Алексей Попов

bhv

DirectX 10 — это просто

Программируем графику на C++

Использование Direct3D 10 для вывода графики

Программирование вершинных, пиксельных и геометрических шейдеров на языке HLSL

Быстрое создание приложений с помощью библиотеки DXUT (DirectX Utility)

Справочник функций DirectX 10 и библиотеки DXUT

+cd

Алексей Попов

Программируем графику на C++

Санкт-Петербург

«БХВ-Петербург»

2008

УДК 681.3.06
ББК 32.973.26-018.1
П58

Попов А. А.

П58 DirectX 10 — это просто. Программируем графику на C++. — СПб.: БХВ-Петербург, 2008. — 464 с.: ил. + CD-ROM — (Внесерийная)

ISBN 978-5-9775-0139-2

Рассмотрено создание графических приложений для Windows на C++ с использованием новой версии компонента Direct3D 10, входящего в состав библиотеки DirectX 10. Описывается вывод двухмерной и трехмерной графики, связанное с этим программирование вершинных, пиксельных и геометрических шейдеров на языке HLSL. Отдельная часть книги посвящена быстрой разработке приложений с помощью библиотеки DXUT (DirectX Utility). Материал сопровождается практическими примерами, а также справочником функций библиотек DirectX 10 и DXUT. На прилагаемом компакт-диске находятся все исходные тексты примеров, рассмотренных в книге.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Юрий Якубович*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн обложки	*Инны Тачиной*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.11.07.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 37,41.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0139-2

Оглавление

Введение

Будущее надвигается стремительно: те компьютеры, которые несколько лет назад были пределом мечтаний, сегодня вызывают лишь усмешку. Быстродействие процессоров все растет, увеличивается объем оперативной памяти и емкость жестких дисков. Технологии тоже развиваются, стремятся использовать возможности нового "железа" на все 100%. За период времени, чуть больший двух десятилетий, был пройден путь от перфокарт до виртуальной реальности. Что ждет нас впереди? К чему готовиться? В этой книге вы можете познакомиться с библиотекой DirectX 10, которая, очевидно, получит широкое распространение в ближайшем будущем. Эта книга о трехмерной графике. Если лично вам было бы интересно уже сегодня прикоснуться к графической библиотеке, на которой, скорее всего, будут построены еще не известные игровые хиты будущего, то вы взяли с полки правильную книгу.

Кому адресована эта книга?

Книга ориентирована на начинающего программиста, имеющего опыт написания программ на языке C++ хотя бы для MS-DOS. Желательно знакомство со средой разработки Visual Studio, хотя краткое описание работы с ней приводится в книге. Дополнительный опыт составления программ для Windows, а также работы с предыдущими версиями DirectX пойдет только на пользу.

Как пользоваться этой книгой?

В книге содержится как учебный (*части I* и *II*), так и справочный материал (*приложения*). Вопросы рассматриваются по мере увеличения их сложности, поэтому желательно знакомиться с главами в том порядке, в котором они расположены в книге. В *части I* мы рассматриваем общие принципы

программирования под Windows и создания приложений Direct3D 10. Вот в каком порядке это излагается.

- *Глава 1* содержит вводные сведения о DirectX, а также расскажет о том, как установить на компьютер DirectX SDK и настроить среду разработки для компиляции программ.
- *Глава 2* познакомит с принципом работы приложений Windows и научит создавать минимальное приложение Direct3D 10.
- *Глава 3* введет в курс некоторых элементов математики, которые используются при программировании трехмерной графики.
- *Глава 4* научит выводить на экран текст и изображения из файла.
- *Глава 5* подготовит к программированию шейдеров с использованием языка HLSL.
- *Глава 6* расскажет, как вывести на экран треугольник и анимировать его.
- *Глава 7* покажет, как добавить в программу расчет освещения и наложить на поверхности текстуры.
- *Глава 8* познакомит с нововведением десятой версии DirectX с геометрическими шейдерами.

В *части II* рассматривается создание программ с использованием каркаса DXUT, который позволяет отвлечься от таких рутинных операций, как создание окон или устройств Direct3D, и сосредоточиться непосредственно на реализации алгоритма программы. Вот какие темы освещаются в главах второй части.

- *Глава 9* содержит общие сведения о программном каркасе DXUT и его функционировании.
- *Глава 10* научит, как загружать и выводить на экран трехмерные модели средствами DXUT.
- *Глава 11* расскажет о работе с интерфейсом пользователя, который DXUT предоставляет в распоряжение программиста.
- *Глава 12* познакомит с созданием собственных диалоговых окон DXUT, а также с использованием стандартного окна настроек Direct3D.

В *приложении 1* находится справочник по функциям DirectX 10, в *приложении 2* можно познакомиться с некоторыми полезными функциями DXUT. Таким образом, книгу можно использовать и как учебник, и как справочник для разбора примеров программ. Также нужно отметить, что в книге используется стиль, который можно условно назвать "стиль университетских лекций", поэтому, если вы видите слово "вы", написанное со строчной буквы, не принимайте это как знак неуважения, это всего лишь обращение к аудитории.

Что нам понадобится?

Поскольку библиотека DirectX 10 в настоящее время имеется только для Windows Vista, для работы нам понадобится компьютер с этой операционной системой. Чтобы в полной мере оценить возможности новой версии DirectX, лучше также иметь видеокарту, аппаратно поддерживающую DirectX 10, однако обязательным условием это не является.

Все программы, описанные в книге, были составлены и скомпилированы в среде Visual Studio 2005. Предыдущая версия, Visual Studio 2003, тоже должна работать, однако это не проверялось. Программы компилировались с использованием DirectX SDK за август 2007 года.

Благодарности

Хочу выразить огромную признательность моей любимой жене Наташе за то, что она поддерживала меня во время подготовки книги и создавала все условия для того, чтобы она увидела свет.

Своих родителей я благодарю за то, что с детства прививали мне интерес ко всему новому и учили мыслить самостоятельно.

Слова благодарности я также должен сказать моим школьным учителям, в особенности Ивашевой Светлане Валерьевне, учительнице математики, и Смолко Светлане Георгиевне, учительнице русского языка и литературы. Они научили меня, как я надеюсь, излагать свои мысли на родном языке. Если вдруг в книге что-то изложено не логично, или просто "не по-русски", то это только потому, что я забыл, чему они меня учили. Еще обязательно нужно поблагодарить Белоусову Любовь Юрьевну, учительницу по информатике, за то, что она поддержала мой интерес к компьютерной технике, когда я учился в школе: не будь ее понимания и поддержки в то время, эта книга, скорее всего, сейчас бы перед вами не лежала.

ЧАСТЬ I

ПОСТРОЕНИЕ ПРИЛОЖЕНИЙ НА ОСНОВЕ DIRECTX 10

Глава 1

Собираемся в путь

Что такое DirectX и зачем он нужен?

Этот вопрос, наверное, покажется глупым. Каждый ведь знает, для чего нужен DirectX. DirectX — это такая штука, с помощью которой пишутся игры. Все ведь понятно! Однако такой ответ, нужно признать, далек от описания полной картины. Более-менее полное определение выглядит так: DirectX представляет собой набор классов, функций и структур, обеспечивающих высокоэффективное выполнение задач, возникающих при разработке компьютерных игр и мультимедийных приложений. К таким задачам относятся: загрузка и анимация трехмерных моделей, наложение текстур на объемные объекты, воспроизведение звуковых файлов и многие другие. Другими словами, DirectX представляет собой средство разработки приложений, интенсивно использующих мультимедийные возможности компьютера, призванное облегчить жизнь программиста. Если быть до конца точным, DirectX состоит из следующих компонентов.

- **Direct3D** позволяет загружать в память трехмерные модели, осуществлять наложение текстур, рассчитывать освещенность и выводить результат на экран. Именно эта составляющая, отвечающая за вывод двухмерной и трехмерной графики на экран, нас и будет интересовать.
- **DirectSound** отвечает за работу со звуком, его можно использовать при разработке программ для воспроизведения и захвата звукового сигнала.
- **DirectInput** организует работу с устройствами ввода: с клавиатурой, мышью, различного вида джойстиками, включая полную поддержку технологии обратной связи.

Также являются частью DirectX, но считаются устаревшими, следующие компоненты. Многие из них не рекомендуются для разработки новых программ.

- **DirectDraw** присутствовал в DirectX до седьмой версии и отвечал за двухмерную графику. После появления Direct3D необходимость в данном

компоненте отпала, его функции теперь несет на себе Direct3D, хотя возможностями DirectDraw можно пользоваться до сих пор.

- **DirectMusic** отвечает за воспроизведение музыки. Имеется возможность воспроизведения файлов MIDI и WAV. Единственный из перечисляемых компонентов, который не считается устаревшим. Как сказано в документации, "...сохранит свой текущий статус, пока в данной области не появится новая технология".
- **DirectShow** служит для воспроизведения различного рода мультимедийных файлов: MPG, MP3 и других. По сути, он представляет собой программируемый проигрыватель.
- **DirectPlay** позволяет создавать многопользовательские игры, отвечая за обмен данными по сети.

Тем не менее, информацию по использованию всех этих компонентов можно найти в комплекте документации, поставляемой с DirectX SDK, и в базе знаний Microsoft по адресу в Интернете **http://msdn.microsoft.com**.

Немного истории: от Windows 95 к Windows Vista

Трудно представить это сегодня, но сравнительно недавно, чуть больше десяти лет назад, не было ни операционной системы Windows, ни, тем более, каких-либо средств разработки мультимедийных приложений для нее. Строго говоря, Windows существовала, хотя и не в виде операционной системы, а в виде своего рода графической оболочки, запускаемой из-под MS-DOS. В компьютерных изданиях того времени встречались рекомендации не использовать ее якобы из-за того, что простота и доступность графического интерфейса впоследствии затрудняет изучение текстовых команд MS-DOS.

При написании игровых программ в системе MS-DOS программистам приходилось самим реализовывать практически все функции для вывода графики. С одной стороны, это усложняло и затягивало разработку, с другой — можно было пользоваться прямым доступом к оборудованию и "выжимать" из него максимум производительности. Можно было, например, формировать изображение на экране, копируя необходимую информацию прямо в память видеокарты.

С выходом операционной системы Windows 95 ситуация изменилась: приложения в новой системе больше не имели прямого доступа к оборудованию. Вывод графики осуществлялся только с привлечением системных функций, а скорость их работы была далека от той, которая требовалась для разработки

динамичных игр. Это было платой за так называемый *уровень аппаратных абстракций* (HAL — hardware abstraction layer) новой операционной системы, который позволял работать с устройствами компьютера без учета их особенностей в зависимости от фирмы-производителя. Конечно, игры под новую систему были, взять хотя бы набор, поставляемый вместе с Windows. От версии к версии он изменялся мало: "Сапер", пожалуй, самая узнаваемая, "Червы", различные виды карточных пасьянсов. А в MS-DOS, уже "старой" системе, можно было поиграть в "Doom"… Таким образом, сравнение возможностей для разработчиков игр пока было явно не в пользу Windows, требовалось срочно принимать меры. И корпорация Microsoft их приняла. Первая версия библиотеки DirectX (тогда она называлась Windows Games SDK) вышла уже в сентябре 1995 года. В то время, конечно, в составе библиотеки еще не было компонента Direct3D, но и трехмерные ускорители к тому времени еще не получили широкого распространения. Историю Direct3D, наверное, следует начинать с 1992 года, когда в компании RenderMorphics началась разработка трехмерного графического интерфейса прикладного программирования (application programming interface, API) "Reality Lab" для использования в медицине и системах автоматизированного проектирования. В феврале 1995 года компания RenderMorphics была куплена Microsoft, и в результате первая версия API для трехмерной графики была включена во вторую и третью версии библиотеки DirectX. Тем не менее, основная часть разработчиков игр перешла на DirectX лишь с выходом его третьей версии, а с восьмой версии библиотека DirectX стала практически стандартом для разработки компьютерных игр. Примерная хронология выхода версий DirectX представлена в табл. 1.1.

Таблица 1.1. *Хронология выхода версий DirectX*

Версия DirectX	**Операционная система**	**Дата выхода**
DirectX 1.0	Windows 95a	30 сентября 1995
DirectX 2.0 / 2.0a	Windows 95 OSR2 и NT 4.0	5 июня 1996
DirectX 3.0 / 3.0a	Windows NT 4.0 SP3 (последняя версия с поддержкой DirectX для Windows NT 4.0)	15 сентября 1996
DirectX 4.0	-	не выходила
DirectX 5.0	Была доступна в бета-версии под Windows NT 5.0, инсталлировалась под Windows NT 4.0	16 июля 1997
DirectX 5.1	Windows 95/98/NT4.0	1 декабря 1997
DirectX 5.2	Windows 95	5 мая 1998

Таблица 1.1 *(окончание)*

Версия DirectX	Операционная система	Дата выхода
DirectX 5.2	Windows 98	5 мая 1998
DirectX 6.0	Windows 98/NT4.0	7 августа 1998
DirectX 6.1	Windows 95/98/98SE	3 февраля 1999
DirectX 7.0	Windows 95/98/98SE/2000	22 сентября 1999
DirectX 7.0a	Windows 95/98/98SE/2000	сентябрь 1999
DirectX 7.1	Windows 95/98/98SE/ME/2000	16 сентября 1999
DirectX 8.0	Windows 95/98/98SE/ME/2000	30 сентября 2000
DirectX 8.0	Игровая приставка Xbox	3 ноября 2000
DirectX 8.0a	Последняя версия под Windows 95	7 ноября 2000
DirectX 8.1	Windows 98/98SE/ME/2000/XP	12 ноября 2001
DirectX 9.0	Windows Server 2003	19 декабря 2002
DirectX 9.0a	Windows 98/98SE/ME/2000/XP	26 марта 2003
DirectX 9.0b	RC2	13 августа 2003
DirectX 9.0c	Windows XP SP2, Windows Server 2003 SP1, Xbox 360	13 декабря 2004
DirectX 9.0c	Совместимые с DX9.0c версии Windows, впервые включена библиотека D3DX	9 декабря 2005
DirectX 9.0c (Shader Model 3.0)	Windows XP. Последнее обновление с поддержкой Windows 98/98 SE/ME/2000 — в августе 2005	Ежемесячные обновления с августа 2005
DirectX 10.0	Новая версия DirectX (только для Windows Vista)	30 ноября 2006

Сравнение DirectX 9 и DirectX 10

Итак, чем именно новая версия отличается от своего предшественника? Можно выделить пять основных отличий:

1. Расширенные возможности программирования. В десятой версии разработчики полностью отказались от использования фиксированных функций графического конвейера. В новой версии все возлагается на шейдеры: расчет освещения, преобразование координат вершин и т. д.

2. Строгие требования к возможностям оборудования. Любая видеокарта, совместимая с Direct3D 10, обеспечивает один и тот же базовый набор возможностей. Это позволяет отказаться от введения в программу нескольких отдельных ветвей, выполняющихся в зависимости от того, какими возможностями обладает видеокарта.
3. Улучшенная производительность. Снижено количество вызовов API для выполнения часто встречающихся операций.
4. Унифицированная архитектура шейдеров. Это нововведение позволяет более рационально использовать процессор видеокарты, выделяя больше ресурсов для задач, требующих интенсивных вычислений.
5. Геометрические шейдеры и потоковый ввод/вывод. С помощью геометрического шейдера можно на основе данных, переданных с предыдущей стадии, строить новую геометрию. Результат работы геометрического шейдера можно затем вновь передать на вход графического конвейера.

Где достать DirectX 10 SDK?

Ну, вот мы разобрались, что представляет собой DirectX. Пришло время попробовать свои силы в программировании. Очевидно, что без самой этой библиотеки много напрограммировать не получится.

Раньше была традиция записывать текущую версию DirectX SDK на прилагаемый к книге компакт-диск. Но с некоторых пор корпорация Microsoft запретила распространение библиотеки третьими лицами. Поэтому на момент написания книги ее можно только скачать со страницы на сайте Microsoft по следующей ссылке: **http://www.microsoft.com/windows/directx/default.aspx**, размер файла около 450 Мбайт.

> ***Замечание***
>
> DirectX распространяется в двух вариантах: вариант для пользователей и вариант для разработчиков. Нас, конечно же, интересует вариант для разработчиков — DirectX SDK. Сориентироваться можно по объему файла: вариант для пользователей занимает примерно 50 Мбайт, вариант для разработчиков, как уже говорилось — порядка 450 Мбайт.

Установка DirectX 10 SDK на компьютер

Итак, на жестком диске компьютера появился файл, который называется, к примеру, dxsdk_feb2007.exe, если мы скачали версию за февраль 2007 года. Для выполнения установки необходимо произвести следующие действия:

1. Запустить файл dxsdk_feb2007.exe.

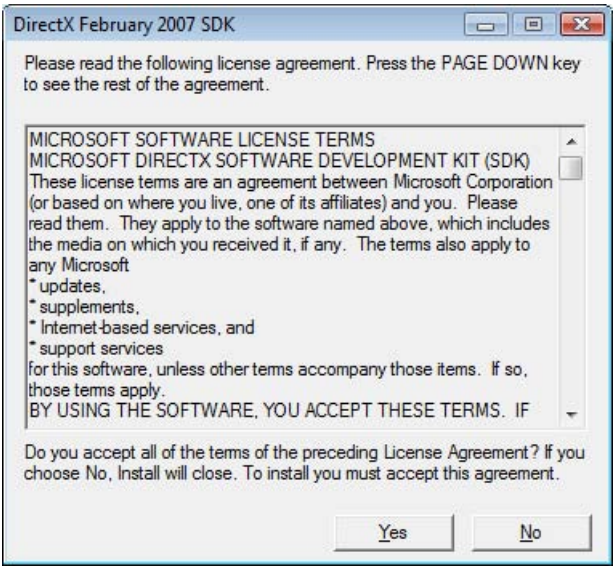

Рис. 1.1. Окно с текстом лицензионного соглашения

2. Появится окно с текстом лицензионного соглашения (рис. 1.1), его необходимо принять (нажать кнопку **Yes** (Да)), чтобы продолжить установку.

> ***Замечание***
>
> Если на компьютере уже установлена какая-либо версия DirectX SDK, перед установкой новой версии ее необходимо удалить.

3. В появившемся окне с заголовком "WinZip Self-Extractor" (рис. 1.2) ввести путь для временного хранения распакованных установочных файлов либо сразу щелкнуть по кнопке **Unzip** (Распаковать).

Рис. 1.2. Окно распаковщика файлов

Рис. 1.3. Окно программы установки

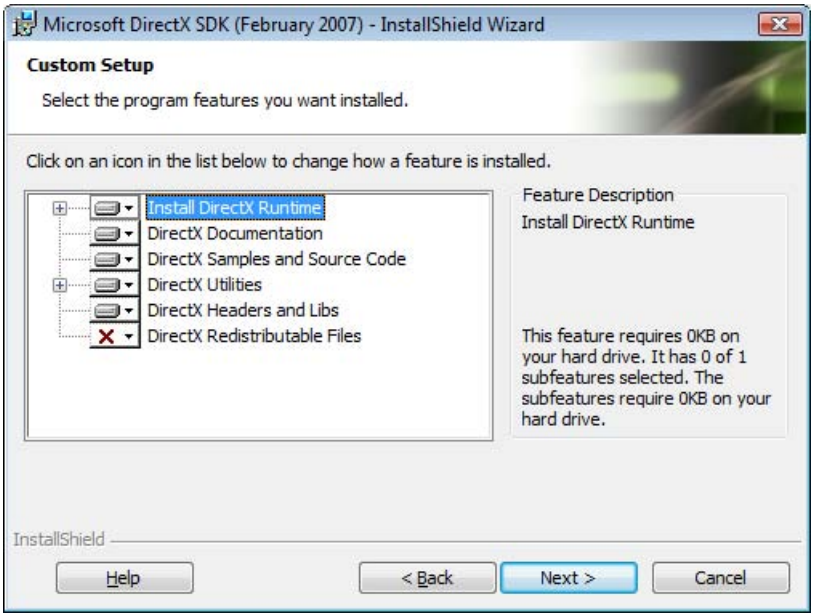

Рис. 1.4. Выбор компонентов DirectX SDK

Замечание

После окончания установки файлы из указанной временной папки автоматически не удаляются, а занимаемое ими пространство на диске составляет около 800 Мбайт. Поэтому лучше либо ввести путь до известной папки (например, C:\Temp), либо скопировать стоящий по умолчанию путь для последующего удаления файлов.

4. После распаковки запустится программа `setup.exe`, в появившемся окне (рис. 1.3) щелкаем на кнопке **Next** (Далее).
5. Для согласия с лицензионным соглашением ставим переключатель в положение "**I accept the terms in the license agreement**" ("Я принимаю условия лицензионного соглашения"), нажимаем кнопку **Next** (Далее).
6. На этом этапе можно выбрать компоненты DirectX SDK для установки (рис. 1.4). Оставляем все как есть, нажимаем кнопку **Next** (Далее).
7. После окончания копирования файлов нажимаем кнопку "**Finish**" ("Завершить"). Все, DirectX SDK установлен!

Среда разработки Visual Studio 2005

Итак, мы установили DirectX SDK на компьютер, но его использование неотделимо от использования среды разработки и компилятора. Рассмотрим работу в среде разработки Visual Studio 2005, в которой мы будем составлять все наши программы. Я думаю, вы уже не раз с ней работали, но, на всякий случай, повторим основные моменты.

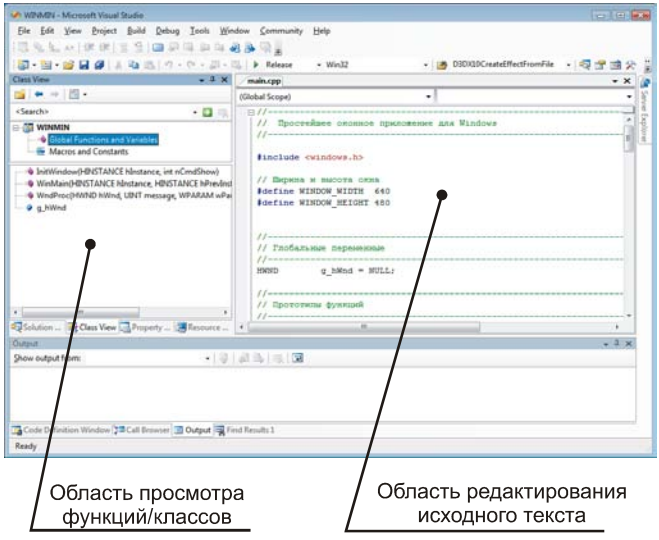

Рис. 1.5. Рабочее окно Visual Studio 2005

Общий вид рабочего окна и элементов интерфейса можно увидеть на рис. 1.5. Для нас будут иметь значение две основные области: область редактирования исходного текста и область просмотра функций/классов (**Class View**). Уже из названия понятно, что и для чего используется.

Создание нового проекта

Рассмотрим создание нового проекта в Visual Studio. В основном меню выбираем: **File | New | Project**… (Файл | Создать | Проект…). В появившемся окне (рис. 1.6) необходимо указать тип создаваемого проекта: в списке **Project types** (Типы проектов) выбираем **Win32**.

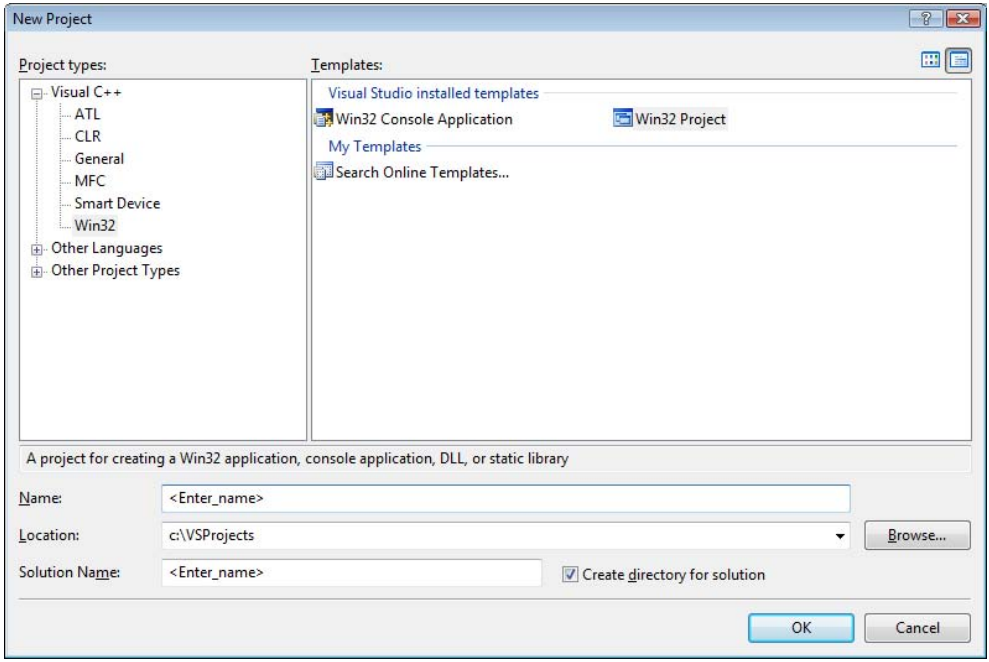

Рис. 1.6. Создание нового проекта в Visual Studio

Также необходимо задать шаблон (то есть какой-то набор настроек, подключаемых библиотек и, быть может, даже готового исходного текста), по которому будет создан наш проект. Для выбора доступны два шаблона: **Win32 Console Application** (Приложение командной строки Win32) и **Win32 Project** (Проект Win32). Нам нужен шаблон **Win32 Project**, поэтому в списке **Templates** (Шаблоны) его и выбираем. В поле **Name** (Наименование) нужно ввести название нашего проекта (здесь нужно придумать что-нибудь впечат-

ляющее, но на всякий случай имейте в виду, что имена Doom и Quake уже заняты). Текст в поле **Name** будет использован как имя для новой папки, в которой будут храниться файлы проекта. В поле **Location** (Расположение) можно указать путь до месторасположения папки с нашим проектом (рис. 1.6). Флажок **"Create directory for solution"** ("Создать директорию для решения") можно снять, на этапе обучения пусть все файлы находятся под рукой, в одной папке. По умолчанию Visual Studio предлагает создать проект в папке "Мои документы" текущего пользователя, где уже содержится папка Visual Studio 2005\Projects. Если такое местонахождение проекта нас устраивает, оставляем все как есть. После того как указаны шаблон для создаваемого проекта, его название и место расположения, нажимаем кнопку **OK**. Нас ожидает еще одно окно, которое предлагает нам настроить дополнительные параметры проекта (рис. 1.7).

Рис. 1.7. Мастер создания приложения Win32

В этом окне нам нужно сделать всего одно изменение: для этого мы переключимся на вкладку **Application Settings** (Параметры приложения), и на ней установим флажок **Empty project** (Пустой проект). Настройки приложения должны выглядеть так, как показано на рис. 1.8.

Рис. 1.8. Настройки нового приложения

Нужно также проверить, что в разделе **Application type** (Тип приложения) переключатель установлен в позицию **Windows application** (Оконное приложение). На этом все подготовительные операции кончаются, и мы наконец-то щелкаем по кнопке **Finish** (Завершить) и получаем новый проект, заготовку для нашей будущей программы.

Добавляем файлы в проект

Проект у нас есть, но он пока что совсем пустой, в нем нет ни строчки исходного текста. Исправить эту ситуацию можно двумя способами: либо мы создаем в проекте новый файл, либо добавляем в него уже имеющийся, например, из ранее созданного проекта. Чтобы добавить файл к проекту, открываем раздел **Project** (Проект) главного меню и в нем выбираем нужный нам пункт: **Add New Item…** (Добавить новый элемент…), если создаем новый элемент, или **Add Existing Item…** (Добавить существующий элемент…), если хотим добавить уже имеющийся. В первом случае откроется новое окно (рис. 1.9), в котором в разделе **Categories** (Категории элементов) мы выбираем пункт **Code** (Текст программы) и затем указываем, какой конкретно файл

нам нужен: с исходным текстом (.cpp) или заголовочный (.h). После всего этого вводим имя нового файла в поле **Name** (Наименование) и щелкаем по кнопке **Add** (Добавить). Во втором случае откроется окно выбора файла, где нужно выбрать файл, который мы хотим добавить в проект.

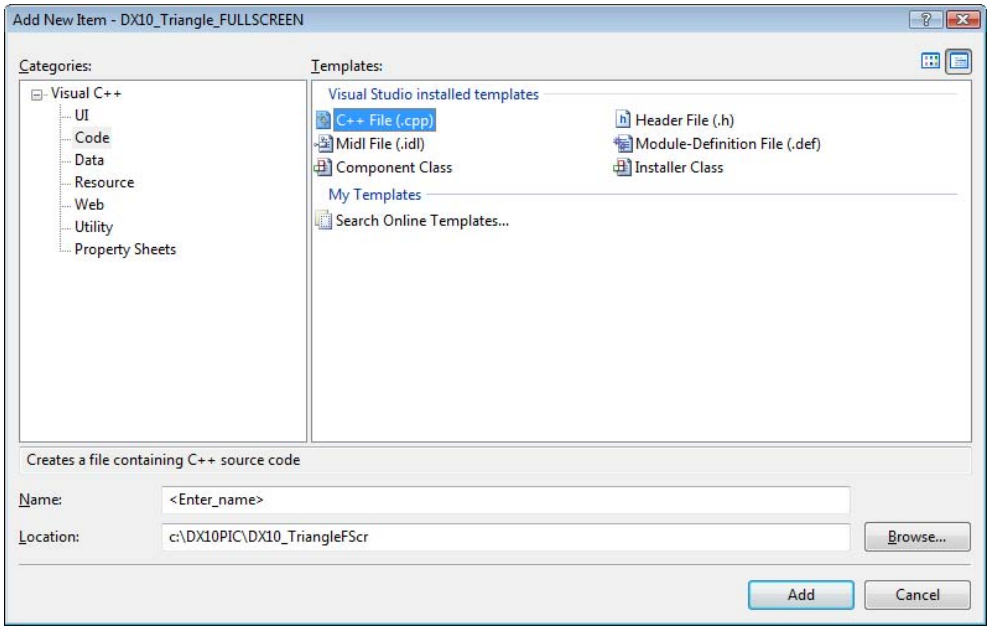

Рис. 1.9. Добавление в проект нового элемента

Настройка путей для компиляции

Если при установке DirectX SDK обнаруживает наличие на компьютере Visual Studio, то пути для всех необходимых файлов программа установки должна добавить автоматически. В случае если этого по каким-то причинам не произошло, пути к файлам необходимо ввести вручную, иначе компиляция программ, использующих DirectX, будет невозможна.

Предположим, мы установили DirectX SDK в папку C:\DXSDK. Как добавить пути в этом случае? Открываем пункт меню **Tools | Options…** (Сервис | Настройки), в открывшемся окне (рис. 1.10) в списке выбираем раздел **Projects and Solutions** (Проекты и решения), в этом разделе нас интересует пункт **VC++ Directories** (Директории VC++). В выпадающем списке **Show directories for** (Отобразить директории для) выбираем **Include files** (Включаемые файлы) и добавляем в список новую строку (щелкаем мышью на кнопке со значком создания новой папки), в которую заносим путь C:\DXSDK\Include.

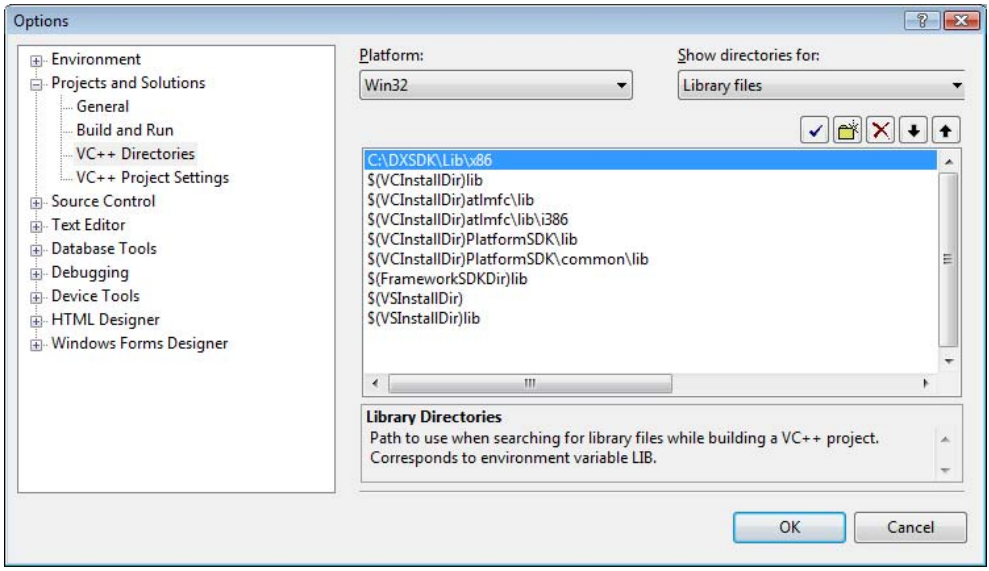

Рис. 1.10. Настройка к путям файлов для компиляции в Visual Studio

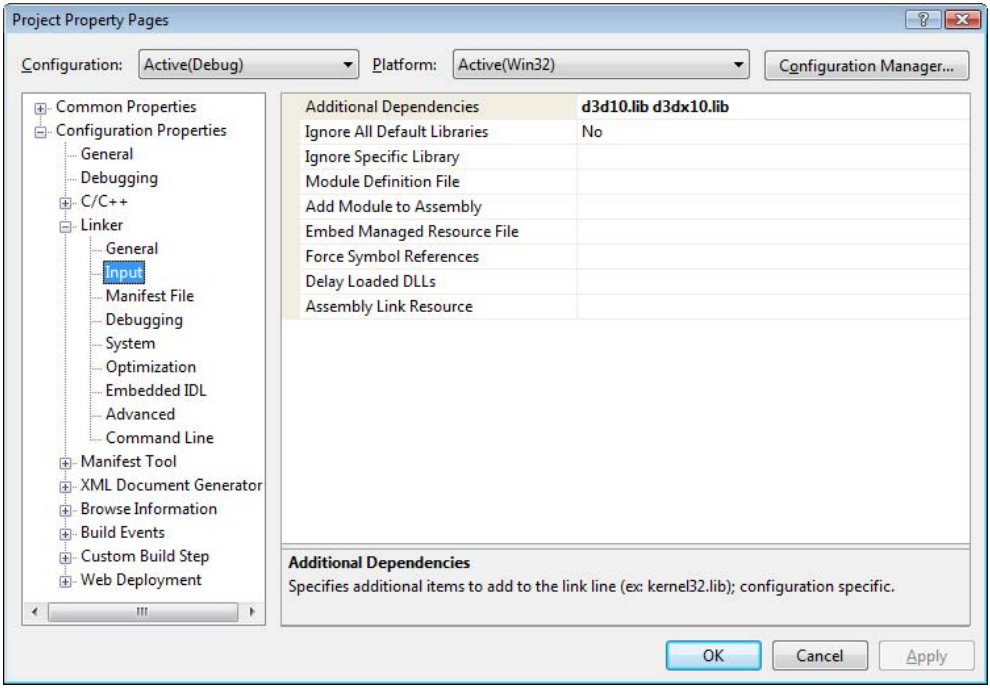

Рис. 1.11. Библиотеки для компиляции программ, использующих Direct3D10

Переключаем список в режим отображения директорий для **Library files** (Библиотечные файлы), добавляем новую строку и пишем в нее следующий путь: C:\DXSDK\Lib\x86, если на компьютере установлен 32-разрядный процессор, и C:\DXSDK\Lib\x64, если процессор 64-разрядный.

Для успешной компиляции программ, использующих Direct3D 10, нужно также предписать использование библиотек d3d10.lib и d3dx10d.lib в свойствах проекта. Сделать это можно таким образом: в главном меню выбираем **Project | Properties** (Проект | Свойства) и в подразделе **Input** (Входные данные) раздела **Linker** (Компоновщик) добавляем в поле **"Additional dependencies"** ("Дополнительные библиотеки") имена файлов библиотек d3d10.lib и d3dx10d.lib (рис. 1.11).

Компиляция и запуск программ

Созданный проект скомпилировать и запустить пока не удастся: мы создали его пустым. Но нужно иметь в виду, что компиляция и запуск программы на выполнение выполняется с помощью кнопки со значком "воспроизведение" (рис. 1.12).

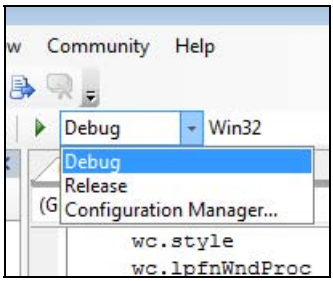

Рис. 1.12. Запуск программы в Visual Studio

Еще, пожалуй, стоит упомянуть, что при отладке и тестовых запусках проекта следует устанавливать в выпадающем списке рядом с кнопкой значение **Debug** (Отладка). Он установлен по умолчанию. При компиляции готовой отлаженной программы "на выпуск" следует установить значение **Release** (Выпуск).

Рабочее место мы подготовили, какой-то минимум информации по работе со средой разработки получили. Для составления программ знаний пока не хватает, но я все постараюсь объяснить по пути. Итак, отправляемся дальше.

Глава 2

Первые шаги

Мы уже умеем создавать новые проекты и компилировать программы в Visual Studio, однако для того, чтобы начать писать программы под Windows, этого еще не достаточно. В программе для Windows должны присутствовать некоторые обязательные элементы, без которых она будет неработоспособна. Чтобы писать программы под DirectX, мы должны уметь создавать простейшее приложение Windows, на "поверхность" окна которого и будет выводиться графика средствами Direct3D. Но прежде чем мы познакомим со структурой приложения Windows, нам необходимо хотя бы в общих чертах представить, как организована операционная система.

Многозадачная операционная система Windows

Начнем с самого начала, с понятия операционной системы. Операционная система (ОС) — это программа, обеспечивающая среду для выполнения приложений и предоставления им доступа к ресурсам компьютера. Под ресурсами понимается оперативная память, процессорное время и вообще любые устройства, которые может использовать приложение во время своей работы.

Что означает слово "многозадачная"? Это значит, что система может обеспечивать одновременное выполнение нескольких приложений. Строго говоря, настоящее параллельное выполнение задач возможно только на процессорах с несколькими ядрами (двух-, четырехъядерные процессоры). При работе на процессорах с одним ядром приложения выполняются по очереди, каждое работает маленький промежуток времени. Промежуток этот носит название кванта времени. За счет того, что длительность кванта времени невелика и достигается иллюзия параллельного выполнения.

В однозадачной системе MS-DOS выполнение программы называлось также "передать программе управление". Это означало, что во время работы программы она монопольно использует все ресурсы компьютера, а чтобы запустить еще одну программу, нужно сначала завершить работу той, которая в данный момент работает. Освобождение ресурсов и передача управления обратно операционной системе происходило по инициативе самой программы. Бывало даже, что если, например, в игровой программе не был предусмотрен выход обратно в систему, то не оставалось ничего другого, как перезагружать компьютер, чтобы завершить ее работу.

> ***Замечание***
>
> На самом деле некоторая возможность параллельного выполнения программ в MS-DOS все же существовала. Имелись так называемые резидентные программы, которые после запуска оставались в оперативной памяти и возвращали управление системе. После этого они могли выполнять какие-либо действия, активизируясь по нажатию "горячей клавиши", даже если одновременно с этим выполнялась какая-то другая программа. С помощью таких программ реализовывалось переключение ввода русских и латинских символов, вывод содержимого графического экрана на принтер либо запись его в файл и т. д.

В многозадачной системе Windows ситуация совершенно другая, приложения выполняются параллельно, одновременно могут работать текстовый редактор, проигрыватель музыкальных файлов, закачиваться данные из Интернета и, вдобавок к этому, выполняться какой-нибудь длительный расчет. Как все это возможно? Как им поделить между собой экран для вывода результатов и разобраться с тем, какому из них адресованы вводимые с клавиатуры символы и которое из них должно реагировать на щелчки мышью? Решение базируется на том, что приложения Windows построены по принципу реакции на события (event-driven application). С точки зрения операционной системы, события представляют собой нажатия на клавиши клавиатуры, движение курсора мыши, а также перемещение, изменение размеров окна приложения и т. д. Приложение не загружает процессор, останавливая всю работу на время ожидания ввода данных, как это происходило в однозадачной системе.

Но как приложение "узнает", например, что ему именно сейчас нужно принимать символы с клавиатуры? Оно получает сообщение от операционной системы. Да, именно так, операционная система Windows управляет приложениями с помощью сообщений. Посылать сообщения могут также и приложения: друг другу и даже сами себе. С каждым окном связана специальная структура — очередь сообщений. В эту структуру и записываются сообщения, предназначенные данному окну. Приложение постоянно проверяет очередь на наличие в ней сообщений. При получении сообщения оно извлекается из очереди и передается на обработку оконной процедуре, специальной

функции того окна, которому предназначено сообщение. Какие действия предпринимать при поступлении того или иного сообщения, определяется именно в этой функции. Фактически в ней содержится оператор `switch`, в качестве параметра которого используется идентификатор поступившего сообщения.

> ***Замечание***
>
> Если быть точным, то очередь сообщений связана с каждым потоком (thread) приложения. Но мы пока будем считать, что она связана с окном, исключительно для простоты объяснения.

Работу оконной процедуры можно понаблюдать следующим образом. Возьмем стандартную программу "Блокнот" и напишем в ней произвольную строку текста. Теперь попробуем выйти из программы, щелкнув мышью на кнопке закрытия окна. Что мы видим? Программа сообщила, что у нас имеется несохраненный текст, и спросила, требуется ли его сохранить. Наша попытка закрыть окно вызвала появление в очереди сообщений программы сообщения `WM_CLOSE` (запрос на закрытие окна). Оно было передано на обработку в оконную процедуру, где и сработала соответствующая ветвь оператора `switch`, в которой предусмотрен запрос на сохранение текста.

Минимальное приложение Windows

Итак, как же должна выглядеть программа под Windows? Структура приложения Windows показана на рис. 2.1.

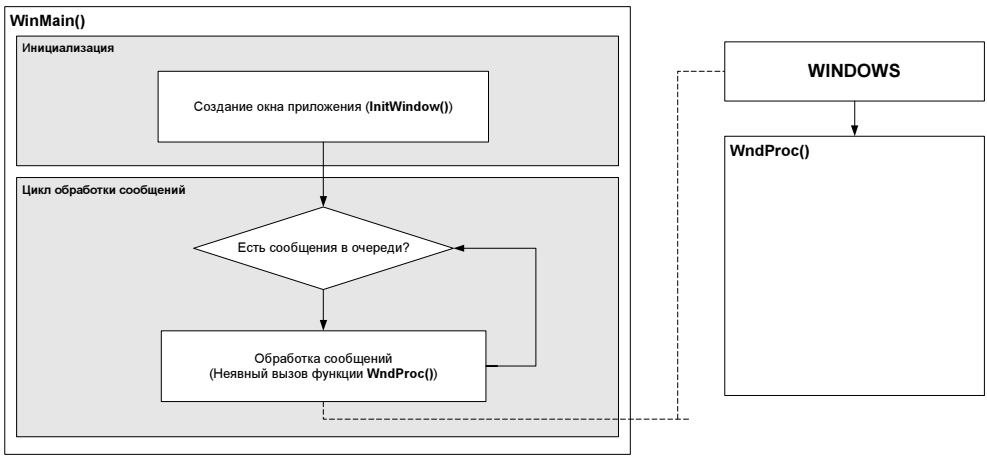

Рис. 2.1. Структура приложения Windows