УДК 681.3

В.В. Карабчевский, С.Н.Магдалина
Донецкий национальный технический университет
karabch@pmi.dgtu.donetsk.ua

Визуальное создание изображений средствами Direct2D с использованием растрово-векторного подхода

Рассматриваются технологии обработки изображений, дается обзор технологии Direct2D, представлена разработка растрово-векторной системы обработки изображений, а также пример ее работы.
***Ключевые слова**: обработка изображений, Direct2D.*

Введение

В настоящее время наблюдается мощный скачок в развитии высокотехнологичных устройств самого разного назначения. При этом возникает проблема представления удобного и привлекательного интерфейса для управления функциями этих устройств. Реш этой проблемы требует более совершенных методов и технологий генерации и обработки двумерной графики. В связи с этим, разрабатываются новые аппаратные и программные решения, которые позволили бы сократить время работы с двумерной графикой до минимума.

В [1] была описана разработка визуального редактора изображений, созданного на базе технологии DirectX 9.0c.

В этом редакторе была предпринята попытка объединить достоинства растрового и векторного подходов к работе с изображениями. Однако, использование технологии Direct3D, которая является частью DirectX 9.0c, дало очень низкое быстродействие – все операции по изменению изображений выполнялись крайне медленно, что очень затрудняло работу с редактором.

Для преодоления этого недостатка на базе вновь вышедшей технологии Direct2D был разработан другой графический редактор, представленный в этой статье.

Постановка задачи растрово-векторного подхода при работе с изображениями

Традиционные графические редакторы достаточно жестко ориентированы на работу либо с растровой графикой (как Adobe Photoshop [2]), либо с векторной (Corel Draw [3]). При этом каждый из подходов имеет свои преимущества и недостатки. Векторный подход к созданию изображений позволяет описать структуру объектов математически, что делает полученную сцену легко масштабируемой и изменяемой. Работа с изображениями на уровне растра отличается высокой точностью коррекции отдельных частей изображения, возможностью работы с отдельными пикселями или группами пикселей.

Задачей растрово-векторного подхода при работе с графикой является объединение основных концепций и преимуществ работы при растровом и векторном подходе. Редактор, разработанный на основе растрово-векторного подхода, должен иметь возможность работать как с растром изображений, так и создавать отдельные векторные объекты, с возможностью дальнейшей их растеризации.

Причины появления Direct2D

До лета 2009 года основными технологиями вывода графической информации были стандартные средства вывода графики в той или иной ОС (GDI – для Windows), OpenGL и DirectX.

Стандартные средства вывода графики, такие, как GDI (Graphics Device Interface) для операционных систем семейства Windows, призваны обеспечить лишь базовые возможности для работы с графикой. Кроме того, такие средства обязаны быть достаточно универсальными, чтобы быть работоспособными на самых разных графических адаптерах. Такая универсальность обеспечивается за счет высокого уровня абстракции от аппаратного обеспечения, что значительно снижает быстродействие таких систем. Очевидно, что GDI подходит лишь для базовых операций с графикой и вывода примитивной 2D-графики, не говоря уж о трехмерных мирах.

OpenGL (Open Graphics Library) – это, по сути, спецификация, обеспечивающая стандартизацию подхода к работе с графикой на самых разных устройствах и платформах. С появлением спецификации стали появляться ее конкретные реализации для тех или иных платформ и систем. Целью появления OpenGL было создание единого подхода к работе с графикой для всех систем и языков программирования. Это стало основной причиной популярности OpenGL, но это же является и основным ее недостатком – такая языковая и платформенная универсальность предполагает множество излишеств и программной эмуляции недостающих компонентов, что приводит к

низкому быстродействию. Конкретные реализации OpenGL, конечно, написаны с использованием тех или иных потребностей разрабатываемых систем, однако все равно не дают того результата, который требуется в современных приложениях.

DirectX – это разработанная корпорацией Microsoft совокупность библиотек, позволяющих напрямую работать с аппаратным обеспечением ПК [4].

Ранее для вывода двумерной графики применялись библиотеки семейства DirectDraw. DirectDraw позволяла организовать прямой вывод графики на экран компьютера. Однако со временем эта технология была признана устаревшей и полностью заменена на Direct3D [5].

Однако, вывод двумерной графики средствами Direct3D также был не совсем удобен и трудоемок так как технология Direct3D ориентирована прежде всего на работу с трехмерной графикой, на формирование трехмерной виртуальной реальности [6].

Direct3D имеет возможность вывода полигонов непосредственно на экран, без применения лишних операций, таких как освещение, к примеру. Однако, это все равно затрудняет вывод сложных по форме объектов, а также динамическое их изменение во времени. Direct3D использует технологию текстурирования объектов, при которой на выводимый объект по определенным правилам накладывается хранящаяся в памяти текстура [7]. Благодаря использованию аппаратного ускорения графических процессоров эта операция выполняется очень быстро. Однако, если дело касается изменения хранящейся в памяти текстуры, то здесь Direct3D по быстродействию проигрывает даже стандартному GDI. Это происходит из-за того, что Direct3D прежде всего ориентирована на вывод трехмерной графики в реальном времени. Объекты Direct3D (и текстуры в том числе) хранятся в закрытых (locked) участках памяти либо видеокарты, либо оперативной. При этом чтение из таких закрытых участков происходит очень быстро, а вот для записи туда требуется немалое время. Поэтому у компьютерных игр периода 2001-2009 года так много времени занимает инициализация и загрузка данных в память, при том, что сами игры обеспечивают достаточно высокий показатель вывода кадров в секунду (fps)[8].

Как мы видим, среди основных технологий работы с графикой не было технологии, позволяющей в полной мере задействовать возможности видеокарт для вывода двумерной графики. Такое положение дел устраивало и разработчиков ПО, и производителей оборудования, так как потребности в качественной двумерной графике особенно не было.

Однако, в конце первого десятилетия XXI века произошел резкий скачок в технологической сфере информационных технологий. Долгие разработки новых устройств наконец дали свои плоды и на рынке IT-технологий стали появляться самые разные устройства – коммуникаторы, часы со встроенными компьютерами, GPS и ГЛОНАСС-навигаторы, информационные панели, банкоматы, интерактивные доски. Все эти устройства должны предоставлять яркий и понятный для работы с ними интерфейс, при этом потребляя минимум аппаратных мощностей для снижения стоимости. Было ясно, что существующие средства вывода графики не смогут обеспечить должной функциональности и красоты без значительного снижения быстродействия.

Обзор Direct2D

Для удовлетворения новых потребностей рынка IT-техноолгий корпорация Microsoft летом 2009 года разработала на базе технологии DirectX 10 набор библиотек для работы и вывода двумерной графики – Direct2D.

На момент написания статьи Direct2D включает в себя 4 заголовочных файла и одну библиотеку:

- d2d1.h – содержит объявления основных функций Direct2D API на языке C и C++;
- d2d1helper.h – содержит вспомогательный структуры, классы, функции;
- d2dbasetypes.h – определяет основные примитивы Direct2D, включен в d2d1.h;
- d2derr.h – определяет коды ошибок Direct2D. Включен в d2d1.h;
- d2d1.lib – двоичная библиотека, содержащая все объявленные в заголовочных файлах функции [9].

Как и DirectX, Direct2D построен на модели COM. Основным объектом, который предоставляет интерфейсы для создания других объектов, является Фабрика Direct2D**,** или объект класса ID2D1Factory**.** Он имеет в своем составе методы типа CreateResourse, которые позволяют создавать объекты более специфических типов.

Все объекты (ресурсы) в Direct2D делятся на два больших типа – устройство-зависимые (device-dependent) и устройство-независимые (device-independent). Устройство-зависимые объекты ассоциируются с конкретным устройством вывода и должны быть реинициализированы каждый раз, когда устройство, с которым они ассоциируются, требует реинициализации. Устройство-независимые ресурсы существуют без привязки к какому-либо устройству и уничтожаются в конце жизненного цикла программы или по желанию программиста. Классификация и основные примеры ресурсов приведены на рисунке 1.

Объекты класса ID2DRenderTarget – это устройство-зависимые объекты, которые ассоциируются с конкретным устройством вывода. Это может быть конкретное окно приложения, битовый образ (изображение) или другое

устройство. ID2DRenderTarget имеет в своем составе методы BeginDraw() и EndDraw(), между которыми выполняются все операции вывода графической информации на устройство вывода.

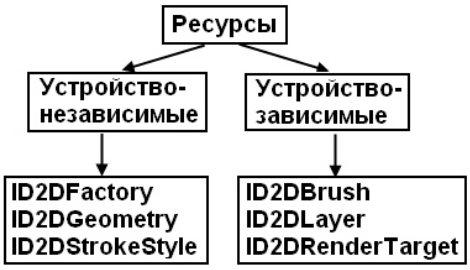

Рисунок 1 – Классификация ресурсов Direct2D

В качестве инструмента вывода используются объекты класса ID2DBrush, которые задают цвет и другие парамеиры выводимых объектов (в т.ч. градиентные заливки).

И ID2DBrush, и ID2DRenderTarget – устройство-зависимые ресурсы и будучи созданными однажды для конкретного устройства, могут применяться лишь к нему, и должны быть уничтожены всякий раз, когда уничтожается их устройство.

Объект класса ID2DGeometry – устройство-независимый ресурс. Будучи созданным однажды, он может быть использован любым объектов ID2DRenderTarget. ID2DGeometry задает двумерную форму, интерполированную треугольниками.

После того, как работа с ресурсами и объектами завершена, они должны быть уничтожены функцией Release(), которая унаследована ими от базового COM-объекта [10]. При этом по возможности стоит избегать частого создания и освобождения ресурсов, так как этот процесс требует достаточно много ресурсов процессора.

Общая схема работы с Direct2D представлена на рисунке 2.

Разработка растрово-векторной модели графического редактора

Разработанный графический редактор Kentus_Painter реализует растрово-векторный подход к работе с графикой. Для реализации этого была разработанная специальная объектная модель, позволяющая комбинировать растровые и векторные элементы объектов.

Рисунок 2 – Общая схема использования компонент Direct2D

В этой модели сцена (формируемый документ) состоит из совокупности объектов. Каждый объект задается расположением в пространстве документа, а также трансформациями, примененными к нему. При этом каждый объект состоит из множества точно таких же объектов, как он сам, которые задаются положением и трансформацией внутри самого родительского объекта. К дочерним объектам также применяются трансформации родительского.

Иерархия объектов системы представлена на рисунке 3.

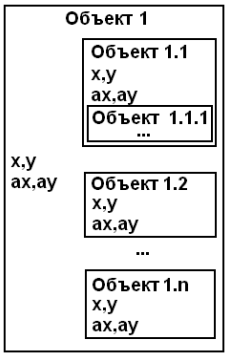

Рисунок 3 – Иерархия объектов в системе

Такая иерархия объектов позволяет создавать сложные объекты из множества простых, при этом каждый из простых объектов может обладать своими собственными свойствами, или же наследовать свойства объекта-родителя.

При этом предусмотрены также и простые объекты – геометрические формы, битовые изображения, маски, которые замыкают эту

иерархическую цепочку и собственно и являются тем, что в конечном итоге будет выведено на экран. Они делятся на два типа – растровые (битовые изображения, маски) и векторные (геометрические фигуры). Таким образом, сложные объекты имеют в своем составе как векторные части, так и растровые.

Результирующая сцена при этом формируется путем применения всех заданных операций (перемещение, трансформация) к каждому из объектов, растеризации объектов в пространстве документа и последовательном отображении объектов на документ. Схематично этот процесс изображен на рисунке 4.

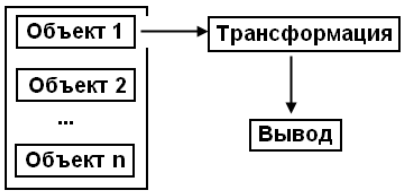

Рисунок 4 – Схема формирования результирующей сцены

Реализация системы средствами Direct2D

Не всегда все, что выводится на экран, является изображением, над которым выполняется текущая работа. Так, элементы интерфейса, дополнительные обозначения, рамки выбора – все это элементы, которые не принадлежат к текущему изображению, текущей сцене. Кроме того, при использовании объекта ID2DRenderTarget, ассоциированного с окном, содержимое вывода нигде не сохраняется – оно лишь отображается в окне.

В связи с этим, целесообразно создать два объекта класса ID2DRenderTarget. Первый – ассоциируется с битовым изображением и хранит в себе текущее состояние сцены. То есть то, каким будет изображение после растеризации объектов и сохранения в файле. Второй – ассоциирован с окном приложения и предназначен для вывода на экран пользователя как текущего состояния сцены, так и сопутствующей информации, не предназначенной для сохранения и к сцене не относящейся.

Процесс создания обоих устройств вывода представлен на рисунке 5.

При формировании изображения все объекты выводятся сначала в RenderTarget, связанный с битовым изображением (RT_BitMap), а затем оно выводится в RenderTarget (RT_HWND), связанный с окном.

Схема взаимодействия разных устройств вывода представлена на рисунке 6.

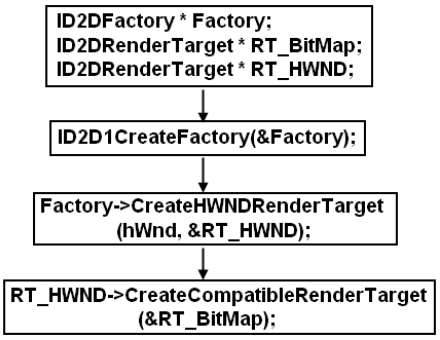

Рисунок 5 – Процесс создания устройств вывода

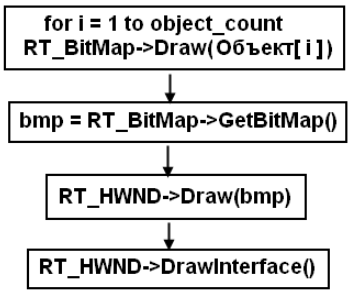

Рисунок 6 – Схема взаимодействия устройств вывода

Суть работы этой схемы можно продемонстрировать на примере.

Сперва загрузим в систему Kentus_Painter некоторое изображение формата jpg. Оно будет загружено в объект класса ID2DRenderTarget, ассоциированный с изображением, то есть в RT_Bitmap. Окно программы после загрузки первого изображения показано на рисунке 7.

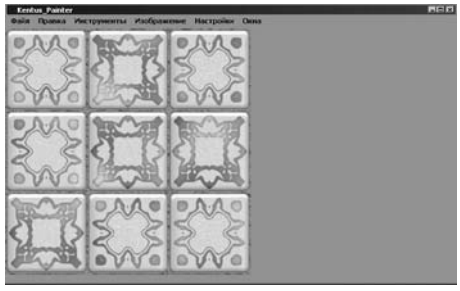

Рисунок 7 – Вид окна системы после загрузки изображения.

При этом пользователь видит загруженное изображение потому, что при отрисовке окна происходит копирование содержимого RT_Bitmap в объект класса ID2DRenderTarget, ассоциированный с окном, то есть в RT_HWND.

Затем, выбрав инструмент “Кисть”, нарисуем на поверхности изображения красную звезду, со значением альфа-канала равным 100. Результат этой операции показан на рисунке 8.

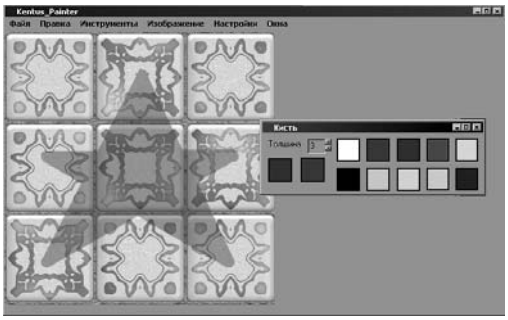

Рисунок 8 – Результат применения инструмента „Кисть”

Измененное изображение также хранится в RT_Bitmap.

Иногда для удобной работы с изображением целесообразно пользоваться координатной сеткой. При включении режима “Сетка” вывод каждого кадра на экран происходит следующим образом : сначала из RT_Bitmap изображение копируется в RT_HWND, а затем RT_HWND выполняет операции по выводу сетки. Таким образом, исходное изображение не затрагивается. Результат активации режима “Сетка” можно увидеть на рисунке 9.

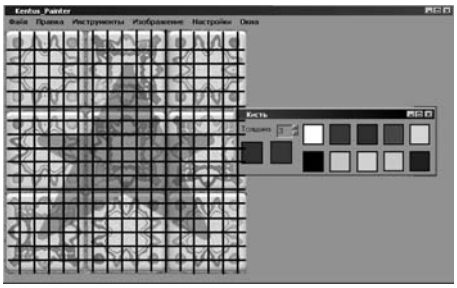

Рисунок 9 – Результат включения режима “Сетка”

Перспективы развития системы

В системе Kentus_Painter планируется реализовать удобную работу как с растровыми частями объектов, так и обработку их векторных составляющих. Для этого будут добавлены инструменты воздействия на растр – заливка, выделение, инструменты тоновой и цветовой коррекции.

Также будут добавлены инструменты для работы с векторными составляющими объектов – добавление простых геометрических форм к объекту, задание и изменение параметров объекта и его частей.

Кроме того, планируется введение специальных сценариев, которые могли бы описывать поведение объектов во времени, тем самым создавая анимационные эффекты и формируя сложные изображения.

Выводы

С появлением новой технологии Direct2D от Microsoft стала возможна разработка графических редакторов нового поколения, которые могли бы использовать всю мощь аппаратного обеспечения современных ПК для достижения сложных и зрелищных эффектов при построении изображений. Одной из главных задач разработки данной системы является использование всех возможных преимуществ Direct2D, а также создание редактора, который упрощал бы создание изображений, а также улучшал бы их качество.

Литература

1. Карабчевский В.В. Визуальное создание двумерных текстур средствами DirectX 9.0c / В.В. Карабчевский, С.Н. Магдалина // Наукові праці Донецького національного технічного університету. Серія “Інформатика, кібернетика та обчислювальна техніка”. – 2009. – Випуск 10 (153). – С. 167–171.

2. Тайц А. Самоучитель Adobe Photoshop 7 / А. Тайц, А. Тайц. – СПб.: БХВ-Петербург, 2005. – 688 с.: ил.

3. Миронов Д. Corel Draw 11. Учебный курс / Д. Миронов. – СПб.: Питер, 2002. – 448 с.: ил.

4. Горнаков С. DirectX 9: Уроки программирования на C++ / С. Горнаков. – СПб.: БХВ – Петербург, 2005. – 400 с.: ил.

5. Фленов М. DirectX и C++. Искусство программирования / М. Фленов. – СПб.: БХВ – Петербург, 2006. – 384 с.: ил.

6. Френк Д. Луна. Введение в программирование трехмерных игр с DirectX 9.0. Wordware Publishing / Френк Д. Луна. – 2006. – 424 с.: ил.

7. Бэррон Т. Программирование стратегических игр с DirectX 9.0. Wordware Publishing / Т. Бэррон. – 2003. – 700 с.: ил.

8. Миллер Т. Managed DirectX 9.0 с управляемым кодом. Программирование графики и игр. SAMS / Т. Миллер – 2003. – 432 с.: ил.

9. http://msdn.microsoft.com – документация по DirectX и Direct2D.

10. Секунов Н.Ю. Самоучитель Visual C++ / Н.Ю. Секунов. – СПб.: БХВ-Петербург, 2002. – 250 с.: ил.

Надійшла до редакції 05.03.2011