ИНТЕРАКТИВНАЯ ТРАССИРОВКА ЛУЧЕЙ НА ГРАФИЧЕСКОМ ПРОЦЕССОРЕ С ПРИМЕНЕНИЕМ ТЕХНОЛОГИИ CUDA

Владимир Фролов, Алексей Игнатенко
Московский государственный университет им. М.В. Ломоносова,
факультет вычислительной математики и кибернетики, Москва, Россия

{vfrolov,ignatenko}@graphics.cs.msu.ru

1. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

В отличие от работы [1], мы использовали разделяемую память видеокарты только для размещения в ней часто используемых, изменяемых данных относительно большого размера (больше 32 байт). Луч и информация о пересечении луча как раз относятся к таким данным. В нашей реализации, низкоуровневая оптимизация кода имела особенность - вынос кода функций с глубоким уровнем вложения (например пересечение луча и треугольника), что позволило:

1. Встраивать текстурную выборку непосредственно в вынесенный код, экономя регистры и эффективно использовать текстурный кэш.
2. Снизить количество ветвлений, что важно на CUDA из-за SIMD архитектуры мультипроцессоров.

1.1 Алгоритм прослеживания пути луча

В данной работе была предпринята попытка использвать классический для CPU алгоритм прослеживания пути луча в kd дереве с длинным стеком. Один узел kd дерева занимает в памяти всего 8 байт (что экономит память и позволяет эффективно использовать текстурный кэш). Классический алгоритм проходит каждый узел не больше одного раза и требует мало памяти на узел — в этом его достоинство. Стек реализован на локальной памяти. Данный подход необычен для GPU, т.к. локальная память имеет невысокую скорость и интересным результатом является то, что производительность полученной системы при этом оказалась удовлетворительной. Это может быть объяснено относительно нечастой потребностью на чтение из стека (только в случае промаха луча в узле kd-дерева) и когерентностью потоков внутри блока, что позволяет снимать весь warp (объединение из 32 потоков) с мультипроцессора при одновременном обращении к локальной памяти всех потоков внутри него.

1.2 Небольшая модификация классического алгоритма

Аналогично алгоритму kd-tree restart [2], при промахе луча внутри узла kd-дерева, его ближняя точка пересечения приравнивается к дальней. Это позволяет сократить количество обращений к локальной памяти на 30% при реализации стека.

2.РЕЗУЛЬТАТЫ

Созданная нами программа позволяет в интерактивном режиме визуализировать сцену, состоящую из тысяч примитивов, с многократными отражениями и преломлениями. Скорость визуализации сравнима с существующими аналогами GPU трассировщиков. Проведенное сравнение с существующим трассировщиком лучей на CUDA [1] показывают, насколько сильное влияние правильное использование ресурсов может сказаться на производительности (рис. 2). Однако, сравнение с большинством другими GPU-трассировщиков ([2],[3]) на данный момент затруднено из-за различия во входных данных и аппаратуры.

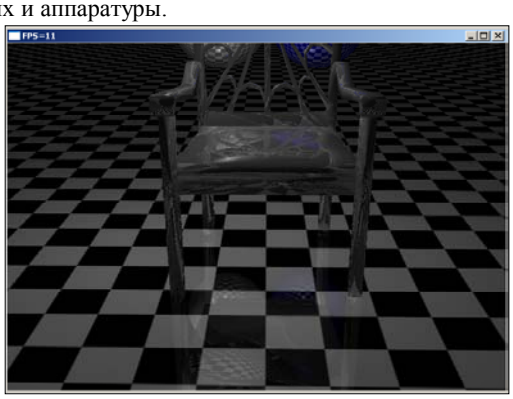

Рисунок 1: стул из 1464 треугольников, 4 переотражения.

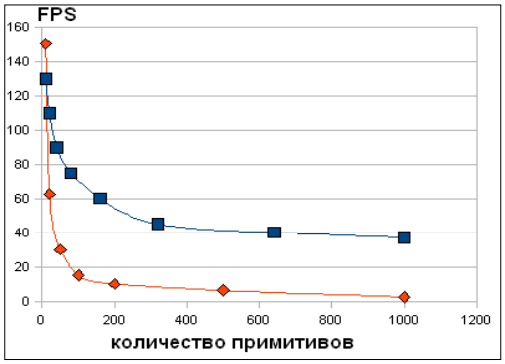

Рисунок 2: производительность трассировщиков. A – из работы [1]. B - полученной системы.

3.ЛИТЕРАТУРА

[1] http://eric_rollins.home.mindspring.com/ray/cuda.html

[2] Foley T., Sugerman J. KD-Tree Acceleration Structures for a GPU Raytracer. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, p. 15-22, 2005.

[3] Horn D., Sugerman J., Houston M., Hanrahan P. Interactive k-D Tree GPU Raytracing. Proceedings of the symposium on Interactive 3D graphics and games on Fast rendering, p. 167-174, 2007.