Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Направление: 09.04.04 Программная инженерия

Профиль: Разработка программно-информационных систем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ТЕХНОЛОГИИ РЕНДЕРИНГА ВЫСОКОРЕАЛИСТИЧНОГО ОКРУЖЕНИЯ ВИРТУАЛЬНЫХ СРЕД И ПРОБЛЕМЫ ОПТИМИЗАЦИИ

Студент 2 курса

группы _____

«___»_____2019 г. Газизов Р.Р.

Научный руководитель

Доктор физ.-мат. наук, профессор

кафедры программной инженерии

«___»_____2019 г. Елизаров А.М.

Консультант

старший преподаватель

кафедры программной инженерии

«___»_____2019 г. Кугуракова В. В.

Директор Высшей школы ИТИС

«___»_____2019 г. Хасьянов А. Ф.

Казань – 2019 г.

Содержание

Введение

В современном мире нас окружает огромное количество объектов. Компьютерные игры, трехмерные симуляторы, анимационные видеоролики погружают нас в свой виртуальный мир. Для создания этого мира, следует создать и заполнить его большим количеством 3d-объектов. Однако большое количество объектов сильно нагружает сцену в игровом движке. Хорошая производительность критична для многих игр. Сцены могут содержать большое количество 3d-объектов с высоким разрешением текстур. Графическая часть игры нагружает в первую очередь две системы компьютера: GPU (графический процессор) и CPU (центральный процессор). Приходится решать проблему оптимизации сцен с большим количеством объектов в кадре.

При создании проекта определяется где происходит действие – в помещении или снаружи, происходит ли смена погоды, меняется ли время суток. Если мы просто видим окружающие нас предметы вдалеке, и не можем приблизиться к ним, можно заменить их высокореалистичным изображением. Также можно и заменить ландшафт.

Какими только возможностями ни обладают сегодня персональные компьютеры при установке на них соответствующих программных средств: они самостоятельно генерируют пароли и шифруют текст, пишут музыку и стихи, создают самые разные изображения и анимацию и даже могут помочь в рисовании вполне реалистических или, наоборот, фантастических пейзажей. Когда на экране современного фильма зритель видит удивительные по красоте пейзажи на заднем плане, он в большинстве случаев и не подозревает, что в действительности всей этой природы не существует, а то, что он наблюдает – всего лишь компьютерная декорация, искусно сделанная средствами трехмерной графики. Определить то, что горные кряжи, волны океанской поверхности и непроходимые джунгли являются лишь результатом просчета процессоров, порой трудно даже тем, кто профессионально работает в 3D.

Непосвященному же человеку такая работа кажется безумно сложной, и, он бы очень удивился, если бы узнал, что можно, с помощью специального программного обеспечения, создать виртуальный мир, такой реалистичный и правдоподобный.

Генераторы ландшафтов позволяют сравнительно быстро создавать фотореалистические земные или безжизненные, инопланетные пейзажи, по своей красоте и реалистичности неотличимые от настоящих фотоснимков и вполне достойные того, чтобы пополнить любую личную фотоколлекцию или стать фоном для дальнейшей работы. Искусственные ландшафты могут стать основой для разнообразных 3D-сцен в трехмерных играх, при подготовке телевизионных заставок и клипов. Также генераторы ландшафтов активно используют историки, палеонтологи и географы: первым и вторым они помогают понять, как выглядела наша планета в тот или иной исторический период, а при изучении географии могут быть образовательным инструментом, более интересным, чем обычные учебники. Даже в совершенно далеком от 3D-искусства бизнесе искусственные пейзажи тоже могут оказаться полезными, например, для наглядного представления расположения каких-то объектов компании.

Популярные решения для моделирования фотореалистичных и фантастических ландшафтов и создания искусственных миров программы - Bryce, Vue 5 Esprit, MojoWorld Pro, World Builder, VistaPro, Terragen.

Объектом исследования является процесс оптимизации сцен с большим количеством объектов в кадре.

Предметом исследования является использование технологий рендеринга.

Целью выполнения данной работы является рассмотрение различных технологий рендеринга высокореалистичного окружения виртуальных сред, решение проблем оптимизации для уменьшения времени получения итогового результата.

Постановка задачи

Для достижения поставленной цели были определены следующие задачи:

- Выявить основные проблемы оптимизации окружения;
- Рассмотреть подходы к оптимизации окружения;
- Описание собственного подхода к оптимизации окружения;
- Реализация окружения в виртуальных средах;
- Сравнительный анализ избранных подходов.

В ходе работы было решено использовать программный продукт для 3D моделирования, анимации, рендеринга и композитинга Autodesk Maya. Эта программа имеет огромный функционал и возможности, предоставляет удобный, понятный интерфейс. Для рендеринга окружения было решено использовать программу для создания окружения и ландшафта Terragen 4. Для создания и редактирования текстур решено использовать программы Substance Painter, Substance Designer и Adobe Photoshop.

Глава 1. Технологии рендеринга высокореалистичного окружения виртуальных сред

1.1 Обзор литературы

В [1] рассматривается создание карты рельефов для визуализации городской и природной среды. В этой статье предлагается рендеринг городской и природной среды с использованием параллакса и рельефного картографирования. Этот подход сочетает в себе преимущества рендеринга полигональных сеток и текстурного подхода. Таким образом, в предлагаемом подходе улучшается как качество с одной стороны и увеличивается скорость, с другой стороны. Применимость метода демонстрируется с помощью параллакса и рельефного картографирования в графическом движке с открытым исходным кодом Irrilicht. В результате проведены тесты для определения возможного использования параллакса и рельефного картографирования при отображении природных и городских условий. Сделан вывод о том, параллакс и рельефное картирование имеют потенциал в рендеринге естественной и городской местности на соответствующих расстояниях просмотра для представления городской и естественной среды.

В [2] рассматривается программная реализация трехмерной визуализации подводной среды с использованием карт высот через Windows Presentation Foundation. Описаны проблемы построения реалистичной трехмерной модели рельефа поверхности. Для построения трехмерной модели рельефа поверхности предложено использовать карты высот и библиотеку текстур. Создан оригинальный механизм редактирования 3D-моделей. Графическая библиотека была разработана для сокращения времени проектирования компьютерной графики.

В [3] рассматривается моделирование прогресса распространения пожара и виртуальном лесном массиве, а также представлена трехмерная система моделирования и визуализации распространения лесного пожара. Интегрированная среда построена на основе OntoPlant, набора виртуального

программного обеспечения, пожарного симулятора FARSITE, разработанного USDA в качестве механизма распространения огня и Open Scene Graph (OSG). Система отображает реалистичный трехмерный сценарий пожара с учетом таких факторов, как местность, тип растительности и погодные условия. Система нацелена на создание интегрированного и сбалансированного средства управления пожарами для повседневной работы. Результатом работы является система для 3D моделирования и визуализации распространения лесных пожаров.

1.2 Виртуальная среда в игровых движках

В наше время для разработки игрового проекта в быстрые сроки используют в основном игровые движки. Они включают в себя:

- физический движок;
- звук;
- систему скриптов;
- анимацию;
- искусственный интеллект;
- движок рендеринга.

Создав качественный игровой движок и используя его как основу для многих игровых проектов, разработчики экономят время и средства.

Компоненты любого игрового движка – графический и физический движки. Графический движок – программное обеспечение, задачей которого является визуализация 2D или 3D компьютерной графики. Графический движок в игровых движках работает в режиме реального времени. Так же и физический движок, который производит компьютерное моделирование физических законов реального мира в виртуальном пространстве трехмерной сцены с некоторой степенью аппроксимации. Физический движок позволяет наполнить виртуальное 3D пространство статическими и динамическими объектами, указать некие общие законы взаимодействия тел и среды в

пространстве, приближенные физическим, задавая характер и степень взаимодействия. Физический движок приближает физическую модель получаемой системы к реальной, передавая уточненные геометрические данные в графический движок [4].

1.3 Виртуальная среда в редакторах трехмерной графики и подходы к моделированию

Трехмерные редакторы применяют для создания трехмерного пространства. Это пространство пустое. В этом пространстве создаются различные 3д-объекты, которыми оно наполняется. Процесс создания трехмерных моделей называется моделированием. Также редактор позволяет создавать имитацию движения объекта, передавать симуляцию физического взаимодействия тел.

Каждый трехмерный редактор имеет свою специфику. В одних удобнее создавать модели как высокополигональные, так и низкополигональные, в других создавать анимации. Процесс моделирования так же отличается в разных программах. Существует следующие подходы: скульптурное, процедурное, твердотельное, полигональное, поверхностное, сплайновое, NURBS моделирование [5].

Скульптинг – технология 3д-моделировния, которое позволяет придавать мешу различную форму, подобно лепке из глины, увеличивать или уменьшать количество полигонов, добавлять части объекту. С помощью этой технологии можно создавать высокоточные, высокодетализированные трехмерные объекты с десятками и сотнями миллионов полигонов, передавая мельчайшие детали реальных объектов, что не получается достичь другими методами моделирования. Преимущественно скульптинг используется для моделирования персонажей, животных, одежды, обуви, реалистичного природного окружения, сложных конструкций, техники и других объектов с искривленными поверхностями. Метод наиболее предпочтителен для

получения фотореалистичных сцен. Сейчас высокополигональные модели широко применяются в производстве кинофильмов, медицине (хирургия), промышленном дизайне, искусстве, 3d-печати. Наиболее популярные программы для скульптинга: ZBrush, Blender.

Твердотельный подход к моделированию позволяет работать с оболочками тел, а не с отдельными поверхностями. При разрезании тела внутри не будет пустоты, как при разрезании реального твердого тела. Этот метод моделирования позволяет создавать простые геометрические фигуры и применять к ним различные операции: булевы операции, разрезание, объединение с другими телами. Метод твердотельного моделирования отлично подходит для создания моделей формы: рамы, шестеренок, зданий, деталей двигателя и т.д. Не применяется при создании персонажей, животных, одежды, мятых тел и т.д. Такой метод моделирования есть в любой Системе Автоматизированного Проектирования (CAD-системы), которые используются не только для получения визуального образа объекта, но и для получения измеримой, рабочей информации будущего изделия.

Поверхностное моделирование позволяет создавать и настраивать определенным образом поверхности, которые описывают отдельные части моделируемого объекта. Поверхностям придают нужную форму и соединяют между собой, лишнее обрезают. Таким образом строится модель из различных поверхностей. Этот метод моделирования используется для создания таких сложных моделей как детали самолетов, автомобилей, детали для станков с ЧПУ. Твердотельное и поверхностное моделирование используются для создания моделей промышленного назначения.

Полигональное моделирование позволяет создать объект любой сложности при помощи настройки вершин, ребер и граней. Полигон состоит из граней, четырехугольных или треугольных. Моделирование происходит путем вытягивания, вращения и перемещения полигонов в пространстве. По сложности модели делятся на низкополигональные и высокополигональные. Чем больше полигонов, тем точнее и более детализированная модель. Однако

такая модель будет занимать больше объема памяти, а это влияет на производительность. Использование той или иной модели зависит от ее предназначения. Высокополигональные модели используются в основном для получения фотореалистичной картинки. Низкополигональное моделирование используется для создания моделей с небольшим количеством полигонов. Такие модели создаются, когда не требуется высокая детализация и для экономии ресурсов. Они преимущественно используются в сфере компьютерных игр. Персонажи, предметы окружения в виртуальных мирах игр создаются с помощью полигонального моделирования. Одна из самых популярных программ для создания полигональных моделей – Autodesk Maya.

Сплайновое и NURBS моделирование дают возможность создавать модели при помощи специальных кривых линий – сплайнов или B-сплайнов. Нужная форма объекта строится по кривым линиям и соединяется все полигонами. Объекты получаются плавной формы. Метод используется для моделирования растений, животных и других моделей гладкой формы.

Процедурное моделирование позволяет проектировать с помощью специального алгоритма 3d модели высокой сложности – машины, механизмы, шерсть, ландшафт, трава и др. При таком моделировании количество и расположение отдельных вершин, граней, ребер задается с использованием определенных процедур и зависимостей. Один из самых популярных пакетов для процедурного моделирования – Side Effects Houdini.

Для ускорения процесса моделирования используются разные методы совместно. Так получения качественной и реалистичной низкополигональной модели для игр, следует для начала создать ее высокополигональную модель для получения карты нормалей. Это делается для прорисовывания мелких деталей, которые можно не моделировать для низкополигональной модели.

1.4 Фотореалистичное окружение в различных виртуальных средах

Виртуальное окружение играет очень важную роль в компьютерных играх, анимационных видеороликах, симуляторах, в фильмах. Человек способен быстро воспринимать большое количество данных, которые содержатся в окружении. Фоновое восприятие работает намного быстрее, чем восприятие речи или текста, и это позволяет получать некую информацию, при помощи анализа деталей и объектов окружения [6]. Так, детали могут рассказать, что происходило в мире, предупреждать о чем-либо, объяснять обстановку, а также создать запоминающий образ. Например, грамотно расположив объекты и детали, можно создать пространство, в котором человек будет чувствовать себя комфортно или наоборот беспокойно. Окружение является мощнейшим инструментом в формировании и передачи атмосферы и настроения, возникающее у человека, при попадании в какую-либо обстановку и также усиления эффекта погружения. Элементами окружения являются такие объекты как ландшафт, растительность, архитектурные строения, природные объекты, объекты интерьера т.д. Также к ним можно отнести свет, цветовой получаемой картинки.

Окружение может быть разного стиля: реалистичное, мультипликационное, выдуманное (см. рис. 1).

Рисунок 1 - Окружение мультипликационного стиля

Реалистичный стиль наиболее точно и объективно передает реальные черты окружения. В мультипликационном - происходит иллюзия оживления созданным человеком объектов, и придания им характеристик деталей предметно-реального мира.

Превращение обычных объектов окружения в часть игрового виртуального мира начинается с поиска референсов, создания концепт-арта. Эта примерное изображение будущего результата. Далее происходит создание трехмерных моделей всех необходимых элементов. Часть предметов можно создать на основе одного меша, другую часть сделать модульными, чтобы объекты складывать из отдельных составных частей. Так, например, дома состоят из дверей, окон, стен, ставней, крыши и т.д. Определяется наиболее оптимальное количество полигонов на 3д моделях. Использование запекания нормалей позволяет снизить количество полигонов. Создается оптимальная uv-развертка моделей. Использование большого количества текстур влияет на производительность, следовательно, можно создать многоразовые трафареты и использовать текстуры с тайлингом. Это позволит сократить количество текстур. При текстурировании используется карты Diffuse, Specular, Normal, Displacement, Gloss, Detail для более фотореалистичного результата. Далее настраивается освещение и камера.

В игровых движках рендеринг происходит в настоящем времени. Используются все разнообразие источников света для создания общего света на сцене, создания эффекта отблеска от поверхностей и т.д. Свет, который просчитывается в реальном времени выдает картинку лучшего качества, но при запекании света производительность значительно увеличивается. Для каждого объекта освещение запекается в отдельную текстуру, которая размещается в lightmap-атласе.

В 3d-редакторах и генераторах ландшафта рендеринг происходит в не реальном времени (пре-рендеринг). Он используется для получения фотореалистичных изображений с корректным наложением света и тени, добавлением цвета. Главный фактор у такого рендеринга получить высокое

качество изображения. Скорость просчета зависит от сложности сцены. Настраивается положение камеры, свет, тени для получения фотореалистичной картинки.

Сложности окружения влияет на производительность сцены как в игровом движке, так и в 3d редакторах. Большое количество моделей, полигонов, источников света, использование текстур с высоким разрешением значительно сокращают время от рисовки сцены.

1.5 Создание окружения процедурными методами

Существует различные способы создания окружения. Довольно часто окружение создается процедурными методами. Происходит наложение моделей и эффектов, с возможностью динамически настраивать параметры каждого эффекта. Этот метод преимущественно используется для получения красивых фоновых изображений.

Программы генераторы 3d ландшафтов и окружения используются для создания реалистичных ландшафтов, окружения, полноценных миров с последующей анимацией и фотореалистичным рендерингом [7]. К ним относятся такие программные пакеты как Microsoft Virtual Worlds, WorldToolKit for Windows, Virtus Walkthrough Pro, Blade Engine, Novelty Visual Novel Maker, Terragen, NREAM 3D, Bryce 5.5, Vue 5 Esprit, MojoWorld 3.1, World Builder 4.2, VistaPro 4.2.7, GenesisIV 6.

Создание окружения начинается с создания ландшафта, рельефа. Настройка происходит путем добавления на поверхность помех, шумов, остроконечных пятен, то есть происходит добавление эффектов на карту неровностей. Так же есть возможность загрузить реальные карты с данными о ландшафте, или нарисовать собственную карту.

Создание водной глади – рек, озер, океанов происходит схожим образом. Определяется и задается высота и ширина волн, уровень воды. Добавляется

визуальный эффект отражения от поверхности воды. Также есть возможность импорта трехмерных объектов, созданных в 3d редакторах.

Добавляются на сцену атмосфера и облака. Настраивается возникновение облаков, их размер, цвет, плотность, высота. Цвет атмосферы влияет на общий цветовой тон сцены. Источники света – солнце используется для освещения сцены. Его можно перенастроить и получить луну для ночных сцен.

Элементами управления камерой можно настроить камеру в нужной позиции. Настройки параметров рендеринга позволяют выбрать необходимый размер, качество и сглаживание для изображения.

Преимущество этой технологии рендеринга в том, что он позволяет производить рендеринг целых планет, огромного количества объектов – деревьев, травы, используя только копии виртуальных объектов, что значительно сокращает время визуализации. Рендеринг эффектов тумана и облаков так же занимает меньше времени чем визуализация в 3d редакторах.

1.6 Обзор инструментов для создания окружения

1.6.1 Terragen

Все настоящие объекты, которые нас окружают, имеют неоднородную, неповторяющуюся структуру. Несмотря на похожую форму, каждая травинка, облако или камень уникальны. В действительности встретить абсолютно точную копию большинства предметов, сделанных природой, невозможно – объекты будут отличаться как по форме, так и по своей окраске. Поэтому для трехмерной модели природного ландшафта также должно соблюдаться условие неоднородности. Изображения, которые получаются после финальной части работы над проектом в Terragen, удивляют своей реалистичностью. Визуализация реалистичных картин с помощью генератора природных ландшафтов возможна благодаря использованию фрактальных

алгоритмов просчета изображения. Именно этот метод построения математической модели лежит в основе большинства генераторов ландшафтов, в том числе и Terragen [7].

Работа в программе поделена на отдельные этапы – моделирование гор, создание водной глади, добавление в сцену атмосферы, создание виртуальных камер и т.д. При создании большинства объектов в Terragen можно использовать отдельные слои, комбинируя тем самым разные свойства создаваемых объектов сцены. Например, можно сделать несколько слоев облачности, и задать им разные параметры анимации, что позволит получить на выходе реалистичную картину движения облаков. Ключевые значения анимированных параметров обозначаются в программе красным цветом.

Окно предварительного просмотра не дает полной информации о конечной сцене. Схематически отображенный результат сильно упрощен, поэтому для выбора правильных параметров трехмерного ландшафта приходится руководствоваться большей частью интуицией, особенно при первом знакомстве с Terragen.

Стандартная версия программы имеет ограниченные возможности создания анимации и не позволяет «выращивать» растительность, поэтому виртуальные миры, генерируемые приложением, выглядят несколько безжизненно. Главное из достоинств программы - Terragen успешное использование не только для создания красивых полиграфических макетов, но и для производства видео.

Для рендеринга программа Terragen использует два разных метода рендеринга. Это метод трассировки лучей и микрополигональный метод. Микрополигональный рендеринг – это метод, который берет некоторую поверхность и разбивает ее на маленький полигоны, которые меньше пикселей на визуализированном изображении. Эти микрополигоны затем смещаются и затеняются. Затенение - это в основном процесс расчета цвета для микрополигона с учетом таких факторов, как цвет поверхности и освещение. Микрополигональный рендеринг особенно подходит для рендеринга

процедурных данных. Процедурные данные - это то, что рассчитывается по мере необходимости с использованием математических формул. Пример, процедурных данных - фрактал. Фрактал можно увеличивать почти бесконечно. Это потому, что фрактал генерируется математически, и при каждом увеличении масштаба, фрактал вычисляется заново.

Процедурные данные позволяют программе визуализировать сложные поверхности с очень высоким уровнем детализации без необходимости хранить массу данных. Микрополигональный рендеринг подходит для рендеринга процедурных данных, потому что он помогает ограничить объем визуализации.

Ray Tracing - другая техника рендеринга, используемая программой Terragen. Он работает, проецируя линии или лучи на сцену. Когда один из этих лучей попадает на элемент сцены, средство визуализации вычисляет затенение элемента сцены. Лучи могут также собирать информацию о затенении при прохождении через сцену, например, при прохождении через облака.

1.6.2 Unity

Unity - это игровой движок, позволяющий создавать игры под большинство популярных платформ. С помощью данного движка разрабатываются игры, запускающиеся на персональных компьютерах, на смартфонах и планшетах на игровых консолях. Эти устройства работают на разных операционных системах: Windows, Linux, Android, IOS [8].

Игровой движок очень популярен среди простых разработчиков, стартаперов и профессионалов. Причин его популярности несколько:

- Список поддерживаемых платформ, где может запускаться приложение;
- Доступная ценовая политика;
- Интуитивно понятный интерфейс редактора и использование простых в освоении языков программирования: C# и JavaScript;

- Распространение этого движка среди разработчиков. Здесь играет свою роль тот факт, что продукт очень качественный и уникальный. Почти каждый юный разработчик выкладывает прототип своей игры на форум или файлообменник;
- Простота разработки приложений для мобильных устройств. На движке Unity разрабатывается просто огромное количество игр под мобильные платформы.

Unity хороший выбор для создания средних и сложности проектов как для ПК так и для мобильных устройств. Этому помогает большое количество готовых ассетов, скриптов. Если же создавать проект трипэл эй(AAA) класса, то Unity может и не подойти, так как использование скриптов будет очень медленно работать. Использование интерпретируемого языка C# не подойдет. Итак, из недостатков можно отметить медленную работу, в сравнении с движками LibGDX, Cocos2D-X, большой размер приложений, сложная плохая архитектура в приложениях.

1.6.3 Autodesk Maya

Программа для 3D-анимации, моделирования и визуализации Maya предоставляет мощный интегрированный инструментарий, который можно использовать для создания анимации, сред, графики движения, виртуальной реальности и персонажей.

Возможности:

- полный набор инструментов для NURBS- и полигонального моделирования;
- мощные средства общей и персонажной анимации;
- развитая система частиц;
- технология Maya Fur (создание меха, волос, травы);

- технология Maya Fluid Effects (моделирование жидкостей, атмосферы);
- динамика твердых и мягких тел;
- широкий набор средств создания динамических спецэффектов;
- UV-текстуры, нормали и цветовое кодирование;
- многопроцессорный гибкий рендеринг.

К плюсам можно отнести огромный функционал и возможности.

Минусы**:** длительное и сложное обучение, высокие требования к системе, высокая цена.

1.6.4 Substance Painter

Продукт для художников по текстурам Substance Painter используется для раскрашивания 3D моделей с большим количеством мощных функций и улучшенным рабочим процессом, позволяющее сделать создание текстур для 3D моделей проще.

К ключевым возможностям программы относятся: полная интерактивность, мощный движок визуализации виртуального пространства, кисти на основе систем частиц, применение любого шейдера для визуализации, эффекты и кисти Substance и многое другое.

Substance Painter дает возможность рисовать не только кисточкой по 3D модельке, но и воздействовать на текстуру, которая уже наложена на 3D модель разными эффектами с частицами: сварка, взрывы, расползающаяся ржавчина и т.д. Также к плюсам можно отнести автоподключение запечённых карт (curvature, AO и т.д.), поддержку мульти-материала, что позволяет назначать разным частям меша разный материал (т.е. увеличивается тексельная плотность. Тексельная плотность показывает отношение количества пикселей к единице площади 3D модели. Другими словами, теперь элемент на текстуре буде занимать больше места, а значит текстура для 3D

модели будет большего разрешения, зеркальное и симметричное рисование текстур.

Глава 2. Основные проблемы оптимизации окружения

Проблема оптимизации окружения очень актуальна в современное время. Довольно часто разработчики сталкиваются с низкой производительностью движка или 3d редактора.

2.1 Проблема большого количества объектов на сцене

Создание окружения начинается с подбора и сбора референсов. Источником референсов может являться интернет пространство. Также получение вспомогательного изображения самостоятельно с помощью фотоаппарата. Если объект выдуманный и его не существует, создается концепт художником. Для передачи максимальной реалистичности объекта копируется его каждая деталь. Но иногда приходится жертвовать правдоподобностью в угоду оптимизации и игровым условностям (см. рис. 2).

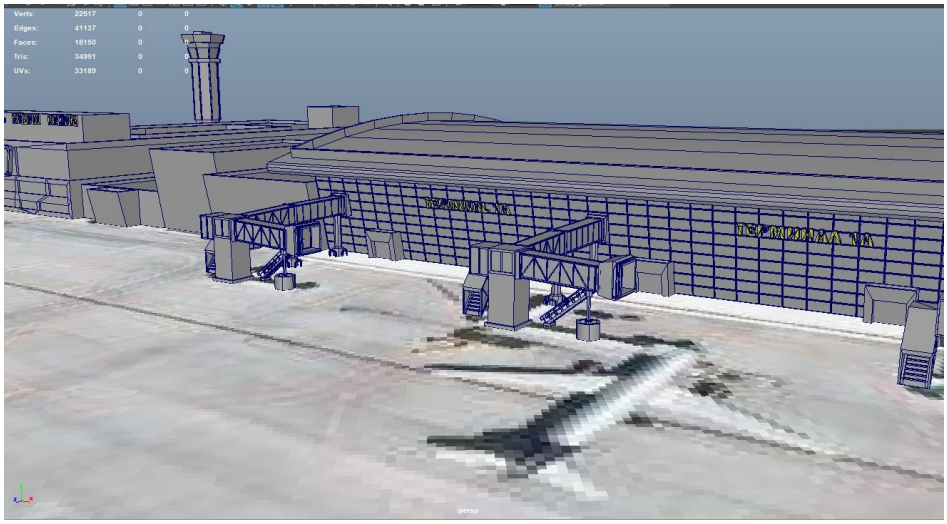

Рисунок 2 - Упрощенная модель здания аэропорта

Имея представление о том, как будет выглядеть будущий объект, создается его 3d модель. Высокополигональная модель максимально правдоподобно передает форму объекта за счет большого количества полигонов и нормалей. Итак, использование высокополигональных 3д моделей одна из проблем загруженности сцены.

Часто бывает так, что сцена содержит только низкополигональные модели, но количество этих объектов довольно большое. Так, например, сцена с природными объектами содержит большое количество однотипных объектов: трава, камни, деревья.

2.2 Не оптимальная UV-развертка и проблема текселерации

После создания 3d модели создается его UV-развертка. Здесь может возникнуть проблема с тем, что на объекте текстура будет выглядеть мутно, хотя изображение с высоким качеством.

Тексель - элемент текстуры, «точка». Текстура состоит из массива текселей. Тексель может представлять собой цветную точку в изображении. Texel density (текселерация, тексель (сокращенно, в контексте)) — это величина отношения размера текстуры (в пикселях) к размерам 3d модели на сцене. Texel density определяет плотность, «качество» текстуры, в общем смысле. Таким образом, текселерация выявляет какая площадь текстуры будет отдана на модель исходя из ее размеров [9]. Высокий texel density означает высокую детализацию текстуре, низкий — меньшую, размытую текстуру (см. рис. 3).

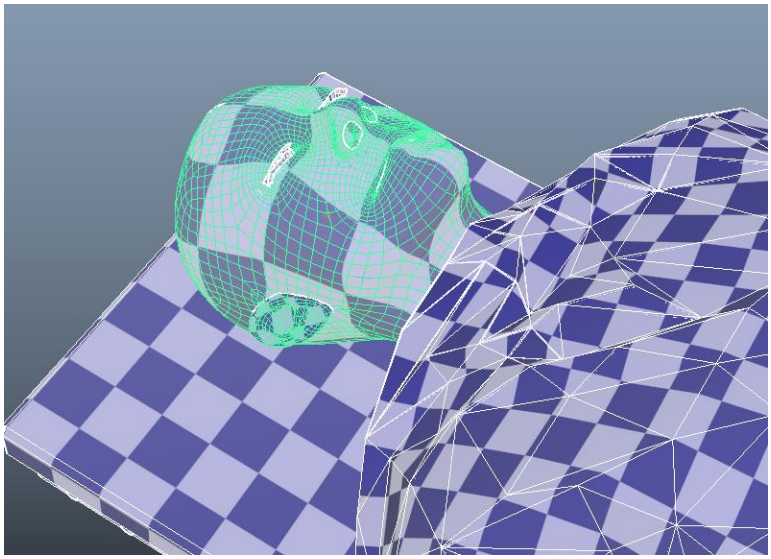

Рисунок 3 - Пример низкой текселерации на 3d модели головы

Окружение имеет большое количество объектов и использование большого количества текстур существенно затормаживает работу. Например, в игре есть деревянный дом, в котором 20 деревянных предметов (столы, стулья, ложки). У каждого объекта своя текстура. В итоге CPU будет 20 раз обращаться к GPU, требуя отрендерить каждый объект по отдельности.

2.3 Сложность реализации прозрачных и отражающих объектов

Отрисовка получаемой картины сцены происходит методами рендеринга: Ray casting, Ray tracing (трассировка лучей), Path tracing (трассировка пути), растеризация. Для получения достаточно качественного и фотореалистичного изображения за определенные затраты вычислительных ресурсов программами используются эти техники рендеринга. Передовые программные обеспечения – игровые движки, 3d редакторы, и генераторы ландшафтов используют методики трассировки лучей и трассировки пути. В компьютерных играх метод трассировки лучей является решением для создания реалистичного освещения, отражения, теней.

Метод Path tracing (трассировка пути), основанный на оптике. отрисовывает сцену виртуальной реальности. В трехмерном пространстве из источника света испускается очень много лучей и прослеживается история пути каждого луча. При встрече луча с преградой, возможно: поглощение, отражение или преломление. Вероятность каждого из них зависит от материала преграды и цвета луча. Лучи, которые попали в камеру рисуются на экране. В оптике время изотропно, это означает, что оба направления времени равноправны. Получается, можно следить за лучами не от источника к глазу, а от глаза к источнику. Это практичнее, path tracing и так очень затратный и ресурсоемким, нельзя тратить время на трассировку лучей, в которых нет шанса быть нарисованными [10].

Ray tracing (трассировка лучей) – другой метод рендеринга, основанный на оптике. Представим, что есть камера. Также есть плоскость

проектирования, которая является набором пикселей. От камеры, к каждому пикселю плоскости проектирования проводятся первичные лучи и каждый луч находит ближайший объект сцены, который он пересекает. Цветом, что соответствует точке пересечения, закрашивается соответствующий пиксель на плоскости проектирования. Эта процедура повторяется для всех точек плоскости проектирования и получается изображение трехмерной сцены. Если же луч не прекращает свое распространение, то он разделяется на три луча-компонента. Каждый новый луч вносит свой вклад в цвет пикселя на плоскости проектирования: отражённый, теневой и преломлённый. Количество этих компонентов определяет глубину трассировки и влияет на качество и фотореалистичность изображения. Благодаря своим концептуальным особенностям, метод позволяет получить очень фотореалистичные изображения, однако из-за большой ресурсоёмкости процесс визуализации занимает значительное время.

Так при рендеринге прозрачных и отражающих объектов отрисовка каждого пикселя занимает еще больше времени, так как лучи отражаются и преломляются [11]. Разные материалы имеют различные коэффициенты преломления. Так же могут возникнуть проблемы с искажением.

2.4 Использование реалистичных анимаций в виртуальной среде игрового движка

Анимация 3d моделей занимает определенное место в памяти. Чем сложнее анимация, тем она занимает больший объем памяти. Также на это влияет и размер самого трехмерного объекта.

К примеру, в проекте виртуальная хирургия есть модель кетгутовой нити, которая служит для сшивания поврежденных тканей. Модель виртуальной нити должна передавать характеристики реальной хирургической нити [12]. Это эффекты изгиба, скручивания, растяжимого поведения. И так как виртуальная сцена имеет большое количество объектов,

модель не должна нагружать проект. При использовании скелетной и объектной анимации для движения небольшой части нити длиной 15 см и состоящей из 450 треугольных полигонов, размер объекта с анимацией при экспорте из 3d редактора стал 1.5 мб, что является слишком большим. А при использовании встроенного инструментария MASH объект стал занимать около 100 мб, что так же не приемлемо. Однако подход с использованием данного инструментария позволяет избежать ошибок в проигрывании анимации в игровом движке.

2.5 Проблема большого количества источников света

Проблема со светом возникает из-за того, что при визуализации из источника света испускается очень много лучей и прослеживается история пути каждого луча. И поэтому чем больше источников света, тем больше лучей и тем больше времени уйдет на просчитывание цвета каждого пикселя на сцене. А на сцене может огромное количество источников света. Например, окружение – ночной городской ландшафт, с большим количеством фонарных столбов.

Глава 3. Авторские подходы к оптимизации окружения

3.1 Использование карты нормалей

Низкополигональная модель содержит меньшее количество полигонов за счет сокращения точек, не влияющих на форму объекта. Также на этой модели удаляются все полигоны, которые не будут видны в игре и отбрасывать тень. Далее создается развертка модели. В зависимости от того где находится объект на сцене, и с какого ракурса он виден определяется его тексель. Для сглаживания низкополигональной модели предстоит перенести нормали с высокополигональной модели. Перенос осуществляется методом запекания в созданную развертку. Получается карта нормалей. Нормаль – это уникальный вектор полигона, отвечающий за то, как ложится тень на поверхность. RGB-значение каждого пикселя карты нормалей соответствует XYZ значения вектора. Используя карту нормалей создается эффект сглаживания и детализации низкополигональной модели (см. рис. 4).

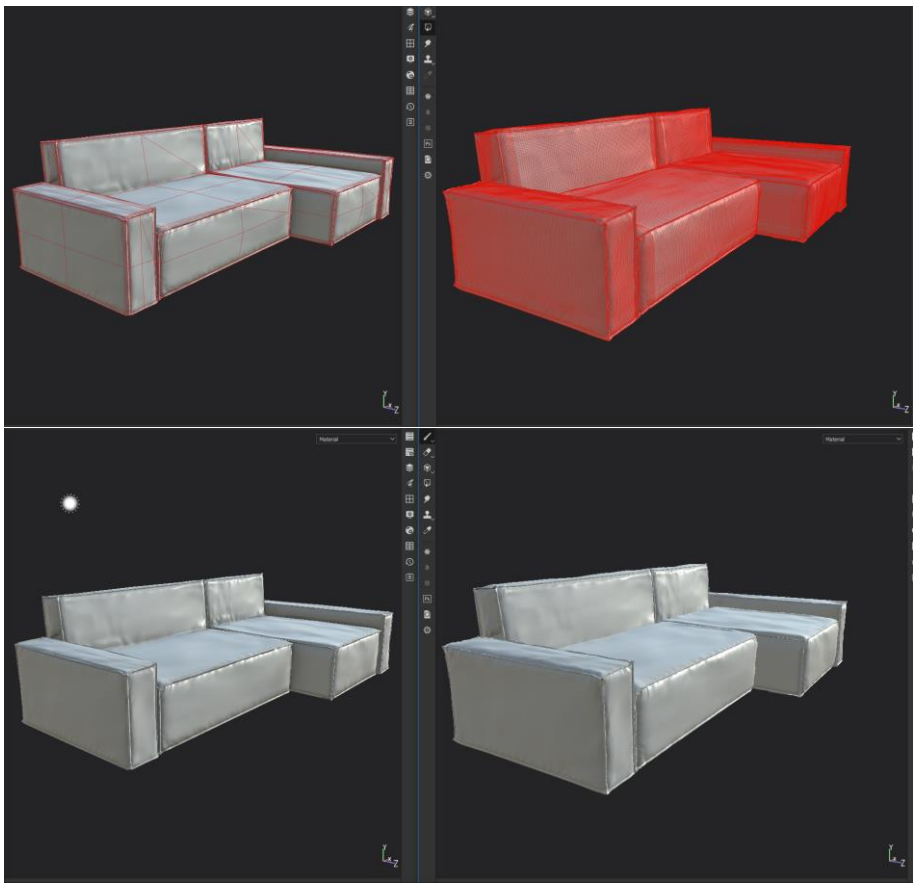

Рисунок 4 - Низкополигональная и высокополигональная модель дивана

Например, на рисунке 4 сравнение 3d модели дивана, низкополигональной и высокополигональной. Количество полигонов у первой модели – 4416, у второй – 474330. При рендеринге векторы карты нормалей умножаются на базовые нормали полигонов.

3.2 Текселерация

Текселерацией имеет большое значение при работе с окружением. Это связано с тем, что в окружение чаще используются тайленные текстуры, текстурные атласы и на объекты окружения редко выделяют отдельные карты. Например, к персонажам, абсолютная текселерация не имеет такого важного значения, т.к. в основном, все персонажи имеют схожие физические размеры. На персонажа вполне может использоваться 2 сета текстур, 2048x2048 для тела, 1024x1024 для головы, и возникает задача выжать максимальный

тексель, правильно сделав UV-развертку. Для элементов окружения, обычно указывают абсолютный тексель (например, 256 px/m), т.к. они сильно разнятся по размерам (стена здания, столб, дорога, окно). Например, для окружения - аэропорт может быть использована одна текстура (см. рис. 5).

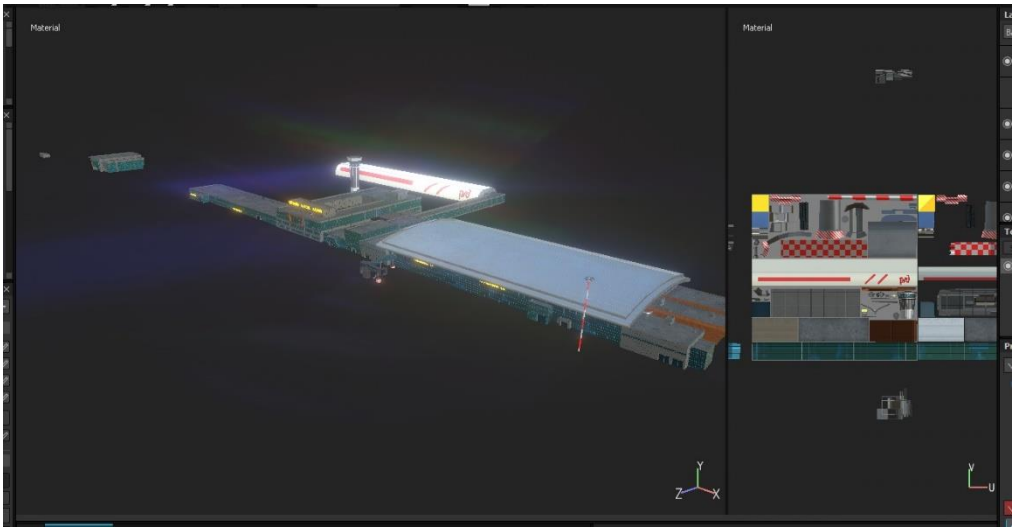

Рисунок 5 - Текстура для аэропорта

3.3 Текстурный атлас

Использование текстурного атласа позволяет уменьшить количество используемых текстур. Он представляет из себя одну большую текстуру, в которой содержатся несколько маленьких текстур. Текстурные атласы создаются для экономии вычислительных ресурсов. Так если назначить всем деревянным объектам один материал, обращающийся к одному текстурному атласу, в который зашиты все 20 текстур, то GPU сможет отрендерить 20 объектов за один вызов (см. рис. 6).

Рисунок 6 - Использование текстурного атласа для модели деревянного дома

3.4 Анимация

Для решения проблемы оптимизации окружения с анимацией виртуальной нити было использован способ анимации при помощи градиентной текстуры. Текстура имеет канал opacity. Там, где текстура не прорисовывается на модели, она будет прозрачной (см. рис. 7). Этот подход обеспечивает стабильное и быстрое создание нити, не влияя на производительность проекта.

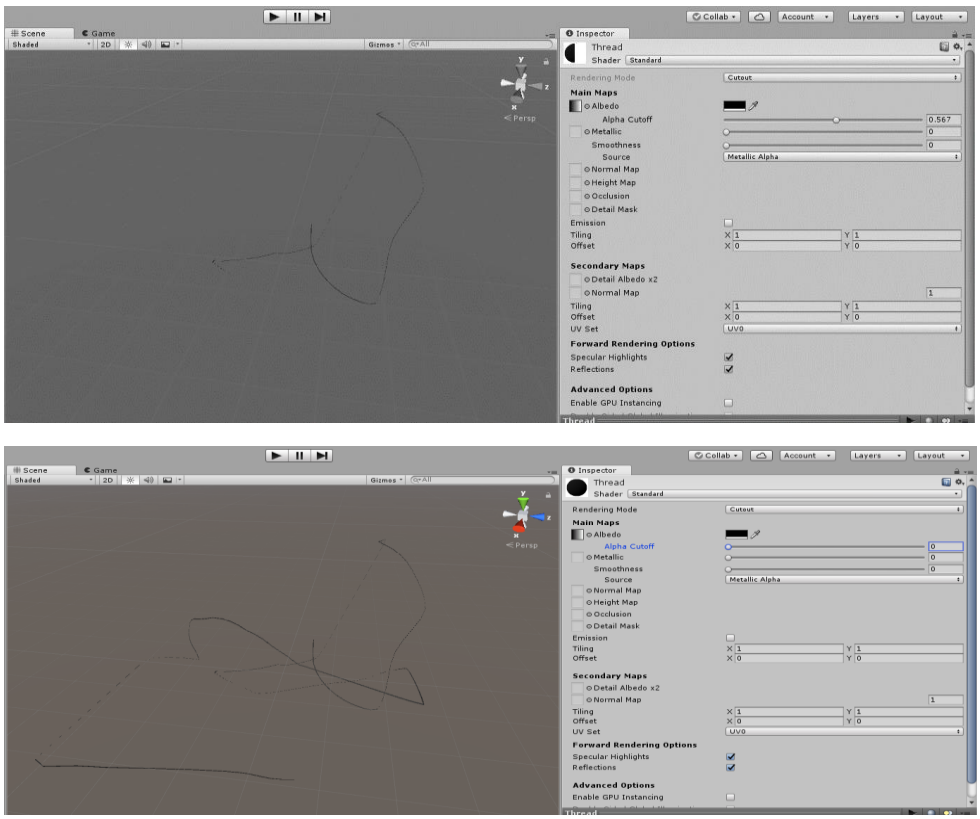

Рисунок 7 - Анимация нити при помощи градиентной текстуры

3.5 Освещение

Для лучшей производительности в игровом движке Unity запекается освещение. Это в какой-то степени решает проблему с большим количеством источников света. Освещение запекается в текстуру. Этот подход не подходит, если на сцене динамический источник света. Например, происходит смена дня и ночи. Также на сцене происходит перемещение по локации, к примеру, поезд, движущийся по локации отбрасывает перед собой тень. В этом случае не получится запечь освещение. К примеру, метод подходит для симулятора, в котором действие происходит в помещении (см. рис. 8).

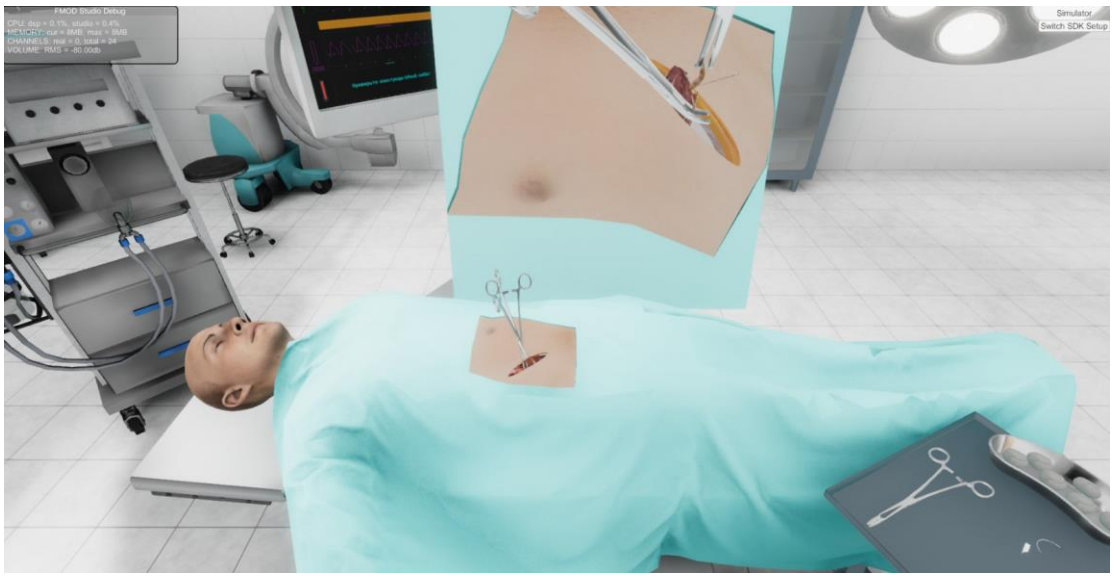

Рисунок 8 - Виртуальная операционная с запеченным светом

Глава 4. Практические примеры реализации окружений в виртуальных средах со сравнительным анализом избранных подходов

4.1 Выбор подхода

Проблема реализации окружения таким образом, чтоб оно не сказывалось на производительности часто возникает в работе разработчика игр. При разработке приложения для виртуальной реальности, в которой пользователь выбирает квартиры в определенном жилищном комплексе, возникла проблема создания высокореалистичного окружения – города Уфа.

Комплекс расположен в окружении лесного массива, открывается вид на долину реки и вдали видны городские постройки. Пользователь видит окружение только с окон. Это позволит погрузиться в атмосферу квартиры при выборе, точно знать, что будет видеть человек глядя в окно.

Использование большого количества 3d объектов на сцене в игровом движке Unity сильно сказалось бы на продуктивности. Даже при использовании максимально низкополигональных деревьев, они занимали большое пространство в оперативной памяти и снижали частоту кадров ниже 20 fps (см. рис. 9).

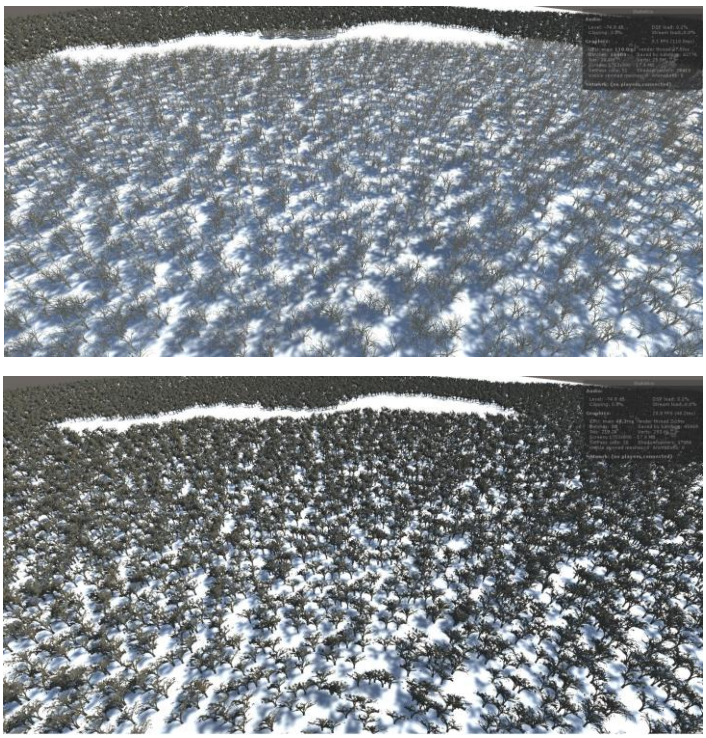

Рисунок 9 - Отрисовка моделей деревьев в игровом движке

Было решено использовать 360 высококачественные панорамные фотографии. Однако получить фотографии с помощью квадрокоптера не получилось, так как комплекс еще не был достроен, поэтому было решено попробовать отрендерить окружение и использовать его в качестве фона (см. рис. 10).

Рисунок 10 - Пробный шот – вид из окна.

Результат оказался приемлемым, производительность и частота кадров не упала.

Итак, при создании городского окружения был выбран участок с определенными референсами (см. рис. 11).

Рисунок 11 - Референсы г.Уфа

4.2 Создание окружения

Для получения высокореалистичных изображений и использования их в качестве фона в игровом движке было определено:

- Положение камеры;
- Охват камеры;
- Определить ландшафт, неровности на выбранной территории.

Полученное изображение должно передавать вид из окна квартиры, расположенной на высоте 8 этажа. Было решено рендерить 360 панорамное изображение, так как в квартире несколько окон, которые дают вид с разных сторон.

Для расположения трехмерных объектов следует определить ландшафт участка, т.к. на референсах хорошо заметны неровности.

Используя программу SketchUp, была выбрана территория города Уфа. С помощью команды Show Terrain был построен ландшафт участка города. На участке побережья в города заметен склон (см. рис. 12).

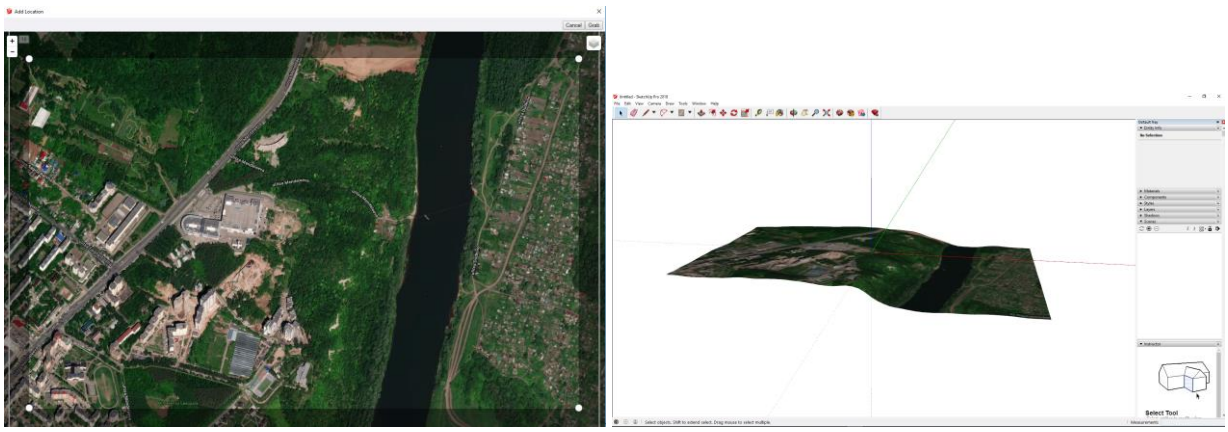

Рисунок 12 - Ландшафты территории г.Уфа в программе SketchUp

В программе Adobe Photoshop была получена карта высот участка побережья (см. рис. 13).

Рисунок 13 - Карта высот участка побережья в городе

Для визуализации окружения была выбрана программа Terragen 4. Сцена окружения города содержит огромное количество объектов, предметов, которые сильно нагружают сцену, и значительно увеличивают время визуализации. Эффекты облаков, созданные с помощью системы частиц (particle system), так же очень долго рендерятся. Проблема объемного рендеринга частиц заключается в том, что, нельзя оценить шейдер один раз на поверхности, как на твердом меше (форма трехмерного объекта). Природные объекты часто имеют фрактальную форму, то есть обладает свойством

самоподобия. Природные объекты, обладающие фрактальными свойствами, отличаются неполнотой и неточностью повторений структуры: границы облаков, линия берега, листья растений, деревья и так далее. Благодаря использованию фрактальных алгоритмов просчета изображения происходит визуализация реалистичных картин с помощью генератора природных ландшафтов [13].

Карта высот была использована в программе Terragen для создания ландшафта (см. рис. 14).

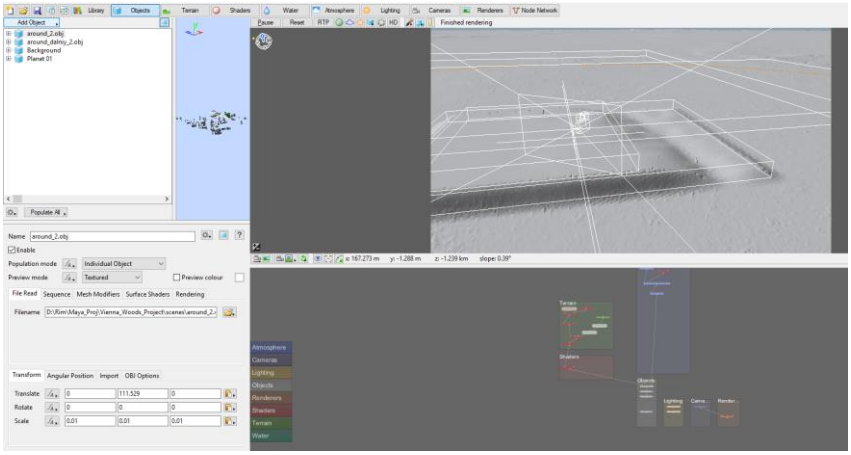

Рисунок 14 - Программа Terragen

В программе Autodesk Maya были созданы 3d модели зданий, деревьев, столбов, тротуара, дорог и других объектов. Модели объектов, которые расположены на переднем плане детализированы (см. рис. 15).

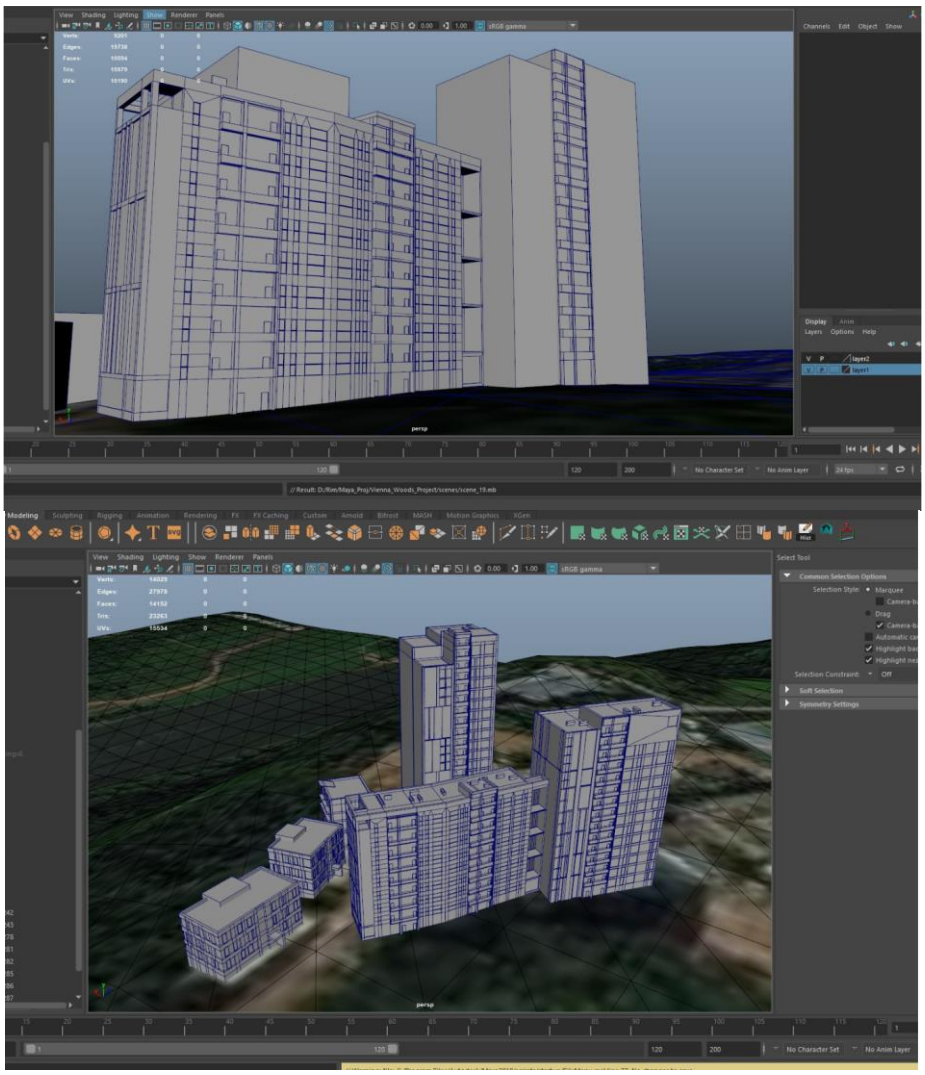

Рисунок 15 - Детализированная низкополигональная модель здания

Объекты на заднем плане не были детализированы, чтобы не нагружать сцену лишними объектами. Так в качестве зданий на заднем плане были использованы прямоугольные параллелепипеды различных размеров и высоты, которые были процедурно замоделированы в программе Houdini (см. рис. 16).

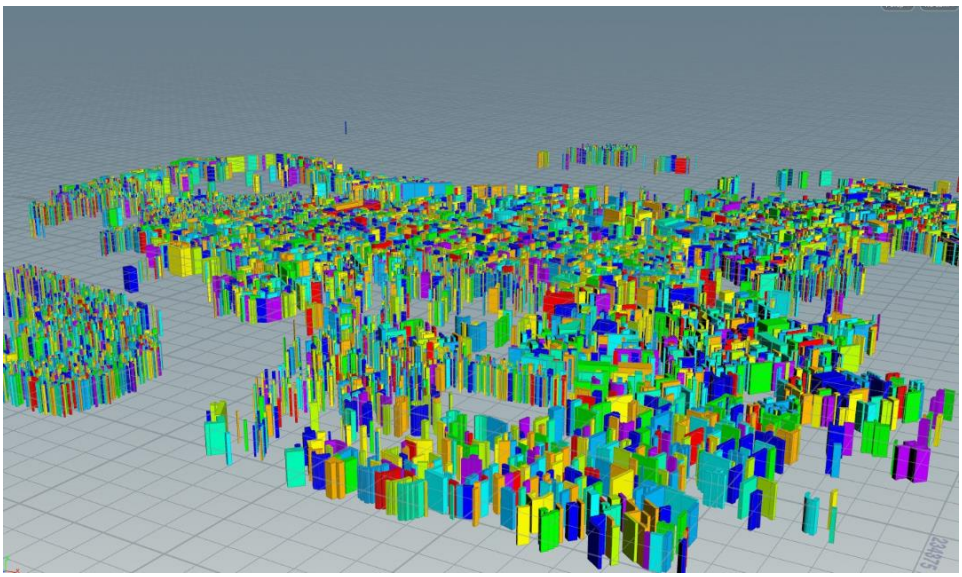

Рисунок 16 - Не детализированные модели здания на заднем плане

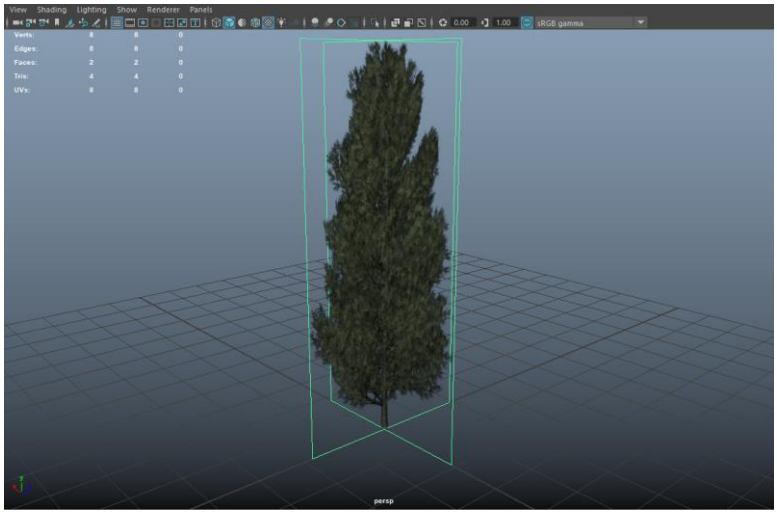

Рисунок 17 - Не детализированная модель дерева

Низко полигональные модели деревьев были созданы из 2 плоскостей с прозрачной текстурой так как в окружении очень большое количество деревьев на дальнем плане и нет необходимости делать их детально (см. рис. 17).

Текстуры 3d моделей были созданы в программе Substance Painter (см. рис. 18).

Рисунок 18 - Текстура здания

Для эффекта света в окнах были созданы альфа текстуры и материал с каналом Emissive для свечения (См. Рис. 19).

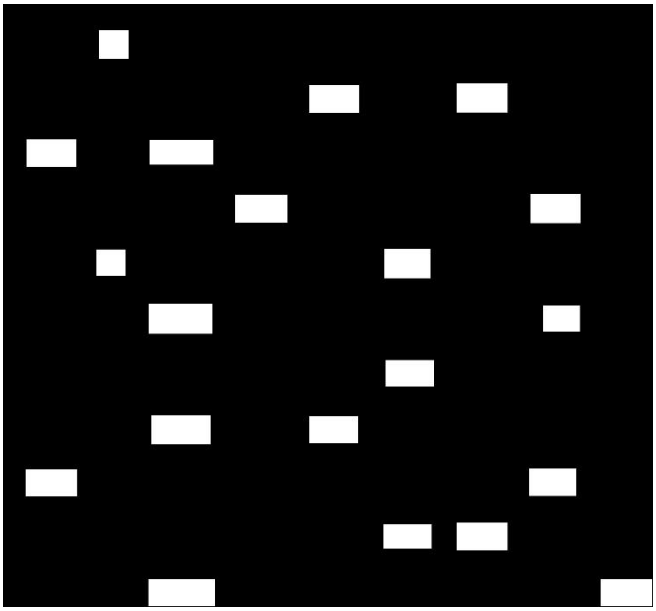

Рисунок 19 - Альфа текстура здания

В программе Terragen была собрана сцена. Все объекты были импортированы (см. рис. 20).

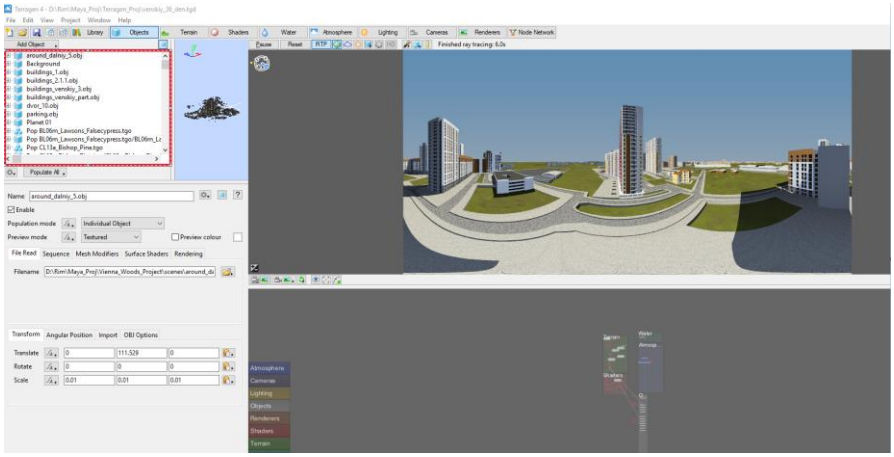

Рисунок 20 - Сборка сцены в программе Terragen

При настройке света указываем параметры солнца в узле Sunlight, который используется для освещения сцены:

- направление;
- высоту;
- цвет;
- силу;
- отбрасывание теней;
- тени атмосферы;
- видимость солнечного диска;
- мягкость теней.

Было решено рендерить 4 времени суток: ночь, утро, день, вечер. Для каждого результата было настроен узел освещения (см. рис. 21). Высота и направление задается в градусах: 0 – север, 90 – восток для направления и 90 – прямо над головой, 0 – на горизонте для высоты.

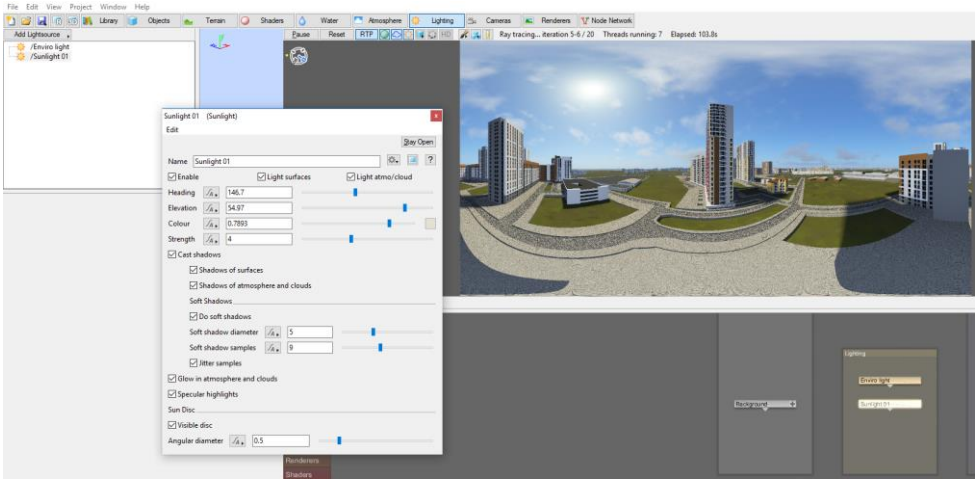

Рисунок 21 - Настройка освещения

Так же на освещение на сцене влияет и узел Атмосфера. Этот узел обеспечивает эффект полной атмосферы планеты, дает возможность создавать голубое небо, красные закаты, увеличивать дымку (см. рис. 22). Были указаны:

- плотность дымки
- цвет горизонта дымки
- тип дымки
- количество свечения дымки
- сила свечения дымки
- свет окружения

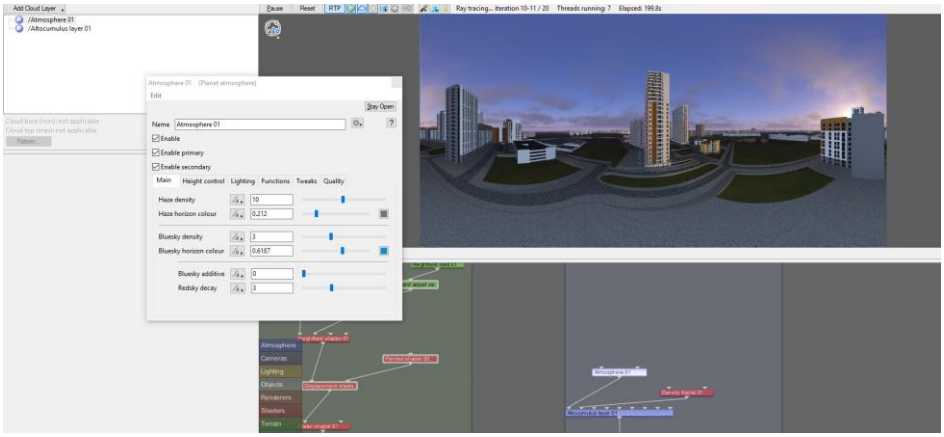

Рисунок 22 - Настройка атмосферы

Облака созданы в узле Облачный слой (см. рис. 23). Узел позволяет создавать слой облаков на всей планете, который может быть двухмерным с моделированным 3D-затенением для более быстрого рендеринга или полным объемным 3D-режимом для максимальной реалистичности, но с более

длительным временем рендеринга. Расположение слоя облаков определяется в первую очередь высотой и высотой слоя облаков. Предусмотрено множество дополнительных настроек для тонкой настройки общего вида облачного слоя для имитации практически любого реального типа облаков. Были указаны:

- высота облаков
- глубина
- центр
- радиус
- шейдер плотности
- резкость края
- плотность

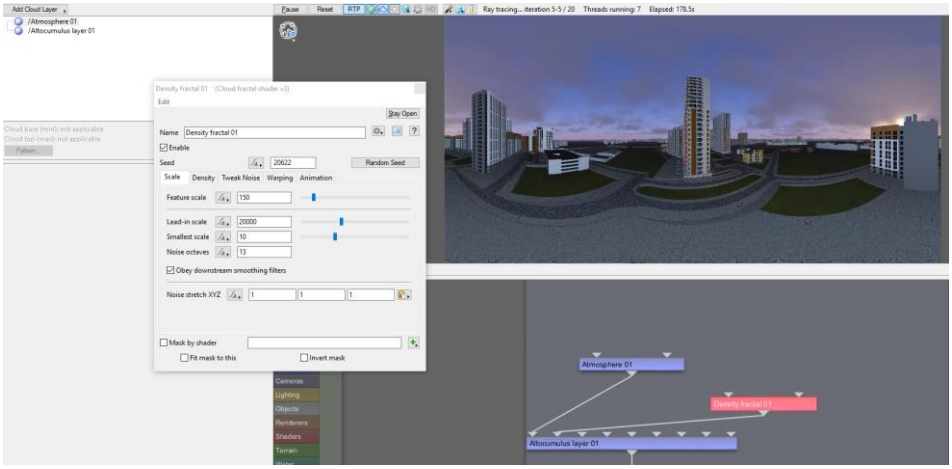

Рисунок 23 - Настройка облаков

Так как было решено рендерить 4 разных времени суток, указываем разные значения для генерации диаграммы облаков. Каждое новое значение генерирует другой шаблон облаков (см. рис. 24).

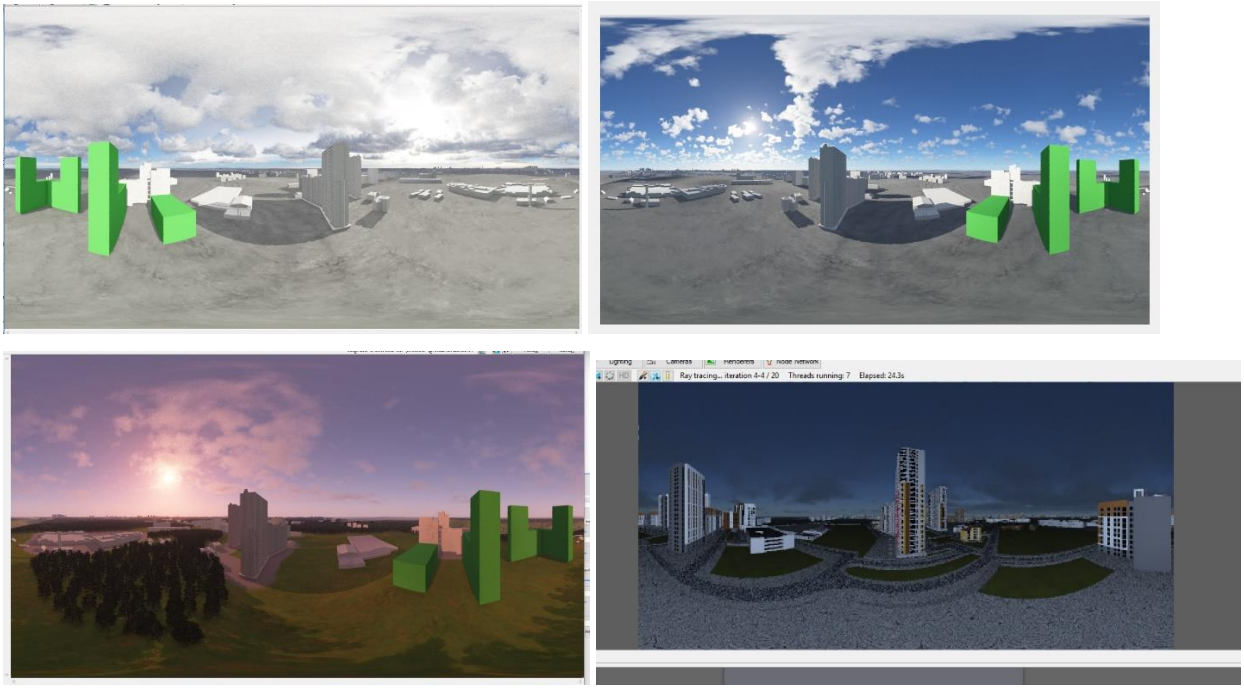

Рисунок 24 - Настройка света и облака

Далее указали позицию камеры и тип сферический для рендера 360 панорамного изображения.

Для получения итогового изображения был настроен узел Render. Узел рендеринга содержит настройки, используемые для создания окончательных рендеров, также для запуска рендеринга анимационных последовательностей. Узел Render работает совместно с узлом Camera, чтобы создать визуализацию сцены с определенной точки. Основные элементы управления включают ширину и высоту выходного изображения, камеру, с которой будет визуализироваться сцена, общую детализацию сцены и сглаживание (см. рис. 25). Также предоставляется множество дополнительных элементов управления для точной настройки детализации и качества сцены, включая настройки глобального освещения (Global illumination, GI), а также функции контрастности и гамма-коррекции.

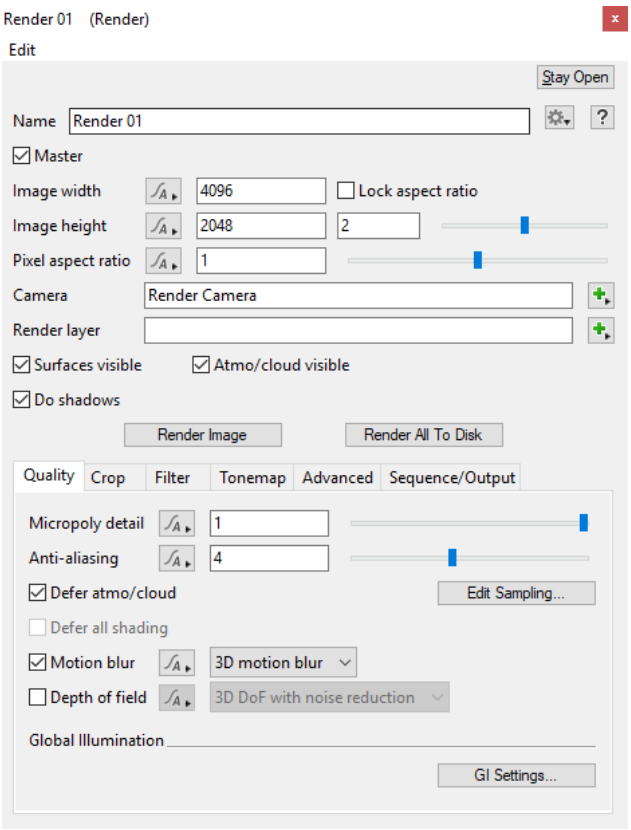

Рисунок 25 - Настройка узла Render

Рисунок 26 - Итоговое фотореалистичное изображение. Утро.

Рисунок 27 - Итоговое фотореалистичное изображение. Ночь.

Получили по 4 кадра с одного ракурса (см. рис. 26-27).

4.3 Анализ результатов

Программы Terragen использует вычислительную мощность центрального процессора. Для визуализации использовались устройства с характеристиками

1. Intel Core i7-4770 3.40GHz 8ГБ Ram
2. Intel Core i7-7700 3.60GHz 32ГБ Ram
3. Intel Core i7-6850K 3.60GHz 64ГБ Ram

Поученные результаты приведены в таблице 1.

Заполненность кадра:

- высокая – от 500 000 до 1 000 000 объектов в кадре;
- средняя – от 100 000 до 500 000 объектов в кадре;
- низкая – менее 100 000 объектов в кадре.

Таблица.1 Результаты визуализации в программе Terragen.

Шот	**Время**	**Количество источников света**	**Заполненность кадра**	**Характеристики машины**
1 Вечер	7 ч 03 мин	1	высокая	Intel Core i7-7700 3.60GHz 32ГБ Ram
2 Утро	8 ч 23 мин	1	высокая	Intel Core i7-7700 3.60GHz 32ГБ Ram
3 Ночь	23 ч 02 мин 53 сек	103	высокая	Intel Core i7-6850K 3.60GHz 64ГБ Ram
4 День	4 ч 47 мин	1	низкая	Intel Core i7-6850K 3.60GHz 64ГБ Ram
5 Вечер	6 ч 57 мин	1	средняя	Intel Core i7-6850K 3.60GHz 64ГБ Ram
6 Утро	7 ч 17 мин	1	средняя	Intel Core i7-7700 3.60GHz 32ГБ Ram
7 Утро	15 ч 27 мин 35 сек	1	низкая	Intel Core i7-4770 3.40GHz 8ГБ Ram
8 Ночь	21ч 52 мин 55 сек	97	средняя	Intel Core i7-7700 3.60GHz 32ГБ Ram

Для рендеринга сцены используется гибридный рендерер, два разных метода используются вместе для рендеринга сцены. Микрополигональный рендеринг используется для визуализации рельефа, воды, фоновых объектов. Для этих же поверхностей трассировка лучей используется как вторичные лучи, для просчета отражения, освещения и теней. Так же трассировка лучей используется для рендеринга импортированных моделей. Эти объекты фактически являются статическими данными и могут быть эффективно отрисованы с помощью Ray tracer. Он дает более высокое визуальное качество, чем это может быть достигнуто с помощью микрополигонального рендерера. В итоге, можно сказать, что с помощью трассировки лучей объекты рендерятся качественнее и быстрее.

Скорость рендеринга итогового изображения увеличивается за счет того, что для объектов, которые находятся на переднем плане используется метод Ray tracing, а для объектов на заднем плане – микрополигональный. А так как сцена содержит на заднем плане свыше 500 000 объектов в кадре скорость рендера значительно увеличивается. Сцена содержит процедурные

данные такие как: облака, туман, деревья. Микрополигональный рендеринг помогает ограничить объем визуализации

По результатам, приведенным в таблице 1, можно сделать вывод что сцены с не высокой заполненностью кадра и одним источником света рендерятся за наиболее быстрое время. Но при этом следует учитывать и параметры вычислительной машины. Так, шот номер 7 был визуализирован на машине с параметрами - Intel Core i7-4770 3.40GHz 8ГБ Ram. Время рендеринга увеличилось два раза, по сравнению с рендером сцен с такой же сложностью на других машинах. На это повлияло низкая производительность процессора Intel Core i7-4770 3.40GHz, а также количество оперативной памяти. Наиболее значительное время было потрачено на визуализацию ночных сцен – шоты 3 и 8. Эти сцены содержат более 100 источников света. Для просчета используется метод трассировки лучей, который увеличивает время рендеринга на этих сценах в 3 раза.

Итак, использование генератора ландшафта Terragen позволило получить высокореалистичное окружение для игрового движка Unity, для значительного сокращения объектов на сцене игрового движка.

Заключение

Проблемы оптимизации сцен актуальны в игровой индустрии и в целом в направлении связанной с 3d графикой.

В ходе работы были рассмотрены различные технологий рендеринга высокореалистичного окружения виртуальных сред в игровом движке Unity, трехмерном редакторе Autodesk Maya и генераторе ландшафта Terragen. Выявлены основные проблемы и подходы к оптимизации окружения.

Описан собственный подход к оптимизации для игрового движка Unity проектов виртуальной реальности. Визуализировано окружение в генераторе ландшафтов. Для этой визуализации также применены методы оптимизации и проведен анализ времени, потраченного на рендеринг сцен различной сложности. Данный подход позволяет не терять производительность проекта, частоту кадров в игровом движке.

По результатам работы по оптимизации окружения с анимацией виртуальной нити представлен доклад на конференции “Электронная Казань 2019” и опубликована статья [12] в сборнике “Ученые записки института социальных и гуманитарных знаний КФУ”, отдельные работы которого будут изданы журналом CEUR с рейтингованием в Scopus. Исследование в области оптимизации планируется продолжить при обучении в аспирантуре, сместив акцент в направление по автоматизации скелетной анимации.

Данная работа доступна по ссылке:

http://gititis.kpfu.ru/RimRadGazizov/VKR-Gazizov-11712

Список источников

1. Jankovic, D. Relief mapping for urban and natural environments rendering [Text] / D. Jankovic, Z. Mihajlovic // 2011 Proceedings of the 34th International Convention MIPRO. – 2011. – P. 1-5.

2. Sorokin, V. A. Three-dimensional visualization of the underwater environment using graphical library “3Dbodies” [Text] / V. A. Sorokin, A. Y. Demin, D. Z. Khasaeva // 2014 International Conference on Mechanical Engineering, Automation and Control Systems (MEACS). – 2014. - P. 1-4.

3. Huang, H. Simulation and visualization of forest fire growth in an integrated 3D virtual geographical environment - a preliminary study [Text] / Hongyu Huang, Liyu Tang, Jianwei Li, Chongcheng Chen // 2012 20th International Conference on Geoinformatics. – 2012. – P. 15-21.

4. Виртуальная реальность. Инструменты и Движки. [Электронный ресурс]. URL: https://software.intel.com/ru-ru/vr/tools-engines (дата обращения: 10 июня 2019).

5. Виды 3d моделирования. [Электронный ресурс]. URL: https://3d-modeli.net/uroki-3d/6175-vidy-3d-modelirovaniya.html (дата обращения: 10 июня 2019).

6. Интересное о дизайне уровней [Электронный ресурс]: Сюжет посредством игрового окружения URL: http://level-design.ru/pro-ld-book-index/04-environmental-storytelling/ (дата обращения: 10 июня 2019).

7. Planetside Software. [Электронный ресурс]: Terragen 4, документация. URL:https://planetside.co.uk/ (дата обращения: 5 июня 2019).

8. Unity Documentation. [Электронный ресурс]: Руководство Unity. Оптимизации рендеринга. URL: https://docs.unity3d.com/ru/ (дата обращения: 5 июня 2019).

9. Texel density. Зачем нужен и как его применять. [Электронный ресурс]. URL: https://habr.com/ru/post/315146/ (дата обращения: 10 июня 2019).

10. Path tracing vs ray tracing. [Электронный ресурс]. URL: https://www.dusterwald.com/2016/07/path-tracing-vs-ray-tracing/ (дата обращения: 10 июня 2019).

11. Худенко, В. Н. О различных подходах к проблеме визуализации классических математических моделей [Текст] / Худенко В. Н // Вестник Балтийского федерального университета им. И. Канта. Серия: Физико-математические и технические науки. 2012. №10. URL: https://cyberleninka.ru/article/n/o-razlichnyh-podhodah-k-probleme-vizualizatsii-klassicheskih-matematicheskih-modeley (дата обращения: 10.06.2019).

12. Газизов Р.Р. Физика веревки для реализации кетгутовой нити в виртуальной операционной [Текст]. / Р.Р. Газизов // Ученые записки института социальных и гуманитарных знаний КФУ. – 2019. – С.353-360

13. Ампилова Н. Б., Соловьев И. П. Алгоритмы фрактального анализа изображений [Текст] / Ампилова Н. Б., Соловьев И. П. // КИО. 2012. №2. URL: https://cyberleninka.ru/article/n/algoritmy-fraktalnogo-analiza-izobrazheniy (дата обращения: 05.06.2019).

Глоссарий

Визуализация (рендеринг) - это процесс построения изображения, кадра посредством компьютерной программы.

Высококачественное(high-quality) рендеринг – это получение фотореалистичного изображения высокого качества.

Вычислительный кластер – группа компьютеров, предназначенных для распределенных вычислений в параллельном режиме.

Вычислительная платформа - программно-аппаратный комплекс инфраструктуры, от которого зависит работа всей информационной системы.

Генератор ландшафтов – программа, которая позволяет сравнительно быстро создавать фотореалистические земные или безжизненные, будто бы инопланетные пейзажи, порою по своей красоте и реалистичности неотличимые от настоящих фотоснимков.

Графический процессор (GPU) — отдельное устройство персонального компьютера или игровой приставки, выполняющее графический рендеринг.

Плагин — независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей.

Проход визуализации (render pass) - отрендеренное изображение определенного свойства сцены отдельно от других.

Рендерер - компьютерная программа, производящая рендеринг.

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

Секвенция или секвенция кадров - это тип сохранения при котором каждый фрейм (кадр) сохраняется в отдельную картинку и имеющий любой вариант формата картинки например jpg,bmp,gif,tiff,png и многие другие.

Центральный процессор (CPU) — электронный блок либо интегральная схема, исполняющая машинные инструкции, главная часть аппаратного обеспечения компьютера или программируемого логического контроллера.

Autodesk Maya – 3d программный пакет, который позволяет заниматься моделированием, текстурированием, анимированием, композитингом, рендерингом.

Terragen — программа ландшафтного моделирования и анимации, позволяющая создавать фотореалистичные пейзажи.

Texel (аббревиатура от двух слов «TEXture» и «ELement» — текстура и элемент) — элемент текстуры, «точка».

3D-редактор - программный пакет, с помощью которого можно моделировать 3D объекты и создавать на основе этих моделей фотореалистичные изображения.

Приложение

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class EnvironmentController : MonoBehaviour
{
    public MeshRenderer[] giEmissives;
    public float[] giEmissivesIntensityMultiplier;
    public Material[] daytimeSkyboxes;
    public Vector3[] sunRotation;
    public Color[] sunColor;
    public float[] sunIntensity;
    public ChangeablePicture[] framesHorizontal;
    public ChangeablePicture[] framesVertical;
    private Material skyboxMat;

    [System.NonSerialized]
    public MyFloatEvent OnDayTimeChanged = new MyFloatEvent();

    // Use this for initialization
    void Start()
    {
        SetDay();
    }

    private void Update()
    {
    }

    public void SetMorning()
    {
        SetDayTime(
            giEmissivesIntensityMultiplier[0],
            daytimeSkyboxes[0],
            (Random.Range(0, 60) * 60f + (Random.Range(0, 2) + 5f) * 3600f) /
86400f,
            sunRotation[0],
            sunColor[0],
            sunIntensity[0]
```

```
        );
    }
    // Sets time to morning from 11:00 to 12:59
    public void SetDay()
    {
        SetDayTime(
            giEmissivesIntensityMultiplier[1],
            daytimeSkyboxes[1],
            (Random.Range(0, 60) * 60f + (Random.Range(0, 1) + 11f) * 3600f)
/ 86400f,
            sunRotation[1],
            sunColor[1],
            sunIntensity[1]
         );
    }

    // Sets time to morning from 18:00 to 20:59
    public void SetEvening()
    {
        SetDayTime(
            giEmissivesIntensityMultiplier[2],
            daytimeSkyboxes[2],
            (Random.Range(0, 60) * 60f + (Random.Range(0, 2) + 18f) * 3600f)
/ 86400f,
            sunRotation[2],
            sunColor[2],
            sunIntensity[2]
        );
    }

    // Sets time to morning from 22:00 to 1:59
    public void SetNight()
    {
        SetDayTime(
            giEmissivesIntensityMultiplier[3],
            daytimeSkyboxes[3],
            ((Random.Range(0, 60) * 60f + (Random.Range(0, 3) + 22f) * 3600f)
/ 86400f) % 1,
            sunRotation[3],
            sunColor[3],
            sunIntensity[3]
        );
```

```
    }

    private void SetDayTime(float giEmissiveMultiplier, Material skyMaterial,
float normalizedTime, Vector3 sunRotation, Color sunColor, float
sunIntensity)
    {
        OnDayTimeChanged.Invoke(normalizedTime);
        RenderSettings.skybox = skyMaterial;
        RenderSettings.sun.transform.rotation =
Quaternion.Euler(sunRotation);
        RenderSettings.sun.color = sunColor;
        RenderSettings.sun.intensity = sunIntensity;
        for (int i = 0; i < giEmissives.Length; i++)
        {
            DynamicGI.SetEmissive(giEmissives[i],
giEmissives[i].material.GetColor("_EmissionColor") * giEmissiveMultiplier);
        }
        DynamicGI.UpdateEnvironment();
    }

    public void LoadPhotos(List<Texture2D> vkAvatars)
    {
        if (vkAvatars != null)
        {
            List<Texture2D> horizontalTextures = new List<Texture2D>();
            List<Texture2D> verticalTextures = new List<Texture2D>();

            for (int i = 0; i < vkAvatars.Count; i++)
            {
                if (vkAvatars[i].height >= vkAvatars[i].width)
                {
                    Texture2D tex = new Texture2D(vkAvatars[i].width,
vkAvatars[i].height);
                    tex.SetPixels(vkAvatars[i].GetPixels());
                    TextureScale.Point(tex, 256, 512);
                    verticalTextures.Add(tex);
                }
                else
                {
                    Texture2D tex = new Texture2D(vkAvatars[i].width,
vkAvatars[i].height);
                    tex.SetPixels(vkAvatars[i].GetPixels());
```

```
                    TextureScale.Point(tex, 512, 256);
                    horizontalTextures.Add(tex);
                }
            }

            for (int i = 0; i < framesHorizontal.Length; i++)
            {
                MeshRenderer frameMR =
framesHorizontal[i].GetComponent<MeshRenderer>();
                for (int j = 0; j < framesHorizontal[i].materialIDs.Length;
j++)
                {
                    if (horizontalTextures.Count == 0)
                    {
                        break;
                    }
                    int randomID = UnityEngine.Random.Range(0,
horizontalTextures.Count);

frameMR.materials[framesHorizontal[i].materialIDs[j]].mainTexture =
horizontalTextures[randomID];
                    horizontalTextures.RemoveAt(randomID);
                }
            }

            for (int i = 0; i < framesVertical.Length; i++)
            {
                MeshRenderer frameMR =
framesVertical[i].GetComponent<MeshRenderer>();
                for (int j = 0; j < framesVertical[i].materialIDs.Length;
j++)
                {
                    if (verticalTextures.Count == 0)
                    {
                        break;
                    }
                    int randomID = UnityEngine.Random.Range(0,
verticalTextures.Count);

frameMR.materials[framesVertical[i].materialIDs[j]].mainTexture =
verticalTextures[randomID];
                    verticalTextures.RemoveAt(randomID);
```

```
                }
            }
        }
    }

    [System.Serializable]
    public class MyFloatEvent : UnityEvent<float>
    {
    }
}
```