

Московский государственный университет имени М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Автоматизации Систем Вычислительных Комплексов

Щербаков Александр Станиславович

**Расчёт освещённости при помощи метода
излучательности на графических процессорах для
интерактивных приложений**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
к.ф-м.н. В. А. Фролов

Москва, 2017

Содержание

1 Введение	4
2 Постановка задачи	5
2.1 Неформальная постановка задачи	5
2.2 Формальная постановка задачи	5
3 Обзор методов расчета вторичного освещения	6
3.1 Instant Radiosity	6
3.1.1 Достоинства метода	6
3.1.2 Недостатки метода	6
3.2 Light Propagation Volumes	7
3.2.1 Достоинства метода	7
3.2.2 Недостатки метода	7
3.3 Voxel Cone Tracing	8
3.3.1 Достоинства метода	8
3.3.2 Недостатки метода	8
3.4 Spherical Harmonics	8
3.4.1 Достоинства метода	9
3.4.2 Недостатки метода	9
3.5 Radiosity	9
3.5.1 Достоинства метода	10
3.5.2 Недостатки метода	11
4 Предложенные оптимизации	12
4.1 Учёт нескольких отражений в матрице форм-факторов	12
4.2 DXT-сжатие матрицы форм-факторов	13
5 Описание практической части	21
5.1 Сравнение требуемого объёма памяти	24
5.2 Сравнение времени выполнения	25
5.3 Сравнение качества изображения	25

6 Заключение **28**

Список литературы **29**

Аннотация

В данной работе предлагается новый подход к реализации метода излучательности с использованием специальных преобразований матрицы форм-факторов. Данные модификации позволяют сократить объём вычислений для нескольких отражений света от поверхностей 3D-сцены за счёт вычислений специальной матрицы форм-факторов и её последующего сжатия методом DXT. Сжатие позволяет получить ускорение вычислений за счёт поддержки аппаратной декомпрессии данных указанного формата. Описанная реализация в 10 раз быстрее и требует в 3 раза меньше памяти, чем классический метод излучательности, реализованный на видеокарте.

1 Введение

Современные 3D-сцены составлены из сотен тысяч треугольников. Во многих приложениях использующих такие сцены возникает необходимость видеть сцену с различных ракурсов с динамическим изменением конфигурации освещения. Наибольшую сложность в таких случаях представляет вычисление вторичного освещения, то есть света, отражённого от поверхностей.

В настоящий момент разработано несколько методов вычисления глобального освещения сцены. Наиболее точным является алгоритм излучательности. В существующих решениях данный алгоритм выполняется на центральном процессоре и вычисленные значения освещения обновляются раз в 5-10 кадров. Выполнение данного алгоритма на видеокартах затруднено высокой вычислительной сложностью и требуемыми затратами памяти.

Предложенные оптимизации классического алгоритма излучательности позволяют увеличить скорость вычислений в 10 раз и уменьшить количество данных, с которыми ведётся работа, в 3 раза.

2 Постановка задачи

2.1 Неформальная постановка задачи

Требуется уменьшить вычислительные затраты при расчёте вторичного освещения в 3D-сцене с динамическими источниками света методом излучательности.

2.2 Формальная постановка задачи

Для заданной трёхмерной сцены (геометрия, источники света, материалы) первичным освещением называется освещение исходящее непосредственно из источников света. Вторичным освещением называется освещение, полученное путём многократного отражения первичного освещения от поверхностей сцены.

Необходимо, используя данные о геометрии и материалах сцены, провести её предобработку. Полученные данные используются во время отрисовки сцены в интерактивном режиме с изменением конфигурации освещения.

Цель работы — уменьшить время визуализации сцены и необходимые ресурсы видеопамяти.

3 Обзор методов расчета вторичного освещения

3.1 Instant Radiosity

Данный метод [8] является одним из самых популярных методов расчёта вторичного освещения. Для визуализации каждого кадра выполняются следующие шаги:

1. На источниках света выбираются точки.
2. Из выбранных точек трассируются случайные лучи. Лучи отражаются столько раз, сколько отражений света необходимо учесть при визуализации.
3. Каждая из точек полученных путей является вторичным источником света. Для них производится рендеринг с целью определить, какие области сцены освещаются источником.
4. В специальном буффере суммируется вклад вторичных источников света в освещение каждого пикселя.
5. Полученные значения освещения используются для финальной визуализации.

Развитием метода Instant Radiosity является алгоритм Reflective Shadow Maps [4]. Он отличается способом выбора вторичных источников света. Вместо трассировки происходит создание карты теней [10] для источников света. По карте теней выбираются вторичные источники как пиксели на карте. Плотность выбора источников на карте теней и их вес увеличивается с расстоянием до центра карты.

3.1.1 Достоинства метода

1. Этот метод значительно меньших вычислительных затрат по сравнению с остальными современными методами.
2. Не требуется предобработка сцены.

3.1.2 Недостатки метода

1. При недостаточном количестве вторичных источников света метод имеет низкую точность.

2. Требуемые вычисления существенно увеличивается с увеличением количества первичных и вторичных источников света.

3.2 Light Propagation Volumes

Этот алгоритм [7] создаёт вторичные источники света так же, как это делается в Instance Radiosity, но расчёт вторичного освещения производится иначе:

1. На первом шаге создаются регулярная трёхмерная сетка для переноса света и сетка с упрощенным представлением сцены. На последующих итерациях алгоритма сетка с геометрией перестраивается при необходимости.
2. Вторичные источники света используются для инициализации освещения на сетке.
3. Производится перенос света по сетке. В этом процессе учитывается вторая сетка с упрощённой сценой для моделирования отражений.
4. Освещение попавшее на поверхности используется для визуализации финального изображения.

3.2.1 Достоинства метода

1. Позволяет использовать различные эффекты связанные с полупрозрачностью среды.
2. Можно динамически изменять сцену.

3.2.2 Недостатки метода

1. Артефакты связанные с дискретизацией пространства.
2. Недостаточное количество итераций переноса даёт некачественное изображение.
3. Сложность вычислений растёт кубически с увеличением точности сетки.

3.3 Voxel Cone Tracing

В методе Voxel Cone Tracing [3] производится сбор освещения падающего на пиксель путём трассировки конусов из этого пикселя. Общую схему алгоритма можно представить следующим образом:

1. Вначале, создаётся воксельная сетка, в которой хранятся несколько mip-уровней сцены. Для каждого нового кадра можно использовать уже созданную сетку, изменяя её при изменении сцены.
2. Производится расчёт освещения для каждого пикселя. Для этого производится трассировка нескольких конусов из пикселя в разных направлениях. Эти конусы собирают падающую освещённость в направлениях трассировки.

При трассировке конусов происходит активное использование mip-уровней воксельной сетки. При удалении от пикселя данные берутся из mip-уровней с меньшей детализацией.

3.3.1 Достоинства метода

1. Высокое качество изображения.
2. Поддержка многих материалов поверхностей.
3. Динамическое изменение сцены.

3.3.2 Недостатки метода

1. Артефакты связанные с воксельным приближением сцены.
2. Высокие требования видеопамяти.
3. Большая сложность вычислений.

3.4 Spherical Harmonics

Техника сферических гармоник [9] основывается на разложении сложных функций освещенности в сумму более простых для вычисления величин. Изначально такое

разложение использовалось в физике для моделирования конфигурации электрона в атоме. Сложная сферическая функция раскладывается по ортонормированному базису. Это один из вариантов трёхмерного разложения функции в ряд Фурье.

Для некоторых точек считаются коэффициенты разложения их функций освещения по базису. При визуализации сферические функции освещения из источников также раскладываются по базису. В итоге вычисление освещения в точке с разложенной в ней функцией освещенности сводится к скалярному произведению векторов состоящих из коэффициентов функции освещённости в данной точке и функции освещения из источника.

3.4.1 Достоинства метода

1. Небольшое количество вычислений при визуализации.

3.4.2 Недостатки метода

1. Для сложных сцен и конфигураций освещения может потребоваться много коэффициентов в разложении функций. Это увеличивает количество необходимых вычислений и количество требуемой памяти.
2. Артефакты связанные с интерполяцией освещения между точками в которых известно разложение функции.

3.5 Radiosity

Метод излучательности (Radiosity) [2, 5, 6, 11] позволяет получать наиболее качественные изображения по сравнению с остальными методами. Однако, время выполнения и требуемые ресурсы очень сильно зависят от сцены. На сценах содержащих сотни тысяч или даже миллионы треугольников он непременим. Поэтому на практике алгоритм излучательности выполняется для упрощённой сцены (содержащей меньшее количество площадок) и результат расчёта переносится на исходную сцену [1].

Одна из наиболее коммерчески успешных реализаций этого алгоритма используется в графическом движке "Enlighten"[12]. В нём расчёты производятся на централь-

ном процессоре. Ввиду большого количества вычислений, которые выполняются на процессоре помимо вычисления вторичного освещения, нет возможности обновлять вторичное освещение для каждого кадра. Поэтому посчитанное вторичное освещение используется для следующих 5-10 кадров. В этом движке для сцены нужна её версия упрощённая вручную.

Алгоритм излучательности требует предобработки сцены:

1. Для сцены происходит её разбиение сцены на площадки (при создании упрощённого аналога вручную также может быть создано разбиение).
2. Считываются форм-факторы для каждой пары площадок.

Форм-фактор — это число показывающее для пары площадок какая часть энергии пришедшей на первую площадку отразится на вторую.

Для визуализации используются предпосчитанные значения форм-факторов и для каждого кадра выполняется расчёт вторичного освещение методом излучательности:

1. Вычисляется, либо задаётся значение для начальной светимости площадок. Из этих значений получается вектор начальной светимости.
2. Вектор начальной светимости умножается на матрицу форм-факторов. Таким образом получается вектор освещения пришедшего на площадки.
3. При умножении значений пришедшего освещения на цвета площадок получается вектор светимости после первого отражения.
4. 2 и 3 шаги повторяются на векторах отраженного освещения несколько раз. На каждом шаге накапливается освещение пришедшее на площадку после каждого отражения.
5. Накопленное вторичное освещение отображается на исходной сцене.

3.5.1 Достоинства метода

1. Наиболее близкое к точным методам изображение.
2. Высокая скорость вычислений для небольшого количества площадок.

3.5.2 Недостатки метода

1. Квадратичных рост сложности вычислений и затрат видеопамяти при увеличении количества площадок.
2. Возможны артефакты при некачественном упрощенном аналоге сцены.

4 Предложенные оптимизации

В данном разделе описаны разработанные оптимизации алгоритма излучательности, позволяющие значительно сократить время выполнения алгоритма и количество требуемой памяти для хранения предподсчитанной матрицы форм-факторов.

4.1 Учёт нескольких отражений в матрице форм-факторов

Первая оптимизация заключается в преобразовании матрицы форм-факторов таким образом, чтобы она учитывала освещение полученное в ходе нескольких отражений.

Пусть сцена содержит n площадок, для которых выполняется алгоритм излучательности. Введём необходимые обозначения:

F — матрица форм-факторов. Имеет размер $n \times n$.

$colors$ — вектор размера n , содержащий цвета площадок в формате RGB. Таким образом, каждый элемент вектора $colors$ — это трёхкомпонентный вектор, содержащий интенсивность цвета по каждому каналу.

$emission$ — вектор значений начальной светимости площадок. Имеет размер n . Элементы этого вектора — трёхкомпонентные векторы.

$excident^{(j)}$ — вектор, структура которого аналогична предыдущим. Его элементами является освещение, полученное путём отражения пришедшего света на j -ой итерации алгоритма.

$incident^{(j)}$ — вектор, той же структуры. Элементы вектора - освещение пришедшее на площадки после j -го отражения.

Обозначим $incident^{(0)} = emission$, чтобы можно было выписать формулы пересчёта освещения в более общем виде.

Таким образом, получается система уравнений:

$$incident^{(0)} = emission, \quad (1)$$

$$excident^{(j)} = F \cdot incident^{(j)}, \quad (2)$$

$$incident^{(j+1)} = colors \circ excident^{(j)}, \quad (3)$$

где \circ — почленное произведение (произведение Адамара).

Итоговое вторичное освещение после k отражений будет равно $\sum_{h=1}^k incident^{(h)}$. Значение $incident^{(0)}$ не включено в сумму, так как оно представляет собой первичное освещение.

Введём "цветную" матрицу форм-факторов F^C , которая будет показывать количество освещения отражённое площадками по каждому цветовому каналу:

$$F_{ij}^C = F_{ij} * colors_i \quad (4)$$

Используя уравнения (2), (3) и (4) получим формулу для пересчёта векторов $incident$ на разных итерациях:

$$incident^{(j+1)} = F^C \cdot incident^{(j)} = (F^C)^{(j+1)} \cdot incident^{(0)} \quad (5)$$

Теперь можно переписать итоговое вторичное освещение после k итерации:

$$\sum_{h=1}^k incident^{(h)} = \sum_{h=1}^k (F^C)^h \cdot incident^{(0)} \quad (6)$$

Как видно из правой части уравнения (6), вектор $incident^{(0)}$ можно вынести за скобки. Тогда, вычисление вторичного освещения сводится к вычислению полинома матрицы F^C и умножению его на вектор $incident^{(0)}$:

$$\sum_{h=1}^k (F^C)^h \cdot incident^{(0)} = \left(\sum_{h=1}^k (F^C)^h \right) \cdot incident^{(0)} \quad (7)$$

Выражение $F^{k\text{-reflections}} = \sum_{h=1}^k (F^C)^h$ не зависит от первичного освещения сцены и, следовательно, может быть вычислено на этапе предподсчёта.

Таким образом, первая оптимизация заключается в вычислении значения $F^{k\text{-reflections}}$ на стадии предобработки сцены. При этом количество необходимых операций для расчёта вторичного освещения становится в k раз меньше. Стоит также отметить, что размер матрицы форм-факторов увеличился в 3 раза, так как теперь она хранит значения для каждого цветового канала.

4.2 DXT-сжатие матрицы форм-факторов

Вторая предлагаемая оптимизация направлена на уменьшение количества памяти, необходимого для хранения матрицы форм-факторов, и уменьшения обращений

к памяти при выполнении алгоритма. Так как данные хранятся в глобальной памяти видеокарты - памяти с наибольшей латентностью, уменьшение размера данных приведёт к росту производительности.

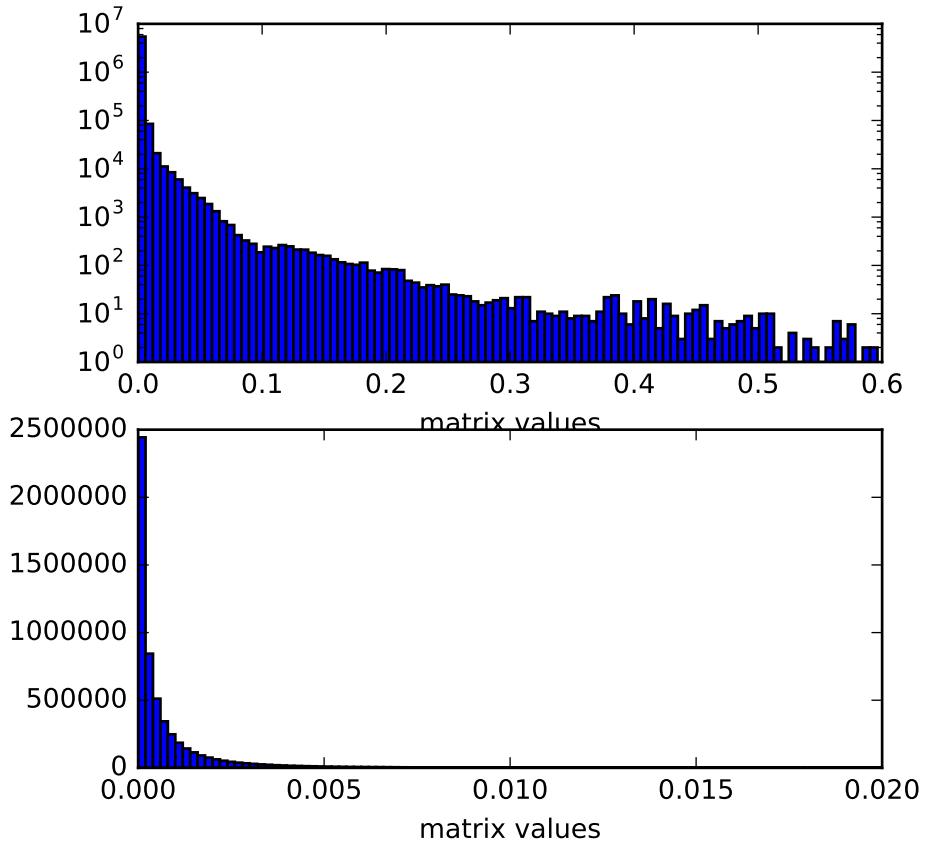


Рис. 1: Распределение значений в матрице форм-факторов на тестовой сцене (логарифмическая и линейная шкалы).

Из рисунка 1 видно, что большая часть значений не превосходит пороговой величины 0.005. Немногочисленные значения превышающие этот порог вносят больший вклад, поэтому их изменение, которое может произойти во время сжатия повлечёт за собой существенное ухудшение качества изображения. В связи с этим, значения матрицы форм-факторов делятся на две части. В первую входят значения которые относятся к 4% максимальных, во вторую - все остальные.

Значение 4% было выбрано в ходе экспериментов, как величина при которой ошибка сжатия не велика, но при этом размер файла минимален (рис. 2).

После того, как первая часть значений матрицы сохраняется отдельно, в матрице эти значения заполняются нулями. Так как оставшиеся значения вносят меньший

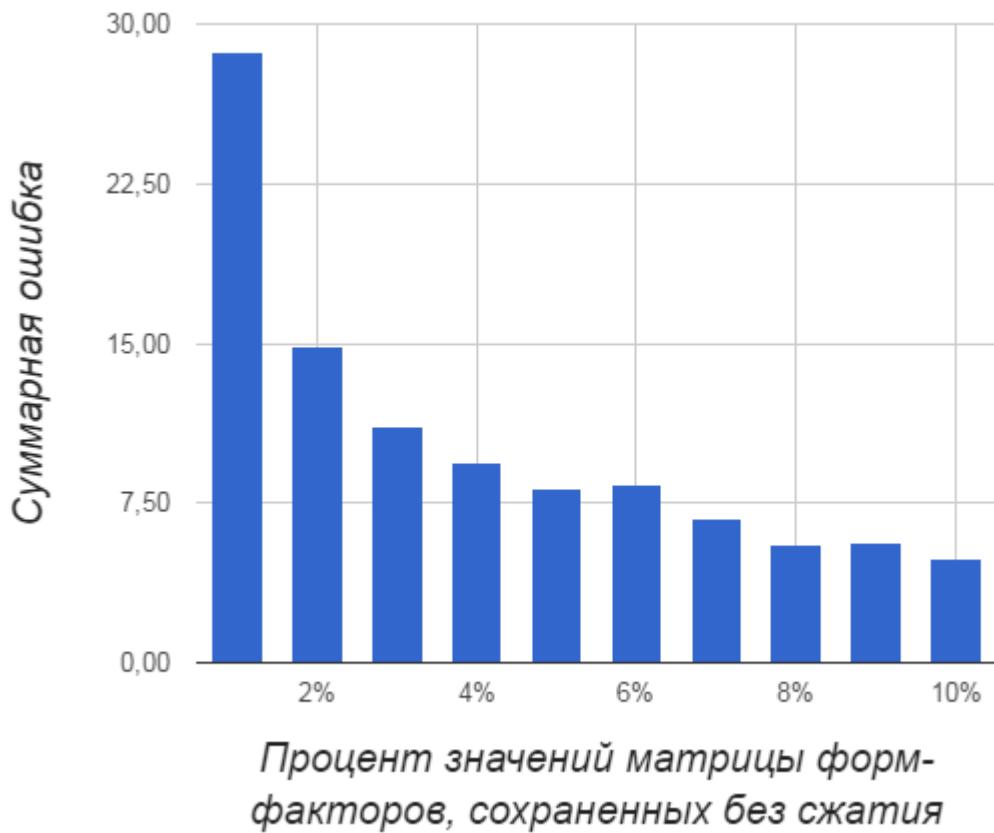


Рис. 2: Потери при сжатии при различном количестве несжатых чисел.

вклад в результат вторичного освещения, то их можно сжать с потерями, при этом качество изображения уменьшится незначительно.

Для того чтобы сохранить матрицу в текстуру нужно привести все значения к промежутку от 0 до 255, но так как оставшиеся значения отличаются на несколько порядков, то требуется провести ряд преобразований над элементами матрицы. Сначала значения матрицы приводятся к промежутку от 0 до 1 (проводится деление значений на максимальное число из оставшихся). Далее производится логарифмирование значений матрицы. Распределение значений после этой операции показано на рисунке 3.

Далее, к числам прибавляется константа 25, чтобы привести их на положительную полуось и происходит их масштабирование на промежуток от 0 до 255. Полученное распределение изображено на рисунке 4.

После этих преобразований матрица может быть сохранена в текстуру (Рис. 5).

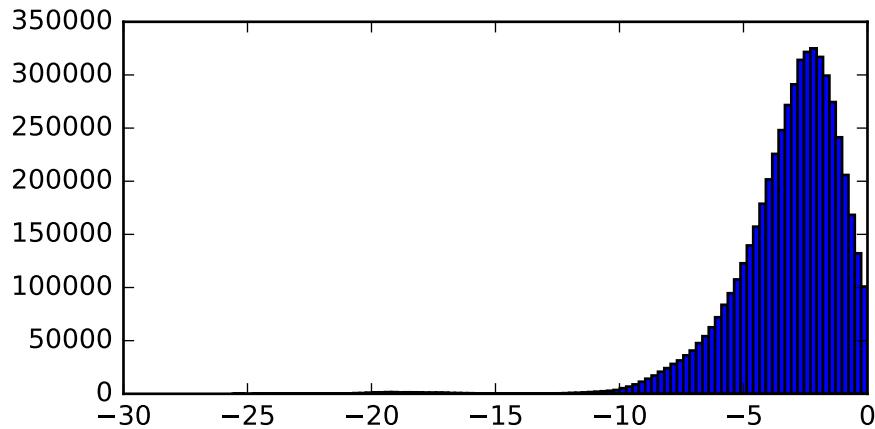


Рис. 3: Распределение значений матрицы форм-факторов после логарифмирования.

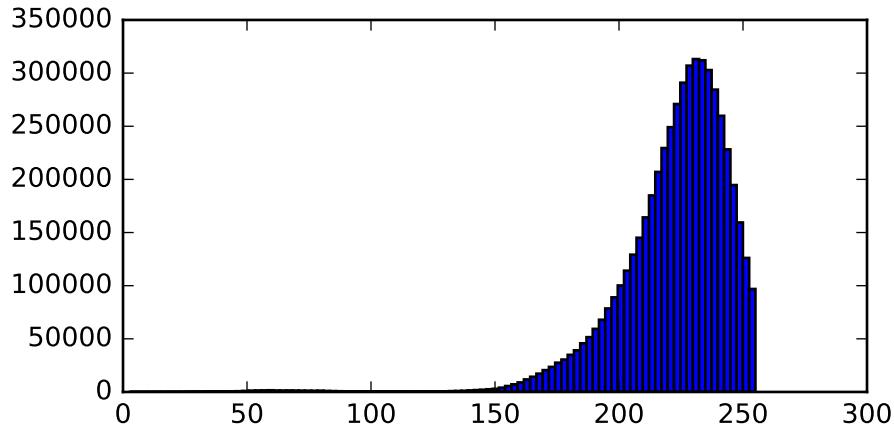


Рис. 4: Распределение значений матрицы форм-факторов после преобразований.

Сжатие DXT-1 происходит по следующей схеме:

1. Изображение разбивается на блоки по 4×4 пикселей;
2. В каждом блоке выбираются два крайних цвета, такие чтобы цвет пикселей в блоке был между этими двумя цветами;
3. Крайние цвета кодируются 16 битами. 5 бит для красного цвета, 6 для зелёного и 5 для синего;
4. Пиксели блока кодируются 2 битами каждый.

Из-за неоднородности распределения чисел в матрице форм-факторов при сжатии могут наблюдаться значительные потери. Чтобы их уменьшить проводится реор-

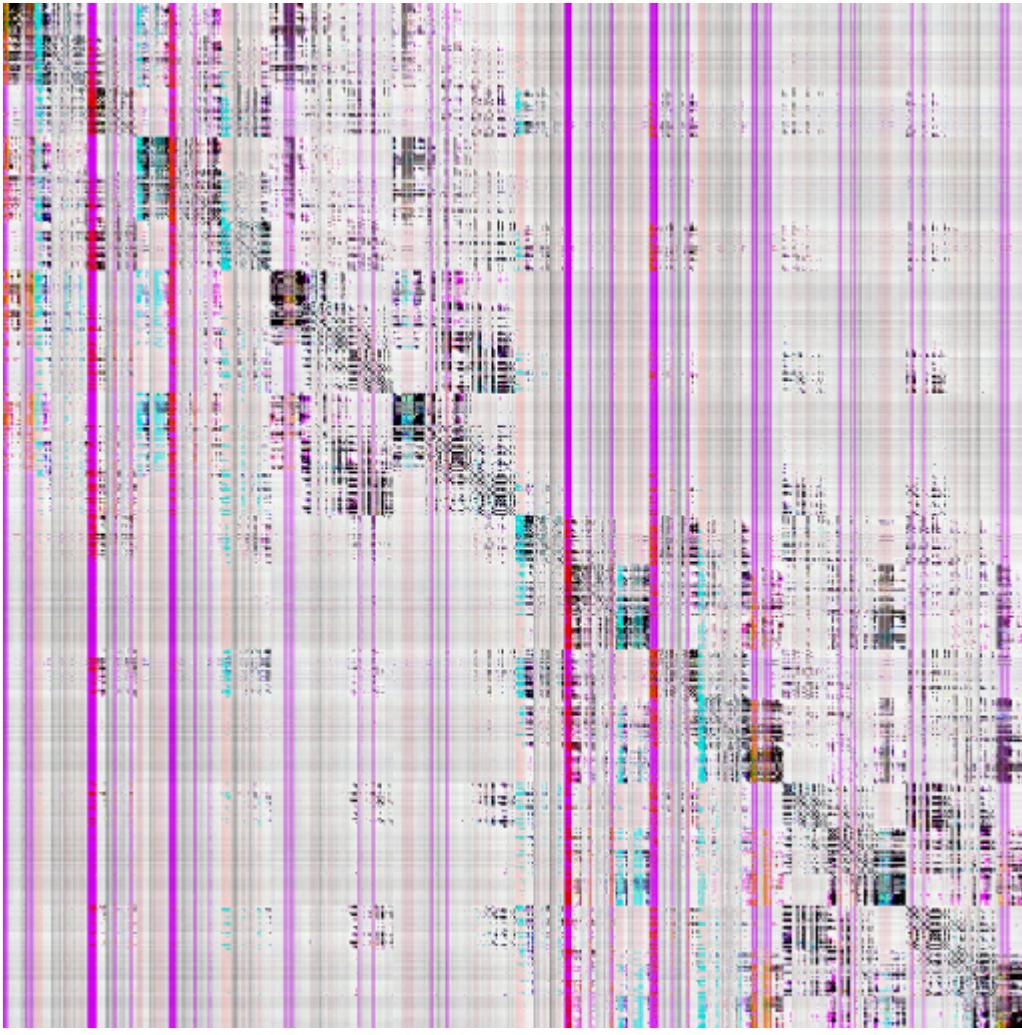


Рис. 5: Матрица форм-факторов.

ганизация матрицы. Строки и столбцы матрицы переставляются местами так, чтобы похожие значения были рядом. Чтобы сохранить свойства матрицы, позволяющие ей быть использованной для корректного расчёта освещения при перестановке i -ой и j -ой строк, необходимо также поменять местами i -ый и j -ый столбцы (рис. 6).

В качестве метрики похожести строк (аналогично столбцов) была выбрана сумма расстояний между координатами векторов. Введём полный граф. Его вершинами будут строки матрицы. Длины рёбер — метрика похожести строк. Таким образом, задача о нахождении оптимального расположения строк в матрице сводится к задаче комивояжёра. Эта задача из класса NP и к точному решению можно прийти только путём перебора. Поэтому был выбран эвристический жадный алгоритм реорганизации матрицы.

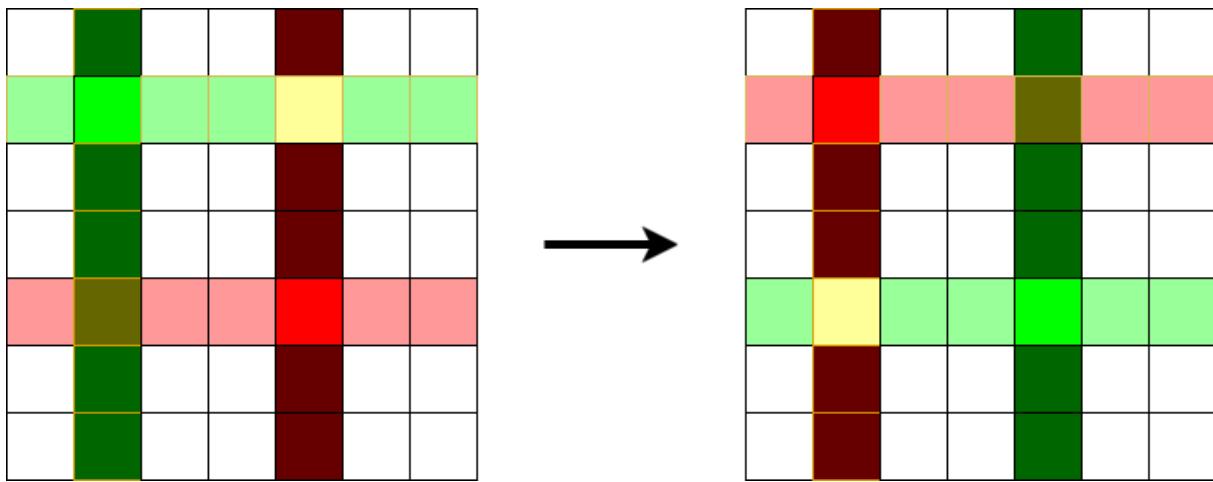


Рис. 6: Схема реорганизации матрицы форм-факторов.

Алгоритм осуществляется в два прохода. В первом метрика вычисляется для строк, во втором — для столбцов.

В одном проходе осуществляется итерирование по всем строкам (столбцам), в ходе которого:

- Вычисляется метрика похожести для текущей строки и всех строк, которые расположены ниже (правее, в случае столбцов).
- Наиболее близкая по метрике строка ставится после текущей. Алгоритм будет обрабатывать её следующей.

На рисунках 7 и 8 показаны матрицы форм-факторов после первого и второго проходов алгоритма реорганизации.

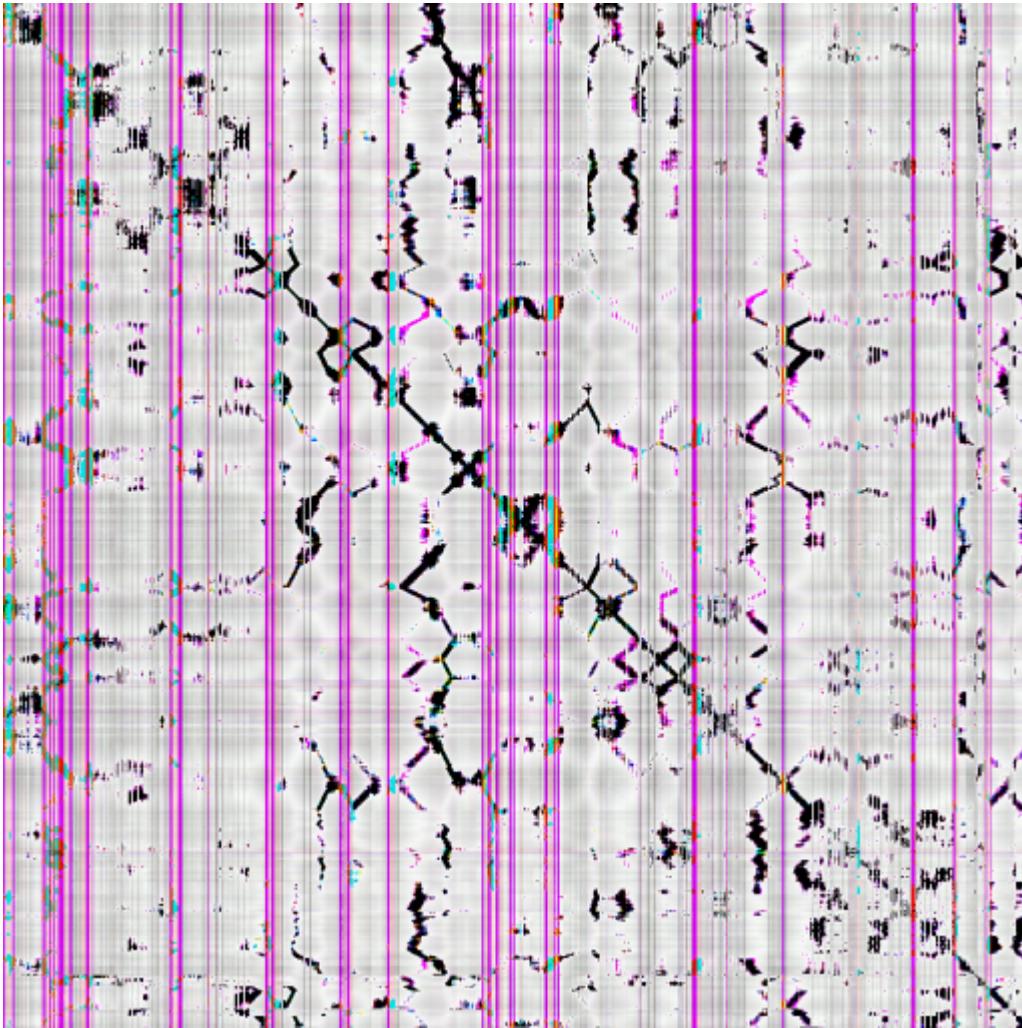


Рис. 7: Матрица форм-факторов после одного прохода реорганизации.

Потери при сжатии при реорганизации матрицы снижаются. Как видно из таблицы приведённой ниже их удалось уменьшить в 5 раз.

	Матрица без реорганизации	После одного прохода алгоритма	После двух проходов алгоритма
Средняя ошибка	1,28E-06	8,13E-07	5,88E-07
Макс. ошибка	0,4622	0,4684	0,1524
Сумм. ошибка	47,12	18,86	9,40

После изменения матрицы, она сохраняется как текстура с DXT-сжатием, данная текстура показана на рисунке 9. Визуально сложно найти отличия от матрицы без

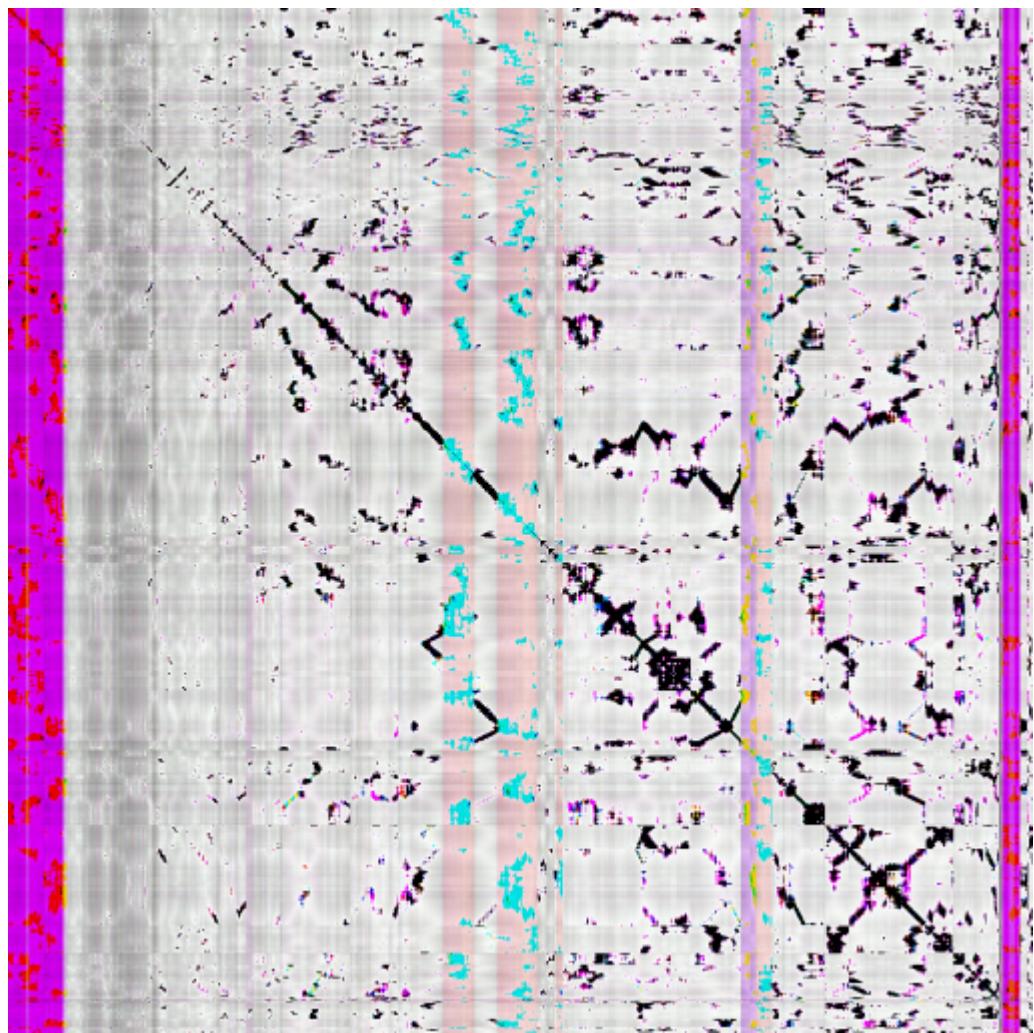


Рис. 8: Матрица форм-факторов после реорганизации.

сжатия, поэтому ниже также приведены разница между сжатой и несжатой матрицами по красному каналу изображения (рис. 10) и по всем каналам (рис. 11). Разница по зелёному и синему каналам имеет схожую структуру.

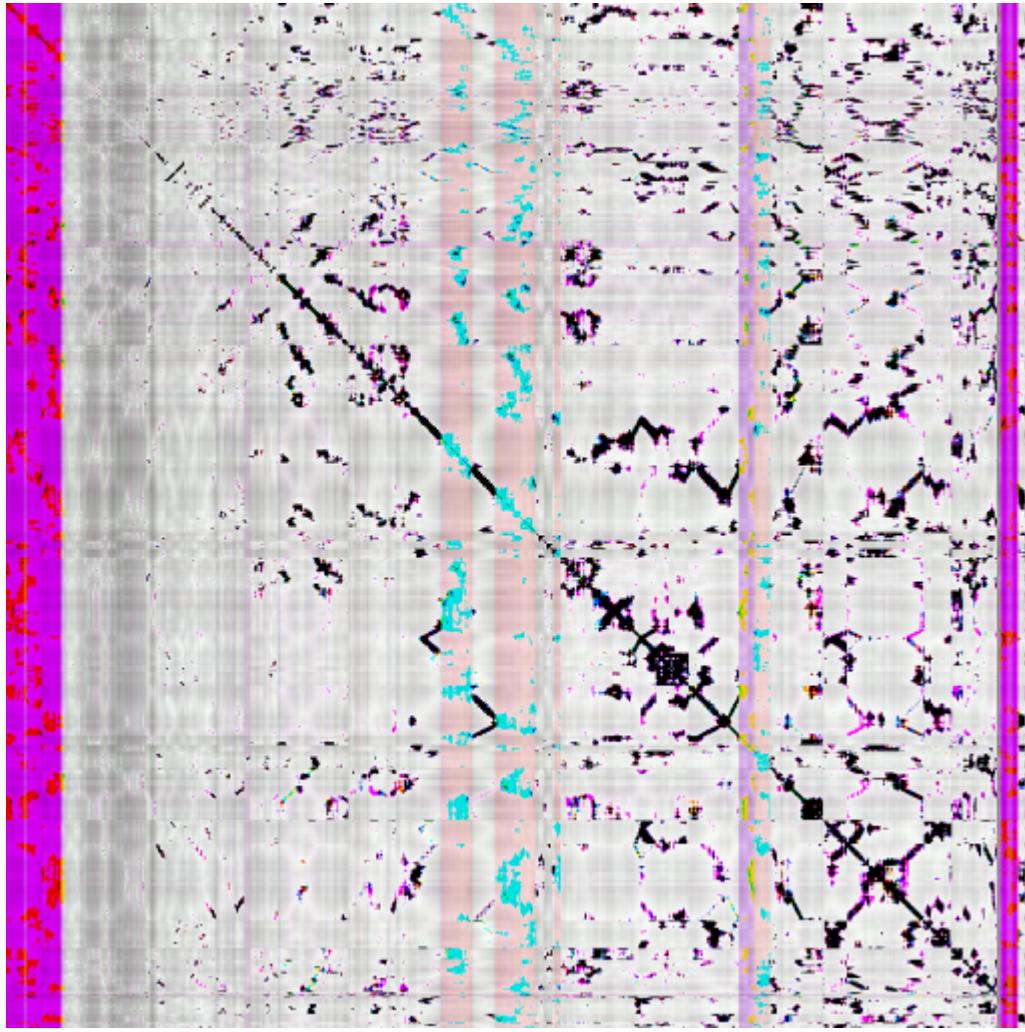


Рис. 9: Текстура, содержащая матрицу форм-факторов.

5 Описание практической части

Реализация алгоритма излучательности и предложенные оптимизации была выполнена на языке GLSL с использованием фреймворка OpenGL. Предподсчёт выполняется на видеокарте с использованием программ на OpenCL.

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.

Стадия предобработки сцены описывается следующими шагами:

1. Автоматическое упрощение исходной сцены с использованием вокселизации.

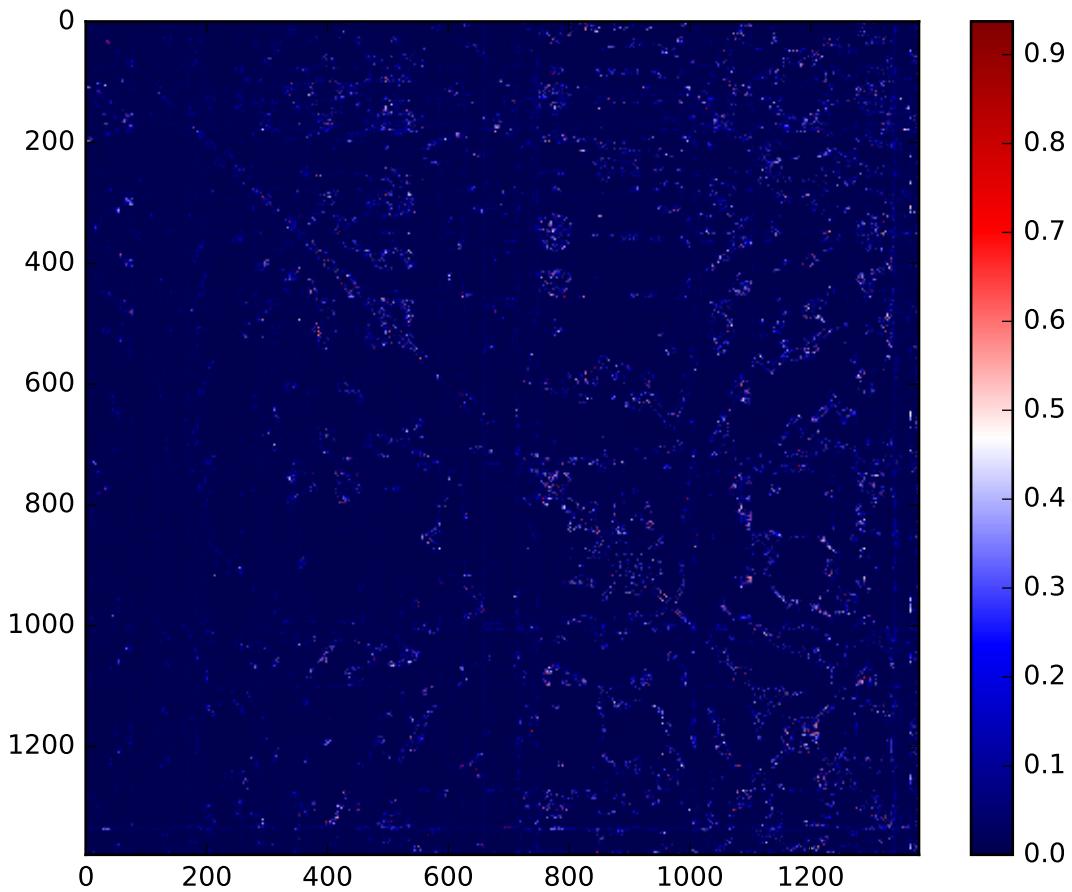


Рис. 10: Изменение красного канала матрицы форм-факторов, после сжатия.

2. Вычисление форм-факторов для площадок сцены.
3. Преведение матрицы форм-факторов к виду, в котором содержится информация о переносе освещения по каждой компоненте.
4. Вычисление полинома матрицы форм-факторов, учитывающего несколько отражений.
5. Перестановка строк и столбцов матрицы
6. Сохранение матрицы форм-факторов с использованием DXT-сжатия.

После генерации этих данных, они используются для отрисовки сцены в интерактивном режиме с динамическим освещением.

1. Создаются карты теней для источников света.

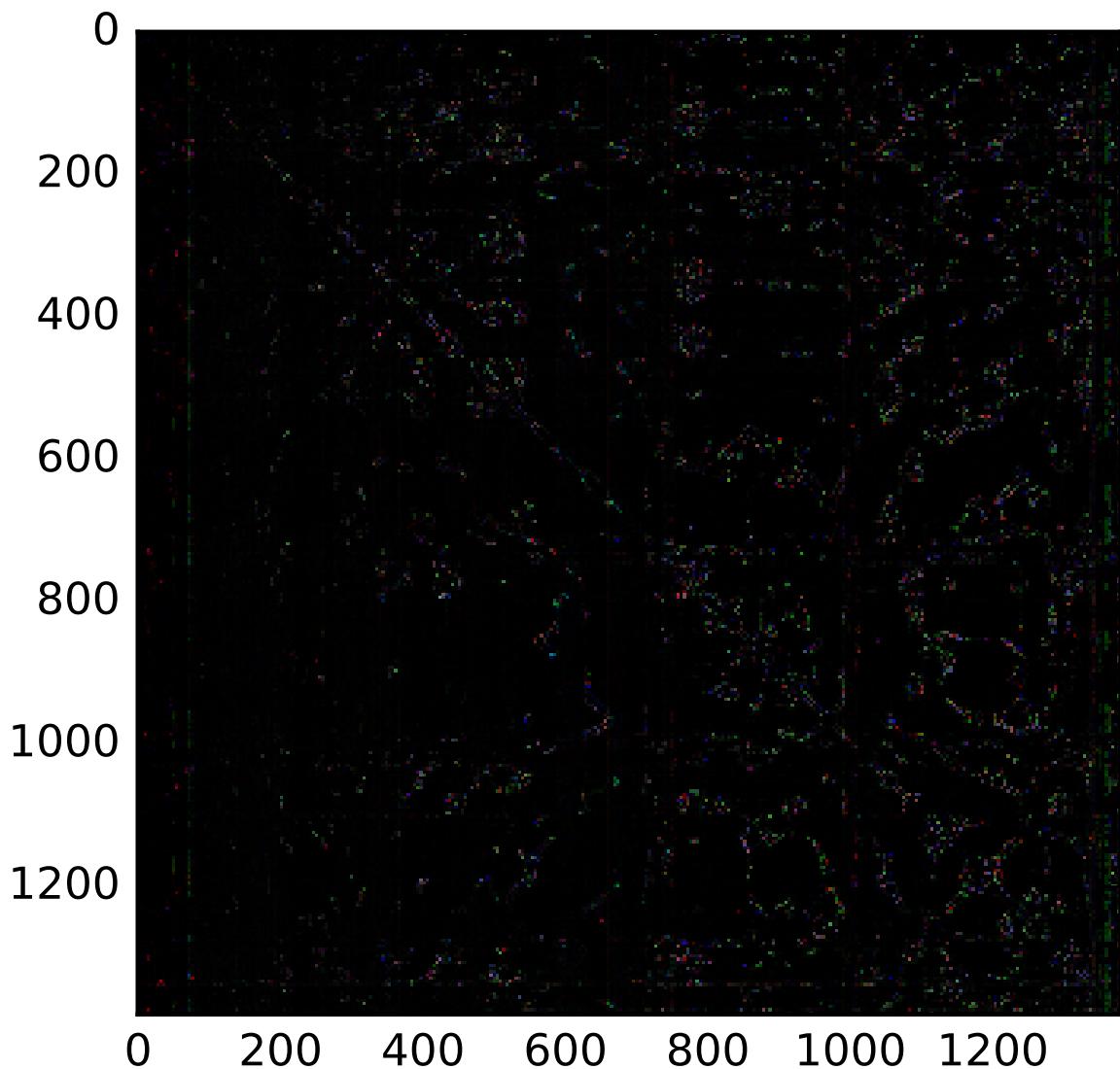


Рис. 11: Изменение матрицы форм-факторов, после сжатия.

2. Для каждой площадки упрощённой сцены вычисляется её освещенность при помощи карт теней.
3. Вектор начальной освещённости умножается на матрицу форм-факторов, хранящуюся в сжатой текстуре.
4. Учитываются значения матрицы форм-факторов, которые хранятся отдельно для увеличения качества.

5. Посчитанное вторичное освещение для упрощенной сцены переносится на исходную сцену.
6. На экран выводится финальное изображение с учётом первичного и вторичного освещения.

В ходе работы были проведены сравнения с классической реализацией алгоритма излучательности, методом трассировки путей и реализацией алгоритма Light Propagation Volumes из движка Unreal Engine 4. Сравнение проводилось на архитектурной сцене, содержащей 66450 треугольников. Материалы всех поверхностей сцены диффузные. Использовался направленный динамичный источник света.

5.1 Сравнение требуемого объёма памяти

На рисунке 12 показано, что применение первой описанной оптимизации увеличивает количество данных в 3 раза, но применение DXT-сжатия позволяет уменьшить размер файлов в 3 раза по сравнению с исходными.

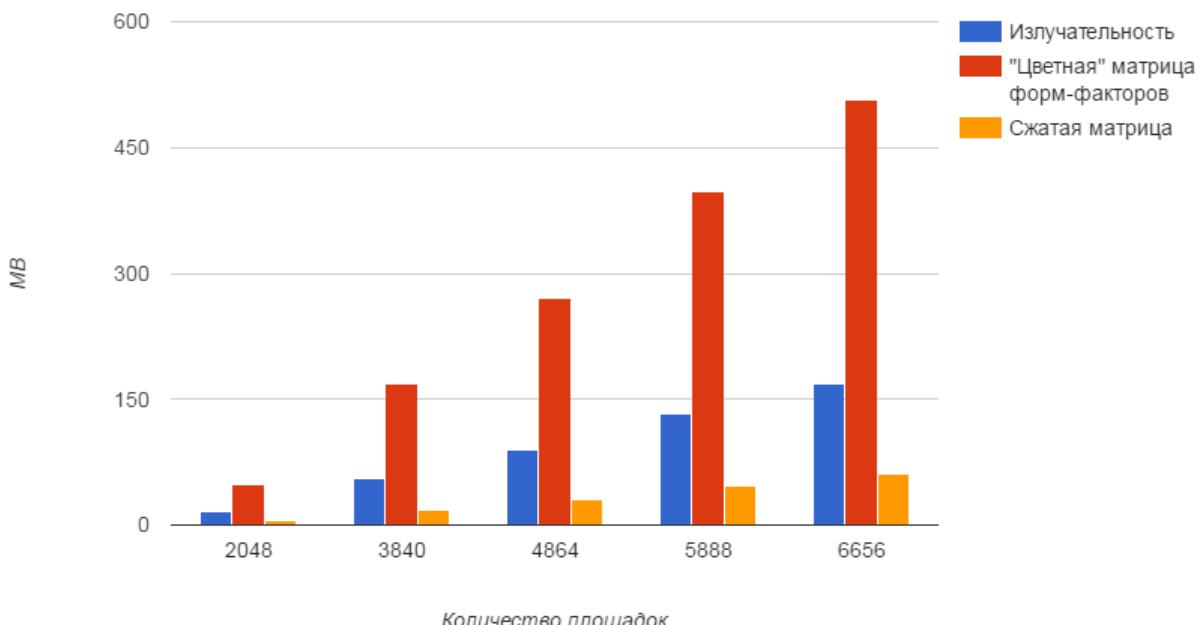


Рис. 12: Сравнение размера файлов форм-факторов.

5.2 Сравнение времени выполнения

Применение каждой из предложенных оптимизаций даёт существенный прирост в скорости выполнения. Первая оптимизация для трёх отражений ускоряет алгоритм в 3 раза. Вторая — в 10 раз быстрее чем классическая реализация алгоритма излучательности (рис. 13).

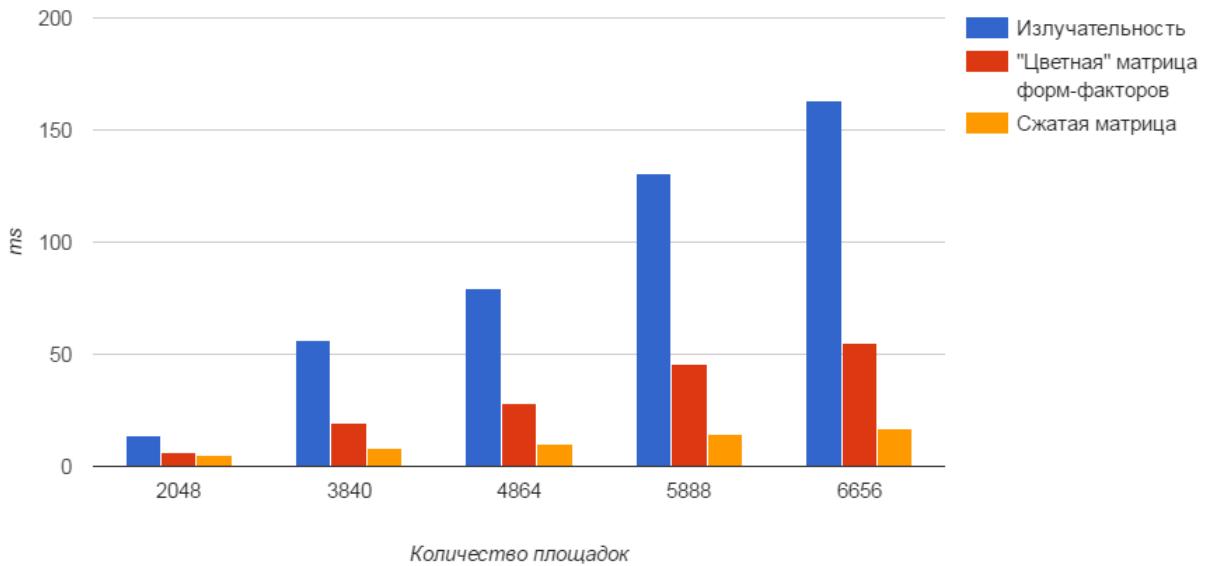


Рис. 13: Сравнение времени выполнения алгоритма.

5.3 Сравнение качества изображения

На рисунке 14 представлено сравнение с методами Light Propagation Volumes, трассировки путей и классической излучательности. Метод трассировки путей, как физически корректный был выбран в качестве эталона. Видно, что изображения полученные методом излучательности, в том числе и с оптимизациями, ближе к эталонному, чем изображение сгенерированное Light Propagation Volumes.

Размер сетки для алгоритма LPV был выбран такой, чтобы частота генерации изображения совпадала с частотой работы предложенного модифицированного алгоритма излучательности. Таким образом, оба изображения получены при частоте 40 FPS.

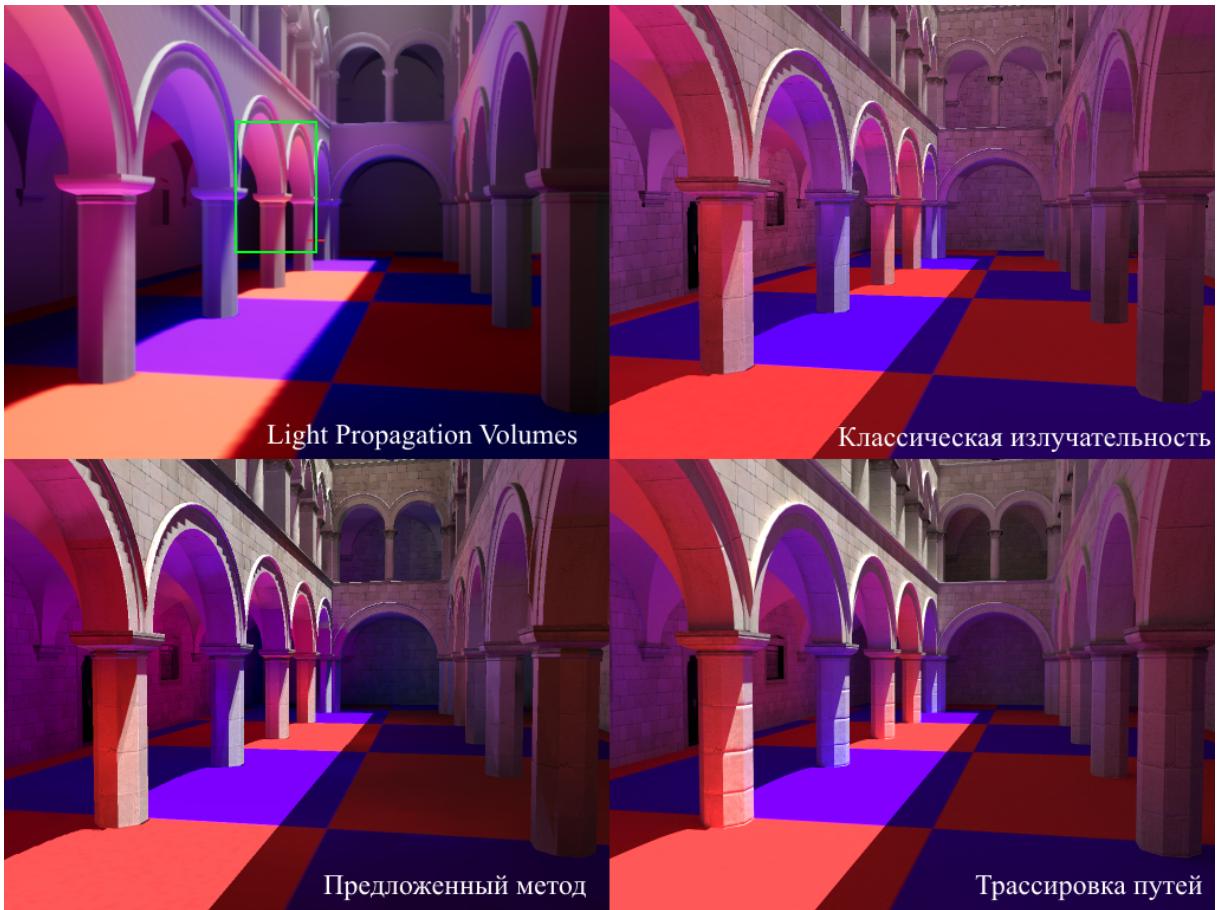


Рис. 14: Сравнение полученных изображений.

Части сцены, на которых заметно, что метод излучательности показывает лучший результат, чем Light Propagation Volumes представлены на рисунках 15 и 16.

Предложенные оптимизации алгоритма излучательности позволяют получать изображение, аналогичное полученному методу трассировки путей, за более короткое время. Методу трассировки путей потребовалось более 5 минут, для создания изображения. Реализации предложенного метода - 17 милисекунд.

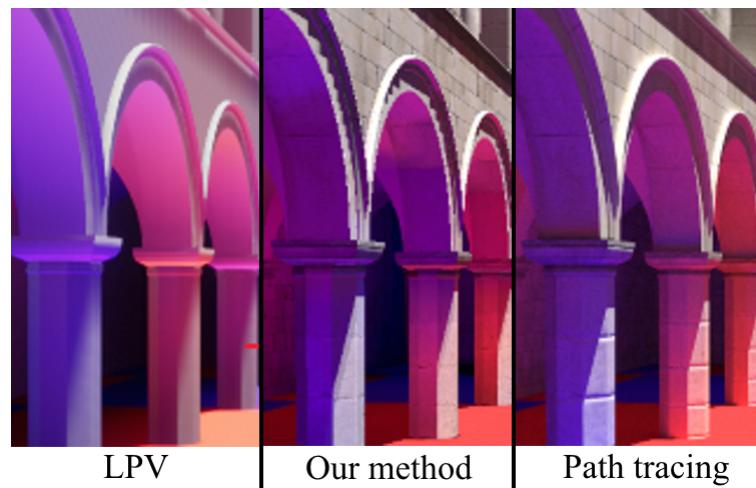


Рис. 15: Сравнение полученных изображений.

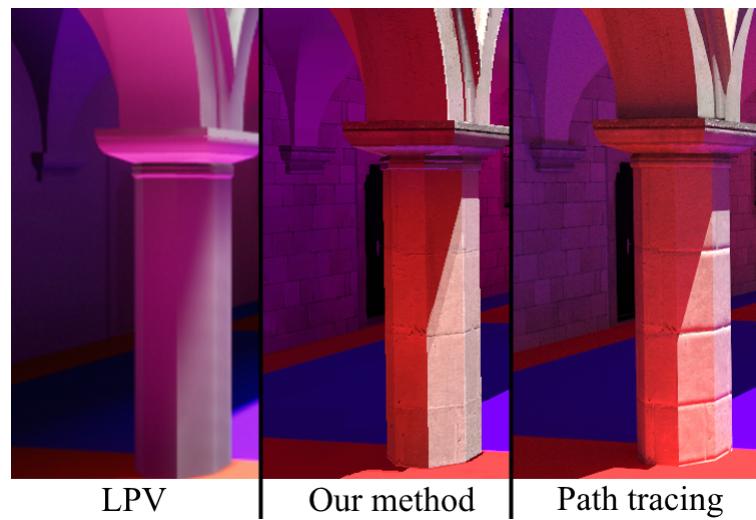


Рис. 16: Сравнение полученных изображений.

Предложенные оптимизации могут быть использованы с другими распространёнными оптимизациями алгоритма излучательности, например иерархической излучательностью. При этом качество изображения может быть улучшено за счёт увеличения количества площадок, для которых выполняется алгоритм, а следовательно и уменьшением ошибки приближения сцены воксельной моделью.

6 Заключение

В ходе работы были разработаны и реализованы оптимизации для оригинального алгоритма излучательности.

Первая из них позволяет ускорить алгоритм излучательности учитываящий k отражений света в k раз. Для случая 3 отражений приведены данные измерений на рисунке 13. Данная оптимизация увеличивает требуемый объём памяти для выполнения алгоритма в 3 раза.

Вторая модернизация классического алгоритма излучательности позволила уменьшить требуемый объем данных в 3 раза по сравнению с исходным. При этом время вычислений уменьшилось в 10 раз относительно первоначального.

Предложенные оптимизации позволяют вычислять вторичное освещение на 3D-сценах в реальном времени с высокой точностью.

Список литературы

- [1] Щербаков А. Фролов В. Автоматическое упрощение геометрии для расчёта вторичной освещенности методом излучательности // Сборник трудов Графикон 2016. — 2016. — С. 34–38.
- [2] Cindy M. Goral Kenneth E. Torrance Donald P. Greenberg, Battaille Bennett. Modeling the interaction of light between diffuse surfaces. — ACM, New York, NY, USA, 1984. — Pp. 213–222.
- [3] Cyril Crassin Fabrice Neyret Miguel Sainz, Green Simon. Interactive indirect illumination using voxelbased cone tracing: an insight // ACM SIGGRAPH 2011 Talks (SIGGRAPH '11). — 2011. — no. 20.
- [4] Dachsbaecher Carsten, Stamminger Marc. Reflective shadow maps. — ACM, New York, NY, USA, 2005. — Pp. 203–231.
- [5] Greg Coombe Mark J. Harris, Lastra Anselmo. Radiosity on graphics hardware. — Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. — Pp. 161–168.
- [6] Jaroslav Krivanek Pascal Gautron Sumanta Pattanaik, Bouatouch Kadi. Radiance caching for efficient global illumination computation // In ACM SIGGRAPH 2008 classes (SIGGRAPH '08). — 2008. — no. 75.
- [7] Kaplanyan Anton, Dachsbaecher Carsten. Cascaded light propagation volumes for real-time indirect illumination. — ACM, New York, NY, USA, 2010. — Pp. 99–107.
- [8] Keller Alexander. Instant radiosity. — ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1997. — Pp. 49–56.
- [9] Lisle Ian G., Huang S.-L. Tracy. Algorithms for spherical harmonic lighting. — ACM, New York, NY, USA, 2007. — Pp. 235–238.
- [10] Michael Wimmer Daniel Scherzer, Purgathofer Werner. Light space perspective shadow maps. — Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2004. — Pp. 143–151.

- [11] *Nathan A. Carr Jesse D. Hall, Hart John C.* GPU algorithms for radiosity and subsurface scattering. — Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2003. — Pp. 51–59.
- [12] *Sam Martin Per Einarsson.* A Real Time Radiosity Architecture for Video Games. — 2010. [advances.realtimerendering.com/s2010/Martin-Einarsson-RadiosityArchitecture\(SIGGRAPH%202010%20Advanced%20RealTime%20Rendering%20Course\).pdf](http://advances.realtimerendering.com/s2010/Martin-Einarsson-RadiosityArchitecture(SIGGRAPH%202010%20Advanced%20RealTime%20Rendering%20Course).pdf).