УДК 004.92

Д.В. Гнатюк, З.Б. Холодная

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков

ИСПОЛЬЗОВАНИЕ ШЕЙДЕРОВ В 3D ПРОГРАММИРОВАНИИ ПРИ СОЗДАНИИ СОБСТВЕННЫХ АЛГОРИТМОВ ВИЗУАЛЬНЫХ ЭФФЕКТОВ

Проведен анализ алгоритмов визуальных эффектов, написанных с помощью шейдеров и отличных от алгоритмов, применяемых в графическом видеоадаптере, и влияния шейдеров на производительность системы. В качестве объекта исследования был выбран графический видеоадаптер, поддерживающий вершинные и вертексные (фрагментные) шейдеры.

Ключевые слова: *графический процессор, вершинные шейдеры, геометрические шейдеры, пиксельные (фрагментные) шейдеры.*

Постановка проблемы

На сегодняшний день в области 3D программирования лидирующие позиции занимают “next-gen” технологии, т.е. технологии, ориентированные на возможности графических адаптеров, которыми те станут обладать в ближайшие несколько лет. Основными отличиями в архитектуре графических адаптеров являются частотные характеристики видео-чипа (по англ. GPU – Graphics Processing Unit), видеопамяти, шины данных и количество отдельных шейдерных процессоров.

Для ряда визуальных эффектов используются алгоритмы, придуманные разработчиками и отличающиеся по конечному результату от алгоритмов, которые используются в видеоадаптере. Если расчет данных алгоритмов производить во время выполнения программы на центральном процессоре, то это сказывается на производительности всей системы в целом. Выходом из данной проблемы может быть реализация алгоритмов, отличных от алгоритмов, используемыех в видеоадаптере, на аппаратном уровне при помощи шейдерных программ.

Цель статьи – произвести анализ производительности компьютерной системы при расчете визуальных эффектов с использование программного метода расчета и с использованием шейдеров.

Развитие видеоадаптеров

У разработчиков 3D приложений для первых поколений графических адаптеров [1] было только два способа обработки графических примитивов: использовать собственные программные алгоритмы, обрабатывающие параметры вершин объектов, но тогда все расчеты ложатся на центральный процессор (по англ. CPU – Central Processing Unit) и как следствие – снижается производительность системы в целом, либо использовать алгоритмы обработки графических примитивов, которые зашиты в видеоадаптер изготовителем, но такие алгоритмы носят общий характер и не всегда подходят для программного продукта.

Постепенно производители графических адаптеров вводили «элементы программируемости». Например: аппаратная реализация нескольких алгоритмов освещения; для выбора конкретного алгоритма освещения необходимо установить соответствующие флаги в используемой графической библиотеке, однако реализация каждого алгоритма в аппаратном виде была невозможна, т. к. почти каждый случай уникален.

Следующим этапом в развитии графических процессоров стало появление программируемости графических процессоров специально написанными программами – шейдерами.

Первым появился пиксельный шейдер (в DirectX, шейдеры работающие с текселями текстур называются пиксельными шейдерами, а в OpenGL – фрагментными шейдерами), в графическом процессоре nVidia NV15 (от 1 до 4 пиксельных конвейера, в зависимости от модели видеоадаптера семейства nVidia GeForce2, и по два текстурных блока на каждом конвейере). Пиксельный (фрагментный) шейдер работает с фрагментами изображения, т.е. с текселем текстуры, которому поставлен в соответствие некоторый набор атрибутов, таких как цвет, глубина, текстурные координаты и используется на последней стадии графического конвейера для формирования фрагмента изображения. Появление данного шейдера поспособствовало появлению так называемого «бесплатного мультитекстурирования», т.е. когда при наложении двух текстур на один тексель не снижается общая пиковая величина fillrate (fillrate – процесс заполнения полигона текселями).

С появлением графических процессоров nVidia NV20 (семейство видеоадаптеров nVidia GeForce3), появились вершинные шейдеры. Вершинный шейдер оперирует данными, сопоставляемыми с вершинами выводимых многогранников. К таким данным,

в частности, относятся координаты вершины в пространстве, текстурные координаты и т.д. Вершинный шейдер может быть использован для видового и перспективного преобразования вершин, генерации текстурных координат, расчета освещения и т.д. Существовавшая на тот момент версия шейдеров была сильно ограничена; каждая шейдерная программа (особенно это относится к пиксельным) была сравнительно малой длины, т.е. могла содержать ограниченный набор команд, и писались на assembly shader language, который близок к ассемблеру для универсальных процессоров. Его низкий уровень доставляет определенные сложности для понимания кода и программирования, особенно, когда код программы большой.

Шейдерная версия 2.0 расширила возможности, предложив более длинные и сложные шейдерные программы и заметно расширила набор команд, была добавлена возможность расчетов с плавающей запятой в пиксельных шейдерах.

Также с выходом второй шейдерной версии появились и языки высокого уровня для программирования шейдеров: High-Level Shader Language, для графической библиотеки DirectX, начиная с 9 версии, производства Microsoft; Graphics Library Shader Language, для графической библиотеки, начиная с версии OpenGL v2.0 и Cg (C for Graphics), совместная кросс платформенная разработка nVidia и Microsoft [2 – 4]. Данные языки позволяют писать шейдерные программы на С-образном языке, который более лёгок в понимании, чем ассемблерный код, применяемый ранее.

В 2008 году, появились шейдеры версии 4.0, в которых был добавлен третий вид шейдеров – geometry shaders (геометрический шейдер) (рис. 1), а также существенно доработаны вершинные и фрагментные шейдеры. Теперь за обработку шейдеров отвечают не отдельные специализированные шейдерные процессоры, а так называемые унифицированные шейдерные процессоры. Унифицированные шейдерные процессоры могут исполнять как вершинные, так и пиксельные и геометрические шейдерные программы. Впервые унифицированная архитектура была применена в видеочипе игровой консоли Microsoft Xbox 360, этот графический процессор был разработан компанией ATI. А в графических процессорах для персональных компьютеров унифицированные шейдерные блоки появились в видео процессоре nVidia G80 (видеокарты семейства nVidia GeForce 8).

Геометрический шейдер, в отличие от вершинного, способен обработать не только одну вершину, но и целый примитив. Это может быть отрезок (две вершины) или треугольник (три вершины), а при наличии информации о смежных вершинах может быть обработано до шести вершин для треугольного примитива. Кроме того геометрический шейдер способен генерировать примитивы «на лету», не задействуя при этом центральный процессор.

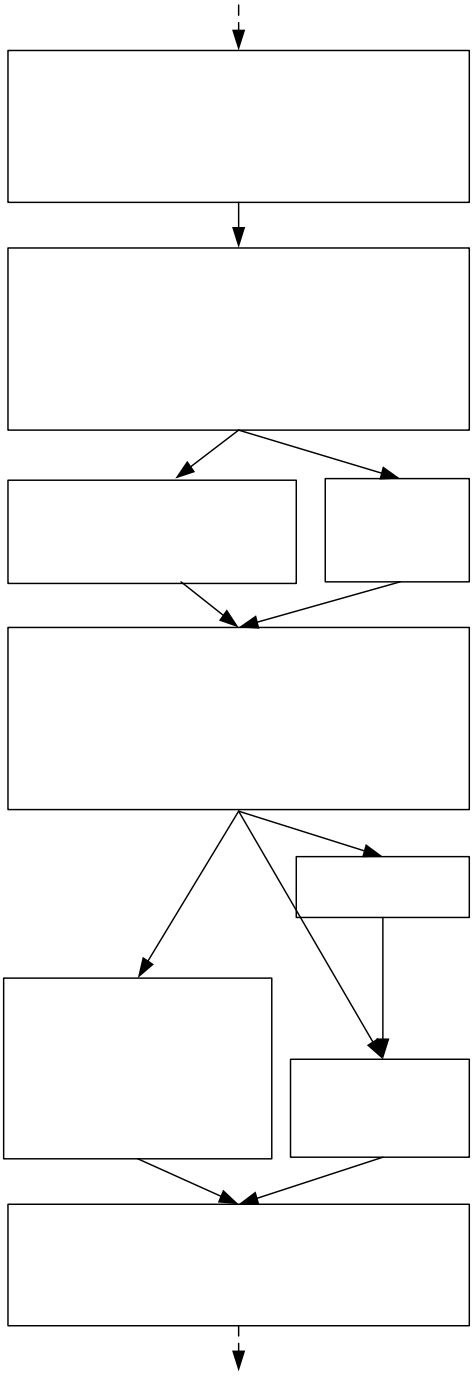

Рис. 1. Пример конвейера рендеринга, применяемого в видеоадаптере

На рис. 1 показан пример стандартного конвейера рендеринга, применяемого в современных графических адаптерах.

Анализ производительности системы, использующей шейдеры, для расчета визуальных эффектов

Программы, написанные специально для проведения оценки производительности системы, могут показывать разную производительность на компьютерах с различной конфигурацией.

Сравнение производительности проводилось на компьютере следующей конфигурации:

- OS: Windows XP Pro SP3;
- CPU: AMD Sepron 2800+ (2000MHz);
- GPU: nVidia GeForce 7600GS 256Mb;
- GPU driver version: 169.21;
- Memory: DDR-3200 1280Mb.

За единицу измерения была взята величина FPS (Frames Per Second — количество кадров в секунду на экране монитора, выдаваемых программным обеспечением видеокарты).

Тест 1. Bump mapping.

Bump mapping – метод в компьютерной графике позволяющий на плоской поверхности полигона моделировать микрорельеф для придания более реалистичного изображения и заключается в том, что отклонение каждого пикселя от нормали к поверхности просчитываемого объекта смотрится в карте высот и применяется перед вычислением коэффициента освещения (например, освещение по Фонгу).

Тест 2. Анимация объекта.

В данном тесте производилась анимация объекта, т.е. вершины объекта изменяли свои координаты в пространстве относительно прошедшего времени. В одном из тестов расчет новых координат производился при помощи центрального процессора, а в другом – координаты рассчитывались при помощи вершинного шейдера видеоадаптера.

Тест 3 Размытие изображения.

Данный тест относится к так называемым post-render эффектам, т.е. все сцена строится в отдельном буфере, после чего на данный буфер применяется фильтр размытия изображения и только после этого конечная сцена выводится на экран монитора.

Расчеты производились по следующей формуле:

$$C_{out,i,j} = \sum_{l,m} \alpha_{l,m} \cdot C_{i+l,j+m},$$

где параметром $C_{i,j}$ обозначены цвета пикселов исходного изображения; параметром $\alpha_{i,j}$ – так называемые коэффициенты свертки; параметром $C_{out,i,j}$ – цвета результирующего изображения; l, m – размерность апертуры, с которой работаем для получения значения цвета пиксела.

Таблица 1

Результаты тестов

Название теста	Без использования шейдеров	С использованием шейдеров
Bump mapping	~1930 fps	~2010 fps
Анимация	~1050 fps	~1300 fps
Размытие изображения	~1890 fps	~1920 fps

Вывод

Использование шейдерных программ в 3D программировании позволяет создавать собственные алгоритмы обработки графических примитивов для каждого отдельного приложения без потери производительности всей системы, а в некоторых случаях и увеличить ее, позволяет также снизить нагрузку на центральный процессор, при этом максимально загрузив процессор видеоадаптера.

Список литературы

1. Randima Fernando. The Cg tutorial / Randima Fernando, Mark J.Kilgard.. – Addison-Wesley, 2003. – 376 p.

2. Ренди Дж. Рост. OpenGL. Трехмерная графика и язык программирования шейдеров / Ренди Дж. Рост. – СПб.: Питерб, 2005. – 428 с.

3. Боресков А. Язык программирования GLSL / А. Боресков. – Петербург, 2007. – 426 с.

4. Миллер Т. DirectX 9 с управляемым кодом / Т. Миллер. – Москва, 2005. – 386 с.

5. OpenGL Shading Language Specification – [Электронный ресурс]. – Режим доступа к документу: http://www.opengl.org/registry/doc/GLSLangSpec.Full.1.30.08.pdf.

Поступила в редколлегию 18.12.2008

Рецензент: д-р техн. наук, проф. В.С. Харчнко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.

ВИКОРИСТАННЯ ШЕЙДЕРОВ В 3D ПРОГРАМУВАННІ ПРИ СТВОРЕННІ ВЛАСНИХ АЛГОРИТМІВ ВІЗУАЛЬНИХ ЕФЕКТІВ

Д.В. Гнатюк, З.Б. Холодна

Проведено аналіз алгоритмів візуальних ефектів, написаних за допомогою шейдерів і відмінних від алгоритмів, вживаних в графічному відеоадаптері, і впливу шейдерів на продуктивність системи. Як об'єкт дослідження був вибраний графічний відеоадаптер, що підтримує вершинні і вертексні (фрагментні) шейдери.

Ключові слова: *графічний процесор, вершинні шейдери, геометричні шейдери, піксельні (фрагментні) шейдери.*

USE OF SHADERS IN A 3D PROGRAMMING AT CREATION OF OWN ALGORITHMS OF VISUAL EFFECTS

D.V. Gnatyuk, Z.B. Holodnaya

The analysis of algorithms of visual effects, written by shaders and different from algorithms, applied in a graphic video adapter, and influencing of shaders on the productivity of the system is conducted. As an object of research a graphic video adapter, supporting tops and vertex (fragmenting)shaders, was chosen.

Keywords: *graphic processor, shaders of tops, geometrical shaders, shaders of pixels (fragmenting).*