Система визуального объектно-ориентированного программирования Delphi

Графические возможности Delphi

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

Программа выводит графику на *поверхность* объекта. Поверхности объекта соответствует свойство **canvas**. 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.

Canvas (Холст)

Поверхности, на которую программа может выводить графику, соответствует свойство Canvas (это объект типа Tcanvas).

Методы этого типа обеспечивают вывод графических примитивов (точек, линий, окружностей, прямоугольников и т. д.), а свойства позволяют задать характеристики выводимых графических примитивов: цвет, толщину и стиль линий; цвет и вид заполнения областей; характеристики шрифта при выводе текстовой информации.

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

Карандаш (Pen) и кисть (Brush)

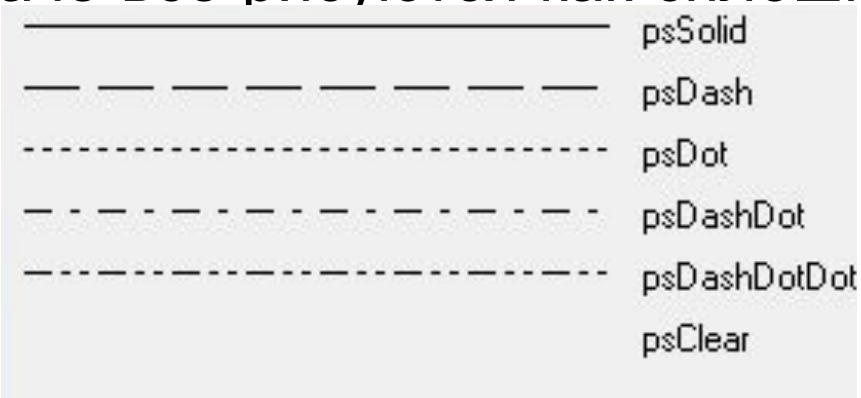
Карандаш (Pen) применяется для вычерчивания линий и контуров, а кисть (Brush) — для закрашивания областей, ограниченных контурами.

Рисование с помощью пера (Pen). Свойства:

Color – цвет пера;

Width – ширина линии в пикселях (по умолчанию 1);

Style – стиль линий (доступны только при ширине равной 1, иначе все рисуются как сплошные).



Mode определяет, как будет формироваться цвет точек линии в зависимости от цвета точек холста, через которые эта линия прочерчивается.

По умолчанию значение **Mode = pmCopy**. Это означает, что линии проводятся цветом, заданным в свойстве **Color**.

Но возможны и другие режимы, в которых учитывается не только цвет **Color**, но и цвет соответствующих пикселей фона.

Наиболее интересным из этих режимов является режим **pmNotXor** — сложение с фоном по инверсному исключающему ИЛИ. Если задан этот режим, то повторное рисование той же фигуры на том же месте канвы убирает ранее нарисованное изображение и восстанавливает цвета пикселей, которые были до первого изображения фигуры.

Эту особенность режима **pmNotXor** можно использовать для создания простенькой анимации.

Свойства для перемещения по канве

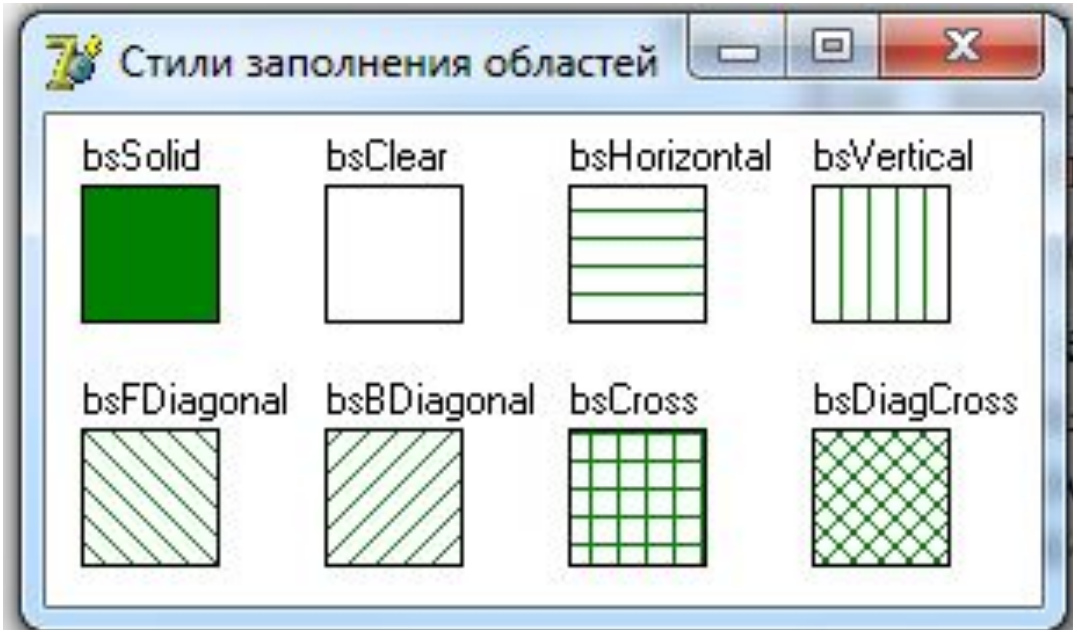
- ***PenPos*** – координаты текущего положения пера;
- ***MoveTo(x,y)*** – перемещение пера без прорисовки;
- ***LineTo(x,y)*** – линия из текущих координат до точки (x,y).

Работа с кистью (Brush)

Определяет фон и заполнение замкнутых фигур на канве. Основные свойства:

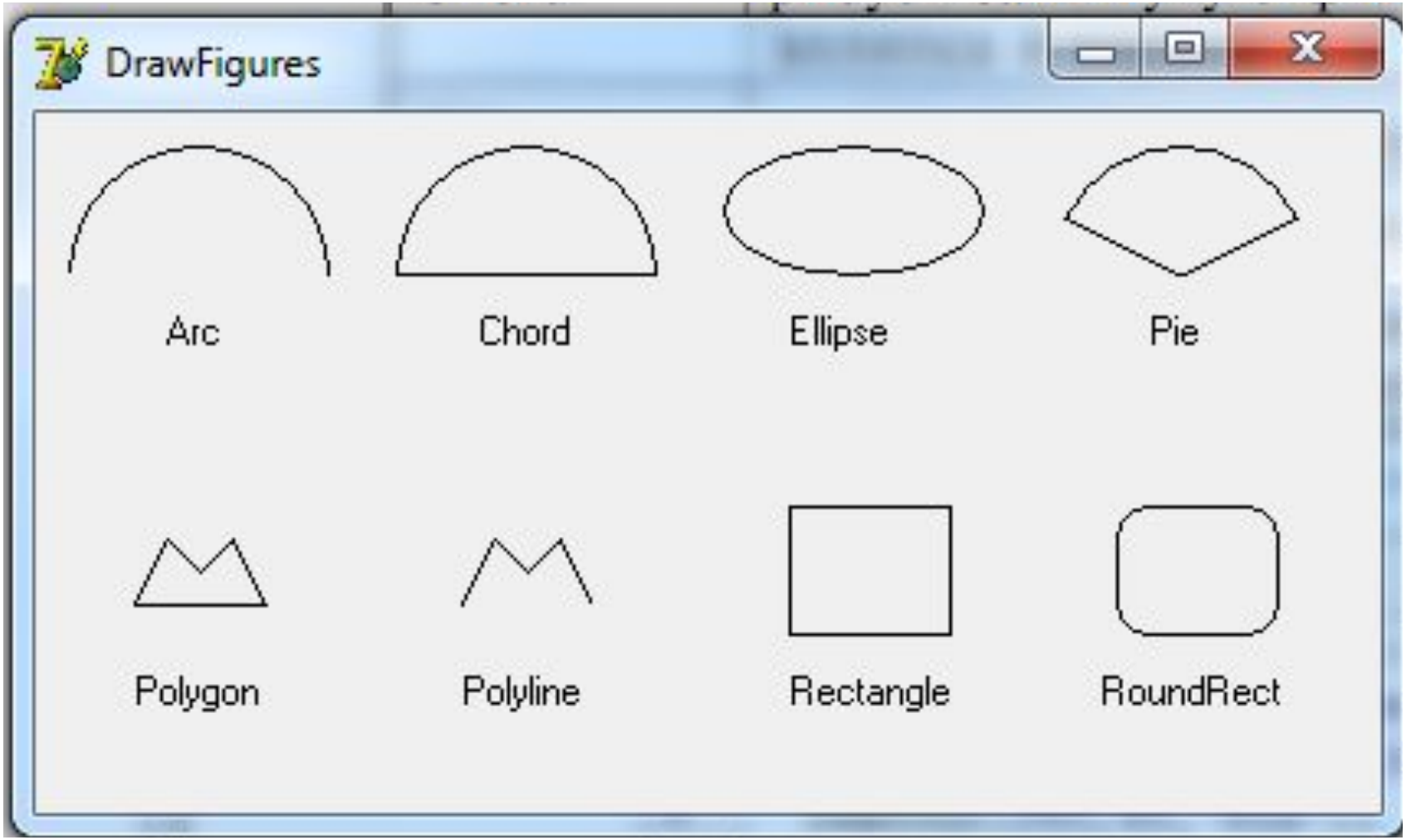
Color – цвет заполнения;

Style – стиль заполнения;



BitMap – нестандартное заполнение заданным шаблоном (битовая матрица 8x8).

Методы канвы для рисования фигур



Вывод текста

Для вывода текста на поверхность графического объекта используется метод TextOut

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.

Шрифт, который используется для вывода текста, определяется значением свойства Font соответствующего объекта canvas.

Область вывода текста закрашивается текущим цветом кисти.

Методы вычерчивания графических примитивов

Линия

Вычерчивание прямой линии осуществляет метод LineTo:

Компонент.Canvas.LineTo(x,y)

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 при помощи метода MoveTo.

Вид линии (цвет, толщина и стиль) определяется значениями свойств объекта Реп 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.

```
Polyline(gr);
```

Где gr: array[1..n] **of** Tpoint.

Метод Polyline можно использовать для вычерчивания замкнутых контуров. Для этого надо, чтобы первый и последний элементы массива содержали координаты одной и той же точки.

Окружность и эллипс

Метод Ellipse вычерчивает эллипс или окружность, в зависимости от значений параметров.

Объект.Canvas.Ellipse(*x1,y1,x2,y2*)

где:

объект — имя объекта (компонента), на поверхности которого выполняется вычерчивание;

x1, y1, x2, y2 — координаты прямоугольника, внутри которого вычерчивается эллипс или, если прямоугольник является квадратом, окружность.

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:

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

где:

x1, y1, x2, y2 — параметры, определяющие эллипс (окружность), частью которого является вычерчиваемая дуга;

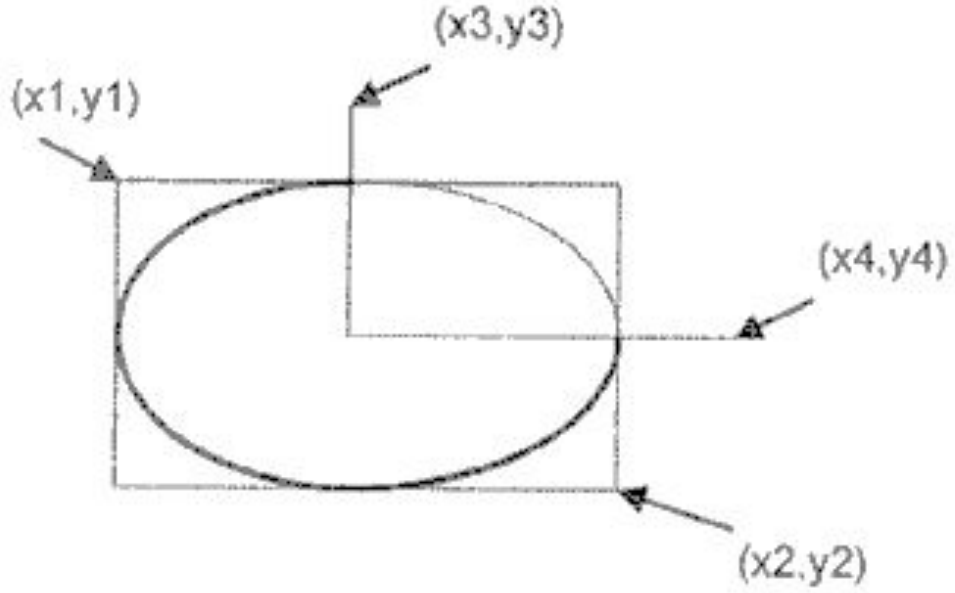
x3, y3 — параметры, определяющие начальную точку дуги;

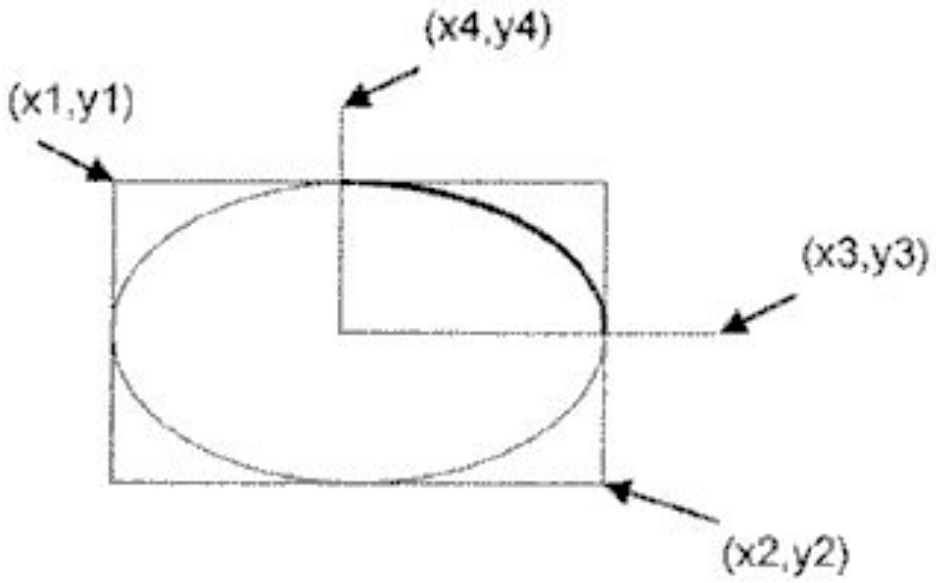
x4, y4 — параметры, определяющие конечную точку дуги.

Цвет, толщина и стиль линии, которой вычерчивается дуга, определяются значениями свойства Реп поверхности (canvas), на которую выполняется вывод.

Начальная (конечная) точка — это точка пересечения границы эллипса и прямой, проведенной из центра эллипса в точку с координатами x3 и y3 (x4, y4).

Дуга вычерчивается против часовой стрелки от начальной точки к конечной.

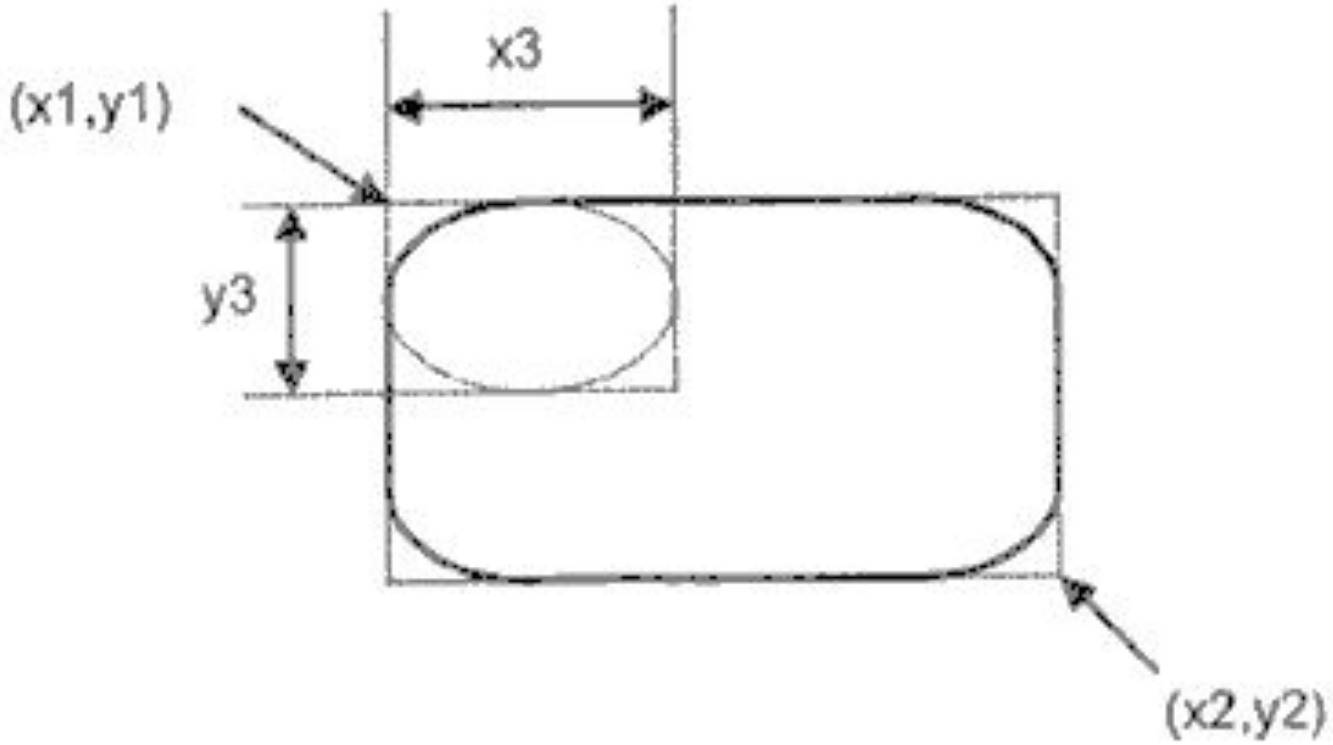




Прямоугольник

Прямоугольник вычерчивается методом Rectangle:
Объект.Canvas.Rectangle(x1, y1,x2, y2) где:
объект — имя объекта (компонента), на поверхности которого выполняется вычерчивание;
x1, y1 и x2, y2 — координаты левого верхнего и правого нижнего углов прямоугольника.

Метод RoundRec тоже вычерчивает прямоугольник, но со скругленными углами:
Объект.Canvas.RoundRec(x1,y1,x2, y2, x3, y3) где:
x1, y1, x2, y2 -- параметры, определяющие положение углов прямоугольника, в который вписывается прямоугольник со скругленными углами;



x3 и y3 — размер эллипса, одна четверть которого используется для вычерчивания скругленного угла.

Есть еще два метода, которые вычерчивают прямоугольник, используя в качестве инструмента только кисть (Brush).

Метод FillRect вычерчивает закрашенный прямоугольник, а метод FrameRect — только контур.

У каждого из этих методов лишь один параметр — структура типа TRect.

Поля структуры TRect содержат координаты прямоугольной области, они могут быть заполнены при помощи функции Rect, параметрами которой служат координаты левого верхнего и правого нижнего углов прямоугольника.

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.

Цвет и стиль границы многоугольника определяются значениями свойства Pen, а цвет и стиль заливки области, ограниченной линией границы, — значениями свойства Brush.

Сектор

Метод Pie вычерчивает сектор эллипса или круга:

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

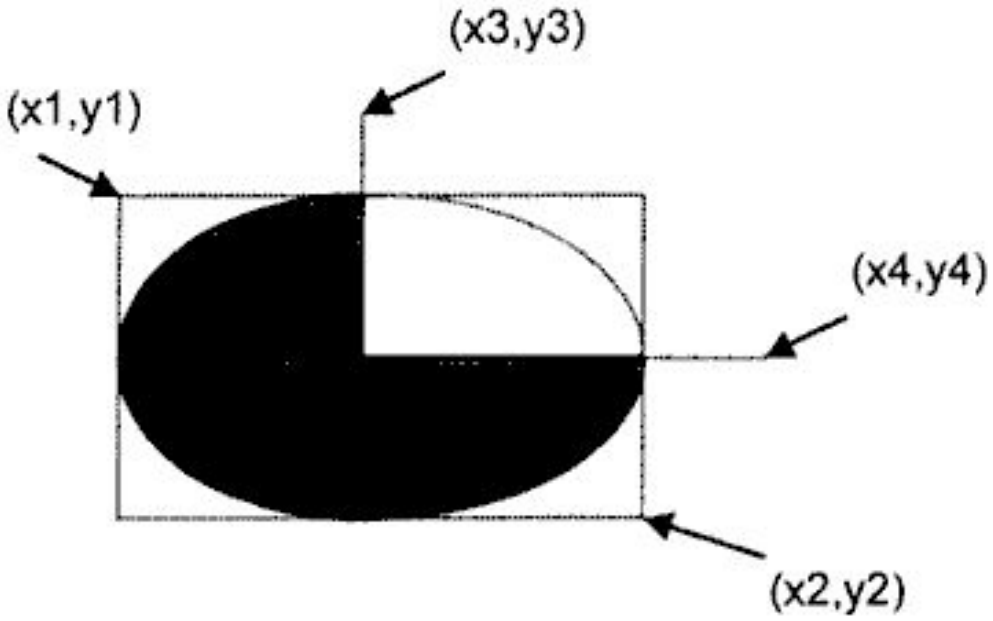
где:

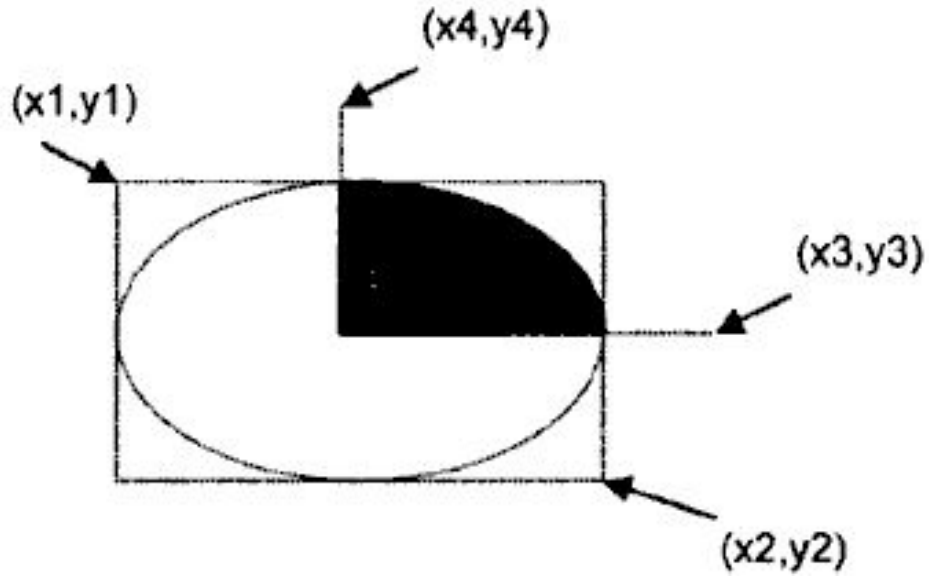
x1, y1, x2, y2 — параметры, определяющие эллипс (окружность), частью которого является сектор;

x3, y3, x4, y4 — параметры, определяющие координаты конечных точек прямых, являющихся границами сектора.

Начальные точки прямых совпадают с центром эллипса (окружности).

Сектор вырезается против часовой стрелки от прямой, заданной точкой с координатами (x3, y3), к прямой, заданной точкой с координатами (x4, y4).





Точка

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.

Размерность массива pixels определяется размером графической поверхности.

Размер графической поверхности формы (рабочей области, которую также называют *клиентской)* задается значениями свойств ciientwidth и ClientHeight, а размер графической поверхности компонента image — значениями свойств width и Height.

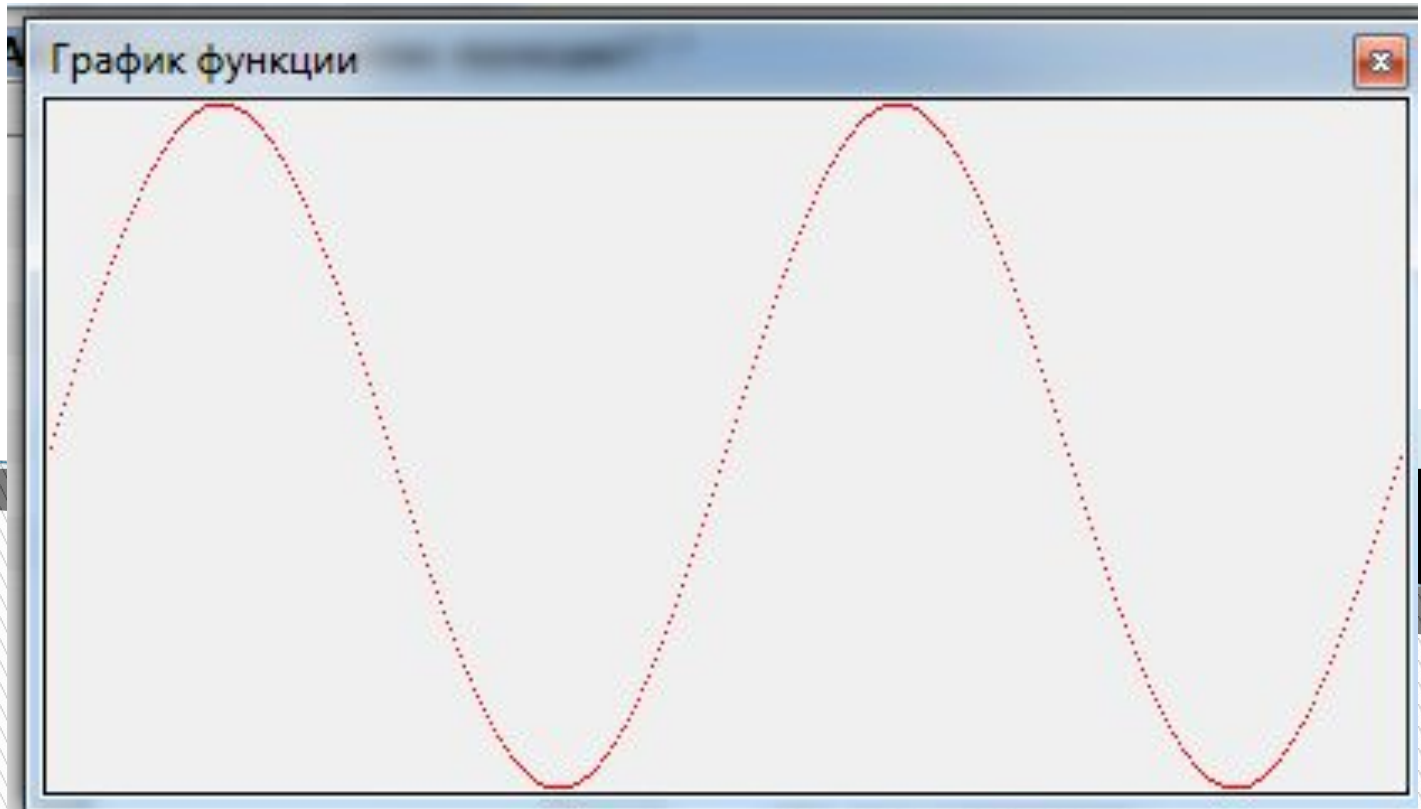
Свойство Pixels можно использовать для построения графиков.

График строится, как правило, на основе вычислений по формуле. Границы диапазона изменения аргумента функции являются исходными данными.

Диапазон изменения значения функции может быть вычислен. На основании этих данных можно вычислить масштаб, позволяющий построить график таким образом, чтобы он занимал всю область формы, предназначенную для вывода графика.

Программа выводит на форму график функции y=sin(x) 0≤x≤4π.

.



```
function F(x:real):real;
begin
  F:=sin(x);
end;
procedure FuncGraph;
var x,y:real;     px,py:integer;
begin
   for px:=0 to Form1.ClientWidth do
   begin
     x:=px*4*PI/Form1.ClientWidth;
     y:=F(x);
 py:=trunc(Form1.ClientHeight/2-y*Form1.ClientHeight/2);
     Form1.Canvas.Pixels[px,py]:=clRed;
   end;
end;
```

```
procedure TForm1.FormResize(Sender: TObject);
Begin
//Очищаем форму
Form1.Canvas.FillRect(Rect(0,0,ClientWidth,ClientHeight));
FuncGraph;
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
  FuncGraph;
end;
```

Вывод иллюстраций

Наиболее просто вывести иллюстрацию, которая находится в файле с расширением bmp, jpg или ico, можно при помощи компонента image, значок которого находится на вкладке **Additional.**

Во время разработки формы иллюстрация задается установкой значения свойства picture путем выбора файла иллюстрации в стандартном диалоговом окне, которое появляется в результате щелчка на командной кнопке **Load** окна **Picture Editor .**

Чтобы вывести иллюстрацию в поле компонента image во время работы программы, нужно применить метод LoadFromFile к свойству Picture, указав в качестве параметра имя файла иллюстрации.

Битовые образы

При работе с графикой удобно использовать объекты типа TBitMap (битовый образ).

Битовый образ представляет собой находящуюся в памяти компьютера невидимую графическую поверхность, на которой программа может сформировать изображение.

Содержимое битового образа (картинка) легко и быстро может быть выведено на поверхность формы или области вывода иллюстрации (image). Поэтому в программах битовые образы обычно используются для хранения небольших изображений.

Загрузить в битовый образ нужную картинку можно при помощи метода LoadFromFlie, указав в качестве параметра имя BMP-файла, в котором находится нужная иллюстрация.

Вывести содержимое битового образа (картинку) на поверхность формы или области вывода иллюстрации можно путем применения метода Draw к соответствующему свойству поверхности (canvas).

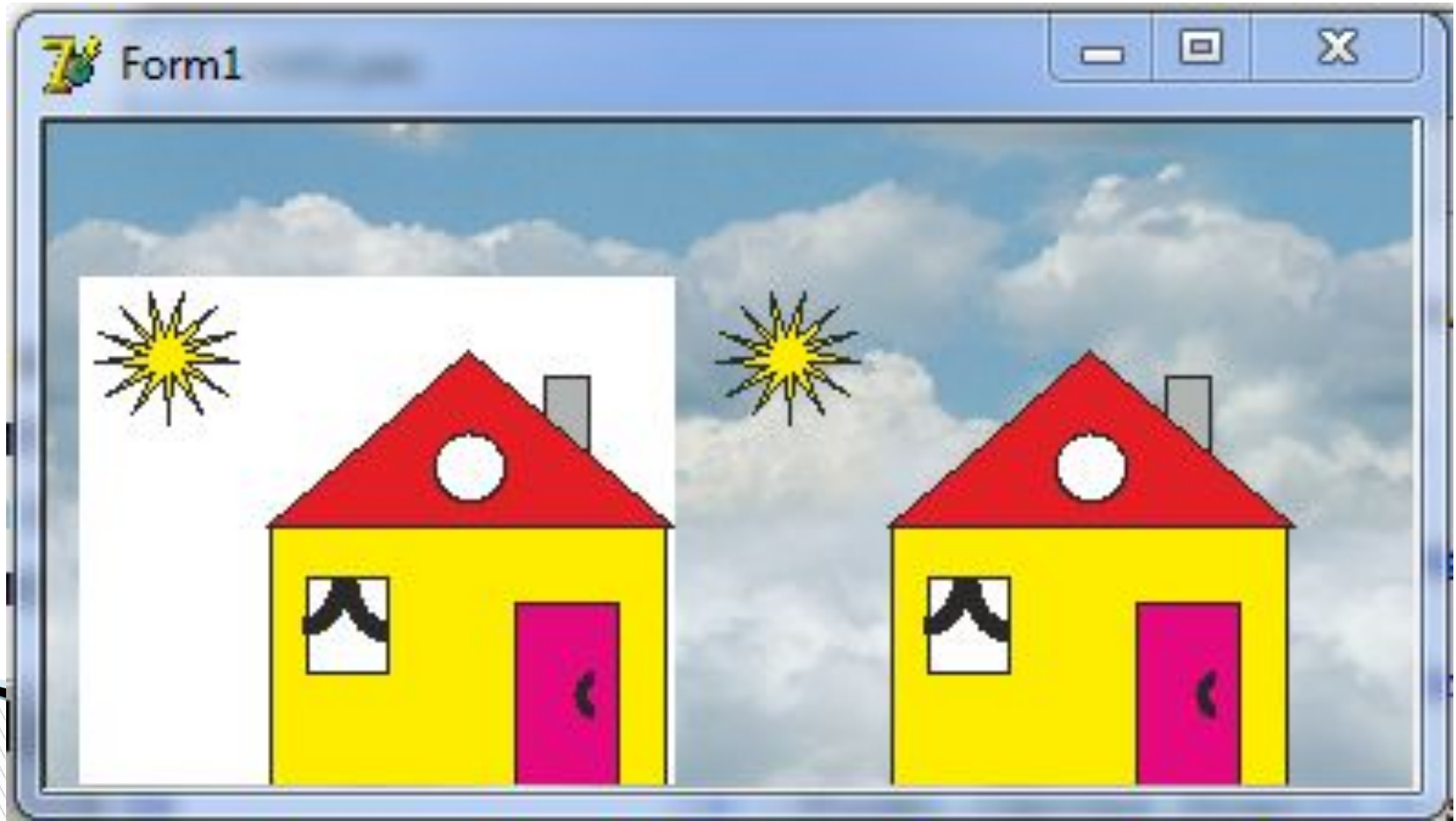
Например

Image1.Canvas.Draw(x,y, bm)

выводит картинку битового образа bm на поверхность компонента image1 (параметры x и y определяют положение левого верхнего угла картинки на поверхности компонента).

Если перед применением метода Draw свойству Transparent объекта TBitMap присвоить значение True, то фрагменты рисунка, окрашенные цветом, совпадающим с цветом левого нижнего угла картинки, не будут выведены — через них будет как бы проглядывать фон.

Если в качестве "прозрачного" нужно использовать цвет, отличный от цвета левой нижней точки рисунка, то свойству Transparentcoior следует присвоить значение символьной константы, обозначающей необходимый цвет.



```
var
sky,home: TBitMap;

procedure TForm1.FormPaint(Sender: TObject);
begin
sky := TBitMap.Create;
home := TBitMap.Create;
sky.LoadFromFile('2.bmp');
home.LoadFromFile('1.bmp') ;
Form1.Canvas.Draw(0,0,sky);
Form1.Canvas.Draw(10,43,home);
home.Transparent:=True;
Form1.Canvas.Draw(180,43,home);
sky.free; home.free;
end;
```

Мультипликация

Под мультипликацией обычно понимается движущийся и меняющийся рисунок. В простейшем случае рисунок может только двигаться или только меняться.

Рисунок может быть сформирован из графических примитивов.

Чтобы обеспечить перемещение рисунка надо сначала вывести рисунок на экран, затем через некоторое время стереть его и снова вывести этот же рисунок, но уже на некотором расстоянии от его первоначального положения.

Подбором времени между выводом и удалением рисунка, а также расстояния между старым и новым положением рисунка, можно добиться того, что у наблюдателя будет складываться впечатление, что рисунок равномерно движется по экрану.

Программа, демонстрирует движение окружности от левой к правой границе

```
var
  x,y: byte;     // координаты центра окружности
  dx: byte;      // приращение координаты x при движении окружности

procedure Ris;
begin
     // стереть окружность
     Form1.Canvas.Pen.Color:=form1.Color;
     Form1.Canvas.Ellipse(x,y,x+10,y+10);
     x:=x+dx;
     // нарисовать окружность на новом месте
     Form1.Canvas.Pen.Color:=clBlack;
     Form1.Canvas.Ellipse(x,y,x+10,y+10);
end;
```

```
// сигнал от таймера
procedure TForm1.Timer1Timer(Sender: TObject);
begin
     Ris;
end;

// начало работы программы
procedure TForm1.FormActivate(Sender: TObject);
begin
  //установка начальных значений
  x:=0;
  y:=25;
  dx:=3;
  // период возникновения события OnTimer - 0.05 сек
  Timer1.Interval:=50;
  Form1.Canvas.Brush.Color := Form1.Color;
end;
```

Метод базовой точки

При программировании сложных изображений, состоящих из множества элементов, используется метод, который называется *методом базовой точки.* Суть этого метода заключается в следующем:

1. Выбирается некоторая точка изображения, которая принимается за базовую.
2. Координаты остальных точек отсчитываются от базовой точки.
3. Если координаты точек изображения отсчитывать от базовой в относительных единицах, а не в пикселах, то обеспечивается возможность масштабирования изображения.

Использование битовых образов

В предыдущем примере изображение формировалось из графических примитивов. Теперь рассмотрим, как можно реализовать перемещение одного сложного изображения на фоне другого, например перемещение домика на фоне неба.

Эффект перемещения картинки может быть создан путем периодической перерисовки картинки с некоторым смещением относительно ее прежнего положения. При этом предполагается, что перед выводом картинки в новой точке сначала удаляется предыдущее изображение. Удаление картинки может быть выполнено путем перерисовки всей фоновой картинки или только той ее части, которая перекрыта битовым образом движущегося объекта.

Будем использовать второй подход.

Картинка выводится применением метода Draw к свойству canvas компонента Image, а стирается путем копирования нужной части фона из буфера на поверхность компонента Image.

Сохранение копии фона выполняется при помощи метода CopyRect, который позволяет выполнить копирование прямоугольного фрагмента одного битового образа в другой. Объект, к которому применяется метод CopyRect, является приемником копии битового образа. В качестве параметров методу передаются координаты и размер области, куда должно быть выполнено копирование, поверхность, откуда должно быть выполнено копирование, а также положение и размер копируемой области.

CopyRect(Dest:TRect;Canvas:TCanvas;Source:TRect)

Информация о положении и размере копируемой в буфер области фона, на которую будет наложено изображение домика и которая впоследствии должна быть восстановлена из буфера, находится в структуре BackRct типа TRect. Для заполнения этой структуры используется функция Bounds.

Bounds(x,y,Width,Height)

где:

x и y — координаты левого верхнего угла области;
width и Height — ширина и высота области.

```
var
Back, bitmap, Buf : TBitMap;
BackRct : TRect;
 BufRct: Trect;
 x1,y1:integer;
 W,H: integer;
procedure TForm1.FormActivate(Sender: TObject);
begin
// создать три объекта – битовых образа
Back := TBitmap.Create; // фон
bitmap := TBitmap.Create; // картинка
Buf := TBitmap.Create; // буфер
// загрузить и вывести фон
Back.LoadFromFile('2.bmp');
Form1.Image1.canvas.Draw(0,0,Back);
```

```
// загрузить картинку, которая будет двигаться
bitmap.LoadFromFile('1.bmp');
// определим прозрачный цвет
bitmap.Transparent := True;
bitmap.TransParentColor := bitmap.canvas.pixels[0,0];
// создать буфер для сохранения области фона, на которую будет накладываться картинка
W:= bitmap.Width;
H:= bitmap.Height;
Buf.Width:= W;
Buf.Height:=H;
Buf.Palette:=Back.Palette;
// Обеспечиваем соответствие палитр
Buf.Canvas.CopyMode:=cmSrcCopy;
//определим область буфера, которая будет использоваться для восстановления фона
```

```
BufRct:=Bounds(0,0,W,H);
// начальное положение картинки
x1 := -W; y1 := 20;
// определим сохраняемую область фона
BackRct:=Bounds(x1,y1,W,H); // и сохраним ее
Buf.Canvas.CopyRect(BufRct,Back.Canvas,BackRct);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
// восстановлением фона (из буфера) удалим рисунок
Form1.image1.canvas.Draw(x1,y1,Buf);
x1:=x1+2;
if x1>form1.Image1.Width then x1:=-W;
// определим сохраняемую область фона
BackRct:=Bounds(x1,y1,W,H);
// сохраним ее копию
```

```
Buf.Canvas.CopyRect(BufRct,Back.Canvas,BackRct);
// выведем рисунок
Form1.image1.canvas.Draw(x1,y1,bitmap);
end;

procedure TForm1.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
// освободим память, выделенную для хранения битовых
образов
Back.Free;
bitmap.Free;
Buf.Free;
end;
```

Загрузка битового образа из ресурса программы

В приведенной программе битовые образы фона и картинки загружаются из файлов. Это не всегда удобно. Delphi позволяет поместить необходимые битовые образы в виде *ресурса* в файл исполняемой программы и по мере необходимости загружать битовые образы из ресурса, т. е. из файла исполняемой программы (EXE-файла).

Преимущества загрузки картинок из ресурса программы очевидны: при распространении программы не надо заботиться о том, чтобы во время работы программы были доступны файлы иллюстраций, все необходимые программе картинки находятся в исполняемом файле.

Создание файла ресурсов

Для того чтобы воспользоваться возможностью загрузки картинки из ресурса, необходимо сначала создать *файл ресурсов,* поместив в него нужные картинки.

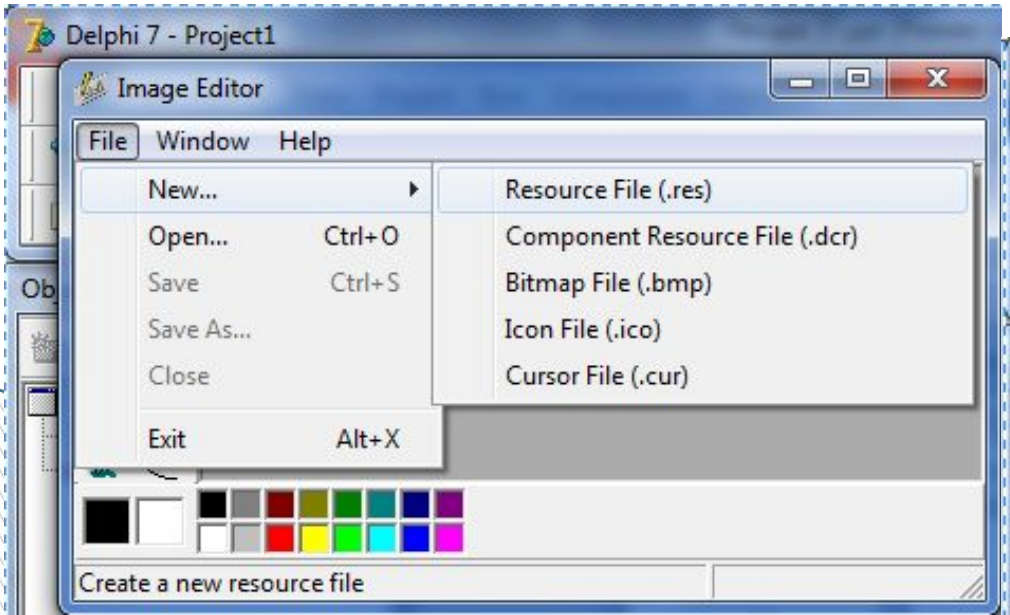
Файл ресурсов можно создать при помощи утилиты **Image Editor** (Редактор изображений), которая запускается выбором команды **Image Editor** меню **Tools.**

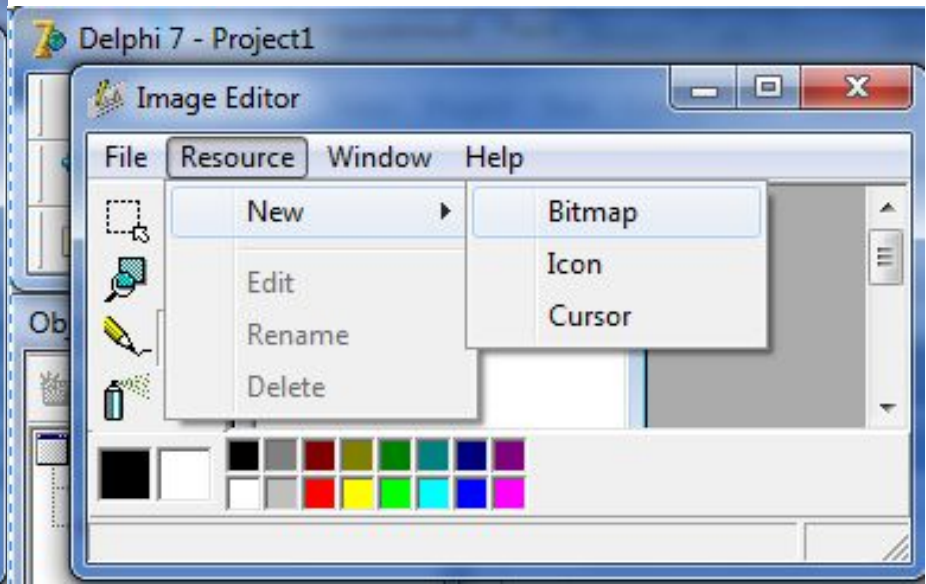
Для того чтобы создать новый файл ресурсов, надо из меню **File** выбрать команду **New**, а затем в появившемся подменю — команду **Resource File** (Файл ресурсов)

В результате открывается окно нового файла ресурсов, а в строке меню окна **Image Editor** появляется новый пункт — **Resource**.

Для того чтобы в этот файл добавить новый ресурс, необходимо выбрать команду **New** меню **Resource** и из открывшегося списка — тип ресурса. В данном случае следует выбрать **Bitmap**. После выбора **Bitmap** открывается диалоговое окно **Bitmap Properties**, используя которое можно установить размер (в пикселах) и количество цветов создаваемой картинки.

Нажатие кнопки ОК в диалоговом окне **Bitmap Properties** вызывает появление элемента **Bitmap1** в иерархическом списке **Contents.** Этот элемент соответствует новому ресурсу, добавленному в файл .





Имя **Bitmap1** может быть изменено выбором команды **Rename** меню **Resource** и вводом нужного имени.

Для создания битового образа необходимо выбрать команду **Edit** меню **Resource,** в результате чего открывается окно графического редактора.

Графический редактор **Image Editor** предоставляет стандартный набор инструментов, используя которые можно нарисовать нужную картинку.

Если нужная картинка уже существует в виде отдельного файла, то ее можно через буфер обмена поместить в битовый образ файла ресурсов следующим образом:

1. Сначала надо запустить графический редактор, например **Paint**, загрузить в него файл картинки и выделить всю картинку или ее часть. В процессе выделения следует обратить внимание на информацию о размере (в пикселах) выделенной области (**Paint** выводит размер выделяемой области в строке состояния). Затем, выбрав команду **Копировать** меню **Правка,** следует поместить копию выделенного фрагмента в буфер.

2. Далее нужно переключиться в **Image Editor** и установить значения характеристик ресурса в соответствии с характеристиками картинки, находящейся в буфере. Значения характеристик ресурса вводятся в поля диалогового окна **Bitmap Properties,** которое открывается выбором команды **Image Properties** меню **Bitmap.** После установки характеристик ресурса можно вставить картинку в ресурс, выбрав команду **Past** меню **Edit.**

3. После добавления всех нужных ресурсов файл ресурса следует сохранить в том каталоге, где находится программа, для которой этот файл создается. Сохраняется файл ресурса обычным образом, т. е. выбором команды **Save** меню **File. Image Editor** присваивает файлу ресурсов расширение res.

Подключение файла ресурсов

Для того чтобы ресурсы были доступны программе, необходимо в текст программы включить директиву, которая сообщит компилятору, что в файл исполняемой программы следует добавить содержимое файла ресурсов.

В общем виде эта директива выглядит следующим образом:

{$R *ФайлРесурсов*}

где *ФайлРесурсов* — имя файла ресурсов. Например, {$R images.res}

Директиву включения файла ресурсов в файл исполняемой программы обычно помещают в начале текста модуля.

Загрузить картинку из ресурса в переменную типа TBitMap можно при помощи метода LoadFromResourceName, который имеет два параметра: идентификатор программы и имя ресурса. В качестве идентификатора программы используется глобальная переменная Hinstance. Имя ресурса должно быть представлено в виде строковой константы.

Например,:

```
Pic.LoadFromResourceName(Hinstance,‘sky') ;
```

Если в приведенном выше примере использовать файл ресурсов, то необходимо только изменить

```
Back.LoadFromFile(‘sky.bmp');
bitmap.LoadFromFile(‘home.bmp');
```

на

```
Back.LoadFromResourceName(Hinstance,'sky');
bitmap.LoadFromresourceName(Hinstance,'HOME');
```

Просмотр "мультика"

Теперь рассмотрим, как можно реализовать вывод в диалоговом окне программы простого "мультика", подобного тому, который можно видеть в диалоговом окне **Установка связи** при подключении к Internet . Эффект бегущего между телефоном и компьютером красного квадратика достигается за счет того, что в диалоговое окно выводятся сменяющие друг друга картинки.

Кадры мультика обычно находятся в одном файле или в одном ресурсе. Перед началом работы программы они загружаются в буфер, в качестве которого удобно использовать объект типа TBitMap. Задача процедуры, реализующей вывод мультика, состоит в том, чтобы выделить очередной кадр и вывести его в нужное место. формы.

Вывести кадр на поверхность формы можно применением метода copyRect к свойству canvas этой формы. Метод CopyRect копирует прямоугольную область одной графической поверхности на другую.

Canvas1.CopyRect (Область1, Canvas2, 0бласть2)

где:

Canvas1 — графическая поверхность, на которую выполняется копирование;

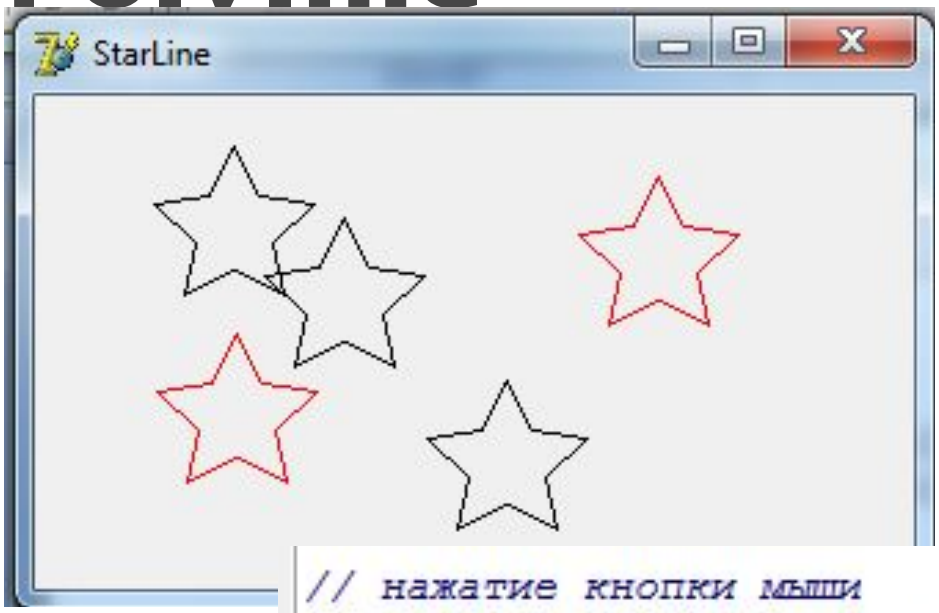
Canvas2 — графическая поверхность, с которой выполняется копирование;

Область2 —- задает положение и размер копируемой прямоугольной области, а параметр Область1— положение копии на поверхности Canvas1.

В качестве параметров область1 и область2 используются структуры типа TRect, поля которых определяют положение и размер области.

Программа «Звезды»

Рисование с помощью Polyline




```
// нажатие кнопки мыши
procedure TForm1.FormMouseDown(Sender: TObject;
Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft  // нажата левая кнопка?
     then Form1.Canvas.Pen.Color := clBlack
     else Form1.Canvas.Pen.Color := clRed;
  //рисовать звезду в точке (x,y)
  StarLine(x, y);
end;
```

```
procedure StarLine(x0,y0,r: integer; Canvas: TCanvas);
var
   p : array[1..11] of TPoint; a: integer ; i: integer;
begin
   a := 18;
   for i:=1 to 10 do
      begin
        if (i mod 2 = 0) then
           begin
             p[i].x := x0+Round(r/2*cos(a*2*pi/360));
             p[i].y:=y0-Round(r/2*sin(a*2*pi/360));
           end
        else
```

```
    begin
            p[i].x:=x0+Round(r*cos(a*2*pi/360));
            p[i].y:=y0-Round(r*sin(a*2*pi/360));
          end;
        a := a+36;
     end;
  p[11].X := p[1].X;
  p[11].Y := p[1].Y;
  Canvas.Polyline(p);
end;
```

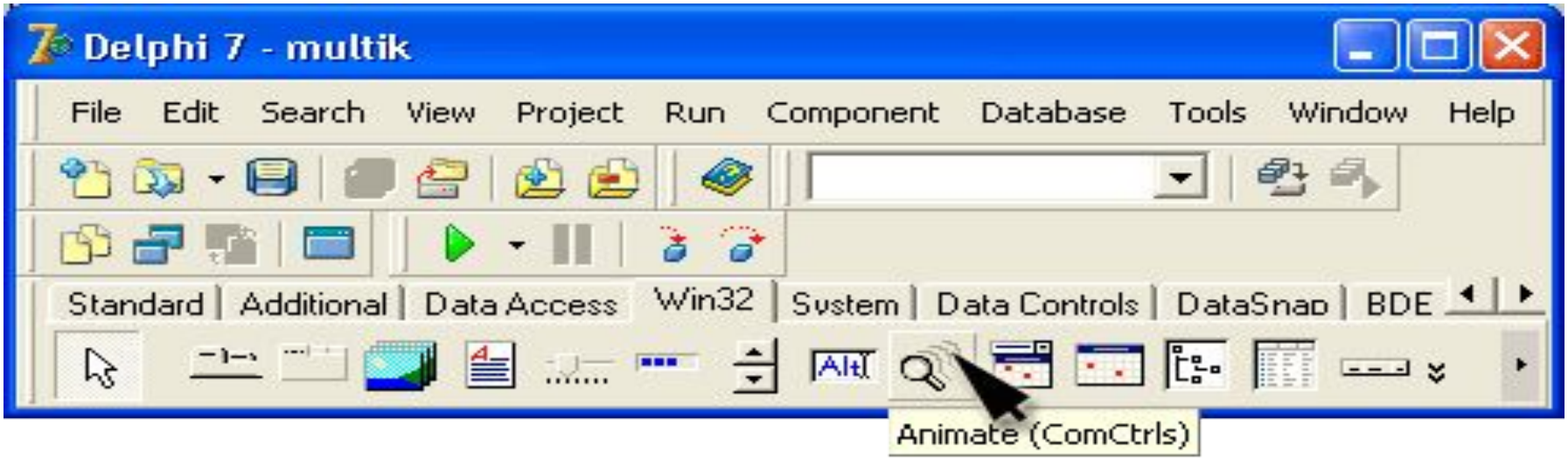
Мультимедийные возможности Delphi

Delphi предоставляет в распоряжение программиста два компонента, которые позволяют разрабатывать мультимедийные программы:

- Animate — обеспечивает вывод простой анимации (подобной той, которую видит пользователь во время копирования файлов);
- MediaPlayer — позволяет решать более сложные задачи, например, воспроизводить видеоролики, звук, сопровождаемую звуком анимацию.

Компонент Animate

Компонент Animate, значок которого находится на вкладке **Win32**, позволяет воспроизводить простую анимацию, кадры которой находятся в AVI-файле.



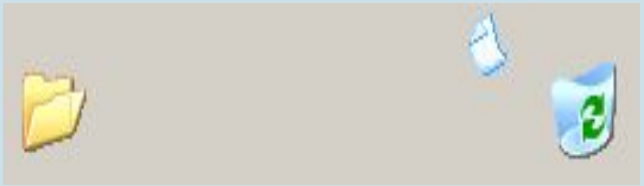
компонент Animate предназначен для воспроизведения AVI-файлов, которые содержат только анимацию. При попытке записать в свойство FileName имя файла, который содержит звук, Delphi выводит сообщение о невозможности открытия указанного файла (**Cannot open** AVI).

Свойства компонента Animate

Свойство	Определяет
Name	Имя компонента. Используется для доступа к свойствам компонента и управлением его поведением
FileName	Имя AVI-файла в котором находится анимация, отображаемая при помощи компонента
StartFrame	Номер кадра, с которого начинается отображение анимации
stopFrame	Номер кадра, на котором заканчивается отображение анимации
Activate	Признак активизации процесса отображения кадров анимации
Color	Цвет фона компонента (цвет "экрана"), на котором воспроизводится анимация
Transparent	Режим использования "прозрачного" цвета при отображении анимации
Repetitions	Количество повторов отображения анимации

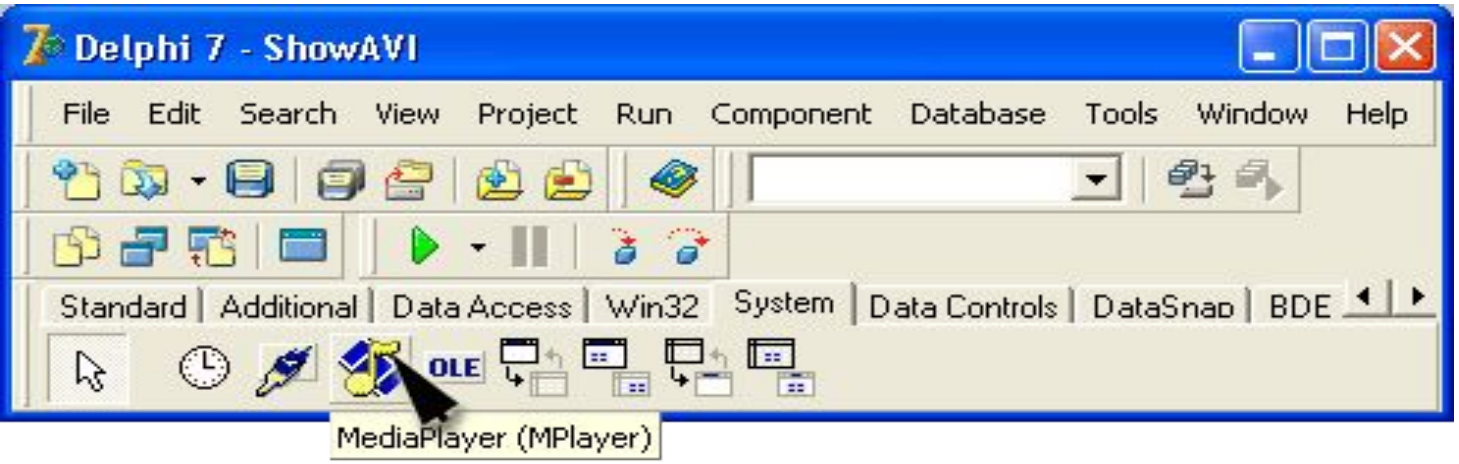
Компонент Animate позволяет использовать стандартные анимации Windows.

Вид анимации определяется значением свойства CommonAVI. Значение свойства задается при помощи именованной константы.

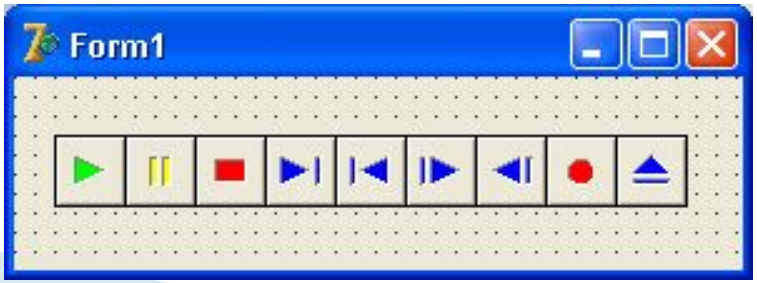
Значение	Анимация	Процесс
aviCopyFiles		Копирование файлов
AviDeleteFile		Удаление файла
aviRecycleFile		Удаление файла в корзину

Компонент MediaPlayer

Компонент MediaPlayer, значок которого находится на вкладке **System**, позволяет воспроизводить видеоролики, звук и сопровождаемую звуком анимацию.



В результате добавления к форме компонента MediaPlayer на форме появляется группа кнопок, подобных тем, которые можно видеть на обычном аудио- или видеоплеере.



Кнопки компонента MediaPlayer

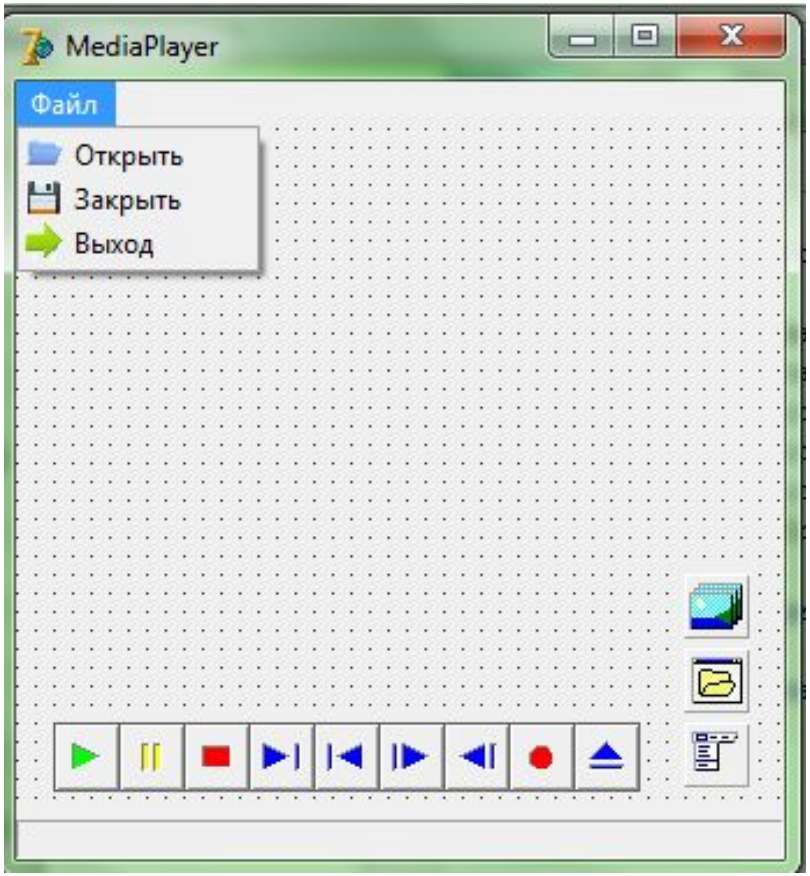
Кнопка	Обозначение	Действие
Воспроизведение	btPlay	Воспроизведение звука или видео
Пауза	btPause	Приостановка воспроизведения
Стоп	btStop	Остановка воспроизведения
Следующий	btNext	Переход к следующему кадру
Предыдущий	btPrev	Переход к предыдущему кадру
Шаг	btStep	Переход к следующему звуковому фрагменту, например, к следующей песне на CD
Назад	btBack	Переход к предыдущему звуковому фрагменту, например, к предыдущей песне на CD
Запись	btRecord	Запись
Открыть/Закрыть	btEject	Открытие или закрытие CD-дисковода компьютера

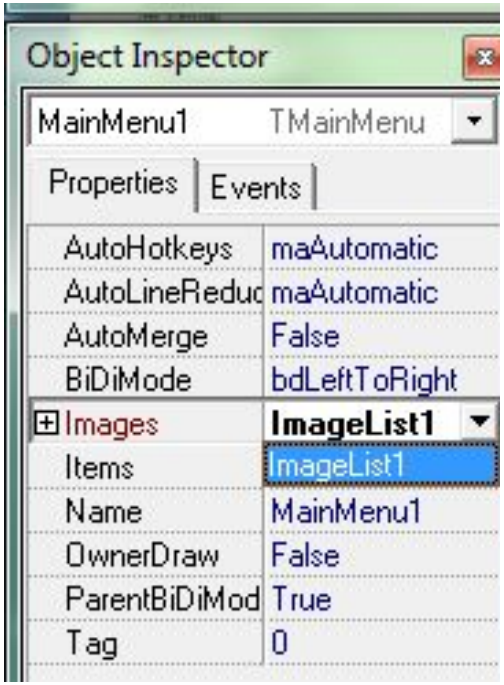
Свойства компонента MediaPiayer

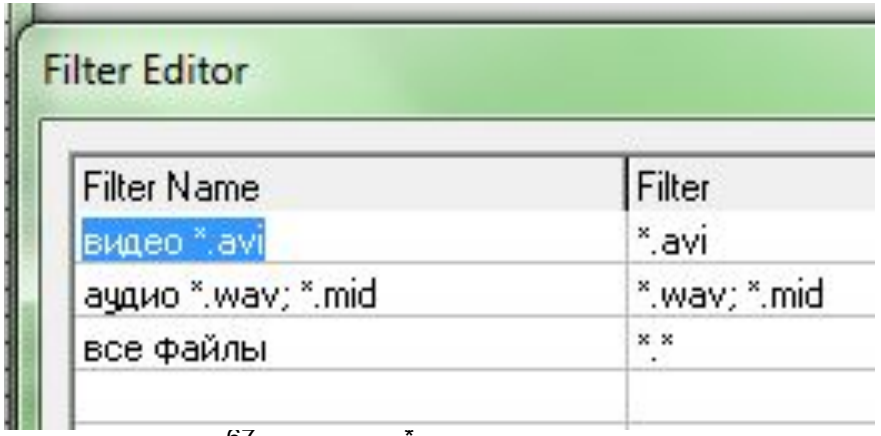
Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента и управлением работой плеера
DeviceType	Тип устройства. Определяет конкретное устройство, которое представляет собой компонент MediaPiayer. Тип устройства задается именованной константой: dtAutoSelect — тип устройства определяется автоматически; dtVaweAudio — проигрыватель звука; dtAVivideo — видеопроигрыватель; dtCDAudio — CD-проигрыватель
FileName	Имя файла, в котором находится воспроизводимый звуковой фрагмент или видеоролик
AutoOpen	Признак автоматического открытия сразу после запуска программы, файла видеоролика или звукового фрагмента
Display	Определяет компонент, на поверхности которого воспроизводится видеоролик (обычно Panel)
VisibleButt ons	Составное свойство. Определяет видимые кнопки компонента. Позволяет сделать невидимыми некоторые кнопки

Программа «Мультимедиа-плеер»

Форма и настройки







Пункты меню Открыть и Выход

```
//меню выход
procedure TForm1.N3Click(Sender: TObject);
begin
  Close;
end;

//меню открыть
procedure TForm1.N2Click(Sender: TObject);
var s:string;
begin
  if OpenDialog1.Execute
  then
  begin
    s:=OpenDialog1.FileName;
    StatusBar1.Panels[0].Text:='Файл: '+s;
    MediaPlayer1.FileName:=OpenDialog1.FileName;
    Form1.Canvas.FillRect(Rect(0,0,Form1.ClientWidth,Form1.ClientHeight));
    MediaPlayer1.Open;
  end;
end;
```

Пункты меню Закрыть и обработка создания формы

```
//меню закрыть
procedure TForm1.N4Click(Sender: TObject);
begin
 MediaPlayer1.Stop;
 MediaPlayer1.Close;
 StatusBar1.Panels[0].Text:='';
 Form1.Canvas.FillRect(Rect(0,0,Form1.ClientWidth,Form1.ClientHeight));
end;
//создание формы - отображение видео на форме
//а не в отдельном окне
procedure TForm1.FormCreate(Sender: TObject);
begin
  MediaPlayer1.Display:=Form1;
end;
```