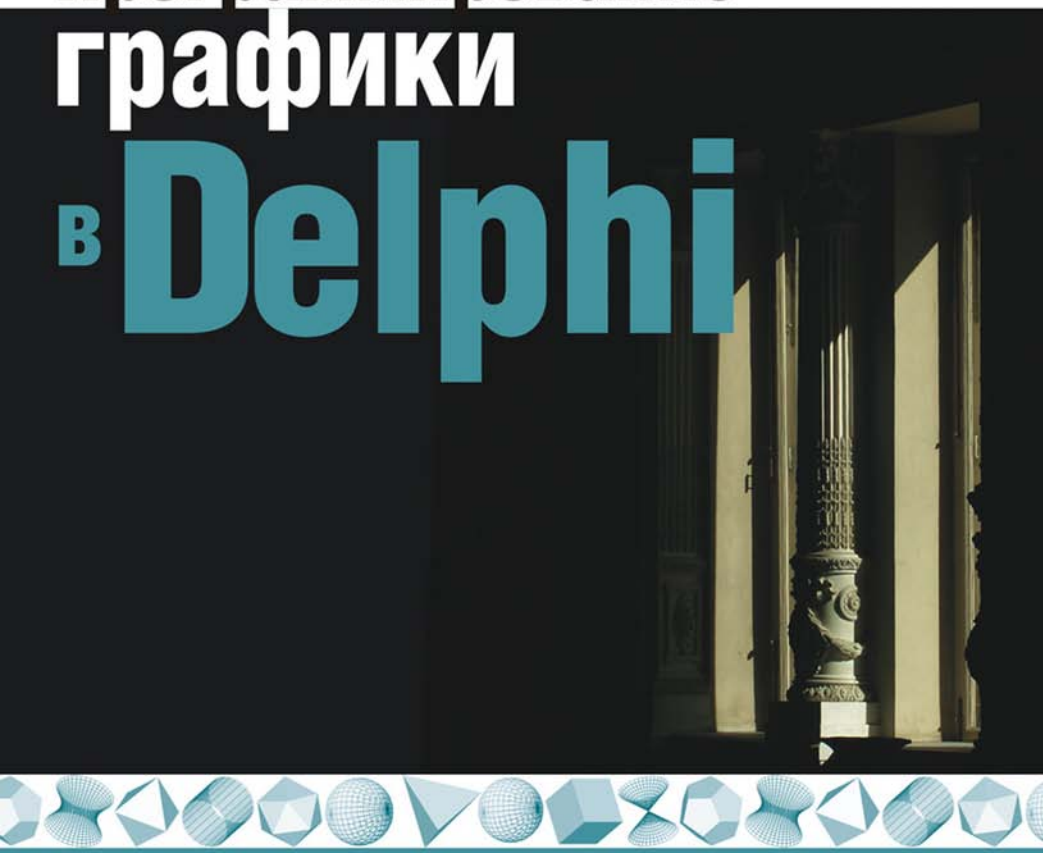
Николай Тюкачев,
Игорь Илларионов,
Виктор Хлебостроев

Программирование графики в Delphi



+CD

- Цветовые модели и форматы графических файлов
- Средства для разработки мультимедийных приложений
- Компоненты для работы с деловой графикой
- Методы построения кривых в задачах интерполяции, сглаживания, аппроксимации, методы Эрмита, Безье и B-сплайнов
- Создание растровых и векторных графических редакторов
- Работа с трехмерными изображениями
- Задачи триангуляции, проектирование и реализация ГИС

bhv

Николай Тюкачёв,
Игорь Илларионов,
Виктор Хлебостроев

Программирование графики в Delphi

Санкт-Петербург
«БХВ-Петербург»
2008

УДК 681.3.068+800.92Delphi
ББК 32.973.26-018.1
Т98

Тюкачѐв, Н. А.

Т98 Программирование графики в Delphi / Н. А. Тюкачѐв, И. В. Илларионов, В. Г. Хлебостроев. — СПб.: БХВ-Петербург, 2008. — 784 с.: ил. + CD-ROM

ISBN 978-5-9775-0253-5

Книга написана на базе курса лекций, читаемых авторами. Рассмотрены основные классы и функции среды Delphi, которые используются для создания графических и мультимедийных приложений. Описаны цветовые модели, основные форматы графических файлов, а также методы построения кривых в задачах интерполяции, сглаживания, аппроксимации, методы Эрмита, Безье и B-сплайнов. Приведены алгоритмы триангуляции поверхностей в трехмерном пространстве. На конкретных примерах показан весь процесс разработки основных типов приложений — пакетов деловой графики, работы с трехмерными объектами, растровых и векторных графических редакторов, геоинформационных систем. Каждый раздел сопровождается задачами различной сложности для самостоятельного решения. На прилагаемом компакт-диске представлено более 30 проектов, описанных в книге.

Для программистов

УДК 681.3.068+800.92Delphi
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Наталья Баскакова*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 27.06.08.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 63,21.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.003650.04.08
от 14.04.2008 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 978-5-9775-0253-5

Оглавление

Введение

Эта книга написана на основе лекций для студентов, обучающихся по специальности "Информационные технологии", и не является введением в визуальную среду программирования Delphi. Предполагается, что читатель уже преодолел несколько первых ступеней, знаком с объектно-ориентированным программированием, со средой Delphi и умеет создавать в ней простые проекты. Это позволит нам не останавливаться на подробном описании последовательности нажатия клавиш и больше внимания уделять принципиальным вопросам разработки проектов.

Для читателей, делающих первую попытку открыть для себя мир Delphi, опубликовано большое количество прекрасных книг. Все книги этого направления можно разбить на три больших группы. Первая группа — с условным названием "Математические основы машинной графики", вторая — "Описание и использование библиотек типа OpenGL или Direct", третья — "Описание методов рисования Windows".

Из числа книг первой группы наиболее полно математические основы графики изложены в книге Д. Роджерса и Д. Адамса "Математические основы машинной графики" [98]. В ней, а также в книге Д. Роджерса "Алгоритмические основы машинной графики" [97], подробно описаны модели и алгоритмы растровой графики:

- алгоритмы Брезенхейма;
- модели освещения, тени, прозрачности, фактуры;
- двумерные и трехмерные преобразования и проекции;
- плоские кривые, пространственные кривые: кубические сплайны, кривые Безье, B-сплайны;
- поверхности, квадратичные поверхности, бикубические поверхности, поверхности Безье, B-сплайн поверхности.

Книга Л. Аммерала "Принципы программирования в машинной графике" [58] содержит меньше материала, чем у Д. Роджерса, но в ней дается введение в язык Си. Майкл Абраш в книге "Программирование графики. Таинства" [56] излагает основы графики еще под MS-DOS: алгоритмы Брезенхейма для отрезка и окружности, заполнение многоугольника, устранение ступенчатости.

В книге Е. В. Шикина и А. В. Борескова "Компьютерная графика. Динамика, реалистические изображения" [120] освещаются те же вопросы, что и у Д. Роджерса и Дж. Адамса, и, кроме того, описывается пакет 3D Studio.

Книга Е. А. Никулина [87] содержит подробное изложение геометрических и алгоритмических основ компьютерной графики — математических моделей графических элементов на плоскости и в пространстве, методов геометрических преобразований и визуализации, способов моделирования оптических объектов. Однако все рассматривающиеся в ней алгоритмы описаны словесно либо в виде блок-схем.

Список второй группы книг может быть открыт книгой Ю. В. Тихомирова "Программирование трехмерной графики в Visual C++" [105], в которой освещаются возможности применения в приложениях на языке Си библиотеки OpenGL. Подробно описаны:

- вершины, примитивы;
- спецэффекты: глубина, туман, цвет;
- геометрические преобразования;
- реалистические изображения: свойства материала, источники света, прозрачность, тени, текстура.

М. В. Краснов в книге "OpenGL в Delphi" [80] переложил для Delphi тот же материал что и Ю. В. Тихомиров, добавив описание графического редактора, использующего OpenGL.

Наиболее ярким представителем третьей группы является книга Юань Феня "Программирование графики для Windows" [125]. Ч. Петзолд в главе 4 книги "Программирование для Windows 95" [92] также описывает графические API-функции Windows, а А. Я. Архангельский в одной из глав своей книги "Программирование в Delphi 6" [60] перечисляет свойства и методы канвы. Впрочем, наиболее полно API-функции, свойства и методы канвы описаны в справочной системе Delphi.

Таким образом, между математическими основами графики и готовыми графическими библиотеками типа OpenGL/Direct, в которых эти основные алгоритмы реализованы, образовался разрыв. Назначение нашей книги — частично заполнить этот разрыв. Средством для заполнения является описание многочисленных приложений.

Мы попытались объединить различные подходы, собрав и систематизировав в *главах 1–3* весь доступный и необходимый нам справочный материал о графических возможностях Delphi, а в *главах 4–14* представив многочисленные примеры проектов. Отметим, что при разработке проектов иногда существенную помощь оказывали интернет-ресурсы. В первую очередь это относится к проблемам моделей цветов, рисования линий произвольным стилем, алгоритмам Брезенхейма.

Мы осознаем, что чтение первых трех глав достаточно утомительное занятие, но без знания (хотя бы поверхностного) возможностей графических классов сложно приступать к разработке самостоятельных графических проектов. После беглого чтения этих глав можно заняться примерами, возвращаясь, по мере необходимости, к справочному материалу.

Delphi опирается на мощный набор классов, предназначенных для работы с графикой. Значительная часть этих классов содержится в модуле `Graphics`. Описанию модуля `Graphics` посвящена *глава 2.* В ней описаны классы инструментов `TFont`, `TPen`, `TBrush` и класс канвы `TCanvas`. Там же приводится описание всех методов рисования на канве.

В *главе 2* обсуждаются также различные цветовые модели: RGB, CMY, CMYK, HSB, Lab, HSV, HLS. Поскольку все цветовые модели являются математическими,

они легко конвертируются одна в другую по простым формулам. Такие конверторы встроены во многие графические программы. Примеры процедур, которые можно использовать для перехода из одной модели в другую, приведены в *разд. 2.4*.

Можно нарисовать сколь угодно сложную картинку, вставляя в код программы вызов методов с заданными значениями координат точек на канве. Но приложения должны быть максимально независимы от конкретных данных. Программа, описанная в *разд. 2.10*, рисует различные рисунки на канве. Для реализации этой идеи нужно записать в текстовый файл данные о рисунке в специальном формате.

Метод канвы `TextOut` позволяет печатать только горизонтально расположенную строку. В *разд. 2.11* описан простой проект, позволяющий печатать строку под любым углом `Angle` в десятых долях градуса.

Из приложения, как это показано в *разд. 2.12*, можно получить доступ не только к канве компонента, имеющего опубликованное свойство `Canvas`, но и к канве любого компонента или ко всему экрану.

Среда программирования Delphi поддерживает некоторые форматы графических файлов. Во-первых, это "родной" формат Windows — формат растровых изображений `BMP` (класс `TBitmap`). Во-вторых, форматы метафайлов `EMF` и `WMF` (класс `TMetaflle`). В-третьих, формат пиктограмм `ICO` (класс `TIcon`). В четвертой версии Delphi добавлен класс, позволяющий работать со сжатыми файлами в формате `JPG` (класс `TJPEGImage`).

В *главе 3* описан абстрактный класс `TGraphic`, являющийся родительским для этих четырех видов изображений, и сами классы `TBitmap`, `TMetaflle`, `TIcon`.

В *главе 4* рассмотрены средства, которые предоставляет среда программирования Delphi для разработки мультимедийных приложений. Сначала приводятся примеры использования компонентов Animate и MediaPlayer, затем продемонстрировано, как можно использовать стандартные функции MCI и WinAPI для решения задач воспроизведения и записи звука, работы с мультимедийными файлами. Подробно рассмотрена структура wave-файла и ее применение для разработки приложений, оперирующих низкоуровневыми структурами.

Программы построения диаграмм составляют неотъемлемую часть офисных программ. Delphi включает в себя несколько специальных компонентов для работы с деловой графикой. Описание свойств и методов этих компонентов представлено в *главе 5*. Подробно рассмотрены методы и свойства компонента Chart. Показаны возможности создания приложений, как с использованием прямого обращения к методам компонента в тексте программы, так и с применением собственных средств компонента.

В *главе 6* рассмотрены вопросы моделирования изображений: преобразование системы координат, проецирование. В этой же главе обсуждаются методы построения кривых линий в задачах интерполяции, сглаживания, аппроксимации, методы Эрмита, Безье и B-сплайнов. Затронуты вопросы математического моделирования освещения и цвета.

В *главе 7* приводятся примеры проектов, использующих методы и свойства графических классов: просмотр файлов форматов BMP, ICO, WMF и JPG, вывод изображений на канву принтера, использование мультимедийных ресурсов, деформация изображений в формате BMP, реализация простейшего графического редактора, проектирование плоских схем.

В этой же главе обсуждаются способы создания мультипликации. Изменение изображений со временем возможно двумя способами. Во-первых, можно воспользоваться процедурой временной задержки Sleep из модуля SysUtils, во-вторых, воспользоваться компонентом Timer, имеющим обработчик события OnTimer. Возможности, связанные с использованием процедуры Sleep, демонстрируются на примере сортировки элементов массива, а использование компонента Timer — на примере морфинга одного предмета (стула) в другой (стол) и в проекте "Движение велосипеда".

Свойство Style определяет стиль линии, рисуемой пером. Во многих случаях узкий диапазон возможных значений этого свойства не устраивает программистов. Для применения стиля, заданного пользователем, необходимо использовать API-процедуру LineDDA, определенную в модуле Windows. Проект, описанный в *разд. 7.4*, показывает, как это реализовать.

Далее обсуждается работа с принтером, для использования которого должен быть подключен модуль Printers. С принтером можно работать, как с текстовым файлом, или как с полотном.

В многочисленных графических редакторах используются различные фильтры, изменяющие первоначальные изображения. Некоторые из этих фильтров меняют геометрию изображения. В *разд. 7.5* описывается проект, решающий задачу деформации изображения из прямоугольной области (0,0), (I2,J2) в произвольный четырехугольник (u0,v0), (u1,v1), (u2,v2), (u3,v3) двумя способами.

В *разд. 7.6* описан проект, реализующий многооконный растровый редактор типа редактора PaintBrush, который позволяет рисовать линию, отрезок прямой, прямоугольник, выделять и перемещать прямоугольник с частью изображения, вставлять изображение.

В *разд. 7.7–7.9* обсуждаются задачи проектирования плоских схем. К этому классу задач относятся задачи построения сложных диаграмм и электрических схем, структурных схем соединения компьютеров в сети и структурных схем программ, проектирования реляционных баз данных, задачи макетирования векторных карт и множество других проектов, в которых требуется установить связи между объектами, изображенными на экране в виде условных обозначений. Программы, предназначенные для проектирования плоских схем, должны, прежде всего, уметь вставлять в проект условные обозначения объектов, устанавливать связи между объектами, перемещать объекты и связи, удалять объекты и связи, менять свойства объектов.

В качестве примера решения такой задачи предлагается проект построения изображения произвольного графа с возможностями добавлять, удалять и перемещать узлы, создавать дуги, а также выбирать вид представления узлов.

Вторым примером подобной задачи является разработка проекта газификации домов. Предполагается, что проект дома готов и отсканирован в формате `BMP` или `JPG`. Программа позволяет подготовить эскизы газификации дома, трассы трубы от газопровода до дома, расставить газовое оборудование на эскизе газификации дома (есть возможность выбора газового оборудования из списка). Имеется возможность редактирования (перемещения, удаления) оборудования и труб. При построении эскиза трассы трубы от газопровода до дома можно учитывать профиль поверхности земли, расставить бетонные трубы и показать глубину их прокладки. После завершения редактирования программа позволяет распечатать подготовленные эскизы и список необходимого оборудования.

Сохранение графической информации существенно зависит от характера изображения. Для издательских систем и рекламных роликов характерно использование растровой графики, при этом в памяти хранится информация о цвете каждой точки. Такое представление информации, естественно, требует большого объема оперативной памяти и дискового пространства, а также эффективных алгоритмов сжатия. В системах автоматизированного проектирования таких, например, как AutoCAD, информация сохраняется в векторном виде. Так, в векторном представлении, для хранения отрезка прямой линии можно записать в файл координаты его начала и конца, информацию о цвете и стиле линии. Информация о плоских схемах, в том числе о графах, также хранится в векторном виде в одном нетипизированном файле.

В *главе 8* описывается векторный редактор типа редактора Corel DRAW, реализация которого сложнее растрового. Поэтому в проекте реализована только часть функциональных возможностей, а именно: построение изображений прямоугольника, эллипса, кривой Безье и фрагмента текста; перемещение, масштабирование и поворот этих объектов.

В *главе 9* продемонстрировано использование графических возможностей Delphi для решения некоторых геометрических задач. Описаны проекты построения графиков функций в двумерном и трехмерном пространствах, интерполяция функций (интерполяционный многочлен Лагранжа, метод наименьших квадратов, кубические сплайны, кривые Безье).

В *разд. 9.1* обсуждаются дополнительные возможности реализации задачи построения графика функции одной переменной. В проекте на инструментальной форме помещены компоненты, позволяющие менять размеры окна "на бумаге", рисовать или не рисовать координатную сетку и строить график функции в обычной или логарифмической системе координат.

Далее рассматривается задача построения в трехмерном пространстве поверхности, описываемой функцией $z = \mathrm{f}(x,y)$ с возможностью управления вращением системы координат указателем мыши.

На практике часто встречаются задачи построения интерполирующих и аппроксимирующих функций для заданного на плоскости множества точек (x_i,y_i), $i = 0, 1,..., n$. В *разд. 9.3* рассмотрены четыре способа решения этой задачи:

интерполяционный многочлен Лагранжа, метод наименьших квадратов, интерполяция кубическими сплайнами и интерполяция кривыми Безье.

В *разд. 9.4* демонстрируется простой проект для рисования семи трансцендентных кривых, заданных параметрически и зависящих от шести параметров.

В *разд. 9.5* рассматривается задача построения графика функции по выражению, заданному в виде строки, т. е. конструкции из переменных, констант, бинарных операций и функций.

Глава 10 посвящена задаче визуального создания сложных отчетов, которые могут включать строковые надписи, таблицы и графики. Надписи могут быть статическими или вычисляться в процессе заполнения отчета. Таблицы могут быть горизонтальными или вертикальными. Они могут содержать статические заголовки колонок и поля, вычисляемые в процессе заполнения отчета. Результатом работы генератора отчетов является набор файлов, в которых содержатся следующие данные:

- общие данные об отчете (название отчета, поля на листе бумаги, шаг сетки);
- данные о строковых надписях (положение на листе бумаги, ширина, высота, шрифт, отметка о статичности или номер функции);
- данные о таблицах (положение на листе бумаги, ширина, высота, шрифт, номер таблицы);
- данные о строках таблиц (название, номер таблицы, отметка о статичности или номер функции).

Генератор отчетов должен уметь создавать и удалять объекты (надписи и таблицы), настраивать свойства объектов, перемещать объекты, изменять общие параметры отчета.

В *главе 11* "Геометрия трехмерных тел" собраны проекты построения трехмерных изображений с учетом перспективы и тени, построения изображения трехмерного тела по трем проекциям, рисования полутонами, бинарные операции над множествами.

Первый проект — "Платоновы тела", предназначен для рисования трех Платоновых тел: тетраэдра, гексаэдра и октаэдра и состоит из двух форм: главной формы и формы для настройки параметров. В проекте реализованы следующие функции: выбор одного из тел; вращение системы координат или тела; рисование тела гранями или ребрами; изменение размера тела; изменение масштаба; окрашивание граней полутонами; рисование тени на плоскости $z = const$; изменение положения точек схода на осях OZ и OY.

Второй проект — "Квадратичные поверхности" предназначен для рисования 10 поверхностей (эллипсоида, различных типов гиперболоидов, параболоидов и цилиндров). В общем случае уравнения поверхностей, заданных полиномами второй степени, имеют вид $Ax^2 + By^2 + Cz^2 + Dxy + Exz + Fyz + Gx + Hy + Iz + K = 0$. Но, после приведения этого уравнения к каноническому виду, число различных квадратичных поверхностей существенно сокращается. В этом проекте, как и в

проекте "Платоновы тела", реализовано стереографическое изображение многогранников.

В начертательной геометрии и черчении одна из основных задач состоит в том, что необходимо спроецировать трехмерное тело на координатные плоскости OX, OY, OZ. Обратная задача, обсуждаемая в третьем проекте *главы 11*, более сложна: по трем (или даже по двум) проекциям нужно восстановить объемное изображение тела. Предлагается простой алгоритм решения этой задачи. Заметим, что в общем случае задача не имеет однозначного решения.

Операции объединения, вычитания, пересечения двумерных и трехмерных тел используются при проектировании в архитектуре и машиностроении. В общем случае эти задачи являются достаточно сложными. Однако бинарная нумерация операций над множествами, используемая в четвертом проекте, позволяет сформулировать общий алгоритм для ЭВМ.

В *главе 12* сначала обсуждаются различные способы моделирования трехмерных тел. Выбор модели трехмерных тел определяет способ решения таких проблем, как выделение видимых частей объекта и назначение цвета каждому элементу объекта. Моделирование может быть реализовано разложением объекта на элементы, трехмерные тела в этом случае будут представляться объединением некоторых базовых блоков. Тип базовых объектов определяет различные методы моделирования этим способом. Самый простой способ описания тела заключается в сопоставлении с ним набора одинаковых кубов (вокселей). Можно уменьшить объем памяти, необходимой для хранения информации об объекте, если хранить информацию только о блоках, относящихся к объекту. В этом случае число элементов пропорционально площади его поверхности, т. е. пропорционально N^2. С этой целью можно использовать октарные или бинарные деревья. В пространственной геометрии объект может задаваться набором примитивов и операций над ними. Такое представление называется твердотельным моделированием. Примитивы являются "строительными блоками" объекта. Под операциями понимаются булевы операции над примитивами, а также геометрические преобразования, такие, как передвижение, поворот, изменение размеров. В отличие от ранее перечисленных моделей, поверхностное представление определяет сплошное тело неявно, путем описания ограничивающей его поверхности. Наиболее простой способ поверхностного описания тел — это описание его набором плоских граней.

В той же главе описан проект, предназначенный для визуальной работы с многогранниками. Он позволяет: читать и записывать данные; перемещать окна просмотра и тело; масштабировать изображение на любой координатной проекции; перемещать вершины на любой проекции; выделять грань и, добавляя к ней новую вершину, увеличивать число граней. Предполагается, что все грани представляют собой треугольники.

В геоинформационных системах (ГИС) существует проблема описания двумерных и трехмерных тел (многоугольников на плоскости и многогранников в пространстве), топологически связанных между собой. Это означает, что граничные вершины

и ребра многоугольников (на плоскости) или граничные вершины, ребра и грани (в трехмерном пространстве) должны быть общими для соседних тел. В этом случае изменение координат общей вершины двух соседних тел не приводит к появлению "ничейной" области между телами. В *разд. 12.3* предлагается структура данных, описывающая систему топологически связанных тел для трехмерного пространства. Стандартное описание системы тел в трехмерном пространстве выглядит следующим образом: проект состоит из массива тел-многогранников. Для каждого тела определены свои независимые массивы вершин, ребер и граней, опирающихся на вершины. В предлагаемом проекте для всех тел используются общие массивы вершин, ребер и граней.

В *главе 13* описывается работа с библиотекой OpenGL, основные команды, а также очень простой проект, демонстрирующий использование плоских примитивов OpenGL. А для демонстрации работы с трехмерными объектами предназначен второй проект "3D OpenGL". Проект позволяет рисовать многогранники модуля `DGLUT` или иные многогранники, квадратичные поверхности, заданные параметрическими уравнениями, поверхность, заданную таблично, различные фигуры, данные о которых записываются в текстовых файлах программами, использующими DirectX, и накладывать на эти поверхности текстуру из файлов BMP или JPG.

В заключение *главы 13* обсуждается проект "Редактор многогранников", в котором реализован ряд функций манипулирования простейшими многогранниками — пирамидами и параллелепипедами. Проект позволяет добавлять тела или группы тел, менять размеры выбранного тела, менять положение выбранного тела или группы тел, менять видимость тел или группы тел, изменять цвет тел, изменять цвет и положение источника света, поворачивать сцену.

Глава 14 посвящена некоторым задачам триангуляции, многочисленные варианты которых возникают при разработке геоинформационных систем. В частности, рассматриваются следующие задачи триангуляции: триангуляция по заданным точкам; триангуляция слоев; сглаживание поверхности; изолинии.

Здесь же рассматривается еще одна задача, связанная с триангуляцией — построение триангуляции невыпуклого многоугольника. Эта задача порождает определенные проблемы при рисовании, так как библиотека OpenGL, например, не может изображать невыпуклые многоугольники.

В заключение главы рассматривается часто возникающая задача построения изолиний и рисования топографической карты поверхности. Эта поверхность может быть дневной поверхностью земли или поверхностью слоя породы или значением объемных геофизических измерений, сделанных на определенной глубине.

В тексте приведены многочисленные примеры программ и процедур. В книге приведены условия задач, решение которых достаточно просто реализуется в среде Delphi.

Авторы с благодарностью примут все замечания и пожелания в адрес предлагаемой книги (**E-mail: nik@cs.vsu.ru**).

Глава 1

Рисование в Delphi

1.1. Моделирование цветов

Мы видим мир цветным, и это одна из самых больших биологических загадок. Человеческий глаз может воспринимать цветовые волны с длинами от 380 нм до 780 нм. Это лишь незначительный диапазон спектра электромагнитных волн. Зрительные пигменты глаза состоят из колбочек трех типов, чувствительных к синему, зеленому и красному цвету. Эффективность поглощения световых волн существенно различается для различных типов колбочек. Особенно хорошо воспринимается зеленый свет, красный свет воспринимается несколько хуже, а чувствительность глаза к синему свету еще ниже. Многочисленные психологические тесты позволяют считать, что яркость можно вычислить по формуле [124]:

Яркость= 0,59×Зеленый + 0,3×Красный + 0,11×Синий

Цвет представляет собой индивидуальное ощущение, не позволяющее судить о его спектральном составе. Для обработки изображений на современной технике такая субъективность нежелательна. Именно для цели обработки изображений были разработаны математические методы точного описания цвета, называемые цветовыми моделями. Одна из них носит название `RGB` (`Red`, `Green`, `Blue`). В рамках этой модели предполагается получение цветов смешением красного, зеленого и синего цветов. Смесь синего и зеленого, например, дает голубой цвет, а смесь красного и синего — пурпурный.

Операционные системы, такие как Windows, позволяют устанавливать различные графические режимы:

- 16-цветный режим;
- 256-цветный режим, при котором для хранения цвета каждой точки (пиксела) выделяется один байт;
- режим `High Color`, при котором для хранения цвета каждой точки выделяется два байта, что позволяет использовать до 65000 цветов;
- режим `True Color`, при котором для хранения каждого пиксела выделяется четыре байта (32 бита). Чаще всего из этих 32 битов используется 24, что позволяет изменять доли красного, синего и зеленого цветов в диапазоне от 0 до 255.

Математически систему RGB можно представить в виде куба, каждая точка которого однозначно определяется координатами R, G и B. Так как в системе RGB цвета определяются смешением основных цветов, то она особенно удобна для устройств, излучающих цветовые волны (видеомонитор, цветной телевизор).

Цветовая система RGB очень проста, но при ее использовании возникают две проблемы. Первая — зависимость от аппаратуры, вторая — невозможность получить все цвета смешением красного, зеленого и синего цветов.

Другой, часто используемой цветовой системой, является модель CMYK. Название этой модели образовано из первых букв названий цветов Cyan, Magenta, Yellow (голубой, пурпурный, желтый) и последней буквы слова blacK (черный) (используется буква K, так как буква B уже задействована в названии цветовой модели RGB). Данная модель предназначена для описания цвета отражающих поверхностей. Именно эта модель служит теоретической основой цветной печати. В отличие от модели RGB, цвета в модели CMYK получаются не аддитивно (суммированием), а субтрактивно (вычитанием). Например, поверхность голубого цвета отражает синий и зеленый, но поглощает красный, пурпурный поглощает зеленый, а желтый поглощает синий цвет.

Применяемые на практике цветные краски далеко не идеальны, и смешением всех красок, как правило, не удается получить черный цвет. Поэтому для повышения контрастности применяется чисто черный краситель.

1.2. Полотно компонентов

Рисовать в Delphi проще, чем на бумаге. При разработке проекта в вашем распоряжении находятся полотно (свойство Canvas), карандаш (свойство Pen), кисть (свойство Brush) и некоторое количество примитивов (линий, прямоугольников, эллипсов и т. д.). Правда, опубликованным свойством Canvas обладают далеко не все компоненты. В частности, этим свойством обладают компоненты форма (класс TForm), таблица (класс TStringGrid), растровые изображения (класс TImage), принтер (класс TPrinter). У карандаша и кисти можно менять цвет (Color) и стиль (Style). Этот набор инструментов позволяет создавать достаточно сложные рисунки математического и инженерного содержания. Кроме того, Delphi позволяет использовать многие ресурсы Windows: графические файлы, фильмы и звуковые файлы.

Полотно — это прямоугольная сетка, состоящая из маленьких квадратов, называемых пикселами (свойство Pixels[X,Y]: TColor). Каждый пиксел имеет свой номер, точнее два номера. Первый номер указывает на горизонтальное расположение пиксела, а второй — на вертикальное. Левый верхний пиксел полотна имеет координаты 0,0 — Pixels[0,0]. Общее количество пикселов по горизонтали доступно через свойство Width, а по вертикали — через свойство Height. Каждый пиксел может быть закрашен любым доступным для Windows цветом. Рисование пиксела на по-

лотне происходит в тот момент, когда мы присваиваем элементу массива `Pixels` номер цвета. Например, присваивание:

```
Image1.Canvas.Pixels[100,100]:= clRed
```

приведет к рисованию красной точки с координатами `100,100`. Мы можем узнать номер цвета любого пиксела обратным действием:

```
Color:= Image1.Canvas.Pixels[100,100].
```

Класс цвета точки `TColor` определен как `longint`:

```
TColor = - $80000000 ..$7FFFFFFF.
```

Четыре байта переменных этого типа содержат информацию о долях синего (B), зеленого (G) и красного (R) цветов и устроены следующим образом: `$00BBGGRR`. Доля каждого цвета может меняться от 0 до 255. Поэтому для рисования красной точки, например, мы должны выбрать цвет с номером `$0000FF`. В Delphi определен набор констант для цветов. Список этих констант можно увидеть в инспекторе объектов или в модуле `Grahics`.

Впрочем, обращение к точкам полотна через двумерный массив `Pixels[X,Y]` — это самый медленный способ рисования. Для более быстрого рисования используют специальные методы канвы `Canvas`: например, `LineTo` — линии, `Arc` — дуги, `RectAngle` — прямоугольники, `TextOut` — вывод текста и т. д.

1.3. Пример использования графики

Для демонстрации возможностей использования графики, рассмотрим простой проект, в котором на полотне компонента `Image1` будем рисовать кривые Лиссажу (рис. 1.1). Проект приведен на компакт-диске (полное содержание которого приведено в *прил. 2*) в папке **Примеры | Глава 1 | Кривые Лиссажу**.

Все помнят из школьного курса физики, что если на вход X и вход Y осциллографа подать синусоидальные сигналы, то на экране появится отрезок прямой, но при некотором сдвиге фаз отрезок расслоится и превратится в эллипс, который превращается в окружность при сдвиге фаз π/2 и одинаковых амплитудах. Это обстоятельство используется для измерения сдвига фаз в любительских радиоизмерениях. В общем виде кривые Лиссажу задаются параметрическими уравнениями:

$$x = A \times \sin(w_x \times t + w_1),$$

$$y = A \times \sin(w_y \times t + w_2)$$

и зависят от четырех параметров: двух частот w_x, w_y и двух фазовых смещений w_1 и w_2. В проекте функции, определяющие кривые Лиссажу, обозначены как `Fx` и `Fy` (листинг 1.1).

Листинг 1.1. Функции, определяющие кривые Лиссажу

```
function TForm1.Fx(t: real): real;
begin
  Fx:=sin(wx*t+w1);
end;
function TForm1.Fy(t: real): real;
begin
  Fy:=sin(wy*t+w2);
end;
```

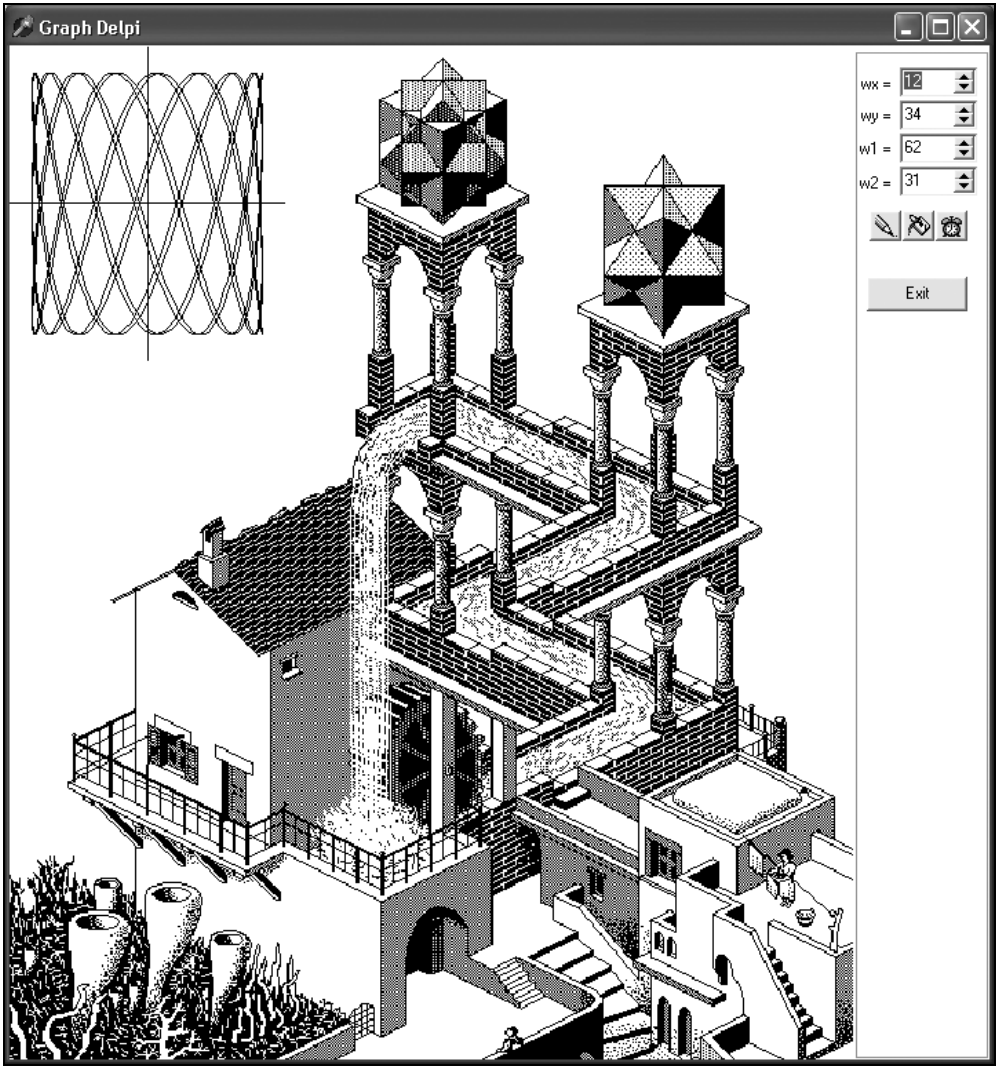


Рис. 1.1. Проект "Кривые Лиссажу" с копией картины Мориса Эшера

Для изменения параметров `wx`, `wy`, `w1` и `w2` во время выполнения проекта поместим на единственную форму (рис. 1.1) четыре компонента `TSpinEdit`. Для более плавного изменения значений параметров используем коэффициент масштабирования равный 1/10. Например:

```
wx:=SpinEditWx.Value/10.
```

Прежде чем рисовать график функции на бумаге, мы выбираем на ней окно рисования, задавая интервалы изменения переменных `x∈[x1,x2]` и `y∈[y1,y2]`. При переходе от "бумажного" окна `(x1,y1),(x2,y2)` к окну на экране `(I1,J1),(I2,J2)` придется масштабировать координаты точки с помощью функций `II(x)` и `JJ(y)` (листинг 1.2).

Листинг 1.2. Функции масштабирования

```
function TForm1.II(x: real): integer;
// функция масштабирования по оси OX
begin
  II:=I1+Trunc((x-X1)*(I2-I1)/(X2-X1));
end;

function TForm1.JJ(y: real): integer;
// функция масштабирования по оси OY
begin
  JJ:=J2+Trunc((y-Y1)*(J1-J2)/(Y2-Y1));
end;
```

Размеры окна на экране и на бумаге зададим при запуске проекта (листинг 1.3).

Листинг 1.3. Инициализация переменных

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Bitmap:=TBitmap.Create;
  Bitmap.Transparent := true;       //false;
  Bitmap.TransparentMode := tmAuto; //tmFixed;
  BitMap.LoadFromFile('Waterfal.bmp');
  BitMap.TransParentColor := clWhite;

  n:=200;
  X1:=-1.2; X2:=1.2; Y1:=-1.2; Y2:=1.2;
  // Задание окна на экране
  I1:=0; J1:=0; I2:=2*Width div 5; J2:=2*Height div 5;
  // вычисление шага по оси OX
  h:=2*Pi/n;
  DrawGraphic;
end;
```

Более подробное обсуждение функций масштабирования отложим до следующих глав, а пока займемся рассмотрением основного для данного проекта метода `DrawGraphic` (листинг 1.4). Первые четыре строчки метода `DrawGraphic` присваивают значения компонентов `SpinEdit` соответствующим параметрам функций. Все остальные события происходят на полотне компонента `Image1`. Сначала назначается белый цвет пера `Pen.Color:=clWhite` и кисти `Brush.Color:= clWhite` и вызовом метода `RectAngle(0,0,Width,Height)` рисуется белый прямоугольник размером во все полотно, т. е. полотно `Image1` очищается.

Листинг 1.4. Рисование графика

```
procedure TForm1.DrawGraphic;
var i: integer;
    t: real;
begin
  wx:=SpinEditWx.Value/10;
  wy:=SpinEditWy.Value/10;
  w1:=SpinEditW1.Value/10;
  w2:=SpinEditW2.Value/10;
  with Canvas do begin
    RectAngle(0,0,Width,Height);
    FillRect(Rect(0,0,Width,Height));
    StretchDraw(Rect(0,0,W,Height),BitMap);
    // Построение осей координат
    MoveTo(II(0),JJ(Y1)); LineTo(II(0),JJ(Y2));
    MoveTo(II(x1),JJ(0)); LineTo(II(x2),JJ(0));
    // Построение графика функции отрезками
    t:=0; x:=Fx(t); y:=Fy(t); MoveTo(II(x),JJ(y));
    for i:=1 to 5*n do begin
      t:=t+h; x:=Fx(t);
      y:=Fy(t); LineTo(II(x),JJ(y));
    end;
  end;
end;
```

Затем, изменив цвет карандаша на черный `Pen.Color:=clBlack`, рисуем оси координат — прямые линии от точки `(x1,0)` до `(x2,0)` и от точки `(0,y1)` до `(0,y2)`, масштабируя "бумажные" координаты в экранные с помощью функций `II(x)` и `JJ(y)`. Прорисовку отрезка прямой производим следующим образом: сначала методом полотна `MoveTo(x1,y1)` переводим экранный указатель в точку `(x1,y1)`, потом методом `LineTo(x2,y2)` перемещаем его в точку `(x2,y2)`.

Переходим теперь к рисованию графика. При активизации формы был вычислен шаг изменения `h:=2*Pi/n` параметра `t`, который на периоде `2*Pi` принимает `n` значений. Перед началом рисования параметру `t` присвоим значение `0` и переместим указатель в точку `(x,y)`.

```
t:=0; x:=Fx(t); y:=Fy(t); MoveTo(II(x),JJ(y));
```

Будем увеличивать `t` на `h` в цикле и рисовать отрезок до следующей точки.

```
t:=t+h; x:=Fx(t); y:=Fy(t); LineTo(II(x),JJ(y));
```

Для компонента `SpinEditWx` назначим обработчик события, который при изменении свойства `SpinEditWx.Value` перерисует график (листинг 1.5).

Листинг 1.5. Событие при изменении *SpinEditWx*

```
procedure TForm1.SpinEditWxChange(Sender: TObject);
begin
  DrawGraphic;
end;
```

Эту же процедуру используем в качестве обработчика события `onChange` для остальных компонентов.

Проект для рисования графика функции готов. Осталось нажать клавишу <F9> для запуска проекта на выполнение.

Усложним наш проект, введя возможность изменения цвета пера и кисти. Для этого поставим на форму две кнопки `SpeedButton1` и `SpeedButton2` и используем метод `ColorDialog1`, вызывающий диалоговое окно **Цвет** для выбора цвета. При нажатии кнопки `SpeedButton1` будем вызывать процедуру обработки события `SpeedButton1Click` (листинг 1.6).

Листинг 1.6. Изменение цвета кисти

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  if ColorDialog1.Execute then begin
    Canvas.Brush.Color:=ColorDialog1.Color;
    DrawGraphic;
  end;
end;
```

После выбора в диалоговом окне **Цвет** нового цвета, цвет кисти изменится `Image1.Canvas.Brush.Color:=ColorDialog1.Color` и будет перерисован график вызовом метода `DrawGraphic`.

При нажатии кнопки `SpeedButton2` тот же метод `ColorDialog1` используется для изменения цвета пера (листинг 1.7).

Листинг 1.7. Изменение цвета пера

```
procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  if ColorDialog1.Execute then begin
    Canvas.Pen.Color:=ColorDialog1.Color;
    DrawGraphic;
  end;
end;
```

Поставим на форму компонент Timer1 и две кнопки SpeedButton3 и SpeedButton4. Единственное событие счетчика времени Timer1Timer(Sender) вызывается через интервал времени, определенный свойством Timer1.Interval, если Timer1.Enabled=true. До начала выполнения проекта зададим значение свойства Timer1.Enabled=false, а для кнопки SpeedButton3 определим событие при щелчке кнопки мыши (листинг 1.8).

Листинг 1.8. Подключение таймера

```
procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  Timer1.Enabled:=not Timer1.Enabled;
end;
```

При нажатии кнопки SpeedButton3 будет запущен механизм таймера, который через 1000 мс будет вызывать обработчик события, который случайным образом меняет значения SpinEdit и цвета кисти и пера (листинг 1.9).

Листинг 1.9. Событие, контролируемое таймером

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  SpinEditWx.Value:=Random(100);
  SpinEditWy.Value:=Random(100);
  SpinEditW1.Value:=Random(100);
  SpinEditW2.Value:=Random(100);
end;
```

В этой процедуре параметры линии и цвета меняются случайным образом.

Для кнопки SpeedButton4 назначим событие выключения таймера (листинг 1.10).

Листинг 1.10. Выключение таймера

```
procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
  Timer1.Enabled:=false;
end;
```

Введем еще одно изменение в проект. На полотно формы выведем изображение из графического файла Waterfal.BMP. Нам потребуется переменная Bitmap: TBitmap, которую инициализируем при запуске проекта в процедуре FormCreate(Sender).

```
Bitmap:=TBitmap.Create;
Bitmap.Transparent := true; //false;
Bitmap.TransparentMode := tmAuto; //tmFixed;
BitMap.LoadFromFile('Waterfal.bmp');
BitMap.TransParentColor := clWhite;
```

Вносим изменения: в этой же процедуре назначаем значение свойства прозрачности изображения `Bitmap.Transparent := true`. Это означает, что точки одного из цветов (в нашем случае `BitMap.TransParentColor := clWhite`) не будут выводиться. Методом `BitMap.LoadFromFile('Waterfal.bmp')` загрузим изображение из файла `Waterfal.BMP` в `BitMap`.

Осталось ввести последнее изменение: в процедуру рисования `DrawGraphic` добавить строчку

```
Canvas.StretchDraw(Rect(0,0,W,Height),BitMap); ,
```

после чего изображение из файла `Waterfal.BMP`, ранее загруженное в `BitMap` методом `LoadFromFile`, переносится в прямоугольник `Rect(0,0,W,Height)` полотна формы.

1.4. Мультимедийные ресурсы Windows

В Delphi включен компонент `TMediaPlayer`, позволяющий легко получить доступ к мультимедийным ресурсам Windows. Этот компонент может воспроизводить аудио- и видеофайлы в форматах WAV, MIDI и AVI. Для нормальной работы компонента `TMediaPlayer` необходимы специальные драйверы и апплеты, но в среде Windows они, обычно, инсталлируются автоматически.

Для создания приложения, реализующего доступ к мультимедийным ресурсам Windows, достаточно разместить на форме (рис. 1.2) компонент `TMediaPlayer` и создать обработчик события для нажатия кнопки, запускающей этот компонент (листинг 1.11).

Листинг 1.11. Воспроизведение аудио- и видеофайлов

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  MediaPlayer1.Close;
  if OpenDialog1.Execute then begin
    MediaPlayer1.FileName:=OpenDialog1.FileName;
    MediaPlayer1.Display := Panel1;
    MediaPlayer1.Open;
    MediaPlayer1.Play;
  end;
end;
```

Если работа метода `OpenDialog1` заканчивается выбором AVI-файла, то начинается воспроизведение этого файла.

Воспроизведение звуковых файлов возможно, естественно, при наличии звуковых колонок и звуковой карты.