Применение когерентной трассировки лучей в задачах физически аккуратной визуализации

Б.Х. Барладян, А.Г. Волобой, К.А. Востряков, В.А. Галактионов, Л.З. Шапиро
Институт Прикладной Математики имени М.В. Келдыша РАН, Москва, Россия

Аннотация

С ростом вычислительной мощности современных микропроцессоров когерентная трассировка лучей становится все более популярной в компьютерной графике, поскольку применение ОКМД (SIMD) инструкций позволяет значительно ускорить этот процесс. Однако после ускорения трассировки лучей становится очевидным, что другие алгоритмы физически аккуратный визуализации, такие как расчет освещенности, нанесение текстуры и др., становятся узким местом производительности.
В настоящей статье, предлагается когерентный физически аккуратный алгоритм визуализации, который позволяет использовать преимущества SIMD инструкций современных процессоров на каждом этапе создания изображения. Представлены когерентные алгоритмы для расчета освещения и материалов, а также для удаления лестничного эффекта и оператора сжатия динамического диапазона. На тестовых сценах было проведено сравнение времени работы алгоритмов, которое показывает значительное ускорение расчетов по сравнению с обычным не когерентным подходом.

Ключевые слова: SSE, SIMD, интерактивная (когерентная) трассировка лучей, оператор сжатия динамического диапазона, удаление лестничного эффекта, текстура, двунаправленная функция отражения света, фотореалистичная визуализация.

1. Введение

Когерентная трассировка лучей (то есть трассировка нескольких когерентных лучей, выполняемая одновременно одним процессором) стала предметом научных исследований в последние годы. Современные микропроцессоры стали поддерживать ОКМД (одна команда - много данных) или SIMD (Single Instruction Multiple Data) расширения, которые позволяют выполнять арифметические и логические действия над несколькими числами с плавающей запятой одновременно. Различные процессоры имеют разные SIMD расширения: Intel SSE (Streaming SIMD Extension) [1], AMD 3DNow! [2] и Motorola AltiVec [3]. Из всех этих расширений именно Intel SSE стало широко применяться для реализации когерентной трассировки лучей. Также с недавнего времени процессоры AMD стали поддерживать SSE, а компьютеры Apple стали использовать процессоры Intel. Таким образом, SSE получило наибольшее распространение и стало фактическим стандартом SIMD инструкций для персональных компьютеров. Поэтому далее термины когерентная, SIMD и SSE трассировка лучей будут использоваться как синонимы.

Сегодня существует несколько проектов, использующих SSE трассировку лучей. Наиболее известными являются проект института Макса Планка [4], который использует SSE для интерактивной трассировки лучей. Среди других проектов можно назвать проект Manta – интерактивный трассировщик лучей с открытым исходным кодом, созданный для визуализации огромных моделей на суперкомпьютерах с разделяемой памятью или многоядерных рабочих станциях [5].

Коллектив отдела компьютерной графики ИПМ им. М.В. Келдыша РАН имеет большой опыт в создании систем физически аккуратной визуализации [6-8]. Им был разработан целый ряд приложений, которые поддерживают различные аспекты фотореалистичной визуализации. Научные результаты этих разработок были доведены до уровня коммерческого продукта [9]. Использование SSE, которое стало широко распространено, существенно ускоряет процесс генерации изображений и приводит для некоторых сцен к интерактивной визуализации. Вычислительные затраты на трассировку лучей являются значительной, но не единственной частью при создании фотореалистичных изображений. Визуализация также включает моделирование рассеивающих свойств поверхностей и физически аккуратного освещения от сложных источников света. Для получения качественного изображения также необходимо использовать алгоритмы устранения лестничного эффекта и сжатия динамического диапазона яркостей, чтобы преобразовать радиометрические величины, в которых происходит моделирование, в цветовые значения, изображаемые графическим монитором.

В настоящей статье представлены подходы, которые позволяют использовать SSE инструкции и получать ускорения расчетов до 3-4-х раз на всех стадиях визуализации: расчет освещенности, обработка сложных материалов и источников света, преобразование физических величин в цвета монитора и др.

Статья построена в следующем порядке. Раздел 2 кратко описывает архитектуру и дизайн системы визуализации Inspirer2, на базе которой была построена когерентная визуализация, и обрисовывает главные контуры системы когерентной трассировки лучей. Раздел 3 посвящен собственно когерентной трассировке лучей. В разделах 4 и 5 рассказывается о когерентной обработке сложных материалов поверхностей, заданных двунаправленной функцией отражения (ДФО) в общем виде, а также описывается множество типов источников света для когерентной обработки. Раздел 6 посвящен оператору сжатия динамического диапазона. В разделе 7 описан адаптивный алгоритм устранения лестничного эффекта для 4-лучевой SSE трассировки. В разделах 8 и

9 приведены оценки производительности работы системы на некоторых конкретных сценах, даны выводы и указаны направления будущих разработок.

2. Архитектура базовой системы визуализации

Физически аккуратная визуализация с помощью когерентной трассировки лучей была реализована на основе системы визуализации Inspirer2 (ранее Fly) [10].

Эта система поддерживает как интерактивную визуализацию, так и построение высококачественных изображений в фоновом режиме. Интерактивный режим был реализован на основе OpenGL. В этом режиме система обеспечивает визуализацию сцен со скоростью близкой к реальному времени (до 20-60 кадров в секунду). При этом главной целью разработки системы было обеспечение наибольшего уровня физической корректности, который возможен при такой частоте обновления кадра. В интерактивном режиме система способна отобразить физически аккуратные тени от точечных источников света, материалы поверхностей, заданных двунаправленной функцией отражения (ДФО), а также аппроксимировать отражения с помощью карт окружения.

В режиме построения высококачественного изображения система обеспечивает моделирование освещенности и физически аккуратную визуализацию, используя двунаправленную трассировку лучей. В этом режиме корректно моделируются точечные, линейные и поверхностные источники света с гониодиаграммами, позволяющие задать практически любое реалистичное освещение. Была разработана возможность задания естественного освещения как непосредственно через географические параметры места и времени, так и через вычисление освещения, задаваемого изображениями с большим динамическим диапазоном (HDR – High Dynamic Range – панорамой)[11]. Пример изображения сцены, освещенной HDR панорамой показан на рис. 1. Также возможна спецификация материалов со сложными свойствами, заданными ДФО в наиболее общей, табличной форме. Табличное задание ДФО позволяет использовать данные, непосредственно измеренные на спектрофотометре [12]. Для учета зеркальных отражений и преломлений используется обратная трассировка лучей. Для расчета глобального освещения используется алгоритм прямой трассировки лучей методом Монте-Карло. Результаты расчета глобального освещения сохраняются в картах освещенности [13], и используются в обоих режимах визуализации.

Рис 1. Пример изображения полученного в Inspirer2 с HDR панорамой.

В действительности, и для интерактивного и для фонового режима визуализации использование SSE трассировки лучей дает свои преимущества. Фоновая, высококачественная визуализация может быть ускорена в 2 – 3 раза, поскольку с помощью SSE команд можно трассировать 4 когерентных луча одновременно. В отношении интерактивной визуализации SSE трассировка луча может быть использована в гибридном подходе, обеспечивая физически аккуратные отражения и преломления поверх OpenGL изображения.

Общая архитектура системы Inspirer2 представлена на рис. 2. С одной стороны она удовлетворяет требованию единого задания сцены в физических величинах, а с другой обеспечивает разное внутреннее представление для различных режимов визуализации. Такое архитектурное решение облегчает процесс интеграции нового режима когерентной трассировки, поскольку позволяет добавить его, разработав свое внутреннее представление.

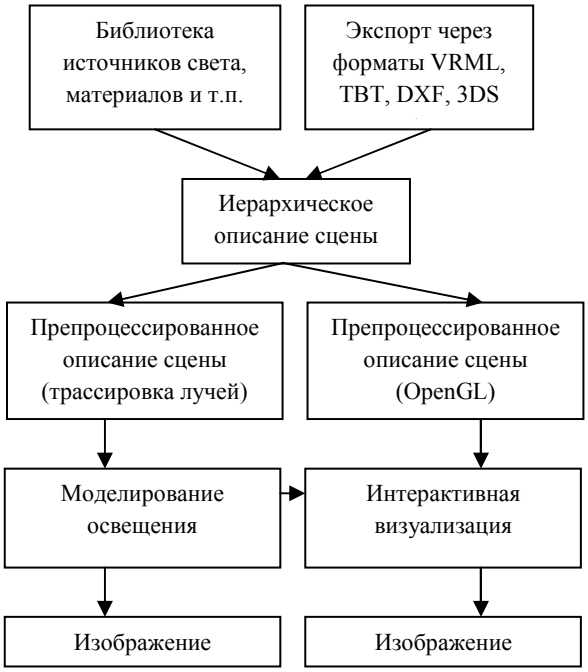


Рис 2. Архитектура системы Inspirer2. Иерархическое описание физических атрибутов сцены преобразуется в два препроцессированных описания сцены: одно для OpenGL визуализации, а другое для трассировки лучей (включая трассировку методом Монте-Карло).

Когерентная трассировка лучей, как и другие компоненты системы, были реализованы на языке C++. Ассемблер не использовался. Для того чтобы использовать SSE функциональность, были разработаны классы, которые представляют высокоуровневую оболочку над SSE интринсиками (специальными функциями, которые транслируются компилятором в SSE код), поддерживаемыми Microsoft/Intel C++ компиляторами. Поскольку современные компиляторы способны оптимизировать высокоуровневый код достаточно эффективно, то это не должно являться причиной большой потери производительности, в то время как значительно облегчается будущая поддержка написанного кода.

3. Когерентная трассировка лучей

Трассировка лучей обычно рассматривается как наиболее затратная по времени часть любого физически аккуратного алгоритма визуализации. По оценке Уиттеда время, затрачеваемое на трассировку лучей, составляет 95% общего времени визуализации [14]. Для физически аккуратного моделирования относительное время трассировки, однако, меньше, и составляет около 65 – 75%, согласно нашим оценкам [15]. Это ставит важность SSE оптимизации трассировки лучей на первое место.

SSE операции выполняются над четырьмя 32-битными числами с плавающей запятой одновременно. Таким образом SSE трассировка лучей позволяет трассировать четыре луча параллельно. Относительно однолучевой трассировки алгоритм принципиально не изменяется. Для ускорения трассировки использовался метод представления пространства в виде бинарного BSP дерева (binary space partition). Таким образом, алгоритм трассировки луча состоит из фазы прохождения по дереву и фазы нахождения пересечений луча с объектами сцены, находящимися в выбранном подпространстве. Для поддержки интерактивно движущихся объектов сцены используется двухуровневая трассировка [16]. При двухуровневой трассировке каждый объект (представленный треугольной сеткой) имеет свое собственное BSP дерево (далее называется BSP деревом второго уровня или BSP деревом объекта) и ограничивающий бокс (прямоугольный параллелепипед, выравненный по осям координат). Объект помещается в сцену вместе с матрицей преобразования. Вся сцена состоит из множества объектов ограниченных боксами. BSP дерево сцены (далее называется BSP деревом первого уровня) упорядочивает в пространстве эти сложные объекты сцены. Поскольку только ограничивающие боксы объектов включены в сцену, то деление может быть не таким эффективным в сравнении с тем, когда BSP дерево строится по треугольной сетке всей сцены, но это плата за возможность иметь в сцене движущиеся объекты. Хотя в очень разреженных сценах такое двухуровневое дерево может немного увеличить производительность вследствие лучшей утилизации пустого пространства между объектами.

Благодаря SSE одновременно может трассироваться до четырех лучей, однако они могут проходить различные ветви дерева в алгоритме трассировки лучей. По существу, это означает, что необходима временная блокировка нескольких лучей из четверки. Для этого используется маска активных лучей. Обычно, это SSE переменная (четверка 32-битных чисел с плавающей запятой), которая содержит либо 0x00000000, либо 0xffffffff (в двоичном представлении) в каждой из четырех позиций. Маска блокирует лучи, которые не проходят через текущую ветвь BSP дерева и те, для которых первое пересечение уже найдено. Маскирование –

это широко используемый прием в SSE программировании, который уменьшает ветвление и позволяет сделать алгоритм более потоковым. В нашей разработке он используется во всех компонентах алгоритма визуализации.

Когерентный алгоритм трассировки лучей выполняется следующим образом. Во-первых, все лучи проверяются на пересечение с боксом сцены. Если все лучи не пересекают его, то алгоритм немедленно сообщает, что пересечений нет. Если некоторые лучи пересекают сцену, то маска лучей обновляется, а лучи, не пересекающие бокс сцены, исключаются ею. Затем алгоритм начинает прохождение по BSP дереву. Если лучи имеют различные знаки в векторе направления, то они могут проходит узлы дерева в разном порядке. Тогда группа лучей делится на подгруппы с одинаковыми знаками в векторе направления (когерентные подгруппы). Это несколько снижает эффективность от использования SSE, но такие случаи бывают достаточно редко. Более того, можно показать, что лучи, которые имеют одно начало, например, лучи из камеры или теневые лучи от точечного источника света, всегда имеют одинаковый порядок прохождения узлов дерева.

После разделения лучей на когерентные группы алгоритм устанавливает маску для текущих лучей и начинает трассирование BSP дерева. Поскольку иерархия двухуровневая, то процедура, применяемая для всей сцены, повторяется для каждого объекта, который тестируется на пересечение. Для каждого не листового узла алгоритм прохождения BSP выполняется следующим образом. Если все лучи идут в один подузел (только в правый или только в левый), то алгоритм лишь обновляет адрес текущего узла и фактически переходит в этот подузел. Если некоторые из лучей проходят через оба подузла, то дальний узел записывается в стек, маска активных лучей обновляется, и алгоритм идет в ближайший подузел. Заметим, что благодаря тому, что лучи разбиты на группы, ситуации, когда лучи проходят узлы в разном порядке, не воспроизводятся. Возможно, однако, что некоторые лучи будут вынуждены пройти оба подузла, хотя реально они проходят только один из них. В этом случае такие лучи блокируют маской. Они будут активированы снова, когда узел будет пройден.

При достижении листового узла дерева выполняется пересечение луча с объектом. Для дерева первого уровня объектами являются объекты сцены, луч преобразуется в систему координат объекта, и алгоритм продолжается таким же образом, как для всей сцены в целом. Для дерева второго уровня объектами являются треугольники.

Для определения пересечения луча с треугольником используется модифицированный барицентрический проекционный алгоритм (тест), реализованный на SSE, как описано в [4]. После тестирования всех объектов в узле, те лучи, для которых пересечение найдено, деактивируются, поскольку они не нуждаются в дальнейшем прохождении по дереву. Если все объекты в листовом узле первого уровня были протестированы, то эти пересечения являются *первыми пересечениями* для этих лучей.

Наша схема, фактически, имеет два метода пересечения. Первый метод, который находит только первое пересечение лучей, описан выше. Другой находит *все пересечения* до первого непрозрачного объекта. Он работает также как и первый за исключением того, что лучи, которые столкнулись с прозрачным объектом, не маскируются немедленно, а продолжают трассироваться до столкновения с первым непрозрачным объектом.

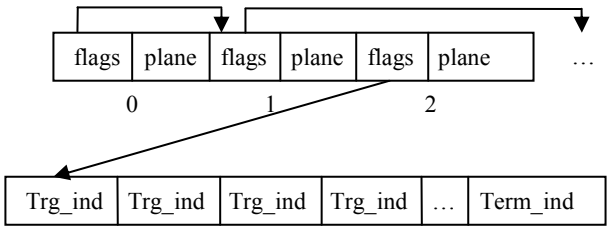


Рис 3. Размещение дерева сцены и индексов треугольников в памяти.

Размещение дерева сцены и данных треугольников в памяти (рис. 3) оптимизироны с учетом кэша процессора. Оба дочерних узла дерева сохраняются друг за другом в одной линии кэша, что сокращает задержки загрузки данных из основной памяти.

Для построения BSP дерева использовался алгоритм, описанный в работе [17]. Поскольку построение BSP дерева требует большого количества времени, использовался двухуровневый подход, который позволил не перестраивать все дерево при изменении положения объекта, а только модифицировать его. Это дало возможность интерактивно работать с динамическими сценами. Однако, как уже говорилось выше, построенные таким образом BSP деревья немного менее эффективны, чем простое одноуровневое BSP дерево.

На рис. 4 показана сцена, визуализированная с помощью SSE трассировки лучей.

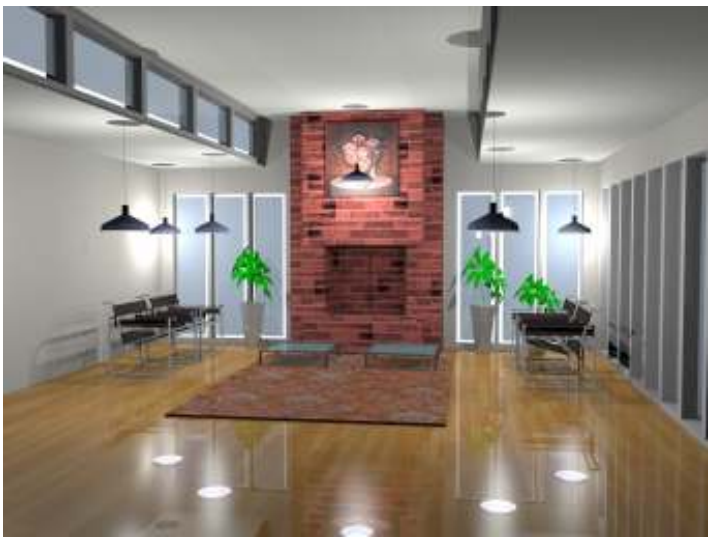


Рис 4. Изображение сцены, полученное с помощью SSE трассировки лучей.

4. Оптические свойства материалов и ДФО (BRDF)

Поскольку когерентная трассировка лучей дает ускорение в 2-3 раза в сравнении с обычными алгоритмами трассировки, другие части алгоритма физически аккуратной визуализации становятся узким местом производительности. Как отмечалось ранее, время, затрачиваемое на трассировку лучей, составляет порядка 70% от общего времени визуализации. Использование SSE ускоряет трассировку лучей в среднем примерно в 3.5 раза, следовательно, общее время генерации изображения сокращается меньше чем наполовину. Поэтому необходимо также ускорить остальные компоненты алгоритма визуализации.

С такой же проблемой встречались и другие разработчики когерентной трассировки. Так в [18] после реализации SSE трассировки авторы отмечают, что вычисление затенения фактически стало узким местом. По их мнению, даже простая модель Фонга может заметно замедлить визуализацию, не говоря уже о включении более сложных двунаправленных функций отражения (ДФО).

Наши разработки были в большей степени ориентированы на моделирование освещенности и генерацию физически аккуратных изображений, нежели на создание визуально правдоподобных эффектов. Поэтому нам необходимо было реализовать когерентную обработку материалов и ДФО.

Физически аккуратные материалы в нашей реализации обычно состоят из следующих компонент:

1. Простого набора атрибутов материала, представимого моделью Фонга.
2. Атрибутов зеркального отражения и преломления.
3. Текстур.
4. ДФО материала.

Первые две позиции могут быть явно реализованы на SSE, фактически включая только простые векторные и цветовые операции. Расчет освещения требует трассировки теневых, отраженных и преломленных лучей, что легко осуществить с помощью эффективной SSE трассировки описанной выше.

Однако текстурирование и поддержка ДФО не так легко реализуется на SSE. Аспекты их реализации описаны в следующих двух разделах.

4.1. Когерентное текстурирование.

Текстура (двумерная матрица цвета) накладывается на поверхность объекта сцены для изменения его внешнего облика. Пример использования текстуры показан на рис. 5.

В нашей системе текстуры принадлежат материалам, а материалы приписываются треугольникам, из которых состоят объекты. Текстурное отображение осуществляется с помощью текстурных координат, заданных для каждой вершины треугольника, содержащего текстурированый материал. Для вычисления текстурных координат в точках столкновения четверки лучей с треугольником используются барицентрические координаты, полученные в процедуре пересечения лучей с треугольником. С их помощью текстурные координаты могут быть проинтерполированы. Это легко сделать с помощью SSE для четырех точек одновременно, если они принадлежат одному треугольнику. Если точки принадлежат разным треугольникам, то интерполяция выполняется за несколько проходов с маскированием неактивных точек.

Рис 5. Изображение сцены с текстурами, полученное с помощью предлагаемого когерентного алгоритма.

Далее интерполируются текстурные значения при помощи трилинейной фильтрации. Вычисление mipmap уровня (предварительно фильтрованного изображения с разным размером фильтра кратным 2) более сложно, потому что включает логарифм по основанию 2 расстояния от точки до камеры. Поскольку требуется только ближайшее целое значение логарифма, то можно воспользоваться тем, как представлено число с плавающей запятой в двоичном виде (мантисса и порядок). Логарифм от числа с плавающей запятой можно представить как сумму: логарифм от мантиссы и порядок, которые легко получить с помощью битовых операций. Мантисса расположена в пределах от 0.5 до 1, и ее логарифм можно легко аппроксимировать полиномом. Точность такой аппроксимации логарифма зависит только от точности выбранного полинома на этом интервале, где логарифм не имеет особенностей.

Mipmap уровень выбирается так, что соседние лучи в пучке почти наверняка попадают в разные пиксели, и поэтому теряется когерентность по данным, необходимая для эффективного использования SSE. Но остается возможность применять SIMD инструкции для билинейной интерполяции RGBA компонент, где когерентность по данным присутствует всегда. Поэтому выполняется билинейная интерполяция в двух ближайших mipmap уровнях, и затем линейная интерполяция между ними.

4.2. Когерентная реализация ДФО

Поддержка сложных свойств материалов критична для физически аккуратной визуализации. Большинство объектов, встречающихся в повседневной жизни, таких как автомобильные краски, дерево, пластик и ткани, обладают сложными оптическими свойствами, которые не могут быть описаны простыми эвристическими моделями, такими как модель Фонга [19] и аналогичными. В таких случаях необходимо использовать более общую модель поверхностного рассеивания.

В нашей системе используется ДФО, основанные на различных физических данных. ДФО могут быть либо измерены на специальной установке, либо смоделированы. Табличное представление является, вероятно, единственным по настоящему универсальным практическим приемом представления таких ДФО.

ДФО параметризуется на основе углового описания направлений освещения и наблюдения. В зависимости от числа углов, использованных в параметризации, ДФО бывает 3-х и 4-х мерными. Трехмерные часто называют изотропными, а четырехмерные – анизотропными.

Поскольку ДФО могут иметь особенности, а также быть многомерными функциями, они не могут быть табулированы регулярно из-за больших требований по памяти. Поэтому для нахождения ячейки, в которой будет осуществляться интерполяция, применяется бинарный поиск.

Опишем алгоритм вычисления ДФО. Прежде всего вычисляются углы лучей падения и наблюдения с помощью обратных тригонометрических функций. Затем выполняется бинарный поиск, для того чтобы определить интерполяционную ячейку. Внутри ячейки значения ДФО интерполируются для данных направлений лучей.

Для того чтобы эффективно реализовать все выше описанное с использованием SSE, были разработаны алгоритмы когерентного бинарного поиска для четырех значений одновременно и аппроксимации вычисления обратных тригонометрических функций. А интерполяция внутри ячейки явно реализуется на SSE. Подробное описание нашего подхода приведено в [20]. На рис. 6 показан пример визуализации сцены, содержащей материал, описываемый с помощью ДФО, с использованием SSE.

Рис 6. Визуализация оптически сложного материала, описываемого ДФО, с помощью разработанного подхода.

В табл.1 приведены результаты сравнения производительности SSE реализации ДФО на примере анизотропной ДФО размером 17 x 7 x 17 x 13. Сравнение производилось на компьютере Intel Centrino 1800 MHz Mobile Pentium-IV CPU с 512 MB 433 MHz RAM.

Количество вызовов	100000	200000	400000
Без SSE (сек.)	0.137	0.248	0.495
SSE (сек.)	0.040	0.078	0.156
Ускорение	3.43	3.17	3.17

Таблица 1. Сравнение производительности реализации анизотропной ДФО с использованием SSE и без него.

Таким образом, было получено ускорение около 3.2 в среднем, что меньше 4, поскольку вычисление ДФО содержит множество ветвлений, которые снижают эффективность использования SSE. Важно отметить, что за раз обрабатывается только один материал. Если четверка лучей попадает не в один и тот же материал, а в разные, то различные материалы обрабатываются по очереди. При этом, ненужные в данный момент лучи маскируются.

5. Источники света

Для того чтобы когерентная реализация физически аккуратной визуализации была достаточно эффективной, освещение должно быть реализовано с помощью SSE инструкций. Термин “освещение” здесь обозначает процесс вычисления падающей интенсивности света в данной точке без учета видимости от источника. Видимость может быть реализована с помощью описаной выше процедуры SSE трассировки для теневых лучей.

Нами рассматривались несколько типов источников света, которые можно разделить на точечные и поверхностные. Для того чтобы рассчитать освещение от поверхностных источников света, необходимо сгенерировать точки на их поверхности и определить интенсивность света, идущую от каждой из точек, т.е. проинтегрировать освещенность по поверхности источника. От таких источников получаются естественные тени с мягкими границами – полутенями.

Другая группа источников света включает различные точечные источники: от простых, таких как равномерно направленный или конический, до сложных, описываемых гониодиаграммами. Для простых источников реализация когерентного освещения осуществляется непосредственно. Как и для материалов, алгоритм работает за раз только с одним источником света. Если по какой-то причине (например, источник находится с другой стороны треугольника), освещение не нужно вычислять для некоторых лучей, то они маскируются.

Поскольку для большинства источников света выполняются достаточно простые вычисления, то те же вычисления теперь выполняются над четверками лучей. Ситуация усложняется с точечными источниками света, которые имеют гониодиаграммы. Гониодиаграммы – это общий индустриальный формат представления исходящей интенсивности источников света. Их поддержка критична для нашей системы, которая нацелена на физически аккуратную визуализацию. Интенсивность источника света с гониодиаграммой табулирована двумерной нерегулярной таблицей, чем очень похожа на ДФО. Для того чтобы определить интенсивность для определенного направления, во-первых, необходимо вычислить сферические координаты, во-вторых, определить ячейку, которой принадлежит текущее направление, и проинтерполировать интенсивность внутри найденной ячейки. Эти шаги в точности повторяют то, что необходимо сделать для вычисления ДФО. Фактически, оба алгоритма используют несколько общих функций.

Производительность когерентной реализации была измерена для различных видов источников света. Обе реализации (SSE и без SSE) имеют достаточно высокую точность, поэтому изображения, полученные обоими подходами, практически неотличимы. SSE подход, однако, работает более чем в 3.5 раза быстрее, чем реализация без SSE.

В табл. 2 приведены оценки эффективности SSE реализации некоторых источников света. Сравнение проводилось на компьютере Pentium 4, 2.8 GHz, 1 GB 433MHz RAM. Результаты приведены в секундах.

Тип источника света	Без SSE	SSE	Ускорение
Равномерно направленный	1.137	0.157	7.24
Конический	0.816	0.211	3.87
Параллельный	0.444	0.103	4.31
Направленный	0.936	0.150	6.24
Линейный	6.696	1.149	5.83
Круглый	27.936	4.828	5.79
Прямоугольный	145.25	24.375	5.96
С гониодиаграммой	2.573	0.588	4.38

Таблица 2. Сравнение производительности реализаций источников света без SSE и SSE.

Как можно заметить, для большинства источников света ускорение превысило 4. Только конический источник получил чуть меньшее ускорение. Также источник света с гониодиаграммой получил меньшее ускорение, чем, например, равномерно направленный, потому что гониодиаграмма требует бинарного поиска, в котором возникает некогерентность данных.

6. Оператор сжатия динамического диапазона яркости

Физически аккуратная визуализация невозможна без оператора сжатия динамического диапазона. Эта необходимость возникает из-за того, что изображение, полученное в результате моделирования, представлено в радиометрических единицах в диапазоне [0, ∞), а монитор обладает ограниченным динамическим диапазоном. Возникает задача сжимающего отображения с сохранением большинства важных деталей.

Алгоритм сжатия диапазона яркости выполняется следующим образом. Во-первых, вычисляется малоразмерная копия изображения с большим динамическим диапазоном. Эта копия используется для вычисления логарифмически средней интенсивности. Это дает общее представление о распределении интенсивности в конечном изображении.

Использовался метод, который описан в работе [21]. Здесь приведен только поход к SSE реализации, за подробным описанием самого алгоритма отсылаем к указанному источнику.

Главной сложностью с реализацией этого алгоритма является вычисление степенной функции x^y, которая может быть представлена с помощью функций логарифма и экспоненты.

$$x^y = 2^{y \log_2 x}$$

Как для эффективной SSE реализации аппроксимировать логарифм описано выше (разд. 4.1). Подобная идея используется и для вычисления экспоненты. Сначала экспонента приводится к основанию 2, а затем представляется в виде $2^x = 2^{\lfloor x \rfloor} 2^{x - \lfloor x \rfloor}$, где $\lfloor x \rfloor$ – это целая часть числа. Первый множитель вычисляется двоичным сдвигом, а второй аппроксимируется полиномом. При этом заметного ускорения в вычислении степенной функции, логарифма и экспоненты можно достичь только с использованием инструкций SSE2, которые позволяют выполнять целочисленные команды одновременно над четверками чисел, при этом данные все время остаются на SSE регистрах.

7. Алгоритм устранения лестничного эффекта для 4-лучевой трассировки лучей

В настоящей статье приводятся только основные положения нового адаптивного алгоритма устранения ступенчатости изображений с использованием 4-лучевой SSE трассировки лучей. Его более подробное описание можно найти в [22].

Каждый раз, когда обнаруживается область с быстрым изменением функции изображения, необходимо наиболее эффективно трассировать 4 новых когерентных луча, а не пытаться трассировать индивидуальные лучи по одному. Предложенный алгоритм использует SSE маску разности цветов ближайших лучей как индекс в таблицы разрывности. Таким образом, быстро определяются зоны с быстрыми изменениями функции изображения. Благодаря нашему алгоритму оказалось возможным трассировать в среднем 1.5–2 лучей на пиксел при качестве аналогичном 25 лучам на пиксел.

Плоскость экрана разбивается на квадратные области («плитки») размером 64x64 пикселя. Вначале для текущей плитки выполняется растеризация в графическом процессоре. Это может нас избавить от трассировки множества дополнительных лучей, которые понадобились бы для определения геометрических разрывов. Создается карта видимости, представляющая собой матрицу индексов треугольников видимых из камеры через пиксели. Эта матрица имеет разрешение в несколько раз превышающее размер плитки (в 4 раза для обычного качества и в 6 раз для высокого качества). После этого для каждого пикселя алгоритм проверяет есть ли внутри него разрыв, т.е. есть ли внутри него значения карты видимости, соответствующие объектам с сильно различающимися нормалями или расстоянием от камеры до них. Так по матрице видимости создается матрица разрывности с размером равным размеру плитки. На этом этапе мы не определяем место разрыва внутри пикселя, а только то, в каких пикселях существуют геометрические разрывы.

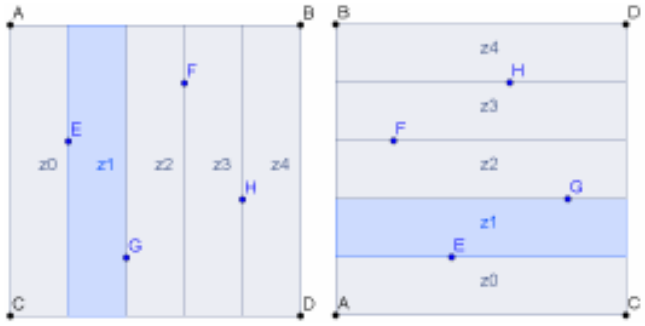
Предлагаемый адаптивный алгоритм имеет три уровня. Следующий уровень последовательно применяется в случаи обнаружения разрыва на предыдущем уровне.

На первом уровне используется регулярная выборка. Лучи трассируются четверками через углы пикселей. Значения цвета для каждого луча записываются в матрицу. После этого выбирается очередной пиксел из плитки, и находятся значения цветов его угловых лучей из матрицы (точки ABCD на рис. 7-а). Алгоритм определяет вид разрыва: горизонтальный или вертикальный. Горизонтальный означает, что существует высокий градиент между точками A и B или точками C и D. Если разрыв определяется неоднозначно, например, существует большая вариация между точками A и B и точками B и D, то алгоритм считает, что существует только горизонтальный разрыв. Если карта видимости обнаружила разрыв, а разности между угловыми значениями малы, то также будем предполагать, что существует горизонтальный разрыв.

Если алгоритм обнаружил вертикальный разрыв, то точки ABCD поворачиваются на 90º против часовой стрелки. Этот поворот обеспечивает дальнейшую линейность алгоритма независимо от вида разрыва (рис. 7-б).

Если и карта видимости, и сравнения угловых значений не обнаружили разрыв, то тогда и только тогда алгоритм определяет цвет данного пикселя как среднее его угловых значений, и переходит к рассмотрению следующего пикселя. Иначе переходим ко второму уровню для данного пикселя.

Внутри выстреливаем четверку лучей через точки E, F, G, H, как показано на рис. 7.



а) б)

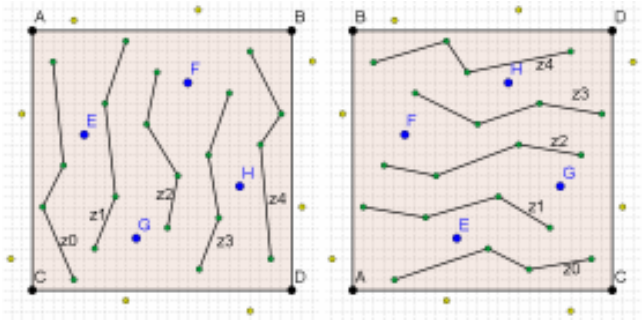
Рис. 7. Геометрические разрывы на пикселах изображения: а) горизонтальный разрыв в зоне z1 между точками E и G; б) повернутая против часовой стрелки на 90º вертикальная версия.

Для определения разрывов между точками первого и второго уровней используем три четверки сравнений по модулю. Каждая четверка сравнений выполняется с помощью SSE команд над R, G, B компонентами цвета. Каждое SSE сравнение дает четырехбитную маску, которая используется как индекс в таблице определения разрывов, которая заполняется заранее. Значениями этой таблицы являются флаги, которые определяют зоны разрывов: z0, z1, z2, z3, z4 (рис. 7). Значения, полученные из трех таблиц (три четверки сравнений), логически складываются (дизъюнкция). Полученные флаги показывают зоны разрывов, где необходимо дополнительно выстреливать лучи.

На третьем уровне в каждую разрывную зону выстреливается четверка лучей зигзагом (рис. 8). Каждый зигзаг покрывает одну из разрывных зон (горизонтальную или вертикальную). Максимально выстреливается пять зигзагов на пиксель. Заметим, что точки каждого зигзага расположены с шагом 1/20 стороны пикселя вдоль высокого градиента разрыва (рис. 8). Другими словами, например, в случае большого горизонтального градиента на каждой вертикальной линии с шагом 1/20 по горизонтали расположено по одной точке зигзага. Кроме того, независимо от того горизонтальный или вертикальный случай рассматривается, всегда ближайшие точки соседних пикселей расположены в одних и тех же местах. Это свойство обеспечивает относительную равномерность расположения точек даже между пикселями с разными вариантами поворота.

Зигзаг может быть протрассирован или проинтерполирован из значений выборки на предыдущих уровнях. Решение о том, что делать: трассировать или интерполировать, принимается на основе флагов, полученных из таблиц определения разрывов. Интерполяция осуществляется по точкам, принадлежащим текущей зоне разрыва, потому что каждый зигзаг обрабатывается независимо. Например, точки зигзага z2 интерполируются только по точкам F и G, а точки E и H в интерполяции не участвуют, поскольку в зонах z1 и z3 могут быть разрывы. Каждая точка зигзага z0 интерполируется по двум точкам из трех: A, C, E. Верхняя точка (рис. 8-а) по

точкам A и E, а остальные три по точкам E и C. Зигзаг z1 интерполируется по точкам E, G. Зигзаг z2 интерполируется по точкам G, F. Зигзаг z3 интерполируется по точкам H, F. Точки зигзага z4 интерполируется по двум точкам из H, B, D.



а) б)

Рис. 8. Точки третьего уровня алгоритма определения разрыва: а) горизонтальный вариант; б) вертикальный.

Расположение всех точек детерминировано, поэтому для того, чтобы получить значение цвета конкретного пикселя, можно предварительно вычислить для каждой из точек вес, с которым ее цвет будет просуммирован,. В результате можно существенно ускорить скорость работы алгоритма. Если используется прямоугольный фильтр (что является достаточно частым), то после обработки очередного пикселя в буфер кадра сразу записывается его цвет, и никакой дополнительной информации о субпиксельных цветах хранить больше не нужно.

Сравнения показали, что в типичных сценах при использовании предложенного адаптивного алгоритма и при среднем количестве лучей на пиксель 1.5 - 2 может быть достигнуто качество сравнимое с 25 лучами на пиксель регулярной выборки.

8. Результаты

Алгоритмы, описанные выше, были реализованы на языке C++ в среде разработки Visual Studio 2003. Для SSE команд ассемблер не использовался. Доступ к SSE инструкциям осуществлялся через интринсики (intrinsics), которые, в свою очередь, были обернуты в классы-оболочки. Такой подход позволил обеспечить хорошую сопровождаемость кода ценой незначительной потери производительности.

В табл. 3 приведены результаты сравнения скорости визуализации с включенным и выключенным SSE режимом для трех сцен (Car, Glass и Room, показанные на рис. 6, 5, и 4 соответственно). Сравнение производилось на компьютере dual 2.13 MHz Athlon MP с 2 GB памяти с одним включенным процессором (второй процессор был отключен). Изображения строились в разрешении 1024 x 768 пикселей. Характеристики сцен, участвующих в тестировании, приведены в табл. 4, где глубина трассировки обозначает максимальную высоту дерева лучей, т.е. глубина 0 соответствует только лучам из камеры, глубина 1 – одному отражению и т.д.

Сцена	Без SSE	SSE	Ускорение
Car	9.0	2.5	3.6
Glass	15.6	6.0	2.6
Room	21.6	6.1	3.5

Таблица 3. Сравнение визуализации с SSE и без SSE при разрешении 1024 x 768.

Сцена	количество треугольников	число источников света	Глубина трассировки
Car	141512	3	3
Glass	44928	2	2
Room	11788	9	2

Таблица 4. Характеристики сцен, участвующих в сравнении.

Сцена Car содержит измеренную табулированную ДФО, прозрачные и преломляющие объекты. Сцена Glass содержит много текстурированных поверхностей.

9. Заключение

В статье представлен алгоритмы физически аккуратной когерентной визуализации, которые ускоряют генерацию изображения более чем в 3 раза. Ускорение было достигнуто в основном за счет использования SSE

инструкций, которые могут дать ускорение до 4 раз в случае полной когерентности. Значительно повлияли на ускорение тщательный подбор алгоритмов и структур данных, а также большие усилия по оптимизации кода.

Достигнутая скорость визуализации не является интерактивной, но уже приближается к этому. Уменьшение разрешения изображения и сокращение количества источников света позволяет достичь и интерактивной скорости визуализации.

Другим направлением для применения SSE трассировки лучей является ускорение расчета глобального освещения. Для визуализации вторичного освещения нами используются карты освещения (illumination maps) [13]. Для их вычисления применяется трассировка лучей методом Монте-Карло. Однако лучи при использовании этого метода обладают значительно меньшей когерентностью и поэтому прямое использование SSE затруднительно.

Сейчас в SSE реализации используются только RGB модель цвета. Было бы интересно, однако, изучить возможность спектрального представления цвета. Обычно спектральное представление содержит 20-40 интенсивностей света, измеренных для разных длин волн, поэтому такой подход скорее всего будет далек от интерактивности. Однако, это может ускорить визуализацию различных спектральных эффектов, которые требуются в промышленной визуализации. Реализация нашей системы в режиме без SSE уже поддерживает спектральные ДФО и оптические свойства материалов. Расчеты для нескольких десятков спектральных значений может хорошо ложиться на SIMD схему, что позволит получить ускорение до 4-х раз.

Версия статьи с цветными иллюстрациями доступна на сайте:
http://www.keldysh.ru/pages/cgraph/publications/cgd_publ.htm

Работа была поддержана грантом РФФИ № 05-01-00345 и компанией Integra Inc. (Токио, Япония).

Литература

[1] IA-32 Intel Architecture Optimization Reference Manual, p. 440
http://www.intel.com/design/pentium4/manuals/24896612.pdf
[2] AMD 3DNow! extensions
http://www.amd.com/us-en/Processors/SellAMDProducts/0,,30_177_4458_4513^1413^2137,00.html
[3] PrPMC800: MPC7410 Processor PMC with AltiVec Technology
http://www.motorola.com/content/0,,5626,00.html
[4] Ingo Wald, Carsten Benthin, Markus Wagner, Philipp Slusallek Interactive Rendering with Coherent Ray Tracing. Proc. of *Eurographics* 2001, vol. 20, № 3, pp. 153 – 164.
[5] Abe Stephens, Solomon Boulos, James Bigler, Ingo Wald, Steven Parker An Application of Scalable Massive Model Interaction using Shared-Memory Systems. Eurographics Symposium on Parallel Graphics and *Visualization (2006)*.
[6] Andrei Khodulev, Edward Kopylov: Physically Accurate Lighting Simulation in Computer Graphics Software. *Proc. GraphiCon'96: The 6-th International conference on Computer Graphics and Visualization*, St. Petersburg, Russia, July 1-5, 1996. Vol.2, pp.111-119.
[7] Баяковский Ю.М., Галактионов В.А. О некоторых фундаментальных проблемах компьютерной (машинной) графики // *"Информационные технологии и вычислительные системы"*, № 4, 2004, с. 3-24.
[8] Волобой А.Г., Галактионов В.А., Дмитриев К.А., Копылов Э.А. Двунаправленная трассировка лучей для интегрирования освещенности методом квази- Монте Карло // *"Программирование"*, № 5, 2004, с. 25-34.
[9] Волобой А.Г., Галактионов В.А. Машинная графика в задачах автоматизированного проектирования // *"Информационные технологии в проектировании и производстве"*, № 1, 2006, с. 64-73.
[10] A. Ignatenko, B. Barladian, K. Dmitriev, S. Ershov, V. Galaktionov, I. Valiev, A. Voloboy: A Real-Time 3D Rendering System with BRDF Materials and Natural Lighting. *Proc. Graphicon'2004*: *The 14-th International Conference on Computer Graphics and its Applications*, Moscow, Russia, pp. 159-162.
[11] А.Г. Волобой, В.А. Галактионов, Э.А. Копылов, Л.З. Шапиро, Моделирование естественного дневного освещения, задаваемого изображением с большим динамическим диапазоном. *"Программирование"*, № 5, 2006, с. 62-80.
[12] А.Г. Волобой, В.А. Галактионов, С.В. Ершов, А.А. Летунов, И.С. Потемин, Аппаратно-программный комплекс для измерения светорассеивающих свойств поверхностей. «Информационные технологии в проектировании и производстве», № 4, 2006, с. 24-39.
[13] E.Kopylov, A.Khodulev, V.Volevich: The Comparison of Illumination Maps Technique in Computer Graphics Software. *Proc. GraphiCon'98: The 8-th International Conference on Computer Graphics and Visualization*, Moscow, Russia, September 7-11, 1998, pp.146-153.
[14] Turner Whitted: An Improved Illumination Model for Shaded Display. *Communications of ACM*, Vol. 23, № 6, June 1980, pp. 343-349.
[15] Волобой А.Г., Метод компактного хранения октарного дерева в задаче трассировки лучей. *«Программирование»*, № 1, 1992, с. 21-27.
[16] Ingo Wald, Carsten Benthin, Philipp Slusallek: OpenRT – A Scalable and Flexible Engine for Interactive 3D Graphics. *Technical Report TR-2002-01, Computer Graphics Group, Saarland University*
http://graphics.cs.uni-sb.de/%7Ewald/Publications/2002_OpenRT/2002_OpenRT.pdf

[17] V. Havran: Heuristic Ray Shooting Algorithms. Dissertation Thesis, Faculty of Electrical Engineering, Czech Technical University, Prague, 2000.
[18] Carsten Benthin, Ingo Wald, Philipp Slusallek: A Scalable Approach to Interactive Global Illumination. Proceedings of *Eurographics 2003, Computer Graphics Forum, v.22, №3,* pp. 621 – 630.
[19] B. Phong: “Illumination for Computer Generated Pictures”. Communications of *the ACM,* vol. 18, № 6, 1975, pp. 311 – 317.
[20] Адинец А.В., Барладян Б.Х., Волобой А.Г., Галактионов В.А., Копылов Э.А., Шапиро Л.З., Когерентная трассировка лучей для сцен, содержащих объекты со сложными светорассеивающими свойствами. Препринт ИПМ им. М.В. Келдыша РАН, № 107, 2005.
[21] Барладян Б.Х., Волобой А.Г., Галактионов В.А., Копылов Э.А. Эффективный оператор сжатия динамического диапазона яркостей // *"Программирование",* № 5, 2004, с. 35-42.
[22] К.А. Востряков, А.Г. Волобой. Алгоритм устранения лестничного эффекта для 4-лучевой SSE трассировки лучей. *Proc. GraphiCon'2007, The 17th International Conference on Computer Graphics and Computer Vision,* Moscow, Russia, June 23-27, 2007.