УДК 004.925.86

Е.А. Башков, О.А. Авксентьева, Н.О. Харченко
Донецкий национальный технический университет
bashkov@pmi.dgtu.donetsk.ua

Возможности платформы CUDA при реализации воксельного представления дуг для объемных 3D дисплеев

Статья посвящена рассмотрению вопросов, связанных с параллельной реализацией модифицированного алгоритма генерации дуги, заданной произвольными параметрами в трехмерном пространстве объемного устройства отображения информации, с помощью средств, предоставляемых программно-аппаратной архитектурой CUDA. Рассмотрены различные методы оптимизации модифицированного алгоритма генерации дуги с точки зрения параллельных вычислений, проведен их сравнительный анализ на основе полученных относительных и абсолютных показателей, характеризующих эффективность и точность данного алгоритма.

Введение

Посредством зрительного аппарата человек получает более 75% всей воспринимаемой им информации. Абсолютное большинство образов, распознаваемых человеческими зрительными органами, являются трехмерными объектами, которые в свою очередь представляют собой отдельные составляющие более масштабных трехмерных сцен.

Основным средством коммуникации средств вычислительной техники и человека на сегодняшний день остаются устройства отображения информации, посредством которых результаты обработки некоторых данных компьютером предоставляются пользователю [1]. Отображение данных в виде естественных или искусственных графических образов стало в настоящее время самым распространенным способом доведения человеку больших объемов информации. На данный момент одним из наиболее перспективных направлений в области развития визуализирующих технологий является разработка объемных (volumetric) трехмерных устройств отображения информации – объемных 3D дисплеев [2, 3]. В такого рода устройствах «единичным» элементом отображения является в некотором смысле малый объем трехмерного пространства - воксель. Объемные 3D устройства визуализации подразумевают отображение сцен, источником которых служат вокселизированные объекты, процесс построения которых, например, с помощью трехмерных сканеров, достаточно подробно описан в литературе [4, 5] в то время, как методы и алгоритмы генерации типичных графических примитивов практически не рассматриваются.

Разработка новых методов и способов построения графических образов в пространстве трехмерных устройств отображения является первоочередной задачей, решение которой может значительно ускорить внедрение объемных трехмерных устройств отображения информации [5] в повседневную жизнь человека.

Постановка задачи воксельного разложения дуги

При постановке задачи принимается [6, 7], что некоторая часть трехмерного евклидова пространства, которое отображается объемным 3D дисплеем, имеет вид трехмерного параллелепипеда:

$$\Omega \circledcirc R^3,\ 0 \le x \le X,\ 0 \le y \le Y,\ 0 \le z \le Z.$$

С учетом возможности масштабирования считается, что X=Y=Z=H, то есть Ω – трехмерный куб. Пространство Ω заполнено вокселями – атомарными элементами, которые отображаются 3D дисплеем. Полагается, что воксель представляет куб с единичной стороной. Центры «соседних» вокселей удалены друг от друга по координатным осям на единичное расстояние. Множество вокселей, заполняющих Ω можно представить как трехмерный массив вокселей $V_{i,j,l}$, где i,j,l – индексы, принимающие значения 1… int(H).

Задача построения воксельного разложения произвольной пространственной дуги заключается в нахождении такой последовательности вокселей, в которой каждый из вокселей лежит, в некотором смысле, «недалеко» от линии дуги и «плотно» заполняет промежуток между начальной и конечной

точками. Данный графический примитив во время отображения его в пространстве 3D дисплея должен восприниматься наблюдателем как дуга в трехмерном пространстве (с учетом масштабирования).

В [7] предложен подход к решению задачи нахождения множества вокселей, которые задают воксельное представление дуги в пространстве трехмерного устройства отображения информации. В [8] разработаны методы генерации графического примитива, воспринимающегося наблюдателем в качестве дуги во время отображения множества найденных вокселей в пространстве 3D дисплея, а также предложены основные пути оптимизации данного алгоритма с точки зрения производительности за счет использования параллельных архитектур в качестве вычислительной среды. В частности, предложен метод ускорения процесса нахождения множества вокселей за счет применения программно-аппаратной платформы CUDA.

Развитие и особенности платформы CUDA [9]

Сам термин GPU был впервые использован корпорацией Nvidia для обозначения того, что графический ускоритель, первоначально используемый только для ускорения трехмерной графики, стал мощным программируемым устройством (процессором), пригодным для решения широкого класса задач, никак не связанных с графикой. Дальнейшая эволюция графических ускорителей (Voodoo, RivaTNT, GeForceFX) подтверждает тезис о том, что традиционные задачи рендеринга целесообразно решать параллельно, т.к. все вершины и полученные в результате растеризации треугольники можно обрабатывать независимо друг от друга. Добавление поддержки текстур со значениями в форматах с плавающей точкой позволило применять фрагментные процессоры для обработки больших массивов данных. По сути это привело к созданию некого параллельного процессора с оригинальной архитектурой, на вход которого можно передать большой массив данных и программу для их обработки и на выходе получить массив результатов. В результате возникло направление GP GPU (General Purpose computing on Graphics Processing Units) – использование графических процессоров для решения неграфических задач. Наиболее популярным представителем GP GPU являются графические процессоры фирмы Nvidia - программно-аппаратная платформа CUDA, на которой реализована концепция многоядерной мультипоточности. Являясь одним из методов повышения производительности графических вычислений, использование данной технологии дает возможность значительно уменьшить время генерации объектов сцены, отображаемой в пространстве трехмерного дисплея.

Одним из основных отличий графических процессоров (GPU) от центральных процессоров (CPU) является их аппаратное построение и организация памяти.

Программно-аппаратная платформа CUDA строится на базе взаимодействии центрального процессора (host) и графического процессора, выступающего в качестве массивно-параллельного сопроцессора. При этом стандартный последовательный код выполняется на CPU, а функции, реализующие параллельные вычисления, выполняются на GPU как набор одновременно выполняемых потоков.

Программные потоки являются отдельными непосредственными исполнителями вычислений. Блок задач, выполняемых в потоках, выполняется на мультипроцессоре частями. Абстрактное объединение нескольких потоков называются warp-ом. Количество потоков, объединённых в одном warp-е, на данный момент. равно 32. Во всех потоках в рамках одного warp-а выполняется одна команда, что напоминает функционирование данного потокового пула по принципу SIMD. В корпорации NVIDIA данный подход получил название SIMT (single instruction, multiple thread – одиночный поток команд, множество потоков).

Наряду с потоковыми пулами потоки объединяются в блоки. Потоки в блоке организованы как трехмерная структура. Синхронизация вычислений между разными блоками невозможна, в то время как взаимодействие потоков в рамках одного блока используется стандартно. Максимальное количество потоков в блоке обусловлено типом данных, над которыми производятся вычисления в рамках блока. Топология размещения потоков внутри одного блока зависит от разработчика: при необходимости потоки в рамках блока можно организовать как вектор, двумерный или трехмерный массив. Программисту предоставляется возможность организации блоков в структуры более высокого уровня абстракции - двумерные сетки (grid).

Специальная иерархия запоминающих устройств в CUDA также является отличительной особенностью платформы. Выделяется регистровая память внутренних процессоров, локальная память (размещена в DRAM), разделяемая память, константная и текстурная памяти, глобальная память, доступная для всех мультипроцессоров GPU. Глобальная, локальная, константная и текстурная памяти физически выполнены в виде единой памяти видеоадаптера

DRAM. CPU имеет доступ только к глобальному, текстурному и константному устройствам памяти.

Реализация метода генерации дуг для объёмных 3D дисплеев на базе архитектуры CUDA

Модифицированный алгоритм последовательного решения задачи генерации дуг, заданных произвольными начальными данными, в пространстве 3D устройства отображения информации представлен в [7, 8]. Сравнение относительных характеристик модифицированного алгоритма по сравнению с базовым алгоритмом из [10] приведено в [8].

Общая схема работы CUDA-реализации программного генератора дуг представлена на рис.1.

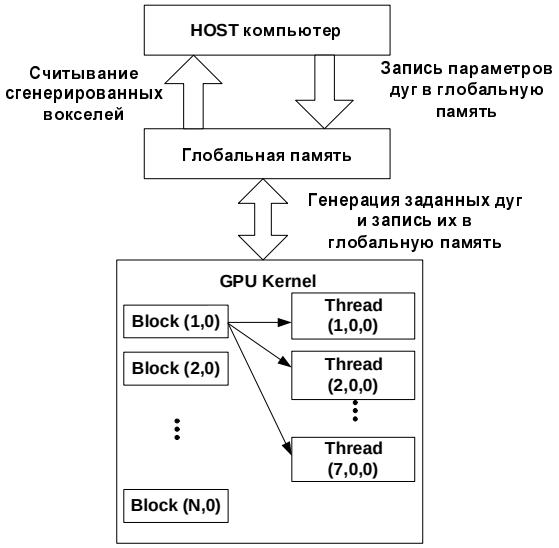

Рисунок 1 – Схема работы CUDA-программы

Входными данными является набор значений, однозначно задающих множество дуг в пространстве. Каждая дуга из данного множества задается следующими параметрами: начальная точка генерации $\boldsymbol{N}$, центральная точка дуги $\boldsymbol{C}$, угол генерации дуги $\boldsymbol{\alpha}$ и некоторая точка, однозначно задающая плоскость генерации дуги $\boldsymbol{G}$. Выходными данными является множество сгенерированных векселей.

Общая схема работы параллельного алгоритма генерации заданного количества дуг с использованием CUDA приведена на рис. 2. При этом реализуется следующая последовательность действий:

1. Инициализация (CPU). Исходные данные, необходимые для инициализации процесса генерации, принимаются из некоторого файла.
2. Начальные вычисления (CPU). Производятся вычисления, связанные с нахождением координат конечной точки генерации для каждой дуги с помощью матрицы поворота, переводом угла генерации дуги в радианную меру, вычислением коэффициентов плоскости генерации, проверкой начальных данных на корректность, переходом от непрерывных координат к дискретным.
3. Выделение памяти на GPU. Определяется топология сетки, количество блоков и потоков в рамках каждого из блоков и в пространстве памяти GPU выделяется соответствующий объем памяти.
4. Копирование данных (из CPU в GPU). Является связующим звеном между host-кодом и device-вычислениями программы.
5. Запуска ядра. При запуске ядра все потоки выполняются асинхронно, поэтому после параллельного выполнения вычислений необходимо выполнить операцию синхронизации. В соответствии с полученной последовательностью входных значений параллельно генерируются все дуги по модифицированному алгоритму генерации дуги в пространстве трехмерного устройства отображения информации из [8].
6. Копирование результатов вычислений (из GPU в CPU).

Разработанная программа использует 1024 блока, в каждом из которых активными являются семь потоков. Количество блоков соответствует количеству генерируемых дуг, заданных произвольными начальными данными. В каждом из потоков происходит вычисление координат одного векселя-претендента на каждой итерации цикла вычисления следующего векселя, входящего в воксельное разложение данного графического примитива. Ядро запускает сразу все используемые блоки, позволяя максимально использовать ресурсы платформы CUDA.

Экспериментальные исследования CUDA реализации модифицированного алгоритма растрового разложения дуги

Одной из важных характеристик методов воксельного разложения графических примитивов с точки зрения построения систем генерации образов в реальном времени является время, затрачиваемое на выполнение разложения. Суммарное время генерации 3D сцены для объемного 3D дисплея во многом определяется как количеством генерируемых графических примитивов, так и их сложностью. Время воксельного разложения дуги прямо зависит от количества векселей, которые входят в воксельное представление данного примитива. Измерение времени генерации одной дуги не является в достаточной степени информативным, т.к. для получения вывода об эффективности

генерации исследуемого алгоритма данных, полученных в результате генерации одного примитива, недостаточно. В ходе экспериментальных исследований данного алгоритма была применена методика, обеспечивающая достаточный уровень информативности, позволяющий сделать вывод об эффективности использования алгоритма генерации графического примитива. Исследование возможностей платформы CUDA выполнялось в виде двух серий экспериментов, при этом каждая серия включала в себя ряд этапов. Первая серия заключалась в генерации некоторой определенной дуги с некоторым заданным количеством повторений. Вторая серия заключалась в генерации определенного количества случайным образом заданных дуг.

Аналогично распространенному параметру, характеризующему производительность графических систем реального времени - количеству кадров, генерируемых системой в единицу времени (fps), для интегральной оценки использовалась производительность ***P***, вычисляемая как количество вокселей, генерируемых системой в единицу времени. Единица измерения - vps (один воксель в секунду).

При генерации Q произвольных дуг производительность "генератора" может быть вычислена как:

$$P = \frac{\sum_{i=1}^{Q} N_i}{\sum_{i=1}^{Q} T_i} \quad (vps),$$

где:

N_i, $i = 1,2,...,Q$ - количество вокселей в воксельном представлении i-го примитива,

T_i, $i = 1,2,...,Q$ - время воксельного разложения i-го примитива.

Численные эксперименты производились на базе компьютера со следующей конфигурацией: Intel (R) Core (TM) i3 CPU 530 2.93GHz, 3,00 Гбайт ОЗУ, видеокарта GeForce GTX 285.

Первая серия. На начальном этапе замерялось время поэтапной генерации шести множеств дуг без использования архитектуры CUDA по модифицированному алгоритму из [8]. При этом наборы дуг содержали 128, 512, 1024, 4096, 10240 и 16384 графических примитивов соответственно. Другими словами, один и тот же графический примитив генерировался различное количество раз. Временные параметры, полученные на этом этапе исследований, приведены в таблице 1.

Таблица 1. Время генерации на CPU множества идентичных дуг

Кол-во дуг	Кол-во вокселей	Время генерации, мс	Произв-ть, тыс. вокс./с
128	255616	188	1359,659
512	1022464	773	1322,721
1024	2044928	1553	1316,759
4096	8179712	6060	1349,787
10240	20449280	15172	1347,830
16384	32718848	24328	1344,904

На втором этапе экспериментов исследовалась работа метода генерации воксельного разложения такого же множества дуг, но уже с помощью архитектуры CUDA. Особенность данного этапа исследований заключалась в последовательном циклическом запуске всех дуг из каждого множества. При этом каждой дуге выделяется один блок, содержащий по 7 параллельно запускающихся потоков (по одному потоку для каждого вокселя-претендента). Временные параметры, полученные на втором этапе исследований, приведены в таблице 2. Анализ второго этапа исследований показал, что применение данной стратегии неэффективно, т.к. время генерации кратно превышает показатели, полученные без использования архитектуры CUDA.

Таблица 2. Время генерации на GPU множества идентичных дуг (циклическое последовательное выделение одного блока на каждую дугу)

Кол-во дуг	Кол-во вокселей	Время генерации, мс	Произв-ть, тыс. вокс./с
128	255616	1725	148,183
512	1022464	6900	148,183
1024	2044928	13777	148,430
4096	8179712	40852	200,227
10240	20449280	136679	149,615
16384	32718848	217469	150,452

На третьем этапе исследований количество блоков, выделяемых для разложения всех дуг из заданного множества, варьировалось (32, 64, 128, 256, 512, 1024, 2048, 4096, 10240, 16384). Причем, количество выделяемых блоков не превышало число генерируемых дуг. Временные параметры, полученные на третьем этапе исследований, приведены в таблице 3.

Рисунок 2 - Общая схема работы параллельного алгоритма генерации заданного количества дуг

Таблица 3. Время генерации на GPU множества идентичных дуг (выделение каждому графическому примитиву по одному блоку)

Кол-во дуг	Кол-во векселей	Время генерации, мс	Произв-ть, тыс. векс./с
128	255616	12,126	21079,993
512	1022464	40,228	25416,724
1024	2044928	70,058	29189,071
4096	8179712	260,86	31356,712
10240	20449280	630,992	32408,144
16384	32718848	1010,87	32367,018

Временные показатели, полученные на третьем этапе, свидетельствуют о том, что применение различного количества блоков для каждого из множеств генерируемых графических примитивов приводит к существенному повышению производительности. Таким образом, можно утверждать, что максимальная продуктивность работы генератора обеспечивается достижением как можно большего уровня распараллеливания работы алгоритма. В таблице 4 приведена сравнительная характеристика вышеперечисленных методов (в таблице не приведены характеристики метода генерации на GPU с последовательным выделением блока для обработки каждого графического примитива, т.к. показатели производительности данного метода худшие из всех вышеперечисленных).

Таблица 4. Сравнительная характеристика производительности генерации на CPU и GPU методов

Произв-ть (CPU), тыс. векс./с	Произв-ть (GPU), тыс. векс./с	Относительное ускорение
1359,659	21079,993	**15,5**
1322,721	25416,724	**19,2**
1316,759	29189,071	**22,2**
1349,787	31356,712	**23,2**
1347,830	32408,144	**24**
1344,904	32367,018	**24**

Результаты экспериментов показывают, что наиболее эффективным способом генерации оказалась модификация алгоритма с использованием архитектуры CUDA, выделяющая по одному блоку для генерации каждой дуги. Однако, в основу экспериментов была заложена упрощенная задача генерации различного количества одинаковых дуг, заданных идентичным множеством параметров.

Вторая серия экспериментов проводилась для приближенного к реальности варианта - генерации дуг с произвольными, случайно выбранными параметрами (W=16384 дуг). Если необходимо сгенерировать U дуг, то, если U<W из W берутся U первых наборов начальных данных, необходимых для генерации графического примитива.

В таблице 5 приведены сравнительные характеристики двух методов генерации графических примитивов: на host-компьютере с использованием CPU, и на GPU с использованием CUDA (выделение блока каждому генерируемому графическому примитиву).

Таблица 5. Сравнительный анализ результатов генерации множеств дуг, заданных произвольными начальными данными

Количество дуг	Количество вокселей	Время генерации, мс		Производительность, тыс.вокс./с		Ускорение GPU относительно CPU
		CPU	GPU	CPU	GPU	
128	203607	328	74	621	2766	**4,4**
512	870849	1297	187	671	4646	**6,9**
1024	1949224	2641	360	738	5417	**7,3**
4096	7100220	10172	1254	698	5662	**8,1**
10240	17551173	25547	2991	687	5869	**8,5**
16384	27933423	41672	4786	670	5837	**8,7**

Выводы

Анализ результатов экспериментов показывает, что использование программно-аппаратной платформы CUDA существенно повышает быстродействие и эффективность работы программ, связанных с массивно-параллельными вычислениями. Показатель относительного повышения производительности по сравнению с CPU больше, чем в 8 раз, свидетельствует о том, что технология CUDA может быть использована в качестве среды реализации сложных вычислений. Также применение данной технологии может быть применено в процессе генерации сложных графических трехмерных сцен с учетом реального времени.

Список литературы

1. Роджерс Д. Алгоритмические основы машинной графики / Д. Роджерс; пер. с англ. — М.: Мир, 1989. — 512 с.
2. Favalora G.E., Volumetric 3D Displays and Application Infrastructure //"Computer", 2005, August, P 37-44.
3. Favalora G.E. et al., "100 million-voxel volumetric display", in Proc. SPIE Cockpit-Displays IX: Displays for Defense Appl, 2002, vol. 4712.
4. Z. Fan, F. Qiu , A. Kaufman , Zippy: A Framework For Computation And Visualization On A GPU Clust Er , Computer Graphics Forum, 27, 2, 341-351, 2008
5. Extended specifications and test data sets for data level comparisons of direct volume rendering algorithms, Kwansik Kim, Craig M. Wittenbrink, and Alex Pang, *IEEE Transactions on Visualization and Computer Graphics* , pages 299-317, Vol. 7, No. 4, Oct.-Dec. 2001. Also available as public Technical Report HPL-2000-40R1 , Hewlett-Packard Laboratories, Palo Alto, CA, Mar. 2001.
6. К построению генератора графических примитивов для трехмерных дисплеев / Башков Е.А., Авксентьева О.А., Аль-Орайкат Анас М. // Науковіпраці Донецького національного технічного університету. Серія: Проблеми моделювання та автоматизації проектування динамічних систем. –2008. – Вип. 7 (150). – С. 203-214
7. Башков Е.А. Метод воксельного разложения произвольной дуги для объемных 3d дисплеев / Е.А. Башков, О.А. Авксентьева, Н.О. Харченко // Наукові праці Донецького національного технічного університету. Серія: Проблеми моделювання та автоматизації проектування. – 2011 – .Вип. 9 (179) – С. 18-25.
8. Харченко Н.О. Улучшенный алгоритм генерации дуги для объемных 3D дисплеев / Н.О. Харченко, О.А. Авксентьева // Информатика и компьютерные технологии-2010. - 2010. – 67с.
9. Боресков А.В. Основы работы с технологией CUDA / А.В. Боресков, А.А. Харламов. – М.: ДМК Пресс, 2010. – 232 с.; ил.
10. Мілютін М.О. Генерація окружності для 3-D дисплеїв / М.О. Мілютін, Аль-Орайкат Анас Махмуд, О.А. Авксентьева // Информатика и компьютерные технологии-2009. - 2009. – 147 с.

Надійшла до редколегії 15.10.2011

Є.О. БАШКОВ, О.О. АВКСЕНТЬЄВА, М.О. ХАРЧЕНКО
Донецький національний технічний університет

Можливості платформи CUDA під час реалізації воксельного представлення дуг для об'ємних 3D дисплеїв

Стаття присвячена розгляданню питань, пов'язаних з паралельною реалізацією модифікованого алгоритму генерації дуги, заданої довільними параметрами в тривимірному просторі об'ємного пристрою відображення інформації, за допомогою засобів, наданих програмно-апаратною архітектурою CUDA. Розглянуто різноманітні методи оптимізації модифікованого алгоритму генерації дуги з точки зору паралельних обчислень, проведено їх порівнювальний аналіз на основі отриманих відносних та абсолютних показників, що характеризують ефективність і точність алгоритму.

воксель, графічний примітив, об'ємний дисплей, програмно-апаратна архітектура CUDA, генератор графічного примітива

E.A. BASHKOV, O.A. AVKSENTIEVA, N.O. HARTCHENKO
Donetsk National Technical University

CUDA Platform Capabilities during the Implementation of Arcs Voxel Representation for 3D Displays.

The article is devoted to the aspects that are closely connected to parallel realization of modified random arc generation algorithm in three-dimensional space of volumetric displaying device using the means of computing CUDA architecture. Also the set of optimization methods is considered from the point of view of parallel computing. Their comparative analysis is based on the obtained set of absolute and relative characteristics.

voxel, graphics primitive, volumetric display, software-hardware CUDA architecture, graphics primitive generator