

ДМИТРИЙ ОСИПОВ



ГРАФИКА В ПРОЕКТАХ DELPHI

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 5-93286-134-7, название «Графика в проектах Delphi» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (piracy@symbol.ru), где именно Вы получили данный файл.

H I G H T E C H

Графика в проектах
Delphi

Дмитрий Осипов



Санкт-Петербург — Москва
2008

Серия «High Tech»

Дмитрий Осипов

Графика в проектах Delphi

Главный редактор	<i>А. Галунов</i>
Зав. редакцией	<i>Н. Макарова</i>
Выпускающий редактор	<i>П. Щеголев</i>
Редактор	<i>Ю. Бочина</i>
Художник	<i>В. Гренда</i>
Корректор	<i>Л. Минина</i>
Верстка	<i>Д. Орлова</i>

Осипов Д.

Графика в проектах Delphi. – СПб: Символ-Плюс, 2008. – 648 с.: цв. ил.

ISBN-13: 978-5-93286-134-9

ISBN-10: 5-93286-134-7

В книге Дмитрия Осипова «Графика в проектах Delphi» представлен уникальный материал, посвященный программированию деловой графики для современных версий Windows. Рассмотрены графический механизм системы, функции прикладного интерфейса программирования GDI (Graphics Device Interface), методы работы с графикой средствами визуальной библиотеки Delphi и тонкости современной графической библиотеки Windows GDI+. Обсуждаются особенности управления цветом и вывода текста, рисование примитивов, страничные и мировые преобразования, форматы растровых и векторных рисунков, организация работы с печатающим устройством, обработка метаданных в современной цифровой фотографии и приемы улучшения качества изображений, цветовая коррекция и многое другое, без чего нельзя создать интерфейс современного программного продукта.

Книгу отличает глубина и ясность изложения материала, поэтому она будет полезна как начинающему программисту, так и профессионалу, который сможет использовать ее как справочник по функциям и методам среды разработки Delphi.

ISBN-13: 978-5-93286-134-9

ISBN-10: 5-93286-134-7

© Дмитрий Осипов, 2008

© Издательство Символ-Плюс, 2008

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 29.08.2008. Формат 70×100¹/16. Печать офсетная.

Объем 40,5 печ. л. Тираж 2000 экз. Заказ №

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

Оглавление

Введение	11
I. Graphics Device Interface (GDI)	15
1. Программирование на Windows API	17
Дескрипторы, или особенность доступа к объектам Windows	18
Описание класса главного окна	19
Регистрация класса окна	21
Создание экземпляра окна	22
Отображение окна	23
Организация цикла обработки сообщений	24
Оконная процедура	28
Отправка сообщений	28
Листинг программы	29
Дочерние окна	32
Субклассирование	33
2. Контекст графического устройства	38
Дескриптор контекста для экрана и печатающих устройств	40
Контекст окна приложения	41
Освобождение дескриптора контекста	46
Контекст в памяти (совместимый контекст)	47
Доступ к стандартным объектам контекста	48
Доступ к текущему объекту контекста	49
Информационный контекст	50
Восстановление состояния контекста	51
Управление объектами GDI	52
Разработка хранителя экрана	54
Приложение. Функция GetDeviceCaps()	56
3. Управление устройствами видеовывода	61
Сбор информации об устройствах видеовывода	62
Изменение настроек экрана	65

4. Кисти	68
Стандартные кисти	68
Атрибуты кисти, структура TLogBrush	70
Создание логической кисти	73
5. Перья	77
Стандартные перья GDI	77
Косметическое перо	79
Геометрическое перо	81
6. Линии и кривые	86
Позиционирование пера	86
Линии	87
Дуги	92
Кривые	95
7. Простейшие геометрические фигуры и заливка областей	100
Функции определения прямоугольных областей	101
Заливка прямоугольной области	105
Простейшие геометрические фигуры	108
Режим заполнения сложной области	112
Имитация элементов управления	112
Имитация фокуса ввода	115
Вывод заголовка окна	116
Пассивное состояние элемента управления	117
8. Траектории	120
Создание траектории	121
Вывод траектории	122
Преобразование траектории в набор отрезков	125
Анализ траектории	126
Преобразование траектории в регион	128
9. Регионы	130
Создание региона	130
Вывод региона	133
Операции с регионами	135
Объединение регионов	136
Преобразование региона в прямоугольный регион	137
Получение информации о регионе	137
10. Отсечение и регионы контекста устройства	141
Окно нестандартной формы	144
Обращение к регионам контекста устройства	145

Определение региона отсечения	146
Проверка вхождения в регион отсечения	149
Определение метарегиона	149
Перерисовка региона	151
11. Системы координат и режимы отображения	153
Взаимные преобразования координат	155
Общие характеристики режимов отображения	157
Настройка страничных координат и координат устройства	160
Метрические режимы отображения	165
Пользовательские режимы отображения	166
12. Мировые координаты и аффинные преобразования	175
Перевод контекста в мировые координаты	175
Аффинные преобразования на плоскости	176
Пример «Стрелочные часы»	182
13. Представление цвета в RGB-модели	186
Хранение данных о цвете в памяти	187
Определение характеристик цвета контекста устройства	189
Макросы	189
Работа с отдельным пикселем	190
Системные цвета	193
Альтернативные цветовые модели	195
14. Цветовые палитры	196
Перевод дисплея в 8-битный режим	197
Структура палитры	198
Макросы для работы с палитрой	199
Системная палитра	199
Логическая палитра	202
Сообщения, связанные с изменением палитры	206
Поиск цвета в палитре	207
15. Аппаратно-зависимые растры	209
Представление монохромного DDB в памяти	210
Создание DDB	211
Загрузка растра из ресурса	216
Универсальная функция отображения битового образа	218
16. Аппаратно-независимые растры	220
Формат DIB-растра и файла BMP	220
Отображение DIB-растра	224
Пример загрузки образа DIB из файла BMP	226

Перенос пикселей между DIB и DDB	228
DIB-секция	229
Пример загрузки образа DIB из файла BMP с помощью DIB-секции	232
17. Растровые операции	235
Участники растровых операций	236
Бинарные растровые операции, ROP2	237
Тернарные растровые операции	239
Кватернарные операции, функция MaskBlt	246
Прозрачность	246
18. Расширенный формат метафайла, EMF	251
Структура метафайла	252
Загрузка метафайла из файла	256
Создание расширенного метафайла	257
Размещение метафайла в памяти	259
Копирование метафайла	259
Построчное воспроизведение метафайла	260
Комментарий к метафайлу	263
19. Шрифты	265
Набор символов	267
Ключевые метрики логического шрифта	269
Описание логического шрифта, структура TLogFont	271
Сбор информации об установленных шрифтах	271
Доступ к системным шрифтам	274
Логический шрифт	274
Инсталляция и удаление шрифта	280
20. Вывод текста	283
Простейшие приемы вывода текста	283
Управление выводом текста средствами контекста	286
Приемы форматирования текста	289
Дополнительные эффекты при выводе текста	294
21. Альфа-наложение и градиентная заливка	299
Альфа-наложение	299
Градиентная заливка области	305
II. Графика в VCL	309
22. Визуальная библиотека компонентов Delphi	311
Концепция ООП и опорные классы VCL	312
Простейшие графические объекты VCL	317

Глобальный объект «экран» – класс TScreen	327
Глобальный объект «монитор» – класс TMonitor	329
23. Холст VCL – класс TCanvas	331
Линии и кривые	333
Простейшие геометрические фигуры	335
Заливка области	337
Вывод текста	338
Работа с холстом в многопоточном режиме	340
24. Растровые и векторные изображения в VCL	341
Класс TGraphic	341
Иконка – класс TIcon	344
Растровое изображение – класс TBitmap	345
Метафайл – класс TMetafile	350
Класс TJPEGImage	352
Хранилище изображения – класс TPicture	354
25. Коллекционируем изображения	356
Контейнер изображений, класс TImageList	357
Экспорт пиктограмм из контейнера	364
Взаимодействие с элементами управления	365
26. Графические элементы управления VCL	367
Класс TGraphicControl	367
Изображение, компонент TImage	376
Фигура, компонент TShape	377
Область для рисования, компонент TPaintBox	378
Разделитель, компонент TSplitter	378
Рельефная панель, компонент TBevel	379
Быстрая кнопка, компонент TSpeedButton	380
Метка, компонент TLabel	381
27. Организация работы с принтером	383
Работа с принтером средствами Windows	384
Технические характеристики принтера	386
Описание принтера в Delphi, класс TPrinter	386
Печать многострочного текста	391
Печать изображений	392
Окно предварительного просмотра	393
Преобразование цветного изображения для печати на монохромном принтере	396
Диалог с принтером	397

III. GDI+	403
28. Введение в GDI+	405
Мифы и реальность	406
Подготовка к работе	408
Соглашение об именовании классов GDI+ в проектах Delphi	409
Основной объект – холст GDI+	409
Представление цвета в GDI+	411
Структуры определения координат и размеров	413
Отладка проектов GDI+	414
29. Кисти GDI+	425
Сплошная кисть TGPSolidBrush	426
Узорная кисть TGPHatchBrush	427
Текстурная кисть TGPTextureBrush	430
Аффинные преобразования кистей	434
Кисть с линейной градиентной заливкой TGPLinearGradientBrush	436
Градиентная кисть сложной формы TGPPathGradientBrush	442
30. Перья GDI+	447
Создание пера	448
Цвет и толщина пера	449
Стиль пера	450
Наконечники пера	454
Стык линий	457
Расслоение пера	459
31. Траектории GDI+	461
Траектория, класс TGPGraphicsPath	462
Последовательный просмотр траектории, класс TGraphicsPathIterator	480
32. Регионы GDI+	484
Регион, класс TGPRegion	484
Холст GDI+ и регион отсечения	495
33. Графические примитивы и заливка областей в GDI+	499
Прямые и ломаные линии	500
Кривые	502
Простейшие фигуры	506

34. Координатные системы и преобразования в GDI+	510
Страничная система координат в GDI+	511
Мировые координаты, матрица TGPMatrix	513
Мировые преобразования, класс TGPGraphics	528
35. Изображения в GDI+	530
Кодеры и декодеры изображений	531
Класс TGPIImage	533
Роль хоста при выводе рисунков, метод DrawImage	551
36. Метаданные EXIF	557
Чтение метаданных EXIF	560
Редактирование метаданных	562
Миниатюра изображения	563
37. Особенности работы с битовыми образами и метафайлами	566
Битовые образы, класс TGPBitmap	567
Метафайл, класс TGPMetafile	574
38. Работа со шрифтами в GDI+	588
Шрифт, класс TGPFont	589
Семейство шрифтов, класс TGPFontFamily	592
Коллекции шрифтов	593
39. Операции с текстом в GDI+	597
Методы TGPGraphics по выводу текста	597
Форматирование текстовой строки, класс TGPStringFormat	600
Вывод символа в точной позиции	607
Исследование строки	609
40. Качество вывода и коррекция цвета	616
Сглаживание	616
Порядок наложения	618
Интерполяция растра	619
Повышение качества вывода текста	621
Сохранение состояния холста GDI+	623
Коррекция цвета, класс TGPIImageAttributes	626
Заключение	637
Литература	639
Алфавитный указатель	641

33

Графические примитивы и заливка областей в GDI+

В этой главе мы вновь возвращаемся к изучению возможностей холста GDI+. На этот раз предметом беседы станут методы класса `TGPGraphics`, нацеленные на рисование на поверхности контекста устройства простейших графических примитивов (линий, многоугольников, кривых и других элементарных фигур). Кроме того, мы рассмотрим способы заливки замкнутых областей.

Во многом перечень предназначенных для вывода графических примитивов методов класса `TGPGraphics` определится набором функций GDI. Однако есть и исключения, точнее, усовершенствования. В первую очередь – это появление принципиально нового способа работы со сплайнами Безье. Теперь отклоняющие точки явным образом определяют траекторию сплайна, находясь прямо на кривой.

Сразу отметим общие черты большей части рассматриваемых в главе методов. Так как каждый метод GDI+ должен быть самодостаточен, то кроме координат будущих отрезков, кривых и других фигур в состав параметров всегда входит описание пера или кисти. Поэтому если речь ведется о черчении линий, то первым параметром всех функций выступает перо `Pen: TGPPen`. Если же мы планируем закрасить замкнутую область, то первым параметром метода всегда будет кисть `Brush: TGPBrush`.

Прежде чем мы перейдем к изучению материала этой главы, позволю себе небольшое лирическое отступление. Иногда при проектировании приложений с применением GDI+ в голову невольно закрадывается крамольная мысль, что над созданием библиотеки трудилась весьма необычная группа разработчиков. Программисты никуда не спешили и даже немножко скучали. Когда подходил ответственный момент очередного отчета о проделанной работе, эта группа специалистов (для того чтобы убедить руководителя проекта в своей исключительной работоспособности) по четыре раза «создавала» один и тот же метод, наделяя его разными по типу, но абсолютно одинаковыми по смысловой нагрузке параметрами. Все остальное время эти ребята

та тихонько дремали, раскладывали пасьянс и вполголоса сплетничали... Шутки шутками, но большинство объектов GDI+ в буквальном смысле слова перегружены одинаковыми методами. Судите сами...

Прямые и ломанные линии

Практически все классы GDI+ славятся большим количеством перегружаемых методов. В тройке лидеров находится холст GDI+. У него даже один из самых простейших методов, позволяющий провести линию из точки А в точку Б, и тот предложен в четырех вариантах:

```
function DrawLine(Pen: TGPPen; x1, y1, x2, y2: Integer): TStatus; overload;
function DrawLine(Pen: TGPPen; x1, y1, x2, y2: Single): TStatus; overload;
function DrawLine(Pen: TGPPen; const pt1, pt2: TGPPointF): TStatus; overload;
function DrawLine(Pen: TGPPen; const pt1, pt2: TGPPoint): TStatus; overload;
```

Для работы всем функциям необходимо передать перо Pen и координаты точек (форматом представления которых и различается каждый из членов перегружаемой четверки). Как видите, метод сразу устанавливает начальную и конечную точки прямой, за один раз объединяя обращения к функциям MoveToEx() и LineTo() из состава Windows API. Надо отметить, что в составе всех опубликованных методов TGPGraphics в принципе отсутствует выделенная функция, отвечающая за установку пера в заданных координатах поверхности графического устройства. Это объясняется тем, что любой из рисующих графические примитивы методов инкапсулирует обращение к позиционирующей функции MoveToEx().



Метод DrawLine() является наглядной демонстрацией системы программирования без информации о состоянии. В метод передаются все данные, требуемые для выполнения логически завершенной операции: перо и координаты начальной и конечной точки.

Для вывода ломаной линии вместо многократного обращения к DrawLine() целесообразно выбрать функцию:

```
function DrawLines(Pen: TGPPen; pPoints: PGPPoint;
                  Count: Integer): TStatus; overload;
function DrawLines(Pen: TGPPen; pPoints: PGPPointF;
                  Count: Integer): TStatus; overload;
```

На этот раз координаты точек содержатся в массиве, на который ссылается указатель pPoints. Последний параметр функции Count определяет число подлежащих выводу точек.

```
Procedure GDI_PAINT(DC:HDC; ClientRect:TRect);
var   Graphics:TGPGraphics;
      Points: array[0..9] of TGPPoint;
      Pen:TGPPen;
      i : byte;
begin
  Randomize; //инициализация генератора псевдослучайной последовательности
```

```

for i:=0 to 9 do {заполняем массив случайными значениями}
begin
  Points[i].X:=Random(ClientRect.Right);
  Points[i].Y:=Random(ClientRect.Bottom);
end;

Graphics:=TGPGraphics.Create(DC);
Pen:=TGPPen.Create(ac1Black);
Graphics.DrawLines(Pen,PGPPoint(@Points),10);
Pen.Free; Graphics.free;
end;

```

Точно по такому же алгоритму осуществляется вывод многоугольника, только теперь вместо метода `DrawLines()` нам потребуется целочисленный или вещественный вариант функции:

```

function DrawPolygon(Pen: TGPPen; pPoints: PGPPoint;
  Count: Integer): TStatus; overload;
function DrawPolygon(Pen: TGPPen; pPoints: PGPPointF;
  Count: Integer): TStatus; overload;

```

Для того чтобы ломаная линия превратилась в многоугольник, метод соединит первую и последнюю точки из массива.

Многоугольник (как, впрочем, и любая другая замкнутая фигура) может быть закрасен. Для этого предназначена группа методов, название которых начинается со слова «Fill». В случае многоугольника следует применять методы:

```

function FillPolygon(Brush: TGPBrush; pPoints: PGPPoint;
  Count: Integer): TStatus; overload;
function FillPolygon(Brush: TGPBrush; pPoints: PGPPointF;
  Count: Integer): TStatus; overload;
function FillPolygon(Brush: TGPBrush; pPoints: PGPPoint;
  Count: Integer; FillMode: TFillMode): TStatus; overload;
function FillPolygon(Brush: TGPBrush; pPoints: PGPPointF;
  Count: Integer; FillMode: TFillMode): TStatus; overload;

```

Естественно, что в операции заливки главное действующее лицо – кисть GDI+. Методы отличаются лишь способом определения вершин фигуры. Кроме того, последняя пара функций предоставляет возможность управлять режимом заливки, для этого предназначен аргумент `FillMode`. Параметр может принимать одно из двух значений – `FillModeAlternate` или `FillModeWinding`. В зависимости от установленного значения будет выбран порядок принятия решения, о том принадлежит ли точка фигуре, иначе говоря – стоит закрашивать точку или нет. Алгоритм нам не нов, достаточно вспомнить функцию GDI `SetPolyFillMode` (за подробностями возвращайтесь к табл. 7.1 главы «Простейшие геометрические фигуры и заливка областей»). На рис. 33.1 продемонстрировано влияние режима `TFillMode` на особенности заливки внутренних областей многоугольника сложной формы.

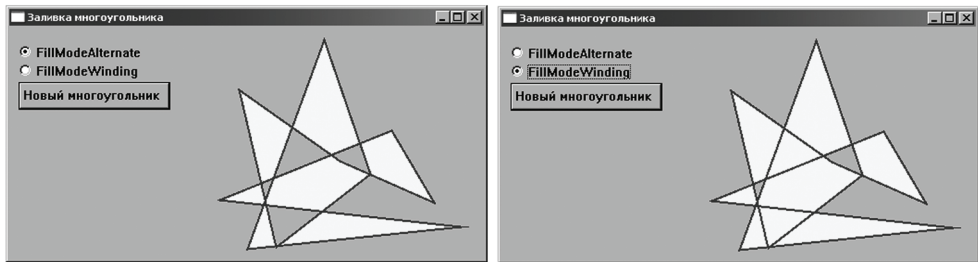


Рис. 33.1. Влияние режима *TFillMode* на закрашку внутренних областей многоугольника сложной формы

Кривые

В GDI+ имеется две разновидности кривых линий. Это унаследованная со времен безраздельного царствования GDI кривая Безье и усовершенствованная кривая GDI+, в некоторой литературе называемая *каноническим сплайном* [5]. Для вывода сплайна в стиле GDI допустимо воспользоваться любым из предложенных ниже методов:

```
function DrawBezier(Pen: TGPPen;
    x1, y1, x2, y2, x3, y3, x4, y4: Integer): TStatus; overload;
function DrawBezier(Pen: TGPPen;
    x1, y1, x2, y2, x3, y3, x4, y4: Single): TStatus; overload;
function DrawBezier(Pen: TGPPen;
    const pt1, pt2, pt3, pt4: TGPPoint): TStatus; overload;
function DrawBezier(Pen: TGPPen;
    const pt1, pt2, pt3, pt4: TGPPointF): TStatus; overload;
```

Метод `DrawBezier()` является аналогом функции Windows API `PolyBezier()`. Здесь первая и четвертая точка определяют начало и окончание линии, вторая и третья точки отклоняют линию от оси, превращая ее в сплайн.

При одновременном выводе нескольких сплайнов применяются методы:

```
function DrawBeziers(Pen: TGPPen; pPoints: PGPPoint;
    Count: Integer): TStatus; overload;
function DrawBeziers(Pen: TGPPen; pPoints: PGPPointF;
    Count: Integer): TStatus; overload;
```

Описания сплайнов загружаются в массив точек, на которые ссылается указатель `pPoints`. Заключительный параметр определяет число точек в подлежащем выводу массиве.

Построенные таким образом кривые обладают всеми характерными чертами старого доброго сплайна: кривая размещается внутри выпуклой фигуры с вершинами в определяющих точках кривой, и прямая, проведенная из конечной в ближайшую отклоняющую точку, всегда представляет собой касательную к сплайну.

В GDI+ предусмотрен альтернативный механизм прорисовки сплайна Безье, здесь принципиально изменилась роль отклоняющих точек. В методах:

```
function DrawCurve(Pen: TGPPen; pPoints: PGPPoint;  
                  Count: Integer): TStatus; overload;  
function DrawCurve(Pen: TGPPen; pPoints: PGPPointF;  
                  Count: Integer): TStatus; overload;
```

точки 2 и 3 не просто отводят кривую от прямолинейной траектории, а явным образом определяют траекторию кривой. Простейшие версии перегружаемого метода требуют от программиста трех параметров: пера Pen, указателя на массив точек кривой pPoints и числа точек, подлежащих выводу, — Count. Дальше становится интереснее — на свет появляется параметр Tension, влияющий на степень натяжения кривой:

```
function DrawCurve(Pen: TGPPen; pPoints: PGPPoint;  
                  Count: Integer; Tension: Single): TStatus; overload;  
function DrawCurve(Pen: TGPPen; pPoints: PGPPointF;  
                  Count: Integer; Tension: Single): TStatus; overload;
```

Благодаря параметру Tension поведение канонического сплайна GDI+ можно сравнить с поведением туго натянутой струны. По умолчанию струна не напряжена (Tension=0.5). При нулевом натяжении кривая превращается в обычную ломаную, а при увеличении степени натяжения можно добиться весьма забавных визуальных эффектов (но порвать струну у нас не получится).

В апогее сложности метод потребует еще два параметра. Это отступ Offset, позволяющий начать вывод кривой с любой точки массива, и NumberOfSegments, указывающий количество сегментов в сплайне.

```
function DrawCurve(Pen: TGPPen; pPoints: PGPPoint;  
                  Count, Offset, NumberOfSegments: Integer;  
                  Tension: Single = 0.5): TStatus; overload;  
function DrawCurve(Pen: TGPPen; pPoints: PGPPointF;  
                  Count, Offset, NumberOfSegments: Integer;  
                  Tension: Single = 0.5): TStatus; overload;
```

Если вы читаете эти строки за компьютером, то настоятельно рекомендую поэкспериментировать с обычным и каноническим сплайнами (точнее, с рисующими их методами). Особое внимание следует уделить различиям во взглядах методов DrawBezier() и DrawCurve() на роль отклоняющих точек в процессе построения сплайна. Если же под рукой компьютера нет, то некоторое представление об этих видах кривых можно получить, просмотрев экранные снимки форм (рис. 33.2).

Верхний ряд снимков демонстрирует поведение классических методов построения кривых — отклоняющие точки лишь оттягивают кривую от прямолинейной траектории. Совсем другой результат получается при применении метода DrawCurve(). В этом случае отклоняющие точки выступают не просто в роли «потусторонней» силы, а явным образом определяют форму канонического сплайна (см. нижний ряд экранных снимков на рис. 33.2). Полагаю, что новый способ формирования кривой в большей степени понятен обычно-


```

function DrawClosedCurve(Pen: TGPPen; pPoints: PGPPointF;
    Count: Integer): TStatus; overload;
function DrawClosedCurve(Pen: TGPPen; pPoints: PGPPoint;
    Count: Integer; Tension: Single): TStatus; overload;
function DrawClosedCurve(Pen: TGPPen; pPoints: PGPPointF;
    Count: Integer; Tension: Single): TStatus; overload;

```

Аргументы функций аналогичны параметрам `DrawCurve()`, поэтому на них мы останавливаться не станем.

Осталось отметить последний метод холста GDI+, имеющий прямое отношение к работе со сплайнами. На этот раз мы говорим о заливке области, ограниченной замкнутой кривой. С этой целью также реализовано 4 метода:

```

function FillClosedCurve(Brush: TGPBrush; pPoints: PGPPointF;
    Count: Integer): TStatus; overload;
function FillClosedCurve(Brush: TGPBrush; pPoints: PGPPoint;
    Count: Integer): TStatus; overload;
function FillClosedCurve(Brush: TGPBrush; pPoints: PGPPoint;
    Count: Integer; FillMode: TFillMode;
    Tension: Single = 0.5): TStatus; overload;
function FillClosedCurve(Brush: TGPBrush; pPoints: PGPPointF;
    Count: Integer; FillMode: TFillMode;
    Tension: Single = 0.5 ): TStatus; overload;

```

В двух последних методах требуется определить режим заливки `FillMode`.

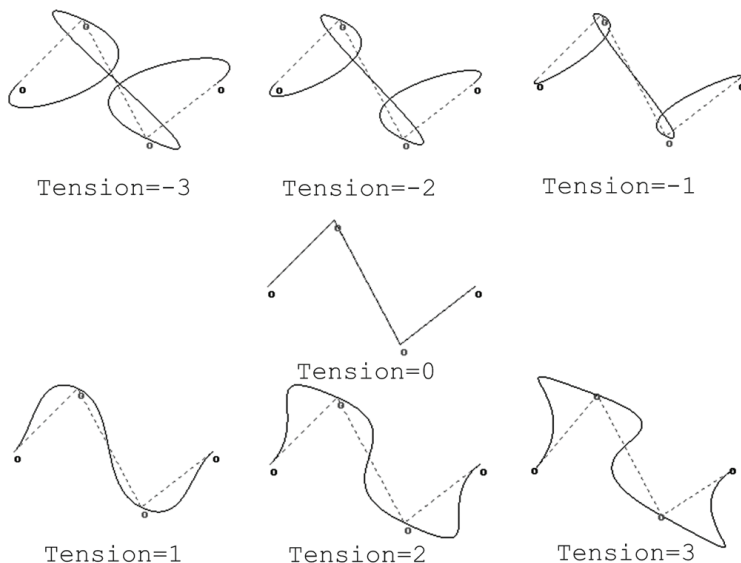


Рис. 33.3. Управление натяжением сплайна GDI+

Простейшие фигуры

Для указания координат вывода большинства простейших фигур, изображение которых вписывается в прямоугольную область, требуется однотипный перечень параметров. В GDI+ предусмотрено 4 варианта аргументов, определяющих местоположение ограничивающего прямоугольника:

1. Прямоугольная область с полями целых чисел – Rect: TGPRect.
2. Прямоугольная область с полями вещественных чисел – Rect: TGPRectF.
3. Координаты левой верхней точки, ширина и высота – X, Y, Width, Height: Integer.
4. Координаты левой верхней точки, ширина и высота в виде вещественных чисел – X, Y, Width, Height: Single.

Именно этими аргументами и отличаются друг от друга все четыре версии перегружаемых функций. В табл. 33.1 представлены названия методов, осуществляющих рисование и заливку прямоугольника и эллипса.

Таблица 33.1. Методы вывода прямоугольника и эллипса

Название метода	Описание
DrawRectangle()	Вывод контура прямоугольника.
FillRectangle()	Заливка прямоугольника.
DrawEllipse()	Вывод контура эллипса.
FillEllipse()	Заливка эллипса.

Вывод на поверхность холста нескольких прямоугольников осуществляет следующая группа методов:

```
function DrawRectangles(Pen: TGPPen;
    pRects: PGPRect; Count: Integer): TStatus; overload;
function DrawRectangles(Pen: TGPPen;
    pRects: PGPRectF; Count: Integer): TStatus; overload;
```

Координаты прямоугольников передаются через указатель на массив pRects. Последний параметр функции определяет количество рисуемых фигур. Точно такой же перечень аргументов содержат методы, осуществляющие заливку прямоугольников.

```
function FillRectangles(Brush: TGPBrush;
    pRects: PGPRect; Count: Integer): TStatus; overload;
function FillRectangles(Brush: TGPBrush;
    pRects: PGPRectF; Count: Integer): TStatus; overload;
```

Ниже вашему вниманию предложена процедура, демонстрирующая порядок работы с методами рисования группы прямоугольников.

```
Procedure GDIPLUS_PAINT(DC:HDC);
var   Pen : TGPPen; Brush : TGPHatchBrush;
      i : integer;
```

```

    R : array [0..9] of TGPRect;
    Graphics : TGPGraphics;
begin
    Graphics:=TGPGraphics.Create(DC);

    for i:=0 to 9 do
        R[i]:=MakeRect(5+i*20,20*i+18,30,30);

    Pen:=TGPPen.Create(ac1Red,2);
    Graphics.DrawRectangles(Pen,pGPRect(@R),10);
    Pen.Free;

    Brush:=TGPHatchBrush.Create(HatchStyleDashedDownwardDiagonal,ac1Green,ac1Blue);
    Graphics.FillRectangles(Brush,pGPRect(@R),10);
    Brush.Free;

    Graphics.free;
end;
```

Дуга и сектор окружности

В основу идеи построения дуги и сектора окружности также положены координаты ограничивающего прямоугольника. У каждого из перечисленных в табл. 33.2 метода есть четыре перегружаемых версии, отличающиеся друг от друга способом описания местоположения прямоугольника. Кроме того, у каждого из представленных в таблице методов предусмотрено еще два завершающих параметра: *StartAngle*, *SweepAngle*: *Single*, отвечающих за определение углов раскрытия дуги или сектора.

Таблица 33.2. Методы вывода дуги и сектора

Название метода	Описание
<code>DrawArc()</code>	Вывод дуги.
<code>DrawPie()</code>	Вывод сектора окружности.
<code>FillPie()</code>	Заливка сектора окружности.

Рассмотрим особенности работы с перечисленной группой функций на примере метода вывода дуги:

```

function DrawArc(Pen: TGPPen; X, Y, Width, Height,
    StartAngle, SweepAngle: Single): TStatus; overload;
```

В процессе вывода дуги задействована функция `AngleArc()` классического GDI (см. главу 6 «Линии и кривые»), только несколько изменен порядок применения параметров. В рассматриваемом методе `DrawArc()` точка с координатами (X,Y) определяет левый верхний угол прямоугольника с шириной *Width* и высотой *Height*. Дуга рисуется по контуру вписанного в прямоугольник эллипса. Стартовая точка дуги определяется точкой пересечения контура эллипса с лучом, направляемым из центра эллипса под углом *StartAngle* (в градусах) к оси X. Размах дуги назначается другим лучом, также направленным из центра эллипса, но теперь под углом раскрытия *SweepAngle* отно-

сительно стартового луча (рис. 33.4). Ко всему сказанному осталось добавить, что величина угла возрастает по часовой стрелке, так что если необходимо изменить направление вывода – передавайте в последний параметр функции отрицательное значение.

Еще раз напомним, что вывод сектора окружности осуществляется функцией `DrawPie()`, методы сектора снабжены идентичным перечнем аргументов, роль которых повторяет назначение аргументов, использующихся в функции рисования дуги. Ко всему прочему, сектор круга (как замкнутую фигуру) допускается закрасить кистью с помощью метода `FillPie()`. В предложенном ниже примере продемонстрирован порядок работы с сектором окружности.

```
Procedure GDI_PAINT(DC:HDC);
var   Pen:TGPpen;
      Brush:TGPBrush;
      Graphics:TGPGraphics;
const X=10;Y=10; Width=250; Height=150;
begin
  Graphics:=TGPGraphics.Create(DC);
  Pen:=TGPpen.Create(ac1Black,1);
  Brush:=TGPBrush.Create(HatchStyleHorizontalBrick,ac1Yellow,ac1Red);

  Graphics.DrawRectangle(Pen,X,Y,Width,Height);
  Graphics.DrawPie( Pen,X,Y,Width,Height,280,90);
  Graphics.FillPie( Brush,X,Y,Width,Height,10,270);

  Pen.Free;
  Brush.Free;
  Graphics.free;
end;
```

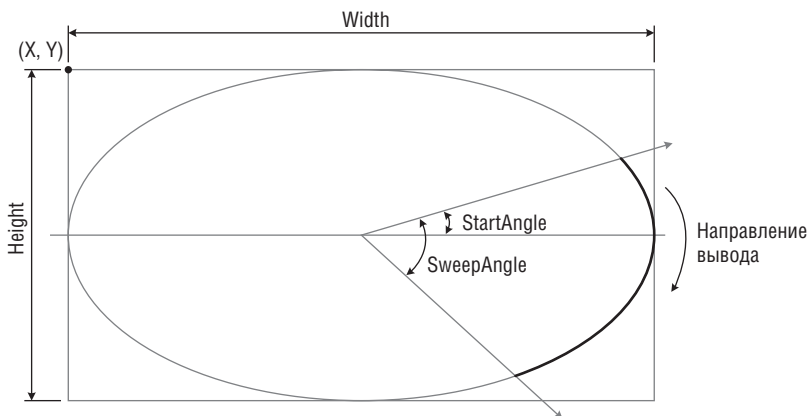


Рис. 33.4. Вывод дуги методом `DrawArc()`

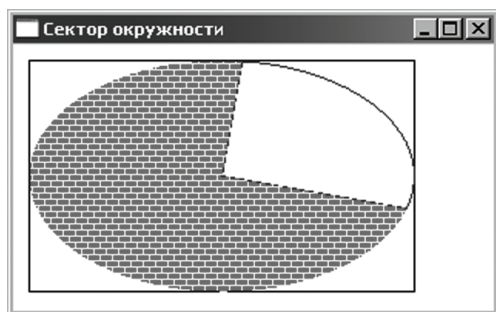


Рис. 33.5. Пример вывода сектора окружности

По ходу процедуры мы последовательно выводим контур ограничивающего прямоугольника, затем контур сектора и в заключение закрашиваем кистью с «кирпичным» стилем оставшуюся часть сектора окружности.

РЕЗЮМЕ

В GDI+ решение вопросов, связанных с выводом графических примитивов и заливкой замкнутых областей, возложено на холст `TGPGraphics`. Хотя функционал класса целиком и полностью построен на базе функций Windows API, холст GDI+ может похвастаться существенным новшеством – усовершенствованным способом описания кривых Безье.

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 5-93286-134-7, название «Графика в проектах Delphi» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (piracy@symbol.ru), где именно Вы получили данный файл.