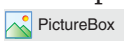
§ 4. Элементы управления для работы с графикой

4.1. Элемент управления для вставки рисунка (PictureBox)

При создании приложений нередко возникает необходимость украсить их графическим изображением. В этом случае можно воспользоваться компонентом **изображение**. На панели компонентов **Стандартные элементы управления** компонент изображение представлен в виде 

, имя объекта PictureBox. Компонент PictureBox, помещенный на форму, получает имя PictureBox N, где N — номер 1, 2, 3... (пример 4.1).

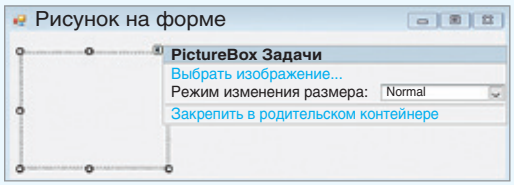
Компонент представляет собой контейнер, в который помещается изображение. Некоторые свойства компонента PictureBox приведены в таблице (пример 4.2).

Используя свойство Image, можно выбрать и загрузить изображение на этапе проектирования приложения. Изображение может быть выбрано в контейнере при нажатии на кнопку  в правом верхнем углу компонента. В этом случае рисунок сохраняется в файле формы и для работы приложения отдельного файла с рисунком не требуется.

Компонент поддерживает вставку рисунков в форматах JPEG, PNG, BMP, GIF. Если требуется обработка изображения (любые изменения рисунка), то рисунок должен быть сохранен в формате BMP. Для рисунков формата PNG или GIF с прозрачным фоном при загрузке сохраняется прозрачность.

Рисунок можно загрузить как фон формы. Для этого предназначено свойство формы BackgroundImage.

Пример 4.1. Компонент *изображение* на форме:

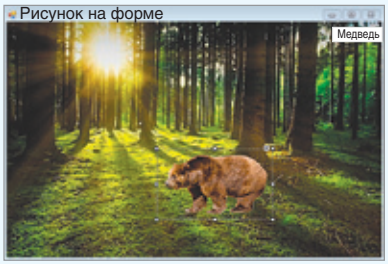


Пример 4.2. Некоторые свойства компонента *изображение*:

Свойство	Назначение
Image	Используется для отображения изображений
SizeMode	Определяет способ отображения рисунка: *Normal* — левый верхний угол рисунка совмещен с левым верхним углом контейнера; *StretchImage* — рисунок вписывается в контейнер; *AutoSize* — размер подгоняется под размер рисунка; *CenterImage* — рисунок будет отцентрирован относительно компонента; *Zoom* — при изменении размеров контейнера будут сохраняться пропорции рисунка
Margin	Определяет промежуток между полями изображения и полями другого компонента (значения All, Left, Top, Right, Bottom)
BackColor	Цвет фона изображения. Может быть прозрачным

Свойство Image компонента PictureBox обладает методом Save, который используется для сохранения изображения. Метод Load компонента PictureBox может быть использован для загрузки изображения при открытии приложения. В этом случае файл с рисунком должен находиться в папке проекта (или нужно прописать полный путь к файлу).

Пример 4.3. Форма на этапе конструирования:

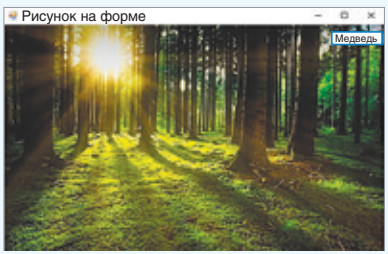


Обработчик события Ckick для Button1:

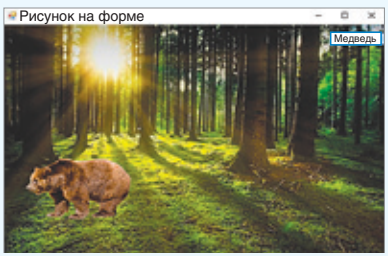
```
procedure Form1.button1_Click
(sender: Object; e: EventArgs);
begin
  PictureBox2.Top := 220;
  var rnd: Random := new Random();
  PictureBox2.Left := rnd.
Next(300);
  PictureBox2.Visible := True;
end;
```

Работающее приложение

До нажатия на кнопку:



После нажатия на кнопку:



Поскольку горизонтальное положение медведя задается случайным образом, то при каждом нажатии на кнопку медведь будет прорисован в новом месте.

Пример 4.3. Создать проект, разместить в нем фоновое изображение на форме. При нажатии на кнопку поверх фонового изображения должно появиться другое изображение.

Этапы выполнения задания

1. Установить размеры формы Height = 450, Width = 670.

2. Загрузить фоновое изображение для формы. Задать для свойства формы BackgroundImageLayout значение Stretch.

3. Поместить на форму компонент *изображение* и кнопку.

4. Для компонента PictureBox установить значение для свойства Visible = = False (изображение невидимо при запуске приложения). Размеры Height = 120, Width = 200. Свойство SizeMode = = StretchImage.

5. Загрузить изображение в компонент PictureBox. Изображение может быть формата PNG, или GIF с прозрачным фоном, или формата BMP с фоном однородного цвета. Свойство BackColor = Transparent.

6. Написать обработчик события OnClick для компонента Button1.

Если при запуске приложения изображение мерцает, то устранить мерцание можно с помощью включения двойной буферизации:

```
DoubleBuffered := true;
```

Эта команда должна быть прописана в обработчике события Load для формы.

4.2. Построение графиков функций

Пространство имен System.Drawing обеспечивает доступ к функциональным возможностям графического интерфейса Windows. Класс Graphics

предоставляет методы для рисования графических примитивов.

Основное событие для построения изображений — Paint.

Если возникает необходимость рисовать по точкам, то для этого используется класс Bitmap (точечное изображение). Каждая точка имеет координаты *X* и *Y*. Система координат такая же, как и для графического окна PascalABC.Net — точка с координатами (0, 0) расположена в верхнем левом углу, ось *OY* направлена вниз. Каждая точка имеет координаты *X* и *Y*. Координаты измеряются в пикселях. Важнейшее свойство пикселя — его цвет. Для задания цвета в PascalABC.Net можно воспользоваться несколькими способами (пример 4.4). Точка изображается с помощью команды `SetPixel(x1, y1, Color));`

Класс Graphics содержит большое количество свойств и методов, позволяющих строить изображения. Многие из методов Graphics совпадают с процедурами, которые использовались в библиотеке GraphABC среды программирования PascalABC.Net. Описание этих методов приведено в приложении.

Пример 4.5. Создать проект и построить график функции $y = x\sin x$ на промежутке, заданном пользователем.

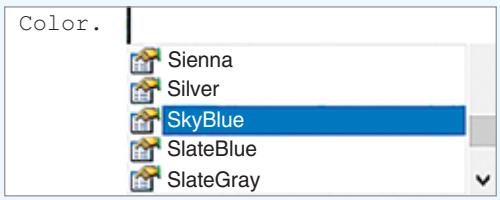
Этапы выполнения задания

1. Поместить на форму компоненты: PictureBox, два компонента Label, два компонента TextBox и компонент Button.

2. Изменить свойства Text у компонентов Label1, Label2 на x0 и xn соответственно.

Пример 4.4. Способы задания цвета в PascalABC.Net:

1. Задание цвета с помощью констант. Константы хранят имя цвета (например, Color.SkyBlue — небесно-синий, Color.Red — красный). Возможные значения появляются в выпадающем списке после ввода в программу ключевого слова Color.

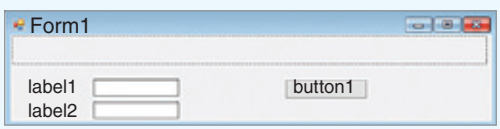


2. Для задания цвета можно воспользоваться функцией `Color.FromArgb(A, R, G, B)`, параметры которой задают прозрачность (альфа-канал) и интенсивность красного, синего и зеленого цветов соответственно. Значения параметров могут изменяться от 0 до 255.

3. Задание цвета с помощью шестнадцатеричных чисел. Пары цифр шестнадцатеричного числа задают прозрачность и интенсивность красного, зеленого и синего цветов соответственно. Полученное шестнадцатеричное число записывается как параметр функции `FromArgb`. Например, `Color.FromArgb($FF0000FF)` — синий цвет. Значения для других цветов: `$FFFF0000` — красный, `$FF00FF00` — зеленый, `$FF000000` — черный, `$FFFFFFFF` — белый.

Шестнадцатеричные числа можно перевести в десятичную систему счисления и пользоваться этими значениями. Например, `Color.FromArgb(4278190335)` — синий цвет.

Пример 4.5. Форма на этапе конструирования:



Пример 4.5. *Продолжение.*

Обработчик события Click для компонента Button1:

```
procedure Form1.button1_Click
 (sender: Object; e: EventArgs);
var x, y, h, k, x0, xn :real;
  x1, y1, n, c_x, c_y : integer;
  gr: Graphics;
  bm: Bitmap;
  p_c: Pen;
begin
//подготовка графической области
//для рисования по пикселям
 bm := new Bitmap (PictureBox1.Width,
       Picturebox1.Height);
 pictureBox1.Image := (Image)(bm);
 gr := Graphics.FromImage
      (pictureBox1.Image);
 //закраска графической области белым
 //цветом
 gr.Clear(Color.White);
 // количество точек
 n := 10000;
 // концы промежутка
 x0 := StrToFloat(TextBox1.Text);
 xn := StrToFloat(TextBox2.Text);
 // центр области построения
 c_x := PictureBox1.Width div 2;
 c_y := PictureBox1.Height div 2;
 // масштабный коэффициент
 k := PictureBox1.Width / (xn - x0);
 // шаг
 h := (xn - x0) / n;
 x := x0;
 //оси
 p_c := new Pen(Color.Black, 1);
 gr.DrawLine(p_c, 0, c_y, 2*c_x, c_y);
 gr.DrawLine(p_c, c_x, 0, c_x, 2*c_y);
 for var i := 1 to n do
 begin
  y := x * sin(x);
  x1 := trunc(x * k) + c_x;
  y1 := trunc(-y * k) + c_y ;
  //проверка попадания точки
  //в графическую область
  if (y1 >= 0) and (y1 < 2*c_y) then
   bm.SetPixel(x1, y1, Color.Blue);
  x := x + h;
 end;
end;
```

3. Изменить свойства Text у компонентов Edit1 и Edit2 на –20 и 20 соответственно.

4. Изменить свойства Text у компонента Button1 на «Построить график».

5. Написать обработчик события Click для компонента Button1 и строить в нем график функции по точкам.

5.1. Нарисовать оси координат в виде двух перпендикулярных линий, пересекающихся в центре компонента PictureBox.

5.2. Чтобы получить видимость сплошной линии, количество точек, которые образуют график функции, должно быть не менее 10 000 (n = 10 000).

5.3. Шаг изменения значения x определяется как $h = \frac{x_n - x_0}{n}$.

5.4. При построении нужно учитывать масштаб: ширина компонента PictureBox должна соответствовать длине заданного промежутка. Тогда масштабный коэффициент можно рассчитать по формуле

$$k = \frac{PictureBox1.Width}{x_n - x_0}.$$

5.5. Поскольку расположение осей координат на экране не совпадает с расположением осей, принятым в математике, то нужно преобразовать координаты: точке (0; 0) должна соответствовать точка в центре компонента PictureBox. Для этого полученное значение x нужно увеличить на величину c_x = PictureBox1.*Width* div 2, а значение y на c_y = PictureBox1.*Height* div 2. Так как ось Y направлена вниз, а не вверх, то у значения Y нужно еще поме-

нять знак на противоположный. На канве будет закрашиваться точка с координатами ($x_{ekr} = x \cdot k + c_x$, $y_{ekr} = -y \cdot k + c_y$).

5.6. Необходимо учитывать, что при вычислении значения x и y будут вещественными, а значения графических координат могут быть только целыми. Поэтому перед прорисовкой точки нужно преобразовать вещественные числа в целые с помощью функции *trunc*.

5.7. Некоторые точки графика при построении могут оказаться за пределами графической области, поэтому необходима проверка значения *y*1: значение должно быть неотрицательным и меньше высоты графической области.

4.3. Построение диаграмм

Основные принципы построения гистограмм и круговых диаграмм разбирались в 10-м классе (примеры 4.6 и 6.8). Используя аналогичные методы Graphics, можно построить диаграммы в оконных приложениях, созданных в PascalABC.Net.

Пример 4.6. Создать проект и построить гистограмму по данным массива из n элементов ($n = 10$). Описать массив с константными данными или добавить данные в массив случайным образом.

Этапы выполнения задания

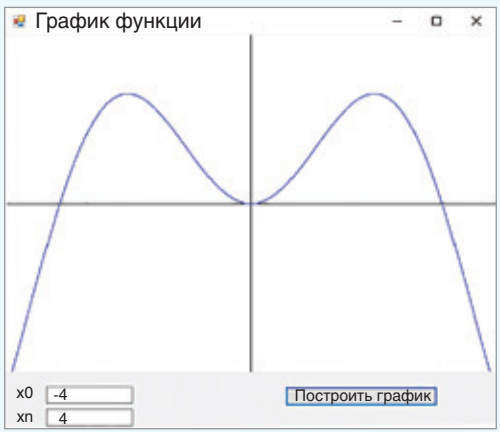
1. Поместить на форму компоненты: PictureBox и два компонента Button.

2. Изменить свойства Text у компонента Button1 на «Диаграмма с константными данными».

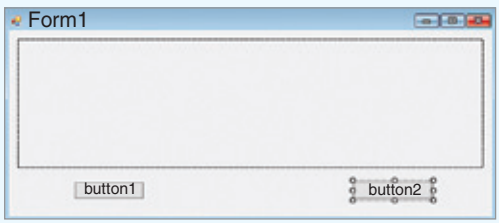
Пример 4.5. *Продолжение.*

Работающее приложение:

Изменение начальных значений концов промежутка:



Пример 4.6. Форма на этапе конструирования:

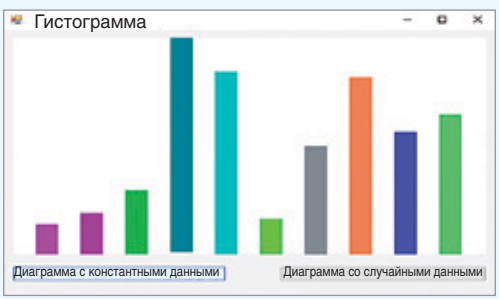


Пример 4.6. *Продолжение.*

Обработчик события Click для компонента Button1:

```
procedure Form1.button1_Click
(sender: Object; e: EventArgs);
 const a: array[1..10] of integer =
(10,14,22,75,63,12,37,61,42,48);
 n = 10;
var max, x, y1, y2, h, i, cr, cg,
  cb : integer;
  m : real;
  gr : Graphics;
  rnd : Random;
  sb : SolidBrush;
begin
 max := a[1];
 for i := 2 to n do
  if a[i] > max then
   max := a[i];
 h:= trunc(PictureBox1.Width/(2*n+1));
 m:= PictureBox1.Height / max;
 x:= h;
//подготовка графической области
//для рисования примитивов
gr := PictureBox1.CreateGraphics;
 gr.Clear(Color.White);
 rnd := new Random();
 for i := 1 to n do
 begin
cr := rnd.next(256);
cg := rnd.next(256);
cb := rnd.next(256);
sb := new SolidBrush(Color.FromArgb
      (cr,cg,cb));
y1 := PictureBox1.Height-1;
y2 := y1-trunc(a[i] * m)-1;
gr.FillRectangle(sb, x, y2, h, y1);
x := x + 2 * h;
 end;
end;
```

Работающее приложение:



3. Изменить свойства Text у компонента Button2 на «Диаграмма со случайными данными».

4. Написать обработчик события Click для компонента Button1, в котором диаграмма строится с помощью прямоугольников.

4.1. Найти максимальный элемент в массиве *max*.

4.2. Рассчитать масштабный коэффициент:

$$m = \frac{PictureBox1.Height}{\max}.$$

4.3. В цикле строить n прямоугольников одинаковой ширины. Ширина прямоугольника

$$h = \frac{PictureBox1.Width}{2n+1}.$$

5. Обработчик для компонента Button2 будет отличаться от обработчика для компонента Button1 только способом получения элементов массива.

5.1. Массив должен быть описан в разделе var: `a: array[1..10] of integer;`

5.2. Элементы массива со значениями от 20 до 100 можно получать следующим образом:

```
rnd := new Random();
for i := 1 to n do
 a[i] := rnd.next(80) + 20;
```

4.4. Анимация

Эффект анимации достигается за счет того, что перед взглядом пользователя происходит быстрая смена изображений. Каждый из кадров анимации остается на экране очень небольшой промежуток времени.

Для замера интервалов времени можно использовать компонент *таймер*.

Он расположен на панели **Компоненты** и представлен в виде Timer, имя объекта Timer. Компонент Timer помещается в отдельную область ниже формы и получает имя TimerN, где N — номер 1, 2, 3... (пример 4.7).

Некоторые свойства компонента Timer приведены в таблице:

Свойство	Назначение
Enabled	Значение true обозначает, что таймер запущен
Interval	Время в миллисекундах, через которое происходит срабатывание таймера и вызов обработчика Tick

Компонент имеет единственный обработчик — Tick, в котором описываются действия, происходящие по истечении интервала срабатывания таймера.

Пример 4.8. Создать проект, в котором самолет будет пролетать над городом.

Этапы выполнения задания

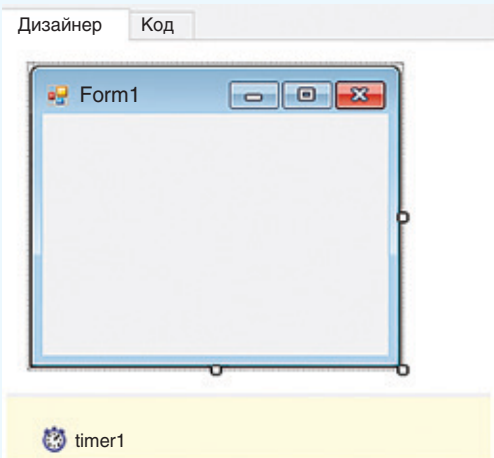
1. Поместить на форму компоненты PictureBox и Button, добавить компонент Timer.

2. Загрузить изображение города в компонент как фон формы (свойство формы BackgroundImage).

3. Установить прозрачный цвет фона для компонента PictureBox1 (значение Transparent у свойства BackColor). Установить режим изменения размера — AutoSize (свойство SizeMode).

4. Написать обработчик события Load для формы и описать начальное

Пример 4.7. Компонент *таймер*:



Компонент Timer не виден при работе приложения, поэтому он размещается в области, специально предназначенной для невизуальных компонентов.

Пример 4.8. Форма на этапе конструирования:

Обработчик события Load для формы:

```
procedure Form1.Form1_Load (sender:
Object; e: EventArgs);
begin
 PictureBox1.Load ('plane.png');
 x := -PictureBox1.Width;
 y := 20;
 PictureBox1.Left := x;
 PictureBox1.Top := y;
end;
```

Пример 4.8. *Продолжение.*

Обработчик события Click для компонента Button1:

```
procedure Form1.button1_Click
(sender: Object; e: EventArgs);
begin
 Timer1.Enabled := True;
end;
```

Обработчик события Tick для компонента Timer1:

```
procedure Form1.timer1_Tick
(sender: Object; e: EventArgs);
begin
 x := x + 1;
 PictureBox1.Left := x;
 if PictureBox1.Left > Width +
    + PictureBox1.Width then
  x := -PictureBox1.Width;
end;
```

Работающее приложение:

Если при запуске анимации возникает мерцание, то включить двойную буферизацию.

положение самолета, указав координаты верхнего левого угла PictureBox1 за пределами формы. Загрузить в PictureBox1 изображение из файла с рисунком самолета.

5. Изменить свойства Text у компонента Button1 на «Полетели!».

6. Установить значение False у свойства таймера Enabled в инспекторе объектов.

7. Установить в инспекторе объектов время срабатывания таймера, равным 10.

8. Написать обработчик события Click для компонента Button1, запустить таймер.

9. В инспекторе объектов установить прозрачность для компонента PictureBox1.

10. Написать обработчик события Tick и менять в нем значение свойства Left у компонента PictureBox1. Если самолет вылетел за границу, то вернуть его в начальное положение.

Двойная буферизация позволяет сделать анимацию более плавной, поскольку все операции рисования сначала выполняются в памяти, а лишь затем на экране компьютера. После завершения всех операций рисования содержимое буфера копируется из памяти непосредственно на связанную с ним область экрана.

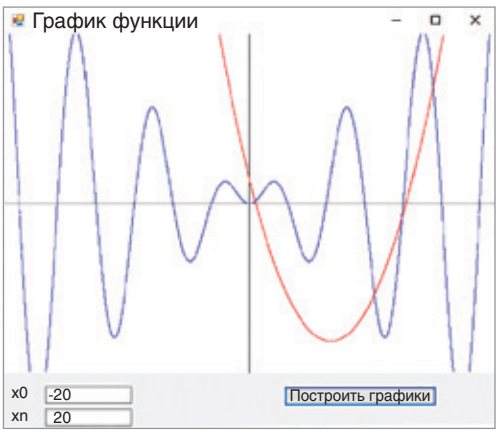
1. Какой компонент используется для размещения изображений?
2. Какое свойство позволяет загрузить готовое изображение в компонент PictureBox?
3. С помощью какого свойства можно установить прозрачный фон для изображения?
4. Какой класс представляет методы для рисования графических примитивов?
5. Какой метод канвы позволяет закрасить пиксель?
6. Какой компонент используется для отсчета времени?

Упражнения

1 Добавьте в проект из примера 4.4. еще одно животное (например, белку). Для размещения белки нужно добавить еще одну кнопку. Местоположение определить случайным образом в верхушке какого-либо дерева.

2 Добавьте в проект 4.5 перечисленные возможности.

1. Оси координат со стрелками и подписями.
2. Единичный отрезок на оси X.
3. График функции $y = 0{,}3x^2 - 4x + 2$ в той же системе координат красным цветом.



Постройте в одной системе координат графики следующих функций. Каждую кривую рисовать своим цветом. Для ввода значений параметров *a*, *b* и *c* добавьте на форму компоненты Text:

1. $y = ax^2$, $y = b\cos x^2$;
2. $y = ax^2 + bx + c$, $y = \frac{1}{x^2 + 3}$;
3. $y = ax^3$, $y = \frac{x}{(x+3)^2}$, $y = bx\ sin\ x$.

3 Измените проект из примера 4.6 так, чтобы строилась линейчатая диаграмма (столбики расположены горизонтально).

4* Измените проект из примера 4.6 так, чтобы значения в массив можно было вводить. Для этого числа необходимо записывать в компонент TextBox, считывать строку из чисел и пробелов, выделять цифры и преобразовывать их в числовые значения.

5 Добавьте в проект из примера 4.8 кнопку «Стоп», при нажатии на которую самолет остановится. *После остановки самолета должен появиться парашют и опуститься вниз.

6* Создайте анимацию движения Луны вокруг Земли. Для расчета координат верхнего левого угла PictureBox1, содержащего Луну, можно воспользоваться параметрическим уравнением окружности: $x = R\sin(t)$, $y = R\cos(t)$, где R — радиус, t — параметр, изменяющий свое значение от 0 до 2π.