Система объемной визуализации реального времени на базе стандартного графического акселератора

М.Ю.Шевцов, Долговесов Б.С.

Институт Автоматики и Электрометрии

Новосибирский Государственный Университет
neomax@sx-lab311.iae.nsk.su

bsd@iae.nsk.su

Аннотация

В предложенной работе рассмотрены методы, которыми может быть решена задача объемной визуализации, с использованием возможностей стандартных графических акселераторов, поддерживающих DirectX9.0. Основа предлагаемого решения состоит в применении «3D-текстуры» (volume texture), расширения, предложенным NVIDIA, а следом и ATI. Основное отличие трехмерной текстуры от двумерной состоит в том, что она позволяет быстро, за один вызов, адресоваться к любому своему элементу по трем координатам. В работе также приведен краткий обзор альтернативных реализаций задачи объемной визуализации и сравнение предлагаемого решения с системой на базе высокоспециализированного аппаратного акселератора.

Возможности DirectX позволяют непосредственно совмещать объемные текстуры с полигональными моделями, например, имплантантами - в медицинских приложениях. Таким образом, предлагаемое решение также позволяет использовать обе технологии - объемной визуализации и полигональной графики - вместе.

Keywords: объемная визуализация, *volume texture, DirectX9.0, пиксельные шэйдеры*.

1. ВВЕДЕНИЕ

Одной из актуальных прикладных задач для любой области науки, работающей с объемными (трехмерными) расчетными или экспериментальных данными является их визуализация с помощью компьютера. Методы получения данных могут быть самыми различными: от моделирования до сканирования с помощью томографии, ультразвука и всевозможных методов, применяемых в сейсмографии и геологоразведке. Общим является то, что объемные данные - это трехмерный массив кубических элементов (вокселей), представляющих единицы 3D пространства. При этом, как правило, этот массив содержит информацию о каждой точке трехмерного пространства. Объемная визуализация – процесс получения изображения объемных данных на компьютерном экране, для представления объекта или явления реалистичным образом, с наглядной передачей его внутренней структуры [1],[2].

2. АЛЬТЕРНАТИВНЫЕ ПОДХОДЫ К РЕШЕНИЮ ЗАДАЧИ ОБЪЕМНОЙ ВИЗУАЛИЗАЦИИ

До последнего времени областью применения «традиционных» графических акселераторов было представление полигональных данных, поэтому такая специфическая задача как объемная визуализация решалась другими способами:

- Решения, использующие вычислительные мощности специальных графических станций (типа ONYX фирмы Silicon Graphics) и специальных серверов (типа AquariusNET Server фирмы TeraRecon). Основными недостатками такого решения были (и есть) плохая переносимость программного обеспечения и высокая цена.

- Программные реализации алгоритмов объемной визуализации. Основным недостатком таких реализаций является то, что они или в принципе не предназначены для визуализации в реальном времени, обходясь представлением нескольких слоев или срезов данных, или не обеспечивают интерактивности.

- Специализированные акселераторы для персонального компьютера, аппаратно реализующие алгоритмы объемной визуализации, например VolumePro [4]. Несмотря на то, что акселераторы этого класса обеспечивают существенное ускорение при работе с 3D-данными, они также имеют существенный недостаток в виде высокой цены. Другим серьезным неудобством может оказаться жесткая реализация основных алгоритмов «в железе», таким образом, что даже небольшая модификация их становится невозможной. Для примера укажем на тот факт, что все акселераторы этого класса реализуют алгоритм Ray-Casting [1, 2] аппаратно, что исключает возможность его замены на, например, алгоритм Ray-Tracing. (Сравнение предлагаемого решения и мощнейшего на сегодняшний день специализированного акселератора VolumePro 1000 приводится далее в данной работе, см. также [3, 6]).

3. ОРГАНИЗАЦИЯ ДАННЫХ 3D-ТЕКСТУРЫ. БАЗОВЫЙ ПОДХОД

Самым общим вариантом организации данных 3D-текстуры является подход, при котором 3D-текстура содержит значения цвета и прозрачности (RGBA) в каждой точке. Эти значения получаются при применении к исходным данным специальной таблицы (CLUT- Color LookUp Table). В таблице содержатся соответствия различных значений данных определенным значениям цвета и прозрачности. Процесс применения CLUT-таблицы происходит при заполнении RGBA значений 3D-текстуры.

3D-текстуры с RGBA значениями в каждой точке поддерживаются начиная с DirectX 8.0 и могут работать на любом графическом акселераторе с поддержкой этой версии DirectX API ([7]) - начиная с NVIDIA GeForce3Ti500. OpenGL также поддерживает работу с 3D-текстурами [8].

Единственным ограничением в таком случае является объем текстурной памяти акселератора. Оговоримся сразу, что существующие и в DirectX 9.0 ограничения на максимальный размер 3D-текстуры по любой оси (на данный момент это 512 элементов), а также то, что этот размер должен быть равен степени двойки, обходятся чисто программным решением: разбиением всех объемов на подобъемы с размерностями степеней двойки. Сортировка по расстоянию до наблюдателя при обходе дерева подобъемов гарантирует правильность отрисовки (см. также главу 5).

Заметным минусом 3D-текстур, организованных таким способом, является необходимость полной перезагрузки данных в текстурную память акселератора при любом изменении таблицы перевода значений данных в цвета и прозрачности.

4. ИСПОЛЬЗОВАНИЕ CLUT-ТЕКСТУРЫ И ШЭЙДЕРОВ

В предлагаемом подходе 3D-текстура содержит только собственно значения данных в качестве своих значений в каждой точке. Как и в подходе, описанном в предыдущей главе, эти значения переводятся в значения цвета и прозрачности через CLUT-таблицу. Но отличием является то, что это происходит не сразу, при загрузке данных, а только на финальной стадии конвейера обработки фрагментов. Таким образом, таблица хранится в текстурной памяти акселератора независимо от данных.

Во-первых, этим самым в четыре раза снижается объем памяти необходимый для 3D-текстуры, при той же точности представления данных (например, данные 8-битной точности, могут храниться в виде 8-битных значений, вместо четырех значений RGBA, 8 бит каждое). Во-вторых, для изменения представления объемных данных по цветам и прозрачности не нужно перегружать сами данные – достаточно изменить только CLUT-таблицу. Перезагрузка CLUT-таблицы требует на порядок меньше времени, чем время кадра и поэтому, в предлагаемом подходе, динамическое изменение CLUT-таблицы возможно в реальном времени.

Другим важным преимуществом является то, что, в случае необходимости, производится интерполяция самих значений данных, а только потом происходит их окрашивание в соответствии с таблицей перевода значения данных в цвета и прозрачности (CLUT). В случае же реализации 3D-текстур с RGBA значениями, интерполируются не сами данные, а соответствующие им значения цвета и прозрачности, что существенно снижает качество визуализации тонких полупрозрачных слоев (см. Рисунок 1 и Рисунок 2).

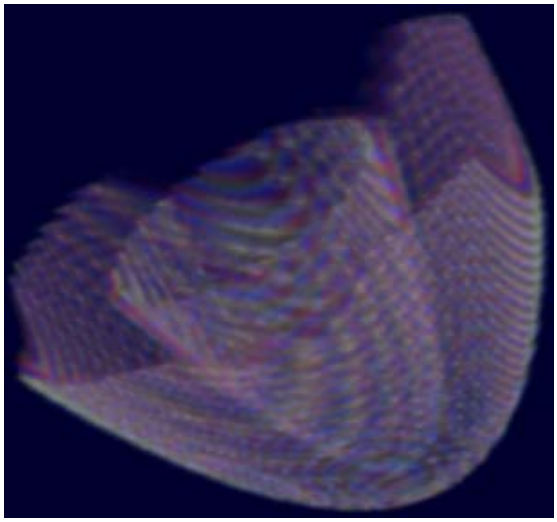

Рисунок 1: Визуализация 8-битного объема (64x64x64) как 3D-текстуры без использования второй (CLUT) текстуры, как массива значений RGBA.

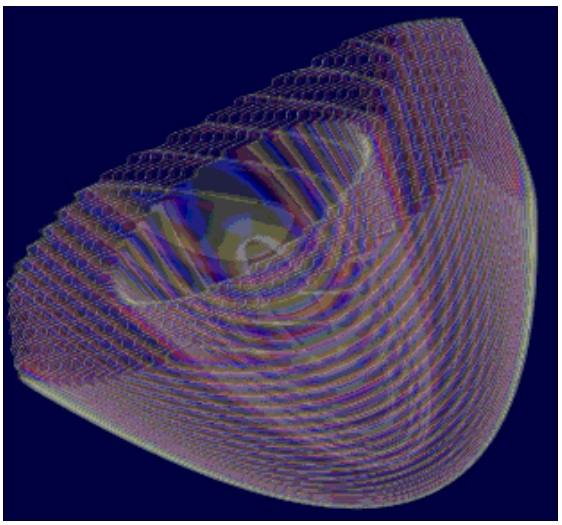

Рисунок 2: Визуализация 8-битного объема 64x64x64 как 3D-текстуры с использованием текстуры, содержащей таблицу перевода значения данных в цвета и прозрачности (CLUT)

Чтобы иметь возможность создать CLUT произвольной (не только 8-битной, что явно недостаточно для данных высокого разрешения) точности было решено хранить её в виде текстуры. Размер такой текстуры-фактически точность CLUT ограничена, также как и размер любой другой текстуры – 2^{27} значений, но это заведомо достаточно для любых приложений. На практике чаще всего используются 12-ти и 16-ти битные CLUT (на 4096 и 65536 значений).

Таким образом, алгоритм использования объемной текстуры со второй CLUT-текстурой может быть выражен кратко так: значения из первой текстуры используются для получения значений цвета и прозрачности из второй (см. Рисунок 3).

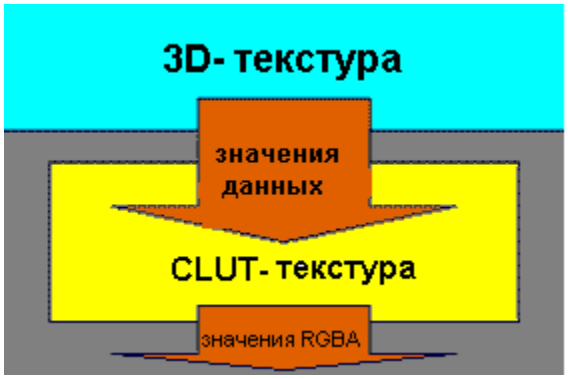

Рисунок 3: Использование таблицы перевода значения данных в цвета и прозрачности (CLUT), реализованной как текстура

В обобщенном случае такой подход называют «зависимым чтением» (dependent reading). И, несмотря на все преимущества зависимого чтения для нашей задачи, на сегодняшний день оно может быть реализовано только с помощью пиксельных шэйдеров.

Шэйдеры – небольшие программы, исполняющиеся непосредственно графическим акселератором. Шэйдеры позволяют, при возможности гибкой конфигурации работы ускорителя, не снижать общий уровень производительности при использовании алгоритмов отличных от стандартных. Иными словами, с помощью шэйдеров стало возможно динамически программировать конвейер графического акселератора, с возможностью включения/выключения и настройки отдельных его узлов.

Итак, шэйдеры могут выполнять работу по отрисовке 3D-текстуры с заданными таблицами перевода значений данных в цвета и прозрачности (CLUT). Другой важной способностью шэйдеров является гибкое управление расчетом освещения, причем для смены модели освещения (например, для ее упрощения на менее мощном акселераторе) достаточно загрузить другой шэйдер. Оговоримся лишь о том, что для расчета освещения необходима информация о нормали в каждой точке (или градиенте). Эта информация также может храниться в виде еще одной объемной текстуры, причем, как показывает опыт, для обеспечения правильной визуализации требуется на порядок меньше информации о значениях нормалей (градиентов данных), чем собственно данных, т.к. значения градиента эффективно интерполируются акселератором аналогично значениям самих данных (см. выше).

5. РАБОТА С БОЛЬШИМИ ОБЪЕМАМИ ДАННЫХ

Как уже упоминалось, существенной оптимизацией является разбиение начального объема на подобъемы. Сортировка по видимости при обходе дерева этих подобъемов позволит снизить количество вычислений и гарантировать правильность отрисовки. Результаты этой же сортировки помогут оптимизировать использование такого критического ресурса как текстурная память: только необходимые данные загружаются в память акселератора, в то время как ненужные временно выгружаются. Это позволит избежать многочисленных проблем, связанных с нехваткой ресурсов акселератора. Для разработки эффективного алгоритма разбиения данных, а также решения вопросов с их подгрузкой, могут применяться решения, хорошо отработанные в визуализации террэйна и функционально-заданных объектов, например, построение восьмеричного дерева [5].

6. РАБОТА С ДИНАМИЧЕСКИ ИЗМЕНЯЮЩИМИСЯ ДАННЫМИ

Эффективность решения поставленных задач во многом зависит от того, на сколько полноценно используются имеющиеся вычислительные ресурсы. В таком случае целесообразно применить общие программные решения по повышению эффективности работы. Планируемая асинхронность архитектуры основных модулей системы позволит загружать динамически изменяющиеся данные: в то время как первый поток загружает новые данные в акселератор, второй дожидается завершения визуализации уже загруженных данных, третий поток отрисовывает на экране результаты работы акселератора и т.д. Система, сконструированная таким образом, способна, за счет эффективного распределения задач, снизить время простоя всех конвейеров графического акселератора.

Пропускная способность шины AGP (даже для AGP4X, а не AGP8X), позволяет, с учетом накладных расходов оценить поток обновления данных как объем порядка 128*128*128 за кадр, при 50 FPS. Отметим еще, что это является величиной порядка 10% от пропускной способности, заявленной фирмой изготовителем для новейшего графического акселератора GeForce 6800 и PCI Express - быстрейшей на сегодня шины. Необходимые измерения в тестах для всех типов данных еще предстоит проделать.

7. ПРИМЕНЕНИЕ СТАНДАРТНЫХ ГРАФИЧЕСКИХ АКСЕЛЕРАТОРОВ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ОБЪЕМНОЙ ВИЗУАЛИЗАЦИИ.

7.1 Сравнение производительности VolumePro 1000 с предлагаемой системой

Итак, как уже говорилось в обзоре возможных подходов к задаче объемной визуализации реального времени, альтернативой реализации системы на базе стандартного графического акселератора является использование высокоспециализированного акселератора. В нашей лаборатории создан программно-аппаратный комплекс объемной визуализации на базе мощнейшего на сегодняшний день специализированного акселератора VolumePro 1000 [4][6]. Развитие же стандартных графических акселераторов происходило совершенно в другом направлении, приводя к появлению специальных решений (например, кэш вершин, сжатие 2D-текстур и z-буфера), практически бесполезных для решения задачи объемной визуализации. Серьезный сдвиг в таком положении вещей наметился относительно недавно, с появлением поддержки 3D-текстур и шэйдеров. Поэтому далее, кроме графика времени кадра VolumePro 1000, приводятся графики работы предлагаемой системы с разными объемами данных на базе стандартных графических акселераторов различных поколений (с использованием индексирования, а значит и шэйдеров, и без):

- GeForce 3 Ti– первый ускоритель (на базе чипа NV20), поддерживающий DirectX8.1 и старше;
- GeForce 5200 – первый акселератор, поддерживающий DirectX9 и зависимое чтение (dependent reading) – пиксельные шэйдеры версии 2_0;
- GeForce 5900;
- GeForce 6800 – мощнейший на сегодня стандартный графический акселератор.

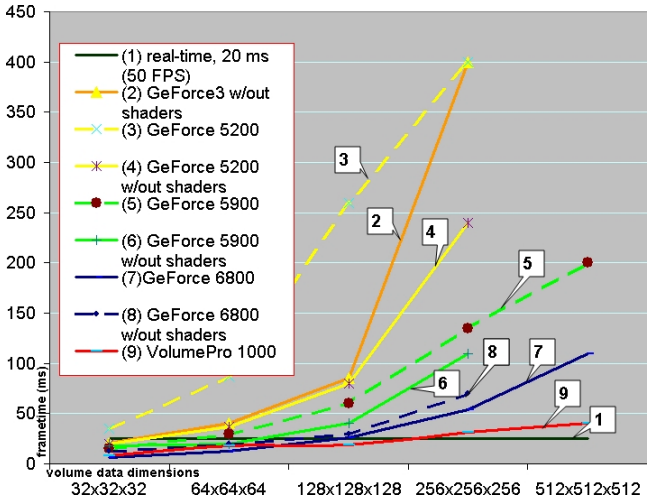

Рисунок 4: График времени кадра (в миллисекундах, по оси Y) для VolumePro 1000, в сравнение со временами предлагаемой системы для разных объемов данных (ось X). Жирной черной чертой обозначена граница «реального времени» - 20 мс (50 FPS).

7.2 Сравнение функциональных характеристик предлагаемой системы с системой на базе VolumePro 1000

Неоспоримым преимуществом VolumePro 1000 является достаточно слабый, линейный, рост времени генерации кадра с ростом объема данных (см. Рисунок 4). Для стандартных графических акселераторов всех поколений этот рост происходит квадратично (см. Рисунок 4). Немаловажным является то, VolumePro 1000 позволяет одновременно работать с объемом до 8192^3, а стандартные акселераторы не более 512^3, хотя это решаемая проблема (см. главу 5).

Не менее важным преимуществом является аппаратная реализация VolumePro 1000 вычисления градиентов и освещения (модель Фонга). В случае реализации освещения на стандартных акселераторах, необходимая информация о нормали в каждой точке (градиенте) должна храниться в виде еще одной объемной текстуры. Для экономии текстурной памяти, здесь также применимо индексирование, и, как показывает опыт, для обеспечения правильной визуализации требуется на порядок меньше информации о значениях нормалей (градиентов данных), чем собственно данных, т.к. значения градиента эффективно интерполируются акселератором, аналогично значениям самих данных. Тем не менее, расчет освещения на стандартных акселераторах потребует дополнительного зависимого чтения (dependent reading), что, в общем случае, заметно ухудшит показатели производительности системы. В то же время остается возможность гибкого управления расчетом освещения - с помощью шэйдеров, причем для смены модели освещения (например, для ее упрощения на менее мощном стандартном акселераторе) достаточно загрузить другой шэйдер. С помощью шэйдеров и stencil-буфера также могут быть реализованы такие специфические режимы отрисовки как Maximum, Minimum или Average Intensity Projection.

Обе системы позволяют изменение таблиц перевода значения данных в цвета и прозрачности (CLUT) в реальном времени, при этом разрядность CLUT стандартных акселераторов может быть произвольной, в отличие от фиксированных 8 или 16 бит для VolumePro 1000 [4].

И по некоторым показателям стандартные акселераторы превосходят VolumePro 1000:

- VolumePro 1000 предоставляет фиксированный набор арифметических и логических операций, над не более чем 4 объемами. Стандартные акселераторы, с помощью шэйдеров, поддерживают любые операции над вплоть до 32 объемами (GeForce 6800), за один проход, правда, за счет дополнительного падения производительности;
- стандартные акселераторы, в отличие от VolumePro 1000, поддерживают отрисовку с перспективой.

8. ЗАКЛЮЧЕНИЕ. ПЕРСПЕКТИВЫ

Итак, задача объемной визуализации уже сегодня может быть решена с использованием возможностей стандартных графических акселераторов. Возможность программирования конвейера акселераторов позволяет реализовать специфическую функциональность.

При этом обе технологии - объемной визуализации и полигональной графики могут применяться вместе. Новые возможности связаны с интегрированием пакета объемной визуализации в систему создания виртуальной реальности. Это позволит повысить эффективность процесса обучения за счет возможности пространственного представления, при котором 3D-данные могут демонстрироваться лектором непосредственно [9]. Взаимодействие лектора с моделируемыми объектами и явлениями способствует их углубленному пониманию за счет эффекта погружения в виртуальную среду. Поэтому, перспективным направлением является использование предлагаемой технологии в создании интерактивных обучающих курсов.

9. ЛИТЕРАТУРА

[1] Р. Ягель. *Рендеринг объемов в реальном времени* //Открытые системы, #05, 1996.

[2] Drebin,R.A., Carpenter,L., and Hanrahan,P., *Volume Rendering* //Computer Graphics, 22(4):65-74, August 1988.

[3] Dolgovesov B.S., Vyatkin S.I., Shevtsov M.Y. *Firmware complex for real-time volume rendering based on VolumePro 1000 accelerator* //GraphiCon'2003, pages 184-187.

[4] Документация по графическому акселератору VolumePro 1000 (доступна на http://www.terarecon.com/products/volumepro_prod.html)

[5] Sergei I. Vyatkin, Boris S. Dolgovesov, Valerie V. Ovechkin, Sergei E. Chizhik, Nail R. Kaipov. *Photorealistic*

imaging of digital terrains, freeforms and thematic textures in realtime visualization system Voxel Volumes //GraphiCon '97, Moscow.

[6] Шевцов М.Ю. *Объемная визуализация научных данных на базе VolumePro 1000* //Труды конференции-конкурса «Технологии Microsoft в информатике и программировании». 2004г. стр. 68-70. Диплом первой степени.

[7] MSDN resources for volume textures. http://search.microsoft.com/search/results.aspx?qu=volume+texture&View=msdn

[8] Volume Visualization with Texture (OpenGL). http://www.opengl.org/resources/tutorials/advanced/advanced98/notes/node224.html

[9] Dolgovesov B.S., Morozov B.B, Shevtsov M.Yu. *USING VIRTUAL STUDIO IN EDUCATION* //Труды 5-ой Международной конференции «Компьютерное моделирование 2004», 2004 год, Санкт-Петербург.

Авторы

Долговесов Борис Степанович– к. т. н., заведующий лабораторией синтезирующих систем визуализации Института Автоматики и Электрометрии СО РАН.

Адрес: Новосибирск, 630090, пр-т Ак. Коптюга, 1, ИАЭ.
Телефон: 8(3832)-33-36-30

E-mail: bsd@iae.nsk.su

Шевцов Максим Юрьевич – аспирант НГУ, инженер-программист лаборатории синтезирующих систем визуализации, Института Автоматики и Электрометрии СО РАН.

Адрес: Новосибирск, 630090, пр-т Ак. Коптюга, 1, ИАЭ.
Телефон: 8(3832)-33-36-30

E-mail: neomax@sx-lab311.iae.nsk.su

Real-time volume rendering system based on typical graphics accelerator

Abstract

Volume rendering is a powerful technique for visualizing three dimensional arrays of sampled data. The particular importance for the exploration and understanding of the volume data concerns to a desire to reveal the inner structure of volumetric objects. The volume visualization today is the one of the primary trends in the computer graphics.

This paper describes approach to volume visualization using pixel shaders and support of "volume texture" by modern graphics accelerators. The system is aimed to achieve projection rates of 20 frames per second for dynamically changing high-resolution datasets (up to 512^3). Detailed functionality description along with features depiction is provided.

Keywords: *Volume rendering, VolumePro1000, volume texture, pixel shaders, DirectX 9.0.*

About the author(s)

Boris S. Dolgovesov is PhD. For more then decade he is the head of the Laboratory of Synthesizing Visualization Systems that is a department of Institute of Automation and Electrometry. His activities include principal design of Virtual Reality systems and real-time visualization for training systems as well as system design for marine, space and flight simulators.

E-mail: bsd@iae.nsk.su

Maxim U. Shevtsov is post-graduate student of Novosibirsk State University. He is programmer and principal researcher of the Laboratory of Synthesizing Visualization Systems that is a department of Institute of Automation and Electrometry.

E-mail: neomax@sx-lab311.iae.nsk.su