2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

doi 10.26089/NumMet.v26r207 УДК 004.925.3

Адаптация методов вычисления глобального освещения на основе алгоритма излучательности к архитектуре кадрового графа

А. С. Щербаков
Московский государственный университет имени М. В. Ломоносова,
факультет вычислительной математики и кибернетики, Москва, Российская Федерация
ORCID: 0000-0002-5360-4479, e-mail: alex.shcherbakov.as@gmail.com

Аннотация: В статье рассматривается интеграция метода излучательности в архитектуру кадрового графа с целью оптимизации потребления памяти графического процессора. Предложены способы адаптации метода матрицы нескольких отражений, метода локальной матрицы и метода темпоральной излучательности с учетом требований к управлению ресурсами на графическом процессоре. Описана схема переиспользования временной памяти между различными этапами обработки глобального освещения, что позволяет снизить нагрузку на видеопамять и минимизировать накладные расходы при выполнении алгоритма. Проведено экспериментальное исследование на тестовых сценах, подтверждающее эффективность предложенного подхода и демонстрирующее снижение потребления памяти.

Ключевые слова: глобальное освещение, вычислительные графы, компьютерная графика.

Для цитирования: Щербаков А. С. Адаптация методов вычисления глобального освещения на основе алгоритма излучательности к архитектуре кадрового графа // Вычислительные методы и программирование. 2025. **26**, № 2. 99–110. doi 10.26089/NumMet.v26r207.

Adaptation of Global Illumination Computation Methods Based on the Radiosity Algorithm to the Frame Graph Architecture

Alexandr S. Shcherbakov
Lomonosov Moscow State University,
Faculty of Computational Mathematics and Cybernetics, Moscow, Russia
ORCID: 0000-0002-5360-4479, e-mail: alex.shcherbakov.as@gmail.com

Abstract: This paper examines the integration of the radiosity method into the frame graph architecture to optimize GPU memory consumption. Methods for adapting the multi-bounce matrix method, the local matrix method, and the temporal radiosity method are proposed, considering resource management requirements on the GPU. A scheme for reusing temporal memory across different stages of global illumination processing is described, which helps reduce video memory load and minimize overhead during algorithm execution. An experimental study on test scenes confirms the effectiveness of the proposed approach and demonstrates a reduction in memory consumption.

Keywords: global illumination, computational graphs, computer graphics.

For citation: A. S. Shcherbakov, "Adaptation of Global Illumination Computation Methods Based on the Radiosity Algorithm to the Frame Graph Architecture," Numerical Methods and Programming. **26** (2), 99–110 (2025). doi 10.26089/NumMet.v26r207.

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

1. Введение. Глобальное освещение играет ключевую роль в формировании реалистичной визуальной картины в компьютерной графике, моделируя сложные световые взаимодействия, включая многократные отражения и рассеивание. Один из наиболее точных методов расчета глобального освещения в приложениях с предобработкой 3D-сцены — метод излучательности [1], основанный на решении уравнения освещенности с учетом передачи энергии между поверхностями сцены. Однако его высокая вычислительная сложность и значительное потребление памяти ограничивают применение в интерактивных системах, особенно в интерактивных приложениях с динамическим освещением.

Современные приложения компьютерной графики используют кадровый граф [2, 3] — архитектурный подход, позволяющий эффективно управлять процессами отрисовки, перераспределяя вычислительные ресурсы и оптимизируя хранение промежуточных данных. При адаптации различных алгоритмов компьютерной графики под данную архитектуру важно строго различать постоянные и временные ресурсы, используемые в алгоритме. Данное различие, как правило, отражается в коде при помощи API конкретной реализации кадрового графа и используется для следующих целей:

1. Оптимизация переиспользования памяти GPU за счет алиасинга памяти [4].
2. Оптимизация расстановки барьеров для ресурсов [5].

Поскольку расстановка барьеров тесно связана с текущим состоянием кадрового графа, адаптация метода для данного параметра не представляется возможной. В то же время, алгоритмы компьютерной графики можно представлять различным способом в виде графа, где ребра представляют собой зависимости по данным, а вершины — операции над данными. При этом различные представления имеют разное потребление памяти.

В данной работе рассматривается схема переиспользования памяти в методах излучательности для кадрового графа.

Разработанные адаптации алгоритмов направлены на оптимизацию расчета освещения в сценах с высокой детализацией и динамическими источниками света. Проведено экспериментальное исследование предложенного подхода, демонстрирующее его преимущества по сравнению с методами, лишенными возможности переиспользовать память GPU для различных стадий обработки кадра.

2. Архитектура кадрового графа. Кадровый граф представляет собой структуру данных, используемую в современных графических системах для управления процессами формирования изображения. Его основная идея заключается в представлении всех этапов вычислений в виде ориентированного ациклического графа, где вершины соответствуют вычислительным операциям (проходам обработки), а ребра — зависимостям по данным между этими операциями. Такой подход позволяет гибко оптимизировать использование ресурсов, упрощает планирование вычислений и снижает накладные расходы на управление памятью.

2.1. Основные компоненты кадрового графа.

1. Ресурсы — представляют собой текстуры, буферы и константы, необходимые для формирования изображения. Ресурсы могут использоваться на нескольких этапах вычислений, поэтому важно управлять их жизненным циклом и распределением памяти.
2. Проходы обработки — отдельные вычислительные этапы, такие как обработка геометрии, расчет теней, вычисление освещенности. Каждый проход использует входные ресурсы и формирует выходные.
3. Зависимости — связи между проходами обработки, определяющие порядок выполнения операций и необходимость синхронизации между ними.

2.2. Преимущества использования кадрового графа.

1. Эффективное управление памятью. Граф позволяет переиспользовать ресурсы, освобождая их после завершения вычислений. Например, буфер освещенности, использованный на одном этапе обработки, может быть задействован на другом без дублирования данных.
2. Гибкость и расширяемость. За счет декларирования зависимостей можно легко модифицировать процесс формирования изображения, добавляя или изменяя этапы обработки без перепроектирования всей системы.
3. Оптимизация производительности. Поскольку граф полностью описывает зависимости, он позволяет заранее планировать выполнение операций и минимизировать избыточные вычисления.

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

4. Минимизация барьеров. Современные графические API, такие как Vulkan и DirectX12, используют примитив барьера [6–8] для перевода ресурса из одного состояния в другое. Например, для перевода из состояния чтения в состояние записи. Такой подход позволяет графическому процессору дополнительно параллелизовать вычисления из разных очередей команд, если отсутствуют конфликты по доступу к данным. Также барьеры указывают видеокарте, когда необходимо записать данные ресурса из кешей в глобальную память.

Таким образом, архитектура кадрового графа предоставляет мощный инструмент для интеграции сложных алгоритмов, но требует адаптации методов с учетом ограничений по памяти и зависимостей между вычислениями.

3. Методы глобального освещения. Глобальное освещение является важным элементом компьютерной графики, моделируя сложные световые взаимодействия, включая многократные отражения, рассеивание и непрямые тени. Методы глобального освещения можно разделить на две основные группы:

- Методы с вычислением во время формирования изображения — трассировка лучей, воксельные представления сцены, методы экранного пространства.
- Методы с предварительными расчетами — карты освещенности, сферические гармоники, метод излучательности.

Каждая из этих групп имеет свои преимущества и ограничения. Методы, работающие во время формирования изображения, обладают высокой адаптивностью, но требуют значительных вычислительных ресурсов. В свою очередь, методы с предварительными расчетами позволяют достичь более высокой точности, но накладывают ограничения на динамику сцены.

3.1. Методы с вычислением во время формирования изображения. Методы направлены на баланс между вычислительной эффективностью и точностью:

- Трассировка лучей [9, 10] — вычисление пересечений лучей с поверхностями для определения освещения. Современные графические процессоры поддерживают этот метод аппаратно, но он остается вычислительно затратным.
- Трассировка вокселей конусами [11] — аппроксимация распространения света в сцене с использованием воксельного представления. Позволяет учитывать многократные отражения и рассеивание, но требует значительных ресурсов памяти.
- Методы экранного пространства — используют данные из буфера глубины для оценки освещения, например SSDO [12] и GTAO [13]. Подход эффективен для моделирования локальных эффектов, но не учитывает объекты за пределами видимой области.
- RTX Global Illumination (RTXGI) [14] — метод глобального освещения, использующий аппаратное ускорение трассировки лучей для обновления сетки проб освещенности в динамических сценах. Освещение моделируется с помощью Dynamic Diffuse Global Illumination (DDGI) [15], где пробы собирают информацию о падающем свете, а затем значения освещенности интерполируются по всей сцене. Для уменьшения вычислительных затрат применяется темпоральная фильтрация и пространственная интерполяция значений проб. Метод позволяет учитывать сложные световые эффекты, такие как многократные отражения и рассеивание, но требует значительных вычислительных ресурсов и видеокарты с поддержкой трассировки лучей.

3.2. Методы с предварительными расчетами. Методы, использующие предварительные вычисления, позволяют заранее определить сложные световые взаимодействия и применять их при формировании изображения:

- Карты освещенности [16] — предвычисленные текстуры с информацией об освещении, широко применяются в игровых движках, но не поддерживают динамические изменения источников света.
- Сферические гармоники [17] — представляют освещение в виде коэффициентов разложения, что позволяет эффективно кодировать диффузное освещение, но накладывает ограничения на точность воспроизведения световых эффектов.
- Метод излучательности — вычисляет распределение световой энергии между поверхностями сцены, решая систему уравнений освещенности. Метод позволяет моделировать непрямое освещение с высокой точностью и не содержит шумов, характерных для стохастических методов.

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

Метод излучательности был выбран для исследования по следующим причинам:

1. Точность моделирования непрямого освещения — метод позволяет учитывать многократные отражения света с любой заранее заданной точностью.
2. Эффективная адаптация для графических процессоров — современные версии метода, позволяют существенно снизить вычислительные затраты.
3. Совместимость с произвольными архитектурами графических приложений — метод не накладывает ограничений на формат данных и наличие определенных текстур или буферов в приложении.

4. Метод излучательности и его модификации.

4.1. Базовый метод излучательности. Метод излучательности [1] — один из классических алгоритмов вычисления глобального освещения, основанный на решении уравнения освещенности с учетом многократного обмена световой энергией между поверхностями сцены. В отличие от методов, использующих трассировку лучей, метод излучательности предполагает дискретизацию сцены на элементы (полигоны или воксели) с последующим расчетом их взаимного влияния через форм-факторы — коэффициенты, определяющие, какая доля энергии передается от одной поверхности к другой.

Общее уравнение излучательности для каждого элемента сцены имеет вид:

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

где B_i — излучательность элемента i, E_i — энергия, непосредственно испускаемая элементом, C_i — коэффициент отражения, F_{ij} — форм-фактор, зависящий от геометрии сцены и ориентации элементов, N — общее число дискретизированных элементов (полигонов, площадок).

Базовый метод предполагает итеративный процесс обновления значений излучательности. Однако вычислительная сложность стандартного подхода $O(N^2K)$, где K — количество учитываемых отражений освещения, делает его непрактичным для сцен с большим числом элементов.

Для данного алгоритма разработано большое количество расширений, которые можно разделить на две группы:

1. Модификации предобработки сцены. В эту группу включаются методы, направленные на оптимизацию этапа предобработки и вычисления форм-факторов [18], и методы, улучшающие точность данных, генерируемых в ходе предобработки [19].
2. Модификации вычисления освещения во время визуализации сцены.

Адаптация для архитектуры кадрового графа требуется только для второй группы методов. Поэтому далее рассмотрены именно эти подходы.

4.2. Иерархическая излучательность. Метод иерархической излучательности [20, 21] был предложен для снижения вычислительной сложности базового алгоритма. Он использует иерархическую дискретизацию сцены, где крупные элементы сначала обмениваются световой энергией, а затем этот процесс уточняется для более мелких деталей. Основные идеи метода — использование адаптивного разбиения сцены и вычисление агрегированных форм-факторов для групп поверхностей.

Хотя метод иерархической излучательности снижает сложность вычислений до $O(N \log N)$, он требует сложной структуры данных и плохо адаптируется к вычислению на графических процессорах, так как при переносе освещения между различными уровнями иерархии требуется до $\log^2 N$ проходов, каждый из которых зависит от предыдущих и требует синхронизации с ними. Ввиду этих ограничений метод практически не используется в современных реализациях.

4.3. Решение системы линейных уравнений. Метод излучательности можно также представить как систему линейных алгебраических уравнений (СЛАУ):

$$(I - F \operatorname{diag} C)B = E,$$

где B — вектор излучательности полигонов, E — вектор энергии, испускаемой полигонами, C — вектор коэффициентов отражения полигонов, F — матрица форм-факторов.

Данный подход также имеет проблему большого количества синхронизаций. Помимо этого, точность метода заметно ниже, чем у базового алгоритма излучательности из-за множественных операций деления, каждая из которых снижает точность результата.

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

4.4. Метод матрицы нескольких отражений. Одной из ключевых проблем метода излучательности является линейная зависимость сложности вычислений от количества переотражений света. В классическом подходе каждое дополнительное переотражение требует выполнения нового этапа вычислений, что увеличивает время обработки.

Метод матрицы нескольких отражений [22] решает эту проблему, расширяя этап предобработки сцены и вводя матрицу форм-факторов F_{multi}, учитывающую несколько отражений:

$$F_{\text{multi}} = \sum_{k=1}^{K} (F \operatorname{diag} C)^k,$$

где K — количество учитываемых переотражений.

Использование предвычисленной матрицы позволяет избавиться от явного итеративного процесса, сократив число необходимых операций при визуализации сцены.

4.5. Метод локальной матрицы. При традиционном подходе матрица форм-факторов охватывает всю сцену, что требует значительного объема памяти и большого количества операций для обновления освещения. Метод локальной матрицы [23] решает эту проблему путем:

- ограничения области хранения форм-факторов только ближайшими к камере элементами сцены,
- адаптивного обновления данных при движении камеры,
- использования потоковой загрузки данных на GPU.

Вместо хранения полной матрицы коэффициентов метод локальной матрицы использует ограниченное подмножество данных, которые обновляются по мере необходимости. Это позволяет существенно снизить нагрузку на видеопамять без заметных потерь качества освещения.

4.6. Метод темпоральной излучательности. Метод темпоральной излучательности [24] распределяет вычисления, необходимые для получения освещения, по нескольким кадрам, что позволяет уменьшить мгновенную нагрузку на аппаратное обеспечение.

Принцип работы заключается в том, что освещенность каждой поверхности обновляется не полностью за один кадр, а частично, с накоплением результатов:

$$B_i^f = w \cdot B_i^{f-1} + (1 - w) \cdot B_i^{\text{new}},$$

где w — коэффициент сглаживания, регулирующий влияние накопленных значений, B_i^{f-1} — освещение площадки с индексом i на предыдущем кадре, B_i^{new} — частичное освещение, вычисленное для текущего кадра, f — номер кадра.

Строка матрицы форм-факторов для одного полигона представляется в виде алиас-таблицы [25], что позволяет заменить умножение матрицы на вектор на выборку из множества алиас-таблиц. Это приводит к существенному сокращению количества вычислений на кадр.

В данной работе рассматривается адаптация к кадровому графу расширений метода излучательности, которые могут быть использованы на графических процессорах и не требуют множественных барьеров для проходов алгоритма.

5. Адаптация алгоритмов для кадрового графа.

5.1. Метод матрицы нескольких отражений. Данный метод упрощает интеграцию вычисления глобального освещения в кадровый граф по сравнению с базовым алгоритмом излучательности, так как он требует единственного умножения матрицы на вектор.

Соответственно, для данного метода нужны только два временных ресурса: E — вектор светимости полигонов, который существует от вычисления теней до вычисления глобального освещения; B — вектор финального освещения, который создается в единственном узле алгоритма и существует до конца применения освещения к сцене (рис. 1).

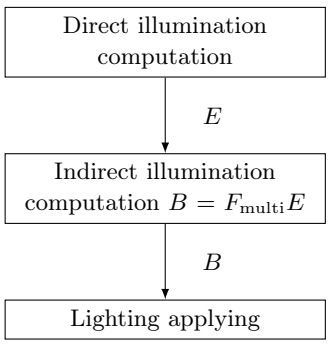


Рис. 1. Подграф вычисления освещения методом матрицы нескольких отражений. E — вектор первичного освещения, B — вектор вторичного освещения

Fig. 1. Subgraph for multi-reflection method of global illumination computation. E — direct lighting vector, B — indirect lighting vector

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

5.2. Метод локальной матрицы. Этот алгоритм добавляет несколько новых этапов в граф, которые становятся активны только при движении камеры, тем не менее их необходимо учесть для оценки пиковой нагрузки на систему аллокации памяти.

Для обновления матрицы форм-факторов, помимо исходных столбца и строки форм-факторов для добавляемого полигона, необходимо вычислить форм-факторы для второго и третьего переотражений света (эти же компоненты в неявном виде содержат последующие переотражения):

$$g_c = f_c\, c\, \big((f_r \circ C)\, f_c\big),$$

$$g_r = f_r\, c\, \big((f_r \circ C)\, f_c\big),$$

$$g'_c = F_{\text{multi}}\, g_c,$$

$$g'_r = g_r F_{\text{multi}},$$

где символом $\circ$ обозначено произведение Адамара, f_c — вектор-столбец форм-факторов нового полигона, f_r — вектор-строка форм-факторов нового полигона, c — цвет нового полигона, C — вектор цветов полигонов, g_c — вектор-столбец форм-факторов для второго переотражения, g_r — вектор-строка форм-факторов для второго переотражения, g'_c — вектор-столбец форм-факторов для третьего переотражения, g'_r — вектор-строка форм-факторов для третьего переотражения, F_{multi} — матрица нескольких отражений.

Финальная матрица форм-факторов вычисляется по формуле

$$F'_{\text{multi}} = F_{\text{multi}} + (g'_c + g_c + f_c)\,(g'_r + g_r + f_r)\,.$$

Последний этап — самый требовательный к ресурсам, так как содержит промежуточный результат в виде матрицы, размер которой равен размеру матрицы форм-факторов. Для минимизации количества потребляемой памяти и операций чтения/записи в глобальную память графического процессора вся последняя формула должна быть вычислена в едином узле графа (рис. 2).

Такой подход позволяет увеличить потребляемую память на 6 векторов размера N, которые необходимы для обновления матрицы форм-факторов.

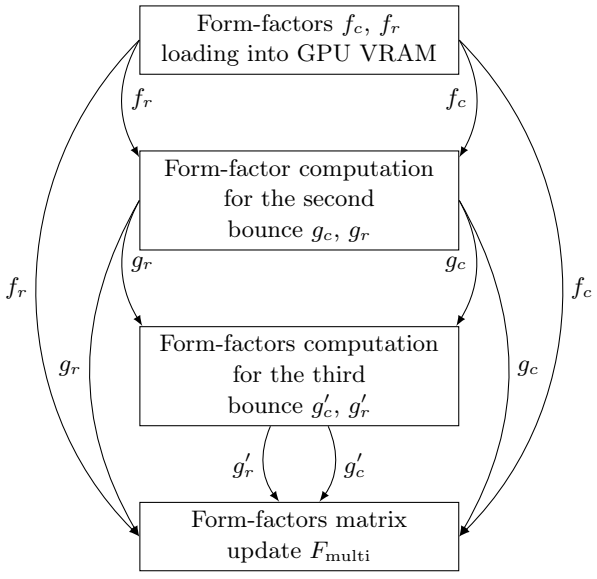


Рис. 2. Подграф обновления локальной матрицы

Fig. 2. Local matrix update subgraph

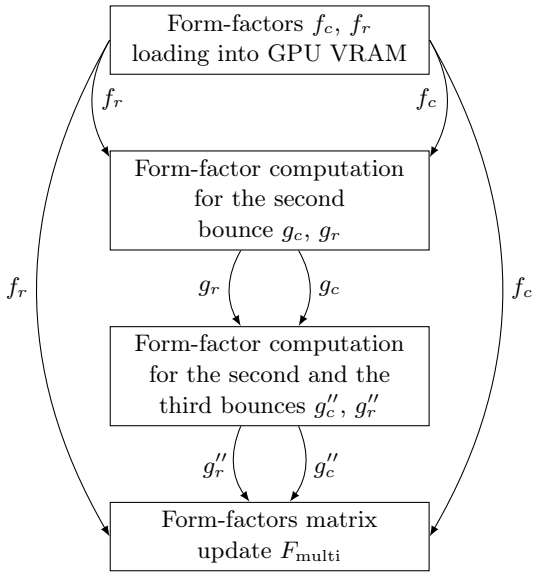


Рис. 3. Оптимизированный подграф обновления локальной матрицы

Fig. 3. Optimized local matrix update subgraph

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

Перераспределив операции между последним шагом (рис. 3) — обновлением матрицы и вычислением векторов для третьего переотражения, получим следующие формулы:

$$g_c'' = F_{\text{multi}}\, g_c + g_c = (F_{\text{multi}} + I)\, g_c,$$
$$g_r'' = g_r F_{\text{multi}} + g_r = g_r\, (F_{\text{multi}} + I)\,,$$
$$F'_{\text{multi}} = F_{\text{multi}} + (g_c'' + f_c)\, (g_r'' + f_r)\,.$$

Данная операция позволяет сократить память, потребляемую методом, до четырех векторов размера N. В общем случае применить такую оптимизацию для объединения значений g_c'' и f_c нельзя, так как при умножении матрицы на вектор каждая рабочая группа графического процессора должна иметь доступ ко всему вектору f_c.

Общее потребление памяти методами вычисления глобального освещения можно сократить, выполняя узлы для обновления локальной матрицы до вычисления первичного и глобального освещения методом матрицы нескольких отражений, тогда время жизни векторов f_c, f_r, g_c'', g_r'' и векторов E, B не пересекается и память может быть переиспользована, т.е. суммарное потребление временной памяти методами глобального освещения будет равно размеру четырех N-компонентных векторов.

5.3. Метод темпоральной излучательности. Метод темпоральной излучательности, как и многие другие методы, использующие текстуры и буферы [13, 26, 27], требует хранения обоих ресурсов на протяжении всего времени работы, хотя только один из них остается активным большую часть времени (рис. 4). Помимо этого, метод требует построения алиас-таблиц из матрицы форм-факторов при ее обновлении. Согласно методу, предложенному в [28], построение алиас-таблицы может быть осуществлено без использования дополнительной временной памяти графического процессора.

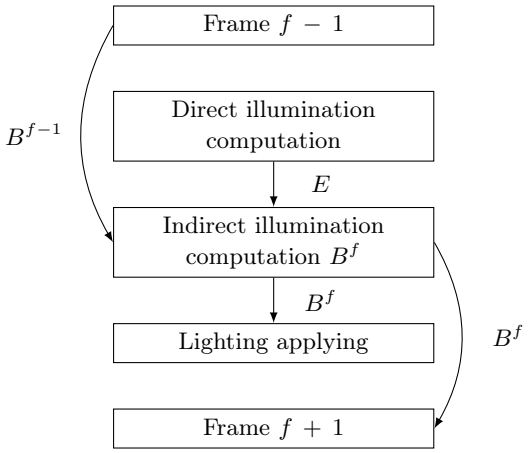


Рис. 4. Подграф вычисления освещения методом темпоральной излучательности. Векторы B^{f-1} и B^f используются в одной и той же вершине

Fig. 4. Subgraph for temporal radiosity method of global illumination computation. Vectors B^{f-1} and B^f are used in the same node

Так как один из двух буферов с освещением предыдущего кадра не активен на протяжении всего кадра, кроме выполнения узла для обновления темпоральной излучательности, память, которую он занимает, может быть переиспользована узлами, обновляющими матрицу форм-факторов для метода локальной матрицы (рис. 5). При этом второй буфер нельзя использовать схожим образом, так как его

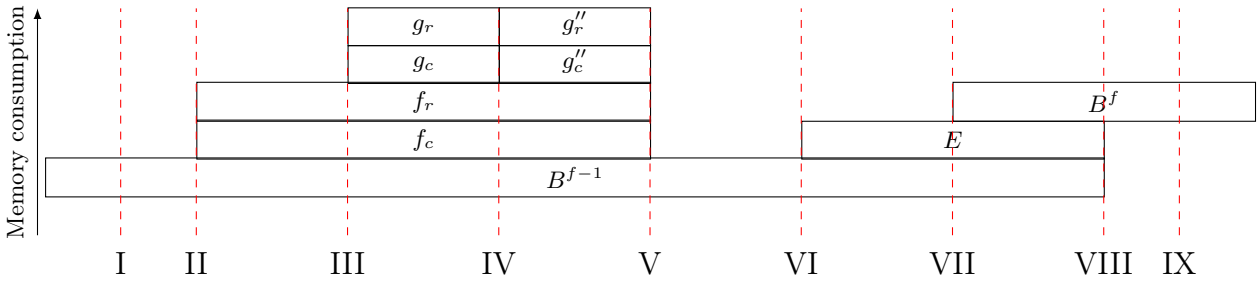


Рис. 5. Распределение памяти в системе вычисления глобального освещения. I — начало кадра, II — загрузка форм-факторов, III — вычисление g_c и g_r, IV — вычисление g_c'' и g_r'', V — обновление матрицы форм-факторов, VI — вычисление первичного освещения, VII — вычисление вторичного освещения, VIII — применение освещения, IX — конец кадра

Fig. 5. Memory distribution in the global illumination computation system. I — begin of the frame, II — form-factors uploading, III — calculation of g_c and g_r, IV — calculation of g_c'' and g_r'', V — form-factors matrix update, VI — direct lighting calculation, VII — indirect lighting calculation, VIII — lighting applying, IX — end of the frame

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

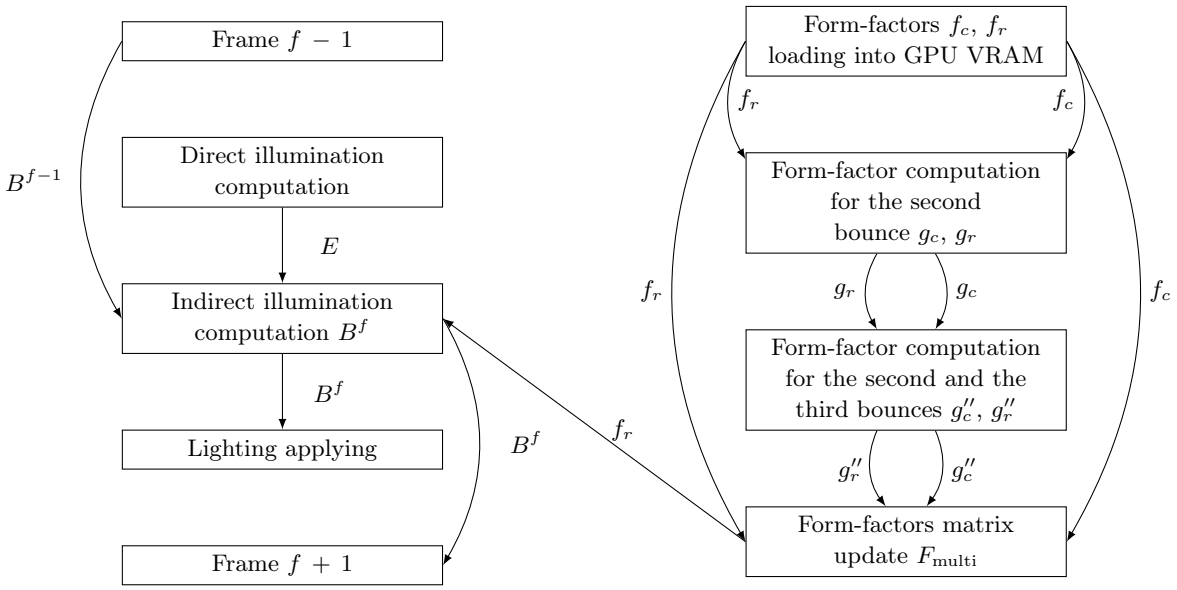


Рис. 6. Итоговый подграф системы вычисления глобального освещения

Fig. 6. Final global illumination system subgraph

данные будут использованы в финальном узле графа, применяющем освещение к сцене. В итоге метод матрицы нескольких отражений добавляет к потреблению временной памяти еще один N-компонентный вектор.

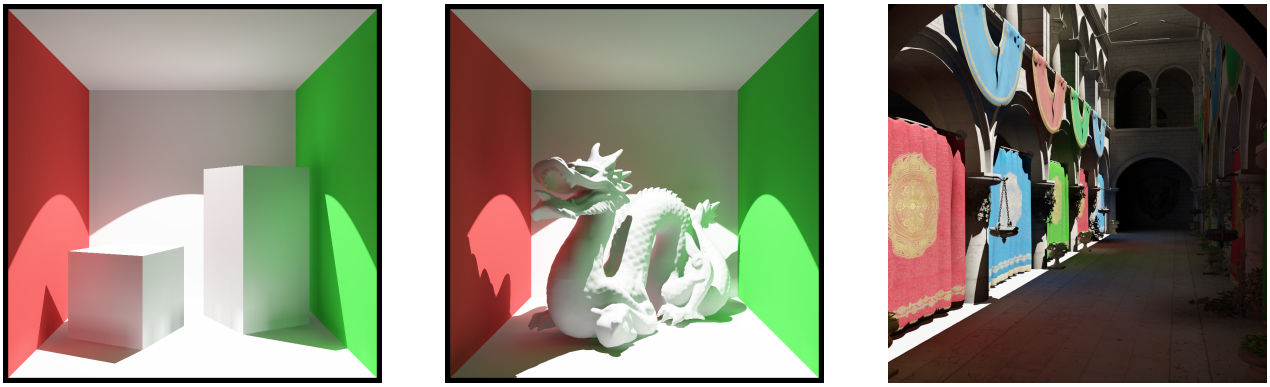
Шаблон использования памяти показан на рис. 5. Итоговый подграф для всех узлов, участвующих в вычислении глобального освещения, представлен на схеме рис. 6.

6. Экспериментальное исследование. Проверка потребления памяти приложением, использующим архитектуру кадрового графа, проводилась на сценах Crytek Sponza, Cornell Box с кубами и Cornell Box с драконом (рис. 7).

Результаты замеров потребления памяти представлены в табл. 1.

При указанных количествах полигонов, участвующих в вычислениях, временная память, потребляемая системой глобального освещения, полностью переиспользуется другими системами, реализованными в тестовом приложении.

Также было проведено сравнение предложенных методов излучательности и RTXGI (ведущий метод на данный момент) (рис. 8). Сравнение показывает их различия в скорости работы, потреблении памяти и точности освещения. RTXGI, основанный на аппаратной трассировке лучей, обеспечивает высокую



a) b) c)

Рис. 7. Тестовые сцены: a) CornellBox с кубами; b) CornellBox с драконом; c) Crytek Sponza

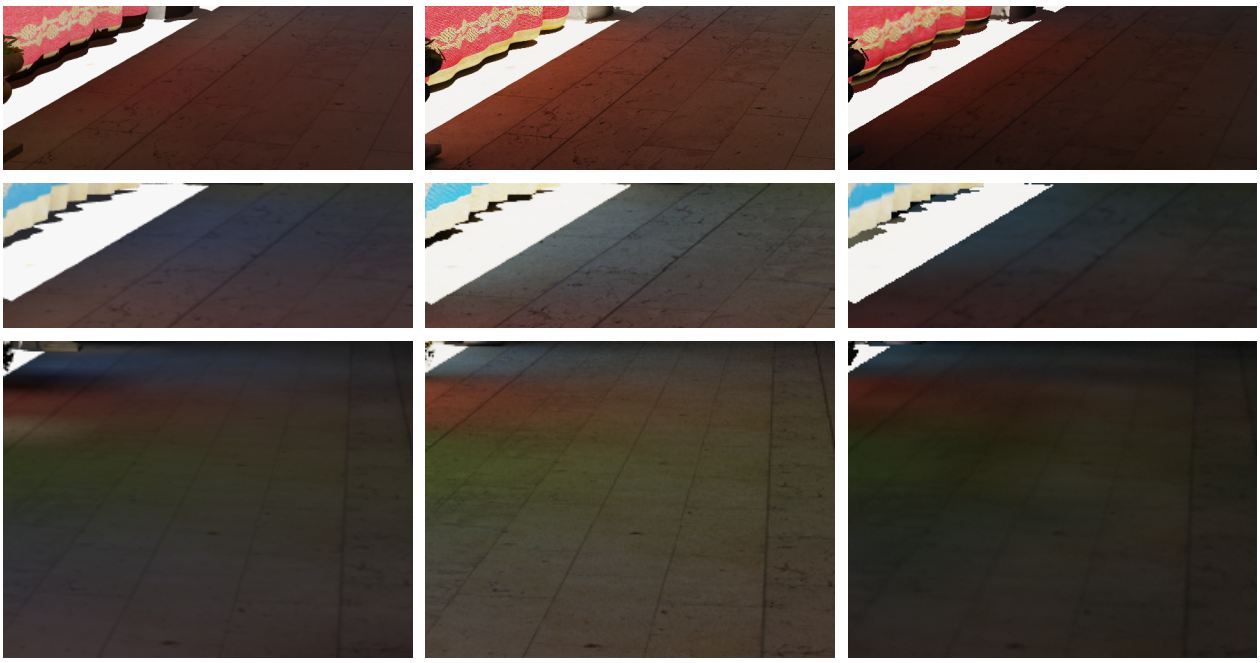
Fig. 7. Test scenes: a) CornellBox with cubes; b) CornellBox with a dragon; c) Crytek Sponza

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

Таблица 1. Потребление временной памяти для тестовых сцен

Table 1. Temporary memory consumption for test scenes

Scene	Polygons count	Non-aliased memory, MB	Proposed method, MB
CornellBox with cubes	1736	0.24	0.13
Crytek Sponza	262267	36	20
Cornell Box with dragon	871414	119.7	66.5
Cornell Box with dragon (5 instances)	4357070	598	332



a) b) c)

Рис. 8. Сравнение изображений: a) метод излучательности; b) эталон; c) RTXGI

Fig. 8. Image comparison: a) radiosity method; b) reference; c) RTXGI

Таблица 2. Сравнение с RTXGI на сцене Crytek Sponza

Table 2. Comparison with RTXGI on Crytek Sponza scene

Method	Time, ms	Accuracy, SSIM	Memory consumption, MB
RTXGI	0.61	0.66	129
Radiosity base method	12.01	0.74	12
Temporal radiosity	0.08	0.73	36
Temporal radiosity + Memory aliasing (Proposed method)	0.08	0.73	20

гибкость и поддерживает динамические изменения геометрии, но требует значительных вычислительных ресурсов. Результаты тестов на сцене Crytek Sponza показаны в табл. 2. Предложенные адаптации методов излучательности превосходят метод RTXGI в точности, скорости работы и потреблении временной памяти, но при этом метод RTXGI позволяет динамически менять 3D-сцену во время визуализации.

7. Заключение. Предложена схема интеграции метода излучательности в архитектуру кадрового графа, позволяющая минимизировать затраты памяти и вычислений при расчете глобального освещения. Основное внимание уделено методам матрицы нескольких отражений, локальной матрицы и темпораль-

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

ной излучательности, которые были адаптированы с учетом переиспользования временных ресурсов и управления последовательностью вычислений.

Экспериментальное тестирование на сценах Crytek Sponza и Cornell Box продемонстрировало, что предложенные методы позволяют эффективно переиспользовать временную память в рамках кадрового графа, обеспечивая баланс между точностью освещения и производительностью.

Список литературы

1. *Sillion F.X., Puech C.* Radiosity and global illumination. San Francisco: Morgan Kaufmann, 1994.
2. *O'Donnell Y.* FrameGraph: extensible rendering architecture in frostbite. Game developers conference. 2017. https://www.gdcvault.com/play/1024612/FrameGraph-Extensible-Rendering-Architecture-in. (Дата обращения: 5 марта 2025).
3. *Wihlidal G.* Halcyon: rapid innovation using modern graphics. https://media.contentapi.ea.com/content/dam/ea/seed/presentations/wihlidal2019-rebootdevelopblue-halcyon-rapid-innovation.pdf. (Дата обращения: 5 марта 2025).
4. *Sandu R., Shcherbakov A.* A resource allocation algorithm for a history-aware frame graph // J. WSCG. 2023. **31**, N 1–2. 63–70. doi 10.24132/jwscg.2023.7.
5. *Sandu R., Shcherbakov A.* GPU cache flush minimization in render graph systems. J. WSCG. 2024. **32**, N 1–2. 71–78. doi 10.24132/jwscg.2024.8.
6. Using resource barriers to synchronize resource states in Direct3D 12. https://learn.microsoft.com/en-us/windows/win32/direct3d12/using-resource-barriers-to-synchronize-resource-states-in-direct3d-12. (Дата обращения: 5 марта 2025).
7. Understanding Vulkan synchronization. https://www.khronos.org/blog/understanding-vulkan-synchronization. (Дата обращения: 5 марта 2025).
8. memoryBarrier(scope:after:before:). https://developer.apple.com/documentation/metal/mtlrendercommandencoder/memorybarrier(scope:after:before:). (Дата обращения: 5 марта 2025).
9. *Wyman C., Hargreaves S., Shirley P., Barré-Brisebois C.* Introduction to DirectX raytracing. 2018. https://intro-to-dxr.cwyman.org/. (Дата обращения: 5 марта 2025).
10. *Stich M.* Introduction to NVIDIA RTX and DirectX Ray Tracing. 2018. https://developer.nvidia.com/blog/introduction-nvidia-rtx-directx-ray-tracing/. (Дата обращения: 5 марта 2025).
11. *Crassin C., Neyret F., Sainz M., et al.* Interactive indirect illumination using voxel cone tracing: a preview. 2011. doi 10.1145/1944745.1944787.
12. *Ritschel T., Grosch T., Seidel H.-P.* Approximating dynamic global illumination in image space. 2009. doi 10.1145/1507149.1507161.
13. *Jimenez J., Wu X.-Ch., Pesce A., et al.* Practical real-time strategies for accurate indirect occlusion. 2016. https://www.activision.com/cdn/research/Practical_Real_Time_Strategies_for_Accurate_Indirect_Occlusion_NEW%20VERSION_COLOR.pdf. (Дата обращения: 5 марта 2025).
14. *Marrs A.* RTXGI: scalable ray traced global illumination in real time. 2020. https://developer.download.nvidia.com/rtx/rtxgi/NVIDIA-RTXGI-03-23-2020-v2.pdf. (Дата обращения: 6 марта 2025).
15. *McGuire M.* Dynamic diffuse global illumination. 2019. https://morgan3d.github.io/articles/2019-04-01-ddgi/. (Дата обращения: 6 марта 2025).
16. Quake lightmaps. 2015. https://jbush001.github.io/2015/06/11/quake-lightmaps.html. (Дата обращения: 6 марта 2025).
17. *Green R.* Spherical harmonic lighting: the gritty details. 2003. https://3dvar.com/Green2003Spherical.pdf. (Дата обращения: 6 марта 2025).
18. *Kahl B.* Hardware acceleration of progressive refinement radiosity using Nvidia RTX. 2023. https://arxiv.org/abs/2303.14831. (Дата обращения: 6 марта 2025).
19. *Щербаков А.С., Фролов В.А., Галактионов В.А.* Виртуальные площадки в алгоритме излучательности // Труды Института системного программирования РАН. 2022. **34**, № 3. 47–60. doi 10.15514/ISPRAS-2022-34(3)-4.
20. *Hanrahan P., Salzman D., Aupperle L.* A rapid hierarchical radiosity algorithm // Computer Graphics. 1991. **25**, N 4. 197–206.
21. *Damez C., Sillion F.X.* Space-time hierarchical radiosity. 1999. https://inria.hal.science/inria-00527746/file/SpaceTimeRad.pdf. Cited March 6, 2025.

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

22. *Щербаков А.С., Фролов В.А.* Матричные преобразования для эффективной реализации алгоритма излучательности на графических процессорах // Светотехника. 2018. № 3. 43–47.
23. *Shcherbakov A., Frolov V.* Dynamic radiosity // Computer Science Research Notes. 2019. 83–90. doi 10.24132/CSRN.2019.2901.1.10.
24. *Щербаков А.С., Фролов В.А., Галактионов В.А.* Темпоральный алгоритм излучательности для 3D-сцен произвольной детализации. Препринт № 76. М.: ИПМ им. М.В. Келдыша, 2023. doi 10.20948/prepr-2023-76.
25. *Walker A.J.* New fast method for generating discrete random numbers with arbitrary frequency distributions // Electronics Lett. 1974. **10**, N 8. 127–128. doi 10.1049/el:19740097.
26. *Yang L., Liu S., Salvi M.* A survey of temporal antialiasing techniques. Computer Graphics Forum. 2020. **39**, N 2. 607–621. doi 10.1111/cgf.14018.
27. Temporal Super Resolution. https://docs.unrealengine.com/5.2/en-US/temporal-super-resolution-in-unreal-engine/. (Дата обращения: 6 марта 2025).
28. *Lehmann H.-P., Hübschle-Schneider L., Sanders P.* Weighted random sampling on GPUs. 2021. https://arxiv.org/abs/2106.12270. (Дата обращения: 6 марта 2025).

Поступила в редакцию
2 марта 2025 г.

Принята к публикации
3 марта 2025 г.

Информация об авторе

Александр Станиславович Щербаков — мл. науч. сотр.; Московский государственный университет имени М. В. Ломоносова, факультет вычислительной математики и кибернетики, Ленинские горы, 1, стр. 52, 119991, Москва, Российская Федерация.

References

1. F. X. Sillion and C. Puech, *Radiosity and Global Illumination* (Morgan Kaufmann, San Francisco, 1994).
2. Y. O'Donnell, FrameGraph: Extensible Rendering Architecture in Frostbite. Game Developers Conference. 2017. https://www.gdcvault.com/play/1024612/FrameGraph-Extensible-Rendering-Architecture-in. Cited March 5, 2025.
3. G. Wihlidal, Halcyon: Rapid Innovation Using Modern Graphics. https://media.contentapi.ea.com/content/dam/ea/seed/presentations/wihlidal2019-rebootdevelopblue-halcyon-rapid-innovation.pdf. Cited March 5, 2025.
4. R. Sandu and A. Shcherbakov, "A Resource Allocation Algorithm for a History-Aware Frame Graph," J. WSCG **31** (1–2), 63–70 (2023). doi 10.24132/jwscg.2023.7.
5. R. Sandu and A. Shcherbakov, "GPU Cache Flush Minimization in Render Graph Systems," J. WSCG **32** (1–2), 71–78 (2024). doi 10.24132/jwscg.2024.8.
6. Using Resource Barriers to Synchronize Resource States in Direct3D 12. https://learn.microsoft.com/en-us/windows/win32/direct3d12/using-resource-barriers-to-synchronize-resource-states-in-direct3d-12. Cited March 5, 2025.
7. Understanding Vulkan Synchronization. https://www.khronos.org/blog/understanding-vulkan-synchronization. Cited March 5, 2025.
8. memoryBarrier(scope:after:before:). https://developer.apple.com/documentation/metal/mtlrendercommandencoder/memorybarrier(scope:after:before:). Cited March 5, 2025.
9. C. Wyman, S. Hargreaves, P. Shirley, and C. Barré-Brisebois, Introduction to DirectX RayTracing. 2018. https://intro-to-dxr.cwyman.org/. Cited March 5, 2025.
10. M. Stich, Introduction to NVIDIA RTX and DirectX Ray Tracing. 2018. https://developer.nvidia.com/blog/introduction-nvidia-rtx-directx-ray-tracing/. Cited March 5, 2025.
11. C. Crassin, F. Neyret, M. Sainz, et al., *Interactive Indirect Illumination Using Voxel Cone Tracing: A Preview.* 2011. doi 10.1145/1944745.1944787.
12. T. Ritschel, T. Grosch, and H.-P. Seidel, "Approximating Dynamic Global Illumination in Image Space," 2009. doi 10.1145/1507149.1507161.
13. J. Jimenez, X.-Ch. Wu, A. Pesce, et al., *Practical Real-Time Strategies for Accurate Indirect Occlusion.* 2016. https://www.activision.com/cdn/research/Practical_Real_Time_Strategies_for_Accurate_Indirect_Occlusion_NEW%20VERSION_COLOR.pdf. Cited March 5, 2025.

2025, **26** (2), 99–110. doi 10.26089/NumMet.v26r207

14. A. Marrs, RTXGI: Scalable Ray Traced Global Illumination in Real Time. 2020. https://developer.download.nvidia.com/rtx/rtxgi/NVIDIA-RTXGI-03-23-2020-v2.pdf. Cited March 6, 2025.
15. M. McGuire, Dynamic Diffuse Global Illumination. 2019. https://morgan3d.github.io/articles/2019-04-01-ddgi/. Cited March 6, 2025.
16. Quake Lightmaps. 2015. https://jbush001.github.io/2015/06/11/quake-lightmaps.html. Cited March 6, 2025.
17. R. Green, *Spherical Harmonic Lighting: The Gritty Details*. 2003. https://3dvar.com/Green2003Spherical.pdf. Cited March 6, 2025.
18. B. Kahl, *Hardware Acceleration of Progressive Refinement Radiosity Using Nvidia RTX*. 2023. https://arxiv.org/abs/2303.14831. Cited March 6, 2025.
19. A. Shcherbakov, V. Frolov, and V. Galaktionov, "Virtual Patches Approach for Radiosity," Proceedings of ISP RAS **34** (3), 47–60 (2022). doi 10.15514/ISPRAS-2022-34(3)-4.
20. P. Hanrahan, D. Salzman, and L. Aupperle, "A Rapid Hierarchical Radiosity Algorithm," Comput. Graph. **25** (4), 197–206 (1991).
21. C. Damez and F. X. Sillion, "Space-Time Hierarchical Radiosity," 1999. https://inria.hal.science/inria-00527746/file/SpaceTimeRad.pdf. Cited March 6, 2025.
22. A. S. Shcherbakov and V. A. Frolov, "Matrix Transformations for Effective Implementation of Radiosity Algorithm Using Graphic Processors," Svetotekhnika, No. 3, 43–47 (2018) [Light Eng. **27** (2), 105–110 (2019)]. doi 10.33383/2017-081.
23. A. Shcherbakov and V. Frolov, "Dynamic Radiosity," Computer Science Research Notes (2019), pp. 83–90. doi 10.24132/CSRN.2019.2901.1.10.
24. A. S. Shcherbakov, V. A. Frolov, and V. A. Galaktionov, *Temporal Radiosity Method for 3D-Scenes of Arbitrary Detailization*, Preprint No. 76 (Keldysh Institute of Applied Mathematics, Moscow, 2023). doi 10.20948/prepr-2023-76.
25. A. J. Walker, "New Fast Method for Generating Discrete Random Numbers with Arbitrary Frequency Distributions," Electronics Lett. **10** (8), 127–128 (1974). doi 10.1049/el:19740097.
26. L. Yang, S. Liu, and M. Salvi, "A Survey of Temporal Antialiasing Techniques," Comput. Graph. Forum **39** (2), 607–621 (2020). doi 10.1111/cgf.14018.
27. Temporal Super Resolution. https://docs.unrealengine.com/5.2/en-US/temporal-super-resolution-in-unreal-engine/. Cited March 6, 2025.
28. H.-P. Lehmann, L. Hübschle-Schneider, and P. Sanders, "Weighted Random Sampling on GPUs," 2021. https://arxiv.org/abs/2106.12270. Cited March 6, 2025.

Received
March 2, 2025

Accepted for publication
March 3, 2025

Information about the author

Alexandr S. Shcherbakov — Junior Scientist; Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics, Leninskie Gory, 1, building 52, 119991, Moscow, Russia.