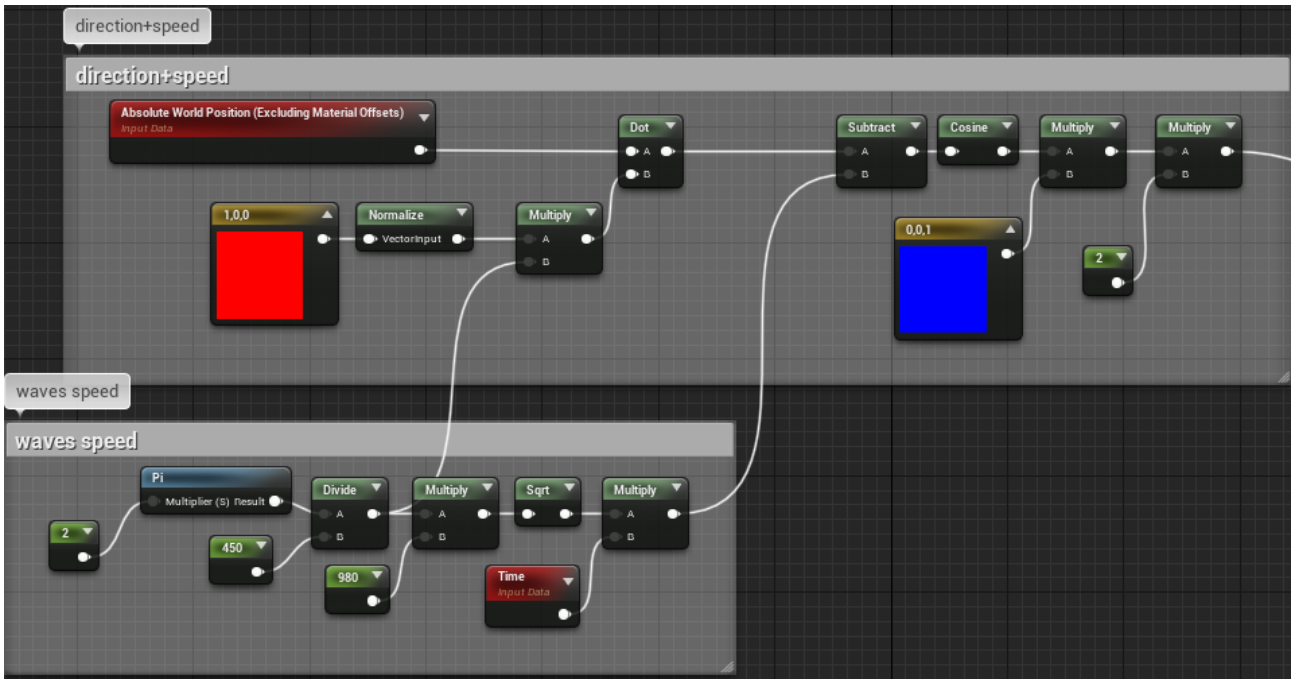

Рисунок 4. Метод Герстнера

2.3 Реализация пены в воде

Эффект пены в воде возникает там, где вода очень быстро сталкивается сама с собой или с другими объектами. Для создания эффекта пены в первую очередь понадобится текстура пены в формате TGA. Данная текстура представлена на рисунке 5.

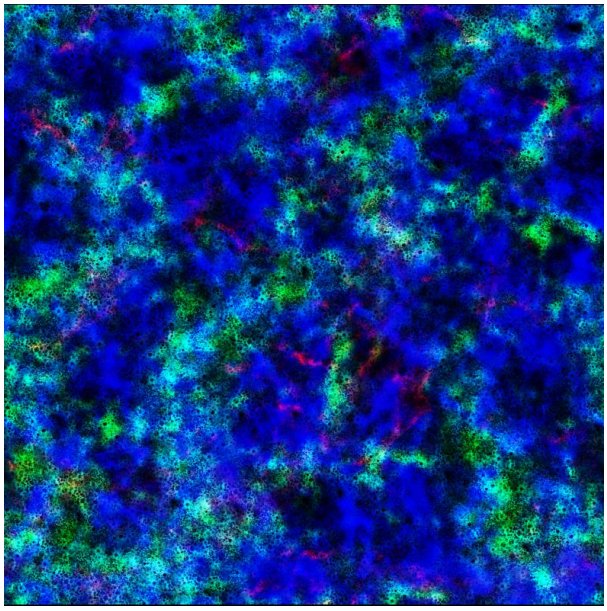

Рисунок 5. Текстура пены

В этой текстуре три канала (R,G,B) отвечают за три разные маски пены – тонкую, среднюю и толстую соответственно. Это позволяет нам сделать пену толстой в некоторых областях и тонкой в других. Позволяет осуществить правильную обработку изменения пены.

Также понадобится текстура градиента для того, чтобы контролировать, какую маску пены показывать в данный момент. Текстура градиента представлена на рисунке 6.

Рисунок 6. Текстура градиента

Эта текстура также разделена на три канала (R, G, B). Красный канал отвечает за то, где будет появляться тонкая пена. Тонкий слой пены будет появляться большую часть времени, а толстая совсем немного, что видно при разложении градиента на каналы цветов. Тонкая пена вступит в силу в начале, средняя – в середине, а толстая пена будет вносить свой вклад только в конце.

На рисунке 7 представлена функция добавления пены для общего материала воды. Для реализации хаотичного движения пены на воде в Blueprint используем узел “Motion_4WayChaos” встроенный в движок Unreal Engine. Узел “saturate” используется, чтобы поместить итоговое значение в диапазоне между 0 и 1.

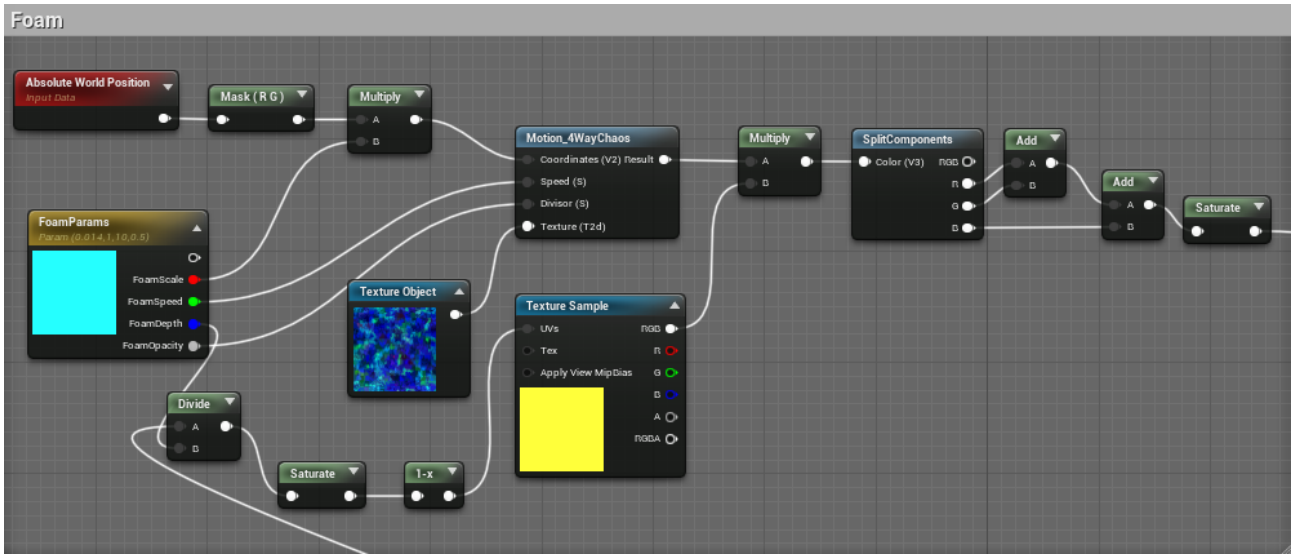

Рисунок 7. Материал пены

2.4 Реализация оттенка глубины воды

Под оттенком глубины воды подразумевается зависимость непрозрачности и цвета воды от толщины водного слоя.

Глубина воды необходима, потому что это один из параметров, контролирующих непрозрачность воды. Узлы «Scene depth» и «Pixel depth» позволяют измерять глубину. «Scene depth» измеряет расстояние от того места, где находится камера до дна через воду. В то время как «Pixel depth» измеряет расстояние до поверхности воды. Это показано на рисунке 8.

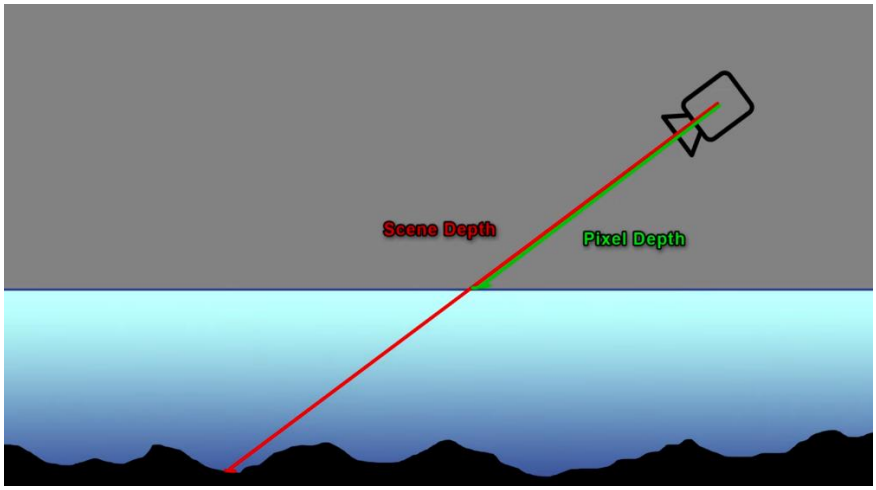

Рисунок 8. Узлы «Scene depth» и «Pixel depth»

На самом деле нас интересует величина от места соприкосновения луча камеру вертикально вниз до дна. Итоговый материал глубины воды представлен на рисунке 9.

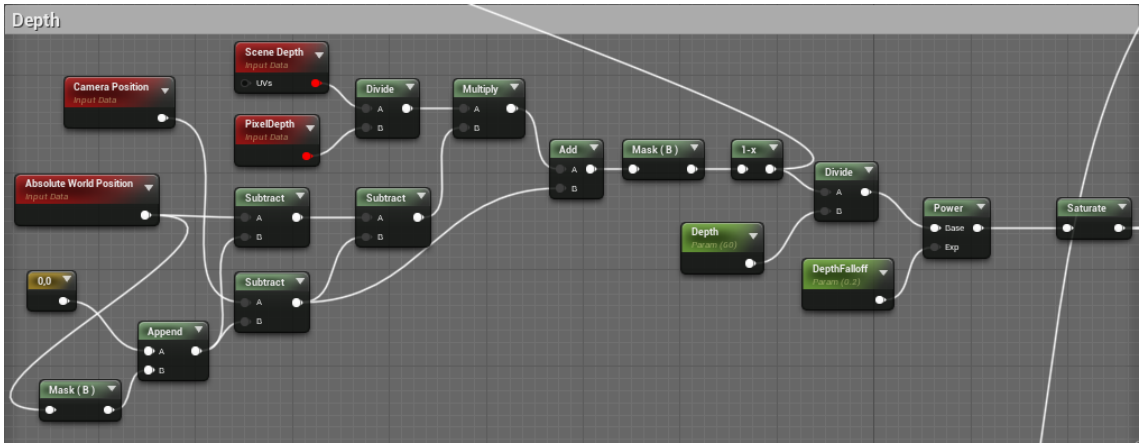

Рисунок 9. Материал глубины воды

2.5 Реализация отражения и преломления воды

Для создания отражений в UE4 может быть использовано несколько методов. Самый простой из них это SkyBox. Это наиболее простой метод создания отражения неба. Он работает сразу же из коробки. Однако, если мы хотим отражать локальные объекты, которые находятся ближе к камере необходимо использовать метод световых зондов. Для этого нам понадобится актер «Box Reflection Capture». Он может быть размещен на сцене практически без влияния на производительность, поскольку рассчитывается до запуска. Недостатком этого метода является то, что отражения выглядят верными только с точки захвата. Во всех остальных – искажается.

Перейдем к рассмотрению следующего метода, который представляет собой отражение пространства экрана. Для этого в корневом узле материала необходимо включить соответствующее свойство «Screen space reflection». Это свойство позволяет взять отображаемый объект на экране и построить отражение на его основе. Недостатком этого метода является то, что он может использовать только объекты, находящиеся в данный момент на экране. Это значит, что если камера будет направлена только на отражение объекта, а самого объекта на экране не будет, то и отражение не получится отрисовать. Также этот эффекта заметен по краям.

Unreal Engine использует все эти методы создания отражений одновременно, комбинируя их, выбирает наиболее подходящий метод.

Преломление в воде – это изменение направления луча на границе двух сред. Для того, чтобы вода выглядела еще более реалистично необходимо добавить этот эффект преломления. В UE это делает очень просто. Необходимо добавить константу с нужным значением преломления к соответствующему узлу корневого узла материала.

3. Освещение

Освещение в Unreal Engine 4 заключается в размещении направленных, небесных и точечных огней, а также в контроле их яркости, падении и цвета. Отличительной особенностью освещения в UE4 являются строгие ограничениями, чтобы сбалансировать качество и функции с производительностью. Стоит отметить, что освещение в движке трассировки лучей медленное, но обычно очень динамичное. Можно легко перемещать свет, модулируя цвет, чтобы имитировать заход солнца. Безупречный GI заполняет каждый уголок, поворот и щель богатым светом и тенью. За все эти преимущества необходимо платить временем. Даже простая сцена с несколькими базовыми примитивами занимает несколько секунд для визуализации.

Подвижный источник света (Moveable) – источник света, который можно перемещать. Такой тип источника света является одним из наиболее интенсивных эффектов производительности для визуализации, особенно если отбрасывает тени. Динамические тени имеют низкое разрешение и, обычно, очень четкие края в отличии от заранее рассчитанных статических теней.

Стационарный источник света (Stationary) использует статические и динамические пути освещения вместе, чтобы создать свет, который не может двигаться, но может отбрасывать динамические тени и модулировать их цвет и интенсивность во время выполнения. Тени от статических мешей запекаются в теневые карты с помощью Lightmass, но прямое освещение и тени от подвижных объектов вычисляются динамически. Stationary Lights создают идеальный солнечный свет в статически освещенных визуализациях.

Статический источник света (Static) – полностью статический источник, вообще не может двигаться или меняться в реальном времени. Вся информация об освещении и тени запекается в текстурах. Статические источники света используются исключительно системой Lightmass GI. Static Lights широко используются в архитектурной визуализации, где качество освещения важнее гибкости.

Общая структура неба и освещенности представлена на рисунке 10.

Рисунок 10. Общая структура неба и освещения

3.1 Динамическая смена дня и ночи

Первая задача, которая стоит при построении динамической смены дня и ночи - отслеживание времени. Для этого, в переменной time типа float будет храниться время в секундах. Важно понимать, как часто запускать таймер, чтобы с одной стороны не было резких рывков светила (при большой скорости), а с другой, чтобы излишне не нагружать процессор (если поставить таймер на 0.01, а скорость равной реальной - это излишняя нагрузка). Для этого создадим специальную функцию GetTimerRate, которая будет возвращать значение - как часто запускать таймер. В ней будем делить какую-то константу на скорость. Константу надо подбирать тестированием. Если взять константу равной 10 и если скорость будет 1000, то таймер должен будет запускаться 1000 раз в секунду, но так не может быть, для этого должно быть fps 1000, так не бывает. Для этого timer rate необходимо ограничить.

Встроенная структура time span позволяет автоматически разбить время из одной величины на разные (секунды в дни, часы, минуты, секунды), с помощью функции Make timespan. Эта функция принимает на вход обязательно целое число, поэтому наше внутреннее время необходимо для начала округлить при помощи узла truncate. Так же добавим функцию для отладки, которая будет выводить время на экран. У родительского класса BP_sky необходимо убрать тени (cast shadow), так как на нашей сцене объект неба будет очень большим и рассчитывать от него тени будет очень ресурсозатратной операцией. Это будет касаться всех типов неба, поэтому эту настройку необходимо отключить в родительском классе.

При реализации перехода между дневным и ночным небом для большей реалистичности будем не просто включать или выключать видимость той или иной сферы, а создадим функцию MF_Transition перехода времени суток. Для сглаживания переходного процесса используем узел lerp. сам процесс будет выполняться по кривой curve_SkyDay_Transition.

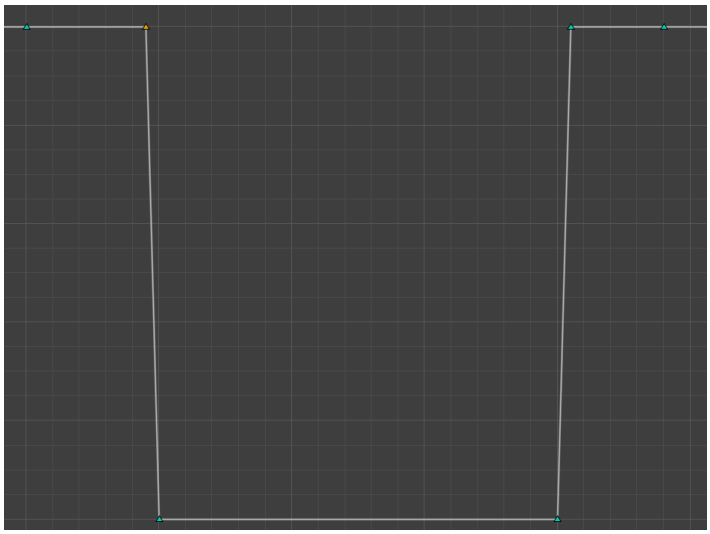

Рисунок 11. Кривая перехода от дневного к ночному небу SkyDay_Transition

Практически обратной будет кривая появления ночного неба. Следует заметить, что сначала полностью пропадает ночное небо и только после этого плавно начинает появляться дневное.

Для установки значения видимости также применим функцию nearly equal (примерно равно) для избегания проблем с багами или артефактами из-за неточностей с графиком. Обязательно убирать видимость при единице, т.к. ночная сфера больше дневной и находится за ней, а мы убираем механизм отсвечивания, а не становится прозрачным.

Реализация фаз луны была сделана с помощью инвертированного альфа канала материала луны с последующим умножением на реальный альфа канал материала. Проблема с черной полоской на луне возникает из-за неправильного сжатия текстуры программными средствами ue4. Для решения этой проблемы необходимо свойство mip gen settings переставить в режим nomipmaps.

Рисунок 12. Проблема с отображением луны

структура функций в BP_DayNightSystem

- sky
  - UpdateSkyDay - в зависимости от кривой прозрачности дневного неба обновляем объект skyDay
  - UpdateSkyNight - тоже самое для ночи
  - UpdateTransition - видимость сфер
  - ChangeTime - изменяем игровое время в зависимости от скорости от частоты обновления таймера
  - UpdateAll
  - UpdateSunDirection - по кривой curveDirLightPitch обновляем позицию объекта солнца
  - GetCurveValue - достает значение из кривой
  - UpdateMoon
- Info
  - GetTimerRate - значение обновление таймера от скорости
  - GetHour - остатком от деления и делением получаем часы
  - GetDay - остатком от деления получаем дни
  - GetTimespan - разбивает время из секунд на все остальные
- Debug
  - DisplayTime - выводит на экран игровое время (minimum integral digits = 2)

3.2 Создание пасмурной погоды

Пасмурное освещение рассеивает свет во всех направлениях и устраняет сильный свет и тени.

Для начала необходимо уменьшить интенсивность направленного света (DirectionalLight). Затем, увеличить его температуру в соответствии со шкалой для достижения более холодного эффекта.

На рисунке 13 изображена шкала цветовой температуры света.

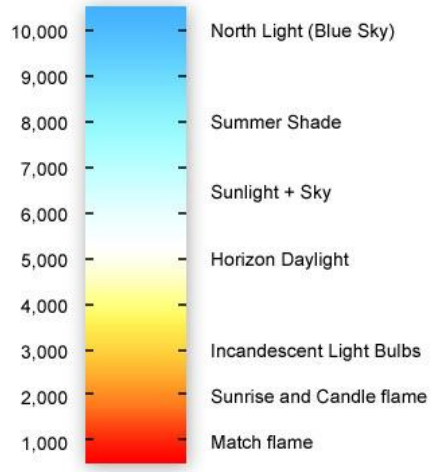

Рисунок 13. Шкала цветовой температуры света

Sky Atmosphere имеет два основных метода рассеяния:

- Рассеяние Ми: как свет взаимодействует с большими частицами в атмосфере (пыль, пыльца, загрязнение воздуха); обычно поглощает свет, вызывая неясность или прозрачность; часто рассеивается, образуя ореолы вокруг источника света; увеличение или уменьшение плотности Ми вызывает большую или меньшую ясность неба
- Рэлеевское рассеяние: как свет взаимодействует с меньшими частицами (молекулами воздуха); сильно зависит от освещения

И наконец, в BP_Sky_Sphere изменим параметр Cloud Speed на 3 для создания визуального эффекта движения облаков по небу.

3.3 Туман

UE4 оснащен системой атмосферного освещения рассеянного тумана, которая симулирует атмосферу земли. Если у Directional Light Actor включить Atmosphere Fog Light, он будет определять местонахождение солнечного диска и устанавливать цвета неба в зависимости от позиции солнца, создавая простой skybox.

Экспоненциальный туман высоты используется для создания эффектов тумана, таких как облака, но с плотностью, связанной с высотой тумана. Он визуализирует туман, плотность которого растет экспоненциально в зависимости от высоты и параметра спада. Переход плавный, поэтому при увеличении высоты вы никогда не получите жесткой отсечки. Метод объемного тумана (Volumetric Fog) вычисляет плотность участвующих сред и освещение в каждой точке усеченной пирамиды камеры, чтобы можно было поддерживать различную плотность и любое количество источников света, влияющих на туман.

Для большей реалистичности картинки дождливой погоды добавим на карту Exponential Height Fog и настроим его в соответствии с рисунком 14.

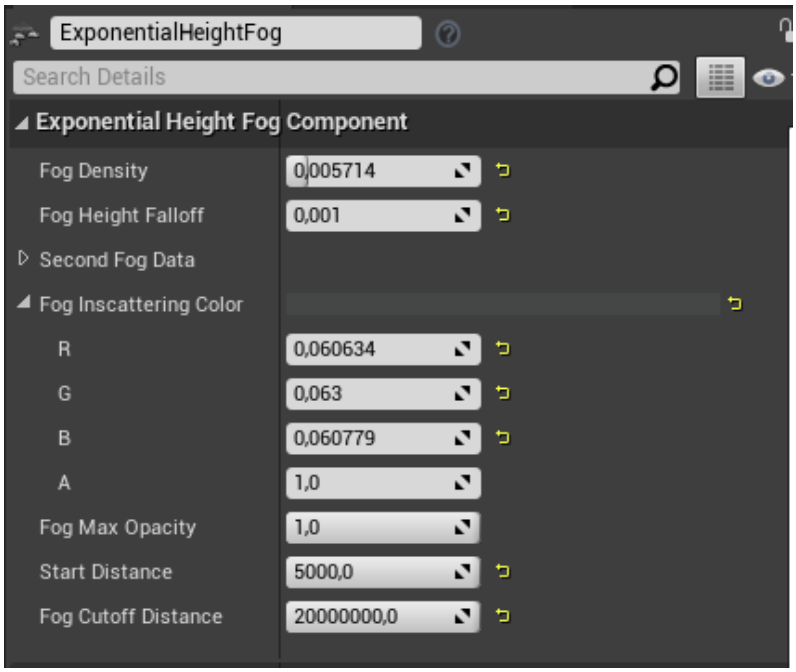

Рисунок 14. Параметры настройки тумана

4. Создание дождливой погоды

Для создания более реалистичного пейзажа города было принято решение воссоздать дождливую, пасмурную погоду. Создание пасмурной погоды описано в предыдущей главе. Ниже на рисунках 15 и 16 представлены пейзажи города с ясной и дождливой погодой.

Рисунок 15. Пейзаж города с ясной погодой

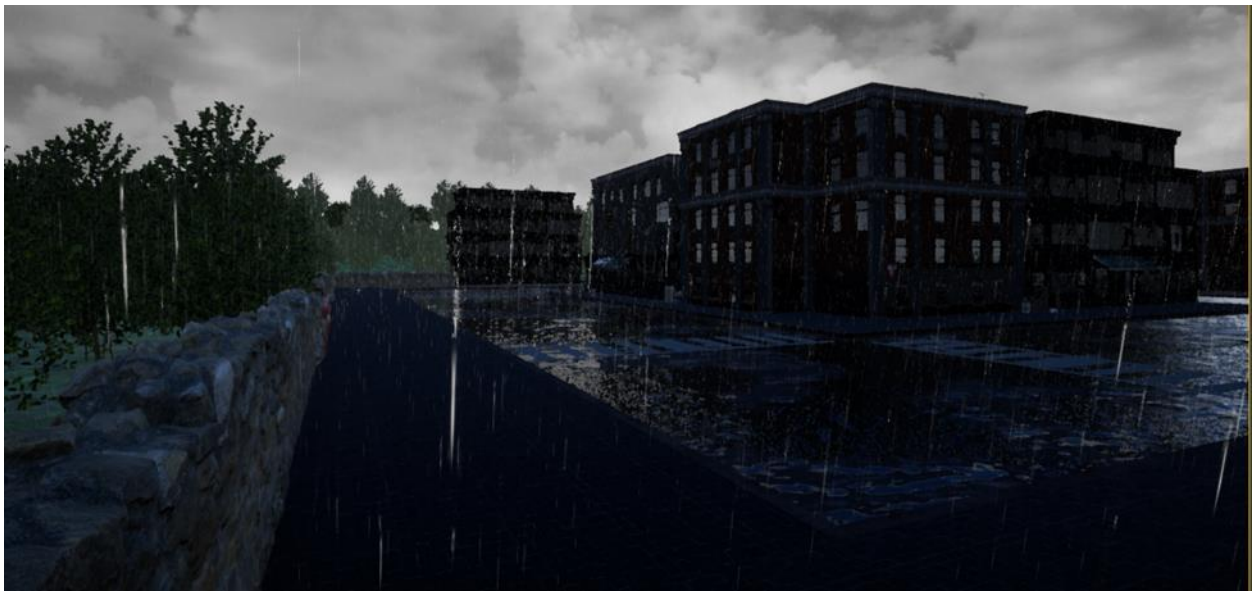

Рисунок 16. Пейзаж города с дождливой погодой

4.1 Создание луж дождя на асфальте

Для создания эффекта луж необходимо создать новый материал. За основу необходимо взять обычный материал дороги и наложить на него эффект воды.

Для этого сначала получаем значения черного и белого из канала Blue основной структуры. Через узел «OneMinus» инвертируем их. Подключение инвертированное значение и текстуру шума к узлу Add позволяет лужам рисоваться поверх материла. Для того, чтобы сделать площадь лужи регулируемой необходимо добавить узлы «Вычесть» и «Разделить» и сам параметр размера воды. Далее необходимо подключить узел «Multiply», чтобы сделать расположение лужи более четким. Подключим узел «Clamp», чтобы ограничить значение полученной области лужи. Затем подключаем узел «Lerp», чтобы смешать лужу с основной текстурой. Цвет воды зададим с помощью узла «Constant3Vector». Таким образом мы получаем новый материал с эффектом луж.

На рисунке 17 изображен итоговый материал асфальта с лужами.

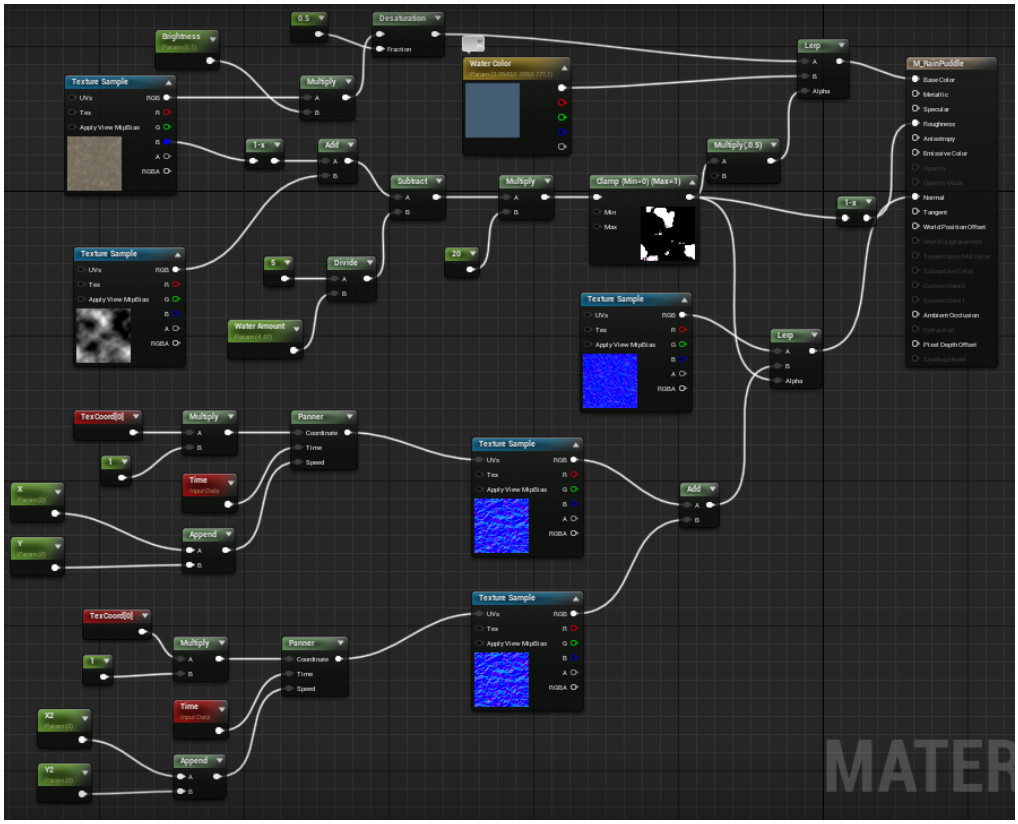

Рисунок 17. Материал асфальта с лужами

4.2 Создание капель дождя, падающих с неба

Во-первых, необходимо создать новый материал для применения его к частицам. В нем установить режим наложения в полупрозрачный и модель затенения в неосвещенную. Далее, создать узел радиального градиента, чтобы получить круглую форму. Изменим значение плотности. В результате получим круг с примененным градиентом.

На рисунке 18 изображен итоговый материал капли.

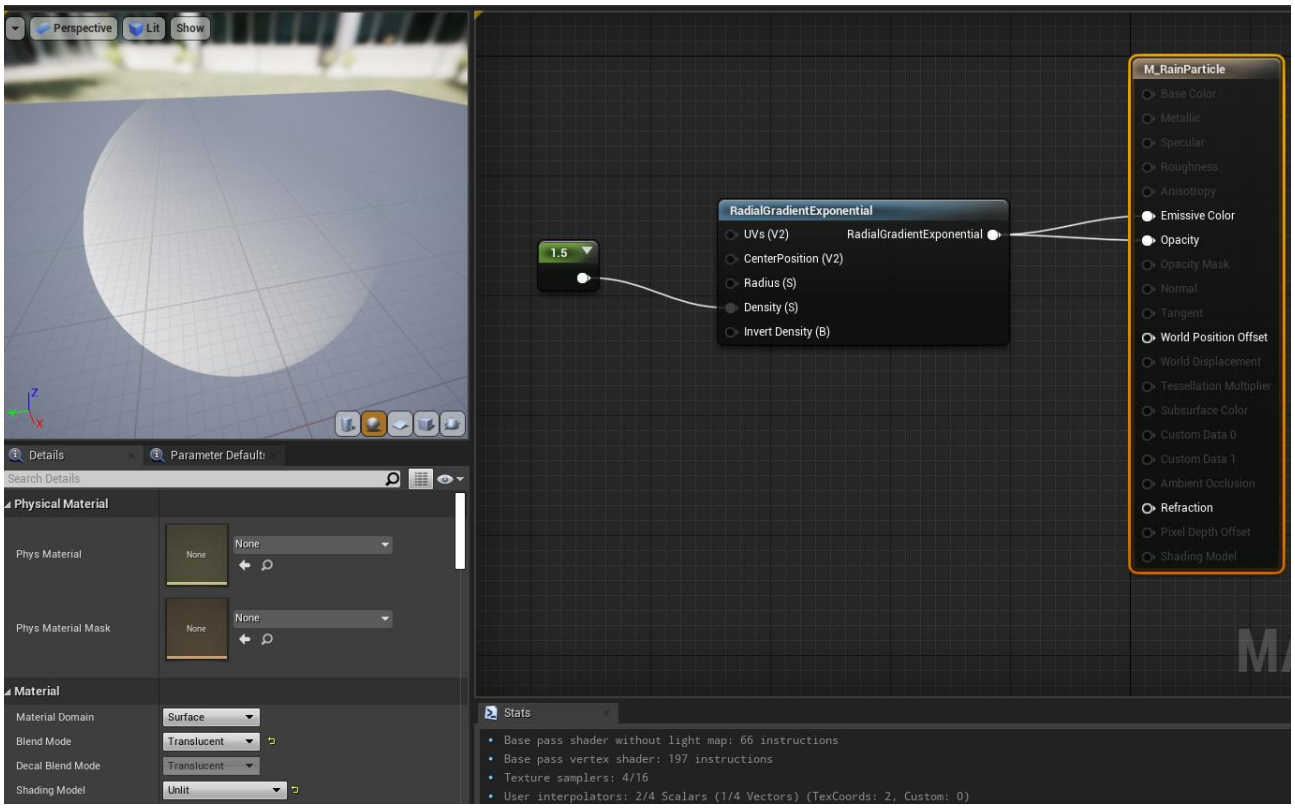

Рисунок 18. Материал капли

Следующим шагом создадим систему частиц (Particle System)

Внутри настраиваем:

- Материал
- Количество генерируемых капель (Spawn)
- Продолжительность жизни частицы (Lifetime)
- Размер частиц (Initial Size)
- Скорость частиц (Initial Velocity) – если задать z отрицательным, то частица будет падать вниз
- Область появления частиц (Initial Location)

- Установим для TypeData значение GPU Sprite, чтобы можно было обрабатывать большое количество частиц
- Сбросим граничную рамку, чтобы все частицы точно было видимы
- Добавим размер по скорости, чтобы капли растягивались (Size by speed)

В результате получим готовые к размещению на карте частицы дождя.

На рисунке 19 изображен итоговый материал системы частиц.

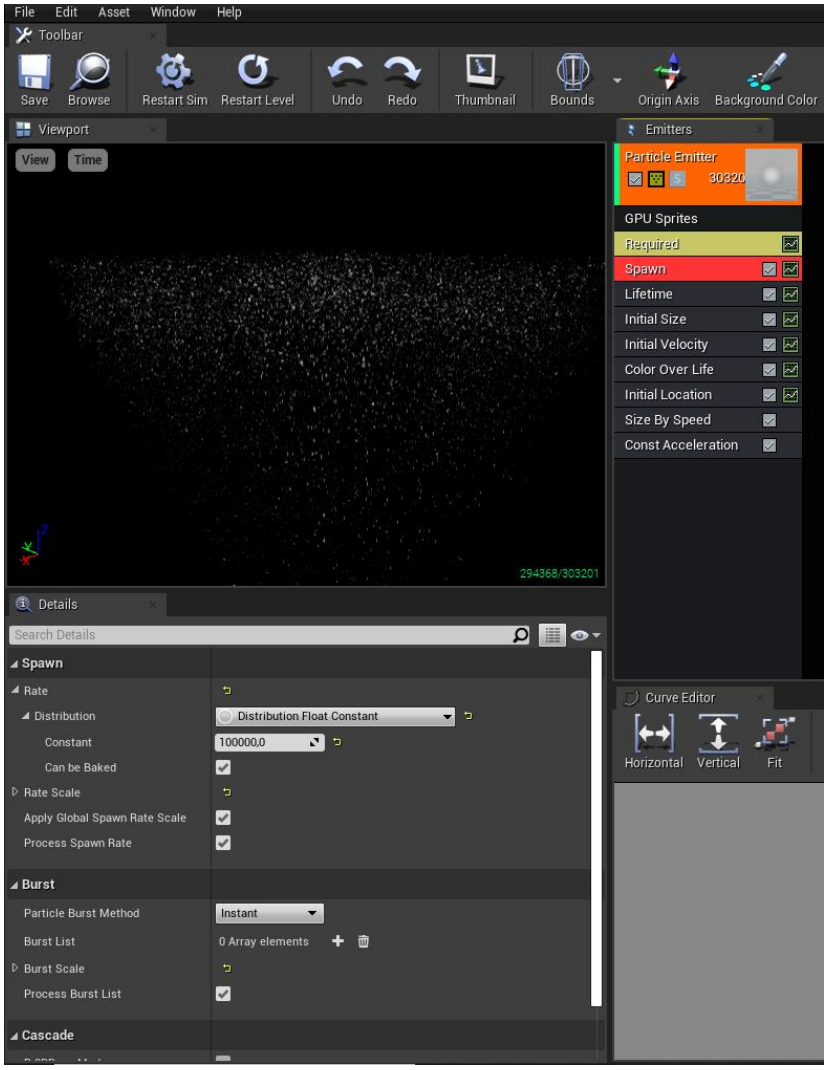

Рисунок 19. Материал системы частиц

4.3 Создание дождевой ряби на поверхности воды

Для создания эффекта ряби капель дождя достаточно немного изменить созданный материал с лужами на асфальте. Для реализации повторяющихся всплесков на воде понадобится узел «Flipbook» и основная структура ряби. Затем подключим различные параметры, чтобы итоговая рябь казалась правдоподобнее.

На рисунке 20 изображен итоговый материал ряби на поверхности воды.

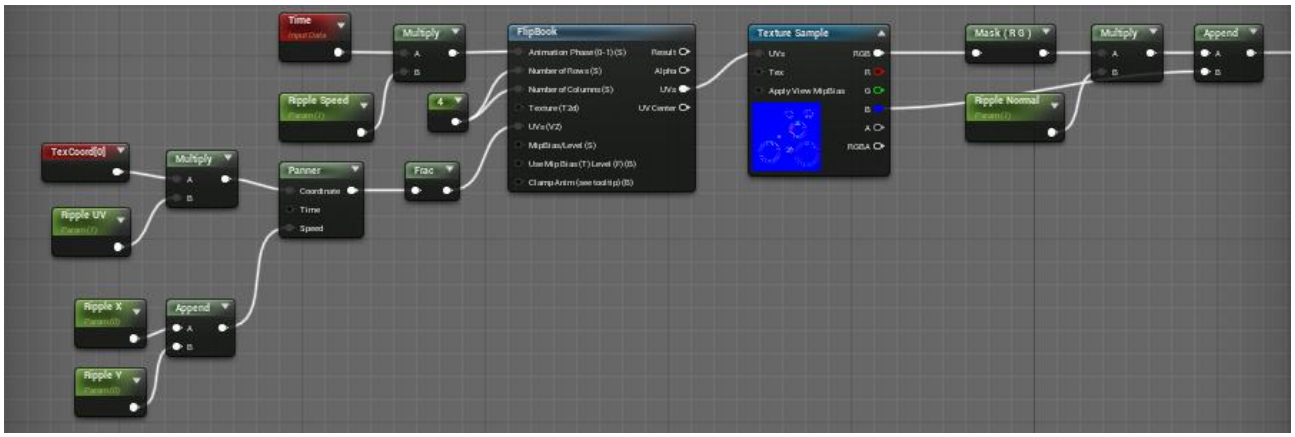

Рисунок 20. Материал ряби

Затем подключим полученный эффект ряби к нормали дождевых луж (Rain puddle normal).

5. Инструмент ue4 для анимированного движения камеры

Редактор Unreal Engine 4 содержит несколько инструментов для создания анимированного движения камеры - синематик и обычных кат-сцен. Данные инструменты позволяют настроить перемещение объектов, камер, частиц и персонажей, настроить постобработку, а также назначить события, которые будут срабатывать в Блупринтах, тем самым производя какое-либо дополнительное действие на сцене.

Все эти инструменты позволяют создать не только качественную кат-сцену, но и даже записать полноценный ролик, рекламу, мультик или небольшой короткометражный фильм.

Кат-сцена — это эпизод, который прерывает обычный ход геймплея, чаще всего отбирает у игрока управление и кинематографично излагает сюжет. Кат-сценами начинают, заканчивают игры, вводят новые квесты, развивают персонажей, заставляют игрока сопереживать им, включают его в атмосферу. По большей части кат-сцены служат для того, чтобы подключить игрока к событиям. Кроме того, кат-сцены используются и в более прикладных целях. Например, ими можно маскировать загрузочные экраны, чтобы игрок во время них не скучал.

Sequencer (секвенсор) - инструмент для создания последовательности кадров. На рисунке 21 показан интерфейс редактора sequencer, с помощью которого была создана кат-сцены.

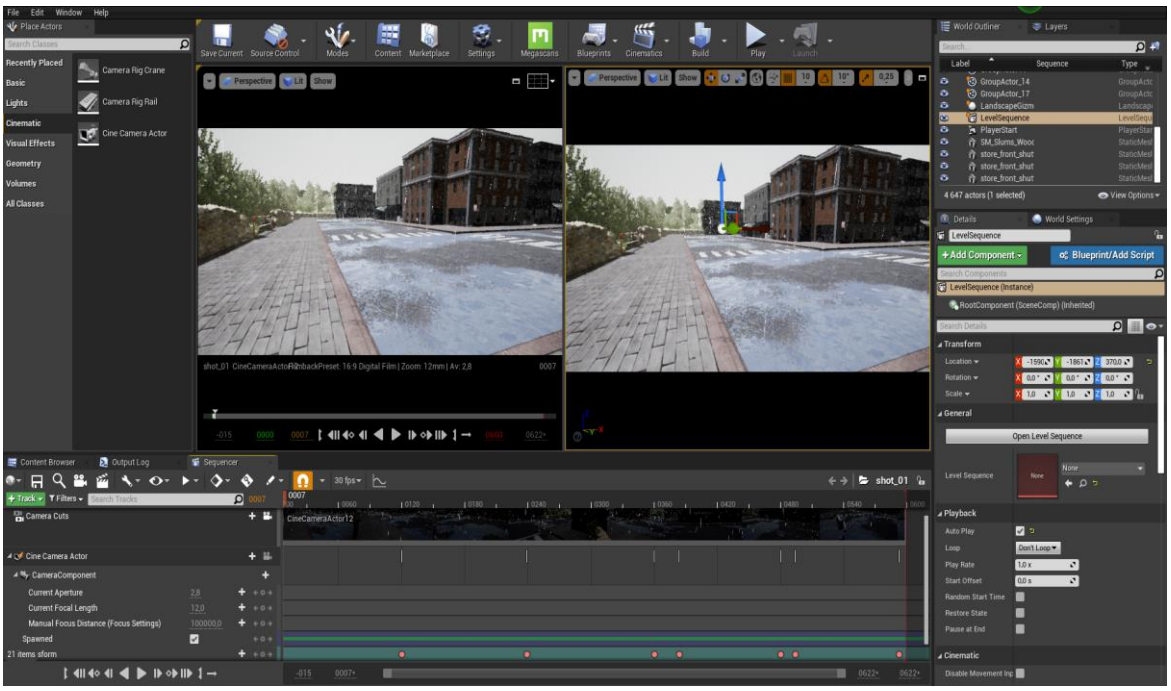

Рисунок 21. Интерфейс редактора sequencer

Итоговые кат-сцены пейзажей представлены в приложениях.

ЗАКЛЮЧЕНИЕ

В данной работе рассмотрены подходы для создания фотореалистичного пейзажа местности. Описаны основные компоненты, а также, рассмотрены принципы создания материалов. Рассмотрена реализация анимации движения солнца по небу и эффекта дождя. Рассмотрен способ создания материала водной поверхности.

В процессе работы выполнены следующие задачи:

1. созданы 3D-пейзажи нескольких придуманных местностей;
2. реализован материал воды с возможностью его настройки;
3. все пейзажи наполнены различной растительностью и другими 3D объектами, взятыми из бесплатных источников;
4. настроено индивидуальное освещение сцены с динамически изменяющимся временем суток;
5. реализована кат-сцена, демонстрирующая получившиеся пейзажи.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. 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: 11.02.2023;
2. Unreal Engine 4 Documentation // Unreal Engine Documentation URL: https://docs.unrealengine.com/. Дата обращения: 11.10.2022;
3. Display aspect ratio // Wikipedia, the free encyclopedia URL: https://en.wikipedia.org/wiki/Display_aspect_ratio/. Дата обращения: 05.11.2022;
4. Geometry instancing // Wikipedia, the free encyclopedia URL: https://en.wikipedia.org/wiki/Geometry_instancing. Дата обращения: 21.11.2022;
5. Свойства материалов // UEngine URL: https://uengine.ru/site-content/docs/materials-shaders/material-inputs/. Дата обращения 20.12.2022;
6. Волны Герстнера // DTF, URL: https://dtf.ru/gamedev/221392-volny-gerstnera-v-unreal-engine-4/. Дата обращения: 05.02.2023;
7. Landscape Technical Guide // Unreal Engine Documentation URL: https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/Landscape /TechnicalGuide/. Дата обращения: 15.02.2023.