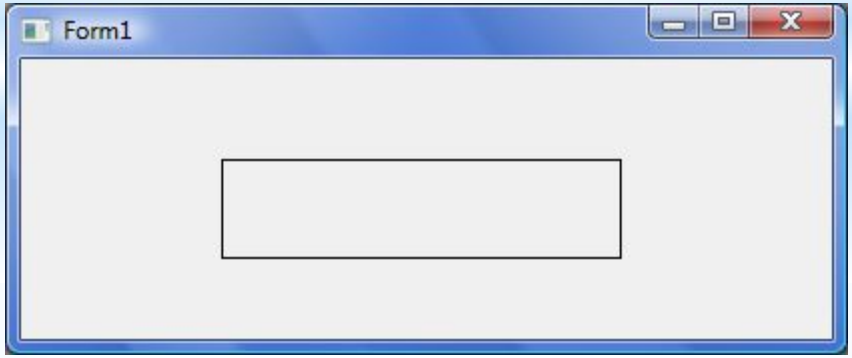
Графические возможности Lazarus

Delphi позволяет программисту разрабатывать программы, которые могут выводить графику: схемы, чертежи, иллюстрации.

Программа выводит графику на поверхность объекта (формы **Form** или компонента **Image**). Поверхности объекта соответствует свойство **Canvas (Холст)**. Для того чтобы вывести на поверхность объекта графический элемент (прямую линию, окружность, прямоугольник и т. д.), необходимо применить к свойству **Canvas** этого объекта соответствующий метод. Например, инструкция **Form1.Canvas.Rectangle(100,50,300,100)** вычерчивает в окне программы прямоугольник.



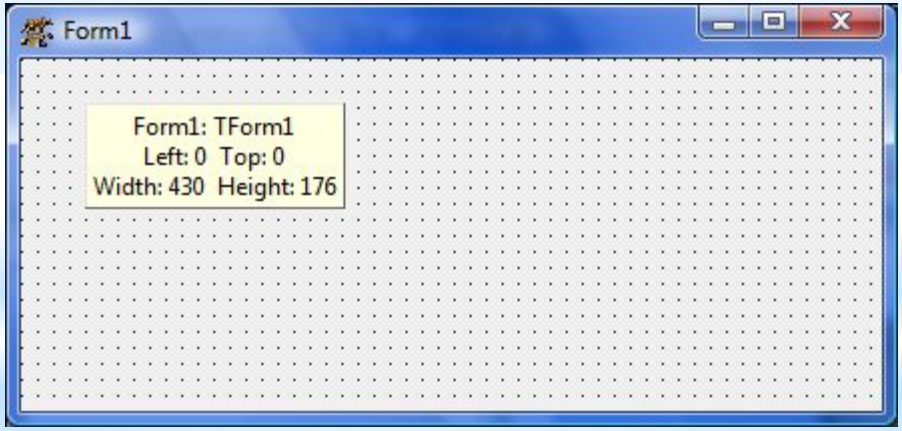
* **ВНИМАНИЕ!!!**
* Если требуется чтобы методы рисования применились при щелчке мыши по форме – используйте событие **OnClick**
* Если требуется чтобы методы рисования применились при создании формы (при запуске приложения), то НЕ следует пользоваться событием **OnCreat** (оно не поможет). Следует использовать событие **OnPaint** (прорисовка). Например:

```
* procedure TForm1.FormPaint(Sender: TObject);
* begin
*       Form1.Canvas.Rectangle(100,50,300,100);
* end;
```

Холст

Canvas – это **свойство** объекта **Form**, т.е. холст находящийся на форме, который в свою очередь является **объектом** типа **TCanvas**. И у объекта **Canvas** есть свои **методы**, позволяющие рисовать на холсте фигуры: прямоугольники, круги и т. д. А также у объекта **Canvas** есть свои **свойства**, позволяющие задать стиль фигур: толщину линий, цвет линий, цвет заливки и т.д.

Холст состоит из отдельных точек — **пикселов**. Положение пиксела характеризуется его горизонтальной (X) и вертикальной (Y) координатами. Левый верхний пиксел имеет координаты (0,0). Координаты возрастают сверху вниз и слева направо (вспоминаем тему Pascal – Модуль Graph). Значения координат правой нижней точки холста зависят от размера холста.

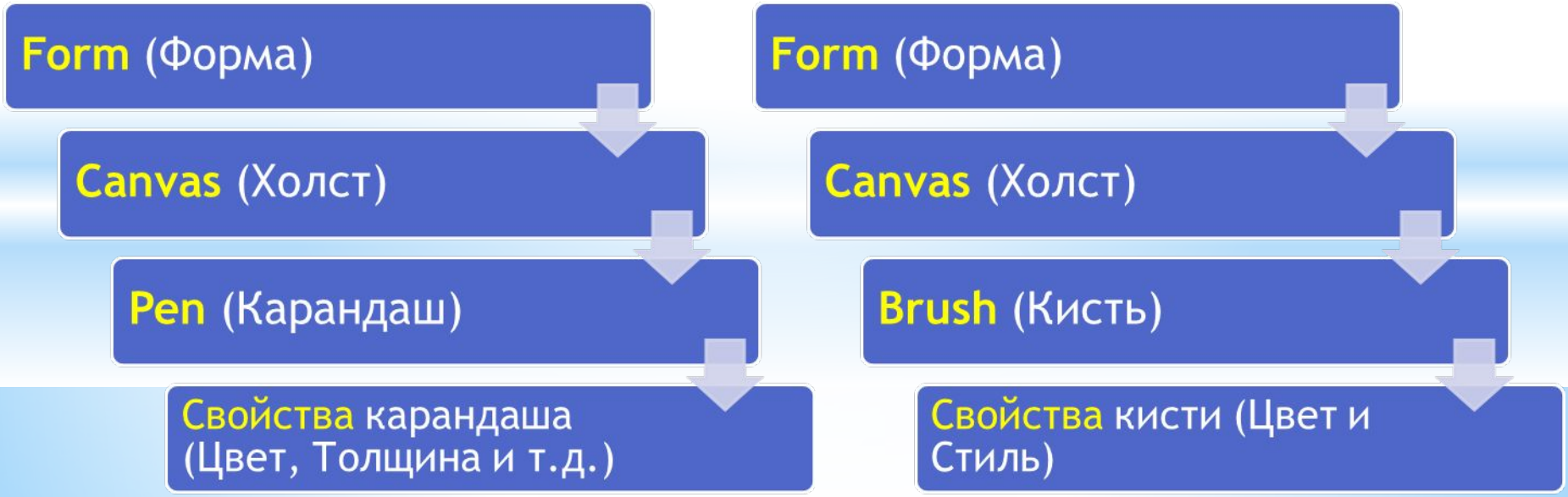


Размер холста можно узнать обратившись к свойствам **Height** и **Width** объекта на котором происходит рисование.

Карандаш и кисть

Художник в своей работе использует карандаши и кисти. Методы, обеспечивающие вычерчивание на поверхности холста графических примитивов, тоже используют карандаш и кисть. **Карандаш** применяется для вычерчивания линий и контуров, а **кисть** — для закрашивания областей, ограниченных контурами.

Карандашу и кисти, используемым для вывода графики на холсте, соответствуют свойства **Pen** (карандаш) и **Brush** (кисть), которые представляют собой объекты типа TPen и TBrush, соответственно. Значения свойств этих объектов определяют вид выводимых графических элементов..



Карандаш (Pen)

Карандаш (Canvas.Pen) используется для вычерчивания точек, линий, контуров геометрических фигур: прямоугольников, окружностей, эллипсов, дуг и др. **Вид линии**, которую оставляет карандаш на поверхности холста, определяют **свойства объекта Pen**:

Свойство	Определяет
Color	Цвет линии
Width	Толщину линии
Style	Вид линии
Mode	Режим отображения

Константа	Цвет
clBlack	Черный
clRed	Красный
clGreen	Зеленый
clWhite	Белый
clNavy	Темно-синий
clBlue	Синий
clGray	Серый

Константа	Вид линии
psSolid	Сплошная линия
psDash	Пунктирная линия, длинные штрихи
psDot	Пунктирная линия, короткие штрихи
psDashDot	Пунктирная линия, чередование длинного и короткого штрихов
psDashDotDot	Пунктирная линия, чередование одного длинного и двух коротких штрихов
psClear	Линия не отображается (используется, если не надо изображать границу области, например, прямоугольника)

Свойство **Mode** определяет, как будет формироваться цвет точек линии в зависимости от цвета точек холста, через которые эта линия прочерчивается. По умолчанию вся линия вычерчивается цветом, определяемым значением свойства **Pen.Color**

Однако программист может задать инверсный цвет линии по отношению к цвету фона. Это гарантирует, что независимо от цвета фона все участки линии будут видны, даже в том случае, если цвет линии и цвет фона совпадают.

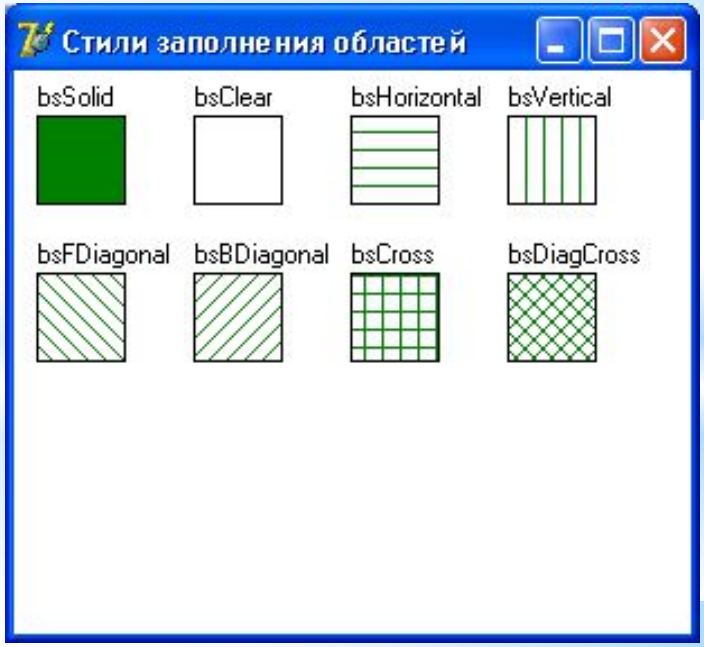
Константа	Цвет линии
pmBlack	Черный, не зависит от значения свойства Pen.Color
pmWhite	Белый, не зависит от значения свойства Pen.Color
pmCopy	Цвет линии определяется значением свойства Pen .Color
pmNotCopy	Цвет линии является инверсным по отношению к значению свойства Pen.Color
pmNot	Цвет точки линии определяется как инверсный по отношению к цвету точки холста, в которую выводится точка линии

Кисть (Brush)

Кисть (Canvas.Brush) используется методами, обеспечивающими вычерчивание замкнутых областей, например геометрических фигур, для заливки (закрашивания) этих областей. Кисть, как объект, обладает двумя свойствами

Свойство	Определяет
Color	Цвет закрашивания замкнутой области
Style	Стиль (тип) заполнения области

Константа	Тип заполнения (заливки) области
bsSolid	Сплошная заливка
bsClear	Область не закрашивается
bsHorizontal	Горизонтальная штриховка
bsVertical	Вертикальная штриховка
bsFDiagonal	Диагональная штриховка с наклоном линий вперед
bsBDiagonal	Диагональная штриховка с наклоном линий назад
bsCross	Горизонтально-вертикальная штриховка, в клетку
bsDiagCross	Диагональная штриховка, в клетку



Вывод текста

Для вывода текста на поверхность графического объекта используется метод TextOut. Инструкция вызова метода TextOut в общем виде выглядит следующим образом:

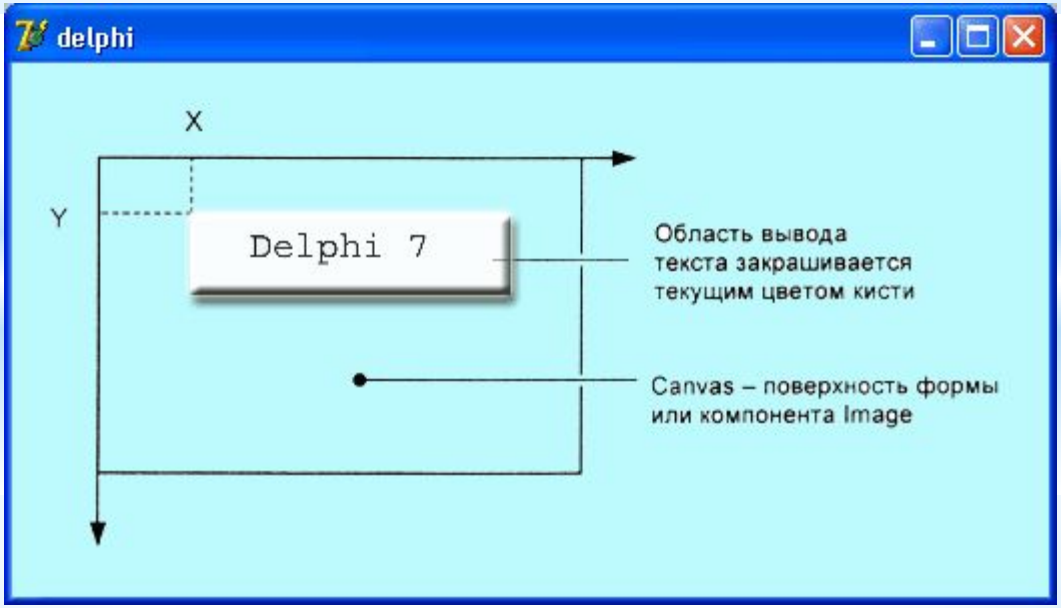
Объект.Canvas.TextOut(x, y, Текст)

Где:

объект — имя объекта, на поверхность которого выводится текст;

x, y — координаты точки графической поверхности, от которой выполняется вывод текста;

Текст — переменная или константа символьного типа, значение которой определяет выводимый методом текст.



Методы вычерчивания графических примитивов

Линия.
Вычерчивание прямой линии осуществляет метод LineTo, инструкция вызова которого в общем виде выглядит следующим образом:

Form1.Canvas.LineTo(x2,y2)

Метод LineTo вычерчивает прямую линию от текущей позиции карандаша в точку с координатами, указанными при вызове метода.

Начальную точку линии можно задать, переместив карандаш в нужную точку графической поверхности. Сделать это можно при помощи метода **MoveTo**, указав в качестве параметров координаты нового положения карандаша.

Form1.Canvas.MoveTo(x1,y1)

Вид линии (цвет, толщина и стиль) определяется значениями свойств объекта **Pen** графической поверхности, на которой вычерчивается линия.

Ломаная линия

Метод **Polyline** вычерчивает ломаную линию. В качестве параметра метод получает массив типа **TPoint**. Каждый элемент массива представляет собой запись, поля **X** и **Y** которой содержат координаты точки перегиба ломаной. Метод Polyline вычерчивает ломаную линию, последовательно соединяя прямыми точки, координаты которых находятся в массиве: первую со второй, вторую с третьей, третью с четвертой и т. д.

Метод **Polyline(P)** можно использовать для вычерчивания замкнутых контуров. Для этого надо, чтобы первый и последний элементы массива **P** содержали координаты одной и той же точки. (Если массив точек заполнен, то метод применяется: **Form1.Canvas.Polyline(P)**)

Если переменные A и B (точки A и B) описаны как:

```
Var  A,B: TPoint;
```

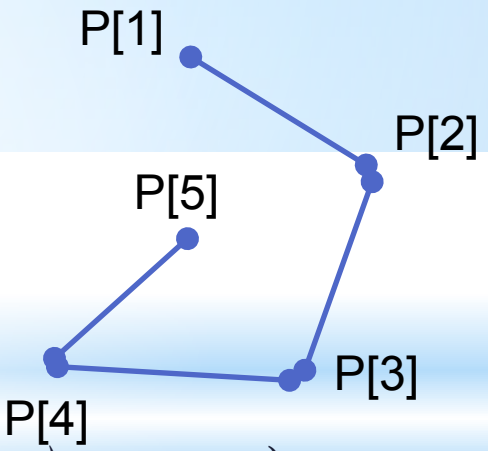
То координаты для точек A и B задаются так:

```
A.X := 50;        {Координаты для точки A}
A.Y := 30;

B.X := 120;       {Координаты для точки B}
B.Y := 150;
```

А если описан массив точек:

```
Var  P: array [1..10] of Tpoint;
```



То координаты для каждого элемента массива (каждой точки) может задаваться в цикле по какому либо закону:

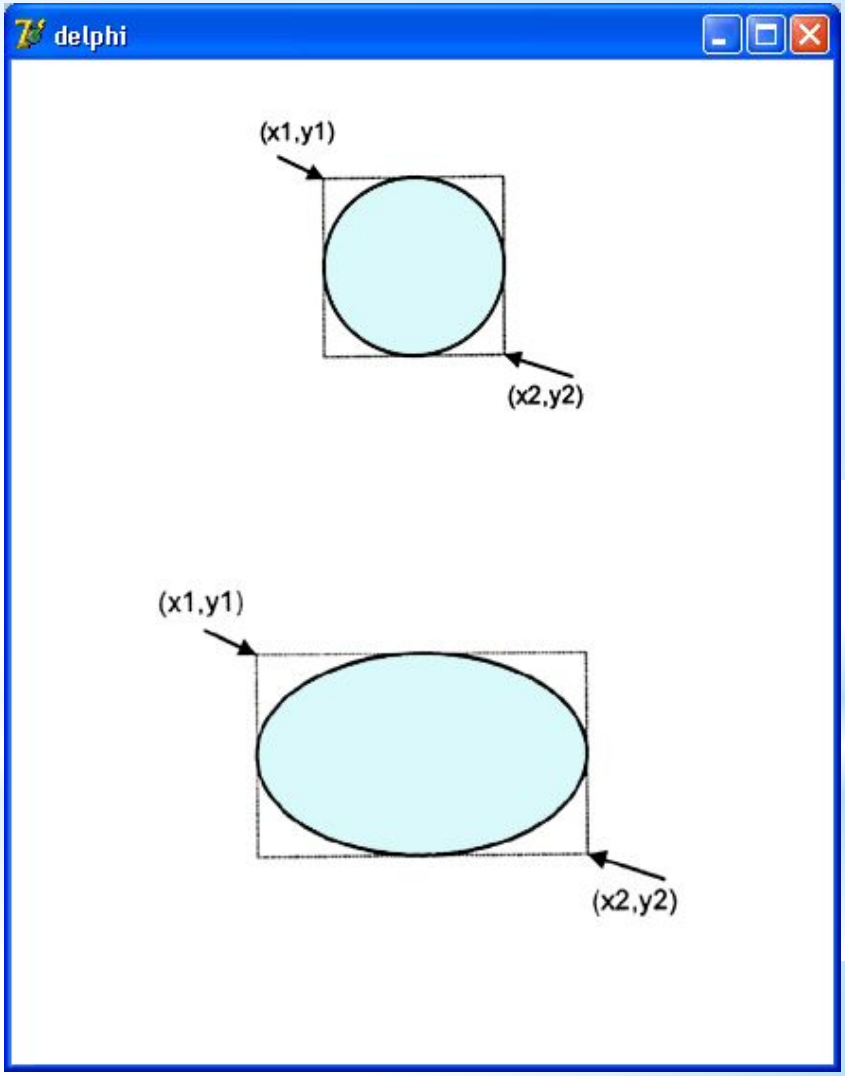
```
for  I := 1  to  10  do  begin
        P[ I ].X := …
        P[ I ].Y := …
        end;
```

Окружность и эллипс

Метод Ellipse вычерчивает эллипс или окружность, в зависимости от значений параметров. Инструкция вызова метода в общем виде выглядит следующим образом:

Объект.Canvas.Ellipse(x1,y1, x2,y2)

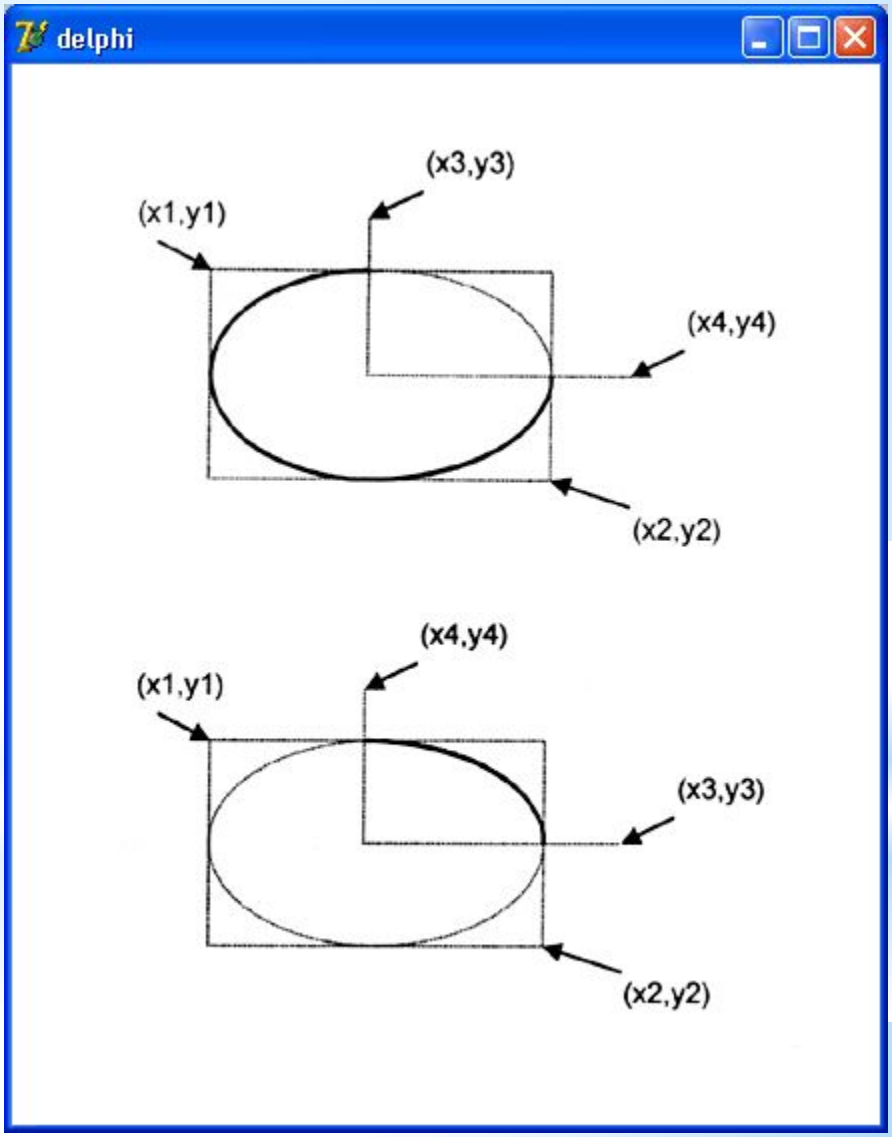
Цвет, толщина и стиль линии эллипса определяются значениями свойства Pen, а цвет и стиль заливки области внутри эллипса — значениями свойства Brush поверхности (Canvas), на которую выполняется вывод.



Дуга

Вычерчивание дуги выполняет метод Arc, инструкция вызова которого в общем виде выглядит следующим образом:

Объект.Canvas.Arc
(x1,y1,x2,y2,x3,y3,x4,y4)



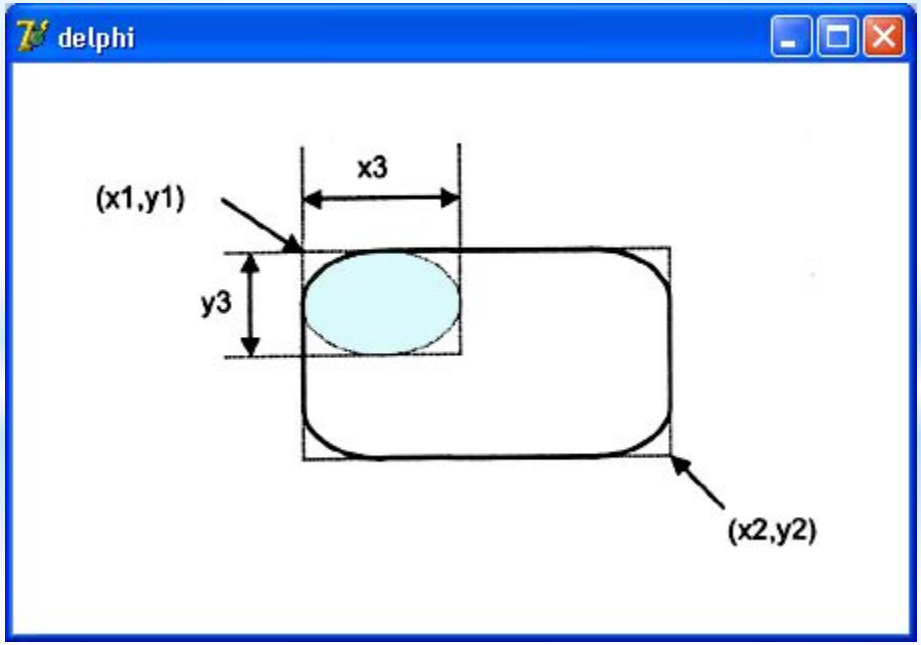
Прямоугольник

Прямоугольник вычерчивается методом Rectangle, инструкция вызова которого в общем виде выглядит следующим образом:

Объект.Canvas.Rectangle(x1, y1,x2, y2)

Метод RoundRec тоже вычерчивает прямоугольник, но со скругленными углами. Инструкция вызова метода RoundRec выглядит так:

Объект.Canvas.RoundRect(x1,y1,x2, y2, x3, y3)

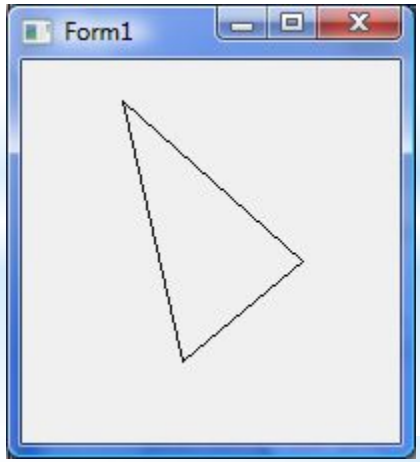


Многоугольник

Метод **Polygon** вычерчивает многоугольник. В качестве параметра метод получает массив типа TPoint. Каждый элемент массива представляет собой запись, поля (x,y) которой содержат координаты одной вершины многоугольника. Метод Polygon вычерчивает многоугольник, последовательно соединяя прямыми линиями точки, координаты которых находятся в массиве: первую со второй, вторую с третьей, третью с четвертой и т. д. Затем соединяются последняя и первая точки.

Ниже приведена процедура, которая, используя метод polygon, вычерчивает треугольник:

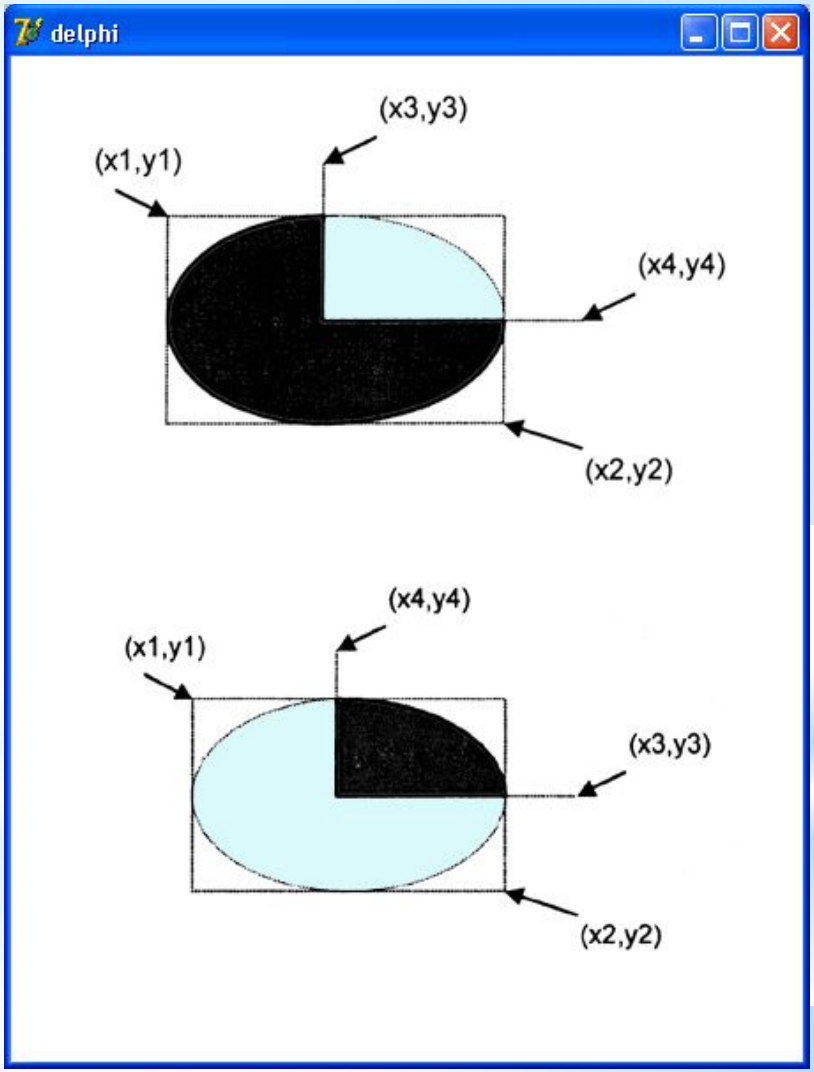
```
procedure TForm1.FormClick(Sender: TObject);      //процедура щелчка мыши по форме
Var  pol: array[1..3] of TPoint;          // координаты точек многоугольника
begin
pol[1].x := 10;
pol[1].y := 50;
pol[2].x := 40;
pol[2].y := 10;
pol[3].x := 70;
pol[3].y := 50;
Form1.Canvas.Polygon( pol ) ;
end;
```



Сектор

Метод **pie** вычерчивает сектор эллипса или круга. Инструкция вызова метода в общем виде выглядит следующим образом:

Объект. Canvas.Pie(x1,y1,x2,y2,x3,y3,x4,y4)



Точка

Поверхности, на которую программа может осуществлять вывод графики, соответствует объект **Canvas**. Свойство **Pixels**, представляющее собой двумерный массив типа **TColor**, содержит информацию о цвете каждой точки графической поверхности. Используя свойство **Pixels**, можно задать требуемый цвет для любой точки графической поверхности, т. е. "нарисовать" точку. Например, инструкция

Form1.Canvas.Pixels[10,10] := clRed

окрашивает точку поверхности формы в красный цвет.

Левой верхней точке рабочей области формы соответствует элемент **Pixels [0,0]**, а правой нижней – **Pixels [CiientWidth - 1, ClientHeight - 1]**.

* http://teacher.ucoz.net/Lection/Pascal/glava_10.pdf
* http://www.titorov.ru/index.php/distant/programmirovanie/546-graph
* http://hi-intel.ru/800/113.html