Ускорение алгоритма излучательности на графических процессорах

А.С. Щербаков[1], В.А. Фролов[1,2]
alex.shcherbakov@graphics.cs.msu.ru|vladimir.frolov@graphics.cs.msu.ru
[1]Москвоский государственный университет имени М.В. ломоносова;
[2]Институт прикладной математики имени М.В. Келдыша РАН

В данной работе предложен метод преобразования матрицы форм-факторов, позволяющий ускорить вычисление вторичного освещения методом излучательности. Рассмотрена адаптация этого метода для графических процессоров. В частности, предложено использование DXT-текстур для хранения матрицы форм-факторов и переупорядочивание столбцов и строк матрицы в целях уменьшения потерь при сжатии. Предложенные оптимизации позволяют повысить скорость работы алгоритма излучательности до 10 раз и уменьшить до 3 раз объем занимаемой памяти GPU.

Ключевые слова: *излучательность, глобальное освещение, GPU.*

Accelerating radiosity on GPUs

A.S. Shcherbakov[1], V.A. Frolov[1,2]
alex.shcherbakov@graphics.cs.msu.ru|vladimir.frolov@graphics.cs.msu.ru
[1]Lomonosov Moscow State University;
[2]Keldysh institute of applyed mathematics

We propose a novel approach to implement radiosity on GPU with specific optimizations via form-factor matrix transformations. The proposed transformations enable to reduce the amount of computations for multiple-bounce global illumination and apply DXT compression (with subsequent hardware decompression when reading formfactors on GPU). Our implementation is 10 times faster running and requires 3 times less memory than the naive radiosity GPU implementation.

Keywords: *radiosity, global illumination, GPU.*

Глобальное освещение 3D-сцен складывается из первичного освещения – полученного из источника света, и вторичного – многократно отраженного от поверхностей сцены. Наибольшую сложность представляет вычисление вторичного освещения, так как размерность интеграла освещенности увеличивается с каждым отражением. Поэтому в приложениях реального времени используют различные методы приближенного вычисления.

1. Обзор существующих методов

1.1. Instant Radiosity

[1] является одним из самых популярных методов благодаря своей простоте. Для расчёта вторичного освещения используются «вторичные» источники света, создаваемые при помощи трассировки лучей из первичных источников. Вторичное освещение, таким образом может рассматриваться как первичное от «вторичных» источников. Развитием метода Instant Radiosity для GPU является алгоритм Reflective Shadow Maps (RSM) [3]. Вместо трассировки лучей в RSM для создания вторичных источников используются карты теней (shadow maps). Основной недостаток данного метода – низкая точность.

1.2. Light Propagation Volumes

[2] создаёт вторичные источники света так же, как это делает Instance Radiosity, но расчёт вторичного освещения производится при помощи распространения света по трёхмерной сетке. Основной недостаток данного метода – высокий расход памяти и низкая эффективность метода при расчёте распространения света через пустые пространства.

1.3. Voxel Cone Tracing

[5] производит сбор освещения для каждого пикселя путём трассировки нескольких конусов из заданной точки на поверхности, имитируя монте-карло трассировку лучей по полусфере. VCT так же как и LPV использует воксельную сетку для представления упрощённой геометрии, а сама трассировка конусов аналогична шаганию по лучу (ray marching). Отличие в том, что с увеличением расстояния выборка производится из более грубых мип-уровней воксельной сетки, за счёт чего и получается геометрическая апроксимация конуса. Недостатком алгоритма является высокая вычислительная сложность и зависимость скорости от разрешения.

1.4. Spherical Harmonics

[8] основывается на разложении сложных функций освещенности в сумму более простых для вычисления величин. Для некоторых точек поверхности (как правило, вершин) вычислаются коэффециенты разложения их функций освещения по базису. Функции освещения из источников также раскладываются по базису. В итоге вычисление освещения в точке с разложенной в ней функцией освещенности сводится к скалярному произ-

ведению векторов состоящих из коэффициентов функции освещённости в данной точке и функции освещения из источника. Данный метод широко используется при визуализации открытых пространств, но уступает в точности на закрытых помещениях методу излучательности.

1.5. Излучательность

[6] позволяет получать качественные изображения для закрытых помещений с диффузными поверхностями, во многом не уступая более современным методам. Однако, время выполнения и требуемые ресурсы очень сильно зависят от сцены. На сценах содержащих сотни тысяч треугольников прямое применение излучательности затруднено из-за квадратичной сложности и затрат памяти в зависимости от количества примитивов. Поэтому на практике алгоритм излучательности выполняется для упрощённой сцены (содержащей меньшее количество площадок) и результат расчёта переносится на исходную сцену [10]. Следует подчеркнуть, что на ряду со сферическими гармониками, алгоритм излучательности переносит основную вычислительную сложность на этап предпросчёта, за счёт чего и достигается хороший баланс точность/скорость по сравнению с остальными методами.

2. Предложенный метод

В классическом алгоритме излучательности используется матрица форм-факторов F, размера $n \times n$, где n – количество площадок сцены. Для вычисления освещения после отражения данная матрица умножается на n-компонентный вектор $emission$, содержащий светимость площадок:

$$incident^{(1)} = F \cdot emission \tag{1}$$

Умножая полученный вектор на отражающую способность площадок ρ, на которые пришёл свет, вычисляют светимость площадок после отражения:

$$excident^{(1)} = incident^{(1)} \circ \rho \tag{2}$$

Элементами векторов в данных формулах являются 3-компонентные векторы, содержащие информацию по каждому цветовому каналу. Элементами матрицы форм-факторов являются вещественные числа от 0 до 1.

Приведённые вычисления можно повторять используя векторы $excident^{(i)}$ вместо изначальной светимости площадок, для получения света, пришедшего на поверхности сцены, после произвольного отражения:

$$incident^{(i)} = F \cdot excident^{(i-1)} \tag{3}$$

Полное освещение сцены после k отражений получается путём суммирования векторов $incident$:

$$indirect = \sum_{i=1}^{k} incident^{(i)} \tag{4}$$

2.1. Предпросчёт нескольких отражений

Предложенная модификация заключается в использовании преобразованной матрицы форм-факторов. Вначале введём «цветную» матрицу форм-факторов:

$$F^C_{ij} = F_{ij} \cdot \rho_j$$

Эта матрица содержит информацию о переносе света между площадками сцены по каждому каналу в отдельности. Таким образом, для её хранения требуется в 3 раза больше памяти. В данной работе рассматривается использование алгоритма излучательности для вторичного освещения сцены. Поэтому вычисление излучательности начинается не с получения вектора $incident^{(1)}$, а с вычисления светимости после первого отражения $excident^{(1)}$. Можно считать, что вектор $incident^{(1)}$ уже посчитан.

Мы можем преобразовать выражение (3) освещённости ветктора используя формулу (2) для произвольного индекса.

$$\begin{aligned} incident^{(i)} &= F \cdot \left(\rho \circ incident^{(i-1)}\right) = \\ &= F^C \cdot incident^{(i-1)} = \left(F^C\right)^{i-1} \cdot incident^{(1)} \end{aligned} \tag{5}$$

Преобразуем формулу (4) с использованием формулы (5):

$$\begin{aligned} indirect &= \sum_{i=1}^{k} \left(F^C\right)^{i-1} \cdot incident^{(1)} = \\ &= \left(\sum_{i=1}^{k} \left(F^C\right)^{i-1}\right) \cdot incident^{(1)} \end{aligned} \tag{6}$$

Матричный полином в скобках не зависит от первичного освещения сцены, а зависит только от геометрии сцены и материалов поверхности. Поэтому он может быть вычислен на этапе предрасчёта.

$$S = \left(\sum_{i=1}^{k} \left(F^C\right)^{i-1}\right)$$

Таким образом вычисление вторичного освещения методом излучательности после k отражений сводится к одному умножению ветокра на матрицу:

$$indirect = S \cdot incident^{(1)}$$

Использование матрицы такого вида позволяет ускорить вычисления в k раз, однако такая матрица требует в 3 раза больше памяти.

2.2. DXT-сжатие

Так как предложенная модификация алгоритма требует большего объёма памяти, был разработан метод хранения матрицы форм-факторов в сжатом виде. В качестве формата хранения был выбран формат DXT-текстур, поддерживаемый многими GPU (имеется ввиду аппаратная декомпрессия при чтении). Для этого

формата все значения в матрице необходимо привести к диапазону от 0 до 255. На рисунке 1 показано распределение значений в матрице форм-факторов. При масштабировании этих чисел в промежутке от 0 до 255 большая часть чисел станет 0. Поэтому большие числа, которые вносят основной вклад в вычисление излучательности сохраняются отдельно от остальной матрицы, а к остальным применяется следующее преобразование:

$$value'_i = \frac{\max\left(log\left(\frac{value_i}{\max\limits_j value_j}\right) + shift, 0\right) \cdot 255}{shift}$$

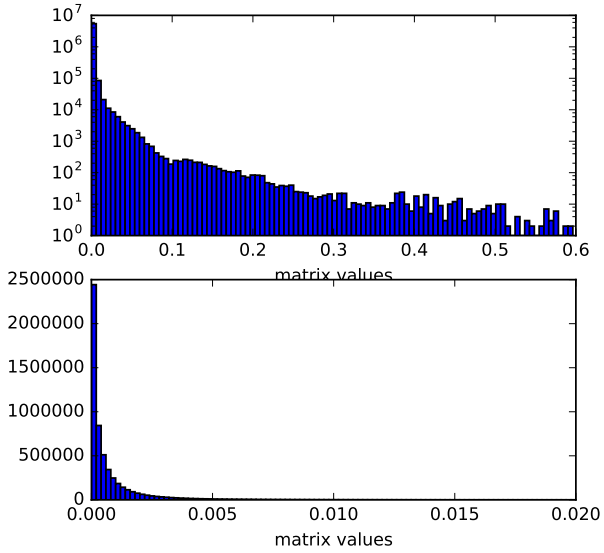

Рис. 1. Распределение значений в матрице форм-факторов (логарифмическая и линейная шкалы).

Текстура, которая получается в результате такого преобразования показана на рисунке 2 слева.

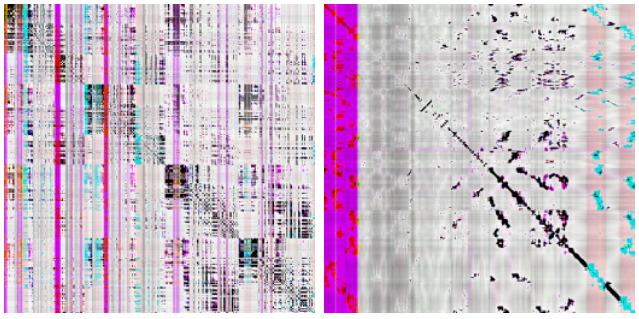

Рис. 2. Фрагмент матрицы форм-факторов до (слева) и после (справа) сортировки строк и столбцов.

Однако DXT-сжатие допускает потери. Чтобы минимизировать потери при сжатии строки и столбцы матрицы пересортируются так, чтобы уменьшить разницу между соседними значениями (рис. 2). Так как матрица является отношением для пар площадок сцены, то строки и столбцы должны меняться местами одновременно (рис. 3). В результате среднеквадратичная ошибка при сжатии уменьшается до 5 раз (рис. 4).

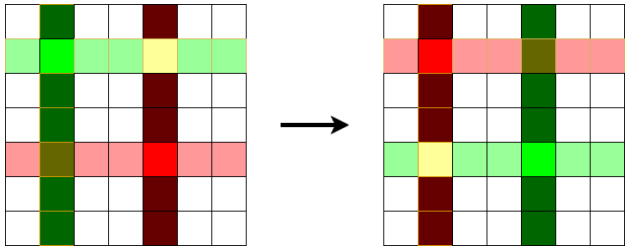

Рис. 3. Схема перестановки строк и столбцов.

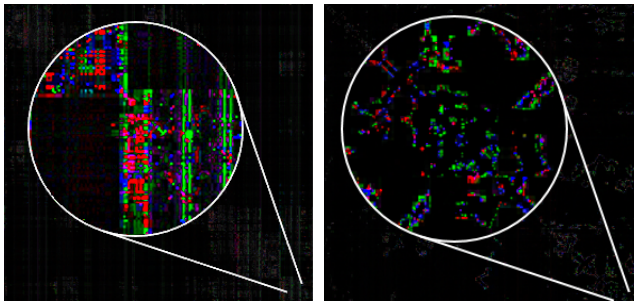

Рис. 4. Визуализированная разница между сжатой и не сжатой текстурами (увеличенный фрагмент) до переупорядочивания строк и столбцов (слева) и после (справа).

3. Детали реализации

Так как современные 3D-сцены содержат сотни тысяч треугольников и выполнение алгоритма излучательности является непреемлемым для таких порядков элементов сцены, для вычисления вторичного освещения была выбрана упрощённая версия той же самой сцены [11].

3.1. Предобработка сцены.

Предобработка сцены осуществляется по следующей схеме:

1. Строится упрощенный аналог сцены на основе вокселизации.
2. Вычисляются форм-факторы для площадок упрощенной сцены.
3. Вычисляется матрица форм-факторов учитывающая несколько отражений света.
4. Числа матрицы больше порогового значения сохраняются в отдельный файл, на их место ставятся 0.
5. Значения в матрице приводятся к диапазону от 0 до 255.
6. Осуществляется перестановка строк и столбцов.
7. Полученная матрица сохраняется в виде DXT-текстуры.

3.2. Визуализация сцены.

Визуализация происходит по следующему алгоритму:

1. Создаётся карта теней;
2. Вычисляется освещение площадок упрощённой сцены источником света.
3. Вычисляется вторичное освещение методом излучательности с помощью матрицы форм-факторов сохранённой в виде текстуры.
4. Вторичное освещение переносится с упрощённой сцены на исходную.

4. Сравнение результатов

Предложенные модификации алгоритма позволяют ускорить алгоритм излучательности в сумме до 10 раз (рис. 5, 8, 7). Использование матрицы форм-факторов учитывающей несколько отражений увеличивает размер файла с матрицей в 3 раза, однако, использование DXT-сжатия позволяет уменьшить требуемую память до 3 раз по сравнению с изначальной матрицей форм-факторов (и классическим алгоритмом излучательности, рис. 6). Мы провели сравнение с изображениями полученными методами Light Propagation Volumes из Unreal Engine 4, классической излучательности и трассировки путей (эталон). Предложенный метод показывает результат сравнимый по точности с классической излучательностью. При этом он ближе к эталону, чем изображение полученное методом LPV при одинаковой частоте кадров.

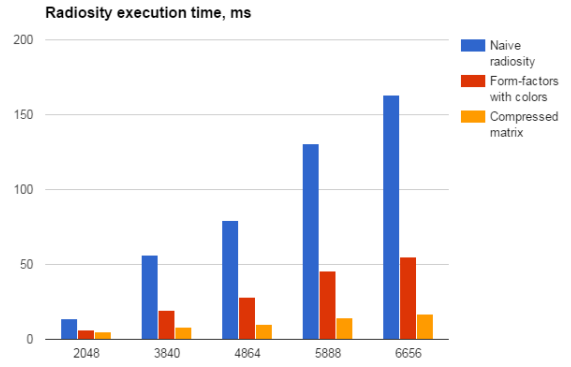

Рис. 5. Сравнение скорости вычисления излучательности. *Сжатие увеличивает скорость*, т.к. алгоритм ограничен памятью а не вычислениями.

5. Выводы и обсуждения

В отличие от других распространённых методов решения уравнения излучательности предложенный метод позволяет вычилить глобальное освещение за $n^2 + O(n)$ арифметических операций и чтений из памяти где n – число площадок, поскольку сводится к *единственному* умножению матрицы на вектор.

Аналогичного результата можно было бы добиться решая СЛАУ при помощи LU-разложения. Однако, эффективная реализация LU разложения на GPU

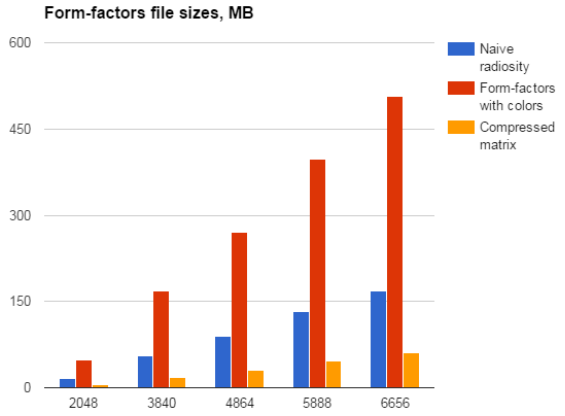

Рис. 6. Сравнение требуемой памяти.

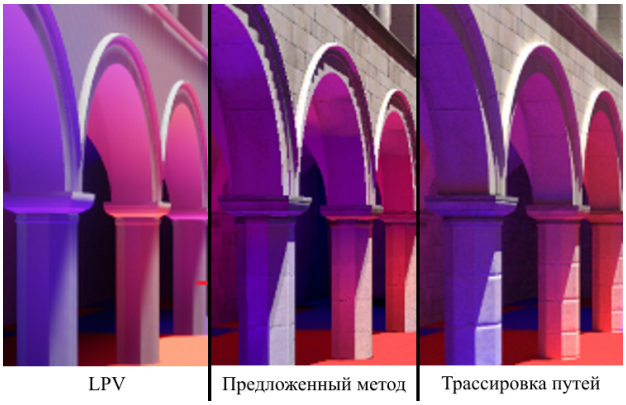

Рис. 7. Сравнение предложенного метода с LPV и трассировкой путей (эталон).

нетривиальна, а при использовании сторонних библиотек (например CUBLAS) отсутствует возможность использовать сжатие. Последнее, как уже было отмечено, критично для алгоритма излучательности (рис. 5, 6).

6. Литература

[1] Alexander Keller Instant radiosity // In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 49-56. DOI=http://dx.doi.org/10.1145/258734.258769

[2] Anton Kaplanyan and Carsten Dachsbacher Cascaded light propagation volumes for real-time indirect illumination // In Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games (I3D '10). ACM, New York, NY, USA, 99-107. DOI=http://dx.doi.org/10.1145/1730804.1730821

[3] Carsten Dachsbacher and Marc Stamminger Reflective shadow maps // In Proceedings of the 2005 symposium on Interactive 3D graphics and games (I3D '05). ACM, New York, NY, USA, 203-231. DOI=http://dx.doi.org/10.1145/1053427.1053460

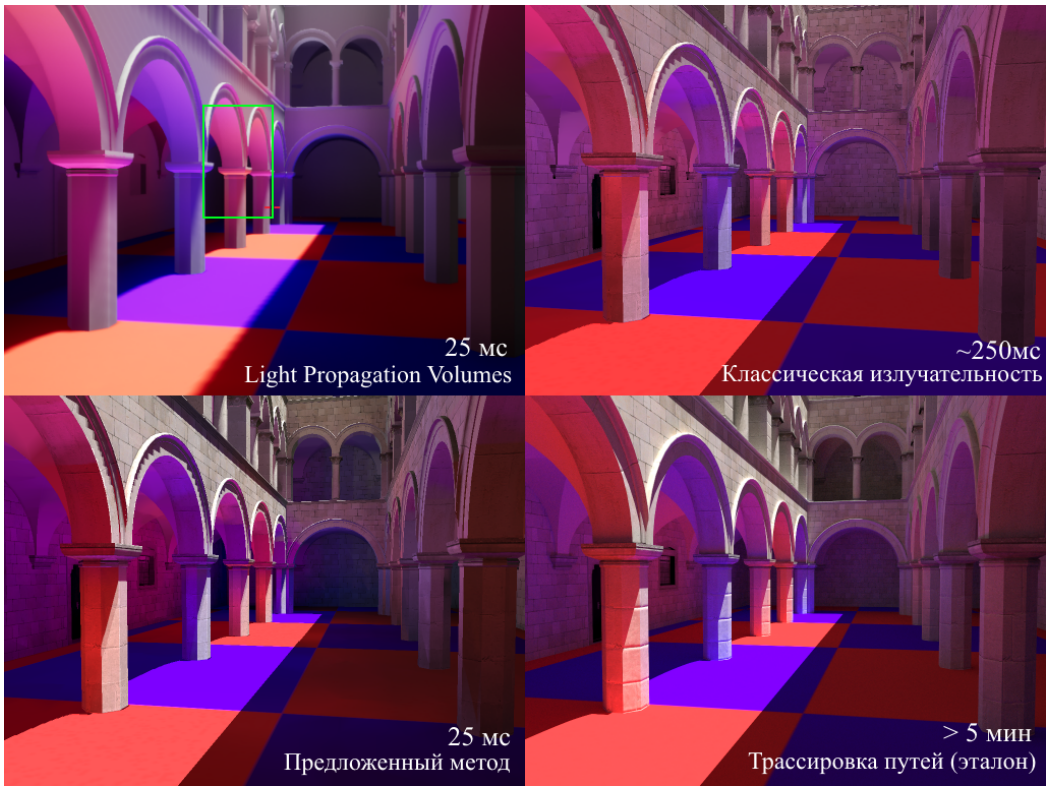

Рис. 8. Сравнение изображений, полученных различными методами и времени генерации изображений.

[4] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile Modeling the interaction of light between diffuse surfaces // In Proceedings of the 11th annual conference on Computer graphics and interactive techniques (SIGGRAPH '84), Hank Christiansen (Ed.). ACM, New York, NY, USA, 213-222. DOI=http://dx.doi.org/10.1145/800031.808601

[5] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann Interactive indirect illumination using voxel-based cone tracing: an insight // In ACM SIGGRAPH 2011 Talks (SIGGRAPH '11). ACM, New York, NY, USA, , Article 20 , 1 pages. DOI=http://dx.doi.org/10.1145/2037826.2037853

[6] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Radiosity on graphics hardware Modeling the interaction of light between diffuse surfaces. // In Proceedings of the 11th annual conference on Computer graphics and interactive techniques (SIGGRAPH '84), Hank Christiansen (Ed.).

[7] Greg Coombe, Mark J. Harris, and Anselmo Lastra Radiosity on graphics hardware // In Proceedings of Graphics Interface 2004 (GI '04). Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 161-168.

[8] Ian G. Lisle and S.-L. Tracy Huang Algorithms for spherical harmonic lighting // In Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia (GRAPHITE '07). ACM, New York, NY, USA, 235-238. DOI=http://dx.doi.org/10.1145/1321261.1321303

[9] Nathan A. Carr, Jesse D. Hall, and John C. Hart GPU algorithms for radiosity and subsurface scattering // In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware (HWWS '03). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 51-59.

[10] SamMartin, Per Einarsson A Real Time Radiosity Architecture for Video Games // Siggraph 2010. http://advances.realtimerendering.com/s2010/Martin-Einarsson-RadiosityArchitecture (SIGGRAPH%202010%20Advanced%20RealTime%20Rendering%20Course).pdf

[11] Щербаков А., Фролов В. Автоматическое упрощение геометрии для расчёта вторичной освещенности методом излучательности // в сборнике Сборник трудов Графикон 2016, ННГАСУ, с. 34-38.

7. Благодарности

Работа поддержана грантом РФФИ 16-31-60048 «мол_а_дк».