СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИНТЕРФЕЙСОВ СОЗДАНИЯ 2D И 3D ГРАФИКИ

Т. А. Рак, О. О. Шатилова
Кафедра вычислительных методов и программирования, Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: {tatianarak, shatilova}@bsuir.by

Рассмотрены особенности применения двух интерфейсов для создания трехмерной графики для компьютерной индустрии. Для анализа выбраны стандарты OpenGL и DirectX. Проведен сравнительный анализ возможностей двух интерфейсов.

Введение

В настоящее время трехмерные графические модели используются практически во всех сферах деятельности человека.

До начала 90-ых годов XX века не существовало никаких стандартов в области компьютерной графики из-за недоступности массовой реализации. До этого момента все программы писались практически с «нуля» или, максимум, использовались ранее полученные наработки для отображения графической информации. С развитием аппаратной части компьютеров, приходом мощных доступных процессоров и графических ускорителей 3D графика стала доступна для персональных компьютеров. Но это привело к тому, что отсутствовали какие-либо стандарты, позволяющие писать программные продукты, независящие от оборудования и операционной системы.

В 1992 году был опубликован открытый стандарт OpenGL, разработанный некоммерческой организацией Khronos Group [1]. На развитие этого стандарта повлияли практически все крупные производители графических процессоров (graphics processing unit, GPU) – nVidia, AMD, Intel. Этот стандарт доступен для большого числа платформ, а для Linux и Mac OS он является основным интерфейсом программирования приложений (API) для взаимодействия с GPU.

В 1995 году вышла частная разработка Microsoft – стандарт DirectX – созданная специально для Windows, недоступная на других операционных системах (исключая виртуализацию и эмуляцию API) [2]. Коллектив авторов решил собрать сведения, касающиеся технических возможностей обоих интерфейсов, и провести сравнительный анализ.

I. Сравнение OpenGL и DirectX

Оба этих стандарта предоставляют доступ к функциям аппаратного ускорения 3D графики.

На наш взгляд, OpenGL «берет» разработчиков простотой применения. Для пользователей разработан стандартизированный бинарный интерфейс приложений (ABI), а это значит, что OpenGL может быть использован из любого языка, который поддерживает вызов функций native библиотек (т.е., практически говоря, из любого вообще). У этого стандарта в основе заложено ядро, которое управляет обработкой примитивов (треугольников). Фактически, OpenGL использует обычные функция языка C и работа с примитивами ведется по средствам передачи данных соответствующим элементам библиотеки. С входных данных функциям можно сообщить, какие параметры должны работать в тот или иной момент времени (например, необходимо ли отрисовывать конкретную текстуру в текущем состоянии). Код, написанный с помощью библиотеки OpenGL, является достаточно простым для восприятия и, несмотря на процедурную реализацию ядра этого API, не имеет проблем с объектно-ориентированными технологиями.

Хотим отметить, что несмотря на то, что оба интерфейса имеют одинаковые конечные цели,DirectX в корне отличается от OpenGL и имеет в своей основе модель стандарта бинарного представления компонентов (COM). Модель COM предполагает не простые вызовы функций, а определенные действия, которые связаны непосредственно с архитектурой библиотеки DirectX. Вызовы DirectX используются внутри кода, который не является легким для понимания. И причем, для того, чтобы нарисовать обычный треугольник, используется очень большой объем кода. Для упрощения работы с таким кодом компания Microsoft разработала DirectX Common Files. Это отдельная библиотека, позволяющая, частично скрыть повторяющиеся куски кода [3].

В OpenGL реально доступное API определяется производителем GPU. Реализация OpenGL позволяет определять расширения к основной спецификации. Приложение может получить список поддерживаемых расширений во время выполнения, и проверить на доступность те, которые оно желает использовать. На самом деле практически весь функционал OpenGL – это расширения. Развитие OpenGL идёт так:

появляется новая необходимая функция, производитель реализует её в своём драйвере и документирует доступное расширение. Приложения могут использовать новые функции прямо сейчас, не дожидаясь включения в официальную спецификацию. Если это расширение специфично для конкретного производителя, то в названии оно несёт его имя (например, вот так: GL_NV_point_sprite, где NV-значит nVidia). Если расширение реализовано многими вендорами, то в названии используется EXT (например, GL_EXT_draw_range_elements). Со временем, если расширение широко используется, оно стандартизируется в ARB, и после этого содержит в имени ARB (например, GL_ARB_vertex_buffer_object). Такое расширение имеет официальный статус. Самые важные расширения со временем становятся частью основной спецификации. Каждая новая версия OpenGL – это по сути старая версия, дополненная несколькими новыми интегрированными расширениями. При этом новые функции доступны наравне с расширениями. Т.е. на самом деле с точки зрения программы может быть вообще всё равно, какая версия OpenGL, главное – какие доступны расширения. Версия OpenGL – это просто способ указать, какой набор расширений гарантированно поддерживается [4].

OpenGL и DirectX сильно отличаются своей архитектурой и принципами работы. Но в процессе развития этих двух стандартов стали заметны некоторые сближения. Это связано с эффективным использованием «железа» и уменьшением времени на преобразование пользовательских команд в аппаратные. Всегда в работе таких вещей как OpenGL и DirectX, одним из главных вопросов является вопрос производительности. Постоянно проходят споры о том, кто быстрее – OpenGL или DirectX. Хотя по итогу все сводится к тому, что скорость у этих библиотек одинакова. Так получается из-за того, что большинство функций и в DirectX и в OpenGL работают с помощью аппаратных ускорителей. OpenGL не поддерживает Pixel Shaders (программы, выполняемые видеочипом во время растеризации для каждого пикселя изображения, они производят выборку из текстур и/или математические операции над цветом и значением глубины (Z-buffer) пикселей), однако это не повод отказываться от работы с этой библиотекой (пример тому Unreal Tournament, игровой движок которого построен на основе OpenGL). Библиотека DirectX умеет эмулировать некоторые функции, не поддерживаемые аппаратно, однако на пиксельном уровне это неосуществимо и программа должна сама проверять возможности такой эмуляции.

II. Выводы

На основании сведений, полученных в ходе изучения технических характеристик двух интерфейсов, коллектив авторов считает, что обе библиотеки облегчают работу конечного программиста и обеспечивают интерфейс программы с конкретным железом. Они не совместимы между собой и имеют свои фирменные особенности, поэтому производителям приходится реализовывать в своих ядрах базовые функции обоих API, причем учитывать дополнительные функции новых версий и расширений.

Окончательно, мы можем сделать следующий вывод: обе библиотеки достаточно мощные и производительные, но их преимущества проявляются в разных областях. DirectX оптимально подходит для создания игр и других графических приложений под операционной системой Windows (что чаще всего и происходит), OpenGL же лучше всего проявляет себя на мощных рабочих станциях и там, где требуется совместимость приложений с различными платформами (как программная, так и аппаратная). Хотя и в сфере игр эта библиотека так же достаточно востребована (Doom III, Unreal Tournament 2004).

Список литературы

1. OpenGL [Электронный ресурс] / Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/OpenGL – Дата доступа: 4.10.2016.
2. DirectX [Электронный ресурс] / Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/DirectX – Дата доступа: 4.10.2016.
3. Адамс Д. DirectX: продвинутая анимация. / Д. Адамс // Издательство: КУДИЦ-ПРЕСС, 2004. – 480 с.
4. Компьютерная графика и мультимедиа. Сетевой журнал [Электронный ресурс] / Режим доступа: http://cgm.computergraphics.ru/content/view/55. – Дата доступа: 3.10.2016.