Практическая модель динамических атмосферных явлений при визуализации открытых пространств в системах визуализации реального времени

Н.А.Елыков, И.В.Белаго, С.М.Козлов С.А.Кузиковский, М.М.Лаврентьев,
Лаборатория программных систем машинной графики ИАиЭ СО РАН. Новосибирск, Россия
{nicolas, bel, smk, stas} @sl.iae.nsk.su, mmlavr@nsu.ru

Abstract

Авторами предложен простой метод анимации погодных явлений и их параметризации с помощью малого набора параметров, для получения широкого спектра атмосферных условий. Также предложен ряд оптимизации существующих алгоритмов визуализации динамических облаков, позволяющих во-первых, уменьшить, а во-вторых, масштабировать затраты ресурсов, необходимые для отображения облаков.

Keywords: *sunlight, skylight, illumination, clouds rendering, impostors*

1. ВВЕДЕНИЕ

Современные промышленные графические ускорители для персональных компьютеров позволяют решать задачи, требующие генерации изображения открытых пространств фотографического качества в реальном времени. Кроме того, высококачественная визуализация различных атмосферных явлений, таких как смена дня и ночи, погоды, облаков и пр., является исходным требованием при создании большинства тренажеров наземного и летного транспорта, а также очень многих приложений виртуальной реальности.

Авторами разработана комплексная аналитическая модель, позволяющая визуализировать подобные явления в различных атмосферных условиях, которые задаются рядом легко подбирающихся параметров.

2. ВИЗУАЛИЗАЦИЯ НЕБА, ЗВЕЗД, СОЛНЦА И ЛУНЫ.

Геометрическая модель неба представляет из себя сегмент сферы с диаметром много меньше размера сцены. Чтобы добиться кажущейся бесконечности, мы двигаем конечную модель неба вместе с наблюдателем. Кроме того, для того, чтобы показать, что небо находится гораздо дальше всех обектов в сцене, можно визуализировать небо раньше всех объектов в сцене, при этом с помощью матрицы проекции добиваясь, чтобы максимальный z неба совпадал с дальней отсекающей плоскостью, либо чтобы при визуализации неба значения буфера глубины не модифицировались. В качестве текстуры неба используется фототекстура “высоких” облаков мэпируемая планарно на сегмент сферы, при этом, изменяя параметры проекции, можно добиться эффекта движения облаков в некотором направлении, более же низкие облака отображаются с помощью системы импостеров, о которой речь пойдет ниже. Кроме того, на небо накладывается линейный туман. Цвет тумана, его ближние и дальние границы, а также цвет неба являются параметрами текущих атмосферных условий, которые задают такие эффекты как освещенность, рассеяние света в атмосфере и пр., анимация этих параметров будет рассмотренна ниже.

Солнце и луна представлются в виде прямоугольников перпендикулярных радиусу сегмента сферы задающей небо, положение солнце и луны в рссчитываются с помощью упрощенной модели описанной в [1]. При наложении текстуры солнца - Sun_{rgb} (см. Рис 1), на купол неба - $Dome_{rgb}$ для результирующего цвета S_{rgb} используется формула $S_{rgb} = Sun_{rgb} * (1 - Dome_{rgb}) + Dome_{rgb}$.

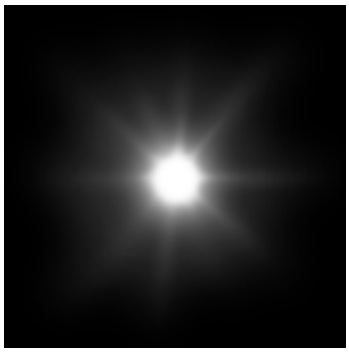

Рисунок 1: Текстура солнца.

Текстура луны не может быть использованна так же как и текстура солнца, т.к. в этом случае, луна не будет иметь фазу. Для моделирования фазы, на текстуру луны $Moon_{rgb}$, накладывается текстура бампа $Bump_{rgb}$, представляющая из себя полусферу, скалярно перемноженная на направление источника солнца $SunVec_{rgb}$ по следующей формуле.

$$S_{rgb} = Moon_{rgb} * (SunVec_{rgb} \bullet Bump_{rgb})$$

Рисунок 2: Текстуры луны.

Для того что бы смоделировать закат и восход солнца и луны, на них накладывается туман, того же цвета что и туман на небе, таким образом, чтобы во время восхода и заката луна и солнце полностью затуманивалась и совпадали по цвету с цветом неба.

Геометрическая модель звездного неба, представляет систему частиц, построенную на сфере единичного радиуса, каждая из частиц обладает своим цветом и размером, в соответствии с звезной величиной и цветом звезды, представляемой данной частицей. Ориентация модели рассчитывается с помощью упрощенной модели описанной в [1]. Также все частицы обладают прозрачностью, которая позволяет плавно проявлять и убирать звезды в момент восходов и закатов. Прозрачность звезд также является параметром текущих атмосферных условий.

3. ВИЗУАЛИЗАЦИЯ ОБЛАКОВ

3.1 Введение

Визуализация облаков является важной задачей в отображении реалистичных открытых сцен, не говоря уж о приложениях, где небо играет основную роль – таких как авиасимуляторы. Из-за своей структуры, облака являются объектами, напрямую непредставимыми в полигональной форме, и существуют несколько способов аппроксимации неба и облаков.

- Заранее нарисованные текстуры неба, интерполяция между ними для смены картины неба. С одной стороны, для сцен, где камера находится около поверхности земли, достигается практически фотореалистичное качество в статике, с другой стороны, для динамического неба и для камеры, летающей между облаками, способ неприемлем.
- Двумерные облака, генерация в текстуре с помощью шума Перлина. Этот способ решает многие проблемы создания динамического неба, включая наличие переменной облачности, эффект клубящихся облаков, и некоторые другие, но всегда оставаясь в рамках двумерной картинки.
- Наиболее близкий к реальности способ - отображение, основанное на системах частиц. В этом случае облако представляется как набор полигонов с текстурами, всегда обращенных к наблюдателю. Микро-детальность формы и краев облака задается текстурой прозрачности на полигонах. Разумеется, такой способ позволяет использовать всевозможные эффекты, связанные с облаками. К сожалению, как просчет формы, так и отображения облаков таким способом требует крайне много ресурсов.

Авторы предлагают возможности оптимизации последнего, наиболее общего и реалистичного подхода, позволяющие достичь качества и производительности, соответствующей требованиям систем реального времени.

3.2 Обзор существующих методов.

3.2.1 Заранее нарисованные текстуры.

Если камера находится достаточно близко к поверхности, то можно использовать заранее нарисованные текстуры для отображения неба. В рамках статической картины, это позволяет достичь практически фотореалистичного качества, так как для этого необходимва текстура достаточного качества, что не является проблемой (Рис.3).

Рисунок 3: Использование фототекстуры**.**

В этом случае проблемы заключаются в визуализации динамического неба. Среди возможных подходов можно перечислить:

- Использование нескольких слоев текстур, и сдвиг их относительно друг друга.
- Интерполяцию между несколькими картинами неба.
- Изменение гаммы в зависимости от времени суток.
- Пост-эффекты свечения солнца и луны.

Хотя применение всех этих методов может создать видимость реалистичной смены дня и ночи, все это работает только при существенных ограничениях. Во-первых, пренебрегается изменением позиции наблюдателя, так как в рамках такого метода невозможно построить вид облака для произвольной позиции наблюдателя. Другими словами, модель работает в предположении, что расстояние до облака настолько далеко, что камера не может сдвинуться на расстояние, при котором бы изображение облака бы изменилось. Это относится и к движению облака, т.е. создает серьезные ограничения динамичности неба.

Кроме того, для разнообразной и реалистичной картины неба необходим большой набор текстур высокого разрешения для интерполяции, что означает значительные затраты текстурной памяти.

Наконец, такой подход ни в коей мере нельзя использовать для визуализации облаков в авиасимуляторах – требуются такие эффекты как выход за пределы облачного слоя, полет сквозь облако, скорости таковы, что облака могут быть видны с совершенно разных углов и т.д.

Можно сказать, что способ вполне приемлем, если предъявляются невысокие требования к динамичности неба и облака находятся далеко от наблюдателя.

3.2.2 Двумерные текстуры, основанные на шуме Перлина.

Шум Перлина [2] – один из способов получения случайных процедурных текстур. Вкратце, он заключается в том, чтобы сложить несколько обычных шумов разной частоты с амплитудой, обратно пропорциональной частоте. Такой подход дает плавные переходы и изображения, более применимые как процедурные текстуры (рис.4).

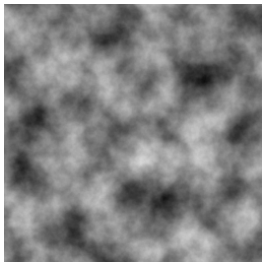

Рисунок 4: Пример шума Перлина.

Одно из классических применений шума Перлина – создание двумерных карт облаков. Для регулирования плотности облаков достаточно установить все пиксели, меньшие какого-то значения в цвет фона. На рисунке 5 показаны результаты отсечения текстуры на рисунке 4 разными значениями.

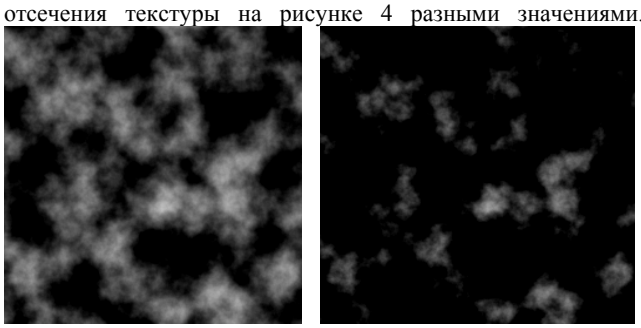

Рисунок 5: Пример исппользования шума Перлина для генерации текстуры облаков.

Видно, что эти рисунки напоминают небо различной облачности. После подстановки нужных цветов в качестве фона и цвета облаков, можно получить изображения, подобные рисунку 6.

Рисунок 6 взят из Terragen [3], системы отображения ландшафтных сцен, работающей не в реальном времени.

Ясно, что в этом случае облака могут сдвигаться относительно наблюдателя, динамически исчезать и появляться, возможны даже эффекты клубящихся облаков. Последний эффект достигается переходом про градиенту карты облачности. Кроме того, так как текстуры процедурно генерируются по мере необходимости, проблема текстурной памяти не стоит так остро.

Тем не менее, облака остаются плоскими, что хорошо заметно для далеких облаков, и реалистичность картины всегда этим ограничена. На форма облака по-прежнему ограничена двумерным изображением, поэтому такой подход очень сложно применять в авиасимуляторах.

Как и с любыми текстурными методами, качество изображения во многом зависит от разрешения текстуры. Даже в Terragen, системе, которая может себе позволить расчет текстур облаков огромного разрешения, видно, что качество вблизи от камеры хуже качества вдали, что является особенностью проекционных преобразований. В системах реального времени вряд ли возможны расчеты таких больших текстур, так как отображение облаков не должно занимать существенных ресурсов, что означает более плохое качество (рис. 7).

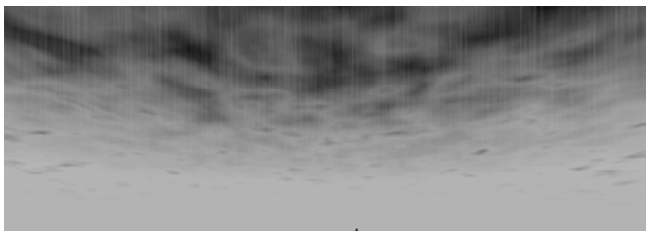

Рисунок 7: Качество изображения в системах реального времени.

Резюмируя, этот метод можно рассматривать как процедурную генерацию текстур для предыдущего подхода. Таким процедурным созданием можно решить многие проблемы создания динамичного неба, присущие этому подходу, разумеется, в ущерб фотореалистичному качеству, и с затратой существенных ресурсов на просчет текстур облаков. К сожалению, облака остаются плоскими, со всеми вытекающими недостатками.

3.2.3 Облака, основанные на системах частиц.

Как уже упоминалось, в силу своей структуры облака не могут быть напрямую представлены полигональными моделями, что затрудняет их отображение как трехмерных объектов. Выход – представление облаков в виде набора частиц с текстурами прозрачности [4-6]. В этом случае контур и изображение облаков определяется перекрытием полупрозрачных частиц, что соответствует физической структуре настоящего облака (рис.8, 9).

Рисунок 8 Представление облаков ввиде системы частиц.

Рисунок 9 Wireframe представление облаков ввиде системы частиц.

Такое представление облаков позволяет управлять освещением на уровне каждой частицы, которое можно обсчитывать на основе моделей различной сложности. К примеру, в [5] приведена достаточно ресурсоемкая модель, учитывающая прохождение света сквозь облако с учетом рассеяния частиц. Как будет показано далее, с небольшими изменениями эта модель может с успехом использоваться в приложениях реального времени.

Основной проблемами при использовании облаков такого типа является затрата ресурсов на их просчет и отображение.

Существуют два подхода для формирования геометрии облака – работа дизайнеров [6] и моделирование физического процесса формирования облаков [4].

Первому способу присущи все недостатки ручных методов – необходимые затраты времени, разработка соответствующих инструментов, ограниченность контента. Эти проблемы усугубляются требованием к динамике неба – нужна разработка средств анимации появления, исчезновения и жизненного цикла облаков.

Для физического моделирования приходится использовать большие массивы данных с существенным с точки зрения затрат вычислительных ресурсов количеством итераций для получения реалистичной картинки. Для получения динамической картины нужно в реальном времени продолжать симуляцию в масштабах видимости облаков (обычно десятки километров), что может означать ощутимые затраты производительности.

Вторая проблема – затраты ресурсов на отображение облаков. В зависимости от необходимой детализации облако может включать от сотен до тысяч частиц, каждая из которых занимает существенный размер на экране, и каждая всегда рисуется с учетом полупрозрачности. Это огромные затраты заливки для графического акселератора, особенно в случае плотного облачного слоя.

С другой стороны, метод является наиболее общим и универсальным методом представления облаков, и легко модифицируется для любых требований – таких, как полет сквозь облако, возможность рассмотреть облако под любым углом, полет над облачным слоем и т. д.

3.3 Предлагаемые оптимизации.

3.3.1 Краткое описание.

В статье предлагаются оптимизации последнего метода, использующего системы частиц для описания облаков, позволяющие его эффективное использование в системах отображения реального времени.

Основные затраты ресурсов, связанные с этим методом таковы:

- Просчет освещения частиц с учетом плотности облака и рассеяния света частицами.
- Физическая симуляция процессов, происходящих в облаках для определения их формы и просчета динамики изменения неба.
- Отображения облаков.

Каждая из этих проблем будет подробно рассмотрена ниже, и будут предложены возможные пути ее решения.

3.3.2 Просчет освещенности в облаке.

В [4] предложен процесс просчета степени освещенности частицы в облаке. Вкратце, он заключается в последовательной отрисовке частиц в отдельную текстуру в порядке отдаления от источника света (рис 10).

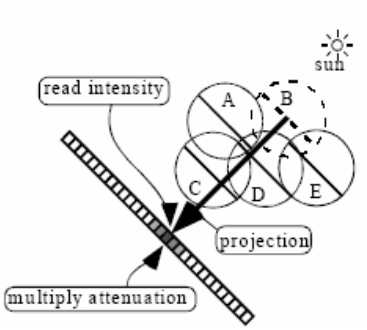

Рисунок 10

Перед началом отрисовки каждой частицы читается значение в текстуре в ее центре, и полагается степенью освещенности. После этого частица рисуется в текстуру со своим коэффициентном прозрачности. Таким образом, чем больше частиц уже отрисовалось на месте текущей частицы, тем меньше ее степень освещенности, что соответствует уменьшению интенсивности света по мере его прохождения сквозь облако.

Более сложная модель включает в себя просчет рассеяния света частицами облака и функции анизотропии для света, рассеивающегося облаком [5].

Значение освещенности определяется следующим рекуррентным соотношением, аппроксимирующим интеграл плотности облака:

$$I_k = \begin{cases} g_{k-1} + T_{k-1} \cdot I_{k-1}, 2 \le k \le N \\ I_0, k = 1 \end{cases}$$

где I_k и T_k – освещенность и прозрачность k-той частицы соответственно, g_k – интенсивность света, рассеиваемого k-той частицей, I_0 – начальная освещенность, k принимает значения от 1 до N, где N – общее количество частиц в облаке.

Такие операции тоже сводятся к отрисовке частиц с использованием стандартных возможностей смешивания современных графических ускорителей, остается только считывать значение из текстуры перед отрисовкой каждой частицы (это нельзя сделать одной операцией после отрисовки всего облака, так как искомые значения в текстуре изменяются сразу после отрисовки следующей частицы).

К сожалению, это совершенно неэффективный метод для современных графических ускорителей. Есть несколько важных факторов, объясняющих, почему такой метод показывает очень плохую производительность.

1. Современные ускорители становятся все более глубоко конвейеризируемыми. Это означает, что для эффективной работы ускорителя нужно подавать на вход большие куски данных (сотни, а лучше тысячи треугольников). Любая остановка и смена установок конвейера – дорогая операция. В этом же алгоритме фактически нужно останавливать конвейер после отрисовки каждой частицы (2 треугольника).

В этом случае накладные расходы на запуск конвейера и обработку запроса окажутся на порядки больше фактических усилий на отрисовку этих двух треугольников.

2. Каждое чтение из текстуры, необходимое для каждой частицы не только разрушает конвейер, но и вызывает синхронизацию процессора и ускорителя, что является еще более дорогой операцией, не только в локальном, но и в глобальном смысле – в том смысле, что синхронизация в середине рендера кадра практически недопустима.

Еще одно противоречие заключается в том, что на самом деле для обсчета освещенности можно ограничиться очень маленьким разрешением текстуры и самих частиц (для вполне приемлемого качества достаточно текстуры размером 32x32 или 64x64 пикселя), то есть сама по себе отрисовка требует очень мало ресурсов.

Предлагаемое решение этих всех противоречий – просчитывать освещенность в облаке на процессоре, без использования видеокарты. В этом случае мы можем ограничиться одним байтом для хранения точности значения освещенности в текстуре и выполнять операции смешивания на процессоре, что при данных разрешениях вполне приемлемо по производительности. Такое одноканальное смешивание простых частиц может быть эффективного реализовано с использованием набора мультимедийных инструкций MMX, SSE и пр., и все необходимые данные вмещаются в объем кэша первого уровня практически всех современных процессоров, что исключает один из самых распространенных факторов, влияющих на производительность.

Кроме того, можно разделять отрисовку по нескольким кадрам, так как обычно просчет освещенности нужен далеко не на каждом кадре.

Проведенные тесты показали, что разница в скорости между имплементациями, реализованными на процессоре и на ускорителе, достигает двух порядков.

3.3.3 Процедурная генерация облаков.

Как уже было упомянуто, проблемы генерации формы облаков заключаются либо в ручном выстраивании частиц (либо более грубых объемов, по которым создаются частицы)[6], либо ресурсоемким просчетом физических моделей.

Предложение авторов заключается в том, чтобы использовать возможности шума Перлина для создания частиц облака.

В алгоритме шума Перлина нет никакого ограничения размерности, он может быть сколь угодно больших измерений. Результаты трехмерного шума Перлина можно с успехом использовать для создания формы объемным облакам. В этом случае яркость выступает размером частицы и может служить ключом к выбору текстуры для этой частицы, что увеличивает возможности влияния на форму облака, а прозрачные места в текстуре означают отсутствие частиц.

Кроме того, нужно добавить случайное смещение в радиусе одного «пикселя» к координате частицы, чтобы избежать заметной дискретности координат частиц.

Результат показан на рисунке 11.

Рисунок 11: Облака, полученные на основе шума Перлина

Использование шума Перлина привносит все свои возможности и достоинства в алгоритм генерации – высокая скорость просчета, возможность управлять плотностью облаков, эффекты клубящихся облаков, постепенного проявления и так далее. Для динамической картины неба с успехом можно использовать четырехмерный шум Перлина, где еще одной координатой выступает время. Обратим внимание, что управление облачностью полностью независимо от такого изменения картины неба, что позволяет реализацию предсказуемых погодных явлений.

3.3.4 Отображение облаков.

Так как каждое облако состоит из большого количества частиц, каждая из которых занимает существенную площадь на экране (рис. 7), очевидно, что требования к заливке на ускорителе очень велики – каждая точку на экране, где есть облака (от половины экрана и больше) рисуется столько раз, сколько частиц проходит луч из этой точки. В тоже время, как уже было сказано ранее, отображение облаков не должно занимать значительную часть ресурсов генерации кадра.

Естественным решением такой проблемы является применение импостеров [5] - использование когерентности соседних кадров. Большинство облаков достаточно далеко от наблюдателя, чтобы их изображение на экране незначительно менялось на протяжении соседних кадров. В этом случае возможно заранее нарисовать облако в некоторую текстуру, и потом выводить эту текстуру на экран, не перерисовывая облако каждый кадр. Разумеется, эту текстуру нужно постепенно обновлять по мере движения наблюдателя, но типично гораздо реже, чем каждый кадр.

При использовании импостеров возникают несколько важных проблем, наиболее плохо освещенной в литературе является проблема упаковки и менеджмента текстурой площади для множества импостеров разных размеров. Нужно сказать, что такие проблемы становятся важными только при массовом использовании импостеров, достигающих большого размера на экране, так как именно тогда проявляется необходимость

нетривиального менеджмента площади текстуры, хранящей импостеры.

Авторы предлагают несколько алгоритмов менеджмента памяти для импостеров, выбор между которыми зависит от особенностей и требований к работе конкретного приложения.

3.3.5 Методы хранения импостеров напрямую.

Методы подразумевают хранение импостеров как прямоугольников различной величины внутри большого прямоугольника – текстуры, хранящей импостеры. Задача расположения прямоугольников внутри некоторого прямоугольного поля так, что стороны прямоугольников параллельны сторонам поля, называется двумерной прямоугольной упаковкой (*2d bin packing)*, и является NP-полной [7]. Существуют различные эвристики и приближенные алгоритмы, работающие за линейное время от количества объектов.

Но проблема хранения импостеров не ограничивается оптимальной упаковкой, так как из-за постоянных обновлений импостеров постоянно изменяются их размеры. Задача становится похожа скорее на организацию менеджера памяти, чем на оптимальную упаковку, так как времена жизни импостеров могут сильно различаться.

Предлагается два алгоритма организации такого менеджмента – с наличием дефрагментации и нет. В случае использования дефрагментации вводится дополнительный параметр – количество позволенных вызовов процедуры дефрагментации в единицу времени.

Основные шаги алгоритмов таковы:

1. Изначально расположить импостеры в текстуре методами прямоугольной упаковки.
2. При обновлении импостера освободить площадь, ранее занимаемую им в текстуре, найти участок требуемого размера по критерию лучшего совпадения (*best fit)*.
3. В случае если найти участок не удается, и используется дефрагментация, проверить, не превышен ли лимит вызова процедуры дефрагментации в единицу времени. Если нет, выполнить процедуру дефрагментации. Процедура дефрагментации заключается в выполнения пункта 1 для всех существующих импостеров (обновившийся импостер рассматривается со своей новой площадью), и фактическом передвижении импостеров на их новые места. Если дефрагментация прошла успешно, закончить процесс обновления.
4. Если дефрагментация не используется или не была успешной, производится поиск максимального участка с теми же пропорциями, что и участок требуемого размера.

Приведенная схема в первую очередь обеспечивает гарантированную стабильность и производительности, и лишь во вторую – качество изображения, что чаще всего и является расстановкой приоритетов при отрисовке облаков.

Все алгоритмы, использующиеся для поиска участков для импостеров (кроме дефрагментации), работают за линейное время от площади текстуры, хранящей импостеры. Часто бывает выгодно округлить размеры импостеров до некоторого предела, много меньшего типичного размера импостера (например, 16) для уменьшения времени поиска.

Критерием использования дефрагментации (что позволяет достичь более высокого качества за счет более редкой необходимости уменьшать размер участка в пункте 4) являются требования к дополнительной текстурой памяти, необходимой для отрисовки фрагментов, изменивших свое положение. Впрочем, эта дополнительная текстурная площадь необходима только как промежуточное хранение данных, и если в приложении уже используется какой-то массив данных подходящего размера, обновляемый каждый кадр (типичным примером могут служить текстуры карты глубины или текстуры для пост-процессинга изображения), то возможно использование дефрагментации без дополнительных затрат текстурной памяти.

3.3.6 Методы, включающие разбиение импостера.

Другой подход к хранению импостеров состоит в разделении импостеров на более мелкие фрагменты, лучше поддающиеся упаковке. Такими фрагментами могут выступать квадратные участки размера, много меньшего размера импостеров (можно ориентироваться на значения 16x16 и 32x32). Очевидно, эти квадраты можно тривиально упаковать в текстуре, хранящей данные, без всяких потерь. Накладные расходы на отрисовку импостера, состоящего из многих квадратов практически пренебрежимо малы, так как процесс отрисовки ограничен скоростью заливки.

С использованием этого метода текстурная площадь текстуры, хранящей данные, используется наиболее эффективно, и необходимо только чтобы суммарная площадь, необходимая для импостеров, не превышала площади текстуры, что легко контролируется.

Среди недостатков этого метода можно выделить несколько позиций:

- Кроме отрисовки импостера, нужно еще разбить его на квадраты отдельной отрисовкой.
- Существуют накладные расходы текстурной памяти, появляющиеся, во-первых, из-за округления размеров импостера размерами квадрата, которые тем больше, чем больше размер квадрата, а во-вторых, из-за необходимой однопиксельной каемки на границе квадрата, необходимой для корректности билинейной фильтрации, которая тем больше, чем размер квадрата меньше. Таким образом, минимизация накладных расходов текстурной площади определяется балансом выбора размера квадратов.
- После долгой работы в таком режиме, текстура очень «замусоривается» - квадраты, принадлежащие одному импостеру, оказываются все дальше друг от друга, что может привести к падению быстродействия из-за меньшей эффективности текстурного кэша. В этом случае может оказаться полезным рассмотреть возможность дефрагментации.

Выбор между предложенными алгоритмами зависит от приоритетов между скоростью отрисовки, качеством картинки, максимального количества облаков в сцене,

требованиями к используемой текстурной памяти и так далее, что является особенностями работы конкретного приложения.

3.4 Результаты

Ниже приведены примеры использования такого метода в двух системах отображения с разными требованиями к визуализации облаков – авто- (Рис.11) и авиасимуляторе (рис. 12).

Рисунок 11

Рисунок 12

4. МОДЕЛЬ АТМОСФЕРНЫХ УСЛОВИЙ.

Атмосферные условия задаются рядом парамеров, описанных выше – цвет неба, цвет и границы тумана на небе, цвет и границы тумана действующего на луну и солнце, прозрачность звед, параметры источников освещающих облака. Кроме того, открытые пространства в основном отсвещается такими источниками, как солнце, луна, и небо. Для моделирования этих источников используются два направленных источника для солнца и луны, и один источник рассеянного света для моделирования освещения от неба. Цвета этих источников являются параметрами погоды. Для моделирования таких эффектов как рассеивание сфета в атмосфере использхуется туман на объектах окружения, цвет и ближняя и дальняя граница тумана так же являются параметрами погоды. Таким образом, мы получили набор параметров, которые задают конкретные погодные условия - $P_{time} = \{param_0, param_1, ...\}$, все эти параметры могут быть легко подобраны дизайнером.

Для анимации суточных изменений, создается набор из восьми наборов параметров погоды, которые задают атмосферные условия ночью, ранним утром, утром, поздним утром, днем, ранним вечером, вечером и позним вечером: $D = \{P_0, P_1, P_1, P_2, P_3, P_4, P_5, P_6, P_7\}$. Таким образом, для получения набора параметров сотвтствующего данному времени, выбираются два соседних по времени набора параметров и производится линейная интерполяция между ними. Пусть $t \in [0..1]$ - текущее время, и $t = 1$ и $t = 0$, соответствуют ночи, тогда набор параметров описывающих текущие параметры для данного времени находятся по формуле - $P(t) = D[t'] * (1-t) + D[t'+1] * t$. Для анимации изменения погодных условий создается два набора D_{bad} -для плохой погоды и D_{good} - для хорощей погоды, при этом пусть $w \in [0..1]$ - это погода, и при $w = 0$ - погода плохая, а при $w = 1$, погода хорошая, тогда набор параметров для текущей погоды и текущего времени находится из формулы:

$$P(w,t) = (D_{good}[t'] * (1-t) + D_{good}[t'+1] * t) * w +$$
$$(D_{bad}[t'] * (1-t) + D_{bad}[t'+1] * t) * (1-w)$$

5. ЗАКЛЮЧЕНИЕ

Основная проблема моделирования динамических атмосферных явлений при визуализации открытых пространств в системах визуализации реального времени - это высокая сложность подобных вычислений. Авторами предложен простой метод анимации погодных явлений и их параметризации с помощью малого набора параметров, для получения широкого спектра атмосферных условий.

Так же авторами предолжен ряд оптимизации существующих алгоритмов визуализации динамических облаков, позваляющих во-первых, уменьшить, а во-вторых, масштабировать затраты ресурсов, необходимые для отображения облаков. Такое масштабирование возможно и для требуемой текстурой памяти, и для требований к

ресурсам видеоускорителя и процессора, что позволяет применять метод визуализации облаков с помощью систем частиц в широком спектре графических приложений.

В результате проведенных иследований была создана система для визуализации динамических атмосферных явлений для отображении открытых пространств в системах визуализации реального времени. При тестировании системы на сценах содержащих 100-500 тысяч полигонов генерация одного кадра изображения занимала ≈15 мс., см рис. 12. Такой скорости в большинстве случаев оказывается достаточно для эффективного применения в системах синтеза визуальной обстановки.

В настоящее время реализованная авторами подсистема, успешно используется в ряде систем виртуальной реальности.

Рисунок 13

6. REFERENCES

[1] Schlyter, *"How to compute planetary position"* http://hem.passagen.se/pausch/comp/ppcomp.html

[2] K. Perlin, *ACM SIGGRAPH'84 "Advanced Image Synt*hesis"

[3] Terragen, http://www.planetside.co.uk/terragen/

[4] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita, *"A Simple, Efficient Method for Realistic Animation of Clouds", SIGGRAPH 2000.*

[5] M.Harris, A. Lastra, *"Real-time Cloud Rendering", EUROGRAPHICS 2000.*

[6] Niniane Wang, "*Realistic and Fast Cloud Rendering in Computer Games", SIGGRAPH 2003*

[7] А. С. Мухачева, А. В. Чиглинцев, М. А. Смагин, Э. А. Мухачева, *"Задачи двумерной упаковки: развитие генетических алгоритмов на базе смешанных процедур локального поиска оптимального решения", приложение к журналу «Информационные технологии», 2001.*

Об авторах:

Все авторы работают в лаболатории - Программных систем машинной графики ИАиЭ СО РАН.

e-mail:

Н.А. Елыков - nicolas@sl.iae.nsk.su

И.В.Белаго – bel@sl.iae.nsk.su

С.М. Козлов - smk@ sl.iae.nsk.su

С.А.Кузиковский – stas@ sl.iae.nsk.su

М.М.Лаврентьев – lavr@sl.iae.nsk.su