УДК 004.925.4

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ОБРАБОТКИ ДИНАМИЧЕСКИ СЖИМАЕМЫХ ТЕКСТУР

И.В. Перминов

Описывается модель работы с текстурами, основанная на механизме представлений (Texture Views). Рассматриваются ограничения современных программных интерфейсов Direct3D 11.2 и OpenGL 4.4 при работе с текстурами, предлагается метод повышения эффективности обработки сжатых текстур путем исключения этапа копирования данных между буферами сжатого и несжатого типов. Приводятся результаты профилирования тестового приложения, осуществляющего динамическое сжатие текстур на каждом кадре.

Ключевые слова: сжатие текстур, Direct3D, OpenGL.

Текстуры повсеместно применяются в трехмерной компьютерной графике и в простейшем случае представляют собой двухмерное изображение, накладываемое на трехмерную поверхность с целью улучшения визуальной детализации без усложнения геометрии. При этом в современных интерактивных приложениях текстуры занимают более половины используемого объема видеопамяти [1]. Одним из эффективных вариантов повышения производительности и снижения требований к подсистеме памяти является использование предварительно сжатых текстур. Они хранятся в памяти, передаются по шинам в сжатом виде и распаковываются аппаратно внутри графического процессора. Это позволяет сэкономить один из наиболее важных ресурсов графического ускорителя – пропускную способность видеопамяти.

С ростом сложности систем визуализации все чаще начинают применяться динамически генерируемые визуальные данные и текстуры. Примером может служить карта окружения, которая обновляется на каждом кадре и используется для визуализации зеркальных поверхностей. Как правило, для таких текстур сжатие не используется ввиду сложности алгоритма сжатия и ограничений как самих графических процессоров, так и программных интерфейсов.

Для работы с текстурами современные программные интерфейсы Direct3D 11.2 [2] и OpenGL 4.4 [3] предлагают модель, позволяющую разделить описание буфера текстуры (Texture), хранящего данные, и представление (View), определяющее тип и доступ к этим данным. Это позволяет, к примеру, одни и те же данные интерпретировать как целочисленные в одной ситуации и как числа с плавающей запятой – в другой. При этом вводятся ограничения на возможные преобразования типов и операции с текстурными буферами. В частности, запрещается отрисовка в текстурный буфер сжатого формата, любое преобразование типов, имеющих разный размер текселя в битах, а также преобразование между любыми сжатыми и несжатыми типами.

Применительно к процессу динамического сжатия текстур это означает, что результат работы фрагментного шейдера не может быть записан в тот же самый буфер, из которого можно будет считывать текстурные данные с использованием аппаратного декодера. Таким образом, даже если шейдер может сформировать уже сжатый блок, он не сможет записать его в текстурный буфер, имеющий сжатый тип. По этой причине при сжатии текстур силами шейдера необходимо записывать сжатые блоки в буфер несжатого типа «под видом» текселей, а затем копировать полученные данные в буфер сжатого типа. При этом размер текселей в битах в промежуточном буфере должен совпадать с размером сжатого блока.

Для оценки затрат на копирование текстурного буфера были произведены тесты с использованием графического адаптера AMD Radeon 7970. В тестовой сцене на каждом кадре сжималась текстура размером 1024×1024 в формат DXT1 (соответствует типу DXGI_FORMAT_BC1_UNORM в Direct3D и типу COMPRESSED_RGB_S3TC_DXT1_EXT в OpenGL), и на экран выводились объекты с использованием сжатой текстуры. Профилирование показало, что формирование одного кадра занимает 170 мкс, из которых 14 мкс приходится на копирование текстурного буфера. При этом время отрисовки одного объекта тестовой сцены при использовании сжатых текстур падает с 45 мкс до 34 мкс.

Суть предлагаемого метода заключается в изменении процесса динамического сжатия текстур путем исключения копирования данных между буферами хранения сжатого и несжатого типов. Расширение функционала представлений текстур позволило бы использовать всего один буфер и полностью исключить этап копирования без увеличения затрат на другие операции при формировании кадра. Однако простого снятия ограничения на преобразование сжатого типа текстур к несжатому недостаточно, так как при записи буфер должен интерпретироваться как текстура с размером текселя, равным размеру одного сжатого блока, а при чтении – как текстура с более высоким разрешением и текселем, соответствующим одной точке сжатой текстуры. Однако существующий механизм текстурных представлений предполагает неизменность размеров текселя.

Возможным решением в данном случае является введение дополнительных функций, позволяющих создавать перекрывающиеся буферы. Альтернативным вариантом в случае OpenGL является модификация функции TextureView с учетом возможных изменений разрешения, создаваемого представления. Дополнительно подобные изменения позволили бы в шейдере напрямую производить частичные обновления сжатой текстуры на уровне блоков, что, к примеру, сделало бы возможным выполнение компоновки нескольких фрагментов в одну сжатую текстуру (compositing).

Предлагаемый метод позволит сократить время сжатия и повысить гибкость при работе со сжатыми текстурами. Таким образом, эффективность обработки сжатых текстур можно повысить, ликвидировав этап копирования данных между буферами сжатого и несжатого типа. Однако это потребует изменения программных интерфейсов Direct3D и OpenGL, а также драйверов графических адаптеров.

1. McDonald J. Eliminating Texture Waste: Borderless Ptex // NVIDIA Corporation, 2013 – 1 p. [Электронный ресурс]. – Режим доступа: https://developer.nvidia.com/sites/default/files/akamai/gamedev/docs/Borderless%20Ptex.pdf, свободный. Яз. англ. (дата обращения 08.08.2013).
2. Direct3D 11 Graphics [Электронный ресурс]. – Режим доступа: http://msdn.microsoft.com/en-us/library/windows/desktop/ff476080(v=vs.85).aspx, свободный. Яз. англ. (дата обращения 08.08.2013).
3. OpenGL 4.4 Core Specification [Электронный ресурс]. – Режим доступа: http://www.opengl.org/registry/doc/glspec44.core.pdf, свободный. Яз. англ. (дата обращения 08.08.2013).

Перминов Илья Валентинович **–** Россия, Санкт-Петербург, Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, аспирант, i.am.perminov@gmail.com