HIGH TECH

ДМИТРИЙ ОСИПОВ

ГРАФИКА В ПРОЕКТАХ DELPHI

СИМВОЛ®

HIGH TECH

Графика в проектах Delphi

Дмитрий Осипов

Санкт-Петербург — Москва
2008

Серия «High Tech»

Дмитрий Осипов

Графика в проектах Delphi

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

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

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 70×100^{1}/$_{16}$. 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

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` определется 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 ребя-

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 ломаные 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(aclBlack);
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 закрашен. 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.

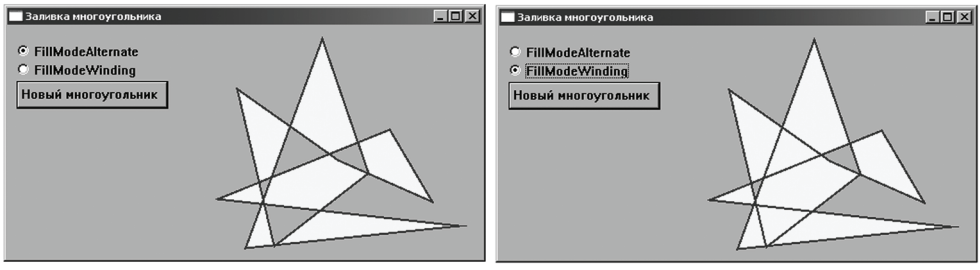

Рис. 33.1. Влияние режима TFillMode на закраску внутренних областей многоугольника сложной формы

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-

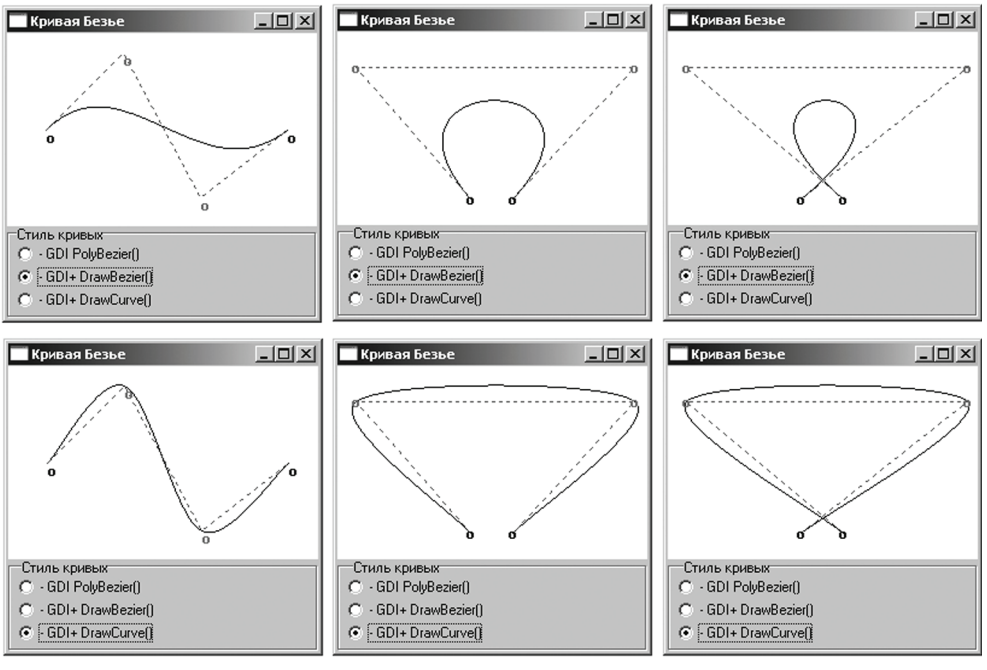

Рис. 33.2. Различия в способах вывода кривых Безье в GDI и GDI+

му пользователю. Отмечу, что нижний ряд рисунков получен с параметром `Tension`, равным 0,5 (значение по умолчанию), малейшие модификации значения степени натяжения кривой способны изменить форму кривой до неузнаваемости (рис. 33.3).

Не затрагивая положения основных и отклоняющих точек, мы управляем натяжением сплайна, изменяя коэффициент `Tension` в диапазоне от –3 до 3 с единичным шагом. Интересно поведение кривой при отсутствии напряженности (нулевое значение), в этом случае сплайн превращается в ломаную линию. При отрицательных значениях коэффициента наша кривая постепенно превращается в «бабочку», сворачиваясь вовнутрь области, ограниченной управляющими точками. И наоборот, переходя в положительный диапазон натяжения, сплайн постепенно выступает за пределы законной области кривой.

В отличие от кубических кривых Безье в GDI канонические сплайны в GDI+ не находятся внутри фигуры с вершинами в определяющих точках кривой. Эта черта сплайнов GDI+ становится особо заметна при увеличении степени натяжения.

На выводе замкнутой кривой (с характером натянутой струны) специализируются еще четыре метода:

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

```
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`.

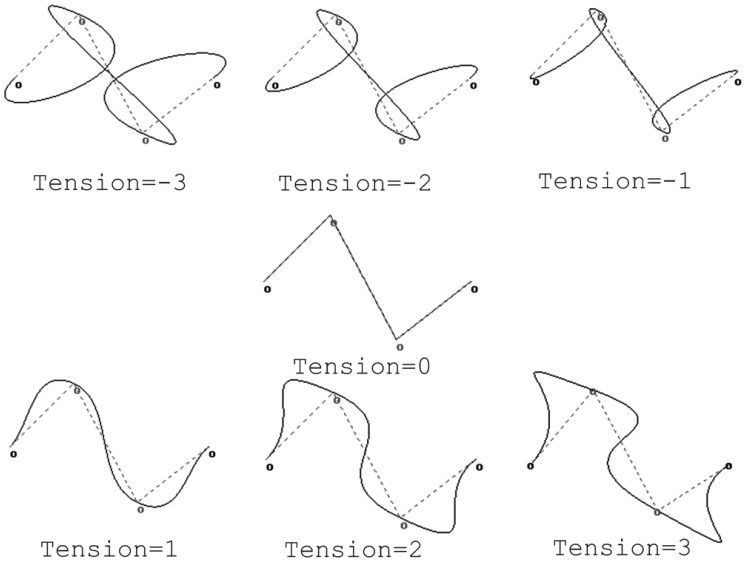

Рис. 33.3. Управление натяжением сплайна GDI+

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(aclRed,2);
Graphics.DrawRectangles(Pen,pGPRect(@R),10);
Pen.Free;

Brush:=TGPHatchBrush.Create(HatchStyleDashedDownwardDiagonal,aclGreen,aclBlue);
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
`DrawArc()`	Вывод дуги.
`DrawPie()`	Вывод сектора окружности.
`FillPie()`	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 напомню, 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:TGPHatchBrush;
      Graphics:TGPGraphics;
const X=10;Y=10; Width=250; Height=150;
begin
Graphics:=TGPGraphics.Create(DC);
 Pen:=TGPPen.Create(aclBlack,1);
 Brush:=TGPHatchBrush.Create(HatchStyleHorizontalBrick,aclYellow,aclRed);

 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;
```

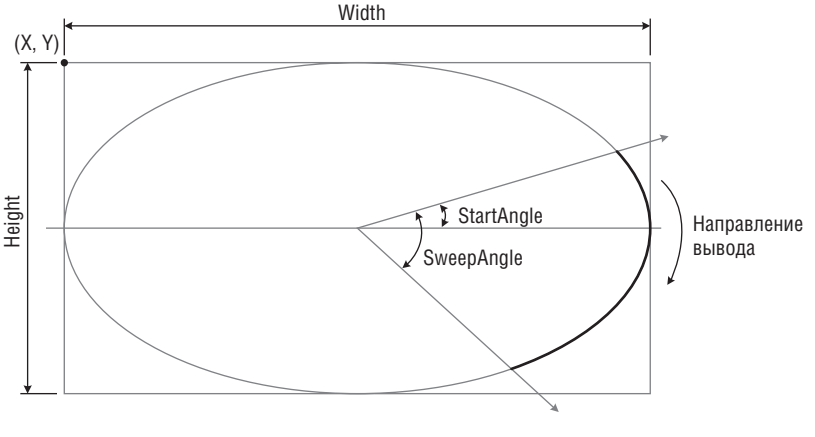

Рис. 33.4. Вывод дуги методом DrawArc( )

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.