УДК 004.92

С.А. Зори

Донецкий национальный технический университет, Красноармейск, Украина

GPU-РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ 3D-СТЕРЕО ВИЗУАЛИЗАЦИИ С ИСПОЛЬЗОВАНИЕМ МЕТОДА ТРАССИРОВКИ ЛУЧЕЙ

В статье рассмотрен общий подход к организации вычислительных процессов в параллельной системе 3D- стерео визуализации и отображения процедуры 3D- стерео синтеза на архитектуру GPU. Выполнено экспериментальное исследование реализации модифицированного алгоритма поиска пересечений на GPU, как одного из наиболее затратных этапов рейтрейсинга. Полученные результаты показали существенный прирост производительности при многопоточной реализации на архитектуре GPU тестовой системы, который может быть улучшен при реализации на новых микроархитектурах GPU и оптимизации конфигурации CUDA- сети.

Ключевые слова: *3D- стерео визуализация, изображение, синтез, трассировка лучей, ускоряющие техники, архитектура, GPU, реализация, эффективность.*

Введение

Повышение скорости и качества систем 3D-пространственной визуализации на основе методов трассировки лучей, позволяющие синтезировать высококачественные 3D- изображения, сегодня является актуальной и перспективной научной задачей [1].

Ведущим перспективным направлением повышения производительности графических систем вычислительных систем сегодня является массовое параллельное выполнение вычислений и разработка соответствующей архитектуры систем.

Особо популярным способом реализации аппаратного ускорения на параллельных вычислительных системах является использование технологий вычислений общего назначения GPGPU на параллельных графических мультипроцессорах, технологий гетерогенных (гибридных) вычислений с использованием разнородных вычислительных устройств - CPU + GPU, а также параллельных GPU-кластеров (GPC). Это наиболее распространенные практические варианты реализации на сегодняшний день, де-факто ставшие самыми доступными подходами к реализации параллельных многопоточных вычислений. Однородность операций в подходах и алгоритмах 3D- визуализации позволяет предположить их относительно простую реализацию и хорошее ускорение на высокопроизводительных GPU-системах [1, 2]. В работе рассмотрена организация вычислительных процессов для параллельной системы 3D-стереовизуализации на основе разработанных автором подходов и алгоритмов [1, 3 – 6].

1. Архитектура системы 3D- стерео визуализации

С точки зрения практической реализации системы 3D- стерео визуализации на выбранном базисе средств параллельной аппаратной поддержки (GPU), напомним, что синтез 3D- стереоизображений повышенного качества предложено выполнять как двойной рейтрейсинг независимых изображений стереопары [1 – 3, 6]:

- параллельная независимая реализация синтеза кадров «левый» - «правый» на GPU;
- параллельная «внутрикадровая» реализация рендеринга методом трассировки лучей на ресурсах GPU, выделенных под каждый канал.

Далее (по необходимости, с учетом конкретики устройства 3D-отображения) на GPU также может быть проведен процесс постобработки полученных изображений стереопары (преобразование кадров в один из форматов стандартного вывода для устройств 3D- визуализации). В связи с этим, общая организация вычислительного процесса процедуры 3D- стерео синтеза изображения методом трассировки лучей на архитектуре GPU может быть проиллюстрирована рис. 1.

Блоки CUDA параллельно выполняют процедуру трассировки для левого и правого кадров изображения, которые разбиты на одинаковое количество растровых кластерных блоков (под-массивов растра), внутри которых также параллельно выполняется SIMT трассировка лучей.

Необходимо отметить, что топологическая организация сети и блоков не единственна и может быть предметом оптимизации для конкретной архитектуры GPU. Для предварительной настройки конфигурации можно воспользоваться, например, CUDA GPU Occupancy Calculator [7]. Отображение процедуры 3D- синтеза на структуру GPC можно представить аналогично, например разместив блоки BLF и BRF в различные кластеры.

Основной базовой процедурой 3D- стерео визуализации при этом является трассировка лучей.

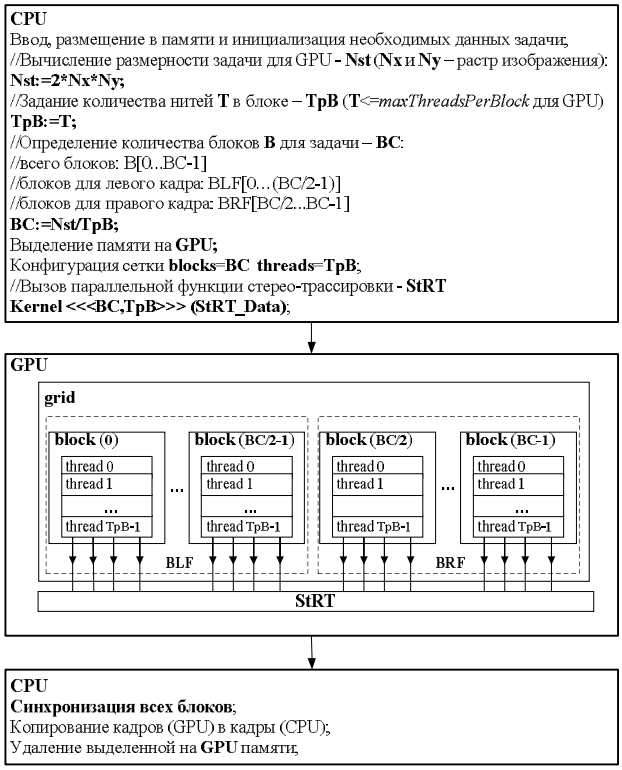

Рис. 1. Отображение процедуры
3D- стерео синтеза на GPU/GPC

При отображении архитектуры рейтрейсинга на архитектуру GPU основной идеей является разделение алгоритма трассировки лучей на несколько kernel- ядер, каждое из которых выполняет только одну конкретную задачу. Также следует учитывать особенности конкретной микроархитектуры GPU, а также особенности самого алгоритма трассировки лучей - параллельная реализация алгоритмов рейтрейсинга осложнена большим количеством факторов, как, например, возможное завершение потоков выполнения в разные моменты времени (в случае, если часть лучей поглощена или не пересекла ни одной поверхности сцены), некогерентность при расчете вторичных лучей, приводящая к разветвлению потоков (если разные лучи пересеклись с поверхностями, которые имеют разные свойства материалов и модели рассеяния света). Следует также учитывать, что для осуществления поиска пересечения луча с поверхностями сцены и расчета освещения в найденной точке потоки выполнения требуют регулярного обращения к описанию сцены, поэтому эффективная организация ускоряющей структуры, размещение данных в памяти и доступ к ней также является немаловажным фактором.

Заметим, что ввиду актуальности проблемы параллельной организации рейтрейсинга, и в первую очередь на GPU, в литературе предложено множество практических решений. Наиболее удачным в смысле универсальности и показанных качественных показателей, является глобальная концепция, предложенная в [8]. При этом авторы в [8, 9] указывают, что одним из наиболее трудоемких этапов визуализации является поиск ближайшего пересечения луча с объектами сцены, для повышения производительности которого применяют ускоряющие структуры.

В связи с вышеизложенным, представляется нецелесообразным рассматривать и разрабатывать подходы к реализации на архитектуре GPU общей методики трассировки лучей (достаточно воспользоваться универсальным подходом, либо более узкоспециализированными, с принятыми упрощениями), а сосредоточиться на ускорении «узких мест» этапов методов путем их отображения на архитектуру GPU.

2. Реализация модифицированного алгоритма поиска пересечений на архитектуре GPU

В [1, 4, 5] для ускорения нахождения пересечений при трассировке лучей была предложена модификация алгоритма нахождения пересечения луча с использованием ускоряющей техники на основе двухуровневой иерархии ограничивающих объемов и AABB для ускорения многопоточных вычислений. Модификация алгоритма была разработана в соответствии с рекомендациями по оптимизации, одной из важнейших которой является уменьшение ветвлений в пределах каждого отдельного блока и warp'a.

Рассмотрим экспериментальные результаты оценки быстродействия предложенной модификации алгоритма поиска пересечений при его реализации на GPU. Конфигурация тестового стенда, на котором проводились исследования, приведена в табл. 1. Следует отметить, что конфигурирование сетки GPU в общем случае неоднозначная задача, решение которой непосредственно влияет на производительность программы, выполняемой GPU. Кроме того, необходимо учитывать также физические ограничения архитектуры конкретного GPU.

Таблица 1

Конфигурация тестового стенда

OS	CPU	RAM	GPU
Windows 8 Ultimate, x64	Intel Pentium Dual-Core T4300 @ 2,10 ГГц	3 ГБ DDR3	NVIDIA GeForce G110M

Степень параллелизма и производительность зависят от: количества регистров мультипроцессора на нить; количества нитей на блок; количества общей (shared) памяти на нить; максимального количества блоков на мультипроцессор; интенсивности и характера обращений к глобальной памяти GPU; наличия расхождения потоков выполнения в пределах варпов. Как было указано в табл. 1, оценки производительности реализации системы на CUDA осуществлялись с помощью GPU NVIDIA GeForce G110M (Compute capability 1.1, 1 поточный мультипроцессор с 16 ядрами [10]).

Для определения теоретической оценки степени загрузки мультипроцессора и конфигуриро-

вания вычислительной сети GPU использовалась утилита CUDA SDK Occupancy Calculator [7]. Согласно теоретической оценке, максимальная загрузка мультипроцессора (100%) возможна при использовании менее 10 регистров на нитку, и размерах блоков в 192, 256 или 384 нити. Загрузка мультипроцессора GPU в 67% возможна при использовании 16 регистров на нитку, и размерах блоков в 192, 256 или 512 нитей. Большее количество выделенных регистров на нитку приводит к понижению загрузки мультипроцессора GPU до 50% (от 17 до 20), до 33% (от 21 до 32) и ниже.

Однако результаты проведенных экспериментов на этих двух конфигурациях показали, что, при теоретической загрузке мультипроцессора в 100% скорость работы выполнения всего алгоритма оказалось меньше, чем при конфигурациях сети GPU с 67% загрузкой (была выбрана для дальнейших экспериментов). Одна из причин – более интенсивное использование глобальной памяти при малом количестве выделенных регистров. Заметим, что это не означает ее оптимальности по быстродействию, экспериментальные исследования по оптимизации вычислительного процесса в CUDA- сети (как использованного для эксперимента GPU, так и на других, более мощных архитектурах GPU) не проводились. Поэтому полученные и описанные ниже экспериментальные оценки могут быть существенно улучшены при реализации на новых GPU с большим количеством вычислительных ядер и мультипроцессоров, а также проведении оптимизации конфигурации вычислительной CUDA- сети. Результаты экспериментальных оценок повышения быстродействия модифицированного алгоритма по сравнению с базовой версией при реализации на архитектуре GPU тестовой системы показаны на рис. 2 и 3.

Видно, что модифицированный алгоритм дает прирост производительности по отношению к базовому варианту от 27% до 32% при его реализации на GPU тестового стенда (однопоточная CPU реализация показала прирост производительности 3-11%), при этом оптимизация реализации на конкретной архитектуре используемого GPU не проводилась.

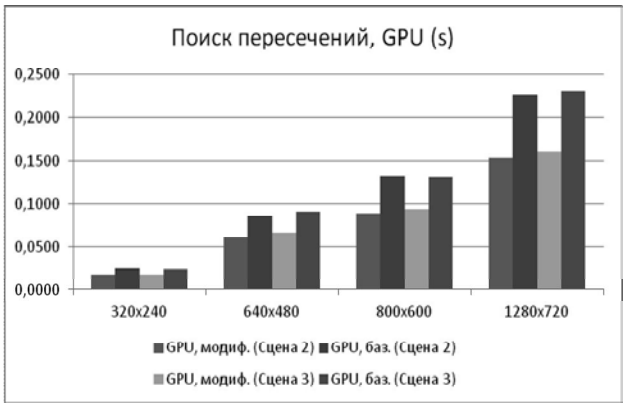

Рис. 2. Время поиска пересечений, GPU

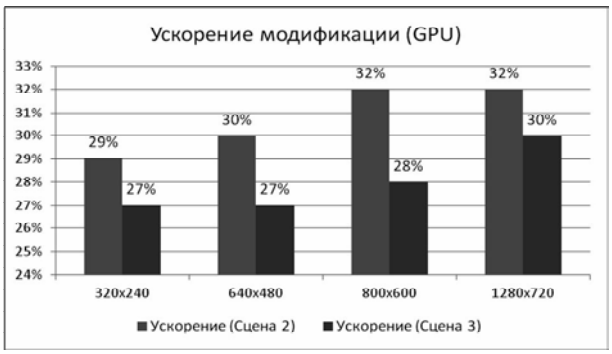

Рис. 3. Эффективность модификации, GPU

Результаты экспериментальных исследований времени реализации на GPU тестовой системы предложенной модификации алгоритма поиска пересечений показаны на рис. 4. Диаграмма обобщает результаты оценок производительности программной реализации алгоритма и позволяют сравнить между собой временные показатели, которые были получены при выполнении модифицированного алгоритма поиска пересечения луча с AABB при однопоточной реализации на CPU, четырех потоках на CPU (4 Hyper Threding ядра на 2 физических ядрах) и на тестовом GPU (16 CUDA ядер).

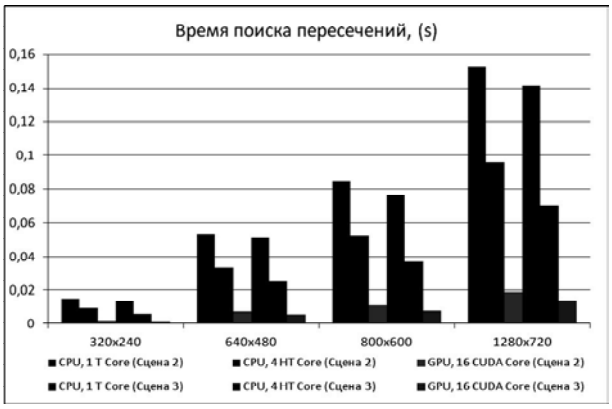

Рис. 4. Время поиска пересечений в тестовых сценах

На рис. 5 показано общее достигнутое ускорение (минимальное, максимальное и среднее значения) при многопоточной реализации на архитектурах CPU и GPU, использованных в тестовых системах.

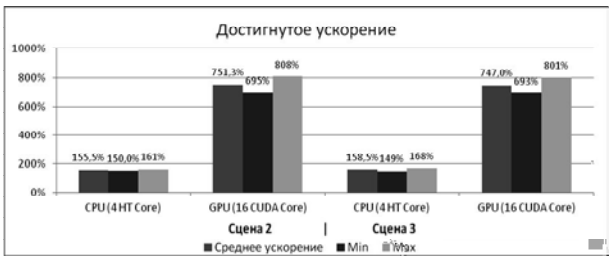

Рис. 5. Ускорение при многопоточной реализации на тестовых CPU и GPU

Реализация на тестовом GPU позволила получить среднее ускорение в 750%. Еще раз следует отметить, что полученные и описанные выше экспе-

риментальные оценки могут быть существенно улучшены при реализации на новых архитектурах GPU с большим количеством вычислительных ядер и мультипроцессоров, а также проведении оптимизации конфигурации вычислительной CUDA- сети (при экспериментах не использовалась).

Выводы

В статье рассмотрен общий подход к организации вычислительных процессов в параллельной системе 3D- стерео визуализации и отображения процедуры 3D- стерео синтеза на архитектуру GPU.

Выполнено экспериментальное исследование реализации модифицированного алгоритма поиска пересечений на GPU, как одного из наиболее затратных этапов рейтрейсинга.

Полученные результаты показали прирост производительности по отношению к базовому варианту от 27% до 32% при его реализации на GPU тестового стенда и общее среднее достигнутое ускорение при многопоточной реализации на архитектуре GPU тестовой системы в 750%.

Полученные экспериментальные оценки могут быть существенно улучшены при реализации на новых микроархитектурах GPU с большим количеством вычислительных ядер и мультипроцессоров, а также проведении оптимизации конфигурации вычислительной CUDA- сети.

Список литературы

1. Башков Е.А. Реалистическая пространственная визуализация с использованием технологий объемного отображения. Монография / Е.А. Башков, С.А. Зори. – Донецк : ГВУЗ "ДонНТУ", 2014. – 150 с.

2. Башков Е.А. Использование вычислительных возможностей графических систем для организации визуализации трехмерных сцен с использованием технологий объемного отображения / Е.А. Башков, С.А. Зори // Известия ЮФУ. Техн. науки. – 2014. – № 4 (153). – С. 15-21.

3. Зори С.А. Синтез стереоизображений на параллельных графических вычислительных системах / С.А. Зори // Комп'ютерна графіка та розпізнавання зображень – Збірник наукових праць НПК. – Вінниця: Вінницький обласний інститут післядипломної освіти, 2012. – С. 75-79.

4. Zori S.A. Productivity Increasing of Realistic Ray Tracing Stereo- Image Synthesis / S.A. Zori, P.A. Porfirov // Journal of Qafqaz University, Mathematics and Computer Science ISSN 1302-6763, Vol. 3, No 1, 2015. – P. 30- 38.

5. Зори С.А. Объемная визуализация алгоритмом трассировки лучей с использованием двухуровневой иерархии ограничивающих объемов и AABB // Наукові праці Донецького національного технічного університету. Серія: "Інформатика, кібернетика та обчислювальна техніка". – 2015. – №2 (21). – С. 5-10.

6. Зори С.А. 3D-визуализация сцен методом трассировки лучей на параллельном графическом процессоре/ С.А. Зори // Наукові праці Донецького національного технічного університету. Сер.: Інформатика, кібернетика та обчислювальна техніка. – 2014. – №1. – С. 32-37.

7. CUDA Occupancy Calculator Helps pick optimal thread block size [Электронный ресурс]. – Режим доступа: https://devtalk.nvidia.com/default/topic/368105/cuda-occupancy-calculator-helps-pick-optimal-thread-block-size/

8. Боголепов Д., Сопин Д., Ульянов Д., Турлапов В. Система интерактивного расчета глобального освещения для гибридных сцен // Труды 22-й межд. конф. по компьютерной графике и зрению GraphiCon'2012. – М.: МАКС Пресс, 2012. – С.. 227-232.

9. Фролов В. Интерактивная трассировка лучей и фотонные карты на GPU / В. Фролов, А. Игнатенко // Proc. of the 19th Int. Conf. on Computer Graphics and Vision GraphiCon'2009, MAKS Press, Moscow. – C. 255-262

10. CUDA LEGACY GPUs [Электронный ресурс]. – Режим доступа: https://developer.nvidia.com/cuda-legacy-gpus.

Поступила в редколлегию 31.03.2016

Рецензент: д-р техн. наук, ст. научн. сотр. С.Г. Семенов, Национальный технический университет «ХПИ», Харьков.

GPU- РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОЇ ОБЧИСЛЮВАЛЬНОЇ СИСТЕМИ 3D-СТЕРЕО ВІЗУАЛІЗАЦІЇ З ВИКОРИСТАННЯМ МЕТОДУ ТРАСУВАННЯ ПРОМЕНІВ

С.А. Зорі

У статті розглянутий загальний підхід до організації обчислювальних процесів в паралельній системі 3D-стерео візуалізації і відображення процедури 3D-стерео синтезу на архітектуру GPU. Виконано експериментальне дослідження реалізації модифікованого алгоритму пошуку перетинів на GPU, як одного з найбільш витратних етапів рейтрейсінгу. Отримані результати показали істотний приріст продуктивності при багатопотоковій реалізації на архітектурі GPU тестової системи, який можна покращити при реалізації на нових мікроархітектурах GPU і оптимізації конфігурації CUDA- мережі.

Ключові слова: *3D-стерео візуалізація, зображення, синтез, трасування променів, прискорюючі техніки, архітектура, GPU, реалізація, ефективність.*

VISUALIZATION 3D-STEREO PARALLEL COMPUTER SYSTEM GPU-REALIZATION WITH THE USE OF RAYS TRACING METHOD

S.A. Zori

In the article the general going is considered near organization of calculable processes in the parallel system of 3D-stereo visualizations and reflections of procedure of 3D- stereo synthesis on architecture of GPU. Experimental research of realization of the modified algorithm of search of crossings is executed on GPU, as one of the most expense stages of RAY-tRAYsing. The got results rotined the substantial increase of the productivity during многопоточной realization on architecture of GPU of the test system, which can be improved during realization on new micro architectures of GPU and optimizations of configuration of CUDA- of network.

Keywords: *3D- stereo visualization, image, synthesis, tracing of rays, accelerating techniques, architecture, GPU, realization, efficiency.*