*ПРОГРАММИРОВАНИЕ, 2023, № 3, с. 37–48*

## КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

*УДК 04.092*

# ИСПОЛЬЗОВАНИЕ МНОГОУРОВНЕВЫХ ХЭШ-ТАБЛИЦ ДЛЯ УСКОРЕНИЯ ПРОЦЕССА РЕНДЕРИНГА

**© 2023 г. Д. Д. Жданов[a,*] (ORCID: 0000-0001-7346-8155), А. И. Лысых[a,**] (ORCID: 0000-0002-2437-5275), Р. Р. Халимов[a,***] (ORCID: 0000-0003-1923-4360), И. Е. Кинев[a,****] (ORCID: 0000-0003-2929-1203), А. Д. Жданов[a,*****] (ORCID: 0000-0002-2569-1982)**

*[a]Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Россия, 197101 Санкт-Петербург, Кронверкский пр., 49*

**e-mail: ddzhdanov@mail.ru*

***e-mail: lysykhai@ya.ru*

****e-mail: khalimov.ruslan@mail.ru*

*****e-mail: igorkinevitmo@gmail.com*

******e-mail: andrew.gtx@gmail.com*

Поступила в редакцию 10.01.2023 г.
После доработки 18.01.2023 г.
Принята к публикации 22.01.2023 г.

Проведен анализ методов реалистичного рендеринга с точки зрения эффективности расчета яркостей каустического и вторичного освещений. В качестве основного подхода для реализации реалистичного рендеринга был выбран метод двунаправленной прогрессивной трассировки лучей с обратными фотонными картами. Проведен анализ основных причин, снижающих производительность данного метода. Показано, что главным фактором, снижающим его производительность, является медленный доступ к данным фотонных карт. Рассмотрены различные варианты построения ускоряющих пространственных структур, исследованы их преимущества и недостатки. В качестве основных подходов были выбраны регулярная пространственная решетка и бинарное kd-дерево. Пространственная решетка обеспечивает высокую скорость доступа к данным при низкой адаптивности разбиения фотонной карты. Kd-дерево обеспечивает высокую пространственную адаптивность разбиения карты при низкой скорости доступа к данным. Предложено комбинированное решение, объединяющее адаптивность kd-дерева с высокой скоростью доступа к данным пространственной решетки. Для этого регулярная решетка накладывается на kd-дерево, построенное по принципу пространственного деления области фотонов на геометрически равные половины. Для уменьшения объемов памяти было предложено, во-первых, использовать многоуровневые пространственные решетки, накладываемые на выбранные узлы kd-дерева, и, во-вторых, для уменьшения объема памяти ускоряющей структуры хранить пространственные решетки в виде хэш-таблиц. В результате был предложен и реализован новый тип пространственных ускоряющих структур, представляющих собой дерево хэш-таблиц. Для разработанной пространственной структуры были реализованы методы поиска ближайших фотонов, сферы интегрирования которых покрывают точку освещения, и методы поиска пересечения сегмента луча со сферами интегрирования фотонов. Разработанные программные решения были реализованы в программном комплексе Lumicept, и для ряда базовых сцен было произведено сравнение скорости работы предложенного метода с методом, основанным на бинарном дереве, имеющемся в Lumicept. Сравнение показало, что новый метод может повысить общую производительность процедуры рендеринга более чем на 40%.

**DOI:** 10.31857/S013234742303007X, **EDN:** DESNPJ

## 1. ВВЕДЕНИЕ

Реалистичная визуализация является неотъемлемой частью задач построения систем виртуального прототипирования, создания устройств виртуальной, дополненной и смешанной реальностей, обучения нейронных сетей, разработки видеоигр, создания визуальных эффектов в кинематографии и в ряде других областей человеческой деятельности. Несмотря на то, что для решения проблемы реалистичной визуализации существует большое число подходов, они, как правило, решают ограниченный круг задач, связанных, например, с расчетом прямого или вторичного излучения методами однонаправленной или двунаправленной стохастической трассировки лучей [1, 2]. Особо следует выделить методы, основанные на интегрировании видимой яркости

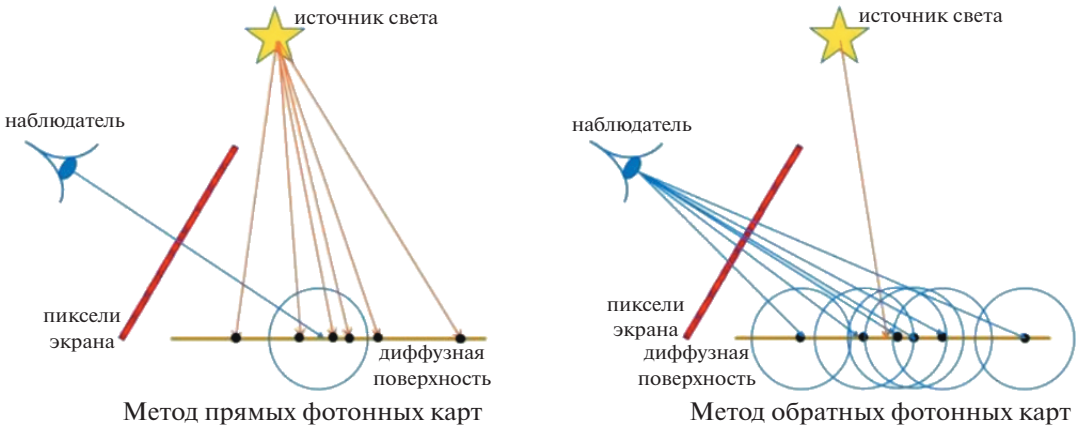


**Рис. 1.** Сбор освещенности с фотонной карты при прямом и обратном методе.

методом Метрополиса, в основе которых лежит интегрирование по схеме Марковских цепей [3–7]. Эти методы обеспечивают высокую эффективность расчета вторичного излучения. Основной недостаток перечисленных выше методов связан с проблемой расчета яркости каустического освещения. В предложенной работе исследовались методы, основанные на фотонных картах, которые позволяют физически корректно рассчитывать сложные сцены с каустическим и вторичным освещением. Однако, несмотря на видимые преимущества методов фотонных карт они имеют ряд серьезных недостатков. Это, во-первых, смещенность метода, вызванная конечным радиусом сфер интегрирования, и, во-вторых, ресурсоемкий процесс поиска пересечения луча со сферами интегрирования.

В данной работе исследуется возможность ускорения процесса рендеринга, использующего метод фотонных карт. Алгоритм вычисления глобальной освещенности с помощью фотонных карт был представлен в работе Х. Йенсена [8] и состоит из трех основных шагов: трассировка фотонов, построение фотонных карт и сбор освещенности с фотонов. Трассировка фотонов осуществляется методом прямой стохастической трассировки лучей. Для построения фотонных карт и эффективного поиска пересечения лучей зрения с фотонами используются ускоряющие структуры пространственного разбиения. Основные требования к программной организации пространственных структур – это : минимально возможный объем дополнительной памяти для хранения и обеспечения доступа к фотонам, быстрое построение пространственной структуры и быстрый доступ к фотонам в ограниченной области пространства сцены.

Сбор освещенности представляет собой трассировку путей от камеры-наблюдателя до геометрии, освещенной фотонами. При попадании $i$-го луча, переносящего световой поток $F_i(\vec{x},\vec{v}')$ в направлении $\vec{v}'$, на поверхность сцены происходит сбор освещенности в области $\vec{x}$, ограниченной сферой интегрирования с радиусом $R$, и последующий расчет видимой яркости $L(\vec{x},\vec{v})$ в этой области в направлении $\vec{v}$:

$$L(\vec{x},\vec{v}) = \sum_{i=all\ photons} \frac{F_i(\vec{x},\vec{v}')}{(\pi R)^2} BRDF(\vec{x},\vec{v},\vec{v}'), \quad (1)$$

где $BRDF(\vec{x},\vec{v},\vec{v}')$ – значение двунаправленной функции рассеивания в области сбора освещенности $\vec{x}$ при освещении в направлении $\vec{v}'$ и наблюдении в направлении $\vec{v}$.

Позднее в работе В. Хавран и др. [9] и в следующих работах [10, 11] был представлен метод обратных прогрессивных фотонных карт, который решает основные недостатки прямого метода, а именно: высокая плотность фотонов вблизи источников освещения, отсутствие критерия, ограничивающего число фотонов, выпускаемых источниками света, и накопление фотонов в скрытых от наблюдателя областях. В этом методе фотоны испускаются из виртуальной камеры, формируя область видимости, а сферы интегрирования сохраняются в фотонной карте для каждого фотона, размер которого рассчитывается относительно расстояния до точки наблюдения. Формула для расчета яркости вторичного и каустического освещения аналогична (1), за исключением того, что сбор световой энергии осуществляется отдельно для каждой точки изображения. Отличия методов расчета яркости глобального освещения для прямых и обратных фотонных карт представлены на рис. 1.

Несмотря на то, что метод стохастических обратных прогрессивных фотонных карт ускоряет процесс расчета глобального освещения, операции обработки запросов к данным структур пространственного разбиения составляют большую часть времени его работы. В предложенной рабо-

те исследуется возможность ускорения процесса рендеринга изображений методом фотонных карт путем минимизации времени доступа к фотонным картам.

## 2. ТИПЫ ОРГАНИЗАЦИИ СТРУКТУР ПРОСТРАНСТВЕННОГО РАЗБИЕНИЯ

Метод фотонных карт требует эффективной структуры пространственного разбиения для хранения и быстрого поиска фотонов, попадающих в сферу интегрирования. Подобные структуры пространственного разбиения также применяются для обеспечения высокой скорости трассировки лучей в сцене.

Структуры пространственного разбиения делят пространство сцены на ячейки, каждая из которых может содержать список фотонов, находящихся внутри нее, элементы геометрии или другие объекты, требующие специальной обработки. Использование таких структур позволяет ускорить поиск объектов, расположенных в локальной области сцены.

Существует множество структур пространственного разбиения, позволяющих добиться увеличения скорости доступа к данным.

### *2.1. Регулярное разбиение пространства*

Регулярная сетка является простейшим методом разбиения пространства [12]. Разрешение такой сетки фиксировано и задается при ее построении. Поскольку все ячейки структуры имеют одинаковый размер, а положение сетки в пространстве определено – по координатам точки можно легко вычислить ячейку, которой она принадлежит. Хранение такой структуры в памяти, как правило, осуществляется с помощью хеш-таблиц, где для доступа к ячейке используются ее координаты в сетке, это обеспечивает быструю скорость доступа к необходимым ячейкам [13]. Кроме того, пустые ячейки можно не хранить в памяти, что сокращает необходимые ресурсы.

Ввиду того, что разрешение сетки фиксируется на этапе ее построения, такая структура обладает недостаточной адаптивностью. При большом размере ячейки, а следовательно, большой концентрации объектов, их поиск будет осуществляться медленно. При увеличении разрешения структура займет больший объем памяти, что в большинстве случаев недопустимо.

### *2.2. Kd-деревья*

Одной из часто используемых структур пространственного разбиения, позволяющих добиться быстрого доступа к данным, локализованным в пространстве сцены, является kd-дерево. Это бинарное дерево, образованное путем последовательного разделения пространства плоскостями.

Деление ячеек плоскостями может осуществляться различными способами: ровно посередине, по количеству объектов в одной и другой части ячейки [14] или более сложным способом, основанным на оценке времени доступа к элементу дерева.

По сравнению с регулярной сеткой бинарное дерево позволяет добиться хорошей адаптивности при неравномерном распределении объектов, но может замедлить доступ к ячейкам при большой глубине дерева, поскольку каждый переход между его узлами требует отдельного обращения к памяти, затрудняя использование кэша процессора. Замедление доступа к листьям дерева усиливается в многопоточном режиме, поскольку несколько потоков могут запрашивать данные, разбросанные в адресном пространстве.

### *2.3. Окто-деревья*

Другой ускоряющей структурой разбиения являются окто-деревья. В данном случае каждая ячейка пространства делится на 8 дочерних путем разделения сразу тремя плоскостями по каждой из осей координат [15]. Как правило, деление ячеек окто-дерева осуществляется через точку по центру ячейки.

По сравнению с kd-деревом такое дерево имеет меньшую глубину, что позволяет увеличить скорость доступа к его ячейкам, однако такая структура требует большей памяти для хранения данных и имеет меньшую адаптивность.

### *2.4. BVH-деревья*

Кроме описанных выше видов деревьев существуют деревья, основанные на иерархии ограничивающих объемов (BVH-деревья) [16]. Данные методы пространственной индексации широко применяются для трассировки лучей, а также для отрисовки сцены, так как обладают хорошей адаптивностью и простотой. Основная идея таких методов заключается в том, чтобы представить сцену в виде подмножества примитивов, которые определяются и сохраняются в узлах на стадии создания дерева. Существует большое количество разновидностей таких деревьев, например: Sphere tree [17], AABB tree [18], OBB tree [19] и т.д. Для разбиения сцены в основном используются две стратегии, которые можно смешивать: разделение по объектам (object split) и пространственное разбиение (spatial split [20]). В первом случае нужно найти и разделить исходное множество примитивов на два подмножества, вычислить ограничивающий объем каждого подмножества, и, если примитив содержится в определенном узле полностью, то он туда и идет. Во втором случае

определяется некая плоскость, по которой происходит деление пространства на две части. В результате примитив добавляется в тот дочерний узел, который он пересекает. Наиболее часто используемой и простой в реализации модификацией BVH-дерева является AABB-дерево, в котором разделение происходит по ограничивающей рамке (AABB). Таким образом, в листьях будут содержаться объекты сцены, которые потом можно будет относительно быстро найти в процессе рендеринга. Достоинствами BVH-деревьев являются: относительная простота реализации, хорошая адаптивность (так как их можно частично перестроить при изменении сцены), быстрое построение структуры и быстрая трассировка. Кроме того, для одинакового набора фотонных карт BVH-дерево будет иметь меньшую глубину, чем kd-дерево, вследствие чего уменьшится количество кэш-промахов при поиске на CPU. Однако существенными недостатками данного метода для поиска фотонов в фотонной карте являются отсутствие критерия для разбиения фотонов по примитивам, возможность пересечения граничных областей и более сложный (чем у kd-дерева) алгоритм поиска фотонов.

### *2.5. Разбалансированные деревья*

В методах прямых фотонных карт для каждого локального расчета яркости необходимо найти k ближайших соседей в kd-дереве. Эта операция является высокозатратной и может занимать больше времени, чем собственно трассировка лучей. Для частичного решения данной проблемы в статье [21] было предложено использовать разбалансированные деревья с эвристикой воксельных объемов (Voxel Volume Heuristic) для оценки плоскостей разбиения и нахождения наилучшего их положения. Авторы утверждают, что данный метод позволяет более эффективно находить k ближайших соседей, при этом без каких-либо приближений. Поскольку бинарный поиск эффективно работает только в сбалансированном дереве, в котором все узлы имеют одинаковые вероятности доступа, то это условие не выполняется при неравномерном распределении фотонов и примитивов в сцене. Для решения проблемы неэффективного доступа в случае неравномерного распределения фотонов авторы модифицируют эвристику площадей поверхности (Surface Area Heuristic) для создания разбалансированного дерева, которое более эффективно работает при неоднородном распределении фотонов в сцене в случае относительно малого среднего радиуса сбора. Таким образом, данный метод эффективно работает в сценах, содержащих сложную каустику, но дает незначительный прирост производительности в сценах, где преобладает диффузное рассеивание. Кроме того, время, занимаемое на построение данного дерева, больше чем на построение kd-дерева.

### *2.6. Комбинированное решение*

Для нашего исследования были выбраны сбалансированные kd-деревья, так как они широко применяются в различных методах рендеринга изображений, обладают высокой адаптивностью и простотой реализации. В работе предлагается использовать комбинированную ускоряющую структуру, которая сочетает в себе преимущества kd-дерева и регулярных структур на основе хэш-таблиц. На определенных уровнях kd-дерева предлагается построить хеш-таблицы, позволяющие непосредственно обращаться к узам этого дерева. Такая структура позволит сохранить адаптивность kd-дерева и увеличит скорость доступа к его данным. Для уменьшения количества памяти, необходимой для хранения структуры, а также для увеличения скорости трассировки предлагается использовать хеш-таблицы на различных уровнях дерева. Анализируя структуру дерева, предлагается выбирать наиболее подходящие для регуляризации уровни, уменьшая среднее время доступа к ячейкам, тем самым повышая эффективность структуры. Принцип построения данной структуры проиллюстрирован на рис. 2.

## 3. ПОСТРОЕНИЕ КОМБИНИРОВАННОЙ УСКОРЯЮЩЕЙ СТРУКТУРЫ

Структура комбинирует два последовательно выполняющихся процесса: построение kd-дерева и создание хеш-таблиц. Для построения kd-дерева необходимо иметь основные данные фотонной карты: координаты и радиусы фотонов. Далее на месте kd-дерева будет построена древовидная структура из хеш-таблиц. Для того, чтобы сделать это оптимальным образом, во время построения хеш-таблиц производится анализ структуры дерева для определения подходящих уровней, на которых будут построены хеш-таблицы.

Первый этап построения структуры начинается с определения ограничивающего параллелепипеда по минимальному и максимальному значению координат накопленных фотонов. Этот параллелепипед будет являться корневым узлом kd-дерева. После создания этого узла запускается повторяющийся цикл, условием окончания которого является проверка на удовлетворение критерия максимального числа фотонов в ячейке. Повторяющийся для каждой ячейки цикл разделения состоит из трех действий: в первую очередь выбирается ось координат, перпендикулярно которой будет происходить разделение ячейки. В данном случае эта плоскость раздела всегда проходит через центр ячейки, поскольку на уровни дерева будет наложена регулярная решетка,

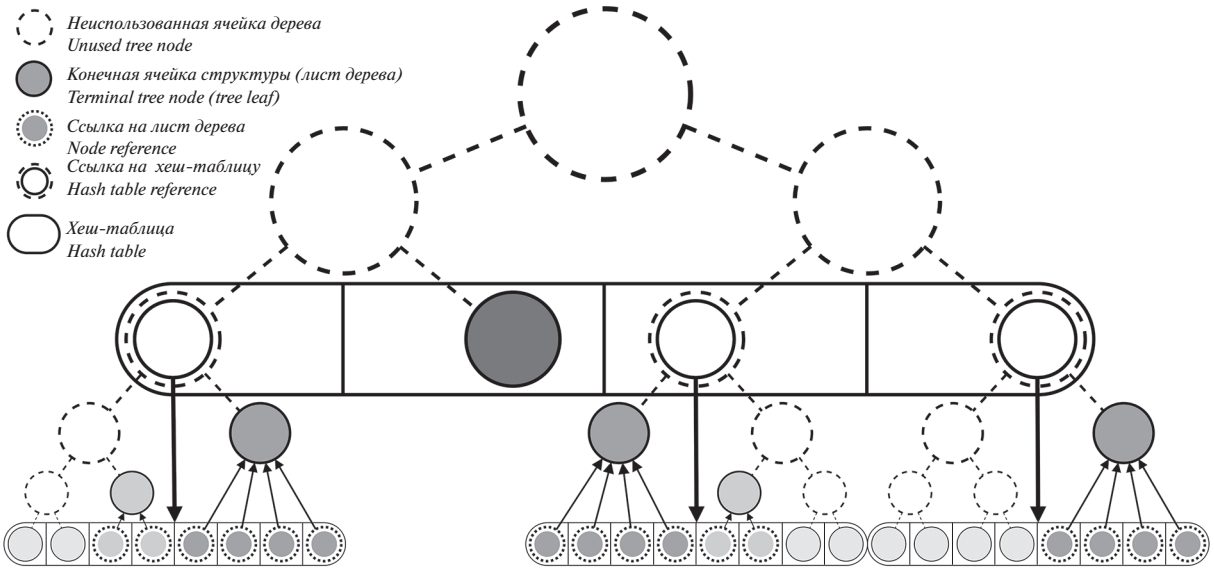


**Рис. 2** Схема ускоряющей структуры, построенной на kd-дереве и хэш-таблице.

для которой границы ячеек строго определены. Далее происходит сортировка фотонов ячейки по областям, образованным секущей плоскостью. После этого создаются две новые ячейки-потомка для текущего узла дерева, в которые добавляются отсортированные фотоны. В случае, если число фотонов в ячейке достигнет требуемого значения, алгоритм построения дерева завершается пересчетом радиусов фотонов в соответствии с плотностью их распределения и формированием ограничивающего параллелепипеда фотонов в каждой ячейке дерева. На рис. 3 представлен алгоритм построения ускоряющей структуры фотонных карт, построенной на kd-дереве и хэш-таблице.

Завершающим этапом построения комбинированной структуры является создание хеш-таблиц на оптимальных уровнях kd-дерева. Критерий оптимальности выбора таких уровней заключается в максимальном заполнении конечными листьями дерева при непревышении некоторой фиксированной глубины. Данный критерий обеспечивает максимальный доступ к фотонам при относительно небольшом размере таблицы. Для определения оптимальных уровней необходимо формировать счетчик числа листьев дерева на каждом его уровне. Данный счетчик может формироваться в процессе построения дерева. В статье [22] было определено оптимальное значение глубины уровня для построения хеш-таблицы при одновременном ее использовании с kd-деревом. Задачей данного исследования является анализ оптимального количества таблиц, а также уровней дерева, на которых они будут размещены.

На рис. 4 представлен алгоритм формирования данных хеш-таблицы для одного узла дерева, не превышающего заданный уровень. Рассматриваются четыре основных случая:

1. узел дерева является пустым листом,

2. узел дерева не является листом,

3. узел дерева непустой лист на заданном уровне,

4. узел дерева непустой лист на уровне выше заданного.

В первом случае ничего не происходит, и хэш-таблица не обновляется. Далее происходит переход на следующий узел дерева, не превышающий заданный.

Во втором случае происходит формирование нового узла хэш-таблицы, обновляются ее параметры, глубина дерева и разрешение таблицы. Далее происходит рекурсивное формирование новой хэш-таблицы для заданного узла kd-дерева, и по окончании этого рекурсивного процесса происходит переход на следующий узел дерева, не превышающий заданный.

В третьем случае формируются индекс решетки, соответствующий данному уровню kd-дерева, и хэш индекс таблицы (остаток от деления индекса на размер хэш-таблицы), которые добавляются в ячейку таблицы. Далее происходит переход на следующий узел дерева, не превышающий заданный.

В четвертом случае ячейка дерева покрывает несколько ячеек регулярной решетки, поэтому происходит сканирование по всем ячейкам регулярной решетки, попадающим в область фотонов ячейки. Для каждой ячейки регулярной решетки формируются ее индекс, соответствующий дан-

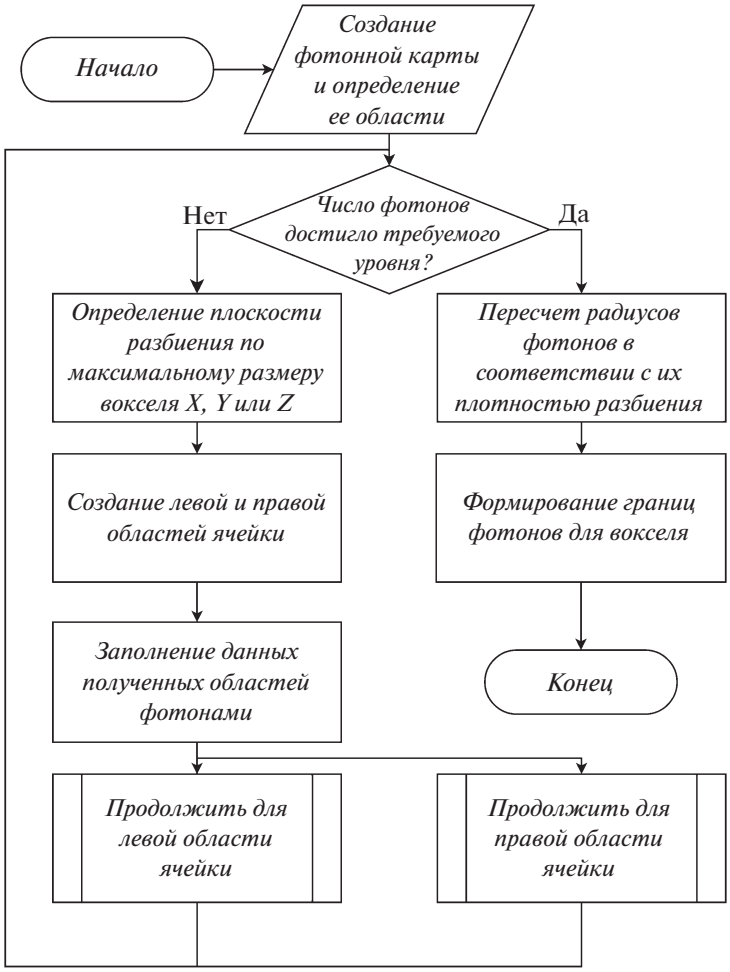


**Рис. 3.** Алгоритм построения kd-дерева.

ному уровню kd-дерева, и хэш-индекс таблицы (остаток от деления индекса на размер хэш-таблицы), которые добавляются в текущую ячейку таблицы. По окончании сканирования всех ячеек решетки происходит переход на следующий узел дерева, не превышающий заданный.

Необходимо отметить, что алгоритм, представленный на рис. 4, не содержит главный цикл обхода по всем ячейкам дерева, который заканчивается на последнем листе дерева. Кроме того, при формировании хэш-таблицы возможны коллизии, перебор которых может быть ускорен с помощью SIMD-операций, позволяющих проводить параллельную проверку сразу восьми ячеек. Такое решение позволяет повысить число коллизий при сохранении времени доступа к данным и уменьшить объем памяти для хранения хэш-таблиц.

В результате данный комбинированный алгоритм формирует дерево хэш-таблиц, которое на практике ограничено, как правило, двумя уровнями, что на порядок сокращает число обращений к разрозненным областям памяти.

## 4. ИСПОЛЬЗОВАНИЕ УСКОРЯЮЩЕЙ СТРУКТУРЫ В ВИДЕ ДЕРЕВА ХЭШ-ТАБЛИЦ

Метод обратных фотонных карт может использоваться как для получения яркости объемного рассеивания, так и для получения яркости на поверхности сцены. В любом из этих случаев предложенная структура пространственного разбиения применяется одновременно и для ускорения трассировки лучей, необходимой при сборе освещенности, и для ускорения поиска фотонов, сферы интегрирования которых попадают в точку пересечения луча с геометрией.

При поиске точки пересечения луча с геометрическим объектом или сферой интегрирования фотона ускоряющие структуры используются для ограничения количества элементов геометрии или сфер интегрирования. Для получения яркости объемного рассеивания необходимо проверять пересечения этим лучом сразу нескольких сфер интегрирования фотонов, находящихся на его пути. В случае поиска фотонов, сферы интегрирования которых покрывают точку пересече-

ния луча с поверхностью сцены, необходимо производить поиск ячейки ускоряющей структуры, содержащей эту точку.

### *4.1. Алгоритм поиска фотонов*

При расчете яркости вторичного и каустического освещений необходимо найти ячейку в ускоряющей структуре фотонной карты, фотонам которой будет передана энергия источника света. Следующий этап – это проверка принадлежности точки пересечения луча с поверхностью со сферами интегрирования фотонов. Исходными данными для поиска ячейки являются только координаты точки пересечения луча с геометрическим объектом.

В регулярных структурах пространственного разбиения поиск ячейки осуществляется с помощью прямого доступа по ее индексу, вычисляемому по координатам начала области, ее размеру и разрешению. При использовании хеш-таблиц поиск ячеек осуществляется путем формирования индекса в регулярной решетке и соответствующего ей хэш-индекса. Если при обращении к ячейке хэш-таблицы возникает коллизия (ячейка хэш-таблицы содержит ряд данных, имеющих этот хэш-индекс), то она разрешается простым перебором и сравнением элементов таблицы с индексом решетки. Для ускорения процедуры перебора ее можно осуществлять с помощью SIMD-инструкций, которые выполняют проверку сразу восьми индексов. Возможны два результата: “элемент не найден” и “элемент найден”. Первый случай говорит о том, что область, в которую мы попали, пустая, и, следовательно, для данного луча яркость не формируется. Второй случай говорит о том, что мы могли попасть в область с фотонами, и необходимо получить данные о фотонах, находящихся в этой области. Поскольку данная реализация предполагает дерево хэш-таблиц, то результатом поиска в хэш-таблице может быть как конечный лист, содержащий список фотонов и их параметров, так и хэш-таблица следующего уровня, содержащая свою регулярную решетку. В последнем случае процедура поиска продолжается, пока конечный уровень не будет достигнут. В большинстве случаев уровень хэш-таблиц не превышает двух. Алгоритм поиска фотонов в многоуровневой хэш-таблице представлен на рис. 5.

Конечным этапом поиска фотонов является проверка на принадлежность точки пересечения сферам интегрирования фотонов и формирование соответствующего списка фотонов. Проверка на принадлежность точки сфере интегрирования также может выполняться с помощью SIMD-инструкций, что позволяет существенно ускорить процедуру проверки.

### *4.2. Алгоритм трассировки лучей*

Трассировка лучей в структурах с регулярным разбиением пространства выполняется с помощью различных модификаций метода быстрого обхода сетки [23]. Регулярные разбиения позволяют добиться высокой эффективности трассировки только в сценах с равномерным распределением объектов. В случае использования хеш-таблиц трассировка лучей в них осуществляется аналогичным образом, поскольку пространство разбито регулярно. Наличие нескольких уровней таких разбиений позволяет добиться лучших результатов для сцен с неравномерным распределением объектов.

Трассировка лучей в пространственных структурах, построенных с помощью бинарных деревьев, из одной ячейки в другую ячейку предполагает перемещение по древовидной структуре как вверх, так и вниз. Это осуществляется тремя основными способами: классическим алгоритмом, использующим стек [24], алгоритмом kd-restart и алгоритмом kd-backtrack [25]. Каждый из алгоритмов обладает как преимуществами, так и недостатками. Алгоритм на основе стека требует хранения и оперирования стеком для каждого обрабатываемого луча, что является неэффективным в случае массового распараллеливания из-за частых обращений к памяти и необходимости работать с большим количеством стеков. Алгоритм kd-restart предполагает отсутствие перемещений по дереву вверх, что позволяет полностью избежать использования стека, однако требует дополнительных операций перемещения по дереву. Поэтому в случае использования многоуровневого дерева эффективность этого алгоритма заметно снижается. Алгоритм kd-backtrack хранит в каждой ячейке дерева не только ссылку на потомков, но и ссылку на родителя, а также bounding-box текущего уровня. Очевидно, этот алгоритм требует большего количества памяти для хранения структуры, однако позволяет избавиться от использования стека и в ряде случаев уменьшить количество необходимых для трассировки операций движения по дереву.

Структура, комбинирующая kd-дерево с хэш-таблицами, позволяет объединить в себе преимущества алгоритмов обеих пространственных структур. Этот алгоритм представляет собой последовательный обход древовидной структуры хеш-таблиц, внутри которых осуществляется обход ячеек таблицы, пересекаемых лучом. В простейшем виде этот алгоритм имеет рекурсивный вид, представленный на рис. 6.

Трассировка лучей представляет собой поиск пересечений луча со сферами интегрирования фотонов, находящихся на глубине трассы луча. Для выполнения трассировки требуется произвести последовательный обход всех ячеек про-

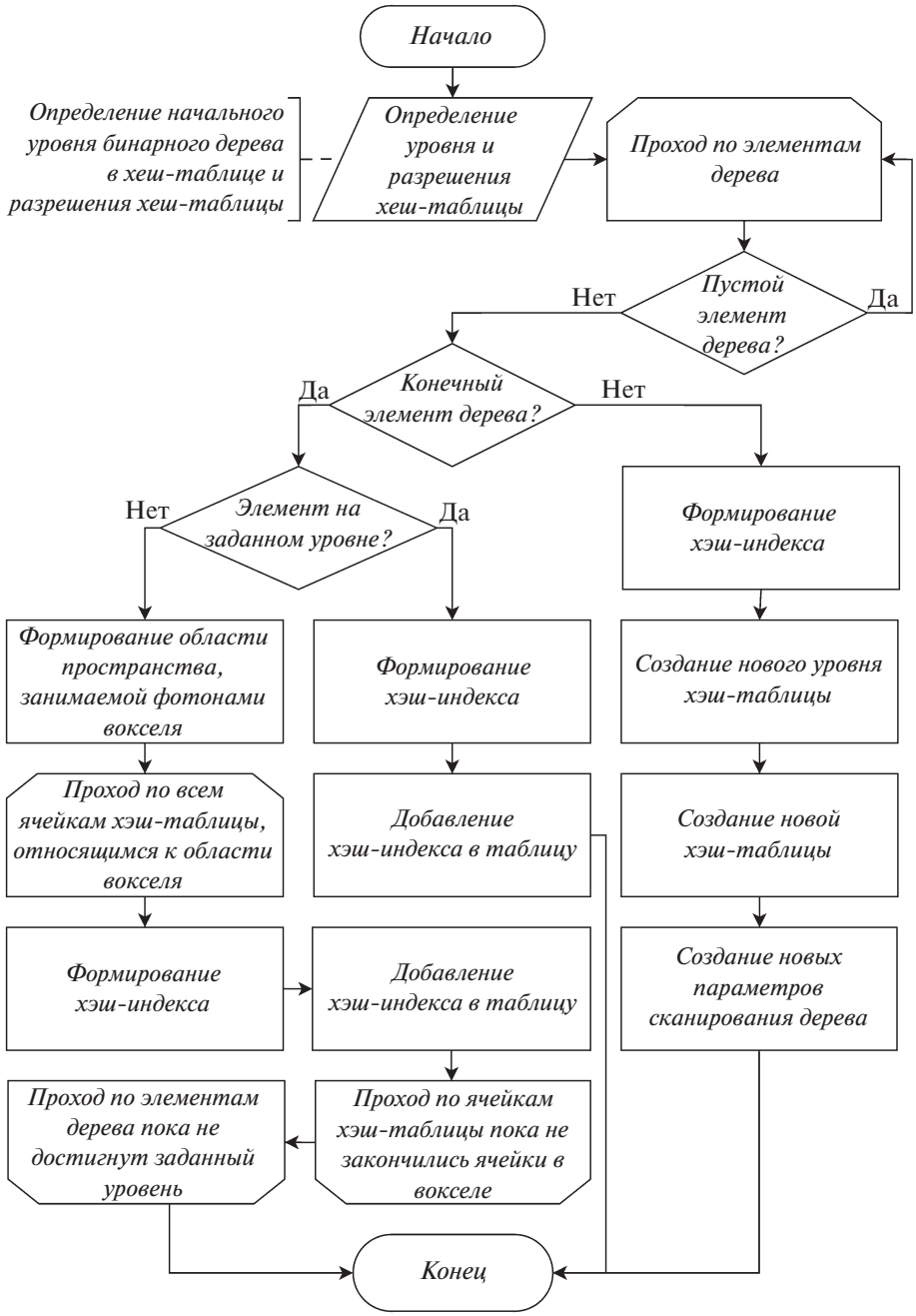


**Рис. 4.** Алгоритм построения хеш-таблиц.

странственной структуры, находящихся на глубине трассы луча, и проверить все пересечения луча со сферами внутри каждой из ячеек. Для начала обхода необходимо определить координаты точки пересечения луча с областью определения пространственной структуры. Если источник луча находится внутри ограничивающего параллелепипеда, координатами точки будут являться координаты источника луча. Если пересечения с параллелепипедом нет, трассировка завершается.

После нахождения начальной точки трассы луча происходит поиск ячейки, содержащей эту точку (алгоритм поиска описан в предыдущем разделе). В каждой ячейке пространственной структуры определяется локальная глубина трассы луча в ячейке и на этой глубине осуществляется поиск точек пересечения луча со сферами интегрирования фотонов ячейки. Формируется список пересеченных фотонов.

Алгоритм движения луча между ячейками осуществляется следующим образом. На текущем

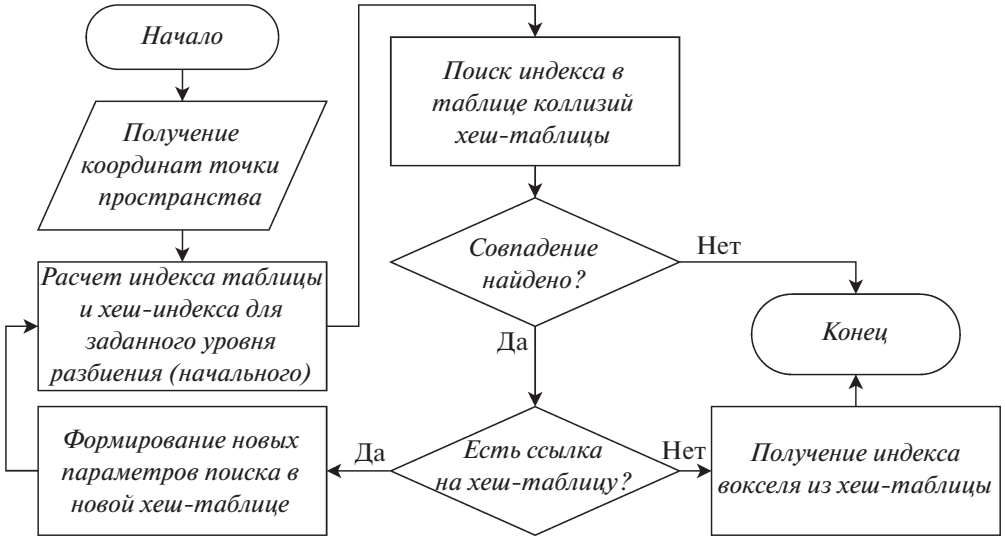


**Рис. 5.** Алгоритм поиска фотонов.

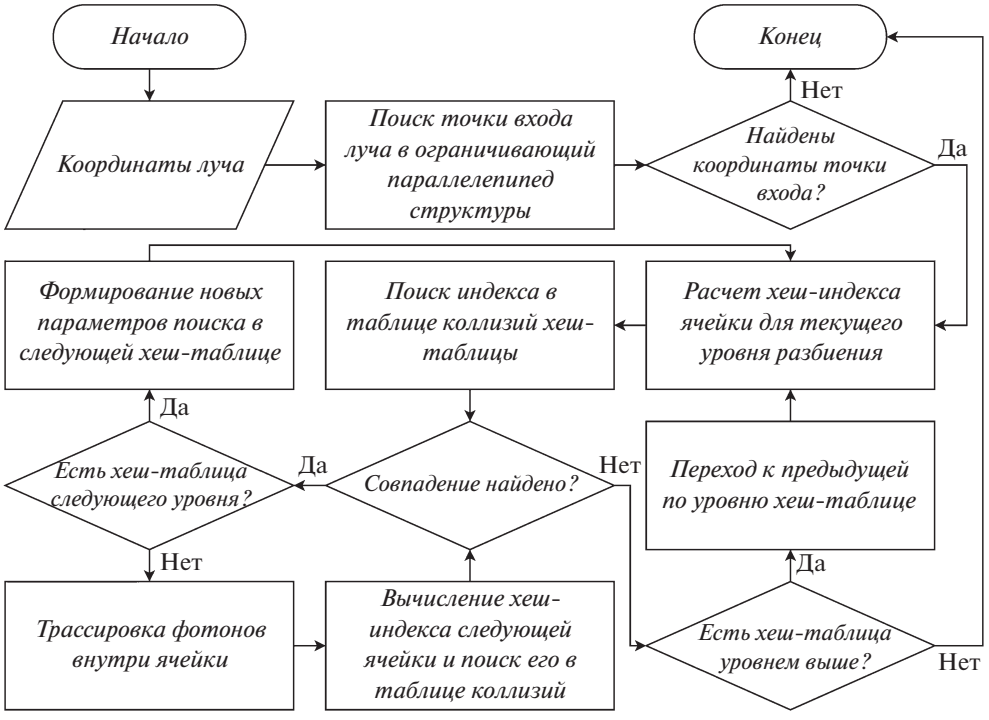


**Рис. 6.** Алгоритм трассировки фотонов.

уровне хеш-таблицы в области ее определения происходит инициализация быстрого обхода регулярной сетки. Для этого рассчитываются расстояния до ближайшей границы области и до границы ячейки. Затем в пределах полученной глубины происходит последовательный обход всех ячеек. Если полная глубина трассы луча не была выбрана на данном уровне хэш-таблицы, то происходит переход на смежную таблицу методом, аналогичным движению в kd-дереве. Алгоритм продол-

**Рис. 7.** Тестовые сцены: Cornell Box слева, Room2 справа сверху, Light Guide справа снизу.

жает движение по дереву хэш-таблиц, пока луч либо не покинет область определения пространственной структуры, либо не исчерпает свою глубину.

## 5. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ АЛГОРИТМА ДЕРЕВА ХЕШ-ТАБЛИЦ В ПРОГРАММНОМ КОМПЛЕКСЕ LUMICEPT

Алгоритм многоуровневых хэш-таблиц был реализован в программном комплексе Lumicept [26] для расчета яркости вторичного и каустического освещений методом двунаправленной стохастической трассировки лучей с использованием обратных фотонных карт. Алгоритм был реализован на центральном процессоре с поддержкой параллельных и распределенных вычислений на серверных компьютерах. Распараллеливание вычислений было реализовано не только для процедуры поиска фотонов, но и для построения фотонных карт. В предыдущей работе [22, 27] было показано, что использование $N$ фотонных карт с $M$ фотонами эффективнее, чем использование одной большой фотонной карты с $N * M$ фотонами. В качестве естественного решения для реализации параллельного алгоритма разбиения одной фотонной карты на $N$ карт было предложено группировать фотонные карты по частям объектов сцены. Предварительно фотонные карты сортируются по числу фотонов и в параллельном режиме первыми начинают обрабатываться карты с наибольшим числом фотонов. В результате последними обрабатываются карты с малым числом фотонов, что позволяет минимизировать потери, связанные с простоем процессов на конечной стадии обработки фотонных карт.

**Таблица 1.** Результаты профилирования процесса рендеринга сцен в системе Lumicept

| Этап рендеринга | Занимаемое процессорное время | | |
|---|---|---|---|
| | Cornell Box | Room2 | Light Guide |
| Поиск фотонов | **24%** | **59%** | **28%** |
| Трассировка фотонов | 11% | 6% | 15% |
| Построение kd-дерева | 2% | 1% | 4% |
| Поиск пересечения с геометрией | 8% | 11% | 22% |
| Расчет яркости | 16% | 5% | 18% |

В программном комплексе был проведен сравнительный анализ разработанного решения, использующего многоуровневые хэш-таблицы, с решением, основанным на использовании бинарного дерева. Сравнение происходило для трех типов сцен: классической Cornell Box, интерьерной сцены Room2 и сцены со светопроводящими элементами Light Guide. Изображения сцен представлены на рис. 7.

Результаты профилирования данных сцен для разбиения фотонов алгоритмом, использующим бинарное дерево, представлены в табл. 1.

Очевидно, что операция поиска пересечения луча с фотонами занимает наибольшее время рендеринга. Поэтому ключевым фактором ускорения процесса рендеринга является ускорение процедуры поиска пересечения луча с фотонами в процессе сбора вторичной и каустической освещенностей. Таким образом, оптимизация структур пространственного разбиения, уменьшающая количество обращений к данным фотонных карт в процессе поиска фотонов, может существенно ускорить процесс рендеринга.

Для сравнения скорости работы предложенного и существующего решений был произведен ряд тестовых вычислений на программном комплексе Lumicept. Анализировалась скорость работы

**Таблица 2.** Сравнение скорости рендеринга сцены Cornell Box при помощи kd-дерева и двухуровневой хэш-таблицы

| Используемый метод | Время рендеринга, с | Количество фаз | Время одной фазы, с | Прирост скорости |
|---|---|---|---|---|
| kd-three | 1819 | 729 | 2.5 | – |
| Two-level hash table | 1810 | 840 | 2.15 | 21% |

**Таблица 3.** Сравнение скорости рендеринга сцены Room2 при помощи kd-дерева и двухуровневой хэш-таблицы

| Используемый метод | Время рендеринга, с | Количество фаз | Время одной фазы, с | Прирост скорости |
|---|---|---|---|---|
| kd-three | 1960 | 63 | 31.11 | – |
| Two-level hash table | 1954 | 90 | 21.7 | 43% |

системы рендеринга с существующим алгоритмом пространственного разбиения фотонных карт в виде kd-дерева и разработанным методом многоуровневых хэш-таблиц. Исследование показало, что для данных сцен хэш-таблицы были ограничены двумя уровнями разбиения. Анализ выполнялся для трех сцен (классической Cornell Box, интерьерной сцены Room2 и сцены со светопроводящими элементами Light Guide), представленных на рис. 7.

Для проведения экспериментов использовался компьютер со следующими характеристиками:

• Процессор Intel Core i5 11400, 2.6 ГГц.

• Оперативная память Kingmax Zeus Dragon, DDR4, 32 ГБ, 2666 МГц.

• Использовано 6 ядер с поддержкой 12 потоков.

При каждом проведении эксперимента проводился рендеринг изображения в течение 30 минут. В качестве показателя скорости использовалось среднее количество выполненных фаз рендеринга. Результаты сравнения для Cornell Box, Room2 и Light Guides представлены в таблицах 2, 3 и 4 соответственно.

Анализ результатов сравнения показал, что реализованная в данной работе структура пространственного разбиения фотонных карт, построенная на многоуровневой хэш-таблице, обеспечивает ускорение процесса рендеринга на 21–43% для большинства типовых сцен.

## 6. ЗАКЛЮЧЕНИЕ

Был проведен анализ существующих методов построения ускоряющих структур для обеспечения быстрого доступа к данным фотонных карт. В качестве двух основных решений были рассмотрены бинарные kd-деревья и регулярные пространственные решетки. Высокая адаптивность и качество пространственного разбиения, обеспечиваемая kd-деревом, имеет существенный недостаток, связанный со значительным временем доступа к данным фотонной карты. Этот недостаток является критическим при выполнении одновременной операции поиска фотонов на ряде потоков на многоядерных компьютерах. С точки зрения времени доступа к данным фотонной карты наиболее подходящим решением является регулярная пространственная решетка. Однако, существенным недостатком данного решения является низкая адаптивность пространственного разбиения, что приводит к значительному увеличению объема памяти для хранения ускоряющей структуры. Предложенное решение комбинирует адаптивность kd-дерева с высокой скоростью доступа к данным фотонной карты, обеспечиваемой регулярной структурой пространственной решетки. Использование многоуровневой хэш-таблицы позволяет уменьшить число коллизий на каждом ее уровне, сохранить адаптивность разбиения фотонов и на порядок сократить количество перемещений по уровням хэш-таблицы. Кроме того, использование SIMD-инструкций позволяет ускорить процесс поиска данных среди коллизий хэш-таблицы.

**Таблица 4.** Сравнение скорости рендеринга сцены Light Guide при помощи kd-дерева и двухуровневой хэш-таблицы

| Используемый метод | Время рендеринга, с | Количество фаз | Время одной фазы, с | Прирост скорости |
|---|---|---|---|---|
| kd-three | 1802 | 305 | 5.91 | – |
| Two-level hash table | 1811 | 373 | 4.85 | 22% |

## ФИНАНСИРОВАНИЕ

Работа была выполнена при финансовой поддержке Российского научного фонда, грант № 22-11-00145.

## СПИСОК ЛИТЕРАТУРЫ

1. *Фролов В.А., Волобой А.Г., Ершов С.В., Галактионов В.А.* Современное состояние методов расчета глобальной освещенности в задачах реалистичной компьютерной графики // Труды Института системного программирования РАН. 2021. Т. 33. № 2. С. 7–48.
2. *Georgiev I., Krivanek J., Davidovic T., Slusallek P.* Light Transport Simulation with Vertex Connection and Merging // ACM Transactions on Graphics. 2012. V. 31. № 6. P. 1–10.
3. *Veach E., Guibas L.J.* Metropolis light transport, in: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques / SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., USA. 1997. P. 65–76. URL: https://doi. org/ https://doi.org/10.1145/258734.258775. doi:. 258775. https://doi.org/10.1145/258734.258775
4. *Wenzel J.* Light Transport on Path-Space Manifolds, Ph.D. thesis. 2013.
5. *Kaplanyan A.S., Hanika J., Dachsbacher C.* The natural-constraint representation of the path space for efficient light transport simulation // ACM Trans. Graph. 2014. V. 33. URL: https://doi.org/10.1145/ 2601097.2601108
6. *Bitterli B., Jakob W., Novák J., Jarosz W.* Reversible jump metropolis light transport using inverse mappings, 2017. arXiv:1704.06835
7. *Gruson A., West R., Hachisuka T.* Stratified Markov Chain Monte Carlo Light Transport // Computer Graphics Forum. 2020. https://doi.org/10.1111/cgf.13935
8. *Jensen H.W.* Global illumination using photon maps / H.W. Jensen // Eurographics Workshop on Rendering techniques. Vienna: Springer, 1996. P. 21–30.
9. *Havran V., Herzog R., Seidel H.P.* Final gathering via reverse photon mapping // Computer Graphics Forum. Oxford, UK and Boston, USA: Blackwell Publishing, 2005. V. 24. № 3. P. 323–332.
10. *Zhdanov A., Zhdanov D.* The Backward Photon Mapping for the Realistic Image Rendering // Proc. 30th Conf. on Computer Graphics and Machine Vision (GraphiCon 2020), CEUR Workshop Proceedings. 2020. V. 2744. P. 1–12.
11. *Zhdanov A.D., Zhdanov D.D.* Progressive backward photon mapping // Programming and Computer Software. 2021. V. 47. № 3. P. 185–193.
12. *Bentley J.L., Friedman J.H.* Data structures for range searching // ACM Computing Surveys (CSUR). 1979. V. 11. № 4. P. 397–409.
13. *Toshiya Hachisuka, Henrik Wann Jensen*. Parallel progressive photon mapping on GPUs / In ACM SIGGRAPH ASIA 2010 Sketches (SA '10). New York, NY, USA: Association for Computing Machinery, Article 54, 1. https://doi.org/10.1145/1899950.1900004
14. *Hunt W., Mark W.R., Stoll G.* Fast kd-tree construction with an adaptive error-bounded heuristic // 2006 IEEE Symposium on Interactive Ray Tracing. IEEE, 2006. P. 81–88.
15. *Knoll Aaron*. A survey of octree volume rendering methods.
16. *Fabianowski Bartosz, Dingliana J*. Compact BVH Storage for Ray Tracing and Photon Mapping.
17. *Bradshaw Gareth, O'Sullivan Carol*. Sphere-tree construction using dynamic medial axis approximation / In Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '02). New York, NY, USA, Association for Computing Machinery. 2002. P. 33–40. https://doi.org/10.1145/545261.545267
18. *Gino van den Bergen*. Efficient collision detection of complex deformable models using AABB trees // J. Graph. Tools. 1997. V. 2. № 4. P. 1–13. https://doi.org/10.1080/10867651.1997.10487480
19. *Gottschalk S., Lin M.C., Manocha D.* OBBTree: a hierarchical structure for rapid interference detection / In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96). New York, NY, USA: Association for Computing Machinery, 1996. P. 171–180. https://doi.org/10.1145/237170.237244
20. *Martin Stich, Heiko Friedrich, Andreas Dietrich*. Spatial splits in bounding volume hierarchies / In Proceedings of the Conference on High Performance Graphics 2009 (HPG '09). New York, NY, USA, Association for Computing Machinery. P. 7–13. https://doi.org/10.1145/1572769.1572771
21. *Wald Ingo, Günther Johannes, Slusallek Philipp, Cani Marie-Paule, Slater Mel*. Balancing Considered Harmful - Faster Photon Mapping using the Voxel Volume Heuristic / The European Association for Computer Graphics 25th Annual Conference EUROGRAPHICS 2004. Blackwell, 2004. V. 23. P. 595–603.
22. *Халимов Р.Р., Жданов Д.Д., Жданов А.Д.* Формирование эффективной пространственной структуры фотонных карт для ускорения процесса рендеринга // Труды Международной конференции по компьютерной графике и зрению "Графикон". 2022. Т. 32. С. 110–123.
23. *Havran Vlastimil*. Heuristic ray shooting algorithms. 2000.
24. *Hapala M., Havran V.* Review: Kd-tree Traversal Algorithms for Ray Tracing // Computer Graphics Forum. V. 30. P. 199–213. https://doi.org/10.1111/j.1467-8659.2010.01844.x
25. *Foley T., Sugerman J.* KD-tree acceleration structures for a GPU raytracer // In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware (HWWS '05). New York, NY, USA, Association for Computing Machinery. P. 15–22. https://doi.org/10.1145/1071866.1071869
26. Lumicept Integra [Электронный ресурс]. URL: https://integra.jp/en/products/lumicept.
27. *Zhdanov A.D., Zhdanov D.D.* The two-level semi-synchronous parallelization method for the caustic and indirect luminance calculation in realistic rendering // CEUR Workshop Proceedings. 2020. V. 2744. P. 1–12.