Генерация Изображения Морской Поверхности В Реальном Времени

Николай А. Елыков, Игорь В. Белаго, Михаил М. Лаврентьев, Юрий Ю. Некрасов
Институт Автоматики и Электрометрии СО РАН
Новосибирск, Россия

Резюме

Мы представляем технику генерации и анимации изображения морской поверхности в реальном времени. В статье описываются и сравниваются несколько техник получения модели поведения морской поверхности. Детально описан алгоритм управления детализацией геометрии в зависимости от положения наблюдатнля. Так же приведена техника генерации и анимации текстуры, накладываемой на морскую поверхность.

Ключевые слова: ocean wave refraction, triangle bin tree, view-depend mesh, frame-to-frame coherence, greedy algorithms, height fields.

1. ВВЕДЕНИЕ.

Бурный рост производительности и качества графических средств персональных компьютеров в последние годы и глобальная стандартизация графических программных интерфейсов существенно расширило применение PC в качестве аппаратной платформы для графических приложений, приложений виртуальной реальности и симуляторов.

Моделирование визуальной обстановки в настоящее время является одной из наиболее динамичных и бурно развивающихся областей в компьютерной индустрии. Программные модели виртуального мира приобретают все большую схожесть с миром реальным. Благодаря присущей ему сложности, реалистичное моделирование природных феноменов является одной из наиболее вычислительно емких областей компьютерной графики. Один из таких феноменов – это морские сцены. Данная статья посвящена исследованию методов визуальной имитации морских поверхностей в реальном времени.

Авторами статьи были исследованы и проанализированы подходы к решению проблемы и выбран метод позволяющий генерировать реалистичные изображения морской поверхности в реальном времени. Метод учитывает эволюцию морской поверхности во времени в зависимости от направления и скорости ветра, формы морского дна и береговой линии.

Описываемая методика генерации изображения морской поверхности состоит из трех частей. Первая часть представляет собой алгоритм, позволяющий получить форму поверхности моря в зависимости от времини и других факторов. Вторая часть позволяет построить оптимальную, с точки зрения качества/производительность, триангуляцию. Третья часть посвещена методам получения текстур для морской поверхности.

2. ТЕХНИКА ГЕНЕРАЦИИ И АНИМАЦИИ ФОРМЫ МОРСКОЙ ПОВЕРХНОСТИ.

Для эффективного использования алгоритм должен удовлетворять ряду требований, сформулированными, например, в [1].

Алгоритм должен:

1. генерировать волны похожие на реальные.
2. не требовать больших вычислительных ресурсов.
3. иметь простую и реалистичную анимацию.
4. обладать возможностью генерировать широкий диапазон волн от небольшой ряби до штормовых волн.
5. “уметь” моделировать такое явление как рефракция (*ocean wave refraction*), т.е. учитывать форму дна и/или берега моря.
6. предоставлять дизайнерам возможность задавать такие физические параметры, как направление и скорость ветра, линию берега и форму поверхности дна.
7. предоставлять возможность получения точки поверхности в произвольном месте (для нерегулярной треангуляции)

Поверхность моря принято задавать с помощью набора высот в узлах равномерной прямоугольной решетки (*height fields*), т.е. для точки *(x,y,z)*, лежащей на представляемой поверхности, достаточно хранить только координаты *(x,y),* а третью координату – *z,* получать с помощью функции *H(x,y,t)=z,* заданной на двухмерном наборе данных функционально или с помощью таблицы. Это достаточно общий тип данных обычно использующийся для представления ландшафтов во многих програмных приложениях, включая самолетные тренажеры (*flight simulators*) и тренажеры наземной техники (*ground vehicle simulators*). Используя *height fields* возможно абстрагировать модуль создания геометрической модели от характера поверхности, т.е. способов ее создания и управления. Этот способ имеет и ряд недостатков, например, с его помощью нельзя моделировать нависания поверхности (т.е. *H (x, y, t)* – принимет несколько значений).

Авторами данной работы были реализованны и испытаны несколько предложенных в литературе алгоритмов генерации морской поверхности, а именно:

1. Алгоритм №1. Суперпозиция синусов [1],[2],[3].

2. Алгоритм №2. Метод частотной фильтрации [2].

3. Алгоритм №3. Метод, предложенный Darwyn R. Peachey [3].

Ниже подробнее рассмотренна суть данных алгоритмов, а так же их достоинства и недостатки выявленные авторами в процессе реализации и испытаний.

2.1 Алгоритм №1. Суперпозиция синусов.

В этом методе морская поверхность задается как суперпозиция нескольких синусоидальных волн с собственной амплитудой, периодом, начальной фазой и направлением распространения:

$$z(x,y,t) = \sum_{i=1}^{n} A_i \sin(k_i x - \frac{t - t_0}{T_i}).$$

Достоинства:

- Единственным существенным достоинством данного метода является простота генерации и анимации.
- Небольшой объем данных, необходимый для работы алгоритма.
- Возможность быстрой смены амплитуды, длины, направления и скорости распространения волны.

Недостатки:

- Получаемое изображение лишь отдаленно напоминает реальность.
- Невозможно моделировать рефракцию.
- Получаемая геометрическая модель является регулярной, т.е. неестественной .

Из-за перечисленных недостатков этот алгоритм использовался лишь на стадии отладки модуля упрощения геометрической модели поверхности.

2.2 Алгоритм №2. Метод частотной фильтрации белого шума.

В качестве фильтра был использован фильтр Пирсона-Московитца [2](Pierson-Moskowitz) : $F_{pm} = \frac{\alpha g^2}{(2\pi)^4 f^5} \exp(-\frac{5}{4}(\frac{f_m}{f})^4)$,где $F_{pm}(f)$,частотный спектр, f – частота, f_m - пиковая частота, α - константа Филипса (0,0081), $f_m = \frac{0.13g}{u_{10}}$, u_{10} – скорость ветра на высоте 10 метров над поверхностью.

Модернизированный Хасельманом (Hasselman) [2] для двумерного случая фильтр принимает вид: $F(f,\theta) = F_{pm} D(g,\theta)$, где D – весовое распределение, зависящее от угла к вектору направления ветра. На рисунке 1 изображен фильтр в пространстве Фурье для скорости ветра 15 м/с, дующего под углом 60°. Используя этот фильтр, был получено изображение морской поверхности представленное на рисунке 2.

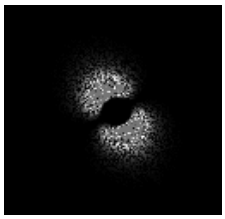

Рисунок 1. Частотный фильтр Пирсона-Московитца в пространстве Фурье.

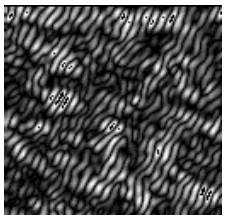

Рисунок 2. Изображение поверхности "открытого" моря, полученного методом частной фильтрации белого шума.

Для анимации поверхности могут быть использованы несколько техник управления фазой в пространстве Фурье.

В процессе реализации и тестирования авторами были выявлены следующие достоинства и недостатки:

Достоинства:

- Алгоритм генерирует достаточно реалистичную картину.
- Широкие возможности анимации.
- Получаемая геометрическая модель выглядит "случайно".

Недостатки:

- Алгоритм не учитывает форму дна и берега, т.е. предназначен только для генерации поверхности "отрытого" моря.
- Необходимо оперировать со сравнительно большими объемами данных, что в большинстве случаев ведет к потери производительности.
- Алгоритм требует значительных вычислительных ресурсов.
- Изменение скорости и направления ветра сопровождается значительными вычислениями.

2.3 Алгоритм №3. Метод D. R. Peachey.

Как говорилось выше, форма морской поверхности представляется как функция трех переменных: $z(x,y,t) = H(x,y,t)$,где x,y – обычные прямоугольные координаты в модельном пространстве, а t - время изменяемое для каждого кадра анимации. Функция *H(x,y,t)* есть сумма нескольких "длинных" гребней волн W_i с амплитудами A_i , распространяющихся в различных направлениях: $H(x,y,t) = \sum_{i=1}^{n} A_i W_i(x,y,t)$,

$W_i(x,y,t) = w_i(fraction[\theta_i(x,y,t)])$, где W_i представляет собой композицию двух функций: w_i – профиль волны (единичная периодическоя функция заданная на промежутке [0,1]) и $\theta_i(x,y,t)$ – функция фазы, состоит из двух компонент:

$$\theta_i(x,y,t) = \theta_i(x,y,t_0) - \frac{t - t_0}{T_i}$$

, временной и пространственной, где T_i – период i-ой фолны, $\theta_i(x,y,t)$ (x,y,t_0) – фаза в некоторый момент времени t_0 (пространстевенная компанента фазы). Пространственная компонента фазы позволяет ввести зависимость от поверхности дна, что делает возможным моделирование таких эффектов как рефракция волн.

Хорошим приближением для случая, когда период почти не зависит от глубины, является трансцендентное

выражение[1]: $ktanh(kh)=k_\infty$,где h – глубина в некоторой точке, k – волновое число, а k_∞ - волновое число на бесконечной глубине. Это уровнение легко решается для предельных значений h: для малой глубины (h<0.05L, где L - период волны) имеем $k = \sqrt{\frac{k_\infty}{h}}$, для глубокого моря (h>0.25L) имеем $k = \frac{k_\infty}{\sqrt{\tanh(k_i h)}}$. Соответственно находим пространственную компаненту фазы :

$$\theta_i(x_i) = \int_0^{\bar{x}_i} k_i(u)du = \sum_0^{\bar{x}_i} \frac{k_\infty}{\sqrt{\tanh(k_\infty h(x_i))}}\Delta x$$

, где осуществлен переход к численному интегрированию. (Нужно отметить, что разделение фазы на пространственную и временную компоненты позволяет выполнять численное интегрирование для данной формы дна, длинны волны и направления распространения только один раз, а не отдельно на каждом кадре).

Профиль волны - w_i(u), некоторая функция определенная на u∈[0,1), со множество значений [-1,1], которая задает форму волны. Для сохранения непрерывности необходимо чтобы $w_i(1)= w_i(0)$. Алгоритм использует два вида профилей: первый – $w_i=cos(2\pi u)$,синусоидальный для "глубокой воды", второй $w_i=8(u - 0.5)^2-1,$ более острый для прибрежных волн.

Для достижения большего реализма осуществляется линейная интерполяция между двумя этими профилями в зависимости от глубины и длинны волны. На

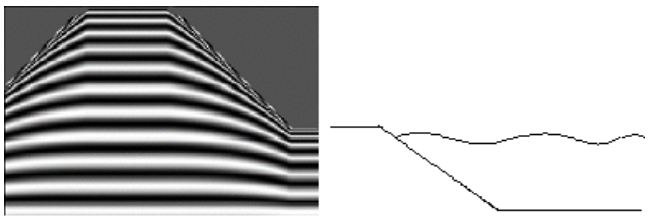

Рисунок 3. Берег с линейно изменяющейся глубиной и набегающие на берег волны (слева вид с верху, справа срез). Хорошо видна рефракция волн.

рисунке 3 представлен пляж с набегающими на него волнами, причем береговая линия представленна ломанной, а глубина меняется линейно с увеличением расстояния от берега

В процессе реализации и тестирования авторами были выявлены следующие достоинства и недостатки:

Достоинства:

- Генерируемая геометрическая модель выглядит реалистично.
- Простота анимации.
- Учитывает форму дна и берега.

Недостатки:

- Смена длины и направления распространения волны сопровождается сложными вычислениями.
- Не учитывает такое явление как дифракция (например, за скалой находящейся в открытом море образуется резкий след).
- Необходимо манипулировать сравнительно большими объемами данных, что в большинстве случаев ведет к потере производительности.
- Получаемая поверхность излишне регулярна.

2.4 Выводы.

При реализации и тестировании вышеперечисленных алгоритмов авторами данной работы было установлено, что все алгоритмы обеспечивают быстрый пространственный поиск и позволяют генерировать поверхность с различными уровнями детализации; алгоритмы 1 и 3 легко позволяют привязать текстурные координаты к создаваемым моделям. Модели, создаваемые алгоритмами 1 и 3, выглядят регулярно (впрочем, алгоритму 3 это свойство присуще в меньшей степени), тогда как поверхность созданная алгоритмом 2 выглядела довольно "случайно". Алгоритмы 2 и 3 предъявляют высокие требования к скорости выполнения операций с плавающей запятой.

Проведенный анализ показал, что метод Peachey (алгоритм 3), несмотря на присущие ему недостатки, обладает наилучшими визуальными качествами наряду с приемлемой производительностью, что и позволило выбрать его в качестве базового варианта для использования в качестве подсистемы генерации и анимации формы водной поверхности.

3. ОПТИМАЛЬНАЯ ТРИАНГУЛЯЦИЯ МОРСКОЙ ПОВЕРХНОСТИ В ЗАВИСИМОСТИ ОТ ПОЛОЖЕНИЯ НАБЛЮДАТЕЛЯ.

При разработке алгоритмов учитывалась необходимость в реальном времени генерировать и визуализировать большие площади морской поверхности, поэтому была разработана подсистема управления детализацией в зависимости от положения камеры и её ориентации. Морская поверхность, в силу своей природы, не разбивается на блоки, чья детализация может настраиваться независимо. Поверхность должна остаться неразрывной, при этом та часть поверхности, которая видна и находится в близи от наблюдателя должна представляться боьшим количеством треугольников.

На подсистему управления детализацией можно наложить несколько формальных требований [4]:

i. Триангуляция должна удовлетворять требованию минимальной избыточности, т.е. важные с точки зрения текущего кадра части морской поверхности должны быть представлены большим количеством треугольников.

ii. В любом состоянии геометрическая модель и ее компоненты, такие как вершины и треугольники, должны быть лекго доступны для других частей приложения, например для подсистемы проверки столкновений (*collision detection*). Так же необходим быстрый пространственный поиск вершин и треугольников.

iii. Динамические изменения сетки, влекущие за собой изменение параметров поверхности или ее геометрии, не должны значительно влиять на производительность.

iv. Высокочастотные особенности поверхности, такие как локальные выпуклости или вогнутости, не должны вести к глобальному усложнению модели.

v. Небольшие изменения положения наблюдателя не должны вести к значительному изменению геометрии модели, во избежание значительных изменений получаемого изображения кадр от кадра и для сохранения скорости генерации одного кадра почти постоянной.

vi. Алгоритм должен иметь возможность заранее задавать сложность получаемой модели (т.е. иметь возможность управлять изменениями качества).

Если по данным, предоставляемым подсистемой, задающей форму морской поверхности, строить геометрическую модель используя равномерную триангуляцию, то получаемая геометрическая модель не отвечает требованиям минимальной избыточности. С помощью подсистемы, производящей упрощения геометрической модели можно значительно уменьшить количество треугольников в сцене без значительной потери визуального качества, что в результате позволяет уменьшить общее время, уходящее на обработку одного кадра.

На рисунке 4 представлена равномерная триангуляция сетки 9x9, содержащая 256 треугольников. На рисунке 5 показана сетка после упрощения (более чем в два раза сокращено общее количество треугольников в сцене, за счет уменьшения количества треугольников, не попадающих в пирамиду видимости, и количество “дальних” треугольников).

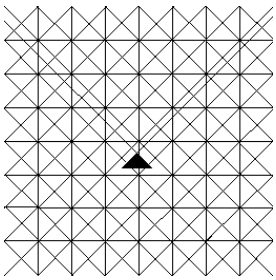

Рисунок 4. Равномерная треангуляция.(содержит 256 треугольников)

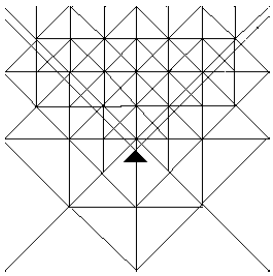

Рисунок 5. После упрощения триангуляция содержит всего 100 треугольников.

В литературе предложенно несколько техник для управления уровнем детализации. В [5], предложен алгоритм илпользующий изменяющюиеся сетки (*progressive meshes*), это относительно новая и хорощая техника для добавления треугольников в нерегулярную сетку (*arbitrary mesh*) при изменении детализации сцены, но к сожалению она излишне сложна и требует значительное количество оперативной памяти. В [6] представленна структура данных *Quad Tree*, использующаяся для представления частей поверхности. *Quad Tree* рекурсивно разбивается на треугольники для создания подходящей апроксимации карты высот. Это очень простая и эффективная техника, однако в качестве основы для подсистемы управления уровнем детализации использовался предложенный в работе [7] алгоритм, он имеет много общего с *Quad Tree*, но он более прост и гибок. Рассмотрим его более детально.

3.1 Представление геометрической модели.

Основой структурой используемой модулем управления детализацией является двоичное дерево треугольников (*triangle bin tree*). На рисунке 6 показаны несколько первых уровней этого дерева.

Корнем дерева является прямоугольный

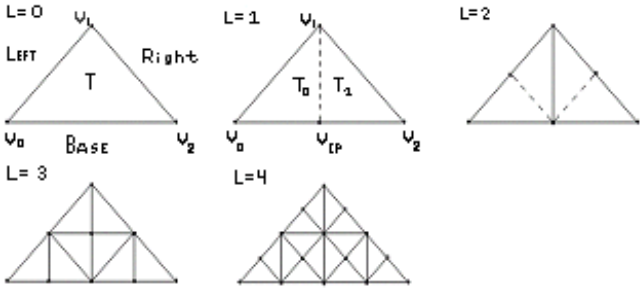

Рисунок 6. Уровни L = 0 – 5 бинарного дерева

равнобедренный треугольник $T = (v_1,v_0,v_2)$ это самый низкий – нулевой уровень детализации. Для любого треугольника ведем обозначения: ребро (v_1, v_0) – левое (*left edge*), (v_0,v_2) – правое ребро (*right edge*), (v_2,v_1) – ребро основания (*base edge*). Соответственно вводятся обозначения для соседних треугольников – правый (*right neighbour*), левый (*left neighbour*) и сосед основания (*base neighbour*). Надо заметить, что у треугольника с уровнем детализации L могут быть соседи только уровней $L,L+1,L-1$. Следующий уровень дерева получается, как показано на рисунке, разбиением треугольника T на два новых прямоугольных равнобедренных треугольника – потомков: $T_0=(v_0,v_a,v_1)$ и $T_1=(v_2,v_a,v_0)$, соответственно с уровнем детализации L=1. Остальные уровни получаются рекурсивно. Геометрическую модель поверхности моря составляют только треугольники, соответствующие самым нижним узлам двоичного дерева треугольников, которые не имеют потомков, см. рисунок 7.

Замечание: с каждым узлом дерева можно хранить

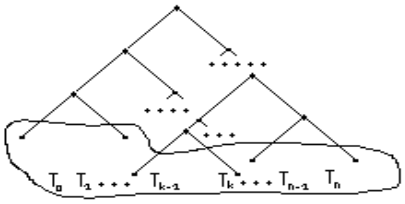

Рисунок 7. Двоичное дерево треугольников. В рамку заключены треугольники, составляющие геометрическую модель.

“оболочки” (bounding boxes) соответствующего ему треугольника, что позволит существенно убыстрить следующие операции: проверку пересечения (collision detection) других объектов виртуальной среды с морской

поверхностью, пространственный поиск треугольников, вершин и ребер (fast spatial indexing), отсечение треугольников, не попадающих в пирамиду видимости (view-frustum culling).

3.2 Динамические преобразования геометрической модели.

Введем операции Split и Merge для модификации бинарного дерева:

- Split: операция по разбиению некоторой висячей вершины двоичного дерева некоторого уровня детализации L и соответствующего ей треугольника T_0 на две новые вершины уровня $L+1$ и соответствующих треугольников T_1 и T_2. См. рисунок 8.

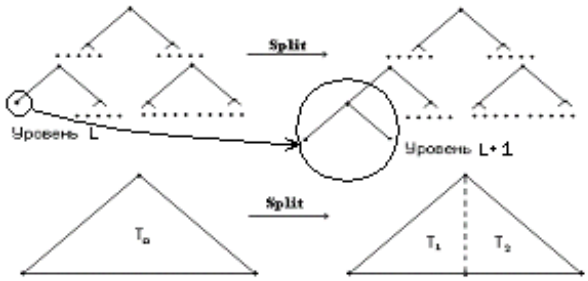

Рисунок 8. Операция Split

В результате появляется новая вершина. Если разбиваемый треугольник T_0 имел соседа по основанию – T_E, то, чтобы избежать разрывов и нестыковок (*T-vertices*), мы вынуждены применить операцию Split к треугольнику T_E. Такой процесс называется Forced Split и он может

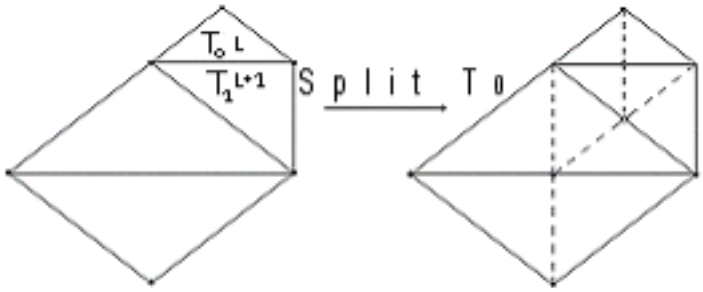

Рисунок 9. Разбиение одного треугольника ведет к появлению десяти новых.

затрагивать несколько треугольников См. рисунок 9.

Замечание: Глубина Forced Split ограниченно максимальной глубиной дерева треугольников (которое в свою очередь ограничено разрешением height field на котором строится дерево). Допустим, разбиваемый треугольник T_0 принадлежит уровню L бинарного дерева, тогда его сосед T_E к которому мы вынуждены для сохранения непрерывности применить операцию Split может лежать на уровене L или L-1. В случае, когда уровень равен L-1 придется повторять операцию для еще одного треугольника, а в случае уровня L процесс разбиения заканчивается.

- Merge: Введем новую геометрическую структуру “ромб” (*diamond*) –(T_L.T_R) См. Рис. 10, состоящий из двух узлов двоичного дерева треугольников (соответствующие им треугольники T_L, T_R) имеющих одинаковый уровень детализации таких, что, во-первых, их дети не имеют потомков, т.е. соответствующие детям треугольники - T_0,T_1,T_3,T_4, содержатся в геометрической модели и во-вторых треугольники T_0,T_1,T_3,T_4 образуют ромб со сторонами из ребер основания соответствующих треугольников. Операция Merge уничтожает узлы T_0,T_1,T_3,T_4 бинарного дерева и соответствующие им треугольники удаляются из геометрической модели, а вместо них появляются треугольники T_R и T_L с более низким уровнем детализации.

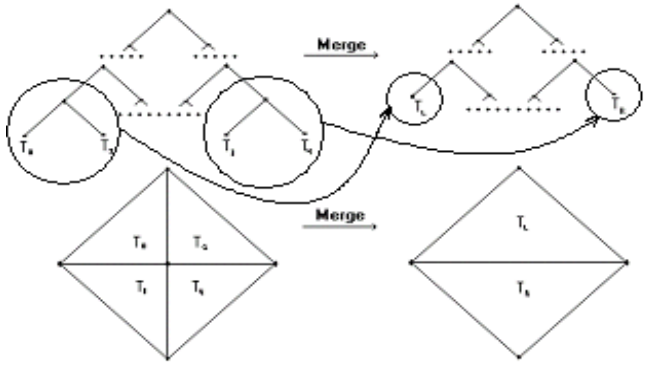

Рис10. Операция Merge

Итак, мы имеем операции позволяющие гибко изменять уровень детализации сцены. Рассмотрим алгоритм, который управляет процессом разбиения и слияния треугольников для получения наиболее подходящей степени детализации. Присвоим каждому треугольнику в сцене некоторый приоритет и, начиная с некоторой триангуляции, будем производить операцию Forced Split для наиболее приоритетного треугольника. Как будет показано далее такой “жадный” алгоритм (*greedy algorithm*) строит последовательность триангуляций, которая, при условии, что приоритеты назначаются монотонно (т.е. приоритет потомка при разбиении меньше приоритета родителя), минимизирует максимальный приоритет сцены. Операция Merge позволяет “жадному” алгоритму продолжать работу при изменении приоритетов в сцене с предыдущей оптимальной триангуляции и, таким образом, пользоваться преимуществами покадровой когерентности.

3.3 Алгоритм производящий оптимальную триангуляцию с наперед заданным количеством треугольников.

Предположим, что каждому треугольнику в триангуляции **T** сопоставлен монотонный приоритет P(T)∈[0,1].

Алгоритм I.

Пусть T начальная триангуляция;

Сопоставим каждому T⊆**T** приоритет P(T)∈ [0,1];

Пока (T не достаточного размера или точности) {

Определить треугольник T с максимальным приоритетом в триангуляции **T**;

Force Spilt T;

}

Т.к. глубина **Forced Split** ограничена максимальной глубиной двоичного дерева треугольников, то количество операций **Split,** производимых алгоритмом, пропорционально количеству треугольников N в конечной триангуляции. Для ускорения процесса поиска имеет смысл поддерживать очередь Q_s из треугольников, составляющих текущую геометрическую модель и отсортированных в порядке убывания приоритетов. Докажем, что данный алгоритм генерирует оптимальную триангуляцию с минимальным максимальным пироритетом на каждом шаге. Рассмотрим любую другую триангуляцию T', которая имеет более низкий максимальный приоритет, чем T. Очевидно, что T', должна содержать только потомков всех треугольников, к которым была применена операция Forced Split при формировании T. Поскольку операция Forced Split делает только минимальные необходимые изменеия, чтобы сохранить непрерывность, то T' не может содержать никаких предков к треугольникам из T. Наконец, поскольку T' имеет более низкий приоритет, она должна содержать только потомков по крайней мере одного из треугольников T. Поэтому триангуляция T' должна содержать больше треугольников чем T, значит T – оптимальная триангуляция.

3.4 Модификация *Алгоритма I* для случая изменяющихся приоритетов.

Пусть теперь для каждого кадра f∈N, каждому треугольнику в триангуляции **T** сопоставлен изменяющийся во времени приоритет $Pf(T)\in[0,1]$. Задача – получить последовательность триангуляций (T_0,T_1,T_2…), для каждой из которых максимальный приоритет треугольников сцены - минимален. Если приоритеты кадр от кадра меняются медленно, то любые две последовательные триангуляции T_f и T_{f-1} оказываются подобны друг к другу. Таким образом, для значительного ускорения работы ***Алгоритма I*** для последовательности кадров триангуляцию T_f лучше получать из триангуляции T_{f-1}. Это достигается путем создания и управления второй очередью приоритетов Q_m , которая содержит в себе все "ромбы" $D = (T_L, T_R)$ в текущей триангуляции, упорядоченные по приоритетам, которые для некоторого ромба (T_L, T_R) вычисляются по правилу $P_f(D) = \max(P_f(T_L),P_f(T_R))$.

Алгоритм II.

Если ($f = 0$) {

Отчистить Qs и Qm;

Положить **T** – начальная триангуляция;

Добавить в Qs все треугольники из **T** и в Qm все "ромбы" в **T**;

Вычислить все приоритеты для Qs и Qm;

} **иначе** {

$\mathbf{T} = \mathbf{T}_f$;

Обновить приоритеты в Qs и Qm;

}

Пока (**T** не подходящего размера или точности или максимальный приоритет треугольников в **T** больше чем минимальный приоритет "ромбов") {

если (**T** содержит слишком много треугольников) {

Определить минимальный по приоритету "ромб" – D в **T** ;

Merge D;

} **иначе** {

Определить максимальный по приоритету треугольник T в **T**;

Force Split T;

}

}

Присвоить $T_f = T$;

Общее количество операций Split и Merge, выполняемое ***Алгоритмом II*** при генерации одного кадра, пропорционально ΔN – количеству отличающихся треугольников в T_f и T_{f-1}. ***Алгоритм II*** производит триангуляцию T_f имеющую тот же приоритет что и триангуляция, получаемая ***Алгоритмом I*** примененным к начальной триангуляции, но из-за межкадровой когерентности (*frame-to-frame coherence*) достигает этого за меньшее количество операций Split и Merge .

3.5 Приоритеты.

В настоящее время существует много способов выбора для некоторого блока геометрической модели соответствующего уровня детализации в зависимости от расстояния до него, ориентации, и геометрических особенностей упрощаемой поверхности и т.д. [4], [6], [7].

Т.к. поверхность, к которой применяется данный алгоритм, достаточно однородная (морская поверхность), то в качестве приоритета для некоторого треугольника берется величина обратно пропорциональная расстоянию до герметического центра данного треугольника, а треугольникам не попадающим в пирамиду видимости присваевается нулевой, т.е. минимальный приоритет. Таким образом, в первую очередь разбиваются наиболее близкие к камере попадающие в пирамиду видимости треугольники. Существуют и другие разновидности измерения приоритетов, которые можно использовать с данным алгоритмом:

- *Уменьшение детализации задних треугольников (Back faces)"* – храня вместе с двоичным деревом треугольников разброс нормалей к граням, можно принудительно уменьшать приоритет треугольников того поддерева, которое состоит целиком из задних треугольников.
- *"Искажения нормалей"* - для поверхностей, освещение которых задается нормалями интерполированными по вершинам, для правильного освещения, необходимо увеличить приоритет треугольникам имеющим больший разброс нормалей в вершинах.
- *"Силуэты"* – для правильного получения силуэтов, необходимо увеличивать приоритеты треугольникам находящимся на краю поверхности.

- *"Атмосферная видимость"* - приоритеты могут быть снижены в условиях плохой видимости (во время тумана, дождя или ночью).
- *"Позиционирование объектов"* – чтобы корректно поставить некоторый объект на поверхность, приоритеты треугольников под каждым из устанавливаемых объектов должны быть искусственно увеличены.

3.6 Выводы.

Перечисленные алгоритмы были програмно реализованны в виде встраиваемого модуля для генерации морской поверхности в реальном времени. На машине с процессором Pentium II 300 Mhz 128 Mb RAM, при движении камеры на небольшой высоте над поверхностью со средней скоростью, на процесс генерации одного кадра содержащего 1500 треугольников, в который входят генерация и упрощение геометрической модели, вычисление и сглаживание нормалей, задание текстурных координат и подготовка к выводу изображения средствами DirectX, в среднем уходило ≈9 ms, и из них только ≈3 ms занимала работа подсистемы управлением уровнем детализации.

Для ускорения работы алгоритма использовалось несколько оптимизаций:

- Создание и управление очередей треугольников и ромбов, отсортированных соответственно в порядке возрастания и убывания приоритетов, позволяющих осуществлять быстрый поиск треугольников с максимальным и ромбов с минимальным приоритетами, использующихся в работе описанных алгоритмов.
- Техника отложенных вычислений:
  1. Вычисление приоритетов ко всем треугольникам в сцене на каждом кадре требует значительных вычислительных затрат, пользуясь межкадровой когерентностью сцены (в случае плавного передвижения камеры приоритеты треугольников кадр от кадра меняются мало) это время можно уменьшить. Для этого поддерживается очередь, элементами которой являются группы треугольников, и на каждом кадре вычисляются приоритеты треугольников только из текущей группы, а затем статус текущей передается следующей группе треугольников.
  2. ***Алгоритм II*** не гарантирует постоянство времени вычислений, но благодаря тому, что он может начинать свою работу с произвольной триангуляции, то по истечению некоторого определенного времени работы алгоритм может быть прерван. Конечно, полученная триангуляция не будет оптимальной для данного кадра (положения камеры), однако, благодаря тому, что шаги разбиения/слияния выполненяются в уменьшающемся порядке важности, частично сделанная работа оптимальна в том смысле, что триангуляция стала астолько близка к оптимальной, насколько позволило время. Техника отложенных вычислений позволяет поддерживать скорость генерации одного кадра почти константной.
- Для еще большего ускорения работы алгоритма предусмотрена возможность отказаться от анимации треугольников заднего плана.
- Оптимизация программ для процессоров Pentium® II и Pentium Pro [8].

В результате проведенных оптимизаций скорость работы модуля выбора уровня детализации уменьшилось, а алгоритм получил возможность при заданном количестве треугольников в сцене менять время генерации данных для визуализации одного кадра, тем самым, освобождая время для визуализации остальных частей системы визуализации виртуальной реальности.

4. ТЕХНИКА ГЕНЕРАЦИИ И АНИМАЦИИ ТЕКСТУР.

Если рассматривать примеры различных природных материалов и явлений, таких как горы, галька и песок, камни, облака, трава в поле, волны на воде, огонь, движение муравьев, рисунок на мраморе, ветви деревьев, то все они обладают двумя общими ("фрактальными") свойствами:

- Свойство "самоподобия", т.е. при различном увеличении объекты выглядят почти одинаково (подобно).
- Объекты задаются регулярной структурой с наложением на нее некоторой "случайности".

Для придания подобной "случайности" работе программы обычно используется датчик случайных чисел, но, к сожалению, обычные датчики случайных чисел являются слишком грубыми (резкими) и не годятся для таких целей, кроме того, обычные датчики случайных чисел не обладают свойством "самоподобия".

Для анимации текстурных координат некоторой искусственной текстуры, представляющей поведение поверхности воды, и для придания естественности морскому дну может применяется шумовая функция Перлина (*Perlin Noise function*) [9],[10],[11].

Функция шума Перлина является мощным инструментом для придания виртуальному миру большей естественности, причем речь идет не только о процедурных текстурах. Область ее примененния значительно шире, например, функцию Перлина можно применять для "оживления" движения цифровых персонажей по виртуальному миру, для генерации исскуственных ландшавтов, для создания и анимации 3D облаков и т.д. При реализации программы функция Перлина использовалась для генерации поверхности морского дна и анимации координат текстуры накладываемой на морскую поверхность. Использование функции Перлина позволило генерировать более реалистичную морскую поверхность.

5. ЗАКЛЮЧЕНИЕ.

Авторами данной работы были программно реализованы все перечисленные модели поведения и алгоритмы генерации геометрической модели морской поверхности, а также алгоритмы анимации и генерации текстуры. Использованные алгоритмы были оптимизированы для получения возможно большего быстродействия, и в случае возможности выбора реализации были выбраны те

способы, которые обеспечивали наилучшее визуальное качество наряду с хорошей производительностью. На основе проведенных исследований авторами был реализован встраиваемый модуль для визуализации морской поверхности, позволяющий пользователю регулировать различные параметры, влияющие на визуальное качество получаемых изображений и на скорость генерации одного кадра (количество треугольников в сцене, степень детализации, время работы подсистемы упрощения геометрической модели и пр.), а также позволяющий регулировать такие "физические" параметры как скорость и направление ветра, рельеф морского дна и линию берега, ширина полосы прибоя и пр. Для тестирования встраиваемого модуля была написана программа, визуализирующая кусок океанской поверхности 3000x2000 $м^2$ см. рисунок 11. При тестировании на машине Pentium II - 400MHz; RAM - 128Mb; Intel I740 video card, 4Mb video RAM, работающей под управлением операционной системы Windows 98, программа тратила на генерацию одного кадра изображения анимированной морской поверхности ≈17 мсек. (≈55 кадров в секунду), причем ≈9 мсек. уходило на создание физической и соответствующей ей геометрической модели поверхности, состоящей из 1500 треугольников, на анимацию текстурных координат и на подготовку вывода изображения средствами DirectX, т.е. на работу встраиваемого модуля. Такой скорости в большинстве случаев оказывается достаточно для эффективного применения библиотеки в системах синтеза визуальной обстановки.

6. ЛИТЕРАТУРА:

[1] Alain Fournier, William T. Reeves *"A Simple Model of Ocean Waves"* Computer Graphics (Proc. SIGGRAPH'86) Aug. 1986 pp. 75-81

[2] Gray A. Mastyn, Peter Watterberg and John Mareda *"Fourier Synthesis of Ocean Scenes."* IEEE Computer Graphics and Applications Mar.1987 pp.16-23

[3] D.R. Peachey *"Modeling Waves and Surfaces."* Computer Graphics (Proc. SIGGRAPH'86) Aug. 1986 pp. 65-74

[4] Peter Lindstrom, David Koller, William Ribarsly, Larry F. Hodges, Nick Faust, Gregory A. Turner *"Real-Time, Continuous Level of Detail Rendering of Height Fields"* SIGGRAPH-96

[5] Huges Hoppe "*Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering.*" Microsoft Research

[6] Peter Lindstrom, David Koller, Larry F. Hodges, William Ribarsky, Nick Faust and Gregory Turner *"Level-of-Detail Management for Real-time Rendering of Photextured Terrain."* Graphics, Visualization and Usability Center, Georgia Tech TR-95-06 1995

[7] Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C.Miller, Charles Aldrich and Mark B. Mineev-Weinstein *"ROAMing Terrain: Real-time Optimally Adapting Meshes."* IEEE Visualization Nov 1997 pp.81-88

[8] Maggie Auerbach, Adina Hagege, Esti Lederer, Vasily Balashov, Ekaterina Voloschenko, Dmitry Kozlov "*Оптимизация программ для процессоров Pentium® II и Pentium Pro*." Intel Corporation © 1996-1998

[9] "*Using MMX™ Instructions for Procedural Texture Mapping. (Based on Perlin's Noise Function).*" Intel Developer Relations Group © 1996

[10] "*Microsoft DirectX 6.0 Programmer's Reference*." Microsoft Corporations © 1998

[11] Dan Goehring, Or Gerlitz "*Advanced Procedural Texturing Using MMX™ Technology.*" Intel Corporation © 1997

Рисунок 11. Пример визуализации морской поверхности.

Авторы:

Николай А. Елыков – магистрант НГУ
E-mail: Nicolas@sl.iae.nsk.su

Игорь В. Белаго – н.с. ИАиЭ СО РАН

Михаил М. Лаврентьев – зав. лаб. ИАиЭ СО РАН

Юрий Ю. Некрасов – н.с. ИАиЭ СО РАН

Abstract

We propose a technique to generate and animate an ocean surface image in real time. Several techniques to obtain an ocean surface behavior model are described and compared. Geometry detail selection algorithm, based on the camera position, is described in detail. A technique to generate and animate the texture used for the ocean surface is also provided.

Keywords: ocean wave refraction, triangle bin tree, view-depend mesh, frame-to-frame coherence, greedy algorithms, height fields.

Authors:

Nikolay A. Elykov – master student, NSU
E-mail: **Nicolas@sl.iae.nsk.su**

Igor V. Belago – scientific researcher, IAE SD RAS

Michail M. Lavrentiev – head of laboratory, IAE SD RAS

Youry Y. Nekrasov – scientific researcher, IAE SD RAS