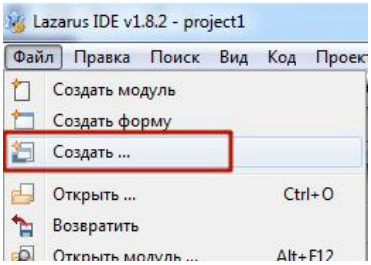
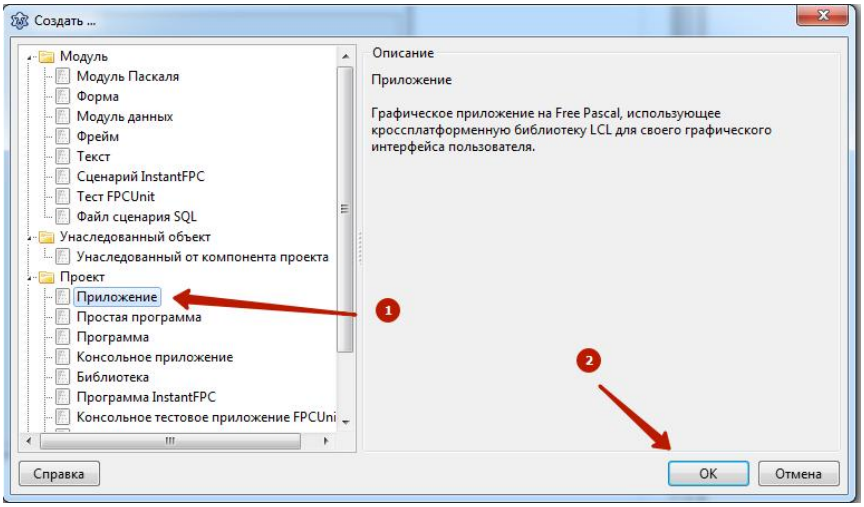
Графические построения и анимация

Для демонстрации возможностей системы Lazarus в отношении графических построений, создадим новый проект (**Файл** – **Создать**).



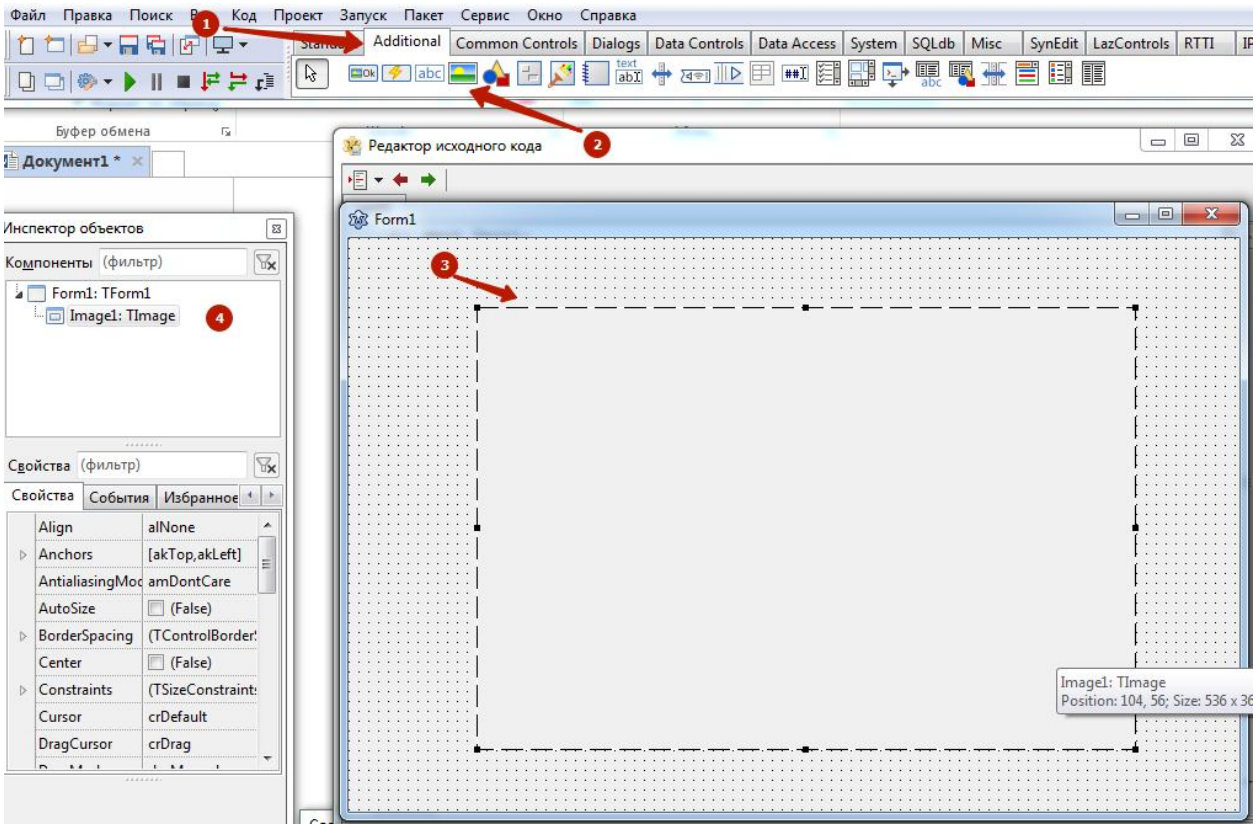
В открывшемся окне «**Создать**»

1. Выбираем пункт «**Проект** - **Приложение**»
2. Нажимаем кнопку «**ОК**»

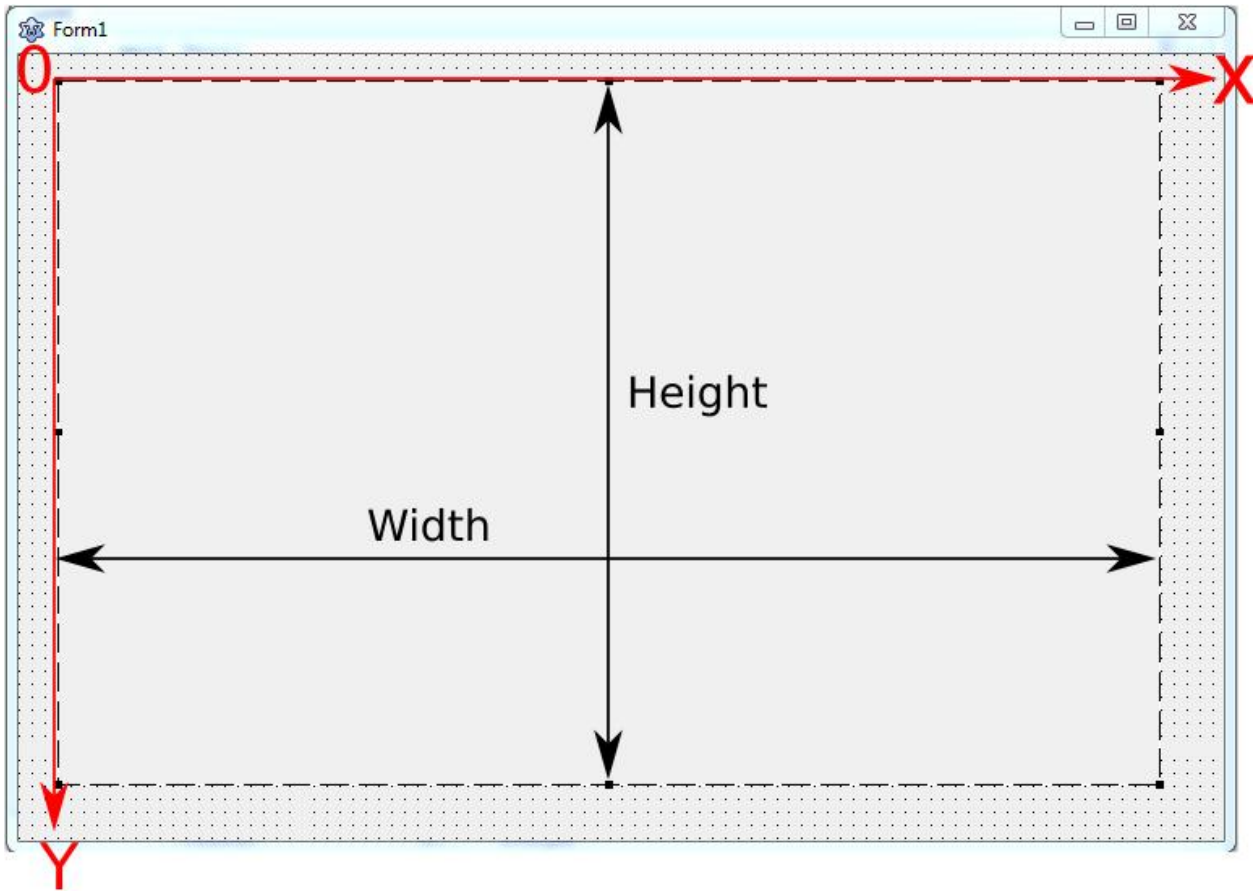


Перед тем, как приступить к созданию графических примитивов, необходимо на форму проекта поместить компонент Image. Для этого

1. Нажимаем на панели инструментов вкладку Additional.
2. Из набора компонентов вкладки выбираем **T**Image.
3. Помещаем его на форму (размер компонента можно произвольно изменять мышкой или изменяя соответствующие параметры компонента Width/Height).
4. В инспекторе объектов появится помещенный компонент на форму (выбрав в списке компонентов необходимый, можем менять его свойства).

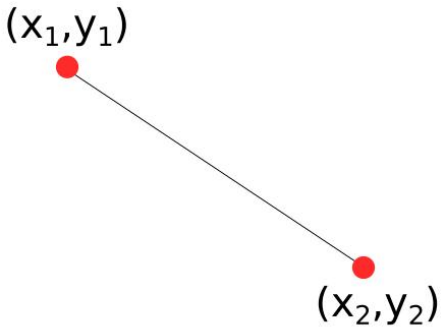


Следует отметить, что система координат компонента Image, берет свое начало в левом верхнем углу. Доступ к горизонтальному размеру объекта можно получить через свойства Name.Width, к вертикальному через Name.Height, где Name – имя компонента. (размеры объектов указываются в пикселях)



Отрезок

Ниже представлена схема построения отрезка проходящего через точки (x_1,y_1) (x_2,y_2).

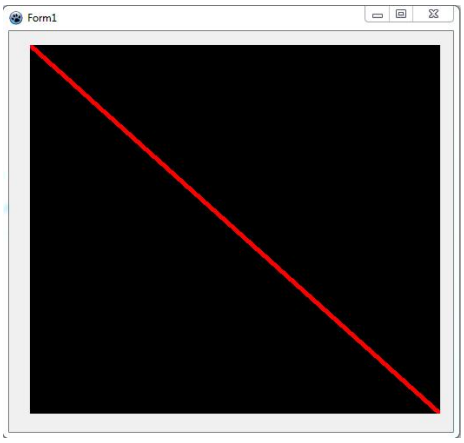


Name.Canvas.MoveTo(x_1,y_1)
Name.Canvas.LineTo(x_2,y_2)

Напишем программу, которая строит по диагонали компонента Image отрезок красного цвета. Для этого щелкнем два раза мышкой по форме проекта – автоматически в редакторе кода создастся процедура FormCreate. Зададим цвет будущего отрезка, толщину и соответствующие координаты.

```
procedure TForm1.FormCreate(Sender: TObject);
begin

  image1.Canvas.Pen.Color:=clRed; // Цвет отрезка
  image1.Canvas.Pen.Width:=5;     // Толщина отрезка
  image1.Canvas.MoveTo(0,0);      // Начало отрезка
  image1.Canvas.LineTo(Image1.Width,Image1.Height); // Конец отрезка
end;
```

После запуска программы увидим отрезок. По умолчанию, Компонент Image заливается черным цветом, если бы мы не поменяли цвет отрезка, то после запуска программы увидели бы черный прямоугольнику, поскольку по умолчанию цвет линии – черный.



Как видно из примера выше, для того чтобы нарисовать отрезок нужного цвета, проходящий через заданные точки, потребуется написать минимум 4 строчки кода. А если в процессе написания программы, потребуется нарисовать 20 линий, то для этого соответственно нужно будит набрать 80 строчек кода. Это очень неудобно. Для уменьшения объема кода программы и для улучшения читабельности кода предлагается

написать процедуру, которая будит рисовать отрезка по заданным параметрам. Входными параметрами будут:

Im – Имя компонента Image, в котором будим рисовать;

x1,y1,x2,y2 – координаты точек через которые проходит отрезок;

width – толщина отрезка;

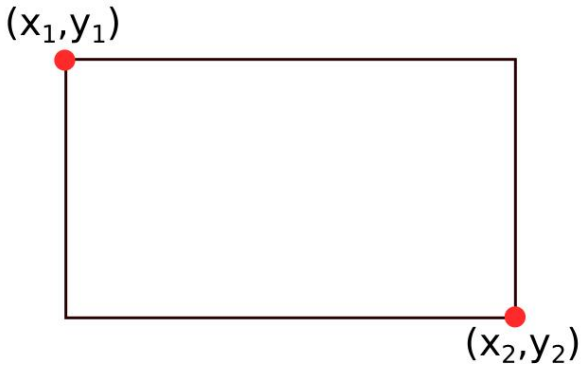
col – цвет отрезка.

```
procedure Lin(im:TImage; x1,y1,x2,y2,width:integer; col:TColor);
begin
   im.Canvas.Pen.Color:=col; // Цвет отрезка
   im.Canvas.Pen.Width:=width;      // Толщина отрезка
   im.Canvas.MoveTo(x1,y1);       // Начало отрезка
   im.Canvas.LineTo(x2,y2); // Конец отрезка

end;

procedure TForm1.FormCreate(Sender: TObject);
begin
   Lin(image1,0,0,Image1.Width,Image1.Height,5,clRed); // Отрезок
end;
```

Как видно из примера выше, после написания процедуры, создание отрезка занимает одну строчку.

Прямоугольник

Ниже представлена схема построения прямоугольника диагональ которого проходит через точки (x_1,y_1) (x_2,y_2).

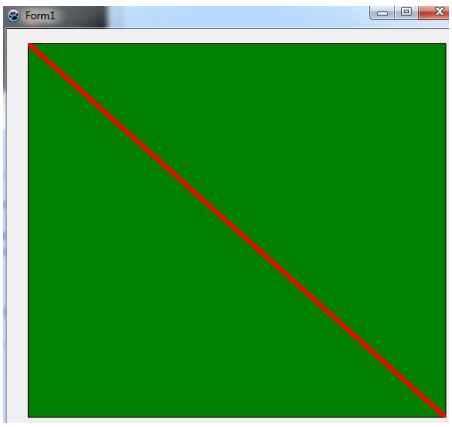


Name.Canvas.Rectangle(x_1,y_1,x_2,y_2)

Попробуем изменить цвет фона Image со стандартного черного на зеленый. Для этого, до того, как создадим отрезок - создадим прямоугольник зеленого цвета, размер которого равен размеру компонента Image. Перед тем, как создать прямоугольник, необходимо поменять цвет кисти (цвет кисти определяет, цвет заливки замкнутого контура).

```
procedure TForm1.FormCreate(Sender: TObject);
begin
   Image1.Canvas.Brush.Color:=clGreen; // Цвет закрашиваемого контура
   Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height); // Прямоугольник размером с компонент Image1
   Lin(image1,0,0,Image1.Width,Image1.Height,5,clRed); // Отрезок
end;
```

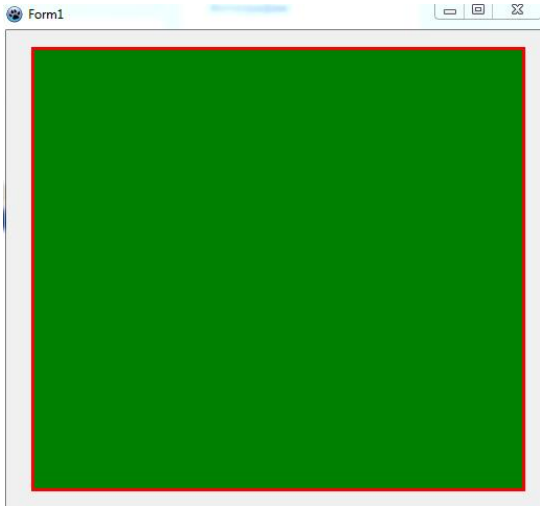
После запуска приложения получим:



Следует отметить, что последовательность создания графических объектов строго определена логикой создаваемого приложения. Если к примеру, поменять последовательность действий, сперва нарисовать отрезок, а потом посредством прямоугольника поменять цвет фона:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Lin(image1,0,0,Image1.Width,Image1.Height,5,clRed); // Отрезок
  Image1.Canvas.Brush.Color:=clGreen; // Цвет закрашиваемого контура
  Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height); // Прямоугольник размером с компонент Image1
end;
```

То в результате получим:

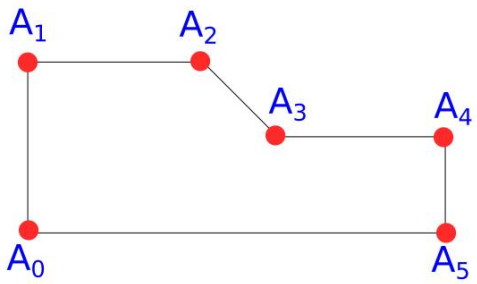


Как видно, прямоугольник закрыл собой отрезок.

Многоугольник

Ниже представлена схема построения многоугольника, проходящего через массив точек.

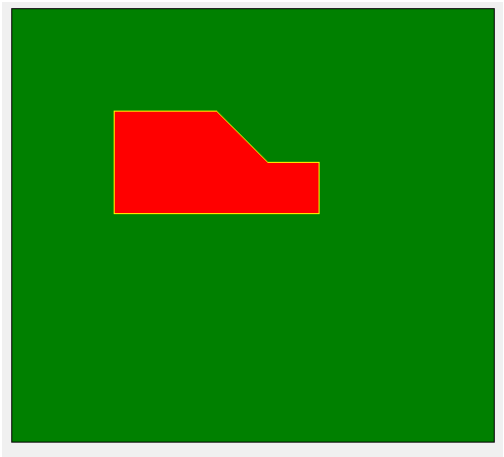
var A:array of TPoint



A[0].X:=x_0 A[0].Y:=y_0
A[1].X:=x_1 A[1].Y:=y_1
A[2].X:=x_2 A[2].Y:=y_2
A[3].X:=x_3 A[3].Y:=y_3
A[4].X:=x_4 A[4].Y:=y_4
A[5].X:=x_5 A[5].Y:=y_5

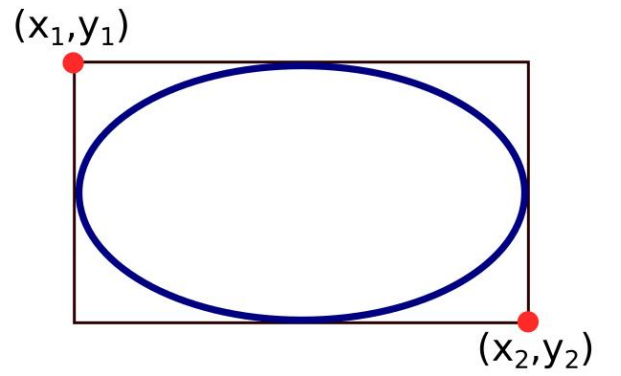
Name.Canvas.Polygon(A)

```
procedure TForm1.FormCreate(Sender: TObject);
 var A:array of TPoint; // динамический массив точек
begin

  Image1.Canvas.Brush.Color:=clGreen; // Цвет закрашиваемого контура
  Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height); // Прямоугольник размером с компонент Image1

  SetLength(A,6); // Указываем сколько точек в массиве

  {Задаем координаты точек}
  A[0].x:=100; A[0].y:=100;
  A[1].x:=200; A[1].y:=100;
  A[2].x:=250; A[2].y:=150;
  A[3].x:=300; A[3].y:=150;
  A[4].x:=300; A[4].y:=200;
  A[5].x:=100; A[5].y:=200;

  Image1.Canvas.Pen.Color:=clYellow; // Цвет контура многоугольника
  Image1.Canvas.Brush.Color:=clRed;  // Цвет заливки многоугольника
  Image1.Canvas.Polygon(A);          // Построение многоугольника
end;
```



Эллипс

Ниже представлена схема построения эллипса.

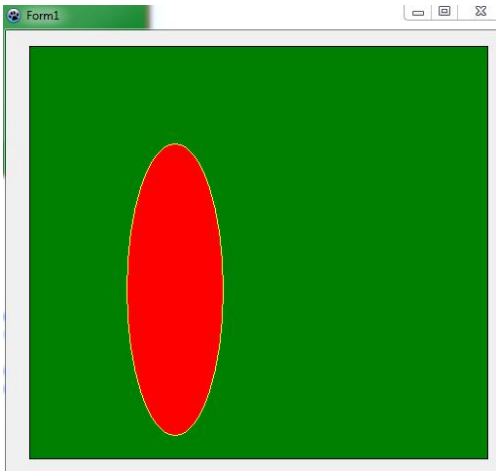


Name.Canvas.Ellipse(x_1,y_1,x_2,y_2)

Как видно из схемы, эллипс является вписанным в прямоугольник проходящий через точки (x_1,y_1) (x_2,y_2).

```
procedure TForm1.FormCreate(Sender: TObject);
begin

   Image1.Canvas.Brush.Color:=clGreen; // Цвет закрашиваемого контура
   Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height); // Прямоугольник размером с компонент Image1

   Image1.Canvas.Pen.Color:=clYellow; // Цвет контура эллипса
   Image1.Canvas.Brush.Color:=clRed;  // Цвет заливки эллипса
   Image1.Canvas.Ellipse(100,100,200,400); // Эллипс
end;
```

x1 y1 x2 y2



Если необходимо, чтобы эллипс отображался в виде окружности, то, диагональ проходящая через точки (x_1,y_1) (x_2,y_2) должна составлять с горизонтом 45^0(прямоугольник, в который вписан эллипс – является квадратом). Ниже представлена процедура создания окружности с заданными параметрами:

im – имя компонента Image на котором происходит отрисовка окружности;

x,y – координаты центра окружности;

rad – радиус окружности(в пикселях);

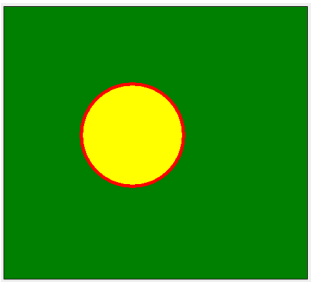
widht – толщина контура окружности;

col_p – цвет контура окружности;

col_b – цвет заливки контура.

```
procedure Cir(im:TImage; x,y,rad,width:integer; col_p,col_b:TColor);
begin
  im.Canvas.Pen.Color:=col_p;    // Цвет линии контура
  im.Canvas.Pen.Width:=width;    // Толщина контура
  im.Canvas.Brush.Color:=col_b; // Цвет заливки
  im.Canvas.Ellipse(x-rad,y-rad,x+rad,y+rad); // Эллипс
end;


procedure TForm1.FormCreate(Sender: TObject);
begin

  Image1.Canvas.Brush.Color:=clGreen; // Цвет закрашиваемого контура
  Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height); // Прямоугольник размером с компонент Image1

  Cir(Image1,200,200,80,5,clRed,clYellow);
  { Круг, радиуса 80 пикселей, центр которого находится в точке (200,200)}

end;
```

После запуска приложения увидим окружность, центр которой находится в точке (200,200).

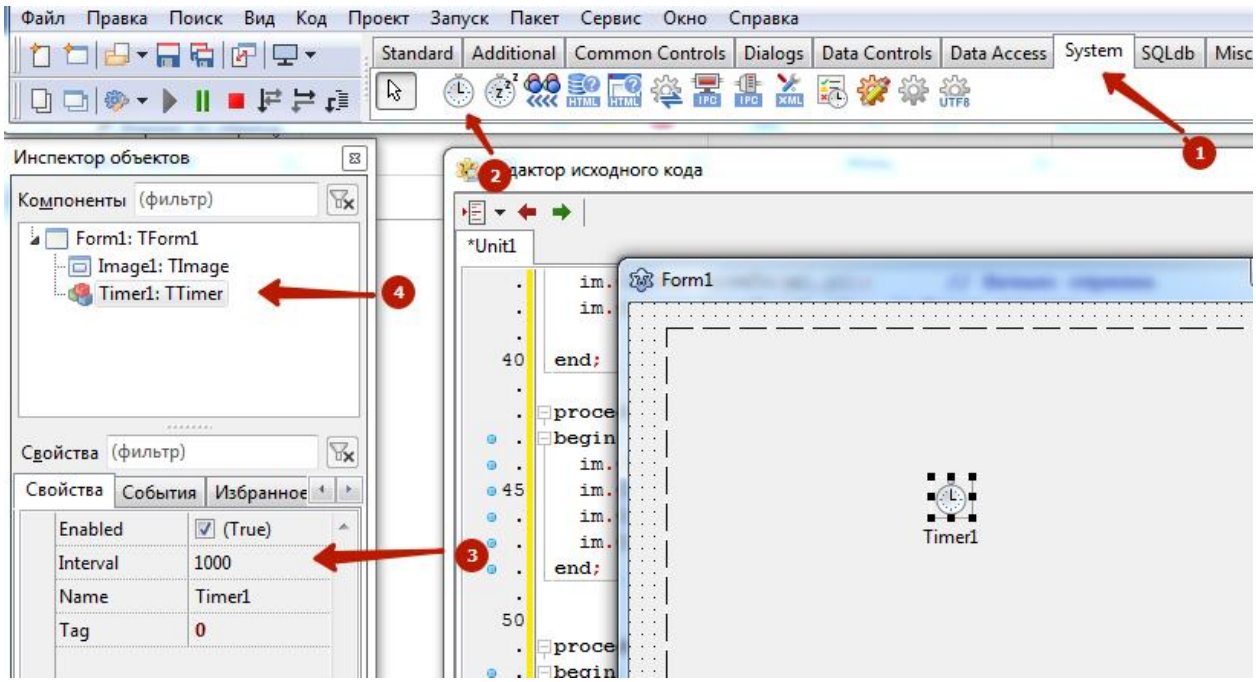


Анимация

Продемонстрируем возможности создания анимации на примере прямолинейного движения окружности.

Перед тем, как приступить к созданию анимации, необходимо поместить компонент TTimer на форму, для этого:

1. На панели инструментов выбираем вкладку System
2. Из набора инструментов выбираем TTimer и помещаем его на форму
3. Свойство компонента, отвечающее за временной интервал между кадрами (Interval) указано в миллисекундах (1000 миллисекунд равняется 1 секунде).
4. В списке компонентов помещенных на форму должен появиться Timer1



Поскольку, по условию задания, окружность должна двигаться, то значить координаты центра окружности должны быть заданы в виде глобальных переменных.

```
var
  Form1: TForm1;
  xc,yc :integer; // глобальные переменные, координаты центра окружности

implementation

{$R *.lfm}

{ TForm1 }
```

После объявления глобальных переменных, необходимо задать их начальное значение

```
procedure TForm1.FormCreate(Sender: TObject);
begin

  Image1.Canvas.Brush.Color:=clGreen; // Цвет закрашиваемого контура
  Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height); // Прямоугольник размером с компонент Image1

  xc:=200; yc:=200;
  Cir(Image1,xc,yc,80,5,clRed,clYellow);
  { Круг, радиуса 80 пикселей, центр которого находится в точке (xc,yc)}

end;
```

Щелкаем дважды по компоненту TTimer помещенного на форму. После чего автоматически создаться процедура Timer1Timer. Эта процедура будит отрабатывать с периодичностью Interval, пока активен таймер. Попробуем в этой процедуре менять координаты центра окружности.

Так же следует отметить, что в процессе анимации, необходимо закрашивать предыдущее состояние кадра. В нашем случае никаких дополнительных объектов не присутствует, поэтому ограничимся только созданием прямоугольника цвет заливки, которого равен цвету исходного фона компонента.

После запуска приложения мы увидим, что окружность стала двигаться с интервалом в 1 секунду, при этом через какое-то время координаты центра окружности выходят за рамки компонента Image и окружность пропадает из поля видимости.

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Image1.Canvas.Brush.Color:=clGreen; // Цвет закрашиваемого контура
  Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height); // Прямоугольник размером с компонент Image1

  xc:=xc+20; // Новое значение xc равно старому значению + 20 пикселей
  yc:=yc+20; // Новое значение yc равно старому значению + 20 пикселей
  Cir(Image1,xc,yc,80,5,clRed,clYellow);
end;
```

Чтобы окружность частично оставалась в поле видимости, напишем ограничение.

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Image1.Canvas.Brush.Color:=clGreen; // Цвет закрашиваемого контура
  Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height); // Прямоугольник размером с компонент Image1

  if(xc+20>Image1.Width)then xc:=Image1.Width else xc:=xc+20;
  {Если новое положение центра окружности по X будит больше соответсвующего
   размера компонента Image1, то новое значение X окружности равно размеру компонента, в противном случае
   смещаем координату на заданное количество пикселей
  }
  if(yc+20>Image1.Height)then yc:=Image1.Height else yc:=yc+20;
  {
   Условие для Y аналогично условию по X
  }
  Cir(Image1,xc,yc,80,5,clRed,clYellow);
end;
```

Самостоятельно

1. Написать программу в которой имеется система координат, проходящая через центр компонента Image. Окружность постоянного радиуса движется по кругу относительно этой системы координат. (См. Приложение 1)
2. Написать программу в которой имеется система координат, проходящая через центр компонента Image. Окружность постоянного радиуса движется по спирали относительно этой системы координат. (См. Приложение 2)
3. Написать программу в которой имеется система координат, проходящая через центр компонента Image. Окружность переменного радиуса движется по спирали относительно этой системы координат. (См. Приложение 3)

При построении системы координат и определении траектории движения центра окружности необходимо будит проводить вычисления, результатом которых не всегда будит целое число, координаты объектов, задаваемые в пикселях, могут быть только целым числом (поскольку мы можем сместить объект только на целое число пикселей). Для получения из числа с плавающей точкой целое число, необходимо воспользоваться функцией округления. Round(A), где а вещественное число, в результате получим целое число.

```
X:=Round(1.01);
```

В результате X=1.