ПРОГРАММИРОВАНИЕ, 2022, № 3, с. 50–58

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.925.3

МОДЕЛИРОВАНИЕ ДИНАМИЧЕСКИХ ТЕНЕЙ РЕЛЬЕФА В МАСШТАБЕ РЕАЛЬНОГО ВРЕМЕНИ НА ОСНОВЕ МНОГОУРОВНЕВОГО РЭЙКАСТИНГА

© 2022 г. П. Ю. Тимохин[a,*] (ORCID: 0000-0002-0718-1436),
М. В. Михайлюк[a,] (ORCID: 0000-0002-7793-080X)**

[a]ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН»
117218 Москва, Нахимовский пр., 36, к.1, Россия
**E-mail: webpismo@yahoo.de*
***E-mail: mix@niisi.ras.ru*

Поступила в редакцию 25.12.2021 г.
После доработки 16.01.2022 г.
Принята к публикации 21.01.2022 г.

В данной статье рассматривается задача моделирования в масштабе реального времени динамических теней рельефа местности на основе многоуровневого рэйкастинга. Предлагается технология, при которой выполняется попиксельное управление детализацией синтезируемых теней, а также проверка на пересечение солнечных лучей не только с наибольшими, но и с наименьшими высотами участков рельефа. Описываются методы и алгоритмы реализации предложенной технологии на GPU. Предложенное решение позволяет повысить скорость визуализации сложных форм рельефа, таких как ущелья и кратеры, а также повысить эффективность визуализации рельефа в областях вывода, имеющих размеры, меньшие, чем у исходной карты высот. Созданные технология, методы и алгоритмы реализованы в программном комплексе; его апробация подтвердила адекватность предложенного решения поставленной задаче. Полученные результаты могут быть применены в системах виртуального окружения, видеотренажерах, системах научной визуализации, виртуальных глобусах, геоприложениях и др.

DOI: 10.31857/S0132347422030104

1. ВВЕДЕНИЕ

Моделирование динамических теней рельефа местности является одной из актуальных задач современных *систем виртуального окружения* (СВО), в частности, космических видеосимуляторов [1–3]. Для таких систем характерно динамическое изменение положения виртуального наблюдателя и источника света (Солнца), а также одновременная визуализация из нескольких виртуальных камер, имитирующих реальные средства наблюдения (иллюминаторы, телекамеры с широким и узким углами обзора, фотокамеры с зумом и др.) [4].

В условиях динамически меняющейся виртуальной обстановки надежным подходом к синтезу качественных и геометрически точных теней рельефа местности является *рэйкастинг* [5] – частный случай трассировки лучей, широко применяемой в области фотореалистичного рендеринга [6]. При данном подходе изображения теней просчитываются на основе исходной цифровой модели рельефа – *карты высот* и их качество не зависит от методов построения и визуализации 3D=модели рельефа [7], которые могут содержать неточности и скрытые уязвимости. В классической реализации [8] производительность рэйкастинга сильно падает с увеличением размеров карты высот. На практике, в случае детализированных карт высот (от 1Kx1K) просчет теней может нарушить интерактивность визуализации, что неприемлемо для СВО. В этой связи возникает задача разработки эффективных технологий, методов и алгоритмов реализации рэйкастинга, обеспечивающих построение теней рельефа на основе детализированных карт высот в масштабе реального времени (не более 40 мс на кадр). Для решения этой задачи в данной работе предлагается технология рэйкастинга теней, при которой детализация теней рельефа понижается в областях изображения, где это незаметно для наблюдателя. Просчет рэйкастинга на каждом уровне детализации тени выполняется на GPU и ускорен с помощью карты локальных экстремумов высоты. Предлагаемое решение реализовано на языке C++ с использованием языка GLSL программи-

рования шейдеров и графической библиотеки OpenGL.

2. СОВРЕМЕННЫЕ ПОДХОДЫ

Можно выделить три крупных подхода к моделированию теней рельефа в реальном времени.

Первый подход основан на построении *теневых карт* виртуальной сцены из позиции источника света и их применении при визуализации сцены из позиции наблюдателя. Одним из наиболее распространенных является метод каскадных карт теней, при котором создается ряд теневых карт одинакового размера для переднего, промежуточных и заднего планов [9, 10]. Благодаря высокой производительности, реализации данного метода часто встречаются в современных играх. Особенностью таких реализаций является необходимость тщательной настройки под конкретный игровой сценарий во избежание появления характерных визуальных артефактов (“ступенчатости” границ теней, ошибочного самозатенения, усечения теней и др.). Для повышения качества синтезируемых теней продвинутые реализации используют процентно-приближенную фильтрацию [11], нерегулярный Z-буфер [12], нелинейное отображение теней [13, 14] и другие решения. Важно отметить, что надежность данного подхода зависит не только от самого алгоритма построения тени, но и от надежности методов построения и визуализации модели рельефа местности. Они могут содержать оптимизации или уязвимости, скрытые в процессе визуализации рельефа, но проявляющиеся при визуализации его тени (в нашем исследовании [5] продемонстрирован пример такой ситуации).

При *втором подходе* построение теней выполняется на основе проверки попадания точек в *теневой объем* – область пространства, закрываемую рельефом от источника света. Распространенная реализация данного подхода предполагает вытягивание полигональной модели теневого объема из силуэта рельефа и подсчет пересечений лучей, испущенных из проверяемых точек, с гранями модели теневого объема с помощью буфера трафарета [15]. Благодаря доступу непосредственно к геометрии рельефа, подход теневых объемов позволяет получать качественные, геометрически точные тени для любого положения наблюдателя. Однако, при этом в виртуальной сцене генерируются сложные вспомогательные геометрические объекты, обработка которых может истратить существенную часть ресурсов, отводимых для работы всей системы визуализации. Кроме того, имеется ряд ограничений (на допустимые представления геометрии рельефа, на попадание в теневой объем ближней/дальней отсекающих плоскостей камеры и др.), которые приводят к усложнению структур данных и отрицательно влияют на масштабируемость системы. В направлении преодоления этих недостатков были предложены методы, основанные на комбинации алгоритмов Z-pass и Z-fail обновления буфера трафарета [16], извлечении силуэтов объектов с помощью геометрического шейдера [17], построении теневого объема по треугольникам с помощью программно-аппаратной архитектуры CUDA параллельных вычислений на GPU [18], ускорении детектирования силуэтов объектов сцены с помощью хэш-таблиц [19] и др.

Третий подход основан на поиске пересечений лучей источника света с поверхностью рельефа, заданной картой высот (*рэйкастинг теней*). Согласно классическому алгоритму рэйкастинга [8] для каждого пиксела изображения рельефа необходимо построить луч направления на источник света и проверить вдоль него друг за другом текселы карты высот, пока не будет найдена высота рельефа, пересекающая луч (или достигнута граница карты высот). Это позволяет получать физически корректные тени рельефа сложной формы, независимо от надежности методов его моделирования и визуализации [20]. Сравнительно долгое время медленная скорость работы на детализированных картах высот существенно ограничивала широкое применение данного подхода в компьютерной графике реального времени. Однако ситуация изменилась, благодаря активному прогрессу архитектуры и технологий параллельных вычислений на GPU, которые дали новый толчок в развитии методов, основанных на трассировке лучей [21]. В работе [22] был предложен метод ускорения поиска пересечений луч-карта высот с помощью конической карты, рассчитываемой на этапе предобработки. На практике, время построения такой карты резко возрастает с увеличением размеров карты высот (около 15 минут для карты 512 × 512 и около 8 часов для карты 1024 × 1024 [23]). В исследовании [23] разработан метод ускорения на GPU рэйкастинга динамической модели рельефа, заданного картой высот, с помощью мип-карты наибольших высот. Автор работы [20] описал гибридный метод ускорения рэйкастинга теней рельефа за счет пропуска проверки точек, не принадлежащих силуэту рельефа (если смотреть из позиции источника света). В исследовании [24] описана реализация распараллеливания рейкастинга теней рельефа с помощью архитектуры CUDA, где выборка текселов вдоль лучей выполняется с одинаковым шагом. В недавней работе [25] предложен алгоритм реализации рэйкастинга теней рельефа, основанный на [23]. Как утверждают авторы, в текущей реализации их алгоритм имеет некоторые неточности, ухудшающие качество теней, что приходится компенсировать сдвигом вершин рельефа.

В данной работе в качестве отправной точки мы взяли идею ускорения рэйкастинга за счет пропуска участков карты высот, если их наиболь-

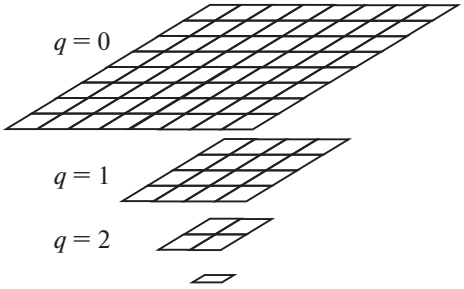

Рис. 1. Мип-пирамида карты высот.

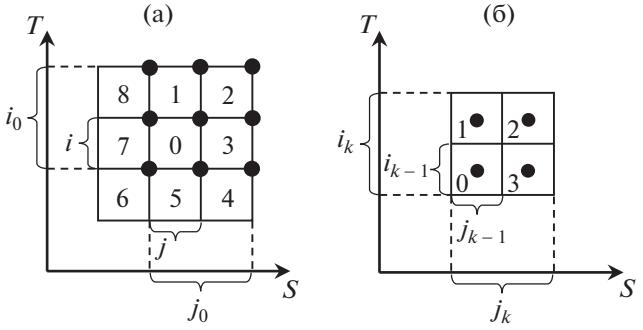

Рис. 2. Матрицы выборки для построения карты локальных экстремумов: (а) 0-го уровня, (б) уровня $k > 0$.

шая высота не пересекает солнечный луч. Мы расширили этот подход, добавив в него проверку на пересечение с наименьшей высотой участка, что позволило досрочно завершать проверку лучей внутри вогнутых объектов рельефа (долин, ущелий, кратеров и т.п.). Информацию о наибольших и наименьших высотах мы храним в иерархической структуре данных – карте локальных экстремумов высоты, которую строим на этапе предобработки данных. Другим нововведением является определение для каждого пиксела необходимого минимального уровня детализации карты высот, по которой будет выполняться проверка пересечений высот и солнечных лучей. Такой подход (мы назвали его *многоуровневым рэйкастингом теней*) дает возможность управлять детализацией теней и существенно экономить вычислительные ресурсы, например, при выводе изображения рельефа в области вывода небольших размеров, при отображении дальнего плана рельефа и т.д. В работе описана технология и методы реализации данного подхода на GPU.

3. ТЕХНОЛОГИЯ МНОГОУРОВНЕВОГО РЭЙКАСТИНГА ТЕНЕЙ

В данной работе мы будем рассматривать задачу многоуровневого рэйкастинга теней участка рельефа по детализированной карте высот размера $2^m \times 2^m$ текселов ($m \geq 10$) без учета кривизны поверхности уровня моря. Предлагаемая технология включает в себя следующие две стадии.

На **первой стадии** (предварительной обработки данных) создается мип-пирамида [26] детализированной карты высот (см. рис. 1) и карта локальных экстремумов высоты (далее карта H_{ext}). Карта H_{ext} представляет собой упорядоченный набор двухканальных текстур (R и G компоненты), обладающий следующими свойствами:

– у 0-й (самой первой) текстуры ширина и высота в 2 раза меньше, чем у исходной карты высот;

– в каждом (i_0, j_0)-м текселе 0-й текстуры хранятся наибольшее h_{max} и наименьшее h_{min} значения, выбранные из исходной карты высот с помощью матрицы 3×3, показанной на рисунке 2а;

– ширина и высота k-й текстуры ($k \neq 0$) в 2 раза меньше, чем у $(k-1)$-й текстуры (текстура 1×1 не создается);

– в каждом (i_k, j_k)-м текселе k-й текстуры хранятся наибольшее h_{max} и наименьшее h_{min} значения, выбранные из $(k-1)$-й текстуры с помощью матрицы 2×2, показанной на рис. 2б.

Отметим, что в матрице 3×3 точками выборки являются правые верхние углы 0, ..., 8-го текселов карты высот, и выборка выполняется с помощью билинейной интерполяции (GL_LINEAR). В матрице 2×2 точками выборки являются центры 0, ..., 3-го текселов $(k-1)$-й текстуры и выборка выполняется по принципу “ближайший сосед” (GL_NEAREST). Построение карты H_{ext} выполняется на GPU с помощью разработанного фрагментного шейдера.

На **второй стадии** (визуализации) выполняется синтез изображения затененной модели рельефа. Это реализуется в два шага. На *первом шаге* для каждого пиксела синтезируемого изображения вычисляется наименьший необходимый уровень q детализации карты высот. На *втором шаге* на основе вычисленного уровня q, мип-пирамиды карты высот и карты H_{ext} для каждого пиксела вычисляется коэффициент k_{shadow} затенения. Рассмотрим методы реализации этих шагов.

3.1. Метод вычисления минимального уровня детализации карты высот

Расчет наименьшего необходимого уровня q детализации карты высот реализуется на основе подсчета числа текселов, которое охватывает проекция пиксела (ее стороны) в текстурной системе координат карты высот. На практике используется эффективное приближение такой проекции в виде параллелограмма (рис. 3), а его длины сторон вычисляются с помощью частных производных текстурных координат (s, t) пиксела по X и Y экранной системы координат (рис. 3а) [27]. Также в данной работе выполняется компенсация анизотропного изменения сторон паралле-

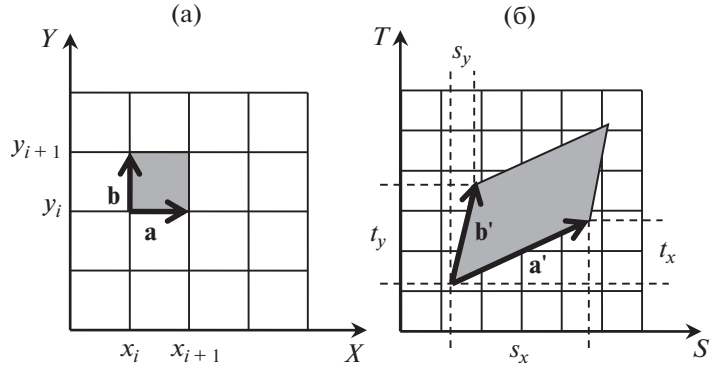

Рис. 3. Отображение пиксела: а) в экранной системе координат; б) в текстурной системе координат карты высот.

лограмма [28], возникающего при наблюдении карты высот под малым углом. Расчет минимального уровня q реализует следующий алгоритм ***Algo 1***:

1. Вычислим частные производные текстурных координат (s, t) пиксела:

$$s_x = w\frac{\partial s}{\partial x}, s_y = w\frac{\partial s}{\partial y}, t_x = h\frac{\partial t}{\partial x}, t_y = h\frac{\partial t}{\partial y},$$ где w и h – ширина и высота карты высот, в текселах.

2. Вычислим квадраты наибольшей d_{max} и наименьшей d_{min} длин сторон параллелограмма (см. рис. 3б):

$$d^2_{\max} = \max(|\mathbf{a}'|^2, |\mathbf{b}'|^2), \quad d^2_{\min} = \min(|\mathbf{a}'|^2, |\mathbf{b}'|^2),$$ где $|\mathbf{a}'|^2 = s_x^2 + t_x^2$, $|\mathbf{b}'|^2 = s_y^2 + t_y^2$.

3. Вычислим коэффициент n_a компенсации анизотропного изменения:

$$n_a = \min\left(\left\lceil \sqrt{d^2_{\max}/d^2_{\min}} \right\rceil, n_{a,\max}\right),$$ где $n_{a,\max}$ – наибольшее число выборок при анизотропной фильтрации текстуры, поддерживаемое видеокартой.

4. Вычислим минимальный уровень q детализации:

$$q = \lfloor \log_2(d_{\max}/n_a) \rfloor = \lfloor 0.5\log_2(d^2_{\max}) - \log_2(n_a) \rfloor.$$

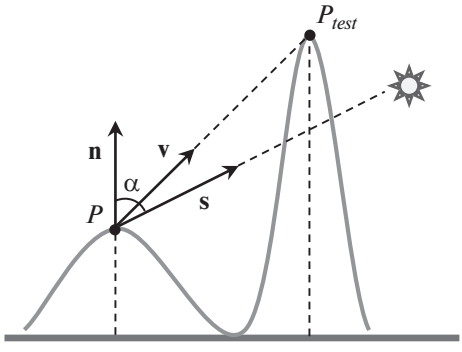

Рис. 4. Затенение точки P.

5. Уточним уровень q с учетом мип-пирамиды карты высот:

$q = \min(\max(q, 0), L)$, где L – самый грубый уровень детализации мип-пирамиды карты высот.

Конец алгоритма.

3.2. Метод вычисления коэффициента затенения пиксела

Определение коэффициента затенения. Обозначим через b_{shadow} булевский флаг наличия тени в рассматриваемом пикселе (1 означает, что тень есть, 0 – нет). Введем коэффициент k_{shadow} затенения пиксела, определяемый по следующей формуле

$$k_{shadow} = (k_{dark} + (k_{bright} - k_{dark})\cos\alpha)^{b_{shadow}} = \\ = (k_{dark} + (k_{bright} - k_{dark})(\mathbf{s}\cdot\mathbf{n}))^{b_{shadow}}, \quad (1)$$

где α – угол между вектором $\mathbf{n}$ нормали к поверхности уровня моря в точке P рельефа (см. рис. 4), соответствующей рассматриваемому пикселу, и единичным вектором $\mathbf{s}$ направления на солнце (бесконечно удаленный источник света); k_{dark} – коэффициент затенения, соответствующий наиболее плотной тени, а k_{bright} – наиболее разреженной, $k_{dark}, k_{bright} \in [0, 1]$. Значения k_{dark} и k_{bright} выбираются, исходя из диапазона яркостей, в котором выполняется рендеринг виртуальной сцены.

Вычисление флага b_{shadow}. Чтобы вычислить k_{shadow} с помощью выражения (1), необходимо найти значение флага b_{shadow}. Вначале определим b_{shadow}. Для этого выберем точку P_{test} рельефа, лежащую в плоскости $\{\mathbf{s}, \mathbf{n}, P\}$ между точкой P и источником света и проведем луч $\mathbf{v}$ из P в P_{test}. Из рисунка 4 видно, что точка P будет затенена, если луч $\mathbf{v}$ будет направлен не ниже вектора $\mathbf{s}$, т.е. будет неположительным смешанное произведение $p = \mathbf{r}\cdot(\mathbf{v}\times\mathbf{s})$, где $\mathbf{r} = \mathbf{s}\times\mathbf{n}$ – единичный вектор, дополняющий $\mathbf{s}$ и $\mathbf{n}$ до правой тройки. Исходя из этого, запишем определение:

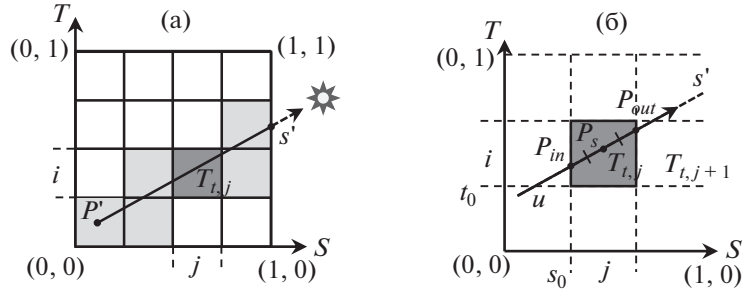

Рис. 5. Тексел $T_{i,j}$, лежащий на трассе солнечного луча (а), транзитные точки $T_{i,j}$-го тексела (б).

$$b_{shadow} = \begin{cases} 1, \text{ если } p \le 0 \text{ и } (\mathbf{s} \cdot \mathbf{n}) \ne 1, \\ 0, \text{ в остальных случаях.} \end{cases} \quad (2)$$

Вычисление значения флага b_{shadow} будем выполнять на q-м уровне детализации мип-пирамиды карты высот (далее – карта H_q). Обозначим через P' точку на карте H_q, соответствующую точке P рельефа, через s' – трассу солнечного луча, проведенную из точки P' в системе текстурных координат карты H_q. Рассмотрим некоторый $T_{i,j}$-й тексел, лежащий на трассе s' (см. рис. 5а). Данный тексел отсекает от трассы s' участок $P_{in}P_{out}$ (см. рис. 5б), которому соответствует своя кривая на поверхности рельефа (далее – кривая g). Теоретически, чтобы найти значение флага b_{shadow} в $T_{i,j}$-м тексе, необходимо выбрать в качестве точки P_{test} (см. рис. 4) наиболее высокую точку кривой g и посчитать смешанное произведение из выражения (2). Наши эксперименты показали, что в случае детализированных карт высот является допустимым (не образует "рваные" края у синтезируемых теней) приближение, при котором в качестве P_{test} берется средняя точка кривой g. Соответствующая ей точка P_s на карте H_q показана на рис. 5б. В данной работе мы вычисляем координаты точки P_s как среднее координат *транзитных точек* – точек P_{in} входа и P_{out} выхода трассы s' из $T_{i,j}$-го тексела. Из рисунка 5б видно, что точка P_{out} выхода из $T_{i,j}$-го тексела будет точкой P_{in} входа в следующий $T_{i,j+1}$-й тексел, таким образом, при проверке каждого $T_{i,j}$-го тексела достаточно вычислить только транзитную точку P_{out}. Обозначим через $\mathbf{u}$ единичный вектор направления трассы s', тогда координаты точки P_{out} можно записать в параметрическом виде

$$P_{out} = P_{in} + \mathbf{u}t. \quad (3)$$

Вычисление параметра t и сдвигов (i_{offset}, j_{offset}) номеров строки и столбца следующего по трассе тексела реализует следующий разработанный алгоритм ***Algo 2***:

1. Сформируем массив K текстурных координат четырех угловых точек $T_{i,j}$-го тексела:

K [4] = $\{\{s_0, t_0\}, \{s_0 + ds, t_0\}, \{s_0, t_0 + dt\}, \{s_0 + ds, t_0 + dt\}\}$, где $ds = 2'/w$, $dt = 2'/h$, а $l = q, ..., L$.

2. Если $|u_x| \le \varepsilon$ и $|u_y| \le \varepsilon$, то: $t = -1$, $i_{offset} = 0$, $j_{offset} = 0$ (ε – машинная погрешность действительных чисел), выходим из алгоритма.

3. Вычислим порядковый номер n_{out} угла $T_{i,j}$-ого тексела, из которого выходит трасса s':

$n_{out} = b_0 + 2b_1$, где b_0, b_1 – булевские флаги: $b_0 = (|u_x| \ge 0))$, $b_1 = (|u_y| \ge 0)$.

4. Запишем разность $dP = K[n_{out}] - P_{in}$.

5. Если $|u_x| \le \varepsilon$, то: $t = |dP_y|/|u_y|$, $i_{offset} = sign(u_y)$, $j_{offset} = 0$ ($sign$ – функция, возвращающая знак переменной), выходим из алгоритма.

6. Если $|u_y| \le \varepsilon$, то: $t = |dP_x|/|u_x|$, $i_{offset} = 0$, $j_{offset} = sign(u_x)$, выходим из алгоритма.

7. Запишем $e = dP/u$ и булевские флаги $b_2 = (e_x \le e_y)$, $b_3 = (e_y \le e_x)$.

8. $t = \min(e_x, e_y)$, $i_{offset} = b_3 sign(u_y)$, $j_{offset} = b_2 sign(u_x)$.

Конец алгоритма.

Подставив полученное значение t в выражение (3), получим координаты точки P_{out}, а, следовательно, и координаты P_s, по которым извлечем из карты H_q значение h_{test} высоты точки P_{test}. Зная высоту h_{test} и нормаль $\mathbf{n}$, нетрудно вычислить координаты точки P_{test} (координаты точки P вычисляются аналогично) и получить значение флага b_{shadow} с помощью выражения (2).

Поиск ненулевого флага b_{shadow}. Для этого на основе разработанного алгоритма *Algo 2* мы проверяем тексели вдоль трассы s', пытаясь с помощью карты H_{ext} локальных экстремумов "проскочить" крупные участки карты высот, начиная с самого грубого уровня детализации. Если проверка по карте H_{ext} показывает, что нет пересечения солнечного луча s с наименьшей высотой, но есть пересечение с наибольшей высотой, то мы спускаемся на более детальный уровень карты H_{ext} и так

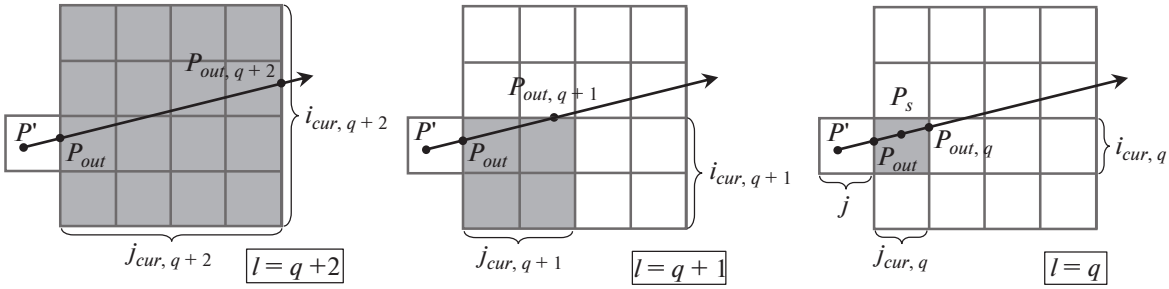

Рис. 6. Продвижение вдоль трассы солнечного луча на карте высот q-го уровня.

далее до q-го уровня детализации (см. рис. 6). На q-м уровне проверяется уже непосредственно тексел карты H_q высот. Для выполнения этих проверок введем следующие обозначения: l – уровень детализации: $l = q$ карта высот, $q+1$, $q+2$, ..., L – это q-й, $(q+1)$-й, ..., $L-1$ уровни карты H_{ext}; (i, j) – номера строки и столбца (далее – номера) стартового текселя на q-м уровне; $(i_{cur,\,l}, j_{cur,\,l})$, $(i_{next,\,l}, j_{next,\,l})$ – номера текущего и следующего (по трассе солнечного луча) текселов на l-м уровне; P_{out} – точка выхода трассы луча из стартового текселя; $P_{out,\,l}$ – точка выхода трассы луча из $(i_{cur,\,l}, j_{cur,\,l})$-го текселя; P – координаты точки модели рельефа (в системе OCS), соответствующей точке P' на карте высот; $P_{min,\,l}$, $P_{max,\,l}$ – координаты наиболее низкой и высокой точек участка модели рельефа (в системе OCS), соответствующие $(i_{cur,\,l}, j_{cur,\,l})$-му текселу (на q-м уровне $P_{min,\,l}$ совпадает с $P_{max,\,l}$). Поиск ненулевого значения флага b_{shadow} реализует следующий разработанный алгоритм ***Algo 3***:

1. Стадия инициализации:

Вычислим P и (i, j); зададим $P_{in} = P'$ в *Algo 2*; вычислим P_{out} и $(i_{next,\,0}, j_{next,\,0})$ согласно *Algo 2*; зададим $(i_{cur,\,q}, j_{cur,\,q}) = (i_{next,\,q}, j_{next,\,q})$, $P_{in} = P_{out}$ в *Algo 2* и $l = q$.

2. Стадия многоуровневого рэйкастинга:

Цикл, пока хотя бы один из номеров $(i_{cur,\,q}, j_{cur,\,q})$ не выйдет за пределы карты высот:

Вычислим $(i_{cur,\,l}, j_{cur,\,l})$ из $(i_{cur,\,q}, j_{cur,\,q})$. Вычислим $P_{out,\,l}$ и $(i_{next,\,l}, j_{next,\,l})$ согласно *Algo 2*.

Если $l > q$, то выполним проверку пересечения луча **s** с наименьшей высотой:

Вычислим $P_{min,\,l}$ на основе $P_{out,\,l}$ и высоты h_{min}, выбранной из $(i_{cur,l}, j_{cur,\,l})$-го текселя.

Вычислим для луча $\mathbf{v} = (P_{min,\,l} - P)$ значение флага b_{shadow} согласно (2);

Если b_{shadow} равен 1, то выходим из алгоритма.

Выполним проверку пересечения луча ***s*** с наибольшей высотой:

Если l равен q, то вычислим $P_{max,\,l}$ на основе P_s (см. рис. 5б) и высоты h_{max}, выбранной из $(i_{cur,\,l}, j_{cur,\,l})$-го текселя карты высот H_q.

В противном случае: вычислим $P_{max,\,l}$ на основе $P_{out,\,l}$ и h_{max}, выбранной из $(i_{cur,\,l}, j_{cur,\,l})$-го текселя карты H_{ext} локальных экстремумов;

Вычислим для луча $\mathbf{v} = (P_{max,\,l} - P)$ значение флага b_{shadow} согласно (2);

Если b_{shadow} равен 1, то:

Если l равен q, то выходим из цикла, в противном случае $l = l - 1$.

В противном случае: зададим $(i_{cur,\,l}, j_{cur,\,l}) = (i_{next,\,l}, j_{next,\,l})$; вычислим $(i_{cur,\,0}, j_{cur.0})$ из $P_{out,\,l}$; зададим $P_{in} = P_{out,\,l}$ в *Algo 2* и $l = L$.

Конец цикла.

$b_{shadow} = 0$.

Конец алгоритма.

Отметим, что в приведенном алгоритме извлечение значений высот из карты H_q высот производится с помощью билинейной интерполяции, а из карты H_{ext} локальных экстремумов – без интерполяции (по номерам строки и столбца текселя). В результате выполнения алгоритма *Algo 3* мы по-

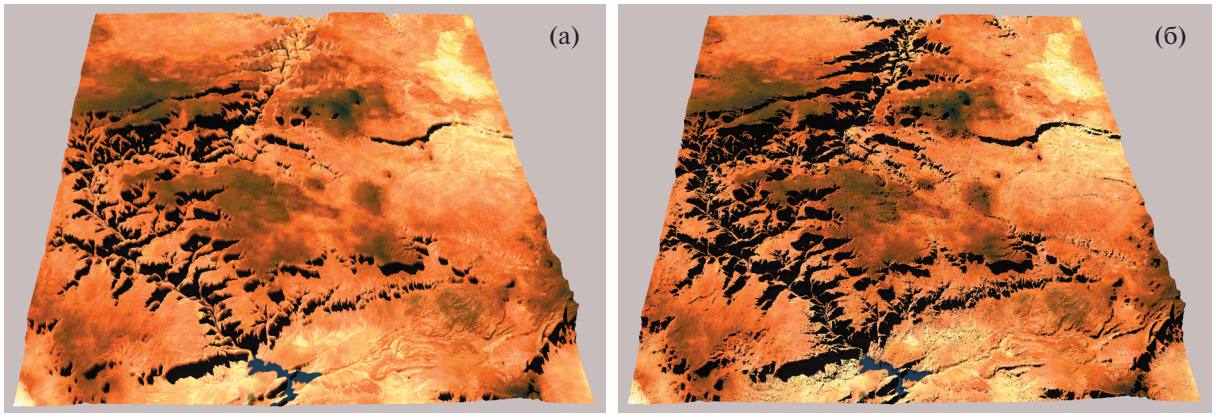

Рис. 7. Построение теней Великого Каньона: (а) каскадные карты теней (4 плоскости отсечения, размеры карт 4K × 4K), и (б) наша технология многоуровневого рэй-кастинга ($q = 0$)

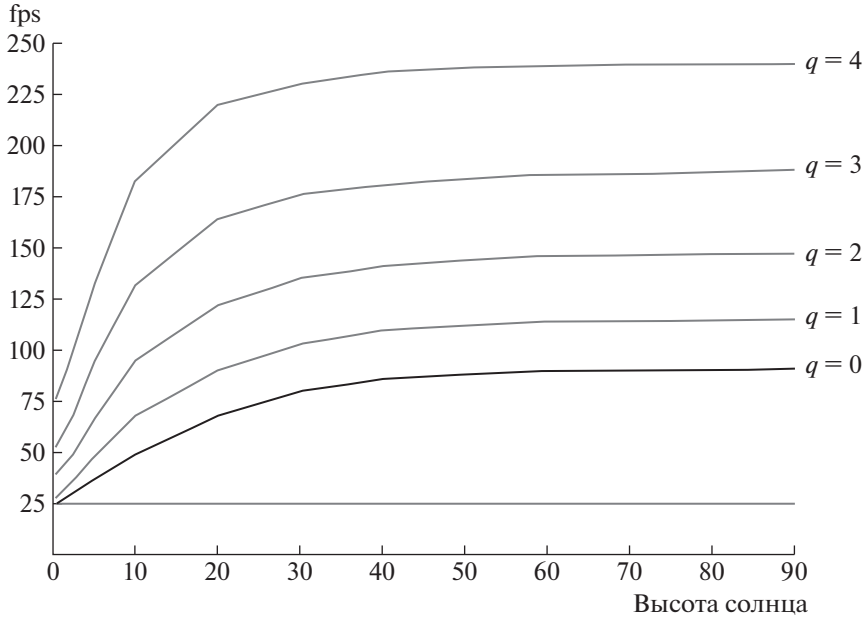

Рис. 8. Изменение частоты визуализации модели рельефа Великого Каньона (наша технология многоуровнего рэйкастинга) при уменьшении детализации карты высот и увеличении высоты солнца (0 – на горизонте, 90 – в зените).

лучаем значение флага b_{shadow} наличия тени для рассматриваемого пиксела. Подставив значение b_{shadow} в формулу (1), мы получаем искомое значение коэффициента затенения k_{shadow}. Описанный в данной работе процесс вычисления уровня q и коэффициента k_{shadow} для каждого пиксела изображения рельефа выполняется полностью на GPU, параллельно, независимо друг от друга с помощью разработанного фрагментного шейдера. При выполнении данного шейдера вычисленные коэффициенты k_{shadow} смешиваются с цветами освещенной модели рельефа, и в результате формируется изображение модели рельефа с тенями.

4. РЕЗУЛЬТАТЫ

Предложенная технология и методы были реализованы в программном комплексе моделирования динамических теней рельефа, входящем в систему виртуального окружения VirSim [2], разработанную в ФГУ ФНЦ НИИСИ РАН. В качестве эксперимента, мы интегрировали наше решение в реализацию метода каскадных карт теней (CSM) от NVidia [10] и сравнили визуализацию теней. Для этого была использована карта высот Великого Каньона [29] размера 2K × 2K (см. рисунок 7). При методе CSM при некотором положении наблюдателя образовалась потеря теней на

заднем плане и внутри ущелья (см. рис. 7а). В нашей реализации все тени были отрисованы аккуратно и точно независимо от расположения наблюдателя и солнца (см. рис. 7б). Тестирование проводилось при разрешении Full HD на персональном компьютере (Intel Core i7-6800K 3.40GHz, 16Gb RAM, NVidia GeForce RTX 2080, 16x анизотропная фильтрация, 8x сглаживание). На рисунке 8 показаны графики изменения частоты визуализации затененной модели рельефа Великого Каньона (с помощью нашей технологии) для различных уровней детализации карты высот при увеличении высоты солнца.

5. ЗАКЛЮЧЕНИЕ

В данной работе рассмотрена задача построения динамических теней рельефа местности в системах виртуального окружения. Для решения этой задачи предложена технология реализации многоуровневого рэйкастинга теней, нововведением которой является попиксельное управление детализацией синтезируемых теней, а также проверка на пересечение солнечных лучей не только с наибольшими, но и с наименьшими высотами участков рельефа. Это дает возможность повысить скорость визуализации таких сложных форм рельефа, как ущелья и кратеры (особенно на малых высотах солнца), а также рационально расходовать вычислительный ресурс GPU при визуализации рельефа в областях вывода, имеющих размеры, меньшие, чем у исходной карты высот. На основе разработанных технологии, методов и алгоритмов был создан программный комплекс и выполнена его апробация на задаче визуализации карты высот Великого Каньона. Апробация подтвердила высокое качество синтезируемых теней и эффективность предложенных методов ускорения рэйкастинга. Полученные результаты могут быть использованы при создании систем виртуального окружения, в видеотренажерных комплексах, системах научной визуализации, виртуальных глобусах и геоприложениях и др.

БЛАГОДАРНОСТИ

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН “Проведение фундаментальных научных исследований (47 ГП)” по теме № FNEF-2022-0012 “Системы виртуального окружения: технологии, методы и алгоритмы математического моделирования и визуализации. 0580-2022-0012”.

СПИСОК ЛИТЕРАТУРЫ

1. *Алтунин А.А., Долгов П.П., Жамалетдинов Н.Р., Иродов Е.Ю., Коренной В.С.* Направления применения технологий виртуальной реальности при подготовке космонавтов к внекорабельной деятельности // Пилотируемые полеты в космос. 2021. № 1 (38). С. 72–88. https://doi.org/10.34131/MSF.21.1.72-88
2. *Михайлюк М.В., Мальцев А.В., Тимохин П.Ю., Страшнов Е.В., Крючков Б.И., Усов В.М.* Система виртуального окружения VirSim для имитационно-тренажерных комплексов подготовки космонавтов // Пилотируемые полеты в космос. 2020. № 4 (37). С. 72–95. https://doi.org/10.34131/MSF.20.4.72-95
3. *Piovano L., Basso V., Rocci L., Pasquinelli M., Bar C., Marello M., Vizzi C., Lucenteforte M., Brunello M., Racca F., Rabaioli M., Menduni E., Cencetti M.* Virtual Simulation of Hostile Environments for Space Industry: From Space Missions to Territory Monitoring // Virtual Reality – Human Computer Interaction, IntechOpen. 2012. P. 153–178. https://doi.org/10.5772/51121. https://www.intechopen.com/chapters/38748.
4. *Тимохин П.Ю., Михайлюк М.В.* Метод сжатия разрядности карт высот на основе критерия визуальной значимости // Труды НИИСИ РАН. 2017. Том 7. № 1. С. 30–35. https://www.niisi.ru/tr/2017_T7_N1.pdf.
5. *Timokhin P., Mikhaylyuk M.* Reliable GPU-based methods and algorithms of implementation dynamic relief shadows in virtual environment systems // Proceedings of the 31th International Conference on Computer Graphics and Vision (GraphiCon 2021). 2021. V. 3027. P. 83–94. https://doi.org/10.20948/graphicon-2021-3027-83-94.
6. *Фролов В.А., Волобой А.Г., Ершов С.В., Галактионов В.А.* Современное состояние методов расчёта глобальной освещённости в задачах реалистичной компьютерной графики // Труды ИСП РАН. 2021. Том 33. Вып. 2. С. 7–48. DOI: https://ispranproceedings.elpub.ru/jour/article/ view/1384/1222. https://doi.org/10.15514/ISPRAS-2021-33(2)-1
7. *Brawley Z., Tatarchuk N.* Parallax Occlusion Mapping: Self-Shadowing, Perspective-Correct Bump Mapping Using Reverse Height Map Tracing // ShaderX3: Advanced Rendering with DirectX and OpenGL. 1st. ed. Charles River Media. 2004. P. 135–154.
8. *Amanatides J., Woo A.* A Fast Voxel Traversal Algorithm for Ray Tracing // Proceedings of the 8th European Computer Graphics Conference and Exhibition (Eurographics ’87), Amsterdam. 1987. P. 3–10. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.3443.
9. *Fan Z., Sun H., Xu L., Lee K. L.* Parallel-Split Shadow Maps for Large-Scale Virtual Environments // Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and its Applications (VRCIA’06), Association for Computing Machinery. New York. 2006. P. 311–318. https://doi.org/10.1145/1128923.1128975.
10. *Dimitrov R.* Cascaded shadow maps // Developer Documentation. NVIDIA Corp. 2007. http://developer.download.nvidia.com/SDK/10/opengl/samples.html#cascaded_shadow_maps.
11. *Bunnell M., Pellacini F.* Shadow Map Antialiasing // GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics. 1st. ed. Addison-Wesley. 2004. P. 185–192. https://developer.down-

load.nvidia.com/books/HTML/gpugems/gpugems_ch11.html.

12. *Johnson G.S., Mark W.R., Burns C.A.* The Irregular Z-Buffer and its Application to Shadow Mapping // Department of Computer Sciences and Texas Advanced Computing Center of the University of Texas at Austin. 2004. https://www.cs.utexas.edu/ftp/techreports/tr04-09.pdf.
13. *Lefohn A.E., Sengupta S., Owens J.D.* Resolution-Matched Shadow Maps // ACM Transactions on Graphics. 2007. v. 26. № 4. article 20. https://doi.org/10.1145/1289603.1289611
14. *Rosen P.* Rectilinear texture warping for fast adaptive shadow mapping // Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D 2012). 2012. P. 151–158. https://doi.org/10.1145/2159616.2159641.
15. *Everitt C., Kilgard M.J.* Optimized Stencil Shadow Volumes // Game Developer Conference. 2003. https://www.nvidia.com/docs/IO/8230/GDC2003_-ShadowVolumes.pdf.
16. *Laine S.* Split-plane shadow volumes // Proceedings of Graphics Hardware. Eurographics Association. 2005. P. 23–32. https://doi.org/10.1145/1071866.1071870. https://users.aalto.fi/~laines9/publications/laine2005gh_paper.pdf.
17. *Stich M., Wächter C., Keller A.* Efficient and Robust Shadow Volumes Using Hierarchical Occlusion Culling and Geometry Shaders // GPU Gems 3. Addison-Wesley Professional. 2007. P. 239–256. https://developer.nvidia.com/gpugems/gpugems3/part-ii-light-and-shadows/chapter-11-efficient-and-robust-shadow-volumes-using.
18. *Sintorn E., Olsson O., Assarsson U.* An Efficient Alias-free Shadow Algorithm for Opaque and Transparent Objects using per-triangle Shadow Volumes // ACM Transactions on Graphics. 2011. v. 30. № 6. article 153. DOI. http://portal.acm.org/ft_gateway.cfm?id=2024187&type=pdf. https://doi.org/10.1145/2024156.2024187
19. *Fu Z., Zhang H., Wang R., Li Z., Yang P., Sheng B., Mao L.* Dynamic Shadow Rendering with Shadow Volume Optimization // Advances in Computer Graphics, Proceedings of the 37th Computer Graphics International Conference (CGI 2020). 2020. v. 12221. P. 96–106. https://doi.org/10.1007/978-3-030-61864-3_9.
20. *Sakalauskas T.* Hybrid Terrain Shadow Ray Casting // Proceedings of the 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'2008). Plzen. 2008. P. 175–182. http://wscg.zcu.cz/wscg2008/Papers_2008/short/!_WSCG2008_Short_final.zip.
21. *Sanzharov V.V., Gorbonosov A.I., Frolov V.A., Voloboy A.G.* Examination of the Nvidia RTX // Proceedings of the 29th International Conference on Computer Graphics and Vision (GraphiCon'2019), Bryansk, Russia (CEUR Workshop Proceedings). 2019. v. 2485. P. 7–12. https://doi.org/10.30987/graphicon-2019-2-7-12.
22. *Policarpo F., Oliveira M.M.* Relaxed Cone Stepping for Relief Mapping // GPU Gems 3. Addison-Wesley Professional. 2007. P. 409–428. https://developer.nvidia.com/gpugems/gpugems3/part-iii-rendering/chapter-18-relaxed-cone-stepping-relief-mapping.
23. *Tevs A., Ihrke I., Seidel H.-P.* Maximum Mipmaps for Fast, Accurate, and Scalable Dynamic Height Field Rendering // Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games (I3D'08). New York. 2008. P. 183–190. https://doi.org/10.1145/1342250.1342279.
24. *Aslandere T., Flatken M., Gerndt A.* A Real-Time Physically Based Algorithm for Hard Shadows on Dynamic Height-Fields // Proceedings of 12. Workshop der GI-Fachgruppe on Virtuelle und Erweiterte Realität, Aachen Verlag, Bonn. 2015. P. 101–112. https://elib.dlr.de/101497/.
25. *Jung D., Schrempp F., Son S.* Optimally Fast Soft Shadows on Curved Terrain with Dynamic Programming and Maximum Mipmaps. 2020. https://arxiv.org/pdf/2005.06671.pdf.
26. *Lengyel E.* Mathematics for 3D Game Programming and Computer Graphics. 3rd. ed. Course Technology. Boston. 2012.
27. *Ewins J.P., Waller M.D., White M., Lister P.F.* MIP-map Level Selection for Texture Mapping // IEEE Transactions on Visualization and Computer Graphics. 1998. V. 4, № 4. P. 317–329. https://doi.org/10.1109/2945.765326
28. OpenGL Extension. Texture Filter Anisotropic. NVIDIA. 2018. http://www.opengl.org/registry/specs/EXT/ texture_filter_anisotropic.txt.
29. Large Geometric Models Archive, Georgia Institute of Technology. https://www.cc.gatech.edu/projects/large_models/.