УДК 681.3

Визуальное создание двумерных текстур средствами DirectX 9.0c

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

Abstract

Karabchevsky V., Magdalina S. Visual 2-D textures creating with using DirectX 9.0c. *The methods of 2-D textures creation are considered, the system for algorithmic 2-D textures creation is presented.*

Введение

В настоящее время двумерные изображения являются одними из самых распространенных объектов компьютерной графики. Они применяются в игровой индустрии в качестве текстур объектов и элементов интерфейса. Интернет также заполнен различными анимированными и статическими изображениями.

В связи с этим, очень важным является вопрос создания качественных текстур и просто изображений, которые могли бы послужить достойным оформлениям сопутствующей им информации. С учетом того, что спрос на программные продукты самого разного типа ныне крайне велик, актуальным является ускорение работы над созданием каждого изображения и упрощение этого процесса.

Это приводит к созданию разнообразных графических пакетов для создания двумерных изображений и анимации.

Анализ существующих программ и методов создания изображений

Наиболее популярными программами создания и редактирования изображений являются MS Paint, Adobe Photoshop, Corel Draw и другие. Каждая программа хоть и имеет множество различных функций, все же ориентирована на выполнение определенной задачи.

Так, MS Paint является наиболее простым из редакторов графических файлов. В его возможности входят простейшие операции над пикселями изображения – прямое рисование на изображении, создание простых одноцветных фигур, заливка. MS Paint является редактором для начинающих и не пригоден для сложной обработки изображений.

Adobe Photoshop – один из наиболее популярных редакторов обработки готовых изображений [1]. Несмотря на наличие достаточно мощных инструментов для непосредственного рисования на изображении, основная его задача – коррекция областей изображения в целом. Он имеет широкие возможности по выделению частей изображений различными способами – непосредственным ограничением области инструментов «Лассо», формированием выделенной области, состоящей из близких по цвету пикселей и другие. В дальнейшем к выделенной области можно применять различные операции – цветовую и тоновую коррекцию, добавление различных эффектов и т.п. Adobe Photoshop – мощный редактор, предоставляющий широкие возможности по обработке фотографий [2]. Среди его недостатков можно указать достаточно высокую сложность в использовании и тот факт, что для успешного применения его инструментов нужен немалый опыт работы с этой программой.

Corel Draw – похож по своему функциональному назначению на Adobe Photoshop. Главным отличием между ними является то, что Photoshop работает с растровой графикой, а Corel Draw – с векторной [3]. Отсюда вытекает основное преимущество этого редактора – представление изображения в виде совокупности объектов, каждый из которых имеет свойства, присущие только объектам его типа. Благодаря этому имеется возможность обработки каждого объекта наиболее подходящим в той или иной ситуации способом [4]. Основная функция Corel Draw – непосредственное создание готового изображения «с нуля», хотя он имеет и инструменты обработки готовых изображений в целом.

Постановка задачи визуального создания текстур

Задачей традиционных редакторов является формирование непосредственно конечного изображения. При этом они оказываются бесполезны, если ныне существует необходимость формирования изображений программными средствами, чтобы созданное изображение отвечало некоторым параметрам, значения которых меняются во времени. Простейшим примером такого формирования может быть некоторый тепловой индикатор, который в зависимости от температуры менял бы

свой цвет от зеленого к красному. Для его реализации приходится писать отдельную программу, которая по значению температуры рисовала бы на экране индикатор соответствующего цвета. Другим, более сложным, примером являются динамические текстуры, используемые в компьютерных играх. Написание программ для их формирования также выполняется вручную, что не всегда быстро.

Возникают две задачи. Первая – автоматизация разработки сходных по способу создания изображений. При этом есть изображения, которые строятся по случайному принципу. К примеру, текстура капель дождя состоит из совокупности капель разного размера и случайного количества.

Вторая является логическим продолжением первой. Если есть последовательность операций (алгоритм), то можно написать программу. Потому второй задачей системы является генерация программного кода, реализующего заданную пользователем последовательность операций.

Структура системы визуального создания текстур

Разработанная система Kentus_DX представляет собой систему, написанную на DirectX 9.0c (версия октябрь 2008). Структура файлов и каталогов системы показана на рисунке 1 (предположим, что корневой каталог системы называется Kentus_DX) .

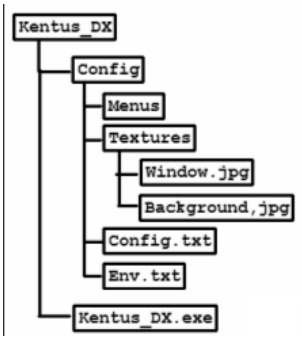

Рисунок 1 – Файловая структура системы

В корневом каталоге находится исполнимый файл программы Kentus_DX.exe и каталог Config, содержащий информацию о настройках системы.

Каталог Config содержит два файла : Config.txt и Env.txt, и два каталога : Menus и Textures.

Файл Config.txt содержит настройки экрана, создаваемого устройства DirectX (IDirect3DDevice), количество окон в системе и задержку на экране информационного окна в кадрах.

Файл Env.txt содержит преимущественно информацию о цветах интерфейса и шрифтах системы.

В каталоге Menus находятся все файлы окон системы (по одному файлу на каждое окно, включая контекстные). Файлы окон имеют расширение *.mnu и названия, соответствующие номеру окна в системе, начиная с нуля. Считывание окон выполняется последовательно, начиная с нулевого.

Каталог Textures хранит обязательные для работы системы текстуры. Background.jpg – текстура фоновой поверхности системы, Window.jpg – текстура задней поверхности окон.

В целом структура системы выбрана так, чтобы она максимально соответствовала назначению системы – созданию текстур и генерации программного кода на языке C++.

Интерфейс системы

Система Kentus_DX имеет свою собственную иерархию элементов интерфейса. Элементами интерфейса в системе являются : окно, изображение, кнопка, поле ввода, выпадающий список. Все элементы представляют собой прямоугольные области и являются аналогами соответствующих элементов операционной системы Windows.

Так, окно системы является вместилищем всех других элементов и может быть обыкновенным и контекстным. При этом контекстное окно закрывается автоматически после потери фокуса. Контекстные окна, как правило, служат для отображения возможных вариантов выбора для элемента управления список. Окна можно перемещать по рабочей поверхности приложения и изменять их размеры. При изменении размера окна размер элементов управления, находящихся в нем, пропорционально меняется.

Пример окна показан на рисунке 2.

Рисунок 2 – Окно настройки параметров графики

Элементы управления – кнопки, выпадающие списки, изображения и поля ввода

реагируют на перемещение мыши, изменяя свой внешний вид в зависимости от действий пользователя. Также каждый элемент управления имеет свои функции-обработчики происходящих событий (нажатие на элементе мышью, завершение ввода в поле ввода и т.п.).

Каждый элемент управления имеет набор текстур, соответствующих его состоянию. Текстуры эти создаются при инициализации приложения в соответствии с параметрами, указанными в файле Env.txt.

Инициализация DirectX

Детальное рассмотрение всех классов, используемых в системе и всех интерфейсов DirectX выходит за рамки данной статьи, однако, рассмотрим основные из них и режимы работы системы.

При запуске программы сперва происходит считывание параметров основных объектов DirectX из файла Config\Config.txt. Это такие параметры, как разрешение экрана, тип обработки вершин, тип устройства DirectX, оконный режим или полноэкранный, частота обновления кадров и еще несколько параметров, не относящихся к инициализации DirectX.

Вообще весь этап по инициализации Direct3D 9 можно разделить на три части. Сначала мы создаем объект Direct3D 9, потом совершаем установки параметров представления для Direct3D 9 и в завершение, на основании произведенных установок, создаем устройство Direct3D 9 [5].

Объект Direct3D 9 (IDirect3D9) создается на основании текущей версии SDK DirectX и особенных параметров при создании не требует.

Параметры представления представлены структурой D3DPRESENT_PARAMETERS и весьма разнообразны. Однако, основными при создании устройства Direct3D9 (интерфейс IDirect3DDevice9) являются : разрешение экрана, полноэкранный режим или оконный, способ обработки вершин и тип устройства Direct3D.

Разрешение экрана в оконном режиме определяет размеры окна приложения, а в полноэкранном изображение будет выводится в область указанного разрешения, а затем растянуто или сужено до размеров экрана.

Тип устройства может быть одним из трех констант :

- D3DDEVTYPE_HAL (значение 1) – использовать аппаратные возможности. Этот параметр является более предпочтительным, потому что система ориентирована на современные видеокарты, которые аппаратно поддерживают DirectX 9;
- D3DDEVTYPE_REF (значение 2) – использовать программные возможности, но если есть возможность, можно применять специализированные инструкции процессора;
- D3DDEVTYPE_SW (значение 3) – использовать программное устройство; [6]

Важным параметром при создании устройства Direct3D является обработка вершин. Вершины представляют собой основные строительные блоки трехмерной (и двумерной) геометрии и могут обрабатываться двумя различными способами: программно (*программная обработка вершин*) или аппаратурой видеокарты (*аппаратная обработка вершин*). Программная обработка вершин всегда поддерживается и может быть использована. Аппаратная обработка вершин, с другой стороны, может использоваться только если она поддерживается установленной видеокартой.

Аппаратная обработка вершин всегда предпочтительнее, поскольку выполняется гораздо быстрее, чем программная. Кроме того, она освобождает центральный процессор, который может выполнять другие задачи [7].

Для установки способа обработки используются три константы:
D3DCREATE_SOFTWARE_VERTEXPROCESSING,
D3DCREATE_HARDWARE_VERTEXPROCESSING,
D3DCREATE_MIXED_VERTEXPROCESSING.
Первая имеет значение 32 и задает программную обработку вершин, вторая имеет значение 64 и задает аппаратную обработку вершин и третья имеет значение 128 и задает смешанную обработку вершин (то есть по возможности будут использоваться аппаратные средства) [8].

Наконец, последним является параметр частоты обновления кадров, задаваемый целым числом кадров в секунду. Однако, работать с ним необходимо очень осторожно, потому что неправильная установка этого значения может привести к поломке монитора. Если же хочется обойтись без лишнего риска, то можно использовать значение частоты обновления, установленное по умолчанию операционной системой. Для этого необходимо задать значение частоты -1 или включить соответствующую опцию в настройках системы [9].

Благодаря тому, что система визуального моделирования текстур имеет такое количество настраиваемых параметров, она может быть настроена под конкретную конфигурацию ПК оптимальным образом.

Формат конфигурационного файла представлен на рисунке 3.

Рисунок 3 – Формат конфигурационного файла

Формирование результирующего изображения

Результатом работы системы является результирующее изображение, которое получается из начального, минуя череду промежуточных состояний – кадров. Каждый следующий кадр получается из текущего путем применения к нему заданных операций. Схема формирования результирующего изображения представлена на рисунке 4.

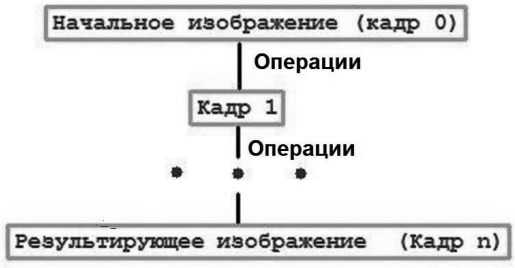

Рисунок 4 – Схема формирования результирующего изображения

При этом каждый кадр состоит из текущего изображения, текущего набора объектов системы и набора операций, применение которых к объектам кадра осуществит переход к следующему кадру. Структура кадра представлена на рисунке 5.

Рисунок 5 – Структура кадра изображения

Объекты системы бывают трех видов – точки, линии и области. Каждый из объектов данного типа имеет свои параметры, определяющие его форму, положение на текстуре, цвет, привязку к другим объектам, а также тип отображения.

Тип отображения определяет, будет ли объект выводиться на текстуру или же он видим только в редакторе и таким образом служит объектом привязки.

Каждый параметр объекта может быть задан тремя способами : непосредственно значением, случайным значением из некоторого интервала и привязан к соответствующему параметру другого объекта. Последний способ означает, что при создании объекта или изменении этого параметра, он (параметр) примет такое значение, которое имеет сходный параметр объекта-привязки. В этом проявляется динамизм создания текстуры – задав случайным образом некий параметр в базовом объекте привязки, все позднее создаваемые объекты, привязанные к базовому также будут создаваться с некоторой степенью случайности.

Наконец, операции в системе могут быть двух основных видов – создание объекта и изменение его параметров. Изменение параметров объекта происходит за некоторое количество шагов, при этом, если объект выводим на текстуру, он выводится в каждом из промежуточных состояний, оставляя за собой своеобразный шлейф. Это позволяет легко добиться весьма необычных эффектов. Так, всего лишь меняя координаты и цвет окружности, можно создать прямую, цвет которой представляет градиент от начального цвета к конечному. Этот эффект продемонстрирован на рисунке 6.

Рисунок 6 – Эффект градиента, полученный путем изменения координаты и цвета окружности

Это лишь простейший пример одной операции над одним объектом.

Более сложные операции в данный момент находятся в разработке, однако уже на данном этапе удалось получить текстуру, показанную на рисунке 7.

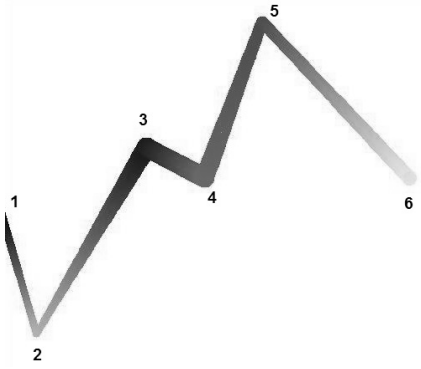

Рисунок 7 – Текстура зигзагообразной линии

Особенностью данной текстуры является что, что точки 1-6 имеют произвольное

расположение по оси Y и произвольное – по оси X. Однако, по оси X никакая следующая точка не может быть «левее» предыдущей. Размер каждой контрольной точки также произволен.

Применяя набор операций, использованных при создании этой текстуры вновь, мы получим другую ломаную линию – другой толщины и с другим расположением изгибов. Такая текстура является прообразом текстуры молнии, каждая ветвь которой вибрирует в заданной области текстуры.

Перспективы развития системы

На данный момент в системе реализована лишь работа с простыми объектами и выполнение несложных операций над ними. В будущем планируется разработка более сложных способов воздействия на объекты. Также будет предусмотрена возможность сохранения последовательности операторов, с помощью которой была получена результирующая текстура. Эта последовательность могла бы быть использована для создания подобных текстур или же как часть другой, более обширной последовательности операций.

Еще одним очень полезным новшеством станет внедрение конвейера операций. Так, при создании продемонстрированной выше текстуры было применено множество операций по созданию объектов-окружностей и точек привязки. Между собой эти операции отличались лишь одним параметров – интервалом по оси X для точек привязки и номерами точек привязки для объектов окружностей. Логично было бы встроить в систему модуль, который генерировал бы однотипные операции с различными параметрами, которые менялись бы по заданному пользователем закону. Схема работы такого конвейера показана на рисунке 8.

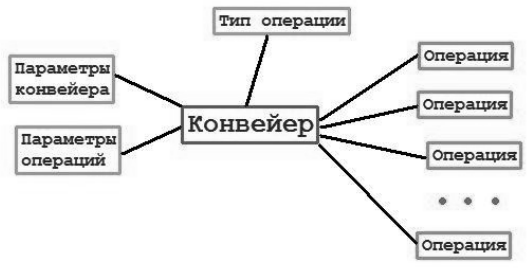

Рисунок 8 – Схема работы конвейера операций

Внедрение такого конвейера позволило бы избежать рутинной работы по вводу однотипных команд.

Кроме этого, будут добавлены функции также непосредственной обработки изображения, позволяющей воздействовать непосредственно на пиксели изображения.

Наконец, самым главным усовершенствованием станет возможность автоматической генерации кода по заданной последовательности операций.

Выводы

Подводя итог, можно утверждать, что были совершены первый шаги по достижении поставленной цели. С помощью разработанной системы можно создавать несложные текстуры, используя привязку или же случайное задание некоторых параметров.

Реализация автоматической генерации кода упростит работу программистов, создающих анимационные подпрограммы в графических приложениях.

Литература

1. Александр Тайц, Александра Тайц. Самоучитель Adobe Photoshop 7. – СПб.: БХВ-Петербург, 2005. – 688 с.: ил.
2. Владимир Легейда. Photoshop CS2. Настоящий самоучитель. Век +, Корона-принт, НТИ, 2006. - 528 с.:ил.
3. Миронов Д. Corel Draw 11. Учебный курс. СПб.: Питер, 2002. – 448 с.: ил.
4. Corel Press. Corel Draw 11. The official guide. McGraw-Hill Osborne Media, 2003. - 827 с.:ил.
5. Станислав Горнаков. DirectX 9: Уроки программирования на C++. СПб.: БХВ – Петербург, 2005. – 400 с.: ил.
6. Михаил Фленов. DirectX и C++. Искусство программирования. СПб.: БХВ – Петербург, 2006. – 384 с.: ил.
7. Френк Д. Луна. Введение в программирование трехмерных игр с DirectX 9.0. Wordware Publishing, 2006. - 424 с.: ил.
8. Тодд Бэррон. Программирование стратегических игр с DirectX 9.0. Wordware Publishing, 2003. - 700 с.: ил.
9. Том Миллер. Managed DirectX 9.0 с управляемым кодом. Программирование графики и игр. SAMS, 2003. - 432 с.: ил.
10. Секунов Н.Ю. Самоучитель Visual C++. СПб.: БХВ-Петербург, 2002. – 250 с.: ил.

Поступила в редколлегию 12.03.09