ОБЪЕМНАЯ ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ ГАЗОДИНАМИЧЕСКОГО МОДЕЛИРОВАНИЯ МЕТОДОМ ПРОЕЦИРУЕМЫХ ТЕТРАЭДРОВ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ CUDA

В.В. Титов

GDT Software Group, Тула

E-mail: titov@cfd.ru

В статье описывается применение метода проецируемых тетраэдров для интерактивной объёмной визуализации результатов газодинамического моделирования. В отличие от предыдущих работ, предложенный в статье метод опирается на использование технологии параллельных вычислений CUDA. Использование вычислительных возможностей графической карты для проецирования ячеек и сортировки позволяет достичь интерактивности при визуализации скалярных данных на структурированной сетке. На текущем графическом оборудовании представленный алгоритм позволяет отображать данные со скоростью 23 млн тетраэдров в секунду.

Введение

Современные методы анализа результатов численного моделирования сложных нестационарных газо/гидродинамических процессов, проблем механики деформируемого твёрдого тела не обходятся без визуального отображения информации. Визуальное представление анализируемых данных может иметь различный характер в зависимости от размерности расчётной области, визуализируемых параметров, требования наглядности представления информации. Наиболее общим и часто используемым графическим представлением информации в трёхмерном пространстве является объёмная визуализация скалярных данных. При этом скалярное поле отражает значение какого-либо конкретного параметра: плотность, давление, температура и т.д. - в каждой точке доступного объёма. Скалярное поле обычно представляется в виде цвета и прозрачности, значения которых определяются с использованием, так называемой, передаточной функции. В представляемой работе рассматривается визуализация скалярного поля, заданного на неструктурированной сетке, каждая ячейка которой представляет собой тетраэдр.

Существует много различных способов создания объёмной визуализации: проецирование ячеек, трассировка лучей, использование заметающей плоскости [1].

Подход, используемый в статье, базируется на методе проецируемых тетраэдров, описанном Shirley и Tuchman [2]. Также были применены методики классификации тетраэдров и расчёта цвета и прозрачности из [3].

1. Метод проецируемых тетраэдров

В основе метода лежит проецирование тетраэдров, составляющих расчётную сетку, в экранную область и их отображение в порядке видимости.

Каждый спроецированный тетраэдр преобразуется в набор треугольников. В процессе проецирования тетраэдр относится к одному из классов в зависимости от его положения в пространстве относительно наблюдателя. Классы разделяются по количеству получаемых треугольников. Среди вершин треугольников выделяют «толстые» и «тон-

кие» по длине луча проецирования, проходящего через первоначальный тетраэдр. «Толстой» является вершина, для которой длина луча максимальна. Для этой вершины определяются координаты точек входа и выхода луча из тетраэдра, длина луча (l) и значения скалярной величины (sf, sb). «Тонкие» вершины попадают в узлы ячейки. Для этих вершин длина луча равна нулю, а значения скалярной величины и координат известны.

Далее выполняется растеризация полученных треугольников и для каждого фрагмента (пиксела) треугольника выполняется интерполяция параметров sf, sb, l, преобразование скалярных величин в цвет с использованием передаточной функции и вычисление цвета и прозрачности фрагмента с использованием интеграла интенсивности излучения. Результирующие цвет и прозрачность помещаются в цветовой экранный буфер для получения конечного изображения с использованием выражения:

Inew = Color + (1 – Alpha) x I,

где I – значение цвета в буфере, Color, Alpha – рассчитанные для текущего фрагмента значения цвета и прозрачности.

2. Реализация метода с использованием CUDA

Разработанная реализация метода была разделена на несколько этапов. На первом этапе для каждого тетраэдра выполняется проецирование. На втором этапе выполняется сортировка тетраэдров по глубине. На третьем формируется массив индексов для OpenGL примитивов. На четвёртом – выполняется отрисовка примитивов с использованием шейдеров.

2.1. Проецирование тетраэдров

На этапе проецирования для каждого тетраэдра осуществляется вычисление параметров «толстой» вершины, глубины центра, класс и количество вершин в проекции. Реализация этапа выполнена с использованием технологии CUDA, при которой для решения задачи используется очень большое количество параллельно выполняемых нитей, при этом обычно каждой нити соответствует один элемент вычисляемых данных. В нашем случае, каждой нити соответствует один обрабатываемый тетраэдр. Набор инструкций, выполняемых нитью, называется ядром.

В нашей реализации этапа проецирования ядро производит следующие действия:

- загрузка координат узлов тетраэдра из глобальной памяти;
- проецирование координат узлов в экранную область и вычисление глубины центра тетраэдра;
- классификация тетраэдра и определение количества вершин в проекции;
- сортировка вершин проекции;
- вычисление параметров пересечения спроецированных граней;
- вычисление координат «толстой» вершины и длины луча в этой точке;
- вычисление скалярной величины для «толстой» вершины;
- формирование выходных данных.

После выполнения вычислений часть данных сохраняется в массивы в памяти GPU для дальнейшей обработки с использованием CUDA, а часть сохраняется в буферный объект для отрисовки.

2.2. Сортировка по глубине

Для правильного отображения полупрозрачных объектов трёхмерной сцены необходима их сортировка по глубине. Может использоваться как сортировка от ближних к дальним, так и обратная – от дальних к ближним. Выбор зависит от установленной схемы смешивания цветов. В нашем случае была выбрана сортировка от ближних к

дальним. Для сортировки была использована функция сортировки по ключу библиотеки Thrust, входящей в состав CUDA Toolkit. Библиотека Thrust позволяет задействовать мощность графической карты для выполнения базовых алгоритмов.

В качестве ключа для функции сортировки стал массив значений глубины центров тетраэдров, а значениями для сортировки стали порядковые номера тетраэдров.

При выполнении сортировки были исключены обмены CPU-GPU, так как все данные были полностью подготовлены на GPU.

2.3. Формирование индексов

Во время визуализации каждый тетраэдр преобразуется в набор треугольников. При этом треугольники удобно заменить на имеющийся в OpenGL примитив GL_TRIANGLE_FUN. Этот примитив задаётся массивом вершин v1, v2, …, vn. Из n вершин формируется n-2 треугольника следующим образом: первый задаётся вершинами v1, v2, v3; второй – v1, v3, v4; *i*-ый – v1, vi+1, vi+2. Для отображения примитивов используется функция glMultiDrawElements, позволяющая выводить последовательности индексных примитивов. Для этой функции необходимо подготовить:

- массив count с количествами выводимых элементов в каждой последовательности, в нашем случае – количество вершин для каждого примитива GL_TRIANGLE_FUN;
- массив pindices с указателями на места расположения индексов для каждой последовательности – у нас этот массив заполняется один раз при загрузке данных, а при каждом выводе изменяются сами индексы.

Подготовка этих массивов перед выводом примитивов была реализована с использованием CUDA. Вычислительное ядро для каждого выводимого примитива выполняет следующие действия:

- выбирается индекс тетраэдра из отсортированного массива;
- считывается информация о классификации тетраэдра;
- в массив count записывается количество вершин для отображаемого вместо тетраэдра примитива;
- в массив indices записываются индексы вершин в соответствии с классом проекции тетраэдра.

После выполнения CUDA-ядра массив count копируется в память CPU, а массив indices отображается в буферный объект в памяти GPU.

2.4. Визуализация примитивов

После того как сформированы массивы индексов можно произвести вывод примитивов. При выводе для хранения информации о вершинах используются буферные объекты. В одном буферном объекте хранится информация о координатах вершин, в другом о значениях скалярного поля и толщине тетраэдров в каждой вершине, в третьем хранятся индексы для ссылки на информацию из первых двух буферных объектов. Для преобразования значений скалярного поля в цвет используется возможность запрограммировать конвейер рисования с использованием языка программирования GLSL. В вершинном шейдере осуществляется обычная обработка по умолчанию, а фрагментный шейдер производит дополнительную обработку для вычисления цвета и прозрачности фрагментов. При этом для текущих значений скалярного поля на передней и задней гранях тетраэдра выполняется выборка цвета из одномерной текстуры палитры, затем производится конечный расчёт цвета и прозрачности в соответствии с предварительно частично рассчитанным интегралом интенсивности излучения.

3. Результаты

Описанный алгоритм был реализован в качестве расширения для пакета научной визуализации Scientific VR на языке C++ с использованием CUDA и GLSL. Для измерения производительности использовалась система с процессором Pentium IV 3.0 ГГц, 4 ГБ ОЗУ и графической картой NVidia GeForce GTX 570 1280МБ. В качестве тестовых наборов данных использовались результаты газодинамического и прочностного моделирования с различным количеством ячеек.

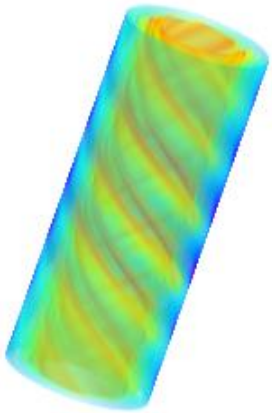

Рис. 1. Объёмная визуализация данных газодинамического моделирования (набор 1)

Рис. 2. Объёмная визуализация данных газодинамического моделирования (набор 2)

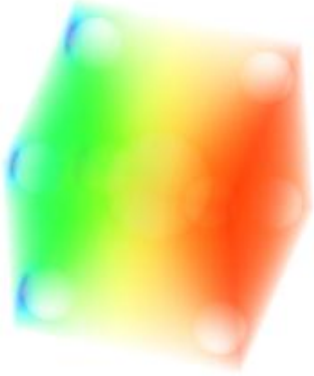

Рис. 3. Объёмная визуализация данных прочностного моделирования (набор 5)

В табл. 1 приведено среднее время исполнения каждого этапа и конечная производительность визуализации.

Таблица 1. Результаты визуализации скалярных данных

	Набор 1	Набор 2	Набор 3	Набор 4	Набор 5
Кол-во узлов, тыс.	768.9	1851	993.5	544.5	129
Кол-во ячеек, тыс.	4374	10764	5760	3198.5	533
1-ый этап, мс	32.35	78.15	40.89	25.81	4.98
2-ой этап, мс	10.16	22.24	12.95	8.74	4.87
3-ий этап, мс	23.94	60.71	30.44	18.10	2.02
4-ый этап, мс	123.13	295.5	158.83	89.73	15.62
Общее время, мс	189.58	456.6	243.11	142.37	27.49
Производительность, млн ячеек/сек	23.072	23.574	23.693	22.465	19.389

4. Заключение

Таким образом, производительность приведённой реализации позволяет формировать изображение объёмной визуализации скалярных данных для сеток с 23 млн ячеек за секунду. Кроме того, было проведено тестирование версии программы, в которой этап отрисовки примитивов выполнен с использованием технологии CUDA: вместо использования конвейера OpenGL и шейдеров на языке GLSL создано CUDA-ядро. Хотя производительность тестовой версии оказалась на порядок ниже - ее предпочтение позволяет реализовать весь алгоритм на CUDA. Такая возможность позволяет использование для визуализации кластера, оснащённого несколькими узлами. Что в свою очередь даст возможность визуализировать сетки со значительно большим количеством ячеек и с большей скоростью.

Литература

1. Johnson C., Hansen C. Visualization Handbook. Academic Press, Inc., Orlando, FL, USA, 2004.
2. Shirley P., Tuchman A. A.: Polygonal approximation to direct scalar volume rendering. In Proceedings San Diego Workshop on Volume Visualization, Computer Graphics. 1990. V. 24, P. 63–70.
3. Marroquim R., Maximo A., Farias R., Esperanca C. GPU-based cell projection for interactive volume rendering. SIBGRAPI: Brazilian Symposium on Computer Graphics and Image Processing. 2006. P. 147–154.