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

К.А. Востряков, А.Г. Волобой
Институт Прикладной Математики имени М.В. Келдыша РАН,
Москва, Россия
{vostryakov, voloboy}@gin.keldysh.ru

Аннотация

В статье описывается новый адаптивный алгоритм устранения ступенчатости изображений с использованием 4-лучевой SSE (Intel Streaming SIMD Extension) трассировки лучей. Каждый раз, когда обнаруживается область с быстрым изменением функции изображения, необходимо наиболее эффективно трассировать 4 новых луча, а не пытаться трассировать индивидуальные лучи. Предложенный алгоритм использует SSE маску разности цветов ближайших лучей как индекс в таблицы разрывности. Таким образом, эффективно определяются зоны с быстрыми изменениями функции изображения. Было достигнуто среднее количество лучей на пиксел 1.5–2 при качестве аналогичном 25 лучам на пиксел.

Ключевые слова: *устранение ступенчатости, реалистичные изображения, когерентная трассировка лучей, SSE.*

1. ВВЕДЕНИЕ

История проблемы ступенчатости изображения в компьютерной графике начинается с работы [1], в которой впервые было показано, что проблема ступенчатости при синтезе изображений является примером хорошо известной проблемы в области обработки сигналов. В соответствии с теоремой Шеннона-Котельникова [2] ступенчатость может быть устранена либо низкочастотным фильтром (ниже частоты Найквиста), либо увеличением плотности выборки до необходимого уровня. Подробный обзор о состоянии дел в приложениях теоремы Шеннона-Котельникова приведен в работе [2].

Фильтрация может быть использована, когда функция яркости изображения известна, например, при выборке значений из текстуры [3]. Если функция неизвестна, то необходимо увеличивать размер выборки.

При синтезе изображений методом трассировки лучей [4] оцениваемая функция яркости неизвестна и определяется путем расчета освещенности видимой сцены и трассировки лучей из камеры через пиксели экрана. Это означает, что для устранения лестничного эффекта необходимо трассировать большое количество лучей. Как известно, трассировка лучей является дорогой операцией. В тоже время ступенчатость проявляется только в тех местах, где значение яркости быстро меняется, и в типичных сценах составляет от 1% до 30% пикселей. Обычно такие пиксели содержать либо геометрические границы объектов, либо границы тени от источников света. Поэтому многими исследователями были предложены адаптивные схемы устранения ступенчатости. Большинство адаптивных схем используют двухуровневую схему. На первом уровне выстреливается относительно небольшое количество одинаково распределенных лучей для определения точек с быстрым изменением яркости. Такие лучи часто называют пилотными. На втором этапе дополнительно обстреливаются только сложные пиксели.

Критерием адаптивности может быть дисперсия как в работах [5, 6] или контраст. Контрастный критерий более соответствует тому, как воспринимает излучение глаз человека [7]. Мы предлагаем использовать более общий подход, чем контрастный критерий, а именно, сравнение RGB составляющих цвета полученных с помощью оператора сжатия динамического диапазона [8].

Недавние исследования в области аппаратной акселерации трассировки лучей [9], [10], [11] позволили ускорить ее в несколько раз. Тем не менее, она все еще остается вычислительно емкой. Кроме того, в сценах все чаще используется задание материалов через двунаправленную функцию отражения (пропускания), которая, как отмечают многие, становится узким местом производительности.

Таким образом, успехи когерентной (SSE) трассировки лучей не снижают важности адаптивной схемы выборки функции изображения. Также SSE трассировка накладывает ограничения на такие схемы.

2. ТРЕБОВАНИЯ SSE АЛГОРИТМА

При использовании SSE лучи трассируются пучками по 4 луча одновременно, при этом каждый пучок лучей должен быть как можно более когерентным.

В работах [11] и [12], использующих когерентную трассировку лучей, пытаются применять перемежающуюся выборку для сглаживания лестничного эффекта. Такой подход позволяет обеспечить когерентность пучков лучей, но ведет к трудностям при оценке локальной ошибки и создании адаптивной схемы устранения ступенчатости.

Общий дизайн предлагаемого алгоритма специально разработан для SSE трассировки лучей. Каждый раз, когда обнаруживается разрыв функции яркости, необходимо наиболее эффективно трассировать 4 новых луча, а не пытаться трассировать индивидуальные лучи. Кроме того, алгоритм должен быть как можно более линейным. Другими словами, он должен содержать минимум ветвлений, которые снижают выгоду от использования SSE. Эти требования были учтены при разработке SSE алгоритма устранения ступенчатости.

Алгоритм делает предположения о том, что существуют два вида разрывов или быстрых изменений (далее просто разрывы) функции яркости: вертикальные и горизонтальные. Причем внутри одной точки предполагается существование разрыва только одного вида. Это не значит, что алгоритм плохо определяет другие виды разрывов. Однако эти виды

разрывов являются наиболее часто встречающимися в типичных, особенно архитектурных, сценах. И алгоритм пытается работать с ними наиболее эффективно.

3. ОПИСАНИЕ АЛГОРИТМА

Плоскость экрана разбивается на квадратные области («плитки») размером 64x64 пикселя, они могут визуализироваться независимо в разных исполняющих потоках. Для того чтобы алгоритм устранения ступенчатости работал правильно в граничных пикселях, плитки должны быть расположены с перекрытием в один пиксель.

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

3.1 Первый уровень

В начале 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, если 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.

На первом уровне используется регулярная выборка. Лучи трассируются четверками через углы пикселей. Значение цвета для каждого луча записывается в матрицу. После этого выбирается очередной пиксель из плитки, и находятся значения цветов его угловых лучей из матрицы (точки ABCD на рис. 1). Алгоритм определяет вид разрыва: горизонтальный или вертикальный. Горизонтальный означает, что существует высокий градиент между точками A и B или точками C и D. Сравнения, которые выполняются помощью одной SSE инструкций для определения вида разрыва, обозначены как a0, a1, a2, a3 на рис. 2. 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.

Введем метрику между двумя точками *p1, p2* на изображении:

$$\rho(p1, p2) = \max_{r,g,b} \left| color(p1) - color(p1) \right|$$

Коротко обозначим:

$$\rho(p1, p2) > eps \quad \text{как d(p1, p2),}$$

$$\rho(p1, p2) < eps \quad \text{как !d(p1, p2),}$$

где eps – порог ошибки.

Будем проверять следующие условия:

Если (!d(A, C) or !d(B, D)) and (d(A, B) and d(C, D)),

то горизонтальный разрыв,

иначе если (!d(A, B) or !d(C, D)) and (d(A, C) and d(B, D)),

то вертикальный разрыв,

иначе если (d(A, B) or d(C, D) or d(A, C) or d(B, D)),

то горизонтальный разрыв,

иначе если карта видимости н е обнаружила разрыв,

то нет разрыва.

Если алгоритм обнаружил вертикальный разрыв, то точки ABCD поворачиваются на 90 градусов против часовой стрелки (рис. 1). Этот поворот обеспечивает дальнейшую линейность алгоритма независимо от вида разрыва.

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.

3.2 Второй уровень

Внутри стреляем четверку лучей через точки E, F, G, H как показано на рис. 1. В случаи вертикального разрыва, будут использованы повернутые точки первого уровня, а для определения точек стрельбы, другая таблица смещения относительно левого верхнего угла пикселя.

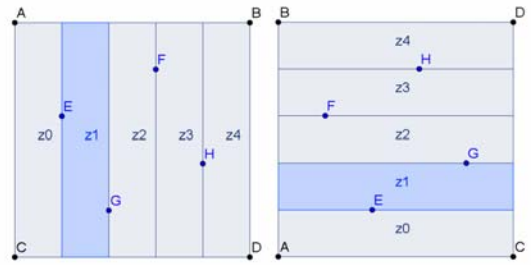


Рис 1: Слева: горизонтальный разрыв в зоне z1 между точками E и G. Справа: повернутая против часовой стрелки на 90 градусов вертикальная версия.

Для определения разрывов между точками первого и второго уровней используем 3 четверки сравнений по модулю b0 – b3, c00 – c03, c10 – c13 показанных на рис. 2. 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. Значениями этой таблицы являются флаги, которые определяют зоны разрывов: z0, z1, z2, z3, z4 (рис. 1). Значения, полученные из трех таблиц (3 четверки сравнений), логически складываются (дизъюнкция). Полученные флаги показывают зоны разрывов, где необходимо дополнительно стрелять лучи.

Если использовать только одну таблицу, то ее длина составит 2^{12} = 4096, что чересчур много. Поэтому использовалось 3 таблицы по 16 элементов (таблица 1). При такой организации важно, чтобы не было переоценки разрывности. Например, имеется разрыв, как показано на рисунке 2. Имеем для

первой четверки сравнений b0 – b3: d(E, F), d(E, G), !d(G, H), !d(F, H). Значит разрыв в зоне z1. О том, что в зоне z2 также разрыв должны проконтролировать сравнения c10 – c13. Если d(A, F), !d(C, E) то зоны z1, z2 содержат разрывы.

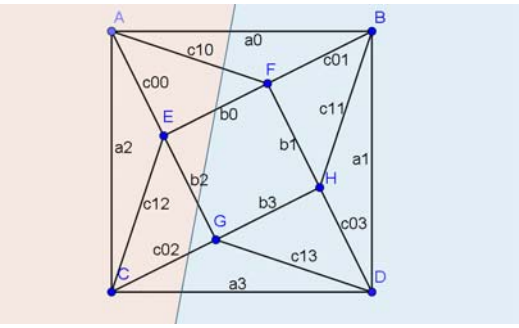


Рис. 2: Линии сравнения цветов выстрелянных лучей.

	b3b2b1b0	c03c02c01c0	c13c12c11c10
0000	0	0	0
0001	z1z2	z0	z1z2
0010	z3	z3	z4
0011	z2z3	z0z3	z1z2z4
0100	z1	z1	z0
0101	z1	z0	z0
0110	z1z2z3	z1z3	z0z4
0111	z1z2z3	z0z3	z0z4
1000	z2z3	z4	z2z3
1001	z2	z0z4	z2
1010	z3	z4	z4
1011	z1z2z3	z0z4	z2z4
1100	z1z2	z1z4	z0z2z3
1101	z1z2	z0z4	z0z2
1110	z1z2z3	z1z4	z0z4
1111	z0-z4	z0-z4	z0-z4

Таблица 2: Значения таблиц разрывности.

3.3 Третий уровень

На этом уровне в каждую разрывную зону выстреливаются четверка лучей зигзагом (рис. 3). Каждый зигзаг покрывает одну из разрывных зон (горизонтальную или вертикальную). Максимально выстреливается 5 зигзагов на пиксель. Заметьте, что точки каждого зигзага расположены с шагом 0.04 (сторона пикселя равна 1) вдоль высокого градиента разрыва (рис. 3). Другими словами, например, в случаи большого горизонтального градиента, на каждой вертикальной линии с шагом 0.04 по горизонтали расположено по одной точке зигзага. Кроме того, независимо от того горизонтальный или вертикальный случай используется, 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 разрывности (таблица 1). 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. 3 слева) по точкам A и E, а остальные три по точкам E и C. Зигзаг z1 интерполируется по точкам E, G, зигзаг z2 – по точкам G, F, зигзаг z3 – по точкам H, F. Точки зигзага z4 интерполируется по двум точкам из H, B, D.

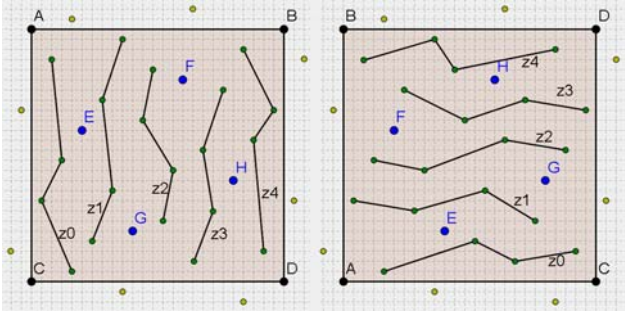


Рис. 3: Точки третьего уровня: слева горизонтальный вариант, справа вертикальный.

Если карта видимости определила разрыв внутри пикселя, а сравнения b0 – b3, c00 – c03, c10 – c13 его не нашли, то имеет место геометрические разрывы, которые были пропущены лучами первого и второго уровней, поэтому выстреливаются все лучи третьего уровня.

Расположение всех точек детерминировано. Поэтому можно предварительно вычислить для каждой из них вес, с которым ее цвет будет просуммирован для получения цвета конкретного пикселя. Это может существенно ускорить скорость работы алгоритма. Если используется прямоугольный фильтр (что бывает достаточно часто), то после обработки очередного пикселя в буфер кадра сразу записывает его цвет и никакой дополнительной информации о субпиксельных цветах хранить больше не нужно.

4. РЕЗУЛЬТАТЫ

Для тестирования использовался Dual Athlon MP 2.13 GHz. На рисунке 4 показан синтетический тест, используемый в работе [7]. Как видно из рисунка качество устранения ступенчатости предложенного адаптивного метода сопоставимо с качеством при трассировке 25 лучей на пиксель. Но так как использовалась детерминистическая выборка, то появился эффект муара (рис. 4c), который при случайной выборке проявляется в виде менее заметного шума (рис. 4b). Тем не менее, муар можно уменьшить, применив более совершенный фильтр [13] (рис. 4d), который при реконструкции принимает во внимание соседние пиксели.

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

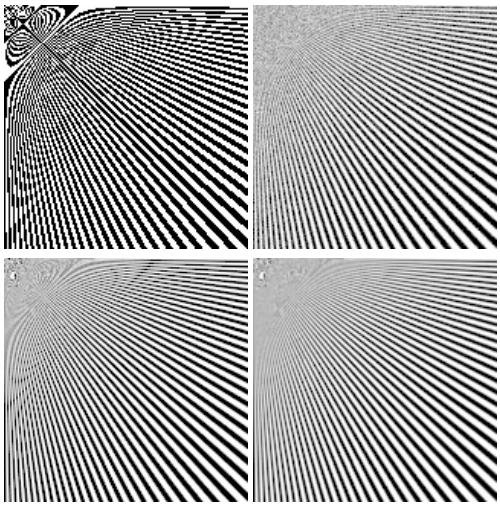


Рис. 4: Синтетический тест a) слева вверху: 1 луч на пиксель, b) справа вверху стратифицировано 25 лучей на пиксель с прямоугольным фильтром, c) слева внизу предлагаемый подход с прямоугольным фильтром и d) справа внизу предлагаемый подход с фильтром Mitchel [13].

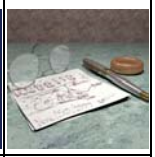
Сцена (число треуголь-ников, глубина трассировки, число источнико в света). Экран 1024x1024	Room (11.7k, 2, 9)	Car (141.5k, 3, 2)	Glass (44.9k, 2, 3)
			
Среднее число лучей на пиксель	1.75	1.5	2.01
Время (сек)	20.6	13.3	25.4

Таблица 2: Тесты на типичных сценах.

5. ЗАКЛЮЧЕНИЕ

Новизной алгоритма является использование битовой маски SSE сравнения в качестве индекса таблицы, определяющей зоны разрыва. Такой подход позволяет обрабатывать десятки и сотни случаев всевозможных сочетаний разностей между пилотными лучами. Впервые была разработана адаптивная схема устранения ступенчатости для когерентной SSE трассировки лучей.

При трассировке зигзага мы выбираем разрывную зону с быстроменяющейся функцией изображения. Поскольку такие разрывы чаще всего встречаются из-за разрывов геометрии, то пакет лучей, встретив разрыв, будет разделен, т.е. он не является высоко когерентным. Это снижает выгоду от SSE инструкций, но, тем не менее, она остается весомой.

Работа поддержана грантом РФФИ № 05-01-00345-а, а также компанией Integra Inc. (Япония).

6. ЛИТЕРАТУРА

[1] Crow, Franklin C., "The Aliasing Problem in Computer-Generated Shaded Images", Comm. ACM, Vol. 20, No.11, 1977, pp 799-805.

[2] Michael Unser, Sampling—50 Years After Shannon. Proceedings of the IEEE, vol. 88, no. 4, April 2000.

[3] Paul S. Heckbert. Fundamentals of Texture Mapping and Image Warping. Master's Thesis. Dept. of Electrical Engineering and Computer Science University of California, Berkley. 1989.

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

[5] Lee, Mark, Richard A. Redner, Samuel P. Uselton, "Statistically Optimized Sampling for Distributed Ray Tracing", Computer Graphics, Vol. 19, No. 3, July 1985, pp. 61-67.

[6] D.B. Kirk and J. Arvo, "Unbiased Sampling Techniques for Image Synthesis," Computer Graphics, vol. 25, no. 4, 1991, pp. 153-156.

[7] Mitchell, Don P., "Generating Antialiased Images at Low Sampling Densities," Computer Graphics, 21(4), July 1987, pp. 65-69.

[8] Б.Х. Барладян, А.Г. Волобой, В.А. Галактионов, Э.А. Копылов. Эффективный оператор сжатия динамического диапазона яркостей. "Программирование", №5, 2004, с. 35-42.

[9] Ingo Wald. Realtime Ray Tracing and Interactive Global Illumination. PhD thesis, Computer Graphics Group, Saarland University, Saarbrücken, Germany, 2004.

[10] Alexander Reshetov, Alexei Soupikov, and Jim Hurley. Multi-Level Ray Tracing Algorithm. ACM Trans. Graph., 24(3), 2005, pp.1176–1185.

[11] Carsten Benthin. Realtime Ray Tracing on current CPU Architectures. PhD theses. Computer Graphics Group. Saarland University. Saarbrücken, Germany. Jan, 2006.

[12] Solomon Boulos, Dave Edwards, J. Dylan Lacewell, Joe Kniss, Jan Kautz, Peter Shirley, Ingo Wald. Interactive Distribution Ray Tracing. SCI Institute, University of Utah, Technical Report UUSCI-2006-022.

[13] Mitchell, Don P., Netravali, Arun. "Reconstruction Filters in Computer Graphics," Computer Graphics, 22(4), 1988, pp. 221-228.

Antialiasing algorithm for a SSE 4-ray ray tracing

Abstract

The paper is devoted to new adaptive antialiasing algorithm with using 4-ray SSE ray tracing. Each time detecting large image variation, we try to shoot 4 new rays rather than trace individual rays. Proposed method uses SSE mask of neighbor rays color variation as an index in discontinuity zones table. In such way we detect regions with large variation fast. We achieved 1.5 – 2 rays per pixel with antialiasing quality about 25 uniform rays per pixel.

Keywords: *Antialiasing, coherent ray tracing, SSE.*

About the authors

Konstantin Vostryakov is a PhD student of the Keldysh Institute for Applied Mathematics RAS. vostryakov@gin.keldysh.ru.

Alexey Voloboy, PhD, senior researcher of the Keldysh Institute for Applied Mathematics RAS. voloboy@gin.keldysh.ru.