РОССИЙСКАЯ АКАДЕМИЯ НАУК
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ имени М.В. Келдыша

А.В. Адинец, Б.Х. Барладян, А.Г. Волобой, В.А. Галактионов,
Э.А. Копылов, Л.З. Шапиро

Когерентная трассировка лучей для сцен, содержащих объекты со сложными светорассеивающими свойствами

Москва
2005

А.В. Адинец, Б.Х. Барладян, А.Г. Волобой, В.А. Галактионов,

Э.А. Копылов, Л.З. Шапиро

Когерентная трассировка лучей для сцен, содержащих объекты со сложными светорассеивающими свойствами

Аннотация

В данной работе описывается подход к реализации поддержки двунаправленной функции отражения (ДФО) материалов в рамках когерентной трассировки лучей. В отличие от подходов, предложенных ранее, в настоящей работе используются реальные измеренные ДФО, представленные в виде трех- или четырехмерных сеток. В результате применения технологии SSE достигается 3-4-х кратное ускорение по сравнению с использованием ДФО в обычных программах синтеза реалистичных изображений методом трассировки лучей света.

A.V. Adinetz, B.H. Barladyan, A.G. Voloboy, V.A. Galaktionov,

E.A. Kopylov, L.Z. Shapiro

Coherent Ray Tracing of complex BRDF objects

Abstract

An approach for providing BRDF support for coherent ray tracing is described. Unlike earlier approaches, this approach is able to handle real-world measured BRDF represented as 3D or 4D grid of samples. The presented SSE implementation speedups BRDF support in 3-4 times comparing with traditional ray tracing BRDF computations.

Работа была поддержана грантом Президента РФ «Ведущие научные школы» № РИ-112/001/278, грантом РФФИ № 05-01-00345, а также компанией INTEGRA Inc. (Япония).

Версию статьи с цветными иллюстрациями можно найти по адресу http://www.keldysh.ru/pages/cgraph/publications/cgd_publ.htm

Содержание

1. Введение

Одной из главных проблем синтеза реалистичных изображений является значительный объем вычислений и, как результат, большое время работы алгоритма. При расчете интенсивности точки экрана обратной трассировкой лучей [1] необходимо протрассировать несколько лучей: луч от точки наблюдения к сцене, лучи от точки пересечения с объектом на источники света, отраженные и преломленные лучи, отраженные от отраженных лучей и так далее. При размере изображения 1280 на 1024 точек количество лучей может достигать 10-20 млн. При трассировке каждого луча рассматриваются пересечения со всеми объектами сцены, и выбирается ближайшая точка пересечения луча и объекта, так как именно она видима. По нашим оценкам процесс трассировки занимает 65% - 75% времени работы всей системы визуализации при построении изображения обратной трассировкой лучей без расчета глобальной освещенности. По данным Т. Уиттеда в данном случае трассировка может занимать до 95% [2] общего времени.

Практически все известные системы реалистичной визуализации, основанные на методе трассировки лучей, используют для ускорения вычислений пространственную иерархию объектов. Все пространство сцены разбивается на подобласти, для каждой из которых составляется список объектов, пересекающихся с ней. Способ разбиения выбирается таким образом, чтобы максимально ускорить процесс трассировки луча от одной подобласти к другой. При пересечении лучом подобласти просчитываются пересечения луча со всеми объектами списка этой подобласти. Если луч пересек один из них, то процесс трассировки можно считать завершенным. При пересечении луча с несколькими объектами выбирается ближайшая к началу луча точка пересечения. Методы, использующие пространственную иерархию объектов, существенно (в десятки и иногда сотни раз) сокращают объем расчетов пересечения луча с объектами и тем самым основную часть времени работы системы.

Другим подходом, дополняющим методы пространственного разбиения сцены, является *когерентная трассировка лучей* [3]. При ее использовании несколько лучей в пучке трассируются параллельно. Для наибольшего ускорения лучи из одного пучка обычно выбираются так, чтобы при трассировке они использовали одни и те же (или же близко расположенные) элементы данных (подобласти пространственного разбиения, объекты сцены и т.д.). Это сходство лучей называется «когерентностью лучей». Отсюда и происходит название метода. Для обеспечения когерентности первичных лучей в один пучок обычно помещают лучи, соответствующие соседним точкам экрана. С большой вероятностью они пересекаются с одним и тем же объектом сцены, отражаются и преломляются в близких направлениях. Таким образом, отраженные лучи и лучи, исходящие из точки пересечения в направлении на источник света («теневые» лучи), также в большинстве случаев оказываются когерентными и трассируются вместе.

Когерентная трассировка лучей хорошо соответствует возможности современных процессоров выполнять одинаковые операции над несколькими потоками данных одновременно через так называемые ОКМД (одна команда, много данных) команды. При правильной реализации когерентная трассировка лучей дает заметный выигрыш в производительности [3]. Полученное ускорение обычно равно количеству элементов данных, одновременно обрабатываемых ОКМД командой. Поэтому когерентная трассировка лучей стала активно развиваться с появлением ОКМД команд в современных процессорах, а именно расширения SSE в процессорах Pentium III и Pentium 4 [4]. Позже это же расширение было реализовано и в процессорах AMD, что сделало его фактическим стандартом для реализации когерентной трассировки лучей. По этой причине когерентная трассировка лучей часто называется «SSE трассировкой лучей». В настоящей статье используются оба термина.

Одна из реализаций когерентной трассировки лучей на базе SSE команд описана в [5], в данном случае для сцен, состоящих только из сфер. Поскольку

SSE команды работают одновременно с четырьмя вещественными числами одинарной точности, то такая трассировка может дать 4-х кратное ускорение по сравнению с реализацией трассировки с использованием обычной системы команд.

Сложные светорассеивающие свойства поверхности могут быть заданы двунаправленной функцией отражения/преломления (ДФО/ДФП). ДФО является одной из наиболее общих форм представления свойств материала в фотореалистичной визуализации. Она описывает перенос энергии между произвольной парой направлений падения и отражения луча света. В общем случае коэффициент переноса энергии может зависеть от длины волны и поляризации падающего света, хотя в большинстве случаев для синтеза реалистичных изображений трехкомпонентного представления цвета без учета поляризации бывает достаточно. Существуют различные способы представления ДФО. Один из них – использовать простые формулы, варьируя их параметры. Таковы, например, модели Фонга [6] и Блина [7]. Более сложное и более общее представление ДФО – это задание в табличном виде значений коэффициентов переноса энергии для набора пар входящих и исходящих направлений.

В разработанных нами системах моделирования освещенности и синтеза реалистичных изображений [1, 8, 9] активно используются ДФО, основанные на реальных физических данных. Они являются результатом либо измерений образцов материала на специальной установке [10], либо вычислений ДФО по заданной микроструктуре материала. Второй подход, в частности, используется для моделирования светорассеивающих свойств различных тканей [11]. ДФО, полученные любым из этих способов, имеют сложную структуру, которую трудно описать при помощи простой модели (например, модели Фонга). Для таких ДФО единственным способом представления является задание в табличной форме.

Необходимость реализации быстрой работы с ДФО на базе SSE команд следует, в частности, из публикации [5]. В ней описывается реальная ситуация,

когда обработка ДФО становится узким местом в случае, когда остальные компоненты системы работают достаточно быстро за счет SSE реализации. Поэтому реализация ДФО вычислений на SSE необходима, она дает определенное ускорение всего процесса генерации изображения.

Данная статья организована следующим образом. В разд. 2 дается обзор проблемы, и рассматриваются различные аспекты реализации вычислений ДФО на SSE. В разд. 3 приводятся результаты, а в разд. 4 рассматриваются возможные направления дальнейших исследований.

2. Вычисление ДФО на базе SSE технологии

Рассмотрим систему координат четырехмерной ДФО (рис. 1).

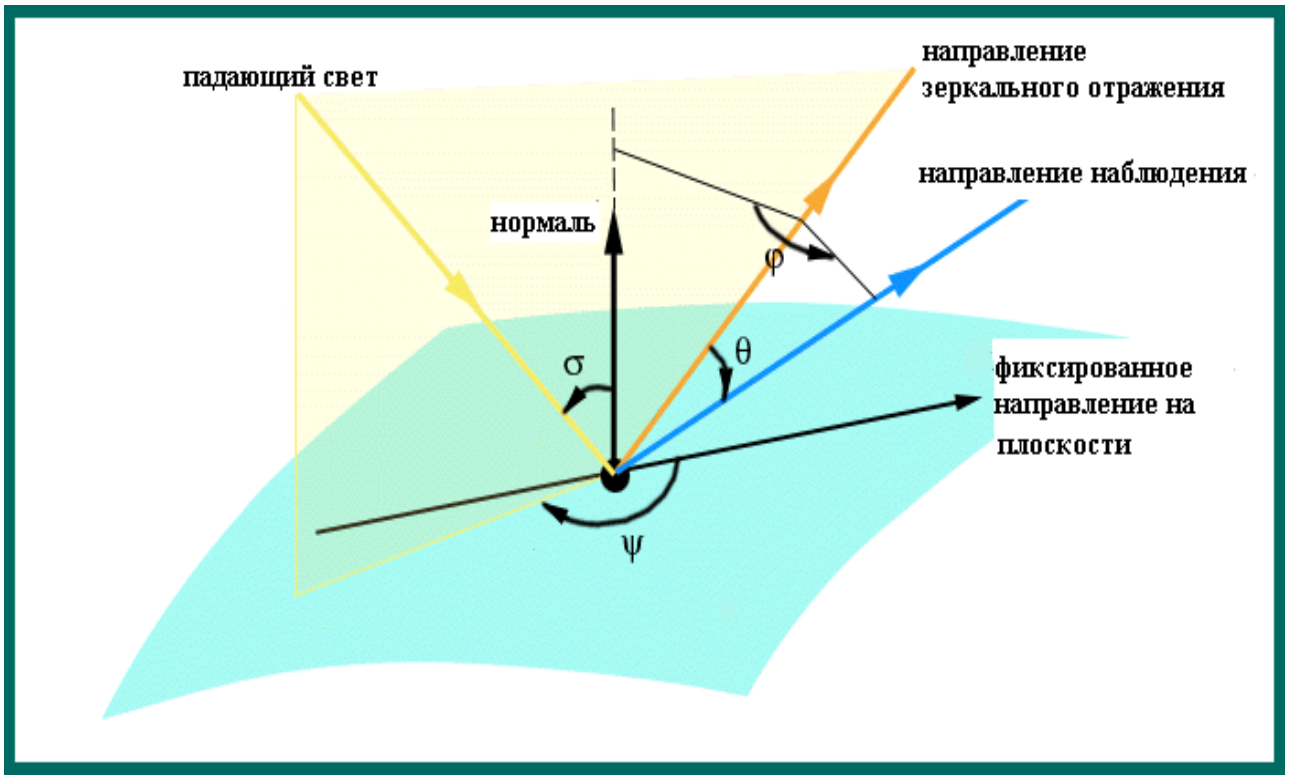

Рис. 1. Система координат 4-хмерной ДФО. Значения углов пояснены в тексте.

Величины σ, ψ, θ и φ являются угловыми координатами ДФО. За основное направление взято направление зеркального отражения света. Обычно именно в этом направлении отражается наибольшее количество света.

Координата σ задает угол падения света или угол зеркального отражения по отношению к нормали поверхности. Координаты θ и φ задают отклонение направления наблюдения от зеркального отражения, как показано на рис. 1. По количеству координат, от которых реально зависит значение ДФО, они могут быть разделены на изотропные, или трехмерные (зависят от 3-х координат σ, θ и φ), и анизотропные, или четырехмерные (зависят от всех четырех координат). Координата ψ в случае анизотропной ДФО задает поворот образца по отношению к заранее заданному фиксированному направлению на плоскости.

Так как ДФО задана в табличной форме (т.е. у нас имеется некоторая сетка значений), то значение функции в конкретной точке вычисляется при помощи линейной интерполяции. Следует заметить, что шаг сетки не должен быть равномерным по всем угловым координатам. Например, по координате θ типичная ДФО имеет максимум при $\theta = 0$, что соответствует пику отражения энергии в направлении зеркального отражения. При возрастании координаты θ интенсивность отраженной энергии быстро убывает. Следовательно, разумная сетка должна иметь мелкий шаг по θ координате около нуля, а при значениях θ, близких к π, шаг сетки может быть большим. Неравномерность сетки требует использования алгоритма бинарного поиска для определения элемента сетки, в котором будет производиться интерполяция.

Следует заметить, что сетка значений задается именно самими углами, а не их синусами или косинусами, чтобы можно было производить линейную интерполяцию. Следовательно, для каждого луча необходимо вычислять углы, что требует реализации обратных тригонометрических функций. Аппаратная реализация этих функций в процессорах x86 обрабатывает только одно значение в течение 250-300 тактов [12].

При реализации вычислений ДФО с использованием технологии SSE задача усложняется. Алгоритмы и приемы, ставшие классическими для обработки одного значения (например, индексирование массива или бинарный поиск), становятся не такими очевидными при переходе к ОКМД случаю.

Таким образом, задача реализации ДФО вычислений на базе SSE технологии состоит из следующих частей:

- вычисление обратных тригонометрических функций в SSE;
- выполнение бинарного поиска в SSE;
- выполнение индексирования и интерполяции в SSE.

Эти проблемы рассматриваются по отдельности в последующих разделах.

2.1. Вычисление обратных тригонометрических функций

Вычисления тригонометрических функций, реализованные аппаратно в процессорах x86, являются медленными [12]. Поскольку вычисление обратной тригонометрической функции требуется производить для каждой размерности ДФО (т.е. 3 раза для изотропной и 4 раза для анизотропной), это существенно замедлит процесс. Следовательно, ускорение вычисления тригонометрических функций является первоочередной задачей.

Заметим, что не требуется вычислять точные значения обратных тригонометрических функций, поскольку углы используются только для интерполяции. Как следствие, можно использовать приближенные формулы.

После рассмотрения различных аппроксимаций была выбрана следующая формула [13], дающая хорошую точность при умеренном времени вычисления:

$$\arcsin x \approx \frac{\pi}{2} - \sqrt{1-x}\left(\frac{\pi}{2} + a_1 x + a_2 x^2 + a_3 x^3\right)$$

Она вычисляет значения только для положительных аргументов. Для отрицательных аргументов используется соотношение arcsin(x) = -arcsin(-x), пользуясь нечетностью функции. Арккосинус вычисляется по формуле

$$\arccos x = \tfrac{\pi}{2} - \arcsin x$$

Получаемая при этом точность вычислений достаточна для данной задачи. График погрешности приведен на рис. 2:

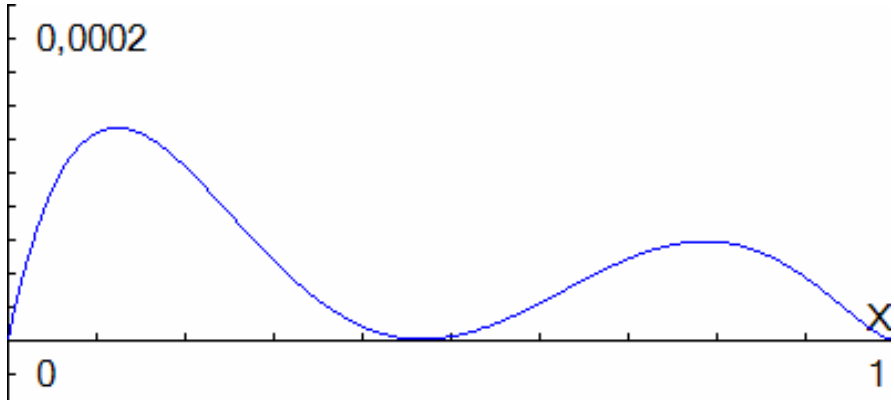

Рис. 2. Погрешность аппроксимации для арккосинуса

Табл. 1 содержит замеры производительности вычисления обратных тригонометрических функций и полученное ускорение.

Кол-во вычисленных значений, млн.	10	20	40
Вычисление без SSE (сек.)	1.479	2.954	5.908
Вычисление с SSE (сек.)	0.084	0.156	0.319
Достигнутое ускорение (раз)	17.6	18.93	18.52

Таблица 1. Замеры производительности для вычислений обратных тригонометрических функций с использованием SSE технологии и без нее.

Как видно из табл. 1, получен значительный выигрыш в производительности по сравнению со стандартными функциями, не использующими SSE технологию. Во время теста SSE версия вычисления вызывалась в четыре раза реже, так как она обрабатывает четыре значения функции за каждое обращение. Из расчета на равное количество вычисленных значений вычисления ускорены в 18 раз.

2.2. SSE реализация бинарного поиска

Следующим компонентом SSE реализации вычисления ДФО является бинарный поиск. Для не-SSE случая он является классическим алгоритмом. В случае использования SSE команд он значительно усложняется.

Основной проблемой при реализации бинарного поиска в SSE является обработка ветвлений, то есть ситуаций, в которых результаты сравнения для различных элементов четверки различны, и согласно алгоритму требуется идти в разные стороны. Для решения этой проблемы используется стек постоянного размера. Размер его равен количеству элементов в SSE минус 1, в данном случае это 3. Как только сравнения дают различные результаты, текущее состояние сбрасывается в стек и поиск идет по одной из ветвей. Когда текущая ветвь двоичного поиска завершена, индексы записываются в возвращаемое значение и проверяется стек. Если он пуст, то происходит возврат из процедуры. В противном случае из стека извлекается сохраненное состояние, и поиск продолжается в другом направлении. Сохраняемое состояние поиска состоит из текущих границ поиска и текущей маски, по которой ограничиваются все результаты сравнения. Начальное значение маски передается как параметр функции.

Псевдокод алгоритма выглядит следующим образом:

```
sse_int sse_binarysearch(array <float> arr, sse_float val, int mask)
 {
  i = 0, j = arr.length();
  sse_int res;
  while(true)
    {
     if (i == j)
       {
        res.set(i, mask);
        if (stack is empty)
          return res;
```

```
        st.pop(i, j, mask);
        continue;
        }
    k = (i + j) >> 1;
    cmp = val < sse_float(arr[k]);
    if (all_true(cmp, mask))
       j = k;
     else if (all_false(cmp, mask))
              i = k;
             else
              {
               push(false_mask(cmp, mask), i, k);
               j = k;
               mask = true_mask(cmp, mask);
              }
   }
}
```

Для повышения производительности существуют специализированные версии SSE бинарного поиска, которые сразу же вычисляют веса или значения одномерной функции. Это позволяет избежать дорогостоящих операций SSE индексирования.

Результаты замеров производительности бинарного поиска даны в таблице 2.

Количество вызовов, млн.	10	20	40
Вычисление без SSE (сек.)	0.9	1.78	3.55
Вычисление с SSE (сек.)	0.41	0.79	1.61
Достигнутое ускорение (раз)	2.23	2.25	2.20

Таблица 2. Замеры производительности для бинарного поиска с использованием SSE и без него.

Достигнутое ускорение составляет примерно 2.2 раза. Этого достаточно для быстрых вычислений ДФО.

2.3. Индексирование и интерполяция значений

Для получения окончательного результата необходимо произвести линейную интерполяцию табличных значений. Это значит, что после получения индексов ДФО таблицы требуется извлечь сами значения, т.е. выполнить операцию индексирования.

Хотя индексирование является тривиальным в случае одного элемента данных, оно становится совершенно нетривиальным при чтении 4-х значений из памяти. Самый простой способ – это взять 4 значения из памяти и поместить их в соответствующие позиции SSE-регистра (или SSE-переменной). Однако это является неэффективным в случае когерентных лучей, так как может привести к тому, что одно и то же значение читается из памяти несколько раз. Если учесть, что нужно прочитать 8 таких значений для изотропных ДФО (или 16 для анизотропных) и при этом каждое значение – это цвет, состоящий из 3 вещественных чисел, то подобная простейшая реализация может существенно замедлить вычисления.

Для ускорения индексирования алгоритм немного усложняется. Берется нулевой элемент SSE-переменной и сравнивается с остальными с учетом текущей маски. Если все активные элементы совпадают с нулевым, то они немедленно загружаются. В этом случае индексирование и интерполяция выполняются сразу же без использования дополнительных переменных, что экономит время. Если же не все активные индексы совпадают с нулевым, то в цикле проверяются все элементы, начиная с нулевого, и определяется минимальный активный элемент с наименьшим номером (номера элементов – от 0 до 3). Далее определяются все индексы элементов, совпадающих с минимальным. Для полученных индексов происходит загрузка данных ДФО, и они исключаются из категории активных. После этого процесс повторяется до тех пор, пока не будут загружены данные для всех элементов.

Для ускорения операции сравнения индексы хранятся в виде вещественных значений. Это также ускоряет вычисление приведенных индексов многомерной таблицы.

Данный алгоритм индексации и интерполяции дает ускорение до 4 раз для полностью когерентного случая и практически не дает ускорения для некогерентного случая. Таким образом, среднее ускорение составляет примерно 2.5 раза. При этом также необходимо учитывать, что обычная реализация интерполяции на четырехмерной сетке значений использует большое количество операций, и некоторые компиляторы могут использовать для оптимизации SSE-инструкции (хотя, конечно, и не так эффективно).

3. Результаты

Было реализовано вычисление значений ДФО, представленной в табличной форме, при использовании SSE технологии для обратной трассировки лучей. Обратная трассировка луча (трассировка в направлении, обратном распространению света – от наблюдателя к источнику света) используется в алгоритмах генерации реалистичных изображений.

Реализация была написана на языке C++ в среде Microsoft Visual Studio .NET 2003. Использовался стандартный компилятор, все опции оптимизации были включены. Ассемблерный код и ассемблерные вставки в реализации не использовались. Доступ к SSE осуществлялся через встроенные функции (intrinsic functions), которые в свою очередь, были организованы в специальные классы, обеспечивающие функциональность четверок вещественных чисел, четверок целых чисел или маски.

Измерения совокупного ускорения производились отдельно для изотропных и анизотропных ДФО. Результаты измерений приведены в табл. 3 и 4.

Количество вызовов	100000	200000	400000
Вычисление без SSE (сек.)	0.085	0.177	0.350
Вычисление с SSE (сек.)	0.024	0.050	0.101
Достигнутое ускорение (раз)	3.54	3.54	3.46

Таблица 3. Замеры производительности для вычисления изотропных ДФО с использованием SSE и без него.

Количество вызовов	100000	200000	400000
Вычисление без SSE (сек.)	0.137	0.248	0.495
Вычисление с SSE (сек.)	0.040	0.078	0.156
Достигнутое ускорение (раз)	3.43	3.17	3.17

Таблица 4. Замеры производительности для вычисления анизотропных ДФО с использованием SSE и без него.

Как видно из таблиц, удалось достичь ускорения порядка 3.1-3.5 раз. Однако конкретные значения зависят от характеристик машины и размера ДФО по каждому из измерений. Для измерений использовались ДФО следующих размеров:

- изотропная: 7 x 8 x 13;
- анизотропная 17 x 7 x 17 x 13.

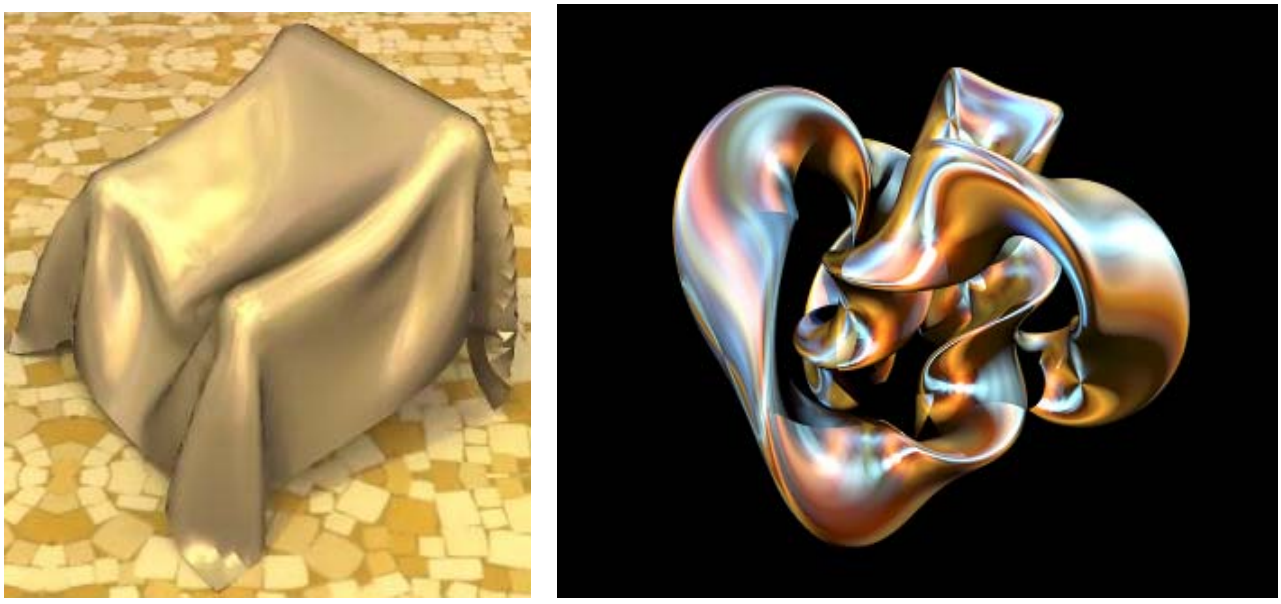

Рис. 3. Примеры изображений, полученных с применением вычислений анизотропной ДФО на базе SSE технологии.

Все замеры производились на ноутбуке с процессором Mobile Pentium-IV 1800 MHz Intel Centrino и 512 MB 433 MHz RAM.

На рис. 3 приведены примеры изображений, сгенерированных с применением вычислений значений ДФО на базе SSE технологии.

4. Заключение

В настоящей работе представлена реализация вычисления значений ДФО для обратной трассировки лучей с использованием SSE технологии. Было получено ускорение в 3 – 3.5 раза в зависимости от размера и размерности ДФО.

Ожидается, что ускорение будет еще выше, если для компиляции использовать компилятор Intel C++. Также дальнейшего ускорения можно ожидать от использования целочисленного набора команд SSE 2 и специальных команд SSE 3.

На данном этапе работы SSE-поддержка имеется только для ДФО с трехкомпонентным представлением цвета для обратной трассировки лучей. Однако программное обеспечение, разработанное нашим коллективом, поддерживает также спектральные ДФО и поляризацию, а также сложные алгоритмы глобального освещения [1]. Поскольку все эти алгоритмы основаны на трассировке лучей, можно получить дополнительных выигрыш в производительности, реализовав их поверх когерентной трассировки лучей. Для прямой трассировки лучей ускорение будет не столь большим, так как когерентность лучей здесь существенно ниже [14]. Для ускорения работы с глобальным освещением нужно модифицировать существующие алгоритмы и разрабатывать новые структуры данных. Существенный выигрыш в производительности можно получить для спектральных ДФО, поскольку они используют большое число цветовых компонент. В этом случае SSE команды могут работать не только с четверками цветов, но и с отдельными цветовыми объектами.

Список литературы

[1] Баяковский Ю.М., Галактионов В.А. О некоторых фундаментальных проблемах компьютерной (машинной) графики. "Информационные технологии и вычислительные системы", № 4, 2004, стр. 3-24.

[2] Turner Whitted, An Improved Illumination Model for Shaded Display. Communication of ACM, Vol. 23, № 6, June 1980, pp. 343-349.

[3] I. Wald, C. Benthin, M. Wagner, P. Slusallek, Interactive Rendering with Coherent Ray Tracing. Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001) Vol. 20, № 3, pp. 153-164.

[4] B. Patwardhan, Introduction to the Streaming SIMD Extensions in the Pentium III: Part I
http://www.x86.org/articles/sse_pt1/simd1.htm

[5] Real-Time Ray Tracing Realization. SSE Optimization.
http://www.digit-life.com/articles/rtraytracing/index.html.

[6] B. Phong, “Illumination for Computer Generated Pictures”, Communications of the ACM, vol. 18, № 6, pp. 311-317, 1975.

[7] J. F. Blinn, Models of light reflections for computer synthesized pictures. SIGGRAPH’77 Proceedings, pp. 192–198, 1977.

[8] A. Khodulev, E. Kopylov, Physically accurate lighting simulation in computer graphics software. Proc. GraphiCon’96 - The 6-th International Conference on Computer Graphics and Visualization, St.Petersburg, 1996.

[9] А.Г. Волобой, В.А. Галактионов, К.А. Дмитриев, Э.А. Копылов. Двунаправленная трассировка лучей для интегрирования освещенности методом квази- Монте Карло. "Программирование", № 5, 2004, с. 25-34.

[10] Letunov A.A., Barladian B.H., Zueva E.Yu., Veshnevetc V.P., Soldatov S.A. CCD-based device for BDF measurements in computer graphics. The 9-th International Conference on Computer Graphics and Vision, Moscow, Russia, 1999.

[11] Vladimir Volevich, Andrei Khodulev, Edward Kopylov, Olga Karpenko. An Approach to Cloth Synthesis and Visualization. The 7-th International Conference on Computer Graphics and Visualization, Moscow, Russia, 1997.

[12] IA-32 Intel Architecture Optimization Reference Manual, p. 440. ftp://download.intel.com/design/Pentium4/manuals/24896611.pdf

[13] Справочник по математике для научных работников и инженеров. Определения, теоремы, формулы. Под общей редакцией И.Г. Арамановича. «Наука», М. 1978.

[14] Johannes Günther, Ingo Wald, and Philipp Slusallek, “Realtime Caustics Using Distributed Photon Mapping”. Eurographics Symposium on Rendering, 2004.