МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

На правах рукописи

Щербаков Александр Станиславович

Разработка и исследование методов вычисления глобального освещения на графических процессорах

Специальность 2.3.5 Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
кандидат физико-математических наук
Фролов Владимир Александрович

Москва — 2025

Оглавление

Стр.

Стр.

Введение

Одной из основных задач компьютерной графики является моделирование распространения света по сцене, то есть вычисление глобального освещения. Глобальное освещение разделяется на первичное и вторичное. Первичное освещение – это свет, падающий непосредственно из источника света на поверхность. Данный феномен достаточно хорошо изучен и в большинстве случаев может быть смоделирован несколькими проверками на видимость (за исключением источников света большой площади со сложной формой, для которых требуются специализированные подходы или Монте-Карло интегрирование с большим числом выборок). Вторичное освещение – это первичное освещение, многократно переотраженное поверхностями сцены. Вычисление этого феномена является гораздо более сложным, так как это вычисление многомерного интеграла, который необходимо оценивать численно для каждого пикселя изображения.

Выделим основные виды приложений, в которых существенным является быстрое и точное вычисление глобального освещения:

1. Приложения для архитектурного проектирования [1—3]. Данные приложения требуют высокой точности вычисленного освещения, так как при моделировании зданий конкретные условия освещения могут быть существенны с точки зрения эстетики и эксплуатации помещений.
2. Симуляторы и тренажеры [4; 5]. Корректно вычисленное глобальное освещение позволяет сделать окружение в симуляторах и тренажёрах максимально близким к реальности, что важно для схожести тренировочной среды с реальным миром. Так как в данных приложениях высок уровень интерактивности и пользователь может производить широкий ряд различных действий, достаточно высоки требования к производительности используемых алгоритмов.
3. Приложения, используемые при создании эффектов в кино и анимации [6; 7]. При финализации кинокартин вычисление освещения производится высокоточными методами. Однако, в процессе настройки и прототипирования визуальных эффектов важны быстрые итерации, поэтому скорость и точность освещения являются существенными в данных приложениях для экономии временных ресурсов.

4. Приложения виртуальной реальности [8; 9]. Требования к производительности для виртуальной реальности более жесткие, так как отрисовка производится на два экрана и требуемая частота кадров выше, чем для приложений на других платформах. Примеры аппаратной конфигурации: два экрана разрешением 3664×1920 с частотой обновления 90 Гц для Oculus Quest 2 256 Gb, два экрана разрешением 2000×2040 с частотой обновления 120 Гц для Sony Playstation VR2 [10].
5. Приложения дополненной реальности [11; 12], в которых корректное вычисление глобального освещения необходимо для получения естественного внешнего вида встраиваемых в реальное окружение объектов.
6. Компьютерные игры [13—15]. В данных приложениях требования к реализму могут быть ниже, чем в остальных примерах и являются специфичными для проекта. Но, в то же время, производительность алгоритмов должна быть выше, так как, помимо отрисовки геометрии и вычисления освещения, вычислительные мощности аппаратуры расходуются на решение множества других задач компьютерной графики: вычисление коллизий, процедурная генерация окружения, вычисление экранных эффектов, моделирование поведения частиц, воспроизведение анимаций.
7. Приложения светотехники [16—18]. Этот тип приложений требует высокой точности вычисления освещения для интерактивного редактирования освещения помещений, зданий и т.д.

Во всех упомянутых приложениях важна точность глобального освещения и скорость его вычисления. Алгоритмы, решающие задачу глобального освещения, разрабатываются на протяжении всего времени существования компьютерной графики. Их разработка не теряет актуальности, так как требования к скорости их работы растут в связи с увеличением разрешений экранов и ростом сложности и детализации 3D-сцен.

Современные экраны имеют разрешение, как правило, не ниже FullHD (1920 на 1080 пикселей). Всё большую популярность приобретают экраны с разрешением 4K и 8K, которые больше разрешения FullHD в 4 и 16 раз соответственно. Таким образом, задача глобального освещения должна быть решена для миллионов или даже десятков миллионов пикселей. Одновременно с этим, целевая частота кадров мониторов в последние годы становится выше: 120 Гц и 240 Гц, вместо 60 Гц ранее. Некоторые приложения могут допустить работу на частоте 60 Гц, но для приложений виртуальной и дополненной реальности такая

частота неприемлема. В частности, компьютерные игры для некоторых VR платформ не могут пройти обязательную сертификацию без выполнения требований к частоте отрисовки кадра.

В глобальном освещении можно выделить два ключевых типа отражения света:

1. Спекулярные отражения — отражения близкие к зеркальным, вызванные влиянием высокочастотной (в значении «быстроменяющейся») компоненты Двунаправленной Функции Отражения (ДФО, англ. BRDF) в интеграле освещённости.
2. Диффузные отражения — примерно равномерные отражения света во все стороны, вызванные влиянием низкочастотной компоненты ДФО в интеграле освещённости.

Задача моделирования спекулярных отражений на данный момент достаточно эффективно решается методами, использующими трассировку лучей. Особенно высокую производительность можно получить, используя видеокарты с аппаратной поддержкой трассировки лучей. При этом, чем более гладкая поверхность, тем ближе направления лучей. Трассировка пучка схожих лучей является более эффективным паттерном для современных алгоритмов трассировки и требует меньшего количества лучей на пиксель.

Как было упомянуто ранее, для корректного моделирования многократного переотражения света необходимо вычислять многомерный интеграл освещённости, что является вычислительно-сложной задачей. Для точного рендеринга задача вычисления спекулярных отражений представляет отдельную сложность, ввиду нетривиальности поведения многократных отражений. В приложениях, требующих высокой производительности, моделирование такого рода эффектов чаще всего избыточно, и можно использовать методы достаточно простые в реализации и алгоритмически.

Таким образом, с учётом специфики современных приложений компьютерной графики можно выделить ключевые особенности, которые требуются для алгоритма глобального освещения, подходящего для перечисленных задач:

1. Точность вычисления выше, чем у аналогов, при той же скорости вычисления (приложения светотехники, приложения для архитектурного моделирования, симуляторы, инструменты для создания кино и анимации).
2. Поддержка динамических источников света (компьютерные игры, симуляторы). Источник света является динамическим, если для него предусмотре-

но интерактивное изменение позиции, направления, размера, яркости или цвета.

3. Использование этапа предобработки сцены для повышения производительности (приложения виртуальной и дополненной реальности, компьютерные игры, симуляторы).
4. Основной объём вычислений не должен зависеть от количества пикселей (приложения виртуальной и дополненной реальности, симуляторы).
5. Обработка геометрии произвольной сложности (приложения для архитектурного моделирования, компьютерные игры).

Методы глобального освещения можно разделить на две ключевые категории:

1. Полностью динамические методы. Они работают для произвольных сцен и условий освещения без какой-либо предварительной обработки. Данный тип алгоритмов имеет более высокую вычислительную сложность, так как методы, обрабатывающие геометрию сцены, могут быть вычислительно затратными.
2. Методы, использующие предобработку сцены. Данные алгоритмы позволяют перенести большую часть вычислений на этап предобработки, что важно, учитывая требования к производительности современных приложений. При этом они накладывают ограничения на изменение 3D-сцены. Некоторые из них не поддерживают динамические источники света.

Таким образом, актуальность данной работы заключается в необходимости разработки новых методов глобального освещения, превосходящих существующие методы в балансе скорости и точности вычисления, а так же учитывающих требования аппаратного обеспечения и данных, визуализируемых приложениями.

Целью данной работы является исследование и разработка метода вычисления глобального освещения, удовлетворяющего следующим требованиям:

1. Возможность поддерживать высокодетализированные сцены и сцены большого масштаба (включающие модели, состоящие из более 100000 треугольников).
2. Поддержка экранов высокого разрешения (4K и выше).
3. Поддержка произвольного количества отражений света (3 и более).
4. Возможность учитывать первичное освещение от динамических источников света.

5. Возможность равномерного распределения вычислительной нагрузки между кадрами без темпоральных (межкадровых) артефактов.
6. Точность вычисленного освещения выше, чем у аналогов, при той же скорости вычисления . Точность определяется сравнением с эталоном (полученным высокоточным рендерингом, например, трассировкой путей).
7. Сопоставимый или меньший уровень потребления памяти по сравнению с аналогами.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. Разработать решение задачи глобального освещения, основанное на методе излучательности, для сцен различной конфигурации со следующими требованиями:
    а) Поддержка произвольного количества отражений без дополнительных накладных расходов во время визуализации.
    б) Поддержка сцен различного масштаба (размер различается более чем в 1000 раз).
    в) Поддержка высокодетализированных моделей (более 100000 треугольников).
    г) Возможность распределения вычислений между кадрами без артефактов.
2. Провести экспериментальное исследование предложенных методов.

Научная новизна: Получены следующие новые результаты в области вычисления глобального освещения на графических процессорах:

1. Предложен метод матрицы нескольких отражений, решающий проблему линейной зависимости сложности вычислений от количества переотражений света без увеличения потребления памяти на GPU.
2. Предложен метод локальной матрицы, решающий проблему сложности вычисления освещения для сцен произвольного размера путём схемы потоковой загрузки данных на видеокарту.
3. Предложен метод темпоральной излучательности, позволяющий распределять вычисления между кадрами без артефактов с сохранением и переиспользованием информации о предыдущем вычисленном освещении.
4. Предложен метод создания виртуальных площадок, позволяющий создавать прокси геометрию, используемую для вычисления освещения, которая требует меньшего количества памяти для 3D сцены (вместо 200000 тре-

угольников используется 12000 виртуальных полигонов), чем методы с явной прокси геометрией. При этом, предложенный метод позволяет сохранить высокую точность вычисления освещения.

Суммарно данные методы позволяют достичь значительно более высокой скорости вычисления освещения (до 100 раз), чем базовый метод излучательности при сопоставимых точности (SSIM отличается не более чем на 0.04) и затратах памяти.

Практическая значимость. Разработанные алгоритмы могут быть разделены на две категории: предназначенные для предобработки сцены и используемые при визуализации. Утилиты, предобрабатывающие сцены, могут быть интегрированы в различные рабочие процессы с графическими движками и не требуют вмешательства пользователя после минимальной настройки. Таким образом, требования к трудозатратам для их использования минимальны. Алгоритмы реализации (для предобработки и для визуализации сцены) выполнены в виде вычислительных шейдеров, которые поддерживаются всеми актуальными программными и аппаратными средствами расчёта на графических процессорах. Результаты алгоритма могут быть представлены в виде различных 3D-структур. Это позволяет пользователю более эффективно контролировать потребление памяти приложением.

Основные положения, выносимые на защиту:

1. 4 разработаных метода позволяют ускорить вычисление глобального освещения на графических процессорах:
    а) Разработаный метод матрицы нескольких отражений позволяет снизить вычислительную сложность решения уравнения излучательности, обеспечивая поддержку произвольного количества переотражений света без увеличения затрат на этапе визуализации. Предложеный алгоритм упаковки форм-факторов с блочным сжатием снижает потребление памяти до 10 раз. Метод частично перераспределяет вычислительные затраты на этап предобработки, увеличивая его время на 20–30%. Данный метод позволяет удовлетворить требование 3 (поддержка произвольного количества отражений света), так как сложность вычислений во время визуализации не зависит от количества переотражений света.
    б) Разработаный метод локальной матрицы уменьшает количество полигонов, для которых требуется хранение матрицы форм-факторов

в видеопамяти. Предложеный подход к обновлению данных матрицы при движении камеры сокращает объем вычислений в 6–15 раз. Алгоритм обеспечивает выполнение требования 1 (поддержка сцен большого масштаба) за счёт потоковой обработки удалённых от камеры участков сцены.

в) Разработаный метод виртуальных площадок обеспечивает применение метода излучательности и его вариаций к 3D-моделям с высокой детализацией. Алгоритм генерирует матрицу форм-факторов и другие данные для вычисления освещения без создания прокси-геометрии, что повышает точность освещения по сравнению с вокселизацией. Предложенный метод удовлетворяет требование 1 (поддержка сцен высокой детализации).

г) Разработаный метод темпоральной излучательности позволяет распределить вычислительную нагрузку между несколькими кадрами без видимых темпоральных артефактов. Алгоритм обеспечивает ускорение вычислений в 10–100 раз по сравнению с классическим методом излучательности без значительных потерь точности освещения. Предложенный метод удовлетворяет требование 5 (распределение вычислений между кадрами без темпоральных артефактов).

2. Проведенное экспериментальное исследование предложенных методов на сценах различной сложности показывает, что предложенные методы обеспечивают более высокую точность при лучшей производительности, чем метод RTXGI (ведущий метод вычисления глобального освещения на основе полей освещенности).

Использование метода излучательности в качестве базового метода позволяет удовлетворить требования 2 (поддержка экранов высокого разрешения) и 4 (поддержка динамических источников света), так как данный метод позволяет проводить вычисления независимо от количества пикселей на экране и конфигурации первичного освещения. Сравнение предложенных методов с RTXGI показало, что предложенные подходы обеспечивают более высокую точность при лучшей производительности. Из проведенного сравнения следует, что комбинация предложенных методов позволяет удовлетворить требования 6 (точность освещения выше, чем у аналогов) и 7 (сопоставимое с аналогами потребление памяти). Сравнения проводились на сценах Crytek Sponza и Cornell Box, ис-

пользуемых в качестве тестовых сцен для алгоритмов глобального освещения производителями GPU [19; 20].

Апробация работы. Основные результаты работы докладывались на следующих конференциях и семинарах:

1. Международная конференция WSCG'2023 (Чехия, Пльзень).
2. Открытая конференция ИСП РАН 2021 (Москва).
3. 31-я Международная конференция по компьютерной графике и машинному зрению ГрафиКон-2021 (Нижний Новгород, НГТУ).
4. Международная конференция WSCG'2019 (Чехия, Пльзень).
5. Международная конференция по компьютерной графике и машинному зрению GraphiCon'2019 (Брянск, БГТУ).
6. Международная научная конференция студентов, аспирантов и молодых ученых "Ломоносов 2019" (Москва).
7. Семинар "Программирование" им. М. Р. Шура-Бура (2023, Москва).

Личный вклад. Содержание диссертации и основные положения, представленные на защиту, отражают личный вклад автора в проведенных исследованиях. В ходе подготовки результатов к публикации было выполнено совместное взаимодействие с соавторами, однако, следует подчеркнуть, что определяющий вклад в исследования принадлежит диссертанту. Все представленные в диссертации результаты были получены исключительно автором самостоятельно. В работах [А.1, А.2] В. А. Фролову принадлежит постановка задачи и консультирование. В работах [А.3] вклад В. А. Фролова и В. А. Галактионова заключается в постановке задачи, консультировании и подготовке тестовых данных. В работе [А.4] все результаты получены автором самостоятельно.

Публикации. Основные результаты по теме диссертации изложены в 4 публикациях, изданных в рецензируемых научных изданиях, определенных в п. 2.3 Положения о присуждении ученых степеней в Московском государственном университете имени М.В. Ломоносова.

Объем и структура работы. Диссертационная работа состоит из введения, 6 глав и заключения. Полный объём диссертации составляет 81 страницу, включая 31 рисунок и 3 таблицы. Список литературы содержит 77 наименований.

Глава 1. Предметная область

1.1 Проблема глобального освещения

Глобальное освещение — это ключевой аспект визуализации трехмерных сцен, который учитывает косвенные эффекты света, такие как переотражения, преломления и тени. Оно играет важную роль в создании реалистичной и убедительной атмосферы в компьютерной графике, виртуальной реальности, а так же для приложений светотехники, где требуется высокая точность визуализации.

Однако, достижение высокого уровня глобального освещения является нетривиальной задачей из-за высокой вычислительной сложности. Проблема глобального освещения заключается в необходимости учета множества сложных взаимодействий света с различными поверхностями сцены. Для каждого пикселя изображения необходимо определить, как свет взаимодействует с окружающими объектами, и какой будет окончательный цвет этого пикселя.

Физически точные методы моделирования глобального освещения, такие как трассировка лучей и методы Монте-Карло, обеспечивают высокую степень реализма, но требуют значительных вычислительных ресурсов. Это ограничивает их использование в приложениях, где требуется обеспечить высокую производительность.

Для формализации проблемы глобального освещения можно воспользоваться формулами и уравнениями, описывающими взаимодействие света с поверхностями сцены. Например, уравнение рендеринга освещения (Rendering Equation) является фундаментальным уравнением для моделирования глобального освещения в трехмерной графике:

$$L_o(x, \vec{v}) = L_e(x, \vec{v}) + \int_{\Omega} \rho_{bd}(x, \vec{v}, \vec{l}, \vec{n}) \cdot L_i(x, \vec{l}) \cdot (\vec{n} \cdot \vec{l})\, d\omega, \tag{1.1}$$

где L_o — яркость излучаемая точкой x в направлении $\vec{v}$, L_e — эмиссивность поверхности в точке x в направлении $\vec{v}$, ρ_{bd} — двулучевая функция отражательной способности, L_i — яркость света, падающего на точку x в направлении $\vec{l}$, $\vec{n}$ — нормаль поверхности в точке x, а Ω — полусфера, заданная нормалью $\vec{n}$ (рис. 1.1).

Разница между L_o и L_i заключается в том, что в первом случае аргумент x является точкой, которая излучает и отражает свет, а во втором — принимает. Как правило, в графических приложениях допускается, что свет исходит от некоторой

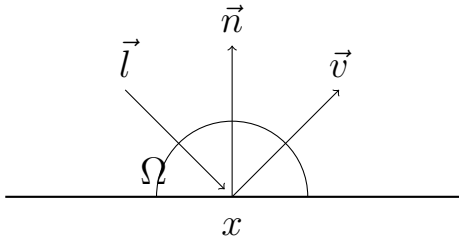


Рисунок 1.1 — Векторы входящего и исходящего света.

поверхности, то есть для $L_i(x, \vec{l})$, есть некоторая точка сцены y, свет от которой попал на точку x в направлении $\vec{l}$. Выражая формулами, получим:

$$L_i(x, \vec{l}) = L_o(y, \vec{l})$$

Таким образом, для вычисления яркости одной точки сцены, нужно знать яркость остальных точек сцены и уравнение для L_o является рекуррентным.

Это интегральное уравнение Фредгольма 2 рода, в котором одна и та же величина L стоит как под интегралом, так и снаружи его. Этим обуславливается вычислительная сложность интегрирования освещённости и многомерность самого интеграла.

На практике глобальное освещение разделяют на два феномена: первичное освещение и вторичное.

Первичное освещение — освещение, падающее на геометрию сцены непосредственно из источника света. Этот тип освещения создаёт тени, которые дают высокий контраст изображениям и существенно необходимы для реалистичных изображений.

При помощи уравнения 1.1, первичное освещение выражается следующим образом:

$$L_o(x, \vec{v}) = L_e(x, \vec{v}) + \int_\Omega \rho_{bd}(x, \vec{v}, \vec{l}, \vec{n}) \cdot L_e(x, \vec{l}) \cdot (\vec{n} \cdot \vec{l})\, d\omega, \tag{1.2}$$

где L_e является функцией, зависящей от материалов поверхностей сцены.

Для простых источников света, таких как

- точечный — задаётся точкой в 3D пространстве, яркостью и цветом;
- направленный — задаётся вектором в 3D пространстве, яркостью и цветом;
- прожектор — задаётся бесконечным конусом, то есть точкой и углом;
- и других

вычисление первичного освещения или теней является задачей трассировки луча. Для них, интеграл в уравнении 1.2 заменяется более простой функцией от L_e.

Более сложную задачу представляет вычисление тени от площадных источников света, так как такие источники создают мягкую тень или полутень. В настоящий момент эта задача решается некоторыми аппроксимациями. Самые популярные методы для её решения — карты теней и трассировка лучей методом Монте-Карло.

Вычисление вторичного освещения, то есть света отраженного поверхностями, является более сложной задачей. С одной стороны, точность вычисления менее важна, чем для первичного освещения, так как незначительные артефакты будут сложно незаметны. С другой стороны, постановка задачи — приближенное решение рекуррентного интегрального уравнения, значительно сложнее чем решение задачи окклюзии.

Первичное освещение, как правило, является полностью динамическим феноменом в современных графических приложениях, то есть оно пересчитывается и остаётся актуальным при изменениях геометрии сцены (деформации, движения), при изменении материалов объектов, при изменении параметров (в том числе и позиций) источников света и при изменении камеры.

Для вторичного освещения полная динамичность требует огромных вычислительных затрат и неприменима в большинстве интерактивных приложений. Чаще всего существующие методы поддерживают изменение источников света и изменения камеры, некоторые методы позволяют менять материалы. Для изменений геометрии чаще всего применяется следующий подход: вторичное освещение рассчитывается для статической геометрии, динамическая геометрия не влияет на освещение статической геометрии (кроме, может быть инициализации, если для неё используются результаты вычисления первичного освещения), на динамическую геометрию влияет освещение, отраженное от статической геометрии. Стоит отметить, что изменение материала — редко поддерживаемое свойство алгоритма, так как в большинстве приложений такая функциональность не нужна.

1.2 Анализ основных литературных источников

1.2.1 Диффузные и спекулярные отражения

В современных графических приложениях как правило используются модели материалов, которые разделяют отражения на диффузные и спекулярные [21].

Диффузное отражение отвечает за рассеянное отражение света от материала в разные направления, что создает мягкие и рассеянные тени. Этот тип отражения особенно важен для матовых и неоднородных поверхностей, таких как дерево, кожа или ткань. Системы материалов позволяют контролировать интенсивность диффузного отражения и подстраивать его под определенные условия освещения, чтобы достичь максимально реалистичного вида.

Спекулярное отражение, с другой стороны, отвечает за блики и отражения света от гладких, блестящих поверхностей, таких как стекло, металл или вода. Это отражение является более направленным и часто имеет яркие характеристики. Системы материалов позволяют контролировать интенсивность спекулярного отражения, а также настраивать его параметры для получения различных эффектов, таких как зеркальные отражения или блеск на поверхности.

Таким образом, разделение отражений на диффузное и спекулярное позволяет создавать более реалистичные и убедительные визуальные образы в графических приложениях, делая виртуальные миры и объекты более правдоподобными для зрителя.

Реализация глобального освещения, учитывающего взаимодействие диффузных и спекулярных отражений, является сложной и требует использования различных аппроксимаций для каждого типа отражения. Дело в том, что диффузные и спекулярные приложения имеют различные физические особенности, которые необходимо учитывать, и применение одних и тех же алгоритмов для обоих типов отражения может быть недостаточно точным и эффективным.

Для моделирования диффузных отражений используются методы, основанные на законах Ламберта [22] и Орена-Наяра [23], которые учитывают рассеяние света в разные стороны от материала. Однако, при применении этих методов к спекулярным поверхностям, они не всегда дают достаточно точные результаты, поскольку не учитывают яркие блики и отражения света от гладких материалов.

Для моделирования спекулярных отражений используются другие аппроксимации, такие как модели Фонга [24] или Блинна-Фонга [25], которые учитывают отражения света в зависимости от угла падения и нормали к поверхности. Эти методы обеспечивают реалистичные блеск и блики на гладких материалах, но при применении к диффузным поверхностям, они могут создавать нереалистичные эффекты и потерю деталей.

Комбинирование этих различных аппроксимаций в одном решении становится сложной задачей, поскольку необходимо балансировать между ними и

учитывать их взаимодействие с окружающей средой и источниками света. Недостаточная точность или неправильное сочетание может привести к искажениям и нереалистичному отображению сцены [26].

1.2.2 Предобработка сцены

Методы глобального освещения в приложениях можно разделить на две группы: методы с предварительным вычислением освещения и методы освещения, поддерживающие полностью динамическое окружение [27; 28].

1. Методы с предварительным вычислением освещения (Предобработкой сцены) [29—33]. В этих методах освещение предварительно вычисляется и сохраняется в текстурных картах или других форматах, которые затем применяются на сцену в режиме визуализации. Это позволяет сэкономить время вычислений во время работы приложения.

   Достоинства:

   – Высокая точность освещения: предварительное вычисление позволяет получить высокоточные результаты, так как учитываются множество освещающих условий и эффектов.
   – Низкие требования к производительности: поскольку освещение уже рассчитано, производительность при визуализации может быть достигнута сравнительно низкими требованиями к аппаратному обеспечению.

   Недостатки:

   – Ограниченная динамичность: предварительно вычисленное освещение не учитывает изменения сцены или источников света во время визуализации, что может ограничить возможности динамических эффектов.
   – Большой объем памяти: хранение предварительно вычисленных данных может требовать значительного объема памяти, особенно для сложных сцен.
2. Динамические методы освещения [34—42]. В отличие от предварительного вычисления, динамические методы освещения рассчитывают освещение полностью во время работы приложения, учитывая текущее состояние сцены и динамические изменения.

   Достоинства:

- Полная динамичность: эти методы позволяют учитывать все изменения сцены, включая движение объектов и источников света, что создает более реалистичные и живые визуальные эффекты.
- Гибкость: использование методов без предобработки геометрии позволяет создавать интерактивные сцены, анимации и эффекты, которые могут реагировать на пользовательские действия.

Недостатки:

- Высокие требования к производительности: расчет освещения может быть вычислительно затратным, особенно для сложных сцен или при использовании сложных алгоритмов освещения.
- Ниже точность: в сравнении с предварительным вычислением, динамические методы могут иметь менее реалистичное освещение из-за ограничений времени вычислений.

В итоге, методы с предварительным вычислением освещения обеспечивают высокую точность и низкие требования к производительности, но с ограниченной динамичностью, в то время как полностью динамические методы освещения обладают большей гибкостью, но могут потребовать более мощных систем для поддержания высокого уровня детализации и реалистичности в динамических сценах.

Глава 2. Обзор существующих решений

2.1 Решения, работающие полностью динамически

2.1.1 Трассировка лучей и трассировка путей

Можно выделить два основных подхода для вычисления глобального освещения динамически, которые напрямую связаны с современным высокоточным рендерингом:

1. Трассировка лучей:

   Трассировка лучей является фундаментальным методом в компьютерной графике, используемым для определения видимости объектов в сцене и вычисления освещения. Основная идея заключается в том, чтобы «бросить» лучи из камеры через каждый пиксель экрана и определить, какие объекты и поверхности они пересекают. Это позволяет вычислить цвет каждого пикселя, учитывая освещение и отражение света от окружающих объектов. Трассировка лучей может быть использована для реализации диффузных и спекулярных отражений, создания теней и других эффектов глобального освещения.

2. Трассировка путей:

   Трассировка путей представляет более сложный и мощный метод для моделирования глобального освещения. В этом методе лучи не только исходят из камеры, но также могут рассеиваться и отражаться на поверхностях, создавая пути света в сцене. Основная идея трассировки путей — моделировать случайные пути света и учитывать взаимодействие с различными материалами и источниками света. Это позволяет создавать более сложные и реалистичные эффекты, такие как мягкие тени, зеркальные отражения и отражения от стекла.

При реализации этих методов, основной проблемой производительности становится перебор полигонов, с которыми нужно проверить пересечение луча. Для решения данной проблемы используют ускоряющие пространственные структуры данных. Основная идея этих структур заключается в рекуррентном разбиении пространства на части. Можно выделить следующие популярные решения:

1. KD-деревья [43; 44].

2. Окто-деревья [45].
3. Иерархия ограничивающих объёмов [46].

Помимо реализации ускоряющей структуры, в приложении должны быть реализованы функции, которые определяют распределение отраженных от поверхности лучей в соответствии с материалом поверхности и генерируют отраженные лучи в соответствии с этим распределением. Чаще всего в качестве математической модели, для которой такие функции уже реализованы выбирают PBR [26] или GGX [47], как дающие результат близкий к реальности и быстрые в вычислении.

С появлением аппаратной поддержки трассировки лучей на видеокартах [48], алгоритмы, основанные на трассировке стали широко распространены в интерактивных приложениях. При этом графические API не только предоставляют функции и поддержку новых шейдеров для трассировки, но и строят ускоряющие структуры.

На данный момент в графических API DirectX12 и Vulkan поддерживаются два варианта работы с трассировкой:

1. Конвейер шейдеров трассировки [49], который содержит специальные шейдеры для генерации лучей, обработки пересечений лучей с геометрией и обработки промахов. Данный конвейер наиболее удобен для реализации сложных алгоритмов трассировки, таких как трассировка путей.
2. Inline tracing [50] — это расширение стандартных шейдеров, позволяющее выполнять операцию трассировки во всех типах шейдеров, что увеличивает гибкость при реализации алгоритмов, использующих трассировку совместно с другими вычислениями или визуализацией.

Отдельно нужно отметить, что так как неотъемлемым атрибутом большинства приложений компьютерной графики являются динамические объекты, отдельный интерес для исследований представляют ускоряющие структуры данных для трассировки, которые могут быстро перестраиваться и меняться для представления меняющейся и деформирующейся геометрии. В современных графических API реализована поддержка динамических BVH, которые могут быть эффективно построены во время визуализации, но качество таких структур ниже, чем у статических.

Основной проблемой трассировки лучей и в особенности трассировки путей является требование большого количества обрабатываемых лучей для получения приемлемого результата. При недостатке лучей, изображение или эффект

отражения получается шумным. Несмотря на наличие аппаратной поддержки в современных видеокартах и параллельную обработку, трассировка всё ещё остаётся ресурсозатратным процессом, поэтому её применяют в комбинации с другими техниками.

Для решения проблемы шума при недостаточном количестве лучей, используются различные методы подавления шума (денойзеры). Такой подход не является полностью физически корректным в силу эвристической природы этих методов, но для приложений, не требующих высокоточного освещения, это может быть приемлемым компромиссом между скоростью и точностью. Существуют как простейшие, хорошо исследованные методы, такие как билатеральный фильтр [51], который опирается на данные в окрестности пикселя, так и более продвинутые алгоритмы, такие как ReSTIR [52]. Метод ReSTIR, разработанный компанией Nvidia, учитывает не только данные вокруг пикселя, но и и данные с предыдущих кадров, таким образом этот алгоритм является темпоральным.

Переиспользование данных предыдущего кадра, является распространённым подходом для алгоритмов, требующих высокого количества вычислений. Естественно, лучше всего такие методы показывают себя при вычислении величин, которые не зависят от позиции и направления камеры. То есть, диффузная составляющая освещения и окклюзия окружающего освещения (Ambient occlusion) являются одними из лучших кандидатов для репроекции данных с предыдущего кадра. Однако, для спекулярных отражений, яркость освещения предыдущего кадра может быть неприменима на текущем кадре из-за разницы позиций и направлений камеры.

С другой стороны, отдельную проблему для вычисления диффузного освещения представляет потребность трассировки лучей по всей полусфере, заданной нормалью в данной точке. Из-за этого требования, необходимо читать различные узлы иерархических ускоряющих структур. При выполнении на CPU и в особенности на GPU, чтение различных узлов приводит к постоянным сбросам данных кэшей, что приводит к увеличению времени трассировки. Обработка спекулярных отражений не подвержена этой проблеме, так как лучи сконцентрированы в достаточно узком конусе.

В некоторых приложениях трассировка лучей (путей) является алгоритмом с предобработкой сцены, если ускоряющие структуры построены до непосредственного запуска приложения и хранятся на диске, но наличие динамических

ускоряющих структур позволяет сделать алгоритм полностью динамическим, с некоторым ущербом для скорости его работы.

Несмотря на возможность получения очень высокой точности освещения и аппаратную поддержку операции трассировки, данные методы имеют ряд существенных недостатков:

1. Аппаратная трассировка лучей поддерживается далеко не на всех устройствах. Наличием такой функциональности выделяются только последние поколения видеокарт компаний Nvidia и AMD. Выбор мобильных устройств со специализированным аппаратным обеспечением для трассировки ещё меньше, чем для персональных компьютеров, а тренажеры и VR устройства не имеют поддержки трассировки вовсе.
2. Даже с наличием аппаратной реализации, трассировка луча остаётся дорогостоящей операцией, поэтому для использования во многих приложениях компьютерной графики, необходимо проводить вычисления в существенно меньшем разрешении с последующими операциями повышения разрешения. Естественно, что на современных 4K и 8K мониторах, проблема усугубляется.
3. Сложность ускоряющей структуры зависит от сложности исходной геометрии, и, как следствие, от сложности геометрии зависит скорость выполнения алгоритмов, использующих трассировку, для данной геометрии. Для уменьшения влияния данной проблемы могут использоваться уровни детализации геометрии [53], однако, это уменьшает точность полученного освещения.

2.1.2 Трассировка вокселей конусами

Трассировка вокселей конусами [34] является аппроксимацией метода трассировки лучей. Вместо получения информации об освещении пришедшего в точку в направлении одного луча, вычисляется усредненное освещение полученное для всех лучей внутри конуса.

При инициализации алгоритма, 3D-сцена вокселизируется, то есть создаётся дискретная аппроксимация в низком разрешении. Для полученной воксельной сетки создаются MIP-уровни, каждый из которых представляет геометрию сцены в ещё меньшем разрешении. Следующий уровень меньше предыдущего в 2 раза по каждой из осей, то есть суммарно содержит в 8 раз меньше данных. Данный этап можно производить только один раз для статической геометрии.

После построения воксельной сцены, производится вычисление освещения для данной геометрии. Таким образом, метод получает аппроксимацию исходной сцены с вычисленным первичным освещением. Пересчёт этих данных требуется при каждом изменении позиций и параметров источников света.

При непосредственной визуализации, для каждого пикселя экрана производится трассировка конусов по вычисленным ранее воксельным сеткам. Чем глубже производится трассировка, тем менее детальный уровень воксельной сетки проверяется на наличие геометрии. Если на очередном этапе встретился воксель с геометрией, его освещение добавляется в освещение обрабатываемой точки.

Нужно отметить, что, как и методы основанные на трассировке лучей, данный алгоритм поддерживает и диффузные и спекулярные отражения. Для спекулярных отражений, трассируются один тонкий конус. Для диффузных — несколько конусов с широким раствором, которые суммарно покрывают большую часть полусферы. Выбор раствора спекулярного конуса осуществляется на основании шероховатости поверхности.

Данный метод имеет следующие ключевые недостатки:

1. Сложность работы алгоритма пропорциональна количеству пикселей экрана. Отчасти эту проблему можно уменьшить вычисляя освещение в меньшем разрешении с последующим применением методов повышения разрешения.
2. Поддержка нескольких отражений делает метод существенно дороже в плане вычислительных затрат, так как приходится производить множество чтений из различных частей воксельной сетки, что негативно сказывается на работе кэшей.
3. Освещение внутри длинных узких коридоров может пропадать вследствие грубого представления таких сцен воксельными сетками, так как на большом расстоянии будут прочитаны данные из низкодетализированных MIP-уровней воксельной сетки, в которых коридоры могут исчезать из-за недостатка точности.
4. При этом, трассировка конусами имеет более низкую точность вычисления, чем трассировка лучами.

2.1.3 Мгновенная излучательность

Данный алгоритм [35] основан на идее вторичных источников света. Как можно видеть из уравнения (1.1), отраженный поверхностью свет суммируется со светом, излучаемым этой поверхностью, то есть это одна и та же физическая величина и процесс отражения может быть рассмотрен как два события: получение освещения от окружающей сцены, излучение отраженного освещения. Именно такое разделение позволяет ввести понятие виртуального источника света.

Суть метода заключается в следующем. Используя первичное освещение, на сцене выбираются несколько точек, освещенных источником света. Данные точки становятся вторичными источниками света, для них вычисляется цвет и яркость исходя из полученного ими света и вычисляется видимость сцены из этих точек, чаще всего картами теней [54]. На следующем этапе к первичному освещению добавляется освещение от вторичных источников. Стоит отметить, что для функции, вычисляющей освещение от источника, разницы между первичным и вторичным источником нет.

Такой процесс можно итеративно повторять, добавляя всё новые и новые источники света, которые будут содержать информацию о втором, третьем и последующих переотражениях света. Естественно, с каждой новой итерацией, будет требоваться всё больше и больше ресурсов для вычисления освещения, так что на практике, чаще всего используют только одну итерацию создания вторичных источников.

Данный метод обладает рядом значимых преимуществ:

1. Динамическая геометрия участвует в вычислении вторичного освещения наряду со статической, при условии, что карты теней перестраиваются каждый кадр.
2. Количество используемых отражений и количество источников света являются настройками алгоритма, которые позволяют гибко настраивать его для получения наилучшей точности для конкретного временного бюджета.

Однако, недостатки связанные в первую очередь с производительностью, делают метод неприменимым для современных приложений:

1. При сложной геометрии сцены, с большим количеством деталей и вариацией нормалей поверхностей, может потребоваться огромное количество вторичных источников света для получения достоверного результата, без шума, вызванного движением камеры, объектов или источников света.

2. Каждый новый вторичный источник света требует создания дополнительной карты теней. В приложении придётся либо хранить большой атлас для теней от статических объектов, что значительно увеличивает потребление памяти, либо растеризовать сцену многократно каждый кадр, что представляет огромный объём вычислений ввиду высокой сложности современных 3D-сцен.

Отчасти количество требуемых виртуальных источников можно уменьшить распределив их обработку между кадрами и накапливая данные об освещении в процессе отрисовки всё новых и новых кадров. Этот способ будет работать автоматически в приложениях, использующих темпоральный анти-алиасинг [55] или темпоральное супер-разрешение [56]. В некоторых случаях может потребоваться дополнительная настройка данных алгоритмов, так как значительная разница в яркости между кадрами может приводить к отбрасыванию данных предыдущих кадров, что негативно сказывается на решении проблемы алиасинга.

2.1.4 Отражающие карты теней

Схожим, но менее ресурсозатратным методом чем мгновенной излучательности, является метод отражающих карт теней (Reflective shadow map, RSM) [36]. Для первичных источников света создаётся карта теней, которая помимо глубины содержит нормали и цвета точек сцены. Таким образом, каждый тексель такой расширенной карты теней имеет достаточно информации для создания вторичного источника света. Для ускорения вычислений авторы данного подхода приняли решение не использовать информацию о видимости для вторичных источников освещения, что существенно ускоряет как подготовку вторичных источников, которая по сути заключается в растеризации сцены в несколько каналов для каждого первичного источника, так и использование этих вторичных источников при вычислении освещения.

Очевидно, что при явном выигрыше в скорости, этот метод обладает рядом недостатков, таких как

1. Отсутствие свойства сохранения энергии, связанного с игнорированием видимости точки из источника света. То есть луч, выпущенный источником проходит все поверхности без препятствий и освещает каждую из них.
2. Существенное увеличение памяти для первичных источников. Отчасти это компенсируется отсутствием карт теней для вторичных источников, но если учесть, что точность (и разрешение) этих карт могут быть низкими, в

некоторых случаях метод отражающих карт теней может требовать больше памяти, чем метод мгновенной излучательности.

2.1.5 Объёмы распространения света

Несмотря на явные недостатки, метод отражающих карт теней, позволяет реализовать более сложные и точные техники вычисления вторичного освещения. Одной из таких техник являются объёмы распространения света (Light propagation volumes, LPV) [37].

Так же как и в методе трассировки вокселей конусами, используется вокселизация сцены. С помощью отражающих карт теней, на вокселизированной геометрии вычисляется начальное освещение, которое записывается в трехмерную текстуру, соответствующую вокселям сцены. Далее, производится одна или несколько итераций распространения освещения между вокселями. При вычислении освещения для конкретной точки поверхности производится выборка 3D-текстуру с освещением.

Метод имеет более выскую точность чем отражающие карты теней, но уступает в точности освещения методу трассировки вокселей конусами и методам трассировки лучей, так как освещение в воксельной сетке меняется плавно и упускает часть деталей. Также для этого алгоритма характерны проблемы с «просачиванием» света сквозь препятствия, связанные с тем, что воксельная сетка является достаточно грубым приближением для исходной геометрии. Динамическая геометрия никак не участвует в вычислении освещения, но может производить выборку из воксельной текстуры для получения освещения.

2.1.6 Поле освещенности

Ещё одним методом, реализующим вычисление освещения внутри воксельной стеки вокруг камеры, является метод поля освещенности [38]. В каждом вокселе хранится сферическая функция, которая возвращает облучённость в зависимости от направления. Как правило, эти функции кодируются сферическими гармониками (см. раздел 2.2.2) или схожими методами.

Для вычисления освещения, геометрия растеризуется и внутри каждого вокселя вычисляется начальное освещение. Затем, происходит интегрирование облучённости для каждого вокселя. Данную операцию можно проводить трассировкой лучей по воксельной сетке, чтобы уменьшить затраты на нахождение пересечения луча и геометрии. Лучи трассируются во всех направлениях сферы. Так как такая операция требует большого объёма вычислений, интегрирование

происходит стохастически, то есть каждый кадр выбирается несколько вокселей для обновления и облучённость пересчитывается только для них.

Наложение освещения аналогично методу объёмов распространения света, то есть это просто выборка значений функций облучённости из воксельной сетки. При этом метод обладает теми же проблемами точности, что и другие методы использующие воксельные сетки.

RTXGI

Для решения проблемы просачивания света, кампанией Nvidia был предложен алгоритм RTXGI [39], который хранит атлас текстур с октаэдральной проекцией, содержащих значения облучённости для разных направлений. Это увеличивает точность значения облучённости. Также отдельно хранится атлас текстур глубины в октаэдральной проекции, который используется при наложении освещения на объект, чтобы проверить, может ли данный воксель освещать рассматриваемую точку поверхности или между ними есть препятствие.

Для построения этих атласов используется аппаратная трассировка лучей, поддерживаемая на последних моделях видеокарт. Таким образом, несмотря на высокую точность, этот подход не является широко распространённым из-за аппаратных ограничений.

2.1.7 Методы вычисления глобального освещения в экранном пространстве

Подходы, которые работают в экранном пространстве, используют информацию на экране, для получения некоторого приближения освещения. Чаще всего для этого используется Gbuffer в приложениях с отложенным освещением и буфер глубины. Естественно, что информации на экране недостаточно для получения полноценного глобального освещения, поэтому эта группа методов используется либо на устройствах с крайне низкой производительностью, либо в комбинации с другими методами, для учета мелких деталей в освещении, которые могут быть упущены в основном методе из-за слишком грубого приближения, но вполне реализуемы с использованием части сцены, видимой на экране.

SSDO

Алгоритм Screen Space Directional Occlusion [40], схож с методом отражающих карт теней, так как использует те же данные схожим образом, разница заключается в том, что данные о цвете пикселей, их нормалях и глубине не генерируются отдельно, а получаются из уже имеющихся данных. Радиус действия

вторичных источников света является параметром алгоритма, который можно настроить, в том числе добавив зависимость от глубины.

Естественно, что этому алгоритму присущи все те же недостатки в точности, как и методу отражающих карт теней, но при применении в небольшом радиусе, артефакты будут незаметны в большинстве случаев. Использовать данных подход можно только в приложениях с отложенным освещением, так как он требует наличия данные о нормалях и цвете пикселей в виде текстур.

Screen space ambient occlusion

Алгоритм screen space ambient occlusion [41] реализует вычисление окклюзии освещения в экранном пространстве. Окклюзия окружения — число от 0 до 1, показывающее сколько освещения от окружения достигает поверхности. В приложениях эта величина используется, для того чтобы добавить детализацию глобальному освещению, вычисленному более точными методами. Для большинства случаев, данных из экранного пространства достаточно для получения окклюзии окружения. Проблемы возникают только на границе экрана.

Ground truth ambient occlusion

Развитием метода SSAO является алгоритм ground truth ambient occlusion (GTAO) [42], который, как и методы HBAO (Horizon based ambient occlusion) и HBAO+ [57; 58] вычисляет окклюзию на основании угла между нормалью и вектором из точки к окклюдеру в некотором направлении. Этот алгоритм распределяет определение окклюзии между несколькими кадрами является темпоральным алгоритмом.

Для увеличения точности вычислений, GTAO не только затеняет вторичное освещение, но и добавляет некоторые детали. Для этого вычисляется полином от значения AO, который используется как множитель для цвета пикселя. Таким образом, это добавочное освещение не учитывает цвет окружающей геометрии исходя из предположения, что в большинстве случаев цвет поверхностей более-менее однороден.

2.2 Методы, использующие предобработку сцены

2.2.1 Карты освещенности

Карты освещенности [29; 30] — это наиболее простой и быстрый метод глобального освещения. На этапе предобработки, для поверхностей сцены произ-

водится вычисление освещения высокоточным методом, например трассировкой путей. Полученное освещение упаковывается в текстурные атласы.

При визуализации, достаточно получить координаты текстеля, соответствующего точке поверхности, внутри текстурного атласа и считать данные об освещении из атласа.

Этот метод позволяет вычислить освещение с практически любой заранее заданной точностью, но обладает рядом серьёзных ограничений:

1. Динамические источники света, не могут влиять на глобальное освещение.
2. Динамические объекты не влияют на вторичное освещение и не учитывают его на себе.
3. Материалы объектов не могут быть изменены во время визуализации.
4. Для современных сцен, карты освещенности высокой детализации будут занимать большое количество памяти.

2.2.2 Сферические гармоники

Сферические гармоники [32] — это метод представления функции, аргументом которой является направление в 3D-пространстве. По сути это разложение функции в ряд и в некоторых случаях использование нескольких первых коэффициентов такого ряда даёт достаточно хорошее приближение для исходной функции. Сферические гармоники хорошо подходят для кодирования диффузного освещения, так как в отличие от спекулярного, оно меняется плавно и не требует большого количества коэффициентов для представления.

Сферические гармоники вычисляются для световых проб — ключевых точек сцены, освещение в которых будет хорошим приближением для глобального освещения.

Есть два основных варианта использования гармоник:

1. Предподсчёт гармоник. В этом случае художники выбирают точки, в которых будут вычислены гармоники. После этого запускается процесс вычисления коэффициентов гармоник для расставленных проб. Эти функции будут содержать только информацию о геометрии, окружающей гармонику. В процессе визуализации, гармоники источников света комбинируются с гармониками, расставленными по сцене и получается финальная функция, используемая для вычисления освещения.
2. Автоматическая расстановка проб. Этот подход может быть полностью динамическим, но он содержит некоторые технические трудности, которые

нужно индивидуально решать для каждого проекта: как размещать пробы так, чтобы они не пересекались с геометрией сцены, какие точки будут давать наилучшее вторичное освещение, сколько коэффициентов будет достаточно для каждой пробы.

Для использования сферических гармоник при визуализации, коэффициенты гармоник умножаются на полиномы, полученные из координат нормали поверхности и складываются. По сути, для каждой пробы хранятся 3 гармоники, по одной для каждого цветового канала.

Главным недостатком этого метода является требование сложной настройки сцен для размещения проб, либо сложность реализации алгоритма для автоматического размещения проб. Отдельной проблемой является выбор компромиссного решения между количеством проб и количеством хранимых коэффициентов каждой пробы, так как оба этих параметра влияют на точность и скорость вычислений.

2.2.3 Предвычисленное поле освещенности

Метод поля освещенности, описанный в 2.1.6, может быть реализован как метод с предобработкой сцены [31]. В этом случае вся сцена покрывается 3D-текстурами (воксельными объемами), для которых irradiance вычисляется на этапе предобработки сцены. При этом объёмы могут пересекаться, что будет давать более детальное освещение в некоторых точках сцены.

Основное преимущество этого подхода заключается в том, что во время визуализации не нужно тратить время на пересчёт освещения и в отличие от карт освещенности, вторичное освещение может быть наложено на динамические объекты. Но в то же время, метод не поддерживает динамические источники света и требует большого количества памяти.

2.2.4 Нейросетевые методы

В этой группе методов для сцены обучается некоторая нейросетевая модель [59; 60], которая предсказывает освещение для конкретных условий источников света. В настоящее время такие методы не применимы во многих приложениях из-за своей высокой вычислительной сложности. Так как модель обучается заранее, она учитывает только статическую составляющую сцены и не предоставляет какие-либо данные, которые могут быть использованы для освещения динамической геометрии. Отдельной проблемой этой группы методов является слабый контроль над освещением со стороны разработчиков, так как

ошибки могут быть исправлены только неявно — изменением обучающей выборки.

Также разработаны нейросетевые методы, которые позволяют ускорить вычисление глобального освещения, основанное на трассировке лучей [61]. Такие подходы позволяют достичь достаточно высокой производительности, однако, требуют дополнительного использования алгоритмов уменьшения шума и наличия аппаратной поддержки трассировки лучей. Этот метод также можно отнести к полностью динамическим методам, с той точки зрения, что у разработчика нет необходимости обучать или дообучать нейросеть для каждой новой сцены или при изменении существующей сцены.

Методы, использующие одну нейросеть для разных сцен и принимающие на вход данные G-буфера [62], не могут учитывать данные об освещении от объектов, находящихся за наблюдателем. То есть это методы работающие в экранном пространстве. Поэтому точность таких методов достаточно низкая и при движении камеры или объектов на сцене освещение будет неправдоподобно изменяться.

2.2.5 Излучательность

Метод излучательности [33] берет своё название от физической величины излучательность (Radiosity), с которой производятся вычисления в данном методе. Изначально алгоритм использовали для высокоточного рендеринга, поскольку он был быстрее трассировки путей и давал хорошее начальное приближение, однако, со временем, метод излучательности был адаптирован для интерактивных приложений с этапом предобработки сцены.

Классический алгоритм

Предобработка сцены в методе излучательности заключается в вычислении форм-факторов.

Форм-фактор F_{ij} для двух полигонов P_i и P_j — число, показывающее какое количество света перешло с полигона P_i на полигон P_j при отражении/излучении света полигоном P_i. Это безразмерная величина, в пределах от 0 до 1. Так как форм-факторы — это попарные отношения между полигонами, то все вместе они формируют матрицу F.

Идея алгоритма излучательности заключается в том, что для пар полигонов вычисляются форм-факторы и эти данные используются, для того чтобы выразить уравнение (1.1) для конечных элементов, которыми являются полигоны сцены P,

а затем, полученные данные используются для вычисления приближенного освещения.

Форм-факторы представляют собой интеграл по точкам полигонов:

$$F_{ij} = \frac{1}{S_i} \int_{x \in P_i} \int_{y \in P_j} \frac{\cos(\vec{r}, \vec{n_x}) \cos(\vec{-r}, \vec{n_y})}{\pi |\vec{r}|^2} V(x, y) dx dy, \tag{2.1}$$

где S_i — площадь полигона P_i, $\vec{r}$ — вектор из точки x в точку y, n_x, n_y — нормали в точках x, y, $V(x, y)$ — функция видимости, возвращающая 1, если между точками x и y нет препятствий и 0 в противном случае.

При наличии матрицы форм-факторов, вычисленной на этапе предобработки сцены, можно свести уравнение (1.1) к следующему виду:

$$B_i = E_i + C_i \cdot \sum_{j=1}^{N} F_{ij} B_j, \tag{2.2}$$

где B_i — излучательность полигона P_i, E_i — светимость полигона, C_i — цвет полигона P_i, N — количество полигонов, участвующих в расчёте.

Из этого уравнения, которое выводится в [33], понятно, что цвет для полигона считается константой и поддерживается только ламбертовская модель отражений, то есть свет отражается равномерно во все стороны. Это допущение используется во многих других алгоритмах глобального освещения, например в трассировке вокселей конусами и поле освещенности.

Для получения излучательности полигонов требуется вычислить значения E_i, которые довольно легко вычисляются из карты теней или трассировкой лучей, и решить рекуррентное уравнение (2.2).

Введем матрицу

$$C_{mat} = \begin{pmatrix} C_1 & C_1 & \dots & C_1 \\ C_2 & C_2 & \dots & C_2 \\ \vdots & \vdots & \ddots & \vdots \\ C_N & C_N & \dots & C_N \end{pmatrix}$$

Теперь уравнение (2.2) может быть представлено в матричной форме:

$$B = E + (C \circ F) \cdot B \tag{2.3}$$

Таким образом можно выразить B:

$$(I - (C \circ F)) \cdot B = E \tag{2.4}$$

Уравнения (2.3) и (2.4) дают два алгоритма решения для метода излучательности:

1. Итеративное решение. Из первичного освещения инициализируется вектор E, затем итеративно вычисляются отражения света: $B^i = C \circ F \cdot B^{i-1}$, где $B^0 = E$. Итоговое освещение — это сумма всех отражений $Lighting = \sum_{i=1}^{K} B^i$, где K — количество учитываемых отражений. При этом нулевое отражение не учитывается, так как оно уже учтено в первичном освещении, которое реализуется другими методами.
2. Решение матричного уравнения. Значение вектора B можно найти, решив матричное уравнение (2.4). Снова необходимо инициализировать вектор E из первичного освещения. Также необходимо найти матрицу обратную матрице $I - (C \circ F)$. Это можно сделать на этапе предобработки сцены. Однако, из-за большого количества элементов матрицы и ошибок возникающих при вычислениях с плавающей точкой, найденное решение может приводить к артефактам и быть менее точным, чем решение полученное итеративным алгоритмом.

Как видно, итеративное решение алгоритма излучательности имеет сложность $O(N^2 \cdot K)$, где N — количество полигонов сцены, а K — количество учитываемых отражений. Такая вычислительная сложность является основной проблемой метода излучательности.

Иерархический алгоритм

Алгоритм излучательности может производить видимые артефакты для больших полигонов, освещение для которых сильно различается в разных его частях. Для решения этой проблемы исходные полигоны, как правило, подразбивают, что увеличивает значение N в сложности вычисления алгоритма.

Метод иерархической излучательности [63] реализует более сложную схему подразбиения полигонов, с сохранением нескольких уровней разбиения. Каждый полигон рекуррентно разделяется на несколько полигонов, пока не будет достигнута максимальная глубина рекурсии или разница форм-факторов новых полигонов и исходного полигона не будет меньше некоторого порогового значения.

Само освещение при этом распространяется между разными уровнями иерархии полигонов, таким образом требуется большее количество матричных

операций, хотя сами эти операции оказываются менее затратными, чем при регулярном разбиении полигонов.

Данный алгоритм позволяет уменьшить количество вычислений, если требуется произвести разбиение больших полигонов, но для современных сцен, которые зачастую имеют большое количество высокодетализированной геометрии, этот метод не представляет интереса.

Динамические решения

Метод излучательности имеет несколько расширений [64—67], позволяющих вычислять освещение для динамических объектов, в том числе и с учетом динамической камеры. Все эти алгоритмы были разработаны для излучательности как для метода высокоточного рендеринга, то есть без учета условий интерактивных приложений. Поэтому под динамическими объектами и камерой подразумеваются объекты и камера, которые движутся по заранее заданной траектории известной на момент предобработки сцены. Соответственно, данное семейство алгоритмов, представляет собой схемы для эффективной упаковки значений форм-факторов для нескольких последовательных состояний сцены и в интерактивных приложениях не применимы.

Монте-Карло Излучательность

Для метода излучательности разработан ряд итеративных подходов с вычислительной сложностью $O(N)$ основанные на последовательном приближении освещения посредством методов Монте-Карло [68—71]. Данные методы разработаны для вычисления статического изображения аналогично методам основанным на трассировке путей и не имеют адаптации для GPU. Основная проблема этих методов, не позволяющая использовать их во многих приложениях компьютерной графики, заключается в том, что промежуточные результаты вычисления будут некорректны и будут содержать артефакты.

Упрощение геометрии

Как было указано ранее, сложность метода излучательности пропорциональна квадрату от количества полигонов. На практике используют упрощенную (прокси) геометрию, которая имеет значительно меньшее количество полигонов. Вторичное освещение вычисляется для прокси геометрии, а затем накладывается на исходную. Создание упрощенной геометрии для метода излучательности решается одним из двух способов:

1. Вручную художниками [72]. Этот подход увеличивает время работы над 3D сценой и требует наличия навыков по созданию качественной прокси геометрии.
2. Автоматически [73]. При этом точность освещения может быть ниже, так как методы упрощения геометрии, как правило, не учитывают, что отражение света на сцене должно быть как можно ближе к оригиналу.

Выбор базового алгоритма

Метод излучательности был выбран в качестве базового для создания на его основе новых методов глобального освещения, которые бы удовлетворяли требованиям современных приложений компьютерной графики. Выбор был сделан с учетом следующих особенностей базового метода излучательности:

1. Возможность вычисления освещения с высокой точностью.
2. Возможность переноса части вычислений на этап предобработки сцены.
3. Основная сложность вычислений не зависит от количества пикселей экрана.

Глава 3. Преобразование матрицы форм-факторов для учета нескольких отражений

В работе [А.1] предлагается метод матрицы форм-факторов, учитывающей несколько отражений.

Как было указано в разделе 2.2.5, в итеративном алгоритме каждое новое отражение света вычисляется из предыдущего:

$$B^i = C \circ F \cdot B^{i-1} \tag{3.1}$$

При этом все учтенные отражения вычисляются простым суммированием:

$$Lighting = \sum_{i=1}^{K} B^i \tag{3.2}$$

Используя эти два уравнения предлагается ввести новый вариант данных, рассчитанных на этапе предобработки, который позволяет избавиться от множителя K (числа отражений) в сложности метода излучательности.

Введем новую матрицу $F_{color} = C \circ F$, которая содержит форм-факторы с учетом цветов полигонов.

Тогда уравнение (3.1) можно переписать как

$$B^i = F_{color} \cdot B^{i-1} \tag{3.3}$$

Многократно подставив в уравнение (3.2) значения B^i из уравнения (3.3), вплоть до $B^0 = E$, получим следующее выражение:

$$Lighting = \sum_{i=1}^{K} F_{color}^i E \tag{3.4}$$

Вектор E можно вынести из суммы:

$$Lighting = \left(\sum_{i=1}^{K} F_{color}^i \right) E \tag{3.5}$$

Введем обозначение $F_{multibounce} = \sum_{i=1}^{K} F_{color}^i$ для новой матрицы форм-факторов, которая учитывает цвета полигонов и многократные переотражения света. Эту матрицу можно вычислить на этапе предобработки и, таким образом,

процесс вычисления освещения сводится к единичному умножению матрицы на вектор:

$$Lighting = F_{multibounce}E$$

Сложность такой операции $O(N^2)$, то есть сложность вычислений во время визуализации не зависит от количества учтенных отражений.

Полученная матрица будет некоторым приближением матрицы обратной $I - (C \circ F)$ из уравнения (2.4). Если теоретически учесть все возможные отражения в матрице $F_{multibounce}$, то она будет вычисляться как бесконечная сумма:

$$F_{multibounce} = \sum_{i=1}^{\infty} F_{color}^{i}$$

И выражение $(I - (C \circ F))^{-1} = (I - F_{color})^{-1}$ будет результатом такой суммы геометрической прогрессии для матрицы F_{color}.

Матрица $F_{multibounce}$ содержит в 3 раза больше данных, чем исходная матрица, так как в ней учитываются цвета полигонов и имеются данные для каждого из цветовых каналов. Таким образом, видеокарте потребуется читать в 3 раза больше данных и потребуется чаще обновлять данные кешей, так как значения цветовых каналов чередуются.

Для решения этой проблемы с матрицей нескольких отражений, был разработан метод для эффективного сжатия такой матрицы: представление матрицы в формате текстуры с DXT-сжатием [74] и предварительной обработкой для уменьшения ошибок сжатия.

Блочные методы сжатия (DXT, BC) повсеместно используются для текстур, которые читаются на видеокартах, так как они позволяют существенно сократить количество памяти занимаемой приложением и читаемой при выполнении шейдера. Декомпрессия таких текстур не имеет существенных накладных расходов, так как она реализована на аппаратном уровне.

Все виды блочного сжатия, используемые на GPU работают по схожим схемам (см. рис. 3.1): внутри блока пикселей размера 4x4, выбираются два крайних цвета. Все пиксели в блоке кодируются несколькими битами (2 в случае DXT-5) как промежуточный цвет между двумя крайними цветами, то есть цвет этих пикселей будет результатом линейной интерполяции между крайними цветами.

Артефакты при таком сжатии могут возникнуть в блоках, в которых пересекаются 3 и более объектов с разными цветами, что случается довольно редко на реальных данных.

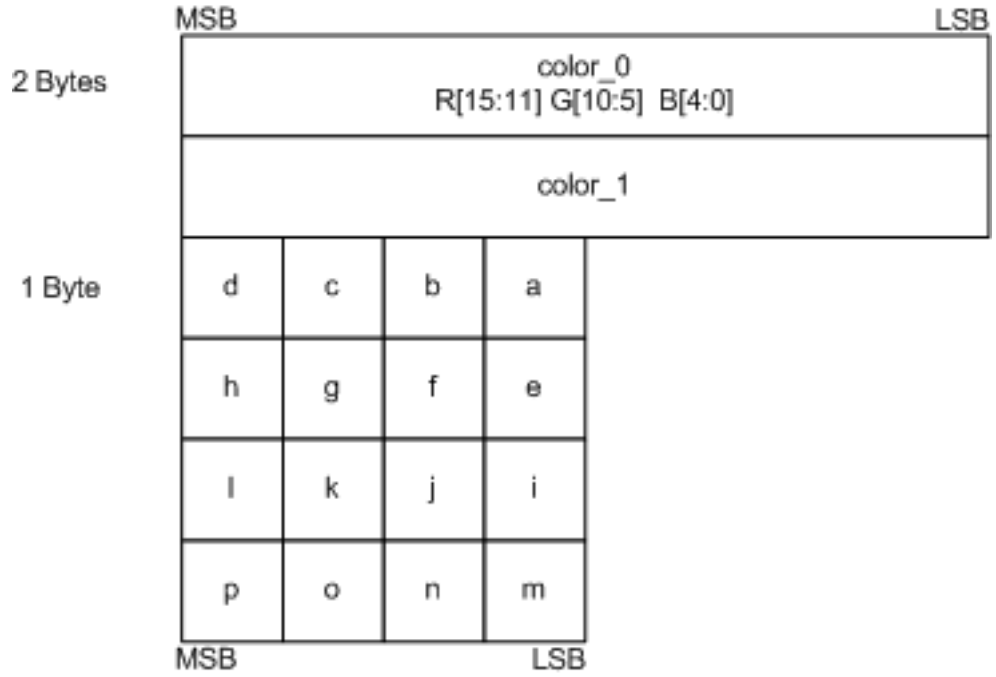


Рисунок 3.1 — Схема блочного кодирования формата DXT

Так как блочное сжатие — это сжатие с потерями, причём наибольший эффект сжатия достигается на данных от 0 до 255, матрица форм-факторов, при сохранении в таком формате, требует некоторой предобработки для уменьшения ошибки сжатия. Если посмотреть на распределение значений в матрице форм-факторов 3.2, можно заметить, что они различаются на несколько порядков. При этом чисел в правой части гистограммы значительно меньше и именно для этих значений наиболее важна точность, так как они вносят больший вклад в освещение. Поэтому предлагается сохранять 5% самых больших чисел в отдельный буфер. Вместо этих значений в матрице сохраняются 0.

Чтобы уменьшить разницу между значениями внутри блоков и тем самым уменьшить ошибку связанную со сжатием текстуры, предлагается жадный алгоритм пересортировки строк и столбцов матрицы. Для его реализации необходимо ввести процедуру перестановки данных в матрице.

Меняя местами два полигона в исходном массиве, необходимо соответствующим образом изменить матрицу форм-факторов. Рассмотрим операцию $swap\,(i, j)$, которая меняет местами полигоны i и j. Для изменения матрицы форм-факторов нужно последовательно выполнить две операции: поменять местами

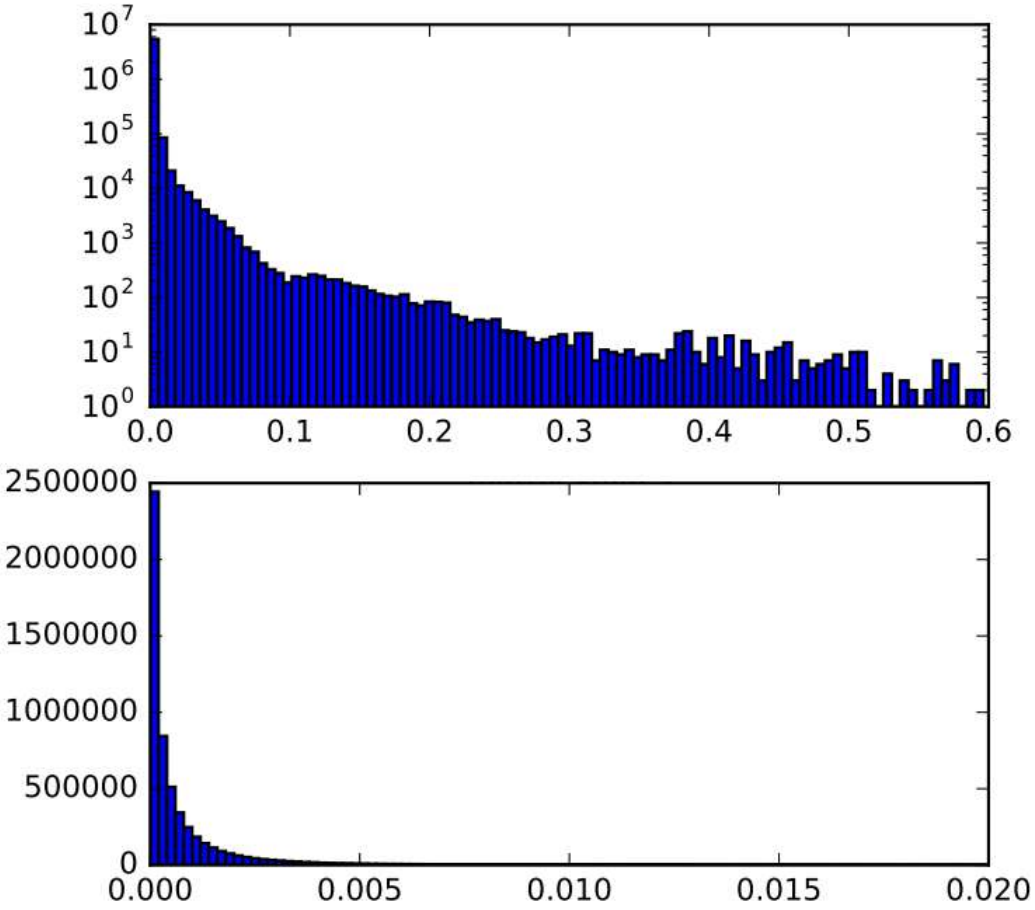


Рисунок 3.2 — Распределение значений форм-факторов. Сверху логарифмические шкалы. Снизу линейные шкалы.

строки i и j, поменять местами столбцы с номерами i и j. При этом диагональные элементы F_{ii} и F_{jj} также поменяются местами.

Жадный алгоритм пересортировки полигонов для уменьшения ошибки сжатия представлен в листинге 3.1.

Функция `polygons_reorder` может быть применена как к обычной матрице форм-факторов, так и к матрице с несколькими отражениями. В предложенном методе данная операция применяется к $F_{multibounce}$. После этого ошибка сжатия (MSE) уменьшается в 5 раз. Результат преобразования матрицы представлен на изображении 3.6.

Несмотря на сжатие с потерями, предложенный алгоритм даёт результаты, которые отличаются от эталона примерно на ту же величину (SSIM: 0.762), что и исходный метод излучательности (SSIM: 0.743) (см. рис. 3.8, 3.9).

Предложенный метод ускоряет вычисление глобального освещения кратно количеству учитываемых отражений света по сравнению с классическим

Листинг 3.1: Жадный алгоритм пересортировки полигонов

```
1:  for c_1 ∈ columns do
2:      closest_dist ← +∞
3:      for c_2 ∈ columns do
4:          if c_1 ≠ c_2 then
5:              dist ← |c_1 − c_2|_2
6:              if dist < closest_dist then
7:                  closest_dist ← dist
8:                  closest_column ← c_2
9:              end if
10:         end if
11:     end for
12:     swap(closest_column, next_column(c_1))
13: end for
14: for r_1 ∈ rows do
15:     closest_dist ← +∞
16:     for r_2 ∈ rows do
17:         if r_1 ≠ r_2 then
18:             dist ← |r_1 − r_2|_2
19:             if dist < closest_dist then
20:                 closest_dist ← dist
21:                 closest_row ← r_2
22:             end if
23:         end if
24:     end for
25:     swap(closest_row, next_row(r_1))
26: end for
```

алгоритмом излучательности 3.10. Так, для 3-х отражений, разработанный алгоритм показывает трехкратное ускорение. Вдобавок к этому, использование DXT-компрессии для форм-факторов, уменьшает количество данных, читаемых видеокартой, что позволяет дополнительно ускорить вычисления. Таким образом, для трёх отражений суммарное ускорение получается около 10 раз 3.10.

Из-за необходимости хранить 3 цветовых канала для форм-факторов вместо одного, количество требуемой памяти увеличивается в 3 раза, но предложенная

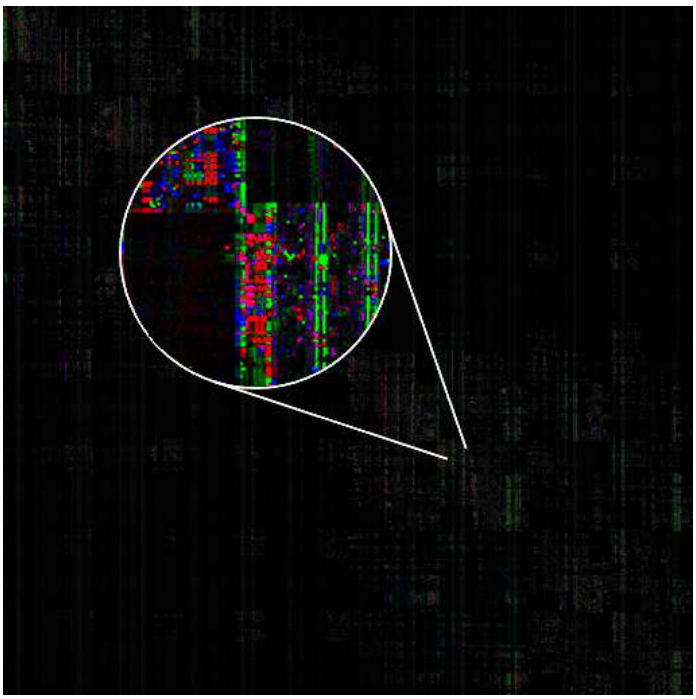


Рисунок 3.3 — Ошибка DXT-компрессии без пересортировки полигонов. Контрастность увеличена.

схема компрессии форм-факторов позволяет уменьшить потребление памяти в 3 раза по сравнению с исходной матрицей форм-факторов 3.11.

Предложенный метод (см. рис. 3.8) имеет меньшую ошибку по отношению к эталонному изображению (SSIM: 0.743) чем метод объёмов распространения света (SSIM: 0.62, см. рис. 3.7) при одинаковом времени отрисовки кадра (около 35 кадров в секунду на GTX 660).

Разработанный метод матрицы нескольких отражений позволяет кратно ускорить вычисление глобального освещения и сократить количество требуемой памяти, но данный метод имеет ряд ограничений:

1. Фиксированные материалы полигонов. Так как цвета полигонов становятся частью форм-факторов, вычисляемых при подготовке сцены, их смена во время визуализации становится невозможной.
2. Фиксированное количество отражений. Данный метод не поддерживает динамическое изменение количества отражений в процессе визуализации. Для большинства приложений это свойство не будет играть никакой роли, так как вычисление глобального освещения для любого количества отражений будет требовать одинакового количества времени. Однако, в приложениях, где требуется исследовать переотражение света на конкретной итерации, предложенный подход будет неприменим.

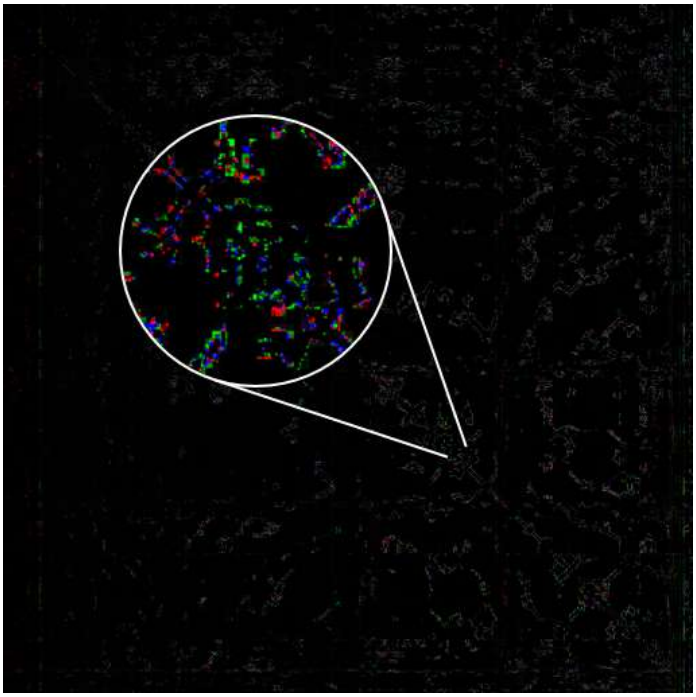


Рисунок 3.4 — Ошибка DXT-компрессии после пересортировки полигонов. Контрастность увеличена аналогично рисунку 3.3.

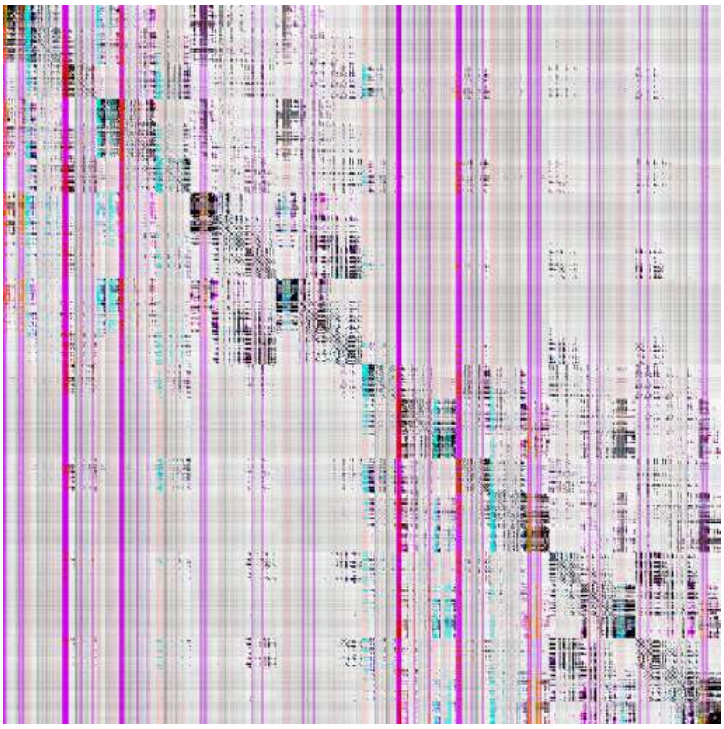


Рисунок 3.5 — Матрица форм-факторов с учетом нескольких отражений.

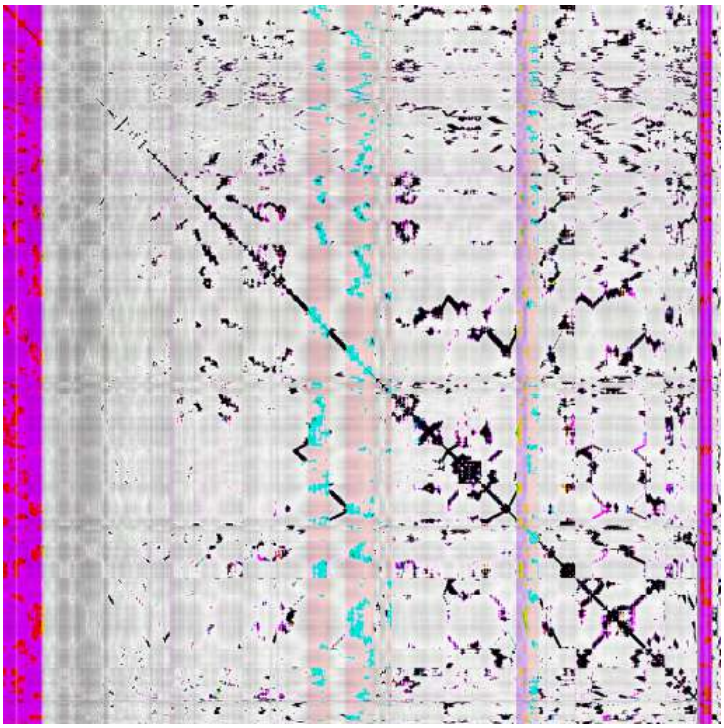


Рисунок 3.6 — Матрица форм-факторов с учетом нескольких отражений после пересортировки полигонов.

Рисунок 3.7 — Освещение, посчитанное методом Объёмов распространения света.

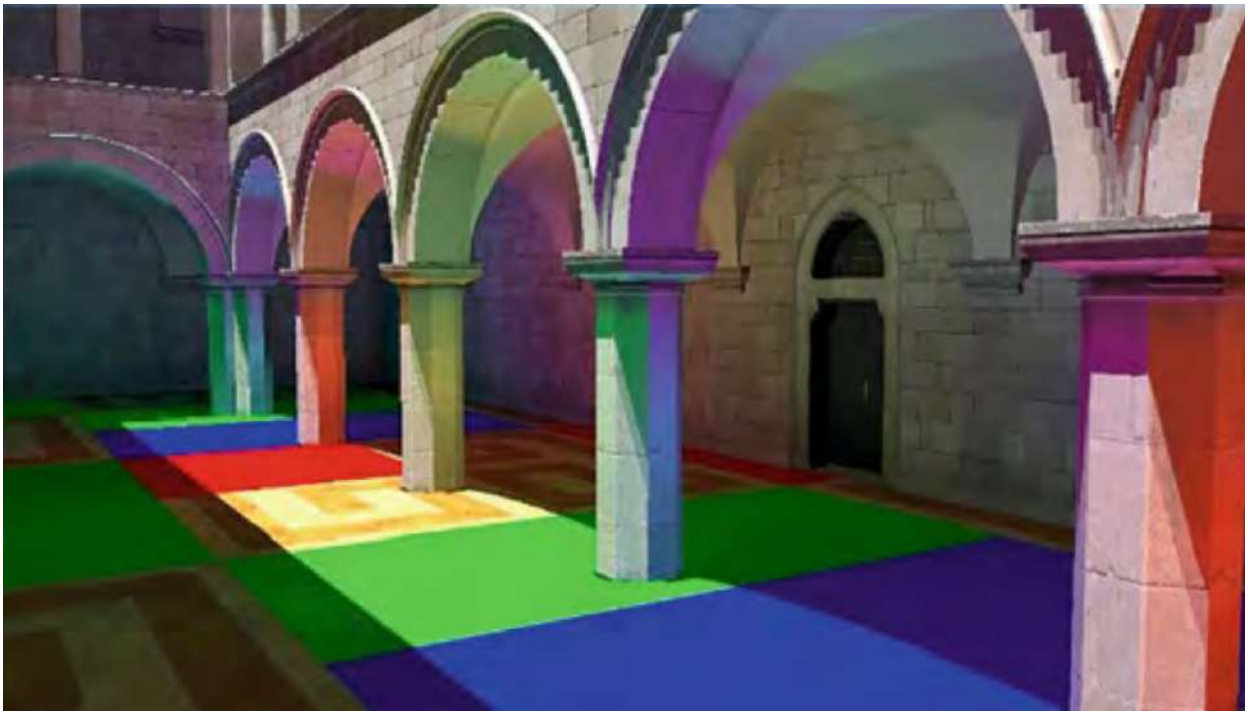


Рисунок 3.8 — Освещение, посчитанное предложенным методом матрицы нескольких отражений.

Рисунок 3.9 — Эталонное изображение, посчитанное трассировкой путей.

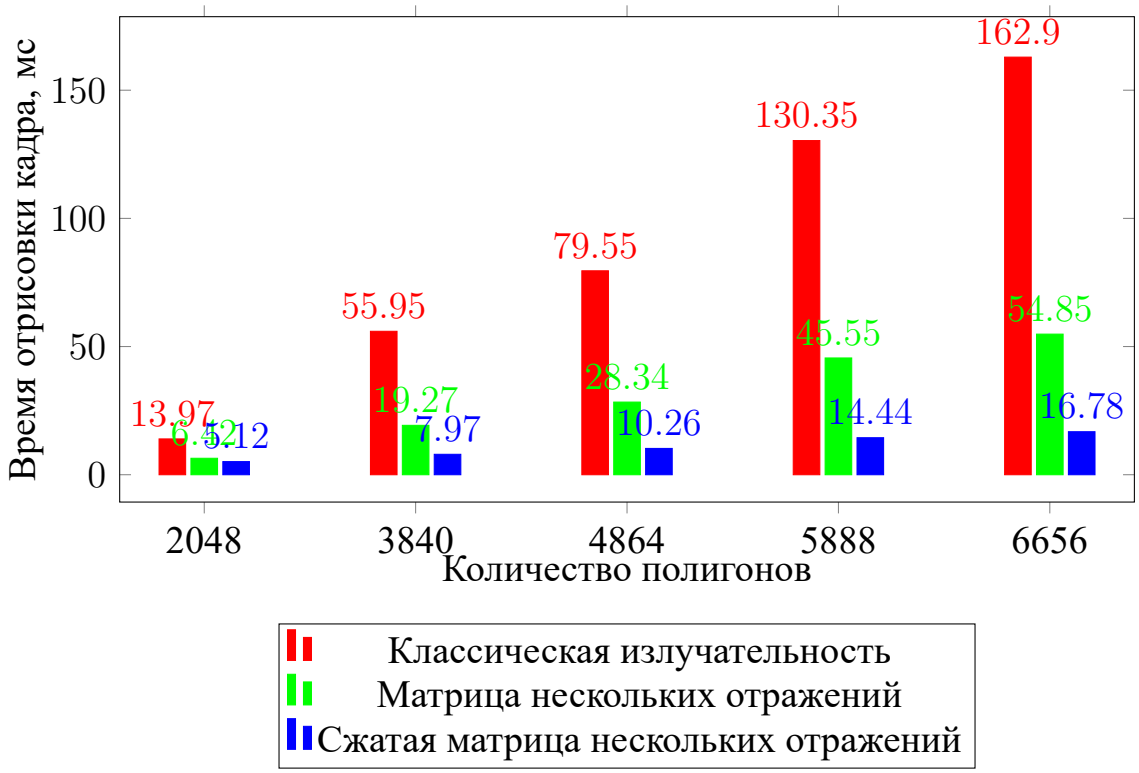


Рисунок 3.10 — Сравнение времени работы предложенного метода с классическим методом излучательности (ускорение до 10 раз).

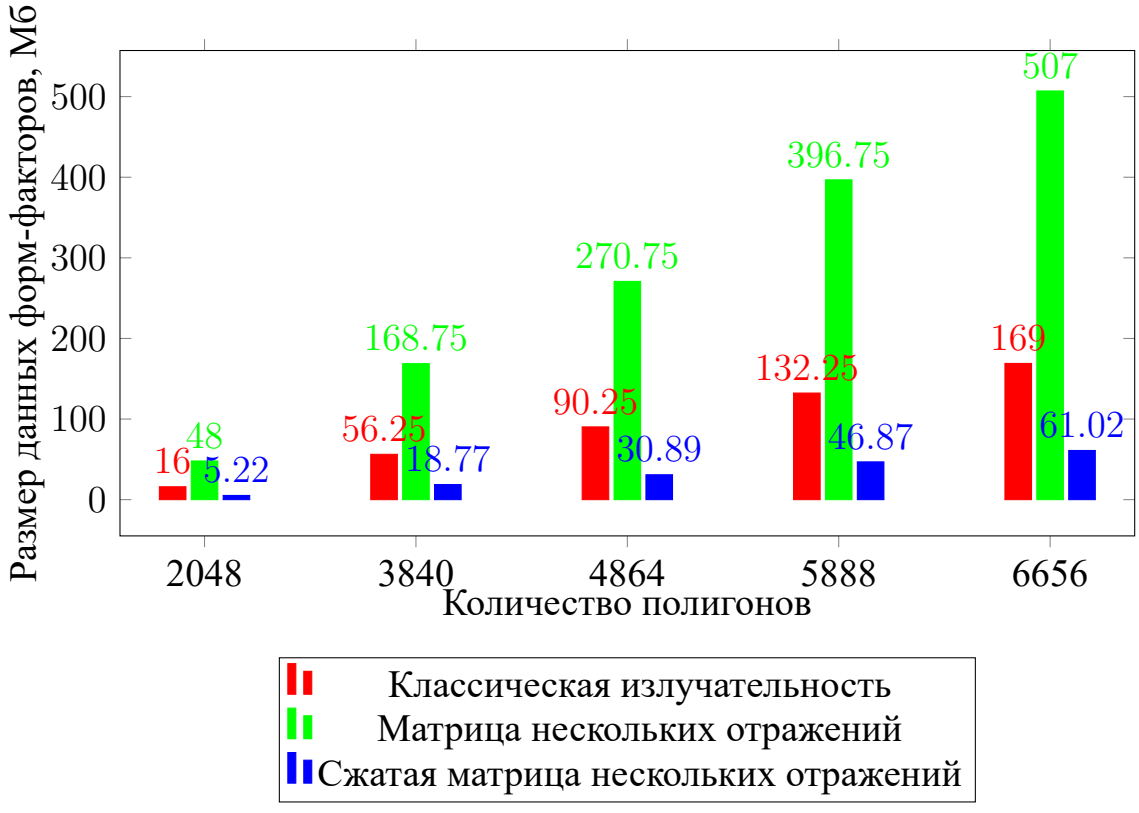


Рисунок 3.11 — Сравнение размера данных форм-факторов предложенного метода с классическим методом излучательности (до 3 раз меньше памяти).

Глава 4. Метод пересчета локальной матрицы для сцен большого масштаба

На больших 3D-сценах, например для приложений с открытым миром, даже при использовании упрощенной прокси геометрии в расчёте освещения будет участвовать большое количество полигонов и размеры матрицы форм-факторов не будут вмещаться в память современных GPU. Отчасти, эта проблема решается разбиением сцены на пересекающиеся кластеры полигонов [72], при этом каждый кластер имеет свою матрицу форм-факторов. Разбиение на кластеры производится в автоматическом режиме. Такой подход имеет недостаток — возможен резкий перепад освещения на границе кластеров. Чтобы избежать такой проблемы, для некоторых сцен может потребоваться большое количество кластеров, причём глобальное освещение необходимо вычислять для каждого из них.

В качестве решения этой проблемы, был разработан метод эффективного пересчёта матрицы нескольких отражений вокруг камеры (локальной матрицы), при её движении [A.2]. В вычислении освещения на каждом кадре берется некоторое подмножество из M полигонов, которые ближе всего расположены к камере. В памяти GPU располагается матрица форм-факторов, учитывающая несколько отражений, размером $M \times M$. Освещение вычисляется только для этик полигонов, как наиболее важных для текущего кадра. При движении камеры список полигонов, наиболее близких к ней, обновляется 4.1. Соответственно этому меняется и матрица форм-факторов для вычисления освещения только актуальных полигонов.

Как не сложно заметить, часть полигонов, находящихся за камерой или закрытых от наблюдателя другими объектами, никак не влияют на итоговое изображение и может быть исключена из вычислений, так как переотражение освещения этими полигонами уже учтено в матрице форм-факторов, поддерживающей несколько отражений. С другой стороны, на освещение влияют только полигоны, на которые попадает первичное освещение. Таким образом, в момент вычисления освещения достаточно определить множество видимых полигонов P_{Vis} и множество освещенных полигонов P_{Lit} и использовать в вычислениях подматрицу матрицы $F_{multibounce}$ построенную на столбцах с индексами из множества P_{Lit} и строках из множества P_{Vis} 4.2. Использование только части данных позволяет существенно сократить вычисления, однако, разница в скорости выполнения

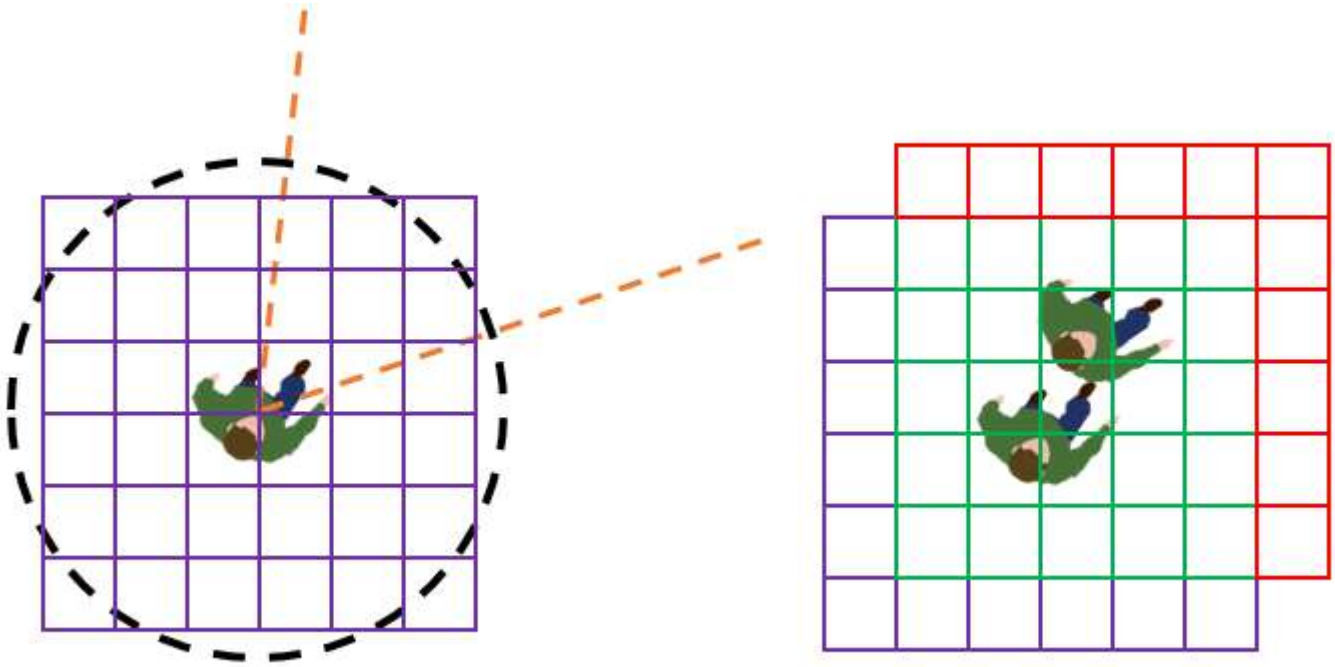


Рисунок 4.1 — Изменение множества актуальных полигонов при движении камеры (наблюдателя).

зависит от конкретного кадра, позиции камеры, позиций и направлений источников света.

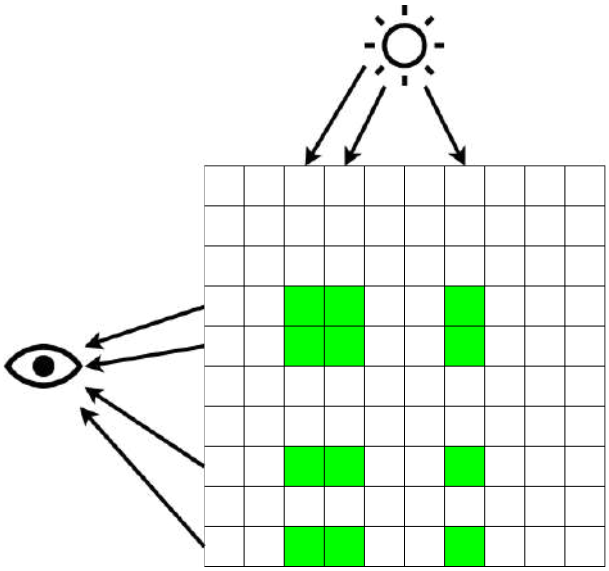


Рисунок 4.2 — Выбор подматрицы, необходимой для вычисления актуального освещения.

Основную сложность в предложенном подходе представляет схема добавления и удаления полигонов при движении камеры, так как требуется эффективная работа с матрицей форм-факторов. На диске предлагается хранить обычную

матрицу форм-факторов, так как она более разрежена, что может привести к существенной экономии. Другая причина использования обычных форм-факторов заключается в том, что вычисление матричного полинома для больших сцен может быть неосуществимо на обычных компьютерах ввиду больших затрат памяти, то есть для больших сцен этот этап предобработки сцены не может быть выполнен.

Операция удаления полигонов и соответствующих форм-факторов является достаточно простой. Соответствующие строки и столбцы просто не учитываются в вычислениях и не попадают во множества P_{Vis} и P_{Lit}. При этом занятое ими место в памяти будет переиспользовано при добавлении новых полигонов.

При добавлении строки f^r (см. рис. 4.4) и столбца f^c (см. рис. 4.3) матрицы форм-факторов для нового полигона, нужно из обычных форм-факторов, получить данные, учитывающие несколько отражений: g^r и g^c.Эти данные вычисляются следующим образом. Вначале вычисляется

$$double\_reflection = (f^r, C \circ f^c) = \sum_{i=1}^{M} f_i^r f_i^c C_i,$$

где C — вектор цветов полигонов, учитываемых в текущей матрице форм-факторов.

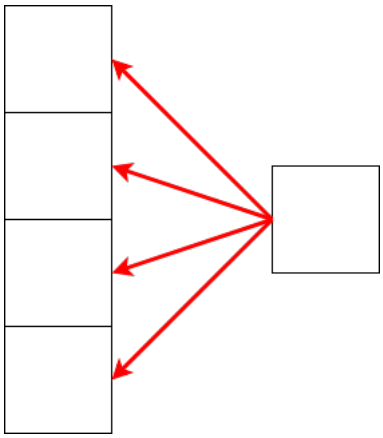


Рисунок 4.3 — Отражение света, описываемое значениями f^c.

$double\_reflection$ — это число показывающее, какая часть света покинув новый полигон, отразится от остальных полигонов и вернётся обратно (см. рис. 4.5).

Далее, вычисляются векторы форм-факторов учитывающие первое отражение света и свет, который отразился от нового полигона, переотразился от

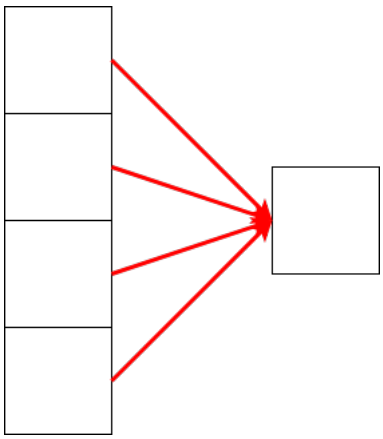


Рисунок 4.4 — Отражение света, описываемое значениями f^r.

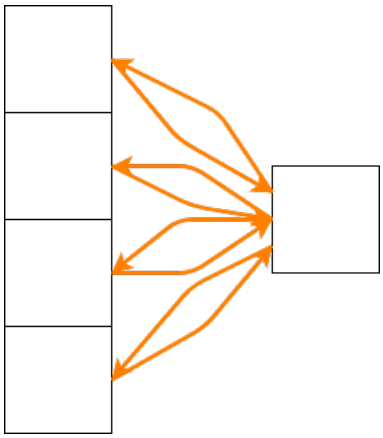


Рисунок 4.5 — Двойное отражение света, описываемое значением $double\_reflection$.

остальных полигонов, вернулся на новый полигон и отразился на один из оставшихся второй раз (см. рис. 4.6, 4.7):

$$f^{r\prime} = f^r + f^r \cdot c \cdot double\_reflection,$$
$$f^{c\prime} = f^c + f^c \cdot c \cdot double\_reflection,$$

где c — цвет нового полигона.

После этого можно вычислить векторы, которые помимо информации об однократном и трёхкратном отражении от нового полигона, будут читывать последующие переотражения света остальными полигонами 4.8:

$$g^r = f^{r\prime} + (C \circ f^{r\prime})^T F_{multibounce},$$
$$g^c = f^{c\prime} + F_{multibounce}(C \circ f^{c\prime}).$$

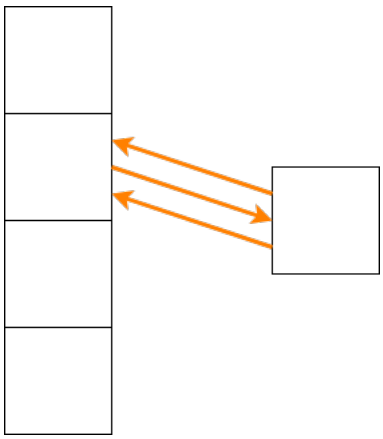


Рисунок 4.6 — Отражение света, описываемое значениями $f^{c\prime}$.

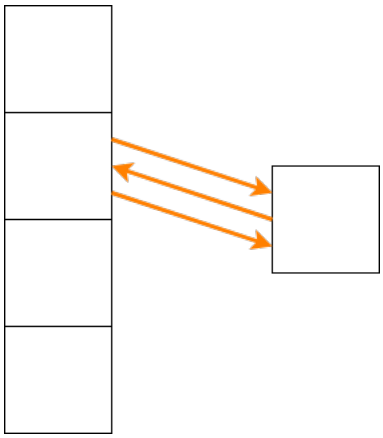


Рисунок 4.7 — Отражение света, описываемое значениями $f^{r\prime}$.

Полученные векторы будут учитывать не менее двух переотражений и с многократным добавлением и удалением полигонов, количество учтенных отражений будет увеличиваться, тем самым повышая детализацию полученного освещения.

Последним этапом в добавлении полигона является изменение значений старой матрицы, чтобы они учитывали новые данные:

$$F'_{multibounce} = F_{multibounce} + (C \circ g^c)g^{rT}.$$

Таким образом, выполнив несколько операций с суммарной сложностью $O(M^2)$, мы получим новую матрицу форм-факторов, учитывающую несколько отражений для актуального набора полигонов. При этом можно выполнять обновление форм-факторов в течение нескольких кадров, так как множество полигонов вокруг камеры меняется не каждый кадр. В этом случае, вычисление вторичного освещения с обновлением матрицы будет требовать вычислений не больше, чем вычисление освещения для двух отражений, а для большинства кадров будет

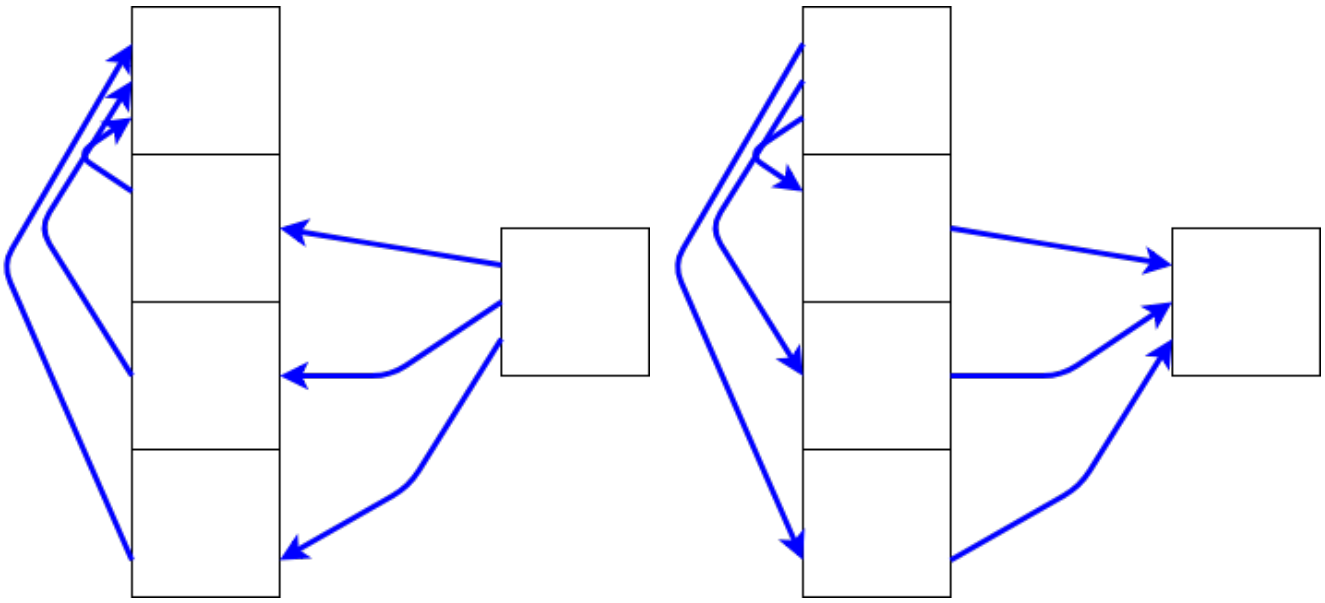


Рисунок 4.8 — Схема отражения света с учетом переотражений между старыми полигонами.

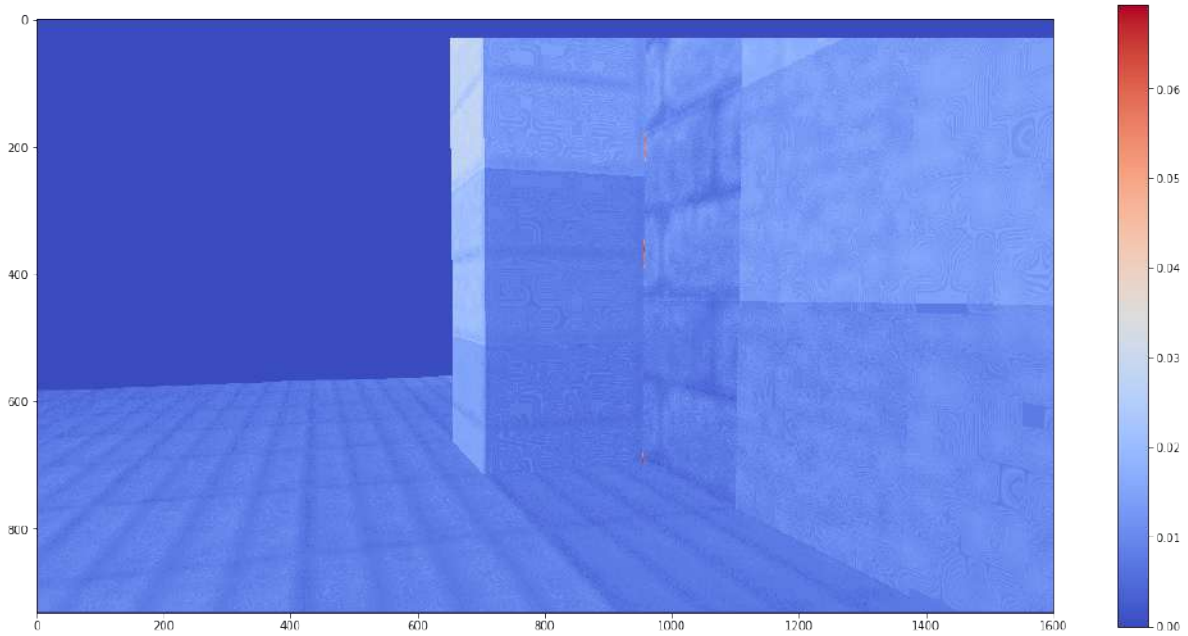
выигрыш по времени при том, что количество учтенных переотражений у предложенного метода больше двух.

Предложенный метод даёт схожие результаты с методом излучательности, которые близки к эталонному изображению по метрике MSE 4.9. Различие в освещении может проявляться для полигонов, находящихся на удалении от камеры. Данная проблема решается поддержкой нескольких каскадов, в каждом из которых будут полигоны с разной детализацией, но конкретное решение зависит от реализации системы уровней детализации объектов в приложении.

Скорость работы алгоритма существенно превосходит обычный метод излучательности на сценах большого размера 4.12, а так же позволяет экономить видеопамять и более эффективно управлять её потреблением, так как не требуется реализовывать загрузку и выгрузку форм-факторов для целых кластеров полигонов.

а) Метод излучательности (SSIM: 0.87).

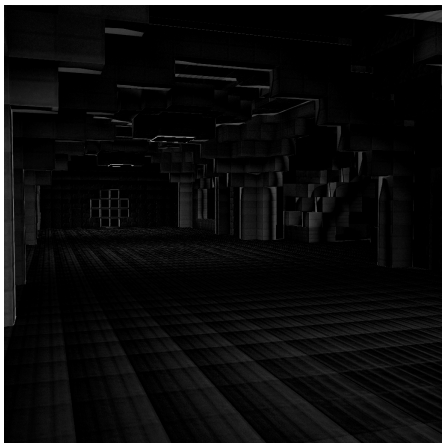
б) Предложенный метод (SSIM: 0.87).



в) Разница между предложенным методом и методом излучательности.

Рисунок 4.9 — Сравнение освещения, полученного методом излучательности и предложенным методом.

а) Метод излучательности (SSIM: 0.846).

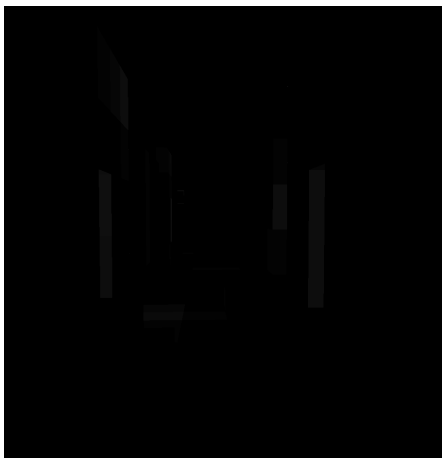


б) Разница между предложенным методом и методом излучательности. Яркость увеличена в 5 раз.

в) Предложенный метод (SSIM: 0.846).

Рисунок 4.10 — Сравнение освещения, полученного методом излучательности и предложенным методом.

а) Метод излучательности (SSIM: 0.835).



б) Разница между предложенным методом и методом излучательности. Яркость увеличена в 5 раз.

в) Предложенный метод (SSIM: 0.831).

Рисунок 4.11 — Сравнение освещения, полученного методом излучательности и предложенным методом.

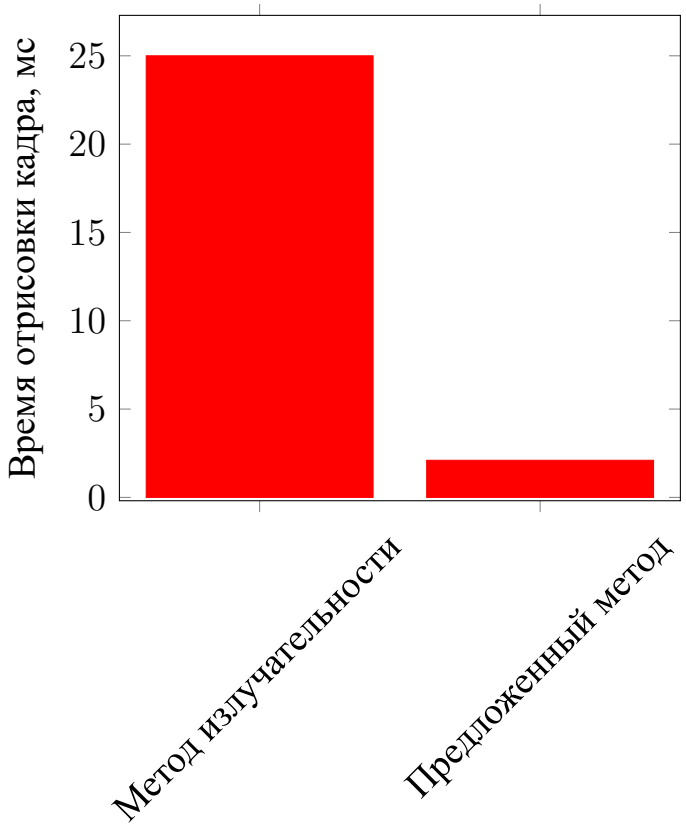


Рисунок 4.12 — Сравнение скорости работы метода излучательности с матрицей нескольких отражений и предложенного метода, использующего локальную матрицу.

Глава 5. Метод виртуальных площадок

Метод локальной матрицы решает только часть проблемы квадратичной сложности метода излучательности, позволяя игнорировать полигоны, находящиеся на достаточном удалении от камеры. Однако, 3D модели с высокой детализацией могут состоять из десятков миллионов полигонов, что делает невозможным использование какого-либо из предложенных методов для таких моделей напрямую. Поэтому требуется наличие упрощенной прокси-геометрии, которую можно использовать для вычисления освещения с последующим наложением на исходные модели.

Для решения этой задачи предлагается метод виртуальных площадок [А.3]. Он работает полностью в автоматическом режиме, не требует хранить прокси-геометрию на диске и минимизирует ошибку связанную с уменьшением данных при упрощении моделей.

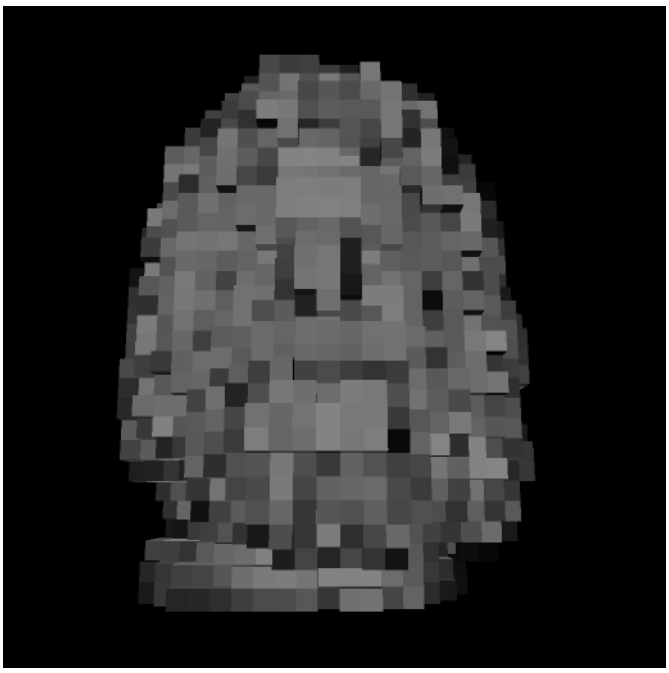
Суть метода заключается в том, что прокси-геометрия используется для вычисления форм-факторов и начального освещения, но эти данные избыточны и для вычисления освещения достаточно иметь матрицу форм-факторов, способ инициализировать вектор E и способ наложить полученное освещение на модель.

Для построения виртуальных площадок и вычисления форм-факторов для них выполняются следующие шаги 5.1:

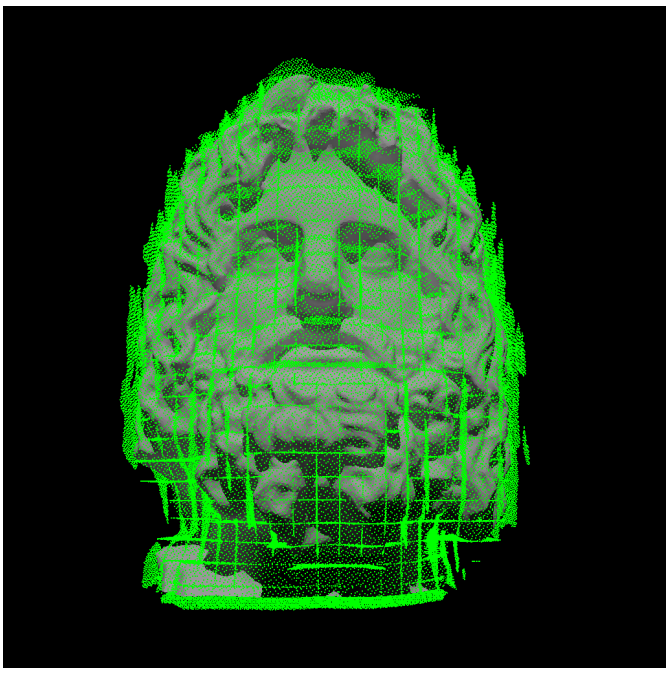
1. Разбиение пространства сцены на воксели (см. рис. 5.1а).
2. Генерация равномерно распределенных точек на границах вокселей (см. рис. 5.1б).
3. Трассировка лучей из точек внутрь вокселей.
4. Получение набора точек на исходной геометрии (см. рис. 5.1в).
5. Вычисление форм-факторов для пар точек.
6. Инициализация виртуальных площадок данными точек исходной геометрии.
7. Иерархическая кластеризация виртуальных площадок в каждом вокселе до достижения максимально поддерживаемого количества площадок.

5.1 Разбиение пространства на воксели

Размер вокселя является настройкой алгоритма, которая позволяет контролировать баланс между скоростью и точностью вычислений. Ограничений на



а) Вокселизация геометрии



б) Генерация точек на поверхностях вокселей

в) Точки на исходной геометрии

Рисунок 5.1 — Этапы построения виртуальных площадок.

размер вокселя или равенство его рёбер нет, что позволяет более гибко настроить алгоритм для моделей и сцен, размер которых по одной из осей существенно превышает размер по другим.

5.2 Генерация точек

Воксели, не содержащие геометрию сцены отбрасываются из дальнейшего рассмотрения. На оставшихся вокселях, на каждой из граней выбираются точки

по распределению Hammersley [75]. Количество точек ещё одна настройка, которая влияет на точность и скорость предобработки сцены, но не имеет значения для скорости вычисления освещения. Таким образом, пользователь может выбирать небольшое количество точек для сцены в процессе работы над ней и увеличивать этот параметр при финальной предобработке.

5.3 Трассировка лучей в вокселях

Для каждой точки производится операция трассировки луча внутрь вокселя в направлении, перпендикулярном грани вокселя. Трассировка осуществляется только в пределах вокселя, то есть если внутри вокселя луч не пересёк исходную сцену, то луч и соответствующая ему точка исключаются из дальнейшего рассмотрения. Если луч пересёк исходную геометрию внутри вокселя, то полученная точка пересечения будет использована на следующем этапе для генерации начальной виртуальной площадки. Ограничений на использование каких-либо фреймворков или инструментов для трассировки нет, возможна как трассировка на CPU, так и на GPU.

5.4 Генерация виртуальных площадок

Полученные трассировкой точки исходной модели будут использоваться как инициализирующие данные для виртуальных площадок. Так как виртуальные площадки имеют форм-факторы, для вычисления которых требуется площадь, то и точки, используемые для создания первичных виртуальных площадок, должны соответствовать некоторой площади поверхности модели.

Для вычисления площади, соответствующей точке, используется следующая формула:

$$s_i = \frac{S_j}{|samples_j|},$$

где i — номер точки, j — номер полигона, которому принадлежит точка, S_j — площадь полигона, $samples_j$ — множество точек, принадлежащих полигону.

Как несложно заметить,

$$s_i = s_k, \forall i, k \in samples_j.$$

Цвет виртуальной площадки выбирается из текстуры, которой текстурируется полигон. Таким же образом, получается нормаль виртуальной площадки. Итого, площадка имеет цвет, нормаль, позицию и площадь.

5.5 Вычисление форм-факторов

Как правило, форм-факторы заданные формулой (2.1) вычисляются при помощи метода Монте-Карло. В случае выборки единственной точки, формула упрощается и суммирование не производится, так как участвуют только две точки:

$$F_{ij} = s_j \frac{\cos(\vec{r}, \vec{n_i}) \cos(-\vec{r}, \vec{n_j})}{\pi \left|\vec{r}\right|^2} V(p_i, p_j), \tag{5.1}$$

где $\vec{r} = p_j - p_i$, $\vec{n_i}$ и $\vec{n_j}$ — нормали в точках, p_i и p_j — позиции точек.

5.6 Инициализация виртуальных площадок

Виртуальные площадки имеют следующие данные:

- Нормаль. Она используется для переноса первичного освещения с исходной геометрии на виртуальные площадки и обратного переноса вторичного освещения с виртуальных площадок на исходную геометрию.
- Форм-факторы, которые нужны для непосредственного вычисления глобального освещения.
- Цвет для поддержки нескольких отражений и возможности вычислять матрицу форм-факторов с несколькими отражениями.

После вычисления форм-факторов, точки имеют все эти данные, так что ими можно инициализировать виртуальные площадки. На этом шаге расположение точек исходной модели и виртуальных площадок в пространстве не используется (за исключением информации о локализации площадки внутри конкретного вокселя), поэтому виртуальные площадки не имеют представления в пространстве и используются как абстрактный набор данных.

5.7 Кластеризация виртуальных площадок

Кластеризация виртуальных площадок осуществляется внутри каждого вокселя. Кластер из виртуальных площадок заменяется на новую виртуальную площадку, что позволяет уменьшить количество данных, с некоторыми потерями в точности.

В данном методе предлагается использовать жадный метод слияния виртуальных площадок, который создаёт дерево с агрегированными виртуальными площадками в промежуточных узлах и начальными площадками в листьях. Поэтому процесс можно рассматривать как иерархическую кластеризацию.

На каждом шаге алгоритма, выбираются две площадки, наиболее схожие по метрике:

$$M(i,j) = \left(1 + \left|c_i - c_j\right|_2\right)\left(2 - (\vec{n_i}, \vec{n_j})\right)\left(1 + \sqrt{\sum_{k=1}^{N}(F_{ik} - F_{jk})^2 + (F_{ki} - Fkj)^2}\right)$$

Для выбранных площадок производится процесс слияния. При этом значения площадки пересчитываются следующим образом:

$$\begin{aligned} F_{new,h} &= \frac{s_i F_{ih} + s_j F_{jh}}{s_i + s_j} \\ F_{h,new} &= F_{hi} + F_{hj} \\ c_{new} &= \frac{s_i c_i + s_j c_j}{s_i + s_j} \\ \vec{n_{new}} &= normalize\left(\frac{s_i \vec{n_i} + s_j \vec{n_j}}{s_i + s_j}\right) \\ s_{new} &= s_i + s_j \end{aligned} \tag{5.2}$$

Процесс слияния продолжается пока не будет достигнуто целевое количество виртуальных площадок.

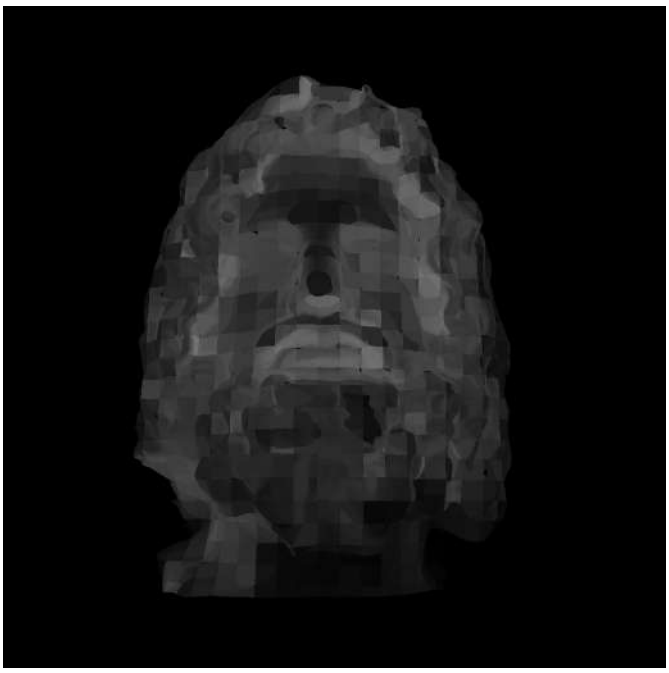
В итоге, пользователю нужно хранить векторы нормалей полученных площадок, их цвета и форм-факторы для каждого вокселя.

Так как количество точек исходной геометрии может быть велико для больших сцен, матрица форм-факторов будет требовать больших затрат памяти. Уменьшить потребление памяти можно объединив вычисление форм-факторов с процессом кластеризации. Вначале вычисляется подматрица форм-факторов со строками для точек модели внутри одного вокселя и всеми столбцами. После этого можно запустить процесс слияния площадок для первого вокселя, тем самым уменьшив размер матрицы форм-факторов. Продолжая такую обработку по одному вокселю, можно добиться меньшего пикового потребления памяти алгоритмом.

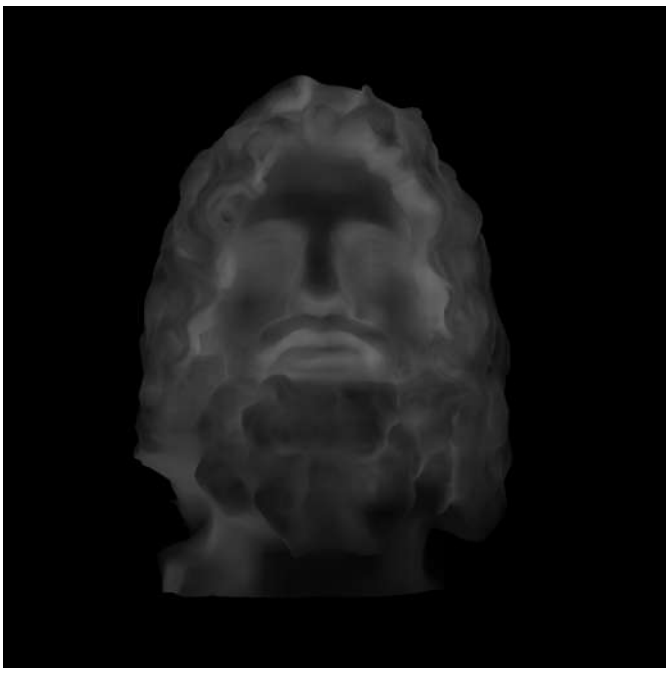
5.8 Использование виртуальных площадок

Для вычисления начального освещения внутри каждого вокселя выбирается несколько точек исходной геометрии. Используя карту теней, определяется степень освещенности этих точек e_i. Тогда первичное освещение для площадки j внутри вокселя вычисляется как:

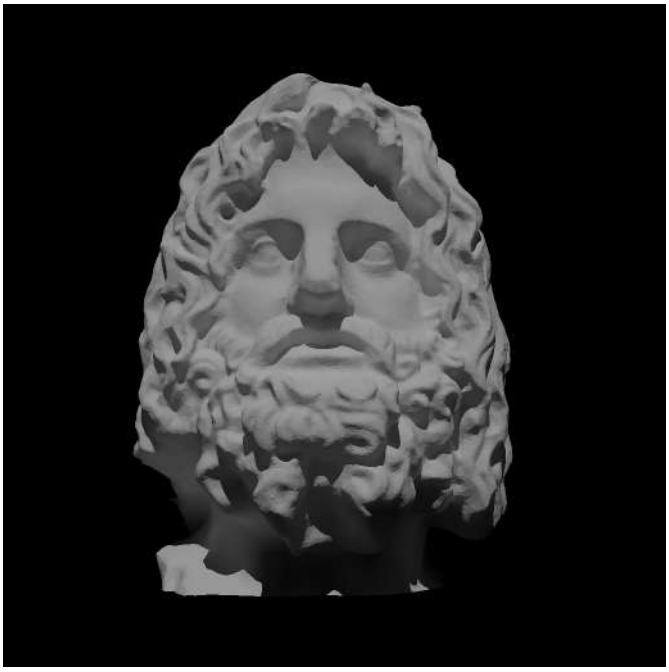
$$E_j = \sum_{i \in samples} e_i \max(0, (\vec{N_j}, \vec{n_i}))$$



а) Вычисленное вторичное освещение



б) Трилинейная фильтрация для плавного вторичного освещения



в) Финальное освещение

Рисунок 5.2 — Этапы использования виртуальных площадок.

, где $\vec{N}_j$ — нормаль виртуальной площадки, $\vec{n_i}$ — нормально точки, $samples$ — множество точек внутри вокселя.

Аналогичные формулы используются при наложении вычисленного глобального освещения на исходную геометрию.

$$lighting = \sum_{j \in patches} B_j \max(0, (\vec{N}_j, \vec{n})), \tag{5.3}$$

где $patches$ — множество площадок внутри вокселя, B_j — излучательность, вычисленная для площадки, $\vec{n}$ — нормально обрабатываемой точки.

Так как площадки в соседних вокселях имеют разные данные об освещении, предлагается использовать трилинейную фильтрацию при выборке освещения из воксельной сетки. Проблему в этом случае представляют пустые воксели, или воксели не содержащие данных освещения для какой-то ориентации полигона. Чтобы решить это затруднение, предлагается игнорировать воксели, которые не вносят вклад в освещение и итоговая формула фильтрации получается следующей:

$$lighting = \frac{\sum_{i \in V} lighting_i w_i}{\sum_{i \in V} w_i},$$

где V — множество не пустых вокселей из блока 2x2 используемого для фильтрации, w_i — веса трилинейной фильтрации, $lighting_i$ — освещение пикселя, посчитанное из данных виртуальных площадок вокселя i по формуле (5.3).

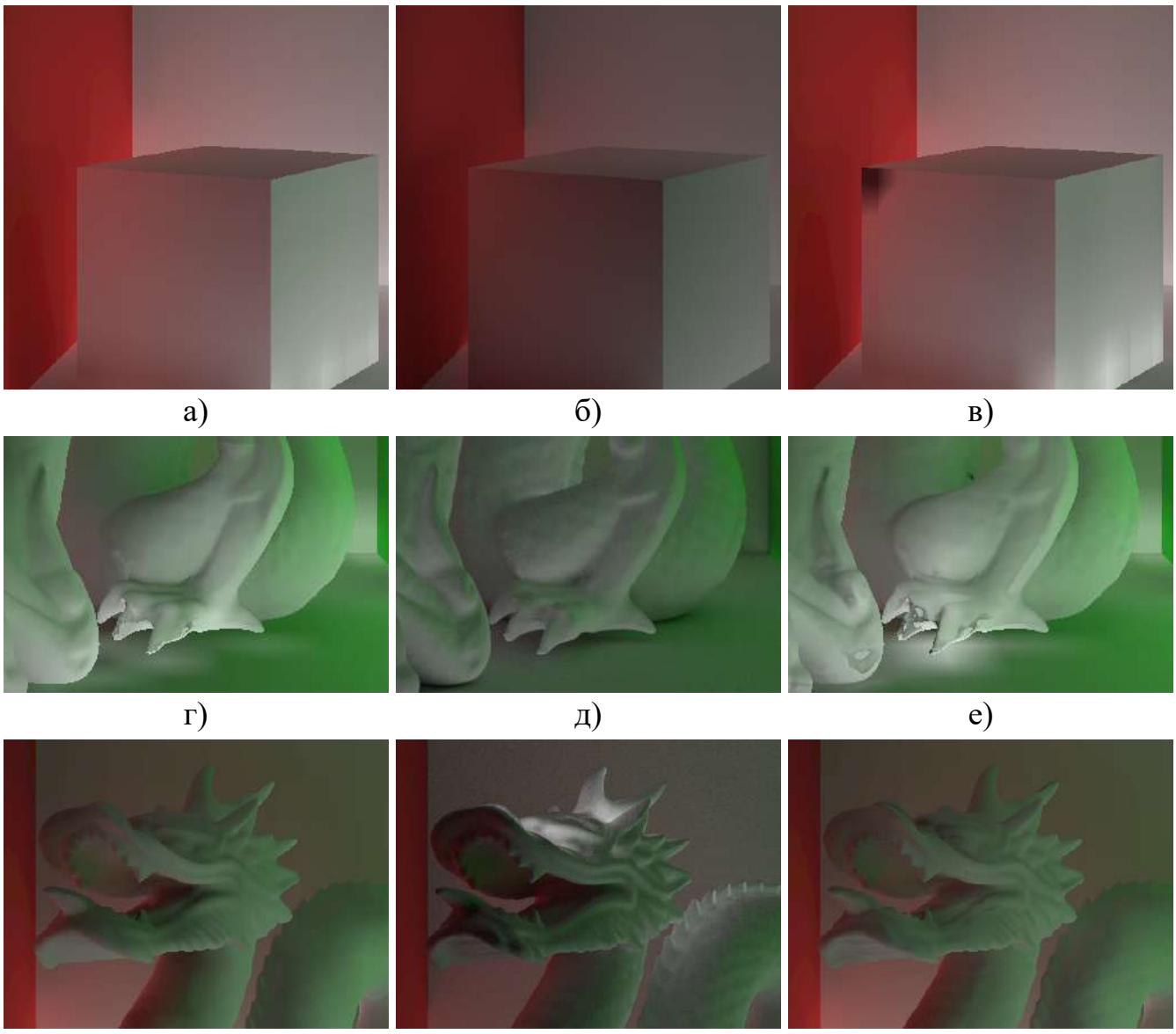
Предложенный метод упрощения геометрии позволяет сократить количество полигонов, для которых требуется вычислить освещение в алгоритмах работающих с излучательностью, создавая виртуальную прокси-геометрию в автоматическом режиме. При этом, полученное представление сцены создаёт меньшее количество артефактов, чем другие методы автоматической генерации площадок для метода излучательности, в частности вокселизация сцены 5.3. Точность представления освещения без учета артефактов также выше по отношению к эталонному изображению 5.4.

5.9 Виртуальные площадки, выровненные по осям координат

Метод виртуальных площадок имеет два недостатка связанные с ориентацией площадок в пространстве: необходимо хранить нормали виртуальных площадок, нормали всех площадок внутри вокселя могут не покрывать все возможные направления так, что $\sum_{j \in patches} \max(0, (\vec{N}_j, \vec{n}))$ всегда будет равно 0 для некоторых полигонов.

Решить вторую проблему можно добавлением дополнительных виртуальных площадок в воксель, но это потребует больших затрат памяти и времени вычисления освещения.

Для устранения описанных выше недостатков предлагается метод виртуальных площадок, выровненных по осям координат. Алгоритм повторяет обычный



а) б) в)

г) д) е)

ж) з) и)

Левая колонка — предложенный метод. Средняя колонка — эталонные изображения, полученные трассировкой путей. Правая колонка — упрощенная геометрия, полученная вокселизацией.

Рисунок 5.3 — Сравнение результатов метода.

метод виртуальных площадок вплоть до вычисления форм-факторов для точек. Далее выполняются следующие шаги:

1. Генерация виртуальных площадок, ориентированных по осям координат.
2. Слияние виртуальных площадок.

Из одной точки модели генерируется от 1 до 3 виртуальных площадок. Значения, описывающие виртуальную площадку, вычисляются следующим образом. Сначала вычисляется вес данной точки i в каждом из направлений осей координат (положительном и отрицательном):

$$
\begin{aligned}
weight_{axis} &= \max(0, (\vec{n_i}, \vec{e_{axis}})), axis = \overline{1,6}, \\
e_1 &= (1,0,0), \\
e_2 &= (-1,0,0), \\
e_3 &= (0,1,0), \\
e_4 &= (0,-1,0), \\
e_5 &= (0,0,1), \\
e_6 &= (0,0,-1)
\end{aligned}
$$

Далее, вес используется для вычисления данных виртуальной площадки h:

$$
\begin{aligned}
F_{hj} &= weight_h F_{ij}, \forall j = \overline{1,M}, \\
F_{jh} &= weight_h F_{ji}, \forall j = \overline{1,M}, \\
c_h &= weight_h c_i, \\
s_h &= weight_h s_i
\end{aligned}
$$

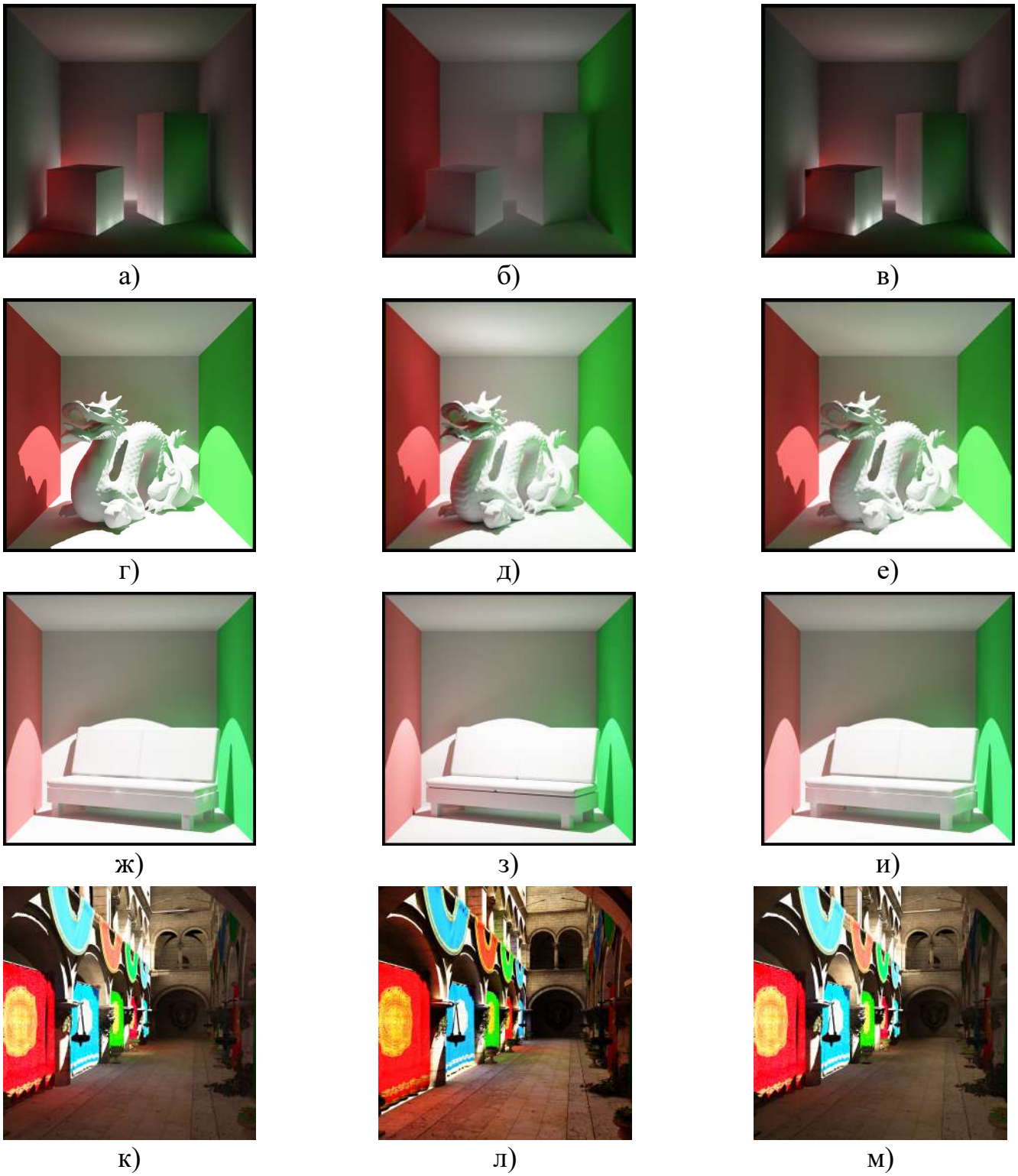
Если вес для некоторого направления равен 0, то виртуальная площадка не создаётся.

Для полученных виртуальных площадок осуществляется процесс слияния по формулам (5.2). При этом никаких операций с нормалями не требуется, так как объединяются только площадки с одинаковым направлением. В итоге, получается не более 6 площадок внутри одного вокселя, направления которых известны и не требуют сохранения в отдельный буфер.

Так как нет ограничений на очередность слияния площадок, формулы (5.2) и сами алгоритмы пересчёта могут быть упрощены:

$$
\begin{aligned}
F_{axis,h} &= \frac{\sum_{i \in patches} s_i F_{ih}}{\sum_{i \in patches} s_i} \\
F_{h,axis} &= \sum_{i \in patches} s_i F_{hi} \\
c_{axis} &= \frac{\sum_{i \in patches} s_i c_i}{\sum_{i \in patches} s_i} \\
s_{axis} &= \sum_{i \in patches} s_i
\end{aligned}
$$

Вычисление освещения и использование виртуальных площадок в данном методе остаётся тем же, за исключением того, что количество площадок в одном вокселе всегда ограничено 6 и код, связанный с вычислением веса для каждой площадки упрощается, так как их направления известны заранее.



а) б) в)

г) д) е)

ж) з) и)

к) л) м)

Левая колонка — предложенный метод. Средняя колонка — эталонные изображения, полученные трассировкой путей. Правая колонка — упрощенная геометрия, полученная вокселизацией.

Рисунок 5.4 — Сравнение результатов метода.

Глава 6. Метод темпоральной излучательности

Несмотря на существенное ускорение, которое даёт использование локальных данных форм-факторов и метод виртуальных площадок, вычисление глобального освещения всё ещё имеет квадратичную зависимость от количества полигонов (площадок). Исходя из предположения, что глобальное освещение не меняется слишком резко (данное предположение актуально для множества приложений, где основным источником света является солнце), можно допустить некоторое отставание вторичного освещения от первичного. Таким образом, вычисления можно распределить между несколькими кадрами, как это делается для различных темпоральных алгоритмов, например темпоральный анти-алиасинг, GTAO, SSR [42; 55; 56].

Для стандартного представления форм-факторов возможны две стратегии распределения вычислений между кадрами:

1. Обновление освещения в кадре, когда процесс умножения матрицы на вектор завершится. В этом случае, на сложных сценах освещение может пересчитываться десятки кадров и разница между первичным освещением в момент начала вычислений и в момент окончания будет влиять на вторичное освещение настолько, что обновлённое освещение будет выглядеть как резкая смена изображения.
2. Частичное обновление освещения с использованием частей матрицы форм-факторов. Из-за неоднородности значений в матрице форм-факторов может сложиться ситуация, при которой на предыдущем кадре были использованы максимальные значения матрицы и освещение резко изменилось, а на текущем кадре будут использованы значения близкие к нулю. В этом случае изображение будет мерцать с яркими вспышками освещения на кадрах с обработкой максимальных значений матрице и затуханием на остальных кадрах.

Таким образом, для распределения вычислений между кадрами требуется иное представление матрицы форм-факторов.

6.1 Форм-факторы как математическое ожидание

Рассмотрим строку матрицы форм-факторов F_i. Для простоты будем считать, что это одноканальная матрица. В случае матрицы нескольких отражений, каналы обрабатываются по отдельности.

Вычисление освещения для полигона i выглядит следующим образом:

$$B_i = \sum_{j=1}^{N} F_{ij} E_j$$

Одно из известных свойств форм-факторов подробно описанное в [33]:

$$\sum_{j=1}^{N} F_{ij} \leqslant 1$$

Введём новый форм-фактор $F_{i0} = 1 - \sum_{j=1}^{N} F_{ij}$, который отвечает за свет пришедший из окружения сцены E_0, например из карты окружения типа skybox. Добавив этот новый форм-фактор в сумму, получим

$$\begin{gathered} \sum_{j=0}^{N} F_{ij} = 1 \\ \sum_{j=0}^{N} F_{ij} E_j = B_i \end{gathered} \tag{6.1}$$

Используя формулы (6.1), можно переформулировать вычисление освещения как нахождение математического ожидания случайной величины E с плотностью распределения F_i:

$$B_i = \mathbb{E}E$$

6.2 Выборка форм-факторов для нахождения математического ожидания освещения

Используя формулировку вычисления освещения как математического ожидания, можно привести вычисление освещения к темпоральному алгоритму [A.4]: на каждом кадре производится выборка случайной величины E с учетом плотности распределения (разной для разных полигонов), вектор освещения B обновляется с учетом новых данных. В этом случае, мы будем накапливать освещение между кадрами, при этом получится избежать резких перепадов освещения, так

как процесс выборки производится всё время и учитывается размер конкретных форм-факторов.

Для того чтобы реализовать выборка с плотностью вероятности F_i, предлагается использование алиас-таблиц [76]. Такие таблицы можно строить как при предобработке сцены, так и в процессе отрисовки. Алгоритмы эффективного построения алиас-таблиц на GPU разработаны [77] и их интеграция не представляет проблем.

Алиас-таблица — вектор из структур следующего вида: два индекса значений, которые могут быть результатами выборки, вероятность выбрать одно из этих значений при выборке. Процесс выборки выглядит следующим образом:

1. Случайным образом выбрать элемент алиас-таблицы.
2. Выбрать один из индексов в записи таблицы с учетом вероятности для этой записи.

С использованием алиас-таблиц и новой формулировки вычисления освещения, алгоритм становится следующим:

1. Для каждой строки выбираем несколько случайных индексов I_i выборкой из алиас-таблицы для этой строки.
2. Получаем усреднённое освещение для выбранных индексов: $B_i^{current} = \frac{\sum_{j \in I_i} E_j}{|I_i|}$.
3. Смешиваем освещение текущего кадра с предыдущим: $B_i = B_i^{previous} w + B_i^{current}(1 - w)$.
4. Обновляем освещение предыдущего кадра: $B_i^{previous} = B_i$.

Параметр w — выбирается в зависимости от конкретной сцены. Чем более разнородные вероятности для индексов в алиас-таблицах, тем ближе к 1 должно быть значение w.

6.3 Производительность темпоральной излучательности

Количество выборок из алиас-таблиц на каждом кадре является параметром алгоритма, который позволяет выбирать баланс между скоростью вычисления освещения и отставанием вторичного освещения от актуального первичного. На практике размер множества $I_i << N$, что позволяет выполнять вычисления близкие к алгоритмам сложности $O(N)$. Разница в скорости между темпоральной излучательностью и обычной излучательностью представлена в таблице 1.

Таблица 1 — Сравнение производительности метода темпоральной излучательности

Количество полигонов	Излучательность с умножением матрицы на вектор, мс	Темпоральная излучательность, мс
1736	0.34	0.02
5485	2.94	0.04
8786	7.34	0.06
10422	12.01	0.08

При значительном увеличении скорости вычисления, разница в точности по отношению к эталону достаточно низка, что видно из различных метрик представленных в таблице 2 для изображений 6.3.

Таблица 2 — Сравнение точности метода темпоральной излучательности по отношению к эталонному изображению, полученному трассировкой путей

Метрика	Излучательность с умножением матрицы на вектор	Темпоральная излучательность
MSE	0.06658	0.06651
PSNR	23.53	23.54
SSIM	0.8354	0.8323

При сравнении предложенного метода с методом RTXGI (таблица 3), темпоральная излучательность позволяет получить более близкое к эталону изображение по метрикам MSE, PSNR, SSIM при вдвое большей скорости вычислений.

Таблица 3 — Сравнение точноти метода темпоральной излучательности по отношению к эталонному изображению с RTXGI, полученному трассировкой путей

Метрика	Темпоральная излучательность	RTXGI
Время работы, мс	0.08	0.17
MSE	0.06651	0.10861
PSNR	23.54	19.28
SSIM	0.73	0.66

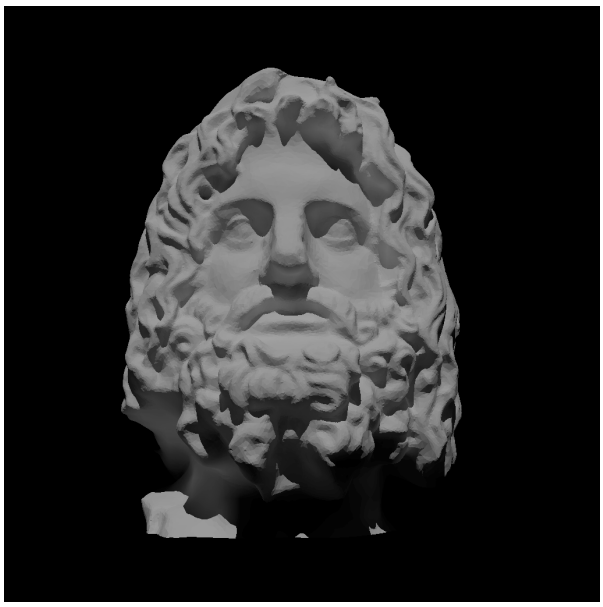
6.4 Отставание вторичного освещения

Основным недостатком предложенного метода является отставание вторичного освещения от первичного на несколько кадров, количество которых зависит от параметра w. Для источников света таких как солнце это не представляет проблемы. Для динамических источников, которые, как правило, имеют небольшой радиус действия из-за больших накладных расходов связанных с пересчётом первичного освещения, можно использовать отдельные небольшие локальные матрицы форм-факторов.

Отдельно стоит выделить отставание многократных отражений. Если алиас-таблицы реализованы для матрицы форм-факторов нескольких отражений, второе и последующие переотражения будут иметь такое же отставание, как и первое.

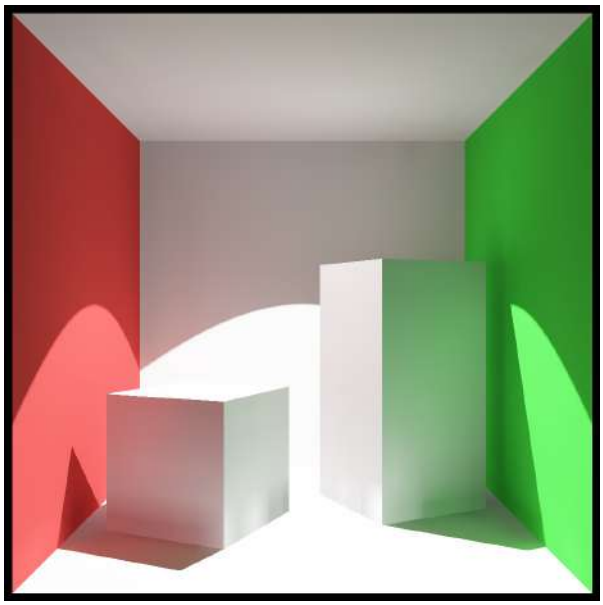
В случае, если алиас-таблицы построены для исходных форм-факторов, предлагается другой способ реализации многократных отражений. Освещение с предыдущего кадра $B^{previous}$ умножается на вектор C и прибавляется к вектору E текущего кадра. Таким образом, вычисляется величина $\mathbb{E}(E+B^{previous}\cdot C)$, которая учитывает многократные отражения, при этом отставание для каждого следующего отражения увеличивается на один кадр по сравнению с предыдущим.

а) Светотехническая сцена

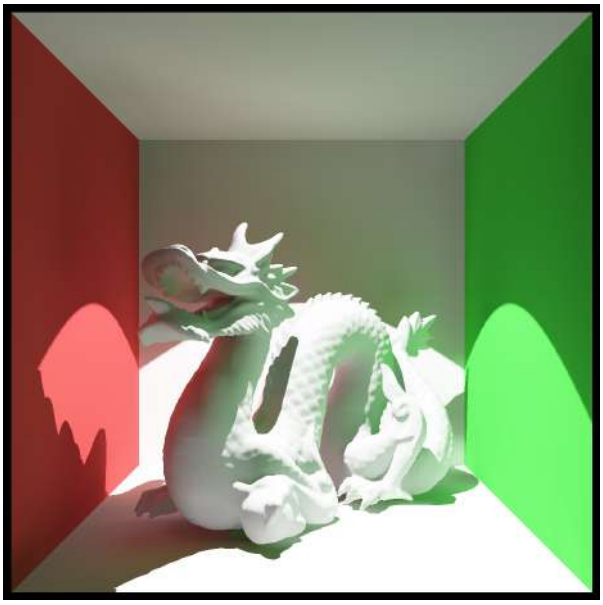


б) Serapis

в) Crytek Sponza*



г) Cornell Box Cubes*



д) Cornell Box Dragon

Рисунок 6.1 — Изображения, полученные методом темпоральной излучательности. (*Из демонстрационного приложения RTXGI от компании NVIDIA)

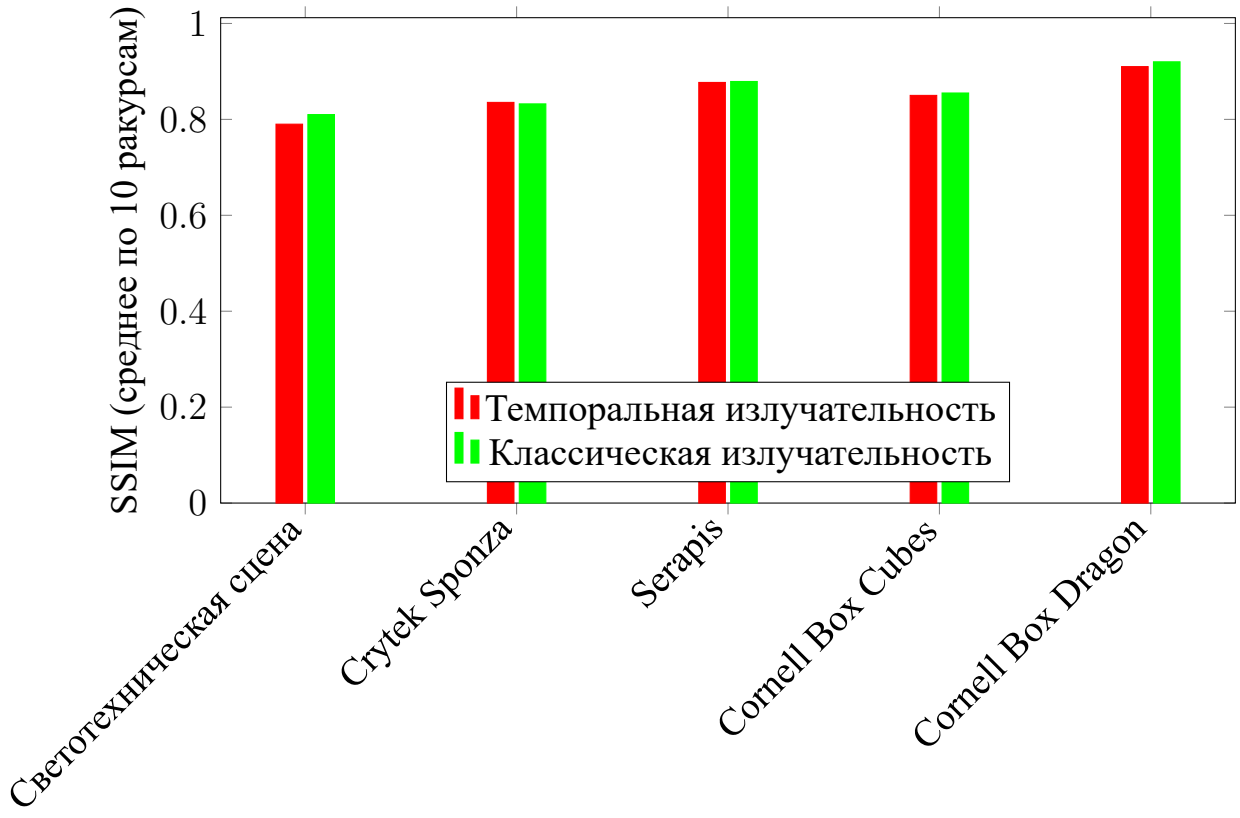
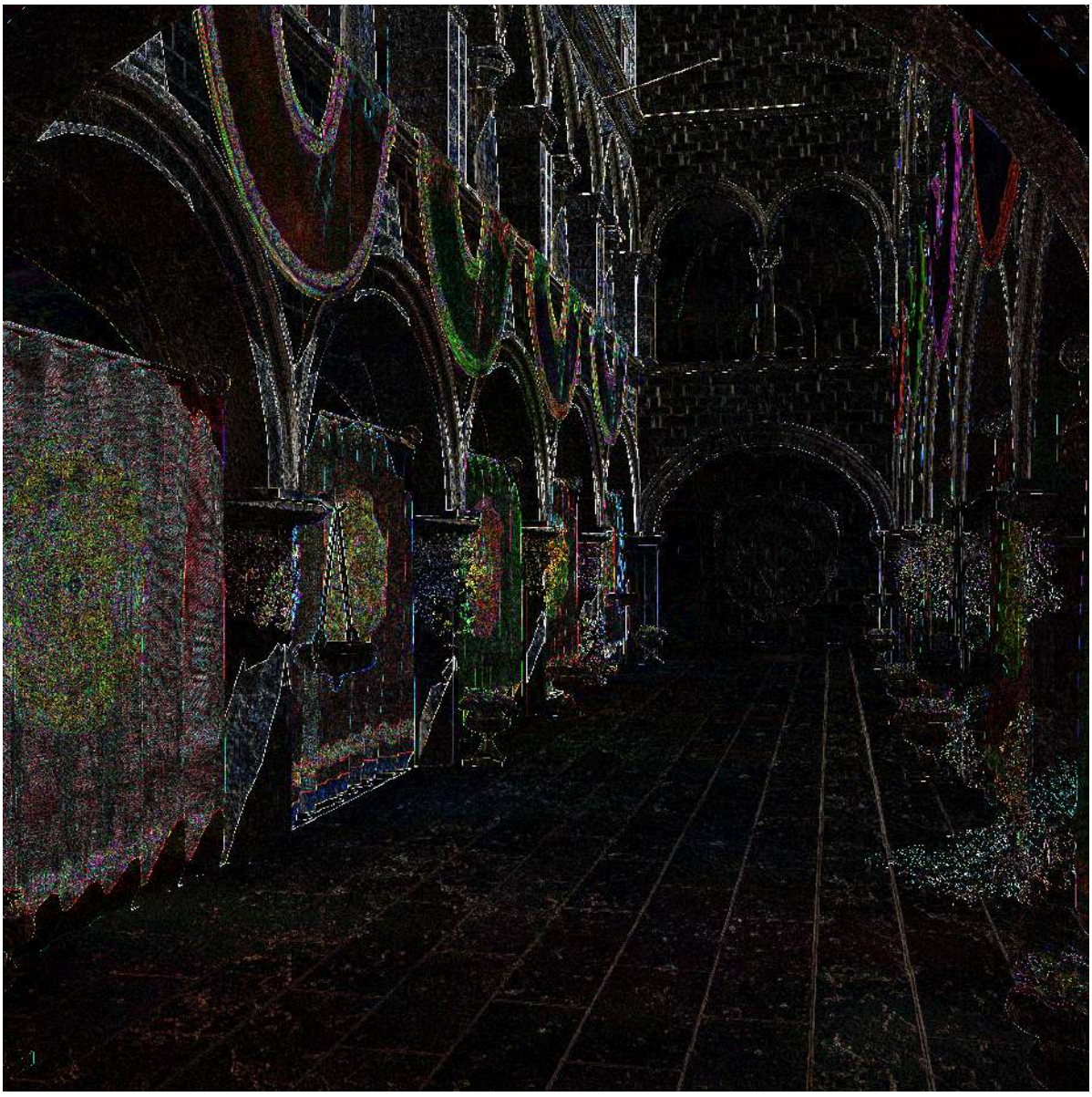


Рисунок 6.2 — Сравнение точности темпоральной излучательности и классической излучательности.

а) Метод излучательности

б) Метод темпоральной излучательности



в) Разница между методом темпоральной излучательности и классическим методом излучательности. Яркость увеличена в 25 раз.

Рисунок 6.3 — Сравнение результатов метода темпоральной излучательности.

Заключение

Основные результаты работы заключаются в следующем.

1. Предложен новый метод матрицы нескольких отражений, позволяющий сократить время вычисления освещения на порядок для трёх учитываемых переотражений света. Также данный метод уменьшает потребление памяти метода излучательности в 3 раза.
2. Предложен новый метод локальной матрицы, позволяющий учитывать влияние отражений света от близко расположенных объектов и эффективно обновлять содержимое матрицы форм-факторов при движении камеры. Данный метод решает проблему сложности вычисления освещения алгоритмом излучательности на сценах большого масштаба.
3. Предложен метод создания виртуальных площадок, позволяющий создавать прокси-геометрию, используемую для вычисления освещения, не имеющую представления в 3D-пространстве. При этом количество используемых элементов геометрии сокращается до 2 порядков (с 200000 треугольников до 2000 виртуальных площадок).
4. Предложен новый метод темпоральной излучательности, позволяющий распределить вычислительную нагрузку на несколько кадров без добавления темпоральных артефактов.

В совокупности предложенные методы существенно сокращают время вычисления освещения (ускорение до 100 раз при использовании методов матрицы нескольких отражений, локальной матрицы и темпоральной излучательности). Помимо этого сложность решения уравнения излучательности на один кадр сведена к $O(NQ)$, где $Q << N$ ($Q \approx 10-200, N \approx 2000-12000$). Данные результаты позволяют использовать разработанные алгоритмы на сложных сценах на различных устройствах.

В заключение автор выражает благодарность и большую признательность научному руководителю Фролову В. А. за поддержку, помощь, обсуждение результатов и научное руководство. Также автор благодарит Щербакову Г. Р. за поддержку и помощь в оформлении диссертации.

Публикации автора по теме диссертации

Научные статьи, опубликованные в рецензируемых журналах, индексируемых в международных базах Scopus, WoS, RSCI

A.1 A. S. Shcherbakov, V. A. Frolov Matrix Transformations for Effective Implementation of Radiosity Algorithm Using Graphic Processors // Light and Engineering. — Russian Federation, 2019. — Vol. 27, № 2. — P. 105—110. – (Scopus Q3, SJR: 0.297) [0.375/0.3125]

A.2 A. S. Shcherbakov, V. A. Frolov Dynamic Radiosity // Computer Science Research Notes. — Pilsen, Czech Republic, 2019. — № 2901. — P. 83—90. – (Scopus Q4, SJR: 0.108) [0.5/0.375]

A.3 Щербаков А. С., Фролов В. А., Галактионов В. А. Виртуальные площадки в алгоритме излучательности // Труды Института системного программирования РАН (электронный журнал). — 2022. — Т. 34, № 3. — С. 47—60. (RSCI, ИФ РИНЦ: 0.367) [0.875/0.75]

A.4 Щербаков А. С. Адаптация методов вычисления глобального освещения на основе алгоритма излучательности к архитектуре кадрового графа // Вычислительные методы и программирование. 2025. – Т. 26, № 2. С. 99–110. (RSCI, ИФ РИНЦ: 0.576) [0.75/0.75]

Список литературы

1. Autodesk AutoCAD: Trusted by millions, built to accelerate your creativity [Text]. — URL: https://www.autodesk.com/products/autocad/overview?term=1-YEAR&tab=subscription (visited on 09/23/2023).
2. Autodesk 3ds Max: Create massive worlds and high-quality designs [Text]. — URL: https://www.autodesk.com/products/3ds-max/overview?term=1-YEAR&tab=subscription (visited on 09/23/2023).
3. SketchUp [Text]. — URL: https://www.sketchup.com/ (visited on 09/23/2023).
4. Microsoft Flight Simulator [Text]. — URL: https://www.flightsimulator.com/ (visited on 09/23/2023).
5. War Thunder [Текст]. — URL: https://warthunder.ru/ru/ (дата обр. 23.09.2023).
6. Storytelling reimagined [Text]. — URL: https://www.unrealengine.com/en-US/solutions/film-television (visited on 09/23/2023).
7. FILM, ANIMATION AND CINEMATICS [Text]. — URL: https://unity.com/solutions/film-animation-cinematics (visited on 09/23/2023).
8. PlayStation VR2 [Text]. — URL: https://www.playstation.com/en-us/ps-vr2/ (visited on 09/23/2023).
9. Fulldive VR [Text]. — URL: https://www.fulldive.com/project/fulldive-vr (visited on 09/23/2023).
10. Лучшие VR-шлемы: от самых бюджетных к премиальным [Текст]. — 2023. — URL: https://habr.com/ru/companies/mvideo/articles/741792/ (дата обр. 23.09.2023).
11. Pokemon GO [Text]. — URL: https://www.pokemon.com/ru/app/pokemon-go/ (visited on 09/23/2023).
12. Star Walk 2 [Text]. — URL: https://vitotechnology.com/apps/star-walk-2 (visited on 09/23/2023).
13. Enlisted [Текст]. — URL: https://enlisted.net/ru/#!/ (дата обр. 23.09.2023).
14. Cyberbank [Текст]. — URL: https://www.cyberpunk.net/ru/ru/ (дата обр. 23.09.2023).
15. Смута [Текст]. — URL: https://smuta.games/ (дата обр. 23.09.2023).

16. DIALux evo - new calculation method [Text]. — URL: https://www.dialux.com/fileadmin/documents/DIALux_evo-_New_calculation_method.pdf (visited on 09/23/2023).

17. Lighting Software Tools [Text]. — URL: https://arendlighting.com/lighting-software-tools/?lang=en (visited on 09/23/2023).

18. Comparison of lighting simulation tools with focus on lighting quality [Text]. — URL: https://www.diva-portal.org/smash/get/diva2:723060/FULLTEXT01.pdf (visited on 09/23/2023).

19. RTXGI Quick Start [Text]. — 2021. — URL: https://github.com/NVIDIAGameWorks/RTXGI-DDGI/blob/main/docs/QuickStart.md (visited on 10/07/2024).

20. Using Sponza in the RTXGI Test Harness [Text]. — 2021. — URL: https://github.com/NVIDIAGameWorks/RTXGI-DDGI/blob/main/samples/test-harness/data/gltf/sponza/README.md (visited on 10/07/2024).

21. *Schlick, C.* A Survey of Shading and Reflectance Models [Text] / C. Schlick // Computer Graphics Forum. — 2002. — Apr. — Vol. 13.

22. *Lambert, J. H.* Photometria sive de mensure et gradibus luminis colorum et umbra [Text] / J. H. Lambert. — 1760.

23. *Oren, M.* Generalization of the Lambertian model and implications for machine vision [Text] / M. Oren, S. K. Nayar // International Journal of Computer Vision. — 1995. — Vol. 14. — P. 227—251. — URL: https://api.semanticscholar.org/CorpusID:2367943.

24. *Phong, B. T.* Illumination for computer generated pictures [Text] / B. T. Phong // Communications of the ACM. — 1975. — Vol. 18. — P. 311—317. — URL: https://api.semanticscholar.org/CorpusID:1439868.

25. *Blinn, J. F.* Models of Light Reflection for Computer Synthesized Pictures [Text] / J. F. Blinn // Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques. — San Jose, California : Association for Computing Machinery, 1977. — P. 192—198. — (SIGGRAPH '77). — URL: https://doi.org/10.1145/563858.563893.

26. *Pharr, M.* Physically Based Rendering: From Theory To Implementation [Text]. Vol. 2 / M. Pharr, G. Humphreys. — 08/2004.

27. Comparative Analysis of Real-Time Global Illumination Techniques in Current Game Engines [Text] / C. Lambru [et al.] // IEEE Access. — 2021. — Sept. — Vol. PP. — P. 1—1.

28. *Nishita*, *T.* Keynote Paper : A Survey on Recent Developments in Global Illumination Techniques [Text] / T. Nishita, H. Otsu //. — 2014. — URL: https://api.semanticscholar.org/CorpusID:666392.

29. *Flatt*, *D.* AMD Radeon Rays Integrated into Unity's GPU Progressive Lightmapper [Text] / D. Flatt. — 2018. — URL: https://blog.unity.com/technology/amd-radeon-rays-integrated-into-unitys-gpu-progressive-lightmapper (visited on 08/26/2023).

30. Quake Lightmaps [Text]. — 2015. — URL: https://jbush001.github.io/2015/06/11/quake-lightmaps.html (visited on 08/26/2023).

31. *Hobson*, *J.* The Indirect Lighting Pipeline of 'God of War' [Text] / J. Hobson. — 2019. — URL: https://www.gdcvault.com/play/1026323/The-Indirect-Lighting-Pipeline-of (visited on 08/26/2023).

32. *Green*, *R.* Spherical Harmonic Lighting: The Gritty Details [Text] / R. Green //. — 2003. — URL: https://api.semanticscholar.org/CorpusID:116856600.

33. *Sillion*, *F. X.* Radiosity and global illumination [Text] / F. X. Sillion, C. Puech //. — 1994. — URL: https://api.semanticscholar.org/CorpusID:31648401.

34. Interactive Indirect Illumination Using Voxel Cone Tracing: A Preview [Text] / C. Crassin [et al.] //. — 02/2011. — P. 207.

35. *Keller*, *A.* Instant Radiosity [Text] / A. Keller // Proceedings of SIGGRAPH. — 2002. — Dec. — Vol. 31.

36. *Dachsbacher*, *C.* Reflective shadow maps [Text] / C. Dachsbacher, M. Stamminger // ACM Symposium on Interactive 3D Graphics and Games. — 2005. — URL: https://api.semanticscholar.org/CorpusID:207156246.

37. *Kaplanyan*, *A.* Cascaded light propagation volumes for real-time indirect illumination [Text] / A. Kaplanyan, C. Dachsbacher // ACM Symposium on Interactive 3D Graphics and Games. — 2010. — URL: https://api.semanticscholar.org/CorpusID:15969508.

38. *Yudintsev*, *A.* Scalable Real-Time Global Illumination for Large Scenes [Text] / A. Yudintsev. — 2019. — URL: https://www.gdcvault.com/play/1026469/Scalable-Real-Time-Global-Illumination (visited on 08/26/2023).

39. *Marrs*, *A.* RTXGI: Scalable Ray Traced Global Illumination in Real Time [Text] / A. Marrs. — 2020. — URL: https://developer.download.nvidia.com/rtx/rtxgi/NVIDIA-RTXGI-03-23-2020-v2.pdf (visited on 08/26/2023).

40. *Ritschel*, *T.* Approximating dynamic global illumination in image space [Text] / T. Ritschel, T. Grosch, H.-P. Seidel // ACM Symposium on Interactive 3D Graphics and Games. — 2009. — URL: https://api.semanticscholar.org/CorpusID:2049022.

41. *Shanmugam*, *P.* Hardware accelerated ambient occlusion techniques on GPUs [Text] / P. Shanmugam, O. Arikan // Proceedings of the 2007 symposium on Interactive 3D graphics and games. — 2007. — URL: https://api.semanticscholar.org/CorpusID:5583249.

42. Practical Real-Time Strategies for Accurate Indirect Occlusion [Text] / J. Jimenez [et al.] //. — 2016. — URL: https://api.semanticscholar.org/CorpusID:222131256.

43. *Foley*, *T.* KD-Tree Acceleration Structures for a GPU Raytracer [Text] / T. Foley, J. Sugerman // Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware. — Los Angeles, California : Association for Computing Machinery, 2005. — P. 15—22. — (HWWS '05). — URL: https://doi.org/10.1145/1071866.1071869.

44. Real-Time KD-Tree Construction on Graphics Hardware [Text] / K. Zhou [et al.] // ACM Trans. Graph. — New York, NY, USA, 2008. — Dec. — Vol. 27, no. 5. — URL: https://doi.org/10.1145/1409060.1409079.

45. Fast, Parallel, GPU-Based Construction of Space Filling Curves and Octrees [Text] / P. Ajmera [et al.] // Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games. — Redwood City, California : Association for Computing Machinery, 2008. — (I3D '08). — URL: https://doi.org/10.1145/1342250.1357022.

46. A Survey on Bounding Volume Hierarchies for Ray Tracing [Text] / D. Meister [et al.] // Computer Graphics Forum. — 2021. — May. — Vol. 40. — P. 683—712.

47. Microfacet Models for Refraction through Rough Surfaces [Text] / B. Walter [et al.] // Proceedings of the 18th Eurographics Conference on Rendering Techniques. — Grenoble, France : Eurographics Association, 2007. — P. 195—206. — (EGSR'07).

48. Introduction to DirectX raytracing [Text] / C. Wyman [et al.] //. — 08/2018. — P. 1—1.

49. *Stich*, *M.* Introduction to NVIDIA RTX and DirectX Ray Tracing [Text] / M. Stich. — 2018. — URL: https://developer.nvidia.com/blog/introduction-nvidia-rtx-directx-ray-tracing/ (visited on 08/26/2023).

50. DirectX Raytracing (DXR) Functional Spec [Text]. — 2023. — URL: https://microsoft.github.io/DirectX-Specs/d3d/Raytracing.html#inline-raytracing (visited on 08/26/2023).

51. *Tomasi*, *C.* Bilateral filtering for gray and color images [Text] / C. Tomasi, R. Manduchi // Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271). — 1998. — P. 839—846.

52. ReSTIR GI: Path Resampling for Real-Time Path Tracing [Text] / Y. Ouyang [et al.] // Computer Graphics Forum. — 2021. — Vol. 40. — URL: https://api.semanticscholar.org/CorpusID:237988528.

53. *Erikson*, *C.* Hlods for Faster Display of Large Static and Dynamic Environments [Text] / C. Erikson, D. Manocha, W. Baxter //. — 03/2001. — P. 111—120.

54. *Scherzer*, *D.* A Survey of Real-Time Hard Shadow Mapping Methods [Text] / D. Scherzer, M. Wimmer, W. Purgathofer // Comput. Graph. Forum. — 2011. — Mar. — Vol. 30. — P. 169—186.

55. *Yang*, *L.* A Survey of Temporal Antialiasing Techniques [Text] / L. Yang, S. Liu, M. Salvi // Computer Graphics Forum. — 2020. — May. — Vol. 39. — P. 607—621.

56. Temporal Super Resolution [Text]. — URL: https://docs.unrealengine.com/5.2/en-US/temporal-super-resolution-in-unreal-engine/ (visited on 08/26/2023).

57. *Bavoil*, *L.* Image-space horizon-based ambient occlusion [Text] / L. Bavoil, M. Sainz, R. Dimitrov // International Conference on Computer Graphics and Interactive Techniques. — 2008. — URL: https://api.semanticscholar.org/CorpusID:45277239.

58. Horizon-Based Ambient Occlusion Plus (HBAO+) [Text]. — URL: https://developer.nvidia.com/rendering-technologies/horizon-based-ambient-occlusion-plus (visited on 08/26/2023).

59. Global illumination with radiance regression functions [Text] / P. Ren [et al.] // ACM Transactions on Graphics (TOG). — 2013. — Vol. 32. — P. 1—12. — URL: https://api.semanticscholar.org/CorpusID:4395124.

60. *Thomas*, *M. M.* Deep Illumination: Approximating Dynamic Global Illumination with Generative Adversarial Network [Text] / M. M. Thomas, A. G. Forbes // ArXiv. — 2017. — Vol. abs/1710.09834. — URL: https://api.semanticscholar.org/CorpusID:12767524.

61. Real-time neural radiance caching for path tracing [Text] / T. Müller [et al.] // ACM Transactions on Graphics (TOG). — 2021. — Vol. 40. — P. 1—16. — URL: https://api.semanticscholar.org/CorpusID:235606372.

62. *Thomas*, *M. M.* Deep Illumination: Approximating Dynamic Global Illumination with Generative Adversarial Network [Text] / M. M. Thomas, A. G. Forbes // CoRR. — 2017. — Vol. abs/1710.09834. — arXiv: 1710.09834. — URL: http://arxiv.org/abs/1710.09834.

63. *Hanrahan*, *P.* A rapid hierarchical radiosity algorithm [Text] / P. Hanrahan, D. B. Salzman, L. Aupperle // Proceedings of the 18th annual conference on Computer graphics and interactive techniques. — 1991. — URL: https://api.semanticscholar.org/CorpusID:9382211.

64. *Damez*, *C.* Space-Time Hierarchical Radiosity [Text] / C. Damez, F. Sillion // Rendering techniques '99 (Proceedings of the 10th Eurographics Workshop on Rendering). — 1999. — Aug.

65. *Martín*, *I.* Frame-to-frame coherent animation with two-pass radiosity [Text] / I. Martín, X. Pueyo, D. Tost // Visualization and Computer Graphics, IEEE Transactions on. — 2003. — Feb. — Vol. 9. — P. 70—84.

66. *Besuievksy*, *G.* The Multi-Frame Lighting Method: A Monte Carlo Based Solution for Radiosity in Dynamic Environments [Text] / G. Besuievksy, M. Sbert //. — 01/1996. — P. 185—194.

67. *Besuievksy*, *G.* A Monte Carlo Method for Accelerating the Computation of Animated Radiosity Sequences. [Text] / G. Besuievksy, X. Pueyo //. — 02/2001. — P. 201—208.

68. *Shirley*, *P.* Time Complexity of Monte Carlo Radiosity [Text] / P. Shirley // Computers & Graphics. — 1992. — July. — Vol. 16.

69. Global multipath Monte Carlo algorithms for radiosity [Text] / M. Sbert [et al.] // The Visual Computer. — 1996. — Feb. — Vol. 12. — P. 47—61.

70. Hierarchical Monte Carlo Radiosity [Text] / P. Bekaert [et al.] //. — 01/1998. — P. 259—268.

71. *Cools*, *R.* An Empirical Comparison Of Monte Carlo Radiosity Algorithms [Text] / R. Cools, Y. Willems. — 1999. — Feb.

72. *Martin*, *S.* A Real-time Radiosity Architecture [Text] / S. Martin. — 2010. — URL: https://www.ea.com/frostbite/news/a-real-time-radiosity-architecture (visited on 08/26/2023).

73. *Щербаков*, *А.* Автоматическое упрощение геометрии для расчёта вторичной освещенности методом излучательности [Текст] / А. Щербаков, В. Фролов // Сборник трудов Графикон 2016. — ННГАСУ, 2016. — С. 34—38.

74. Texture Block Compression in Direct3D 11 [Text]. — 2020. — URL: https://learn.microsoft.com/en-us/windows/win32/direct3d11/texture-block-compression-in-direct3d-11 (visited on 08/26/2023).

75. *Dammertz*, *H.* Hammersley Points on the Hemisphere [Text] / H. Dammertz. — 2012. — URL: http://holger.dammertz.org/stuff/notes_HammersleyOnHemisphere.html (visited on 08/27/2023).

76. *Walker*, *A. J.* New fast method for generating discrete random numbers with arbitrary frequency distributions [Text] / A. J. Walker // Electronics Letters. — 1974. — Vol. 10. — P. 127—128. — URL: https://api.semanticscholar.org/CorpusID:121641787.

77. *Lehmann*, *H.* Weighted Random Sampling on GPUs [Text] / H. Lehmann, L. Hübschle-Schneider, P. Sanders // CoRR. — 2021. — Vol. abs/2106.12270. — arXiv: 2106.12270. — URL: https://arxiv.org/abs/2106.12270.