Адаптивная тесселяция на GPU виртуального рельефа с помощью патчей-треугольников

Михаил Михайлюк, Петр Тимохин, Андрей Мальцев
ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН», Москва, Россия
webpismo@yahoo.de

Аннотация

В статье предлагается новый метод реализации адаптивной тесселяции виртуального рельефа на графическом процессоре (GPU), работающий в масштабе реального времени в сложных многообъектных виртуальных сценах. Моделирование рельефа осуществляется с помощью патчей-треугольников различных уровней детализации на видеокартах с поддержкой программируемой тесселяции. Вычисление патчей-треугольников каждого уровня детализации выполняется параллельно и независимо друг от друга. В работе предложен эффективный алгоритм разбиения (тесселяции) патчей-треугольников, который обеспечивает в модели рельефа безразрывную стыковку соседних треугольников с различным уровнем детализации. На основе предложенного решения был разработан и успешно апробирован программный комплекс визуализации рельефа земной поверхности. Полученные методы и алгоритмы могут быть применены в имитационно-тренажерных комплексах, системах виртуального окружения, обучающих геоприложениях и др.

Ключевые слова: *адаптивная триангуляция, тесселяция, рельеф, многообъектная среда, реальное время, GPU.*

1. ВВЕДЕНИЕ

Одной из актуальных задач многих видеотренажерных комплексов и систем виртуального окружения является моделирование и визуализация рельефа земной поверхности на основе детализированных регулярных сеток высот в масштабе реального времени [1]. Эффективным подходом к решению этой задачи является видозависимая адаптивная триангуляция сетки высот рельефа, при которой участки рельефа с перепадом высот, незаметным на экране, представляются меньшим числом треугольников, чем остальные [2-7]. Однако прямая реализация данного подхода, при которой полигональная модель рельефа формируется на стороне CPU для каждого кадра и отправляется на видеокарту, существенно ограничивает скорость работы системы визуализации, ввиду интенсивной нагрузки на канал CPU-GPU, являющийся узким местом. В связи с этим многие современные исследования в данной области направлены на реализацию адаптивной триангуляции сеток высот на GPU.

В работе [10] предложен метод ускорения адаптивной триангуляции рельефа с помощью GPU, основанный на применении 4-х типов геометрических шаблонов, предварительно записываемых в видеопамять. Авторы исследования [11] предложили ускорить построение адаптивной триангуляции путем генерации треугольников на GPU с помощью геометрического шейдера. С развитием аппаратно-программной архитектуры GPU у разработчиков появилась возможность управлять разбиением 3D-моделей на полигоны непосредственно на GPU (программируемая тесселяция), избегая покадровой передачи полигональных данных по каналу CPU-GPU, которая, так или иначе, присутствовала в более ранних решениях.

Данная технология основана на добавлении в графический конвейер GPU двух новых программируемых стадий – шейдера управления тесселяцией и шейдера вычисления тесселяции, а также на введении специальных параметрических графических примитивов (патчей) нескольких типов. В работе [12] продемонстрирован принципиальный подход к реализации программируемой тесселяции рельефа с помощью графической библиотеки Direct3D версии 11. В работе [13] предложен метод адаптивной тесселяции рельефа с помощью четырехугольных патчей (патчей-квадов) одинакового размера. Во избежание образования разрывов (cracks) в местах стыковки треугольников типа «вершина-сторона» в работах [4, 13] строятся незаметные треугольники-заслонки.

Как показала практика, реализации, основанные на патчах-квадах, склонны к образованию избыточной триангуляции в модели рельефа. Так, например, если в патч-квад попадает небольшой участок, требующий высокого уровня полигонального разбиения, то необходимо тесселировать до этого уровня весь патч-квад. И, если для сеток высот небольших размеров применение такой «грубой силы» компенсируется вычислительной мощностью видеокарты, то в случае детализированных сеток высот это приводит к существенному падению эффективности работы системы визуализации. Одно из направлений ухода от такой избыточности состоит во введении древовидных структур из патчей-квадов разных уровней детализации (см. например, работу [14]), просчитываемой для каждого кадра на CPU. Однако такое решение используют передачу данных по каналу CPU-GPU, интенсивность которой возрастает с увеличением размеров сетки высот. Как упоминалось выше, это является нежелательным узким местом для системы визуализации, работающей в масштабе реального времени.

В отличие от предыдущих исследований в данной работе предлагается новый метод адаптивной тесселяции рельефа с помощью патчей-треугольников разных уровней детализации, который работает полностью на GPU в масштабе реального времени. В отличие от патчей-квадов с помощью патчей-треугольников можно эффективно локализовать на рельефе участки, требующие повышенной детализации, избегая принудительного разбиения на треугольники прилегающих к ним областей. Для обеспечения безразрывной стыковки соседних патчей-треугольников в данной работе применяется близкий к работам [4, 13] подход, но в отличие от [4, 13] заполнение стыков выполняется в процессе вычисления тесселяции патчей-треугольников и не требует реализации дополнительного этапа расчета специальных треугольников. Предлагаемое решение реализовано программно с помощью графической библиотеки OpenGL и шейдерного языка GLSL 4.0.

2. ТЕХНОЛОГИЯ МОДЕЛИРОВАНИЯ РЕЛЬЕФА С ПОМОЩЬЮ ПАТЧЕЙ-ТРЕУГОЛЬНИКОВ

Основным строительным элементом, с помощью которого в данной работе выполняется моделирование и визуализация рельефа, является патч-треугольник. Он имеет ряд параметров, которыми управляется его разбиение на более мелкие треугольники. Это число равных отрезков, на которые будет разбита каждая из его сторон (внешние $l_{OUT,0}$, $l_{OUT,1}$, $l_{OUT,2}$ уровни тесселяции), а также число равных отрезков, на которые будут временно разбиты все стороны патча-треугольника для построения вложенных треугольников (внутренний уровень l_{IN} тесселяции). На рис. 1 изображен пример построения вложенных треугольников для $l_{IN} = 4$ (образуется 2 вложенных треугольника, один из которых вырожден в точку).

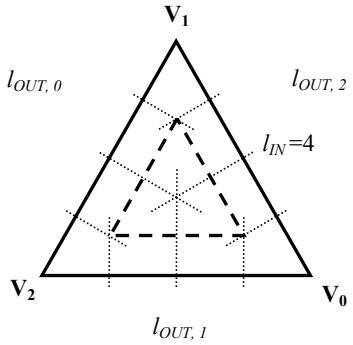

Рисунок 1: Патч-треугольник

В данной работе триангуляция сетки высот строится на основе патчей-треугольников 0, …, L-го уровней детализации, как показано в примере на рисунке 2. Для простоты рассматривается случай, когда в сетку высот укладывается целое число патчей-треугольников L-го уровня детализации.

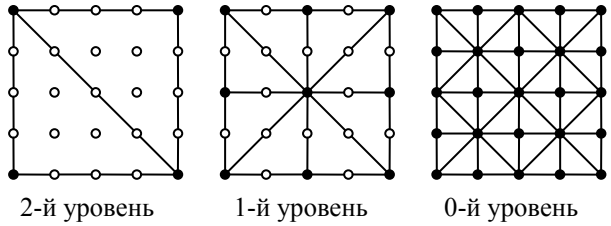

Рисунок 2: Разбиение сетки высот на патчи-треугольники 3-х уровней детализации

Технология моделирования рельефа состоит из следующих этапов. На первом этапе поверхность, относительно которой заданы высоты в сетке высот (уровенная поверхность), разбивается на патчи-треугольники L-го уровня детализации. Это выполняется один раз во время предварительной подготовки исходных данных. На втором этапе (при загрузке виртуальной сцены в систему визуализации) полученная (стартовая) полигональная модель и сетка высот записываются в видеопамять. Сетка высот хранится в видеопамяти в виде 2D-текстуры (карты высот). В ходе визуализации (на третьем этапе - просчете текущего кадра) выполняется параллельная обработка на GPU патчей-треугольников стартовой модели с помощью разработанной шейдерной программы. Данная программа включает в себя выполнение шейдера управления тесселяцией (Tesselation Control Shader, TCS), шейдера вычисления тесселяции (Tesselation Evaluation Shader, TES) и геометрического шейдера (Geometric Shader, GS).

В шейдере TCS для патча-треугольника определяется ограничивающий объем в виде треугольной призмы и выполняется его проверка на попадание в поле зрения виртуальной камеры. Не прошедший проверку патч-треугольник исключается из дальнейшей обработки путем обнуления значений всех его уровней тесселяции. Для прошедшего проверку патча-треугольника вычисляются значения $l_{OUT,0}$, $l_{OUT,1}$, $l_{OUT,2}$ на основе оценки экранной ошибки. Более подробно описание экранной ошибки и технологии ее оценки можно найти в работах [3, 6]. Если хотя бы одно из вычисленных значений $l_{OUT,0}$, $l_{OUT,1}$, $l_{OUT,2}$ превышает 1, то оно округляется до 2, а значение l_{IN} устанавливается равным 4, в противном случае l_{IN} равно 1. Таким образом, значения внешних уровней тесселяции могут быть 1 или 2, а внутреннего – 1 или 4. Также в шейдере TCS вычисляются флаги b_{AB}, b_{BC}, b_{AC} сторон $\triangle ABC$ (флаг для наибольшей стороны равен 1, остальные – 0).

После выполнения шейдера TCS генератор примитивов выполняет разбиение патча-треугольника на треугольники в соответствии с вычисленными значениями уровней тесселяции (это фиксированный этап графического конвейера с аппаратно зашитым алгоритмом). Вершины сгенерированных треугольников заданы барицентрическими координатами в точечном базисе исходного патча-треугольника.

В шейдере TES из треугольников, полученных на выходе генератора примитивов, вычисляются треугольники L-1-го уровня детализации. Это реализуется с помощью разработанного алгоритма коррекции барицентрических координат, который описывается в разделе 3.

В шейдере GS треугольники, полученные после шейдера TES, проходят обработку из трех шагов. На первом шаге треугольники проверяются на вырожденность (см. раздел 3). Вырожденные треугольники шейдер отбрасывает. На втором шаге у треугольников, не разбитых генератором примитивов (при $l_{OUT,0}$, $l_{OUT,1}$, $l_{OUT,2}$ равных 1), шейдер смещает вершины в соответствии с картой высот. Полученные новые треугольники добавляются в область видеопамяти, выделенную для хранения «меша» (полигональной модели) рельефа, - буфер B_{OUT}. На третьем шаге у разбитых генератором примитивов треугольников шейдер смещает вершины до уровенной поверхности, и полученные треугольники добавляются во вспомогательный буфер в видеопамяти.

После обработки всех патчей-треугольников L-го уровня детализации аналогичным образом выполняется шейдерная обработка треугольников L-1-го уровня детализации из вспомогательного буфера, и так далее до 0-го уровня или пока вспомогательный буфер не опустеет. По завершении процесса обработки патчей-треугольников в буфере B_{OUT} формируется искомая триангулированная модель рельефа.

3. ТЕССЕЛЯЦИЯ ПАТЧА-ТРЕУГОЛЬНИКА

Рассмотрим более подробно работу шейдера TES. Одна из ключевых задач адаптивной триангуляции сетки высот состоит в обеспечении безразрывной (непрерывной) стыковки треугольников разных уровней детализации. Генератор примитивов выполняет тесселяцию поступившего на его вход треугольника в соответствии со значениями $l_{OUT,0}$, $l_{OUT,1}$, $l_{OUT,2}$ и l_{IN}. В процессе тесселяции исходный треугольник либо не изменяется (т.е. тесселяции нет), либо

внутри него создается вложенный треугольник с вершиной внутри (вершина – вырожденный треугольник), и оба эти треугольника разбиваются на более мелкие (см. рис. 3а). При этом на сторонах исходного треугольника могут быть добавлены дополнительные вершины. Хотя в примере рис. 3а добавлена только одна вершина K, мы рассматриваем также возможные вершины на остальных сторонах (точки M и J). Все эти точки имеют следующие барицентрические координаты:

$A=(0,1,0)$, $B=(1,0,0)$, $C=(0,0,1)$, $D=(\alpha,4\alpha,\alpha)$, $E=(4\alpha,\alpha,\alpha)$, $F=(\alpha,\alpha,4\alpha)$, $G=(\beta,\beta,\alpha)$, $H=(\beta,\alpha,\beta)$, $I=(\alpha,\beta,\beta)$, $J=(\gamma,\gamma,0)$, $K=(\gamma,0,\gamma)$, $M=(0,\gamma,\gamma)$ и $O=(2\alpha,2\alpha,2\alpha)$, где $\alpha=1/6$, $\beta=5/12$, $\gamma=0.5$. Растянем вложенный треугольник так, чтобы он совпал с исходным, а вершину O сместим на середину наибольшей стороны $\triangle ABC$ (см. рис. 3б). Шейдер TES вычисляет барицентрические и декартовы координаты (в объектной системе координат модели рельефа) всех вершин, используя следующий параллельный

Алгоритм вычисления координат отдельной вершины тесселированного треугольника

1. Передаем в шейдер TES:
   - из генератора примитивов: барицентрические координаты P_{bar} вершины P в $\triangle ABC$;
   - из шейдера TCS: координаты A_{OCS}, B_{OCS}, C_{OCS} вершин $\triangle ABC$ в объектной системе координат рельефа (OCS) и флаги b_{AB}, b_{BC}, b_{AC}.
2. Вычисляем флаги $b_A,\ldots,b_O$ барицентрических координат $A,\ldots,O$ (флаг равен 1, если P_{bar} совпадает с соответствующей вершиной):
   $b_A=(P_{bar}==A)$, где « == » - операция равенства.
   Аналогично b_A вычисляем остальные флаги.
3. Вычисляем флаги $b_{A,D}$, $b_{B,E}$, $b_{C,F}$, $b_{J,G,AB}$, $b_{K,H,BC}$, $b_{M,I,AC}$ вершин $\triangle ABC$ и середин его сторон (флаг $b_{A,D}$ равен 1, если P_{bar} совпадает с вершиной A или D, а флаг $b_{J,G,AB}$ равен 1, если P_{bar} совпадает с вершиной J или G, или с вершиной O, а сторона AB - наибольшая):
   $b_{A,D}=b_A\,||\,b_D$, $b_{J,G,AB}=b_J\,||\,b_G\,||\,(b_O\,\&\&\,b_{AB})$,
   $b_{B,E}=b_B\,||\,b_E$, $b_{K,H,BC}=b_K\,||\,b_H\,||\,(b_O\,\&\&\,b_{BC})$,
   $b_{C,F}=b_C\,||\,b_F$, $b_{M,I,AC}=b_M\,||\,b_I\,||\,(b_O\,\&\&\,b_{AC})$,
   где « || » - логическое ИЛИ, « && » - логическое И. Легко видеть, что только один из этих флагов будет равен 1.
4. Вычисляем модифицированные барицентрические координаты P'_{bar} вершины P:
   $P'_{bar}=b_{A,D}A+b_{B,E}B+b_{C,F}C+b_{J,G,AB}J+b_{K,H,BC}K+b_{M,I,AC}M$.
   Таким образом, эти координаты будут совпадать с координатами одной из вершин линейной комбинации.
5. Вычисляем координаты P_{OCS} вершины P:
   $P_{OCS}=P'_{bar,x}\cdot A_{OCS}+P'_{bar,y}\cdot B_{OCS}+P'_{bar,z}\cdot C_{OCS}$.

Отметим, что в п. 2 алгоритма проверка на равенство выполняется с учетом машинной погрешности представления действительных чисел. На рисунке 3б показаны треугольники, полученные после применения разработанного алгоритма. Шейдер GS проверяет все полученные треугольники на вырожденность. Для этого из шейдера TES в шейдер GS для каждой вершины проверяемого треугольника передаются шестерки флагов $b_{A,D},\ldots,b_{M,I,AC}$, вычисленные в п. 3 алгоритма. В данной работе предлагается критерий *isDeg*, при котором треугольник считается вырожденным, если в нем не менее 2-х вершин имеют флаги одного типа, равные 1:

$$isDeg=\left(\sum_{i=0}^{2}b_{A,D}^{V_i}\ge 2\right)||\left(\sum_{i=0}^{2}b_{B,E}^{V_i}\ge 2\right)||\ldots||\left(\sum_{i=0}^{2}b_{M,I,AC}^{V_i}\ge 2\right),$$

где V_0, V_1, V_2 - вершины проверяемого треугольника. Вырожденные треугольники шейдер GS не создает на выходе. Треугольники, прошедшие проверку, шейдер GS обрабатывает, как описано в разделе 2, и добавляет либо в буфер B_{OUT} меша рельефа, либо во вспомогательный буфер.

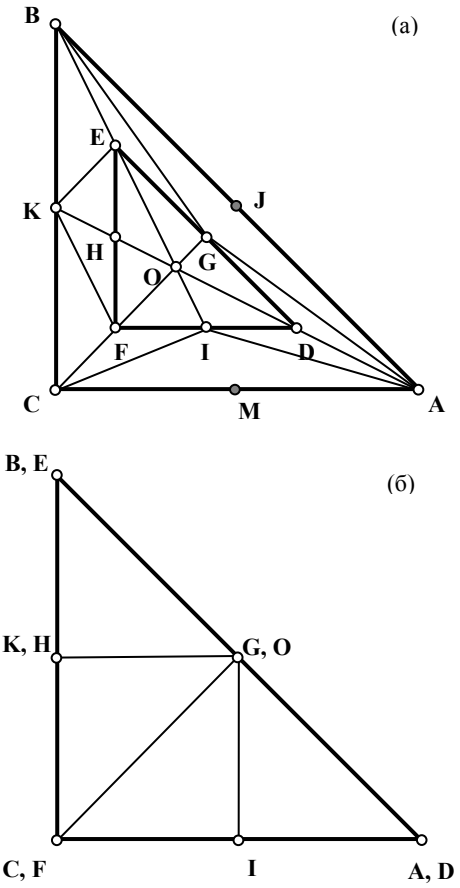

Рисунок 3: Расположение вершин треугольников, созданных генератором примитивов: (а) - до шейдерной обработки, (б) - после шейдерной модификации барицентрических координат

4. РЕЗУЛЬТАТЫ

Разработанная технология параллельной адаптивной тесселяции рельефа была реализована в программном комплексе визуализации виртуальной модели земной поверхности. Моделирование рельефа выполнялось на основе сетки высот размером $16K\times 8K$ узлов с количеством уровней детализации

патчей-треугольников, равным 6. На рисунке 4 приведен пример синтезированных в реальном времени изображений Земли с высоты 300 км. Применение разработанного решения позволило уменьшить среднее количество отправляемых на визуализацию треугольников с 2,5 млн. до 130 тыс. Среднее время расчета модели рельефа составило около 2 миллисекунд на кадр. Апробация проводилась с помощью видеокарты GeForce GTX Titan Black при разрешении Full HD. Анализ результатов в различных имитационных сценариях показал, что предложенное решение по скорости расчета модели рельефа сопоставимо с существующими быстрыми методами тесселяции и при этом обладает некоторыми преимуществами при работе с детализированными картами высот.

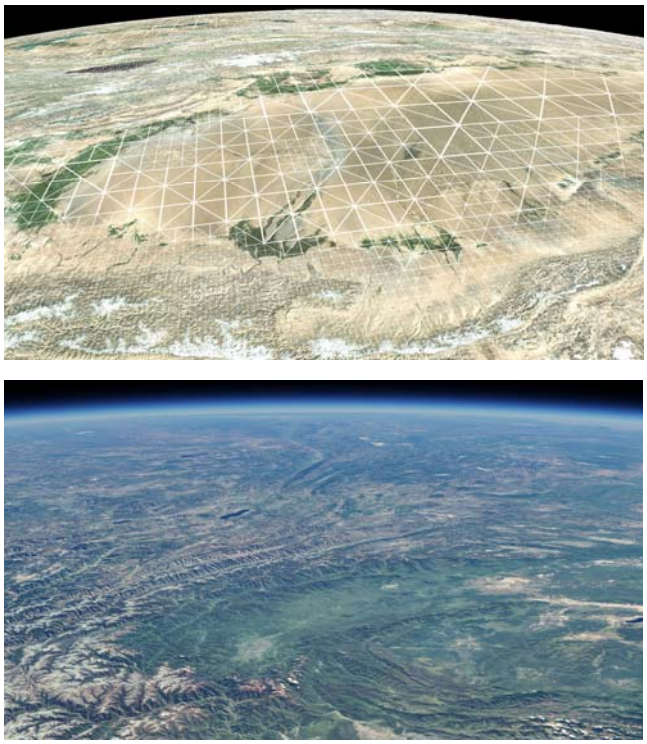

Рисунок 4: Пример синтезированного изображения модели Земли с триангуляцией рельефа (сверху) и с освещением, рассчитанным с учетом атмосферы (внизу)

Моделирование геометрии рельефа земной поверхности по сравнению с более простыми имитационными техниками (бамп- и нормалмаппинг) дает возможность решать ряд важных задач: осуществлять расчет высокореалистичной светотеневой обстановки с учетом теней от гор и в долинах («световых колодцев») при перемещении Солнца, а также моделировать корректный вид рельефа земной поверхности при использовании виртуальных средств наблюдения с высокой кратностью увеличения, что очень актуально, например, для космических видеотренажерных комплексов.

5. ЗАКЛЮЧЕНИЕ

В работе предложена новая технология реализации адаптивной триангуляции регулярных сеток высот, которая работает полностью на GPU в масштабе реального времени. Построение триангуляции сетки высот выполняется путем управляемой параллельной тесселяции на GPU патчей-треугольников различных уровней детализации. В работе предложен алгоритм коррекции барицентрических координат треугольников, образуемых в результате тесселяции, который обеспечивает стыковку соседних треугольников в модели рельефа без разрывов в геометрии.

Разработанное решение имеет хорошую масштабируемость и обеспечивает высокую локальную адаптивность получаемой на выходе триангуляции рельефа. На основе созданной технологии, методов и алгоритмов был реализован программный комплекс визуализации детализированного рельефа всей земной поверхности в масштабе реального времени. Созданный комплекс был успешно апробирован в составе сложных многообъектных виртуальных сцен, содержащих порядка миллиона полигонов.

В качестве следующего шага планируется развитие разработанного решения для работы со сверхбольшими сетками высот (размер и объем которых превышает аппаратные ограничения графических ускорителей).

6. БЛАГОДАРНОСТИ

Работа была выполнена при финансовой поддержке РФФИ в рамках гранта № 16-07-01104.

7. ССЫЛКИ

[1] М. В. Михайлюк, М. А. Торгашев. *Система «GLVIEW» визуализации для моделирующих комплексов и систем виртуальной реальности. Вестник Российской академии естественных наук, №2, 2011, – стр. 20-28.*

[2] В. Л. Ерухимов, А. Д. Капустин, А. В. Малашкина. *Реализация алгоритма визуализация местности в режиме реального времени // Материалы конф. по комп. граф. и визуализации «ГрафиКон'99». – Москва, 1999. – стр. 110-116.*

[3] W. H. de Boer. *Fast terrain rendering using geometrical mipmapping*, 2000 (http://www.flipcode.com/articles/article_geomipmaps.pdf).

[4] T. Ulrich. *Rendering massive terrains using chunked level of detail control. SIGGRAPH Course Notes, vol. 3, № 5, 2002.*

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

[6] А. В. Аниченков, М. В. Михайлюк. *Визуализация и морфинг виртуальных ландшафтов. Информационные технологии и вычислительные системы, № 2, 2004, – стр. 86-93.*

[7] L. M. Hwa, M. A. Duchaineau and 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, № 4, pp. 355–368, 2005.*

[8] A. Asirvatham, H. Hoppe. *Terrain rendering using GPU-based geometry clipmaps. GPU Gems, vol. 2, № 2, pp. 27–46, 2005.*

[9] M. Clasen, H.-C. Hege. *Terrain rendering using spherical clipmaps. Eurographics/IEEE-VGTC Symposium on Visualization, 2006.*

[10] Y. Livny, Z. Kogan, J. El-Sana. *Seamless patches for GPU-based terrain rendering. The Visual Computer, vol. 25, pp. 197–208, 2008.*

[11] O. Ripolles, F. Ramos, A. Puig-Centelles, M. Chover. *Real-time tessellation of terrain on graphics hardware. Comput. Geosci. 41, pp. 147-155, 2012.*

[12] I. Cantlay. *Directx 11 terrain tessellation. NVIDIA WhitePaper, 2011.*

[13] E. Yusov, M. Shevtsov. *High-performance terrain rendering using hardware tessellation. WSCG 19(3), pp. 85–92, 2011.*

[14] H.Y. Kang, H. Jang, C.S. Cho, J.H. Han. *Multi-resolution terrain rendering with GPU tessellation. The Visual Computer, 2014, vol. 31, № 4, pp. 455-469.*

Сведения об авторах

Михайлюк Михаил Васильевич, зав. отделом, Научно-исследовательский институт системных исследований РАН, mix@niisi.ras.ru.

Тимохин Петр Юрьевич, научный сотрудник, Научно-исследовательский институт системных исследований РАН, webpismo@yahoo.de.

Мальцев Андрей Валерьевич, старший научный сотрудник, Научно-исследовательский институт системных исследований РАН, avmaltcev@mail.ru.