

Николай Тюкачев,
Игорь Илларионов,
Виктор Хлебостроев

Программирование графики в Delphi



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 В-сплайнов

Создание растровых и векторных графических редакторов

Работа с трехмерными изображениями

Задачи триангуляции, проектирование и реализация ГИС

+CD

bhv[®]

УДК 681.3.068+800.92Delphi
ББК 32.973.26-018.1
Т98

Тюкачев, Н. А.

Т98 Программирование графики в Delphi / Н. А. Тюкачев, 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 В-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	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Наталья Баскакова</i>
Компьютерная верстка	<i>Натальи Смирновой</i>
Корректор	<i>Виктория Пиотровская</i>
Дизайн серии	<i>Игоря Цырульникова</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 27.06.08.

Формат 70×100¹/₁₆. 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

© Тюкачев Н. А., Илларионов И. В., Хлебостроев В. Г., 2008
© Оформление, издательство "БХВ-Петербург", 2008

Оглавление

ВВЕДЕНИЕ.....	1
ГЛАВА 1. РИСОВАНИЕ В DELPHI.....	9
1.1. Моделирование цветов.....	9
1.2. Полотно компонентов	10
1.3. Пример использования графики.....	11
1.4. Мультимедийные ресурсы Windows.....	17
ГЛАВА 2. МОДУЛЬ GRAPHICS И СПЕЦИАЛЬНЫЕ ПРИЕМЫ РИСОВАНИЯ	19
2.1. Структура классов	19
2.2. Цвет	22
2.3. Цветовые модели	24
2.3.1. Модель RGB	24
2.3.2. Модель CMY	25
2.3.3. Модель CMYK	26
2.3.4. Модели HSB и HSV	27
2.3.5. Модель Lab	28
2.4. Проект "Цветовые модели".....	28
2.4.1. Процедуры для модели RGB	29
2.4.2. Процедуры для модели HSV.....	30
2.4.3. Процедуры для модели HSI	32
2.5. Класс <i>TFont</i>	34
2.6. Класс <i>TPen</i>	39
2.7. Класс <i>TBrush</i>	43
2.8. Класс <i>TCanvas</i>	46
2.9. Методы канвы	49
2.10. Чтение данных из текстового файла.....	62
2.11. Вывод строки под углом	66
2.11.1. Установка угла для печати строки	66
2.11.2. Тип логического шрифта.....	68
2.12. Рисование на экране	73

ГЛАВА 3. ГРАФИЧЕСКИЕ КЛАССЫ.....	77
3.1. Класс <i>TGraphic</i>	77
3.2. Класс <i>TPicture</i>	83
3.3. Класс <i>TBitmap</i>	86
3.4. Класс <i>TMetafile</i>	92
3.5. Класс <i>TIcon</i>	94
3.6. Функции для работы с графикой.....	95
3.7. Класс <i>TImage</i>	101
3.8. Класс <i>TJPEGImage</i>	105
3.9. Класс <i>TPrinter</i>	110
3.10. Заключение.....	116
ГЛАВА 4. МУЛЬТИМЕДИА	117
4.1. Компонент <i>Animate</i>	118
4.2. Компонент <i>MediaPlayer</i>	124
4.3. Проект с использованием компонента <i>MediaPlayer</i>	154
4.4. Процедуры воспроизведения звуков <i>Beep</i> , <i>MessageBeep</i> и <i>PlaySound</i>	158
4.5. Интерфейс управления мультимедийными устройствами — MCI.....	161
4.5.1. Проект "Консольное выполнение команд MCI".....	163
4.5.2. Проект "Проигрыватель аудио-CD".....	168
4.5.3. Проект "Запись звука с использованием команд MCI".....	174
4.6. Программирование мультимедийных приложений с использованием WinAPI.....	175
4.6.1. Структура RIFF-файла.....	176
4.6.2. Проект "Низкоуровневое чтение файла".....	180
4.6.3. Проект "Низкоуровневое воспроизведение файла".....	183
ГЛАВА 5. КОМПОНЕНТЫ ДИАГРАММ БИБЛИОТЕКИ <i>TEECHART</i>.....	189
5.1. Деловая графика.....	189
5.2. Подготовка к работе.....	191
5.3. Создание новой диаграммы с компонентом <i>TChart</i> или <i>TDBChart</i>	195
5.4. Соединение диаграммы с разными типами данных.....	201
5.5. Свойства компонента <i>TChart</i>	203
5.6. Типы <i>Series</i>	204
5.6.1. Серии <i>Line</i> и <i>Fast Line</i>	204
5.6.2. Серия <i>Bar</i>	205

5.6.3. Серия <i>Horizontal Bar</i>	210
5.6.4. Серия <i>Area</i>	211
5.6.5. Серия <i>Point</i>	212
5.6.6. Серия <i>Pie</i>	212
5.6.7. Серия <i>Arrow</i>	213
5.6.8. Серия <i>Bubble</i>	214
5.6.9. Серия <i>Gantt</i>	215
5.6.10. Серия <i>Shape</i>	216
5.6.11. Комбинированные серии	218
5.7. Функции для вычисляемых серий	219
5.7.1. Функция <i>TAddTeeFunction</i>	221
5.7.2. Функция <i>TSubtractTeeFunction</i>	222
5.7.3. Функция <i>TMultiplyTeeFunction</i>	222
5.7.4. Функция <i>TDivideTeeFunction</i>	224
5.7.5. Функция <i>THighTeeFunction</i>	224
5.7.6. Функция <i>TLowTeeFunction</i>	224
5.7.7. Функция <i>TAverageTeeFunction</i>	226
5.8. Особенности разработки приложений, использующих диаграммы	226
5.8.1. Обработка событий нажатия кнопок	226
5.8.2. Рисование на диаграмме	228
5.8.3. Работа с осями	233
5.8.4. Действия с сериями	236
5.8.5. Изменение масштаба изображения	241
5.8.6. Особенности разработки проектов, работающих в реальном масштабе времени	244
5.9. Проект с использованием диаграмм	245
5.9.1. Генерация данных и добавление серий	247
5.9.2. Изменение свойств серии	250
5.9.3. Изменение общих свойств диаграммы	251
5.9.4. Изменение 3D-свойств диаграммы	251

ГЛАВА 6. АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ГРАФИКИ

6.1. Задачи компьютерной графики	253
6.2. Классификация алгоритмов	254
6.3. Построение растровых изображений	255
6.3.1. Алгоритм Брезенхейма для отрезка прямой	257
6.3.2. Алгоритм Брезенхейма для окружности	261
6.3.3. Экранная система координат	263
6.3.4. Проект "Алгоритмы Брезенхейма"	264

6.4. Геометрические основы компьютерной графики.....	278
6.4.1. Графические элементы на плоскости	279
6.4.2. Графические элементы в пространстве	281
6.5. Задачи интерполяции, сглаживания и аппроксимации.....	284
6.5.1. Интерполяция полиномами	284
6.5.2. Интерполяция кубическими сплайнами	286
6.5.3. Сглаживание и аппроксимация	287
6.6. Аффинные преобразования координат.....	291
6.6.1. Аффинные преобразования на плоскости	291
6.6.2. Аффинные преобразования в пространстве.....	297
6.7. Проецирование.....	301
6.7.1. Ортографическое проецирование.....	303
6.7.2. Аксонометрическое проецирование	304
6.7.3. Косоугольное проецирование	307
6.7.4. Центральное проецирование.....	308
6.7.5. Проект "Проекция"	312
6.8. Моделирование трехмерных тел	321
6.8.1. Каркасные модели	322
6.8.2. Граничные модели	323
6.8.3. Сплошные модели	323
6.9. Освещение	325
6.10. Моделирование цвета.....	327
6.11. Удаление невидимых ребер и граней	328
ГЛАВА 7. ПРОСТЫЕ ГРАФИЧЕСКИЕ ПРОЕКТЫ	331
7.1. Просмотр файлов BMP, ICO, WMF, EMF и JPG	332
7.2. Мультипликация	335
7.2.1. Сортировка элементов массива	336
7.2.2. Морфинг	339
7.2.3. Падение мяча.....	344
7.2.4. Велосипед	347
7.3. Рисование на канве принтера	352
7.4. Векторный стиль линии	353
7.4.1. Рисование линии стандартными способами	355
7.4.2. Применение векторного стиля линии	355
7.4.3. Проект "Рисование линии произвольным стилем".....	360
7.5. Деформация изображений	364
7.6. Растровый редактор.....	370

7.7. Проектирование плоских схем	379
7.7.1. Структура данных	380
7.7.2. Структура проекта	383
7.7.3. Добавление нового объекта в эскиз	387
7.7.4. Перемещение объектов и линий связи на эскизе.....	389
7.7.5. Удаление объектов и линий связи на эскизе.....	395
7.8. Редактирование графа	396
7.8.1. Структура данных	398
7.8.2. Изображение графов.....	399
7.8.3. Чтение и запись графов	400
7.9. Проект газификации домов	402
7.9.1. Структура проекта	404
7.9.2. Структура данных	405
7.9.3. Рисование эскиза газификации дома	406
ГЛАВА 8. ВЕКТОРНЫЙ РЕДАКТОР.....	411
8.1. Структура данных.....	412
8.2. Масштабирование.....	414
8.3. Кривые Безье.....	417
8.4. Создание объектов.....	418
8.5. Перемещение объектов	422
8.6. Поворот объектов	426
8.7. Перемещение точек	427
8.8. Прорисовка объектов	428
8.9. Печать	430
8.10. Запись и чтение данных	430
ГЛАВА 9. ГРАФИКИ ФУНКЦИЙ.....	437
9.1. График функции одной переменной.....	437
9.2. График функции двух переменных.....	444
9.3. Интерполяция функций.....	453
9.3.1. Проект "Построение интерполяционных кривых"	454
9.3.2. Интерполяционный многочлен Лагранжа.....	459
9.3.3. Метод наименьших квадратов.....	461
9.3.4. Кубические сплайны.....	464
9.3.5. Кривые Безье	469
9.4. Параметрические кривые	470

9.5. Построение графика функции с помощью интерпретатора	473
9.5.1. Структура данных	473
9.5.2. Анализ строки	476
9.5.3. Вычисление переменной	485
ГЛАВА 10. ВИЗУАЛЬНЫЙ ГЕНЕРАТОР ОТЧЕТОВ	489
10.1. Постановка задачи	489
10.2. Описание структуры данных	491
10.3. Структура проекта	494
10.4. Рисование страницы эскиза	496
10.5. Добавление объектов.....	503
10.6. Редактирование объектов	507
10.7. Перемещение объектов	510
10.8. Изменение размеров объектов	511
10.9. Печать отчета	513
10.10. Заключение.....	515
ГЛАВА 11. ГЕОМЕТРИЯ ТРЕХМЕРНЫХ ТЕЛ.....	517
11.1. Платоновы тела.....	517
11.1.1. Построение платоновых тел	518
11.1.2. Проект "Платоновы тела".....	519
11.2. Квадратичные поверхности	535
11.2.1. Уравнения квадратичных поверхностей в явной форме.....	535
11.2.2. Параметрическое представление квадратичных поверхностей	537
11.2.3. Проект "Квадратичные поверхности".....	539
11.3. Построение тела по трем проекциям	545
11.4. Бинарные операции с многоугольниками	552
ГЛАВА 12. ГРАФИЧЕСКИЕ РЕДАКТОРЫ ТРЕХМЕРНЫХ ТЕЛ.....	563
12.1. Упрощенный проект "Редактор многогранников"	563
12.1.1. Описание проекта	563
12.1.2. Чтение и запись данных	566
12.1.3. Анализ данных и рисование	569
12.1.4. Новый многогранник.....	574
12.1.5. Добавление вершины	575

12.1.6. Переключение инструментов	577
12.1.7. Выравнивание дочерних окон	578
12.1.8. Нажатие кнопки мыши на дочерних формах	579
12.1.9. Обработка перемещения указателя мыши на формах.....	581
12.2. Редактор для топологически связанных трехмерных тел.....	584
12.2.1. Структура данных	584
12.2.2. Структура данных проекта.....	584
12.2.3. Трехмерный редактор многогранников.....	587
12.2.4. Пересечение двух тел	591
12.2.5. Создание нового тела	600
ГЛАВА 13. ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКОЙ БИБЛИОТЕКИ OPENGL	611
13.1. Введение	611
13.2. Установка и завершение работы с OpenGL	614
13.2.1. Получение дескриптора контекста воспроизведения.....	615
13.2.2. Установка формата пикселей.....	615
13.2.3. Инициализация библиотеки OpenGL.....	619
13.2.4. Завершение работы с OpenGL	621
13.3. Команды и примитивы OpenGL.....	621
13.3.1. Синтаксис команд	621
13.3.2. Вершины	622
13.3.3. Примитивы	623
13.4. Плоская графика	624
13.5. Трехмерная графика	628
13.5.1. Инициализация OpenGL.....	629
13.5.2. Многогранники модуля DGLUT	630
13.5.3. Списки команд	633
13.5.4. Изображение квадратичных поверхностей	635
13.5.5. Изображение поверхности, заданной табличным способом	637
13.6. Геометрические преобразования.....	640
13.7. Цвет, освещение, свойства материала	643
13.7.1. Цвет	644
13.7.2. Нормали	645
13.7.3. Свойства материала	645
13.7.4. Источники света.....	647
13.8. Текстура.....	648
13.8.1. Назначение точки карты текстуры вершине	649
13.8.2. Задание параметров текстуры.....	649
13.8.3. Создание двумерной карты текстуры	652

13.8.4. Включение режима наложения текстуры.....	654
13.8.5. Текстура на сфере, конусе и чайнике.....	654
13.8.6. Привязка текстуры к многоугольникам.....	656
13.8.7. Текстура на поверхности, заданной табличным способом.....	657
13.9. Чтение данных из текстового файла.....	660
13.10. Проект "Редактор многогранников"	664
ГЛАВА 14. АЛГОРИТМЫ ТРИАНГУЛЯЦИИ ПОВЕРХНОСТЕЙ В ТРЕХМЕРНОМ ПРОСТРАНСТВЕ.....	673
14.1. Триангуляция поверхности.....	673
14.1.1. Алгоритмы триангуляции	675
14.1.2. Структура данных	679
14.1.3. Реализация алгоритма	681
14.1.4. Удаление "лишних" треугольников	688
14.2. Триангуляция всех слоев участка.....	689
14.2.1. Структура данных	690
14.2.2. Алгоритм построения триангуляции слоев	692
14.3. Сглаживание триангуляции	697
14.3.1. Структура данных	698
14.3.2. Бикубическая поверхность Безье	699
14.3.3. Вспомогательные функции	700
14.3.4. Алгоритм сглаживания триангуляции	702
14.4. Триангуляция боковой поверхности слоя	715
14.4.1. Структура данных	716
14.4.2. Алгоритм определения номеров граничных точек.....	716
14.4.3. Построение треугольников боковой поверхности.....	723
14.5. Триангуляция невыпуклого многоугольника.....	724
14.6. Изолинии	728
ПРИЛОЖЕНИЯ	735
ПРИЛОЖЕНИЕ 1. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ	737
Задания по темам главы 3	737
Задания по темам главы 4	737
Задания по темам "Компонент <i>Animate</i> ", "Процедуры воспроизведения звуков <i>Beep</i> , <i>MessageBeep</i> и <i>PlaySound</i> "	737
Задания по теме "Компонент <i>TMediaPlayer</i> "	739

Задания по теме "Интерфейс управления мультимедийными устройствами – MCI"	740
Задания по теме "Программирование мультимедийных приложений с использованием WinAPI"	742
Задания по темам главы 6	743
Задания по темам главы 7	743
Задания по темам главы 9	744
Задания по темам главы 11	744
Задания по темам главы 12	745
ПРИЛОЖЕНИЕ 2. ОПИСАНИЕ ПРИЛАГАЕМОГО КОМПАКТ-ДИСКА	749
СПИСОК ЛИТЕРАТУРЫ	752
ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ.....	759

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]:

$$\text{Яркость} = 0,59 \times \text{Зеленый} + 0,3 \times \text{Красный} + 0,11 \times \text{Синий}$$

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 удастся 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 пикселями (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 покрашен 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 .. $FFFFFFF.
```

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 `Grahic`s.

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 $\pi/2$ 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;
```

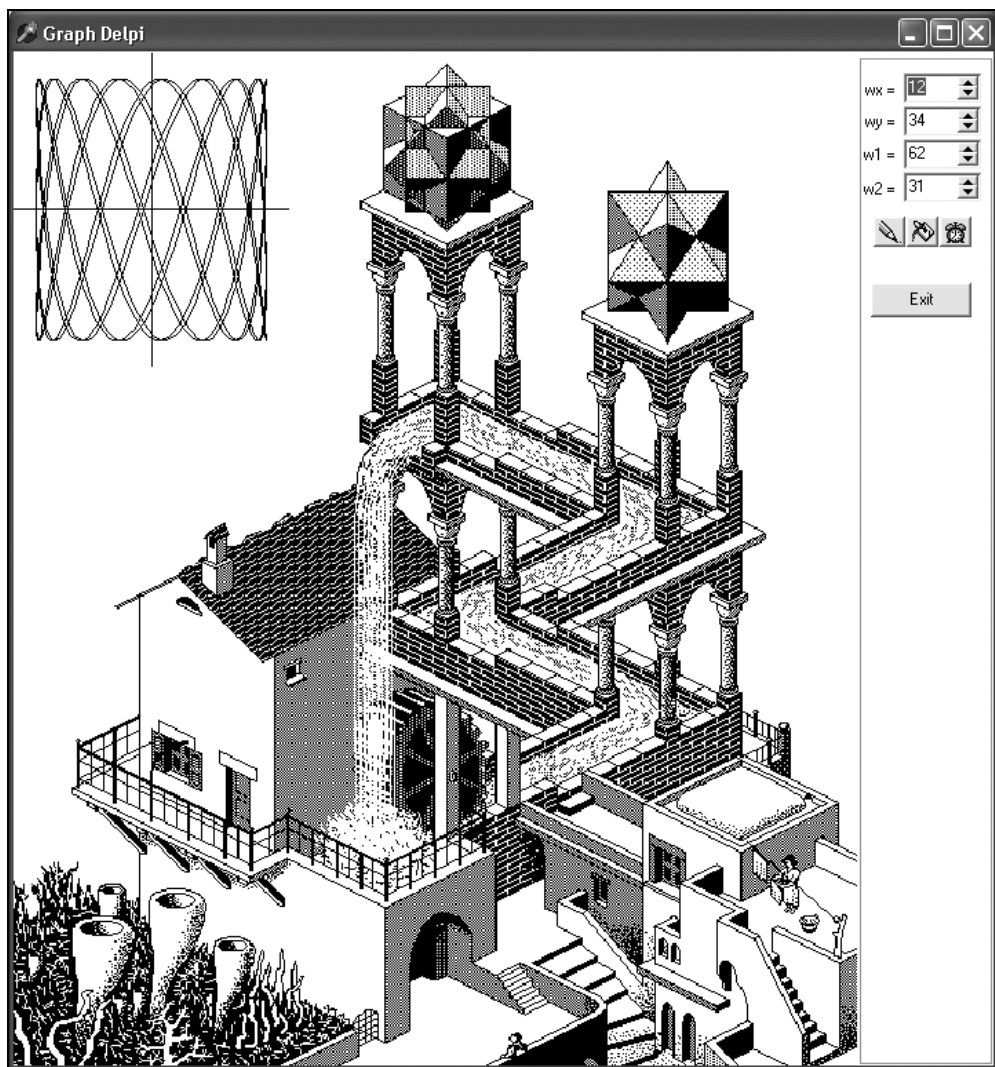


Рис. 1.1. Проект "Кривые Лиссажу" с копией картины Мориса Эшера

Для изменения параметров w_x , w_y , w_1 и w_2 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 $x \in [x_1, x_2]$ и $y \in [y_1, y_2]$. При переходе от "бумажного" окна (x_1, y_1) , (x_2, y_2) к окну на экране (I_1, J_1) , (I_2, J_2) 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 $(x_1, 0)$ до $(x_2, 0)$ и от точки $(0, y_1)$ до $(0, y_2)$, 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 (x_1, y_1) , потом методом `LineTo(x2,y2)` перемещаем его в точку (x_2, y_2) .

Переходим теперь к рисованию графика. При активизации формы был вычислен шаг изменения $h:=2*\text{Pi}/n$ параметра t , который на периоде $2*\text{Pi}$ 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 OpenFileDialog1.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.



Рис. 1.2. Воспроизведение AVI-файла

На этом мы заканчиваем краткий обзор графических возможностей и переходим к более последовательному изучению графических структур Delphi.

ГЛАВА 2



Модуль *Graphics* и специальные приемы рисования

2.1. Структура классов

Основным источником информации о графических классах Delphi, их методах и свойствах является справочная система Delphi. Однако часть информации можно найти, например, в книгах А. Я. Архангельского [60], П. Дарахвелидзе [70].

В приложениях, разработанных в среде Delphi, рисование возможно только на компонентах, обладающих свойством `Canvas` (холстом, канвой, полотном). В частности, этим свойством обладают компоненты `TForm`, `TStringGrid`, `TImage`, `TPrinter`. Родительским для класса `TCanvas` и для всех остальных графических классов является класс `TPersistent`.

В листинге 2.1 представлена структура классов и их основных свойств, используемых при рисовании. Потомками класса `TPersistent` являются:

- ❑ класс `TGraphicsObject`, порождающий (инкапсулирующий) классы инструментов;
- ❑ класс `TCanvas`, содержащий инструменты и методы рисования;
- ❑ класс `TGraphic`, порождающий четыре класса изображений со своим форматом файлов (`TBitmap`, `TIcon`, `Tmetafile`, `TJPEGImage`);
- ❑ класс `TPicture`, надстройка над `TGraphic`, точнее, над его потомками. Он содержит поле `Graphic`, которое может содержать `TBitmap`, `Ticon`, `TMetafile` или `TJPEGImage`.

Листинг 2.1 достаточно велик, но мы решили включить его в настоящий раздел для того, чтобы более отчетливо отобразить иерархию классов и перечислить основные свойства объектов.

Листинг 2.1. Структура графических классов *TPersistent*

```
TGraphicsObject
    Tfont                // класс шрифта
    PixelsPerInch: Integer // число точек на дюйм
```

```

Color: TColor           // цвет шрифта
Height: Integer        // высота шрифта
Name: TFontName        // имя шрифта
Pitch: TfontPitch      // ширина символов шрифта
Size: Integer          // размер
Style: TFontStyles     // стиль
TPen                   // класс карандаша
Color: TColor           // цвет пера
Mode: TPenMode         // режим рисования линии
Style: TPenStyle       // стиль линии
Width: Integer         // толщина линии
TBrush                 // класс кисти
Bitmap: TBitmap        // Bitmap для кисти
Handle: HBrush         // указатель кисти
Color: TColor           // цвет кисти
Style: TBrushStyle     // стиль заливки
TCanvas                // класс канвы
ClipRect: TRect       // область отсечения канвы
Handle: HDC            // указатель канвы
PenPos: TPoint         // позиция указателя пера
Pixels[X, Y: Integer]: TColor // массив пикселей канвы
OnChange: TnotifyEvent // событие после изменения
OnChanging: TNotifyEvent // событие перед изменением
Brush: TBrush          // свойство кисть
CopyMode: TCopyMode   // режим копирования
Font: TFont            // свойство шрифт
Pen: TPen              // свойство перо
TControlCanvas        // родительский класс для канвы
TBitmapCanvas         // канва класса TBitmap
TPrinterCanvas        // канва принтера
TMetafileCanvas       // канва метафайла
TGraphic
TMetafile              // класс метафайлов
  CreatedBy: String    // имя автора
  Description: String  // описание файла
  Enhanced: Boolean    // способ сохранения на диске
  Handle: HENHMETAFILE // указатель на экземпляр
  MMWidth: Integer     // ширина в сотых долях миллиметров
  MMHeight: Integer    // высота в сотых долях миллиметров
  Inch: Word           // число единиц на дюйм
TBitmap                // класс, содержащий изображение
  Canvas: TCanvas     // свойство полотно
  Handle: HBITMAP     // указатель на экземпляр
  HandleType: TBitmapHandleType // признак аппаратной зависимости
  Height              // высота в пикселах

```

```
IgnorePalette: Boolean// игнорировать палитру
MaskHandle: HBITMAP // указатель на битовый массив GDI
Monochrome: Boolean // монохромность
Palette // палитра
PixelFormat // формат пикселей
ScanLine[Row: Integer]: Pointer // указатель к линиям пикселей
TransparentColor: TColor // прозрачный цвет
TransparentMode: TTransparentMode // режим прозрачности
Width // ширина канвы
Modified // модифицированность
PaletteModified // модифицированность палитры
Transparent // прозрачность
TJPEGImage // класс для файлов формата JPG
CompressionQuality // отношение между качеством и размером файла
Empty // содержит ли JPG-графику
GrayScale // оттенки серого цвета
Height // высота в пикселях
Palette // палитра
Performance // отношение между качеством и скоростью
// декомпрессии
PixelFormat: TPixelFormat // формат пикселя
ProgressiveDisplay // коэффициент уменьшения изображения
ProgressiveEncoding // показывает, уменьшено ли изображение
Scale // определяет размер изображения
Smoothing // сглаживание краев
Width // ширина
Modified // модифицированность
PaletteModified // модифицированность палитры
Transparent // прозрачность
TIcon // класс для файлов формата ICO
Handle: HICON // указатель на экземпляр класса
TPicture // класс изображений
Bitmap: TBitmap // свойство TBitmap
Graphic: TGraphic // свойство TGraphic
PictureAdapter: IChangeNotifier // интерфейс OLE
Height: Integer // высота
Icon: TIcon // свойство типа TIcon
Metafile: TMetafile // свойство типа метафайл
Width: Integer // ширина
OnChange: TNotifyEvent// событие после изменения
OnProgress: TProgressEvent // событие в процессе изменения
```

Для рисования используются методы класса TCanvas и три класса инструментов: TFont (шрифты), TPen (карандаш, перо), TBrush (кисть).

Абстрактный класс `TGraphic` является родительским для четырех видов изображений, соответствующих четырем форматам графических файлов: пиктограммы — формат ICO (класс `TIcon`), метафайлы — форматы EMF, WMF (класс `TMetafile`), растровые изображения — формат BMP (класс `TBitmap`) и сжатые фотоизображения — формат JPG (класс `TJPEGImage`). Из этих четырех графических классов только `TBitmap` обладает опубликованным свойством `TCanvas`.

Эти классы, постоянные и некоторые функции описаны в модуле `Graphics`.

2.2. Цвет

Класс цвета точки `TColor` определен как длинное целое `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`, т. е. цвет можно вычислить через доли `R`, `G`, `B` по формуле:

$$Color = 256^2 \times B + 256 \times G + R,$$

или, используя битовую операцию "сдвиг налево" `shl`,

$$Color = B \text{ shl } 16 + G \text{ shl } 8 + R.$$

В модуле `Graphics` приведены некоторые константы для цветов (листинг 2.2).

Листинг 2.2. Предопределенные константы для цветов

```
Const
  clBlack = TColor($000000);
  clMaroon = TColor($000080);
  clGreen = TColor($008000);
  clOlive = TColor($008080);
  clNavy = TColor($800000);
  clPurple = TColor($800080);
  clTeal = TColor($808000);
  clGray = TColor($808080);
  clSilver = TColor($C0C0C0);
  clRed = TColor($0000FF);
  clLime = TColor($00FF00);
  clYellow = TColor($00FFFF);
  clBlue = TColor($FF0000);
  clFuchsia = TColor($FF00FF);
  clAqua = TColor($FFFF00);
  clLtGray = TColor($C0C0C0);
```