Построение на GPU в масштабе реального времени адаптивной модели рельефа Земли на основе эллипсоида вращения

П.Ю. Тимохин, М.В. Михайлюк, А.В. Мальцев

***Аннотация*—В статье рассматривается задача моделирования в масштабе реального времени рельефа Земли на основе детализированных глобальных карт высот, заданных относительно эллипсоида вращения WGS-84. Предлагается технология адаптивной тесселяции треугольных патчей на графическом процессоре (GPU), которая обеспечивает построение в реальном времени сложных полигональных моделей рельефа Земли. Технология включает в себя этап разбиения видимого эллипсоида на крупные патчи (низкого уровня детализации) и этап их тесселяции до треугольников модели рельефа, выполняемой параллельно и независимо друг от друга на ядрах GPU. В работе предложены новые, распределенные методы и алгоритмы отбора патчей, необходимых для визуализации текущего кадра, повышения их уровня детализации в соответствии с картой высот и разрешением экрана и преобразования в полигоны рельефа. Новизна работы состоит в том, что тесселяция треугольных патчей эллипсоида выполняется на основе оригинальной схемы, которая позволяет эффективно локализовывать участки рельефа, требующие повышенной полигональной детализации. Это существенно уменьшает время построения модели рельефа и повышает качество создаваемых изображений.**

Разработанные технология, методы и алгоритмы были реализованы в программных модулях и апробированы в системе визуализации виртуальной поверхности Земли. Моделирование и визуализация рельефа Земли выполнялись с детализированной текстурой подстилающей земной поверхности и расчетом освещенности с учетом атмосферы. Апробация проводилась в составе космической виртуальной сцены, включающей подробную полигональную модель Международной космической станции (МКС). Полученные результаты подтвердили адекватность предложенного решения поставленной задаче и его применимость для построения космических видеотренажерных комплексов, систем виртуального окружения, виртуальных лабораторий и др.

***Ключевые слова*—визуализация, Земля, рельеф, GPU, виртуальная модель, эллипсоид вращения, реальное время, тесселяция, распределенные вычисления.**

Статья получена 20 сентября 2019.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 17-07-00243.

П.Ю. Тимохин, ФГУ ФНЦ НИИСИ РАН, старший научный сотрудник (e-mail: tpyu@yandex.ru).

М.В. Михайлюк, доктор физико-математических наук, профессор, ФГУ ФНЦ НИИСИ РАН, главный научный сотрудник (e-mail: mix@niisi.ras.ru).

А.В. Мальцев, кандидат физико-математических наук, ФГУ ФНЦ НИИСИ РАН, ведущий научный сотрудник (e-mail: avmaltcev@mail.ru).

I. Введение

В настоящее время создание виртуальных прототипов реальных объектов актуально для многих отраслей науки и промышленности [1, 2]. Важным направлением является наземная подготовка космонавтов с помощью космических видеотренажеров. Такие системы должны обеспечивать высоко реалистичное моделирование визуальной обстановки и генерацию изображений в масштабе реального времени (с частотой не менее 25 раз в секунду) [3] на современных экранах со сверхвысоким разрешением.

Одной из важных задач космических видеотренажеров является моделирование и визуализация рельефа земной поверхности с учетом ее несферичности. Особенно это востребовано в тренировках по проведению визуально-инструментальных наблюдений Земли из космоса [4], которые включают в себя использование телекамер и фотоаппаратов с многократным увеличением [5].

Эффективным путем является моделирование рельефа Земли на основе детализированных глобальных сеток (карт) высот, заданных относительно эллипсоида вращения в международном стандарте WGS-84 [6]. Можно выделить два главных подхода к решению этой задачи.

Первый подход основан на испускании лучей из положения наблюдателя через пикселы экрана до пересечения с моделью земной поверхности [7, 8]. Преимуществом данного подхода является высокое качество синтезируемых изображений земной поверхности. Недостаток заключается в существенном падении скорости синтеза изображений с ростом числа обрабатываемых пикселов (просчитываемых лучей), что препятствует выполнению визуализации в масштабе реального времени на экранах со сверхвысоким разрешением. Недавнее появление аппаратной поддержки трассировки лучей у графических карт NVidia (линейка GeForce RTX) призвано улучшить данную ситуацию.

Второй подход основан на создании по карте высот полигональной (триангулированной) модели рельефа Земли. Преимуществом этого подхода является высокая эффективность визуализации треугольников на современных видеокартах за счет мощной аппаратной поддержки на уровне архитектуры графического

процессора (GPU). Однако, при этом качество визуализации модели рельефа ограничивается числом треугольников, которое видеокарта может визуализировать в масштабе реального времени (оно существенно меньше числа, необходимого для полного покрытия детализированной карты высот). Эффективным путем обхода этого ограничения является *видозависимая адаптивная триангуляция* карты высот, при которой построение треугольников выполняется только для видимых участков карты высот, а участки рельефа, на которых перепады высот не заметны на экране, представляются меньшим числом треугольников, чем остальные [9-11]. При реализации данного подхода на детализированных картах высот полигональная сложность (число треугольников) получаемой модели рельефа может существенно изменяться от кадра к кадру. В этой связи возникает задача разработки эффективных методов и алгоритмов построения в масштабе реального времени сложной адаптивной триангулированной модели земного рельефа, основанных на распределенных вычислениях и распараллеливании графических расчетов на современных многоядерных GPU.

В исследовании [12] построение полигональной модели рельефа выполняется на основе сферы, а карта высот записывается в видеопамять в виде концентрических текстур с убывающей детализацией. В работе [13] предлагается метод построения на GPU адаптивной полигональной модели рельефа на основе четырех типов предрассчитанных геометрических шаблонов. Авторы исследования [14] распараллелили построение треугольников адаптивной модели рельефа с помощью геометрического шейдера. Несмотря на перенос ряда графических вычислений на GPU, в приведенных работах ключевые расчеты, связанные с управлением тесселяцией (разбиением на треугольники) реализуются последовательно, что накладывает серьезные ограничения на размеры карт высот, которые могут быть обработаны в масштабе реального времени.

Добавление в архитектуру GPU *программируемой тесселяции* [15] открыло возможность управлять разбиением на треугольники параллельно на ядрах GPU (с помощью шейдеров управления и вычисления тесселяции). В работе [16] предложен метод адаптивной тесселяции на GPU рельефа с помощью параметрических четырехугольников (патчей-квадов) одинакового размера. В работе [17] для построения модели рельефа используется древовидная структура из патчей-квадов различных уровней детализации. Недостатком таких решений является образование разрывов геометрии (cracks) в местах стыковки смежных треугольников типа “сторона-вершина”. Для скрытия разрывов в [16] строятся дополнительные треугольники-заслонки, а в работе [18] предлагается использовать динамические сшивающие стрипы (ленты-треугольников). Другим недостатком патчей-квадов является склонность к образованию избыточной триангуляции в модели рельефа (если в крупный патч-квад попадает маленький участок карты высот, требующий высокой полигональной детализации, то весь патч-квад должен быть тесселирован с такой детализацией).

В данной работе предлагаются новые методы и алгоритмы адаптивной тесселяции на GPU рельефа Земли на основе эллипсоида вращения, которые обеспечивают построение сложных полигональных моделей рельефа Земли в масштабе реального времени. В отличие от моделирования рельефа на основе сферы, восстановление высот рельефа Земли относительно эллипсоида вращения является более точным, однако требует дополнительного объема вычислений, связанных с несовпадением нормали к поверхности эллипсоида и радиус-вектора. В данной работе эти вычисления выполняются параллельно и независимо друг от друга на ядрах GPU с помощью технологии шейдерной тесселяции. Новизна предлагаемого решения состоит в том, что тесселяция эллипосоида по карте высот впервые выполняется с помощью оригинальной схемы разбиения патчей-треугольников [19], которая в отличие от патчей-квадов позволяет эффективно локализовывать участки рельефа с повышенной детализацией, избегая принудительной тесселяции смежных областей. Еще одним преимуществом предложенной схемы является безразрывная стыковка патчей-треугольников, которая выполняется автоматически во ходе тесселяции при переходе от одного уровня детализации к другому. В разделе II приводится математическая модель восстановления координат точек рельефа Земли по карте высот, заданной относительно эллипсоида вращения. В разделе III предлагается метод и алгоритм построения полигональной модели эллипсоида, используемой в качестве основы для тееселяции. В разделе IV описываются разработанные методы и алгоритмы адаптивной тесселяции полигональной модели рельефа Земли. Предлагаемые методы и алгоритмы реализованы на языках C++ и GLSL (шейдерная модель четвертого поколения) с использованием графической библиотеки OpenGL. В разделе V приводятся полученные результаты.

II. Моделирование рельефа Земли на основе карты высот

Рассмотрим задачу восстановления координат точек рельефа Земли по карте высот, заданной относительно эллипсоида вращения WGS-84 [6].

Определим пространственную *геоцентрическую систему* $O_Г X_Г Y_Г Z_Г$ координат, начало которой совпадает с центром эллипсоида, ось $O_Г Z_Г$ - направлена на северный полюс Земли, ось $O_Г X_Г$ - в точку пересечения начального нулевого меридиана (в системе WGS-84 это Опорный меридиан [6]) с экватором, а ось $O_Г Y_Г$ дополняет систему до правой (рис. 1). В этой системе будут определяться координаты точек рельефа Земли.

Рассмотрим произвольную точку $P_{рел.}$ рельефа земной поверхности. Проведем через эту точку нормаль **n** к

поверхности эллипсоида (см. рис. 1). Обозначим через $P_э$ точку пересечения нормали $\mathbf{n}$ с поверхностью эллипсоида, через Q - точку пересечения меридиана, проходящего через $P_э$, с экватором, а через h - длину отрезка $P_э P_{рел.}$ (высота точки $P_{рел.}$ над поверхностью эллипсоида).

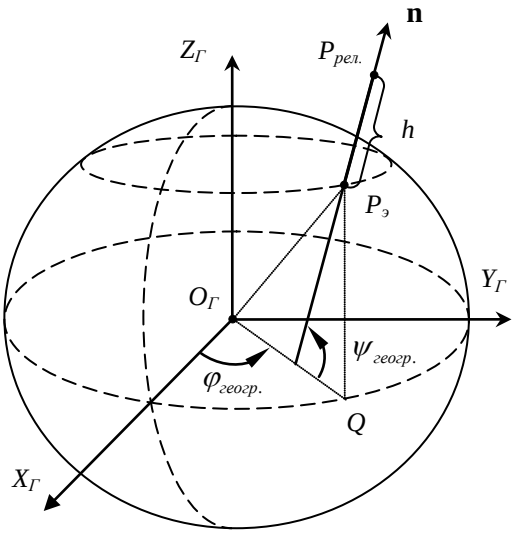

Рис. 1. Высота h точки рельефа Земли относительно эллипсоида вращения

Значение высоты h записано в карте высот рельефа в *географической системе координат* $(\psi_{геогр.}, \varphi_{геогр.})$, в которой $\psi_{геогр.} \in [-\pi/2, \pi/2]$ - угол между плоскостью экватора и нормалью $\mathbf{n}$ (географическая широта), а $\varphi_{геогр.} \in [-\pi, \pi)$ - угол между осью $O_Г X_Г$ и вектором $\mathbf{O_Г Q}$ (географическая долгота). Угол $\psi_{геогр.}$ положительный, если точка $P_э$ расположена в северном полушарии, и отрицательный, если $P_э$ - в южном полушарии. Угол $\varphi_{геогр.}$ положителен, если $P_э$ находится в восточном полушарии, и отрицателен, если $P_э$ - в западном полушарии (см. рис. 1).

Карта высот представляет собой двухмерную текстуру, в которой тексел с индексами $(0,0)$ хранит значение высоты рельефа для $\psi_{геогр.} = -\pi/2$, $\varphi_{геогр.} = -\pi$ (см. рис. 2). Для удобства работы с картой высот введем дополнительную систему координат $(\psi'_{геогр.}, \varphi'_{геогр.})$, в которой $\psi'_{геогр.} \in [0, \pi]$ - угол между осью $-O_Г Z_Г$ и нормалью $\mathbf{n}$, а $\varphi'_{геогр.} \in [0, 2\pi)$ - угол между осью $-O_Г X_Г$ и вектором $\mathbf{O_Г Q}$. Угол $\psi'_{геогр.}$ откладывается против часовой стрелки, если смотреть из конца вектора $\mathbf{n} \times \mathbf{O_Г Z_Г}$, а угол $\varphi'_{геогр.}$ - если смотреть из конца оси $O_Г Z_Г$ на плоскость экватора. Система $(\psi'_{геогр.}, \varphi'_{геогр.})$ связана с системой $(\psi_{геогр.}, \varphi_{геогр.})$ следующим образом:

$$\psi'_{геогр.} = \psi_{геогр.} + \pi/2\,,\ \varphi'_{геогр.} = \varphi_{геогр.} + \pi\,. \qquad (1)$$

Согласно [6] координаты $(x_э, y_э, z_э)$ рассматриваемой точки $P_э$ в системе $O_Г X_Г Y_Г Z_Г$ можно вычислить как

$$P_э = \begin{pmatrix} x_э \\ y_э \\ z_э \end{pmatrix} = \begin{pmatrix} N \cos\psi_{геогр.} \cos\varphi_{геогр.} \\ N \cos\psi_{геогр.} \sin\varphi_{геогр.} \\ N\left(1-e^2\right)\sin\varphi_{геогр.} \end{pmatrix}, \qquad (2)$$

где $N = a \Big/ \sqrt{1 - e^2 \sin^2 \psi_{геогр.}}$, $e = \sqrt{1 - b^2/a^2}$ - первый эксцентриситет эллипсоида, a и b - большая и малая полуоси эллипсоида вращения.

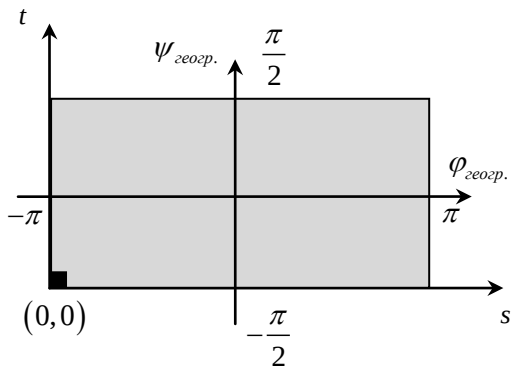

Рис. 2. Системы $(\psi_{геогр.}, \varphi_{геогр.})$ и (s,t) координат карты высот Земли

Запишем координаты точки $P_э$ через углы $\psi'_{геогр.}$ и $\varphi'_{геогр.}$, выразив из (1) углы $\psi_{геогр.}$, $\varphi_{геогр.}$ и подставив их в (2):

$$P_э = \begin{pmatrix} x_э \\ y_э \\ z_э \end{pmatrix} = \begin{pmatrix} N \cos(\psi'_{геогр.} - \pi/2) \cos(\varphi'_{геогр.} - \pi) \\ N \cos(\psi'_{геогр.} - \pi/2) \sin(\varphi'_{геогр.} - \pi) \\ N(1-e^2)\sin(\varphi'_{геогр.} - \pi) \end{pmatrix} =$$

$$= \begin{pmatrix} -N \sin\psi'_{геогр.} \cos\varphi'_{геогр.} \\ -N \sin\psi'_{геогр.} \sin\varphi'_{геогр.} \\ -N(1-e^2)\sin\varphi'_{геогр.} \end{pmatrix}, \qquad (3)$$

где

$$N = a \Big/ \sqrt{1 - e^2 \sin^2\left(\psi'_{геогр.} - \pi/2\right)} = a \Big/ \sqrt{1 - e^2 \cos^2 \psi'_{геогр.}}\,. \qquad (4)$$

Согласно [20] нормаль $\mathbf{n}$ к поверхности эллипсоида в точке $P_э$ будет совпадать с градиентом функции $F(x,y,z) = \frac{x^2+y^2}{a^2} + \frac{z^2}{b^2} - 1$ (из уравнения эллипсоида) в этой точке. Исходя из этого, запишем координаты (n_x, n_y, n_z) нормали $\mathbf{n}$ в системе $O_Г X_Г Y_Г Z_Г$:

$$\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \begin{pmatrix} \dfrac{\partial F(x_э, y_э, z_э)}{\partial x} \\ \dfrac{\partial F(x_э, y_э, z_э)}{\partial y} \\ \dfrac{\partial F(x_э, y_э, z_э)}{\partial z} \end{pmatrix} = \begin{pmatrix} \dfrac{2x_э}{a^2} \\ \dfrac{2y_э}{a^2} \\ \dfrac{2z_э}{b^2} \end{pmatrix} = \frac{2}{a^2} \begin{pmatrix} 1 \\ 1 \\ \dfrac{a^2}{b^2} \end{pmatrix} P_э\,. \qquad (5)$$

Обозначим через $\mathbf{n_e}$ нормализованный вектор нормали $\mathbf{n}$. Тогда координаты рассматриваемой точки $P_{рел.}$ рельефа Земли в системе $O_Г X_Г Y_Г Z_Г$ можно найти, сместив точку $P_э$ вдоль вектора $\mathbf{n_e}$ на высоту h:

$$P_{рел.} = P_э + h\mathbf{n}_e . \quad (6)$$

Построение полигональной модели рельефа Земли будет выполняться в два этапа. На ***первом этапе*** (на стадии подготовки данных для визуализации) построим полигональную модель эллипсоида (*ПМЭ*), состоящую из крупных полигонов. На ***втором этапе*** (на стадии визуализации) на основе полигонов ПМЭ, необходимых для визуализации текущего кадра, и карты высот выполним построение полигональной модели рельефа Земли. Рассмотрим эти этапы.

III. Метод построения ПМЭ

Пусть имеется карта высот рельефа размера $2^{m+1} \times 2^m$ текселов, где $m \geq 8$. Чтобы построить ПМЭ, выполним следующие шаги:

а) Выберем на эллипсоиде (равномерно по углам $\psi_{геогр.}$ и $\varphi_{геогр.}$) $n_{пар.} = 2^{m-6}$ параллелей и $n_{мер.} = 2^{m-5} - 1$ меридианов (единица вычитается во избежание повторного учета меридиана $\varphi_{геогр.} = -\pi$).

б) Зададим координаты $(x_э, y_э, z_э)$ позиций и текстурные координаты (s,t) вершин ($s,t \in [0,1]$) в точках пересечений параллелей и меридианов. Отметим, что вершина на пересечении параллели с меридианом $\varphi_{геогр.} = -\pi$ будет иметь текстурные координаты $(0,t)$ или $(1,t)$ в зависимости от того, какому треугольнику она будет принадлежать (см. следующий шаг). Поэтому для каждого такого пересечения зададим пару вершин с одинаковыми позициями, но разными текстурными координатами. Таким же образом зададим пересечения меридианов с Северным и Южным полюсами. Ввиду этого общее число вершин в ПМЭ будет равно

$$n_{верш.} = n_{пар.} \times (n_{мер.} + 1) .$$

Пронумеруем полученные вершины как показано на рисунке 3.

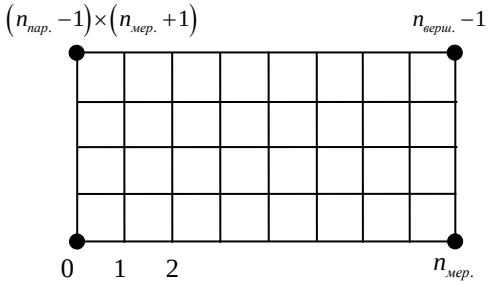

Рис. 3. Нумерация вершин ПМЭ

в) Соединим полученные вершины в треугольники, как показано на рисунке 4. Из рисунка видно, что на Северном и Южном полюсах между соседними меридианами треугольники являются одиночными, а не парными, как в остальных случаях. Объединим каждый одиночный треугольник Северного полюса с противоположным ему одиночным треугольником Южного полюса в пару. Тогда число всех пар треугольников в ПМЭ будет равно

$$n_{п.} = n_{мер.} \times (n_{пар.} - 2) .$$

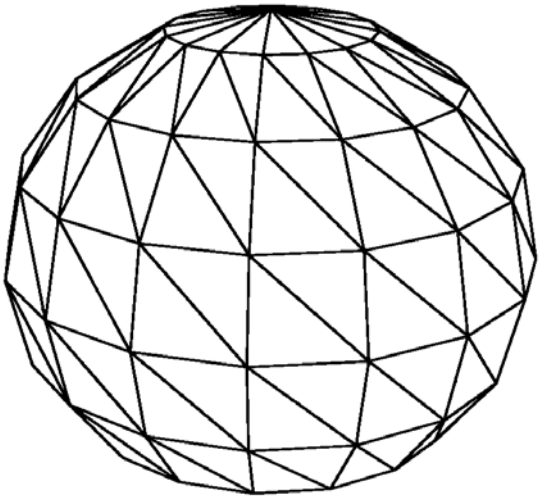

Рис. 4. Полигональная модель эллипсоида

Пронумеруем полученные пары треугольников, как показано на рисунке 5, и зададим индексы вершин каждого треугольника. Число $n_{инд.}$ всех индексов будет равно $6n_{п.}$.

Для выполнения описанных шагов создадим одномерные массивы $M_{поз}$ позиций $(x_э, y_э, z_э)$ и $M_{текс.}$ текстурных координат (s,t) длиной $n_{верш.}$, а также одномерный массив $M_{инд.}$ индексов вершин треугольников длиной $n_{инд.}$.

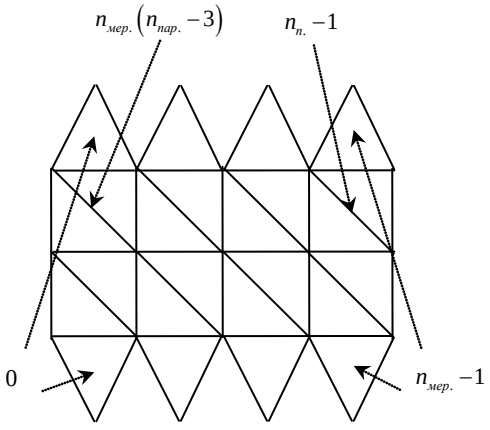

Рис. 5. Нумерация пар треугольников ПМЭ

Построение полигональной модели эллипсоида реализует Алгоритм 1.

1. Запишем в массивы $M_{текс.}$ и $M_{поз}$ текстурные координаты и позиции (в $O_Г X_Г Y_Г Z_Г$) вершин ПМЭ:

Вычислим шаги Δs меридиана и Δt параллели по координатным осям s и t (см. рис. 2):

$$\Delta s = 1/n_{мер.} , \ \Delta t = 1/n_{пар.} - 1 .$$

Цикл по i от 0 до $n_{пар.} - 1$

Цикл по j от 0 до $n_{мер.}$

Вычислим номер v (i, j)-ой вершины (см. рис. 3) и ее текстурные координаты (s,t):

$v = i \cdot \left(n_{мер.} + 1\right) + j$, $s = j\Delta s$, $t = i\Delta t$.

Если $i = 0$, то:

$M_{поз.}[v] = (0, 0, -b)$, $M_{текс.}[v] = (s, 0)$.

Перейдем к след. j.

В противном случае, если $i = n_{пар.} - 1$, то:

$M_{поз.}[v] = (0, 0, b)$, $M_{текс.}[v] = (s, 1)$.

Перейдем к след. j.

В противном случае, если $j = n_{мер.}$, то:

Вычислим номер v' $(i, 0)$-й вершины:

$v' = i \cdot (n_{мер.} + 1)$.

$M_{поз.}[v] = M_{поз.}[v']$, $M_{текс.}[v] = (1, M_{текс.}[v'].t)$.

Перейдем к след. i.

В противном случае:

$M_{текс.}[v] = (s, t)$, $\psi'_{геогр.} = \pi t$, $\varphi'_{геогр.} = 2\pi s$.

Вычислим $N(\psi'_{геогр.})$ по (4).

Вычислим позицию вершины эллипсоида $P_э(\psi'_{геогр.}, \varphi'_{геогр.}, N(\psi'_{геогр.}))$ согласно (3).

$M_{pos}[v] = P_э$.

Конец цикла.

Конец цикла.

2. Запишем в массив $M_{инд.}$ индексы треугольников ПМЭ:

Запишем $n'_{мер.} = n_{мер.} + 1$, $n'_{пар.} = n_{пар.} - 1$, $n''_{пар.} = n_{пар.} - 2$.

Цикл по i от 0 до $n_{пар.} - 3$

Цикл по j от 0 до $n_{мер.} - 1$

Вычислим номер t пары треугольников, соответствующей (i, j)-ой вершине: $t = in_{мер.} + j$.

Запишем $i' = i + 1$, $j' = j + 1$.

Если $i = 0$, то:

// для треугольника Южного полюса

$M_{инд.}[6t] = j$, $M_{инд.}[6t + 1] = n'_{мер.} + j'$,

$M_{инд.}[6t + 2] = n'_{мер.} + j$.

// для треугольника Северного полюса

$M_{инд.}[6t + 3] = n''_{пар.} n'_{мер.} + j$,

$M_{инд.}[6t + 4] = n''_{пар.} n'_{мер.} + j'$,

$M_{инд.}[6t + 5] = n'_{пар.} n'_{мер.} + j$.

Перейдем к след. j.

В противном случае:

$M_{инд.}[6t] = in'_{мер.} + j$,

$M_{инд.}[6t + 1] = in'_{мер.} + j'$,

$M_{инд.}[6t + 2] = i'n'_{мер.} + j$,

$M_{инд.}[6t + 3] = i'n'_{мер.} + j'$.

$M_{инд.}[6t + 4] = M_{инд.}[6t + 2]$.

$M_{инд.}[6t + 5] = M_{инд.}[6t + 1]$.

Конец цикла.

Конец цикла.

3. Запишем массивы $M_{поз}$, $M_{текс.}$ и $M_{инд.}$ в видеопамять с помощью технологии вершинных буферов.

Алгоритм 1. Построение ПМЭ.

В результате работы данного алгоритма получим в видеопамяти полигональную модель эллипсоида, вершины которой выровнены по узловым точкам карты высот. Далее будем выполнять построение полигональной модели рельефа (ПМР) Земли на основе полигонов ПМЭ, необходимых для визуализации текущего кадра, и карты высот. Рассмотрим это подробнее.

IV. Метод построения ПМР Земли

Чтобы построить полигональную модель рельефа Земли, в каждом кадре визуализации треугольники ПМЭ обрабатываются на запрограммированном определенным образом (см. ниже) графическом конвейере GPU.

В результате одного *прохода* через конвейер:

- каждый треугольник ПМЭ, который не нужен для визуализации текущего кадра, будет удален;
- каждый оставшийся треугольник ПМЭ, имеющий размеры, достаточно маленькие для качественной аппроксимации рельефа, будет преобразован в участок ПМР;
- каждый непреобразованный треугольник ПМЭ будет разбит на треугольники ПМЭ меньшего размера (от 4 до 6 треугольников, см. раздел 3.2), как показано на рисунке 6, которые будут отправлены на новый проход через графический конвейер.

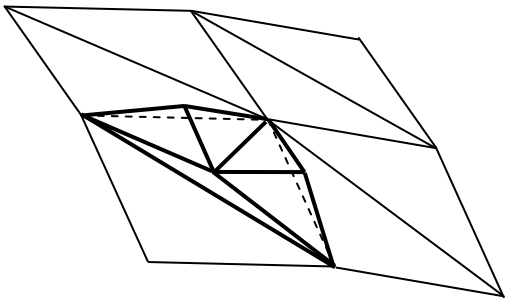

Рис. 6. Разбиение треугольника ПМЭ

Такие проходы выполняются в цикле до тех пор, пока не будут получены все участки ПМР, необходимые для визуализации текущего кадра.

Для реализации описанной схемы построения ПМР предлагается использовать программируемую тесселяцию [21]. В рамках этой технологии каждый треугольник ПМЭ, поступающий на графический конвейер, будет автоматически снабжен дополнительными параметрами тесселяции [21, 22], управляя которыми, мы будем либо удалять треугольник ПМЭ, либо разбивать его на треугольники по аппаратно «зашитому» алгоритму, либо оставлять без разбиения. Такой параметризованный треугольник ПМЭ назовем *патчем-треугольником.*

Рассмотрим некоторый i-ый проход графического конвейера. В рамках этого прохода обработка каждого патча-треугольника будет производиться в отдельном потоке GPU с помощью разработанного комплекса шейдерных программ: шейдера управления тесселяцией, шейдера вычисления тесселяции и геометрического шейдера (см. рис. 7). Рассмотрим их более подробно.

A. Управление тесселяцией

Данный шейдер работает одновременно на всех ядрах графического процессора и на каждом ядре обрабатывает свой патч-треугольник. Рассмотрим некоторый патч-треугольник $\triangle ABC$, поступивший на вход шейдера управления тесселяцией (см. рис. 8). Координаты вершин $\triangle ABC$ заданы в объектной системе координат (Object Coordinate System, OCS [23]) полигональной модели рельефа (совпадает с системой $O_\Gamma X_\Gamma Y_\Gamma Z_\Gamma$). Введем следующие обозначения:

– $\mathbf{n_A}$, $\mathbf{n_B}$, $\mathbf{n_C}$ - нормали к поверхности эллипсоида в вершинах $\triangle ABC$;

– R - участок поверхности рельефа Земли, ограниченный плоскостями $\{\mathbf{n_A},\mathbf{n_B}\}$, $\{\mathbf{n_B},\mathbf{n_C}\}$, $\{\mathbf{n_C},\mathbf{n_A}\}$;

– $\triangle A'B'C'$ - треугольник, вершины которого лежат в угловых точках участка R;

– $h_{A'}$, $h_{B'}$, $h_{C'}$ - высоты вершин $\triangle A'B'C'$ над поверхностью эллипсоида;

– d_+ - наибольшее из расстояний от плоскости $A'B'C'$ до точек участка R рельефа (в направлении от эллипсоида); d_- - аналогичное расстояние в обратном направлении;

– $\triangle A'_+B'_+C'_+$ и $\triangle A'_-B'_-C'_-$ - треугольники, вершины которых лежат на прямых AA', BB', CC' на расстояниях d_+ и d_- от плоскости $A'B'C'$.

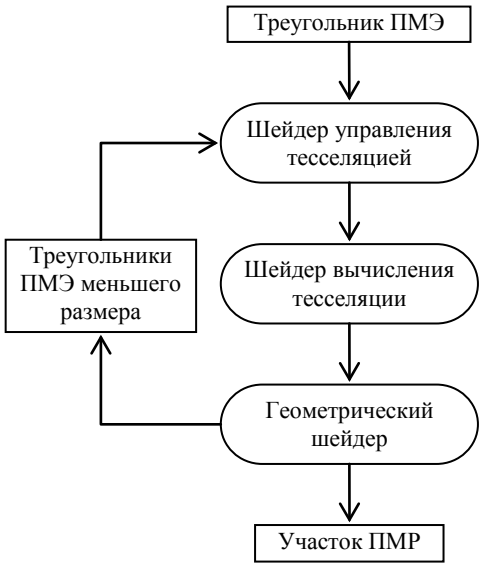

Рис. 7. Схема построения участка ПМР

Обработка $\triangle ABC$ шейдером управления тесселяцией включает в себя следующие шаги:

1. Вычисление координат вершин пятигранника $ABCA'_+B'_+C'_+$ (см. рис. 8), ограничивающего участок R рельефа.
2. Проверка нахождения пятигранника $ABCA'_+B'_+C'_+$ в поле зрения (в пирамиде видимости) виртуальной камеры.
3. Проверка видимости пятигранника $ABCA'_+B'_+C'_+$ по горизонту эллипсоида.
4. Расчет параметров тесселяции $\triangle ABC$.

С шагами 1 и 2 можно ознакомиться в работе [19]. Рассмотрим более подробно шаги 3 и 4.

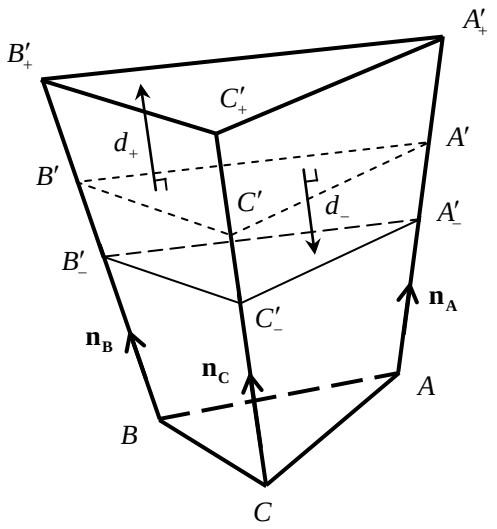

Рис. 8. Ограничивающий пятигранник участка *R* рельефа

Проверка видимости по горизонту. Обозначим через P_e - позицию камеры в системе OCS в рассматриваемой сцене на некотором заданном расстоянии от центра O_Γ эллипсоида, а через $\mathbf{q}$ - вектор $\mathbf{P_eO_\Gamma}$. Проведем из P_e произвольную касательную к эллипсоиду и обозначим через α - угол между $\mathbf{q}$ и этой касательной. Будем называть α *углом горизонта* эллипсоида. Рассмотрим предельные случаи: а) угол α_e горизонта на экваторе и б) угол α_p горизонта на полюсе эллипсоида (см. рис. 9). Из рисунка видно, что $\alpha_p \le \alpha \le \alpha_e$, где $\alpha_p, \alpha_e \in (0, \pi/2]$.

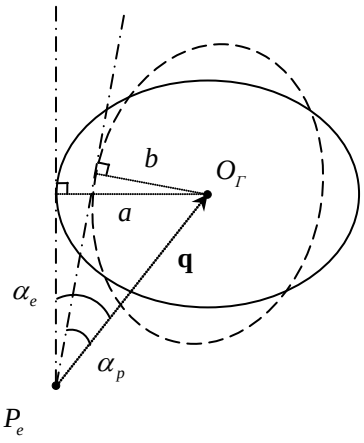

Рис. 9. Углы горизонта на экваторе и полюсе эллипсоида

Введем флаг b_{vis} видимости пятигранника $ABCA'_+B'_+C'_+$ в пределах наименьшего угла α_p горизонта ($b_{vis} = 1$ означает, что пятигранник виден). Тогда, вычис-

лить значение флага b_{vis} можно, проверив, что $\triangle ABC$ не лежит полностью на обратной стороне эллипсоида, а $\triangle A'_+B'_+C'_+$ - полностью в конусе с осью $\mathbf{q}$ симметрии и углом $2\alpha_p$ раствора. Это реализует Алгоритм 2.

1. Инициализируем флаг $b_{vis} = 0$.
2. Проверим, что $\triangle ABC$ не лежит полностью на обратной стороне эллипсоида:
   Запишем $\mathbf{r}_1 = \mathbf{AP_e}$, $\mathbf{r}_2 = \mathbf{BP_e}$, $\mathbf{r}_3 = \mathbf{CP_e}$.
   Проверим знаки $\cos(\mathbf{r}_1, \hat{\mathbf{n}}_\mathbf{A})$, $\cos(\mathbf{r}_2, \hat{\mathbf{n}}_\mathbf{B})$ и $\cos(\mathbf{r}_3, \hat{\mathbf{n}}_\mathbf{C})$:
      Если $(\mathbf{r}_1, \mathbf{n}_\mathbf{A}) \geq 0$ или $(\mathbf{r}_2, \mathbf{n}_\mathbf{B}) \geq 0$, или $(\mathbf{r}_3, \mathbf{n}_\mathbf{C}) \geq 0$, то $b_{vis} = 1$.
3. Если $b_{vis} = 0$, то проверим, что $\triangle A'_+B'_+C'_+$ не лежит полностью в конусе $2\alpha_p$:
   Запишем $\mathbf{p}_1 = \mathbf{P_e}\mathbf{A}'_+$, $\mathbf{p}_2 = \mathbf{P_e}\mathbf{B}'_+$, $\mathbf{p}_3 = \mathbf{P_e}\mathbf{C}'_+$.
   Введем углы $\beta_1 = (\hat{\mathbf{p}}_1, \mathbf{q})$, $\beta_2 = (\hat{\mathbf{p}}_2, \mathbf{q})$, $\beta_3 = (\hat{\mathbf{p}}_3, \mathbf{q})$, где $\beta_1, \beta_2, \beta_3 \in (0, \pi)$.
   Проверим знаки $\cos\beta_1$, $\cos\beta_2$ и $\cos\beta_3$:
      Запишем $g_1 = (\mathbf{p}_1, \mathbf{q})$, $g_2 = (\mathbf{p}_2, \mathbf{q})$, $g_3 = (\mathbf{p}_3, \mathbf{q})$.
      Если $g_1 \leq 0$ или $g_2 \leq 0$, или $g_3 \leq 0$, то $b_{vis} = 1$.
   Если $b_{vis} = 0$, то проверим отношения $\cos^2\alpha_p / \cos^2\beta_1$, $\cos^2\alpha_p / \cos^2\beta_2$ и $\cos^2\alpha_p / \cos^2\beta_3$:
      Запишем общий сомножитель этих отношений:
      $d = (\mathbf{q}, \mathbf{q}) - b^2$.
      Если $d(\mathbf{p}_1, \mathbf{p}_1)/g_1^2 \geq 1$ или $d(\mathbf{p}_2, \mathbf{p}_2)/g_2^2 \geq 1$, или $d(\mathbf{p}_3, \mathbf{p}_3)/g_3^2 \geq 1$, то $b_{vis} = 1$.

Алгоритм 2. Проверка видимости по горизонту.

Отметим, что п. 3 Алгоритма 2 также обеспечивает корректный учет сложного неявного случая видимого рельефа, когда основание возвышенности рельефа скрыто за горизонтом, а вершины находятся выше линии горизонта.

Полученное в результате выполнения Алгоритма 2 значение флага b_{vis} мы будем использовать при расчете значений параметров тесселяции $\triangle ABC$ на следующем шаге.

Расчет параметров тесселяции. Согласно технологии шейдерной тесселяции тесселируемый треугольник (в нашем случае это $\triangle ABC$) считается равносторонним, а его разбиение на треугольники выполняется на основе разбиения его сторон на отрезки и построения концентрических равносторонних треугольников [22]. Управление этими геометрическими операциями осуществляется с помощью параметров тесселяции – трех внешних и одного внутреннего уровней тесселяции.

Обозначим через $l_{out,A}$ первый внешний уровень тесселяции $\triangle ABC$ - число равных отрезков, на которые будет разбита сторона BC, а через $l_{out,B}$ и $l_{out,C}$ - второй и третий внешние уровни тесселяции (они определены аналогично уровню $l_{out,A}$). Также обозначим через l_{in} - внутренний уровень тесселяции $\triangle ABC$ - число равных шагов на каждой стороне $\triangle ABC$, через которые по перпендикулярам (см. рис. 10) будут построены вершины концентрических треугольников. На рисунке 10 изображены два концентрических треугольника для $l_{in} = 4$ (один из них вырожден в точку).

Расчет уровней $l_{out,A},...,l_{in}$ тесселяции выполняется, если $b_{vis} = 1$ (т.е. пятигранник $ABCA'_+B'_+C'_+$ виден). В противном случае значения всех уровней тесселяции обнуляются, что удалит $\triangle ABC$ из графического конвейера.

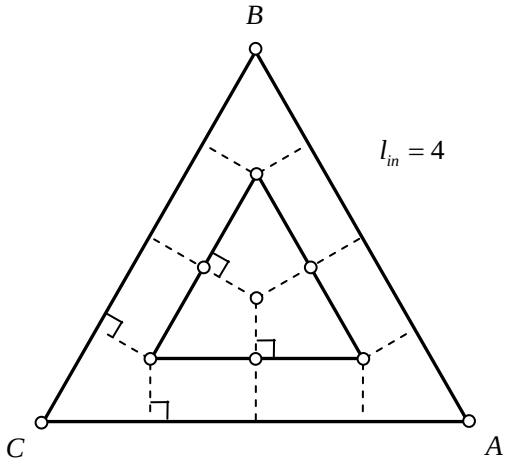

Рис. 10. Принцип построения концентрических треугольников

Чтобы вычислить уровни $l_{out,A},...,l_{in}$, используется система координат стандартного (нормализованного) объема видимости (Normalized Device Coordinate System, NDCS [23]). Рассмотрим ребро $A'_-A'_+$ пятигранника $ABCA'_+B'_+C'_+$. Обозначим через $P_{A'_-}$ и $P_{A'_+}$ координаты его вершин в системе OCS, а через $P^*_{A'_-}$ и $P^*_{A'_+}$ - координаты в системе NDCS. Построим проекцию $\mathbf{r}$ вектора $\mathbf{P}^*_{\mathbf{A}'_-}\mathbf{P}^*_{\mathbf{A}'_+}$ на плоскость $X_{NDCS}Y_{NDCS}$ (см. рис. 11).

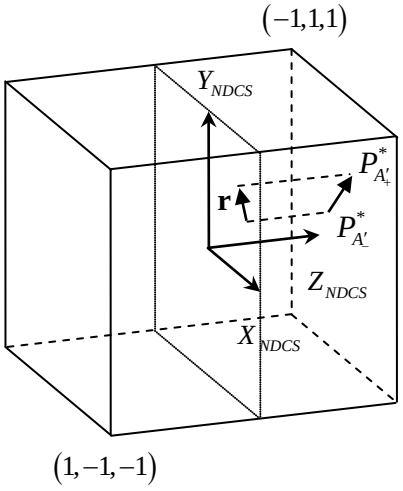

Рис. 11. Нормализованный объем видимости

Обозначим через $q_{A'_-A'_+}$ квадрат длины $\mathbf{r}$, а через $q_{B'_-B'_+}$ и $q_{C'_-C'_+}$ - квадраты аналогичных длин для ребер $B'_-B'_+$ и $C'_-C'_+$. Введем некоторое пороговое значение q_{thres} для $q_{A'_-A'_+}$, $q_{B'_-B'_+}$ и $q_{C'_-C'_+}$, при котором замена пятигранника $ABCA'_+B'_+C'_+$ на $\triangle A'B'C'$ будет не заметна на экране для наблюдателя:

$$q_{thres} = (n_{px} \max(2/w_{scr}, 2/h_{scr}))^2,$$

где w_{scr} и h_{scr} - ширина и высота экрана, в пикселах, а n_{px} - пороговое число пикселов на экране (1-2 пиксела). Вычислим значения уровней $l_{out,A},...,l_{in}$ тесселяции на основе сравнения $q_{A'_-A'_+}$, $q_{B'_-B'_+}$, $q_{C'_-C'_+}$ с q_{thres}, используя разработанный Алгоритм 3.

1. Передадим в шейдер управления тесселяцией порог q_{thres} и произведение M_{PMV} матрицы перспективной проекции на модельно-видовую матрицу [23].
2. Вычислим значение $q_{A'_-A'_+}$:

   Запишем координаты $P^*_{A'_-}$:

   $P^*_{A'_-} = M_{PMV} P_{A'_-}$,

   $P^*_{A'_-} = (P^*_{A'_-,x}/P^*_{A'_-,w}, P^*_{A'_-,y}/P^*_{A'_-,w}, P^*_{A'_-,z}/P^*_{A'_-,w}, 1)$.

   Запишем $P^*_{A'_+}$ аналогично $P^*_{A'_-}$.

   $\mathbf{r} = (P^*_{A'_+,x} - P^*_{A'_-,x}, P^*_{A'_+,y} - P^*_{A'_-,y})$. $q_{A'_-A'_+} = |\mathbf{r}|^2 = (\mathbf{r},\mathbf{r})$.
3. Вычислим $q_{B'_-B'_+}$ и $q_{C'_-C'_+}$ аналогично $q_{A'_-A'_+}$.
4. Запишем флаги $b_{A'_-A'}$, $b_{B'_-B'_+}$, $b_{C'_-C'_+}$:

   $b_{A'_-A'} = (q_{A'_-A'} \le q_{thres})$, $b_{B'_-B'_+} = (q_{B'_-B'_+} \le q_{thres})$,

   $b_{C'_-C'_+} = (q_{C'_-C'_+} \le q_{thres})$.
5. Запишем значения уровней $l_{out,A},...,l_{in}$:

   $l_{out,A} = 2 - b_{B'_-B'_+} b_{C'_-C'_+}$, $l_{out,B} = 2 - b_{A'_-A'} b_{C'_-C'_+}$,

   $l_{out,C} = 2 - b_{A'_-A'} b_{B'_-B'_+}$, $l_{in} = 4 - 3 b_{A'_-A'} b_{B'_-B'_+} b_{C'_-C'_+}$.
6. Запишем флаг b_{tess} тесселяции $\triangle ABC$ ($b_{tess} = 0$ означает, что тесселяции $\triangle ABC$ не будет):

   $b_{tess} = 0$, если $l_{out,A} = l_{out,B} = l_{out,C} = 1$.

Алгоритм 3. Расчет параметров тесселяции.

В результате выполнения Алгоритма 3 мы получим значения внешних уровней тесселяции ($l_{out,A}$, $l_{out,B}$, $l_{out,C}$) равные 1 или 2, а внутреннего уровня - 1 или 4. Отметим, что введенная в пп. 4 и 5 алгоритма замена условных переходов на вычисляемые выражения позволяет избавиться от лишних ветвлений, которые снижают эффективность выполнения алгоритмов на GPU.

Также в дальнейшем (при вычислении тесселяции $\triangle ABC$) нам понадобится информация о том, какая сторона у $\triangle ABC$ является наибольшей в системе OCS. В общем случае ею может быть любая сторона $\triangle ABC$, т.к. начало обхода его вершин не фиксировано. Чтобы задать наибольшую сторону, введем флаги b_{AB}, b_{BC} и b_{AC} сторон $\triangle ABC$, где только у наибольшей стороны флаг будет равен 1, а остальные флаги равны 0. Расчет флагов b_{AB}, b_{BC}, b_{AC} выполняется в системе OCS следующим образом:

1. Запишем квадраты q_{AB}, q_{BC} и q_{AC} длин сторон AB, BC и AC.
2. Запишем флаги b_1, b_2, b_3:

   $b_1 = (q_{AB} \ge q_{AC})$, $b_2 = (q_{AB} \ge q_{BC})$,

   $b_3 = (q_{BC} \ge q_{AC})$.
3. Запишем флаги b_{AB}, b_{BC}, b_{AC}:

   $b_{AB} = b_1 b_2$, $b_{BC} = b_3(1 - b_{AB})$.

   $b_{AC} = (1 - b_{AB})(1 - b_{BC})$.

После выполнения шейдера управления тесселяцией $\triangle ABC$ вместе с рассчитанными значениями $l_{out,A},...,l_{in}$ передается в *генератор примитивов*. Это фиксированный этап графического конвейера с аппаратно «зашитым» алгоритмом. В соответствии с нашими значениями $l_{out,A},...,l_{in}$ генератор примитивов либо оставит $\triangle ABC$ без изменения (если все уровни тесселяции равны 1), либо разобьёт его на треугольники. Во втором случае внутри $\triangle ABC$ будет создан концентрический треугольник с вершиной в центре, а пространство внутри и снаружи этого треугольника будет заполнено более мелкими треугольниками (их число зависит от значений $l_{out,A}$, $l_{out,B}$, $l_{out,C}$). При этом на сторонах $\triangle ABC$ могут быть добавлены дополнительные вершины. В примере на рисунке 12 добавлены две вершины M и K, однако в данной работе рассматривается также возможная вершина J на стороне AB.

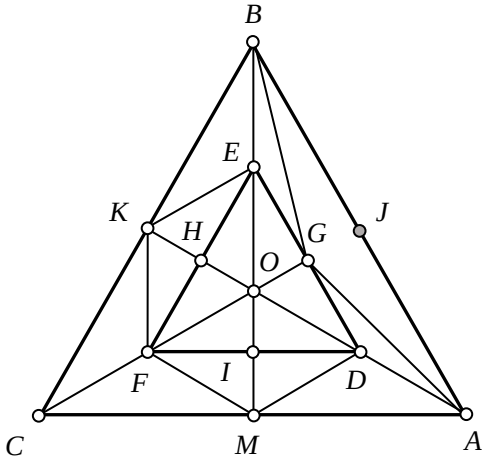

Рис. 12. Разбиение $\triangle ABC$ на треугольники в барицентрической системе координат при $l_{out,A} = l_{out,B} = 2$, $l_{out,C} = 1$

Вершины сгенерированных треугольников заданы барицентрическими координатами в точечном базисе A, B, C (эти координаты приведены в работе [19]). Вместе с барицентрическими координатами сгенерированные вершины передаются на следующий этап графического конвейера – шейдер вычисления тесселяции.

B. Вычисление тесселяции

На данном этапе графического конвейера координаты вершин сгенерированных треугольников ПМЭ преобразуются из барицентрической системы координат в систему OCS, как показано на рис. 6. Для этого мы растянем ΔDEF в барицентрической системе координат, так, чтобы он совпал с ΔABC, вершины G, H и I совместим с вершинами J, K и M, а вершину O сместим на середину наибольшей стороны ΔABC, заданной одним из флагов b_{AB}, b_{BC}, b_{AC} (см. пример на рисунке 13).

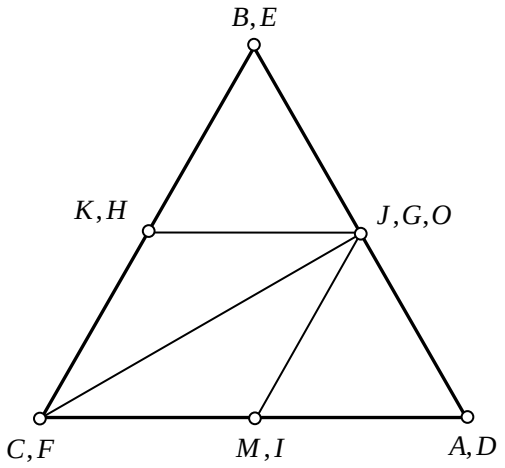

Рис. 13. Пример коррекции барицентрических координат вершин

Описанную процедуру реализует разработанный *шейдер вычисления тесселяции*. Данный шейдер выполняется на всех ядрах GPU и на каждом ядре обрабатывает свою вершину (одну из вершин $A,...,O$). Обозначим через V одну из таких вершин, поступившую на вход шейдера вычисления тесселяции. Заменим ее барицентрические координаты на координаты одной из вершин A, B, C, J, K, M и, используя их, вычислим для вершины V текстурные координаты T_V, позицию P_V (в системе OCS), а также идентификатор I_V (см. ниже). Это реализует разработанный Алгоритм 4.

1. Передадим в шейдер вычисления тесселяции следующие параметры:

а) барицентрические координаты P_{bar} вершины V (из генератора примитивов);

б) барицентрические координаты $P_{bar,A},\ldots,\ P_{bar,O}$ вершин $A,...,O$ (см. работу [19]);

в) текстурные координаты T_A, T_B, T_C и позиции P_A, P_B, P_C (в системе OCS) вершин ΔABC;

г) флаги b_{AB}, b_{BC}, b_{AC} (из шейдера управления тесселяцией).

2. Запишем флаги $b_A...,b_O$ вершин $A,...,O$ (флаг $b_A = 1$, если $\left|P_{bar} - P_{bar,A}\right| \le \varepsilon$, где ε - машинная погрешность представления действительных чисел).

3. Запишем взаимоисключающие флаги b'_A, b'_B, b'_C, b'_J, b'_K, b'_M (только один из них будет равен 1, остальные - 0):

$$b'_A = b_A \vee b_D,\ b'_B = b_B \vee b_E,\ b'_C = b_C \vee b_F,$$
$$b'_J = b_J \vee b_G \vee \left(b_O \wedge b_{AB}\right),\ b'_K = b_K \vee b_H \vee \left(b_O \wedge b_{BC}\right),$$
$$b'_M = b_M \vee b_I \vee \left(b_O \wedge b_{AC}\right).$$

4. Запишем новые барицентрические координаты P'_{bar} вершины V:

$$P'_{bar} = \sum_{U \in \{A,B,C,J,K,M\}} b'_U P_{bar,U}.$$

5. Вычислим текстурные координаты T_V и позицию P_V вершины V:

$$T_V = P'_{bar,x} T_A + P'_{bar,y} T_B + P'_{bar,z} T_C.$$

Если $b'_A = 1$ или $b'_B = 1$, или $b'_C = 1$, то:

$$P_V = b'_A P_A + b'_B P_B + b'_C P_C.$$

В противном случае:

Запишем координаты $(\psi'_{геогр.}, \varphi'_{геогр.})$ вершины V (см. раздел 1):

$$\psi'_{геогр.} = \pi T_{V,t},\ \varphi'_{геогр.} = 2\pi T_{V,s}.$$

Вычислим $N(\psi'_{геогр.})$ согласно (4).

Вычислим $P_V(\psi'_{геогр.}, \varphi'_{геогр.}, N(\psi'_{геогр.}))$ согласно (3).

6. Запишем целочисленный идентификатор I_V вершины V (в 0-й бит запишем значение флага b'_A, в 1-ый бит - флага b'_B и т.д.).

Алгоритм 4. Вершинная коррекция.

Результатом применения данного алгоритма ко всем исходным вершинам будут являться вершины треугольников ПМЭ в системе OCS полигональной модели рельефа. На рисунке 14 показан результат применения алгоритма к вершинам из рис. 12 (при условии, что AB - наибольшая сторона ΔABC). Из рисунка 14 видно, что некоторые треугольники (ΔAGD, ΔGOD, ΔEKH и др.) будут вырожденными (мы их удалим на следующем шаге, см. ниже), а вертикальный ΔABG будет обеспечивать безразрывную стыковку полученной триангуляции с соседним, более крупным треугольником ПМЭ. Как можно заметить, в общем случае в результате тесселяции ΔABC мы получим до 6 невырожденных треугольников: 4 наклонных треугольника и от 0 до 2 вертикальных треугольников. Полученные треугольники ПМЭ передаются на следующий этап графического конвейера – геометрический шейдер.

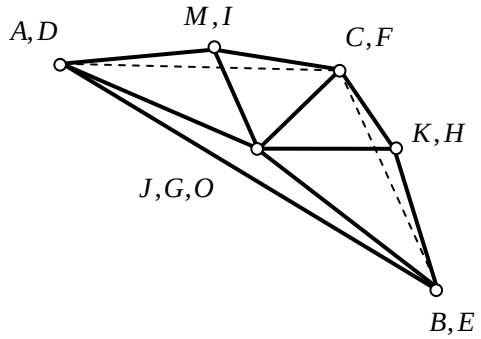

Рис. 14. Результат тесселяции ΔABC

C. Построение выходной геометрии

Геометрический шейдер является завершающим этапом рассматриваемого i-го прохода графического конвейера. Данный шейдер выполняется на всех ядрах GPU и на каждом ядре обрабатывает свой треугольник ПМЭ. На данном этапе из каждого невырожденного треугольника ПМЭ будет создаваться либо треугольник ПМР, либо треугольник новой ПМЭ и накапливать такие треугольники в видеопамяти. Для этого используется вершинный буфер B_{PMR} треугольников ПМР, а также буферы $B_{PASS,0}$ и $B_{PASS,1}$ треугольников новой ПМЭ (для четного и нечетного проходов графического конвейера). Все эти вершинные буферы создаются в видеопамяти один раз на стадии загрузки системы визуализации.

Рассмотрим процесс обработки геометрическим шейдером треугольника ПМЭ на четном i-ом проходе графического конвейера (для нечетного прохода он будет аналогичен). Обозначим такой треугольник через $\Delta V_0V_1V_2$. Из предыдущих этапов графического конвейера следует, что $\Delta V_0V_1V_2$ будет либо исходным ΔABC, либо одним из треугольников, образованных в результате тесселяции ΔABC. В первом случае мы будем создавать в видеопамяти из $\Delta V_0V_1V_2$ треугольник ПМР – таковым будет $\Delta A'B'C'$ (см. рис. 8), а во втором случае – сам $\Delta V_0V_1V_2$. Это реализует Алгоритм 5.

1. Передадим в геометрический шейдер следующие параметры:
   а) флаг b_{tess} тесселяции ΔABC (из шейдера управления тесселяцией);
   б) текстурные координаты T_{V0}, T_{V1}, T_{V2}, позиции P_{V0}, P_{V1}, P_{V2} и идентификаторы I_{V0}, I_{V1}, I_{V2} вершин $\Delta V_0V_1V_2$ (из шейдера вычисления тесселяции).
2. Если $b_{tess} = 0$, то:
   Вычислим текстурные координаты $T_{A'}$, $T_{B'}$, $T_{C'}$ и позиции $P_{A'}$, $P_{B'}$, $P_{C'}$ вершин $\Delta A'B'C'$:
   $T_{A'} = T_{V0}$, $T_{B'} = T_{V1}$, $T_{C'} = T_{V2}$.
   Запишем высоты вершин $\Delta A'B'C'$:
   $h_{A'} = F_{MAP}(T_{A'})$, $h_{B'} = F_{MAP}(T_{B'})$, $h_{C'} = F_{MAP}(T_{C'})$, где F_{MAP} - функция выборки из карты высот.
   Вычислим нормали $\vec{n}_{V0}$, $\vec{n}_{V1}$, $\vec{n}_{V2}$ к поверхности эллипсоида в вершинах $\Delta V_0V_1V_2$ по формуле (5).
   Вычислим позиции $P_{A'}(P_{V0}, \vec{n}_{V0}, h_{A'})$, $P_{B'}(P_{V1}, \vec{n}_{V1}, h_{B'})$ и $P_{C'}(P_{V2}, \vec{n}_{V2}, h_{C'})$ по формуле (6).
   Сгенерируем $\Delta A'B'C'$.
   Добавим текстурные координаты и позиции вершин $\Delta A'B'C'$ в буфер B_{PMR}.
   В противном случае:
   Запишем флаг b_{DEG} вырожденности $\Delta V_0V_1V_2$:
   Если $I_{V0} = I_{V1}$ или $I_{V0} = I_{V2}$, или $I_{V1} = I_{V2}$, то $b_{DEG} = 1$ ($\Delta V_0V_1V_2$ вырожден), в противном случае $b_{DEG} = 0$.
   Если $b_{DEG} = 0$, то создадим треугольник новой ПМЭ:
   Сгенерируем $\Delta V_0V_1V_2$.
   Добавим текстурные координаты и позиции вершин $\Delta V_0V_1V_2$ в буфер $B_{PASS,0}$.

Алгоритм 5. Создание выходного треугольника i-го прохода.

Отметим, что в буфер $B_{PASS,0}$ запись координат вершин ведется с начала буфера, а в буфере B_{PMR} перед выполнением i-го прохода мы смещаем начало записи согласно количеству треугольников ПМР, накопленных за все предыдущие проходы.

После выполнения этапа геометрического шейдера к имеющимся в буфере B_{PMR} данным от предыдущих проходов будут добавлены данные треугольников ПМР i-го прохода, а в буфер $B_{PASS,0}$ будут записаны только данные выходных треугольников ПМЭ i-го прохода. Если число таких треугольников не равно 0, то мы используем буфер $B_{PASS,0}$ в качестве новой ПМЭ для $(i+1)$-го прохода, а для накопления выходных треугольников используем буферы $B_{PASS,1}$ и B_{PMR} и т.д. Когда число выходных треугольников ПМЭ станет равным 0, цикл проходов через графический конвейер будет завершен, а в буфере B_{PMR} будет сформирована результирующая полигональная модель рельефа Земли.

V. Результаты

На основе предложенных методов и алгоритмов был реализован программный комплекс построения и визуализации трехмерной полигональной модели рельефа Земли. Моделирование рельефа осуществлялось на основе карты высот размером 16K×8K, тесселяция выполнялась с использованием патчей-треугольников 6 уровней детализации.

На рисунке 15 показан пример кадра полученной визуализации гор Мьянмы с высоты 400 км с четырехкратным увеличением. Визуализация построенной модели рельефа выполнялась с наложением детализированной текстуры земной поверхности и расчетом освещенности, учитывающим физические свойства атмосферы (рассеяние, поглощение). Предложенная схема тесселяции рельефа позволила значительно сократить число визуализируемых треугольников (в среднем с 3,2 млн. до 174 тысяч) без видимых потерь качества синтезируемых изображений. Тесселяция рельефа Земли выполнялась с помощью видеокарты GeForce GTX 1080 Ti на экране с разрешением Full HD, среднее время построения полигональной модели рельефа составило менее 1 мс.

VI. Заключение

В работе рассмотрена задача построения на GPU в масштабе реального времени адаптивной полигональной модели рельефа Земли на основе детализированных глобальных карт высот, заданных в стандарте WGS-84 относительно эллипсоида вращения. Предложенное решение включает новые, распределенные методы и алгоритмы построения на GPU полигональной модели рельефа с помощью технологии шейдерной тесселяции. Распределенное построение полигональной модели рельефа выполняется на основе разбиения эллипсоида вращения на патчи-треугольники, выровненные по узлам карты высот. Обработка патчей-треугольников выполняется полностью на GPU, параллельно и независимо друг от друга, с помощью разработанных шейдера управления тесселяцией, шейдера вычисления тесселяцией и геометрического шейдера. Реализованная в шейдерах оригинальная схема тесселяции эллипсоида позволила эффективно снизить число визуализируемых треугольников и сделать менее подверженными альясингу синтезируемые изображения Земли. Созданные распределенные методы и алгоритмы были реализованы в программном комплексе построения и визуализации адаптивной трехмерной полигональной модели земного рельефа. Разработанный комплекс был апробирован на детализированной карте высот Земли в составе сложной космической виртуальной сцены (с подробной моделью МКС, около 2 млн. полигонов). Полученные результаты подтвердили адекватность разработанных методов и алгоритмов поставленной задаче и их применимость для космических видеотренажеров, систем виртуального окружения, образовательных приложений и др.

Библиография

[1] В.А. Кузнецов, Ю.Г. Руссу, В.П. Куприяновский, “Об использовании виртуальной и дополненной реальности,” *International Journal of Open Information Technologies*, т. 7, № 4, стр. 75-84, 2019.

[2] М.А. Бондаренко, В.А. Сухомлин, “Анализ алгоритмов совмещения видеоинформации в авиационных системах,” *International Journal of Open Information Technologies*, т. 4, № 10, стр. 76-81, 2016.

[3] М.В. Михайлюк, М.А. Торгашев, “Система визуализации “GLView” для имитационно-тренажерных комплексов подготовки космонавтов,” *Пилотируемые полеты в космос*, № 4(9), стр. 60-72, 2013.

[4] В.В. Коваленок, А.С. Иванченков, С.В. Авакян, “Результативность визуально-инструментальных наблюдений в долговременных пилотируемых полетах,” *Пилотируемые полеты в космос*, № 4(21), стр. 103-117, 2016.

[5] П.Ю. Тимохин, М.В. Михайлюк, “Метод сжатия разрядности карт высот на основе критерия визуальной значимости,” *Труды НИИСИ РАН*, т. 7, № 1, стр. 30-35, 2017.

[6] Л.М. Бугаевский, *Математическая картография*. М.: «Златоуст», 1998, 400 с.

[7] P. Cozzi, K. Ring, *3D Engine Design for Virtual Globes*, Boca Raton: CRC Press, 2011.

[8] P. Cozzi, D. Bagnell, “A WebGL Globe Rendering Pipeline,” *GPU Pro 4. Advanced Rendering Techniques*, CRC Press, 2013, pp. 39-48.

[9] T. Ulrich, “Rendering massive terrains using chunked level of detail control,” *SIGGRAPH Course Notes*, vol. 3, no. 5, 2002.

[10] P. Cignoni, F. Ganovelli et al, “Planet-sized batched dynamic adaptive meshes (P-BDAM),” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 2003, pp. 147-154.

[11] L.M. Hwa, M.A. Duchaineau, K.I. Joy, “Real-time optimal adaptation for planetary geometry and texture: 4-8 tile hierarchies,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 4, pp. 355-368, 2005.

[12] M. Clasen, H.-C. Hege, “Terrain rendering using spherical clipmaps,”

Рис. 15. Пример кадра визуализации построенной на GPU модели рельефа Земли (горы Мьянмы, высота 400 км, 4-х кратное увеличение)

Eurographics/IEEE-VGTC Symposium on Visualization, 2006.
[13] Y. Livny, Z. Kogan, J. El-Sana, "Seamless patches for GPU-based terrain rendering," *The Visual Computer*, vol. 25, pp. 197-208, 2008.
[14] O. Ripolles, F. Ramos et al, "Real-time tessellation of terrain on graphics hardware," *Computers & Geosciences*, vol. 41, pp. 147-155, 2012.
[15] I. Cantlay, "DirectX 11 terrain tessellation," *NVIDIA WhitePaper*, 2011.
[16] E. Yusov, M. Shevtsov, "High-performance terrain rendering using hardware tessellation," *Journal of WSCG*, no. 19(3), pp. 85-92, 2011.
[17] B. Mistal, "GPU Terrain Subdivision and Tessellation," *GPU Pro 4. Advanced Rendering Techniques*, CRC Press, pp. 3-20, 2013.
[18] L. Zhang, J. She et al, "A Multilevel Terrain Rendering Method Based on Dynamic Stitching Strips," *ISPRS International Journal of Geo-Information*, vol. 8, no. 6: 255, 2019.
[19] M.V. Mikhaylyuk, P.Y. Timokhin, A.V. Maltsev, "A Method of Earth Terrain Tessellation on the GPU for Space Simulators," *Programming and Computer Software*, vol. 43, no. 4, pp. 243-249, 2017.
[20] М.Я. Выгодский, *Справочник по высшей математике*, М.: АСТ: Астрель, 2006, p. 991.
[21] M. Bailey, S. Cunningham, *Graphics Shaders: Theory and Practice. Second Edition*, CRC Press, 2011, 518 p.
[22] M. Segal, K. Akeley, "The OpenGL Graphics System: A Specification. Version 4.6, Core Profile," *The Khronos Group Inc.*, 2006-2018. URL: https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf (review date: 19.09.2019).
[23] R.J. Rost, B. Licea-Kane, *OpenGL Shading Language (3rd Edition)*. Boston, USA: Addison Wesley Professional, 2009, 792 p.

Real-time construction on the GPU of adaptive terrestrial relief model based on the spheroid

P.Yu. Timokhin, M.V. Mikhaylyuk, A.V. Maltsev

***Abstract*—The paper considers a task of real-time modelling of Earth relief based on detailed global height maps specified relative to the rotational ellipsoid (spheroid) WGS-84. The technology of adaptive tessellation of triangular patches on the GPU is proposed, which provides real-time construction of complex polygonal models of the Earth's relief. The technology includes the stage of dividing the visible ellipsoid into coarse patches (low level of detail) and the stage of their tessellation into triangles of the relief model, executed in parallel and independently on the GPU cores. The paper proposes new, distributed methods and algorithms to extract the patches needed for visualization of the current frame, to increase their level of detail in accordance with the height map and screen resolution, and to transform to relief polygons. The novelty of the work is that the tessellation of triangular patches of an ellipsoid is based on the original scheme, which allows relief areas that require high level detail to be effectively localized. This significantly reduces the time of relief model construction and improves the quality of the created images.**

The developed technology, methods and algorithms were implemented in program modules and tested in the visualization system of the Earth's virtual surface. Modelling and visualization of the Earth's relief was carried out with a detailed texture of the underlying Earth's surface and the calculation of illumination taking into account the atmosphere. The approbation was carried out in a virtual space scene comprising a detailed polygonal model of the International Space Station (ISS). The obtained results confirmed the adequacy of the proposed solution to the task and its applicability for building of space video simulators, virtual environment systems, virtual laboratories, etc.

***Keywords*—visualization, Earth, relief, GPU, virtual model, rotational ellipsoid, real-time, tessellation, distributed computing.**

REFERENCES

[1] V.A. Kuznetsov, J.G. Russu, V.P. Kupriyanovsky, "On the use of virtual and augmented reality," *International Journal of Open Information Technologies*, vol. 7, no. 4, pp. 75-84, 2019.
[2] M.A. Bondarenko, V.A. Sukhomlin, "Analyze of video information alignment algorithms in aviation systems," *International Journal of Open Information Technologies*, vol. 4, no. 10, pp. 76-81, 2016.
[3] M.V. Mikhaylyuk, M.A. Torgashev, ""GLView" – visualization system for simulation and training complexes for cosmonaut preparing," *Manned Spaceflight*, no. 4 (9), pp. 60-72, 2013.
[4] V.V. Kovalenok, A.S. Ivanchenkov, S.V. Avakyan, "Effectiveness of Visual-Instrumental Observations in Long-Term Manned Space Flights," *Manned Spaceflight*, no. 4 (21), pp. 103-117, 2016.
[5] P.Y. Timokhin, M.V. Mikhaylyuk, "The method of height map bit depth compression based on visual significance criterion," *Proceedings of SRISA RAS*, vol. 7, no. 1, pp. 30-35, 2017.
[6] L.M. Bugaevskiy, *Mathematical cartography*. M.:"Zlatoust", 1998, 400 p.
[7] P. Cozzi, K. Ring, *3D Engine Design for Virtual Globes*, Boca Raton: CRC Press, 2011.
[8] P. Cozzi, D. Bagnell, "A WebGL Globe Rendering Pipeline," *GPU Pro 4. Advanced Rendering Techniques*, CRC Press, 2013, pp. 39-48.
[9] T. Ulrich, "Rendering massive terrains using chunked level of detail control," *SIGGRAPH Course Notes*, vol. 3, no. 5, 2002.
[10] P. Cignoni, F. Ganovelli et al, "Planet-sized batched dynamic adaptive meshes (P-BDAM)," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 2003, pp. 147-154.
[11] L.M. Hwa, M.A. Duchaineau, K.I. Joy, "Real-time optimal adaptation for planetary geometry and texture: 4-8 tile hierarchies," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 4, pp. 355-368, 2005.
[12] M. Clasen, H.-C. Hege, "Terrain rendering using spherical clipmaps," *Eurographics/IEEE-VGTC Symposium on Visualization*, 2006.
[13] Y. Livny, Z. Kogan, J. El-Sana, "Seamless patches for GPU-based terrain rendering," *The Visual Computer*, vol. 25, pp. 197-208, 2008.
[14] O. Ripolles, F. Ramos et al, "Real-time tessellation of terrain on graphics hardware," *Computers & Geosciences*, vol. 41, pp. 147-155, 2012.
[15] I. Cantlay, "DirectX 11 terrain tessellation," *NVIDIA WhitePaper*, 2011.
[16] E. Yusov, M. Shevtsov, "High-performance terrain rendering using hardware tessellation," *Journal of WSCG*, no. 19(3), pp. 85-92, 2011.
[17] B. Mistal, "GPU Terrain Subdivision and Tessellation," *GPU Pro 4. Advanced Rendering Techniques*, CRC Press, pp. 3-20, 2013.
[18] L. Zhang, J. She et al, "A Multilevel Terrain Rendering Method Based on Dynamic Stitching Strips," *ISPRS International Journal of Geo-Information*, vol. 8, no. 6: 255, 2019.
[19] M.V. Mikhaylyuk, P.Y. Timokhin, A.V. Maltsev, "A Method of Earth Terrain Tessellation on the GPU for Space Simulators," *Programming and Computer Software*, vol. 43, no. 4, pp. 243-249, 2017.
[20] M.Y. Vygodskiy, *Handbook of higher mathematics*, M.: ACT: Astrel', 2006, 991 p.

[21] M. Bailey, S. Cunningham, *Graphics Shaders: Theory and Practice. Second Edition*, CRC Press, 2011, 518 p.
[22] M. Segal, K. Akeley, "The OpenGL Graphics System: A Specification. Version 4.6, Core Profile," *The Khronos Group Inc.*, 2006-2018. URL: https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf (review date: 19.09.2019).
[23] R.J. Rost, B. Licea-Kane, *OpenGL Shading Language (3rd Edition)*. Boston, USA: Addison Wesley Professional, 2009, 792 p.