лекция №3

Тема: Двумерное векторное изображение. списки элементов

цель: Создать формат сохранения векторного изображения. Добавить подвижные элементы.

ПЛАН

1. Изображения "страус".

2. Задание изображения в виде списка точек и полигонов.

3. Реализация анимации.

1. Изображения страус

В прошлых работах в качестве примеров использовались простые изображения. Для более сложных изображений, таких как "страус" на рис. 1, рисование является более сложным. Более сложной проблемой является задание произвольного векторного рисунке, который может храниться в файле и в композициях сменять друг друга.

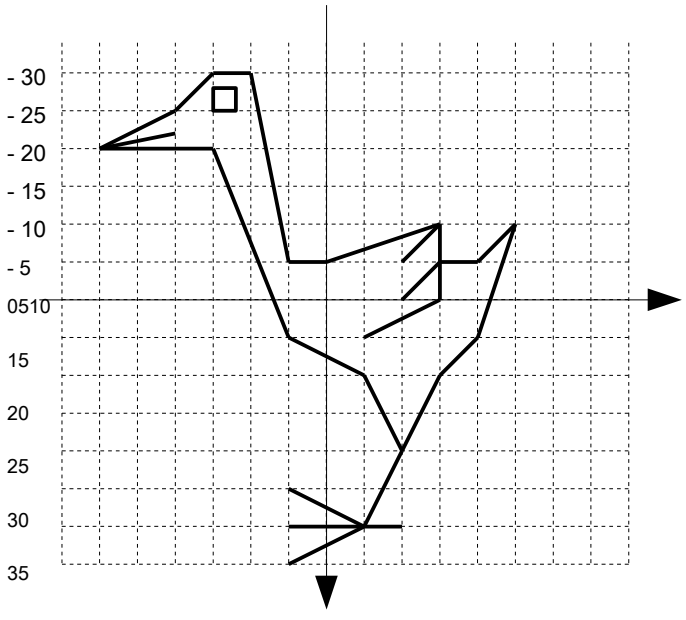

Рис. 1 - Векторное изображение "страус"

Чтобы нарисовать такого страуса нужно начертить 28 линий, при этом использовать циклы для упрощения рисования и уменьшение строчек кода нельзя. Поэтому код для рисования будет слишком длинным и приводится здесь только часть кода без настройки:

MLine (-30, -20, -20, -25) MyLine (-20, -25, -15, -30); MLine (15, 30, 10, -30);
MyLine (-10, -30, -5, -5) MLine (5, -5,0, -5) MyLine (0, -5,15, -10) MLine (15
-10,15,0) MyLine (15,0,5,5) MLine (15 -5,20, -5) MyLine (20 -5,25, -10) MLine (25
-10,20,5) MyLine (20,5,15,10) MLine (15,10,10,20) MyLine (10,20,5,10) MLine
(5,10, -5,5) MyLine (-5,5, 15, -20) MLine (15, 20, 30, -20) MyLine (-30, -20, -20,
-23) MLine (10,20,5,30) MyLine (-5,30,10,30) MLine (-5,25,5,30) MyLine
(-5,35,5,30) MLine (10,0,15, -5) MyLine (10 -5,15, -10) MLine (15, 25, 15, -28)
MyLine (15, -28, -12, -28) MLine (12, -28, -12, -25) MyLine (12, 25, 15, -25)

Разумеется, каждый объект для рисования таким образом задавать довольно обременительно, поэтому предлагается записывать последовательность пар точек в файл, а потом читать с него данные.

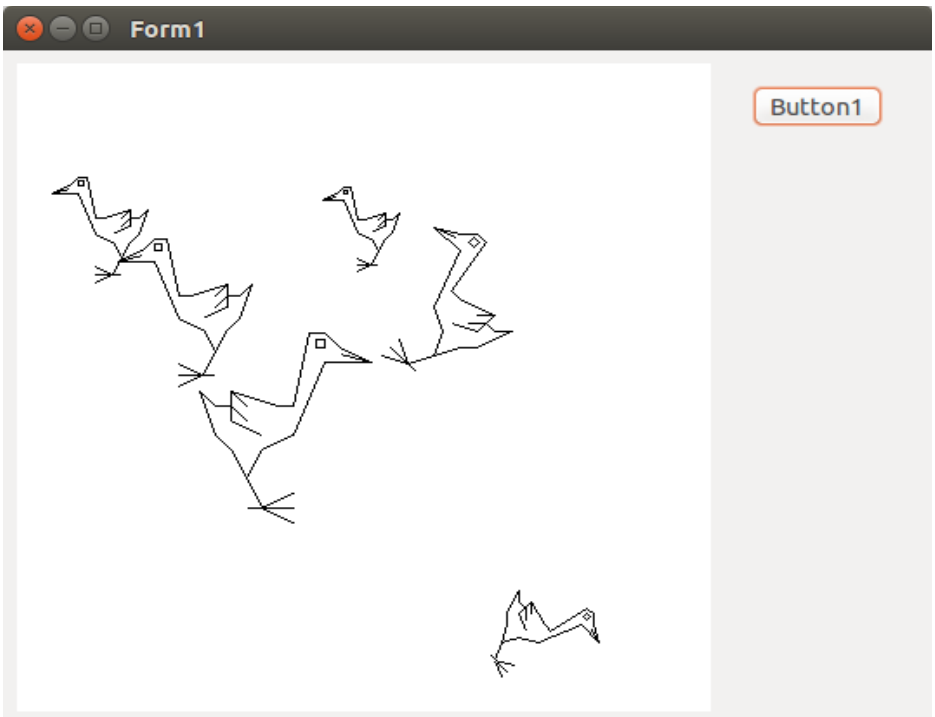

Рис. 2 - Преобразование изображения "страус"

Также легко заметить - большинство точек при рисования встречается несколько раз. Это явление довольно распространенное и его можно использовать для экономии размера файла, уменьшение параметров вызова процедур и функций и уменьшению объема расчетов. Для такой экономии данные изображения разбиваются на два массива: 1) массив координат точек, которые получают свой порядковый номер; 2) последовательности номеров точек, образующих ломаную линию (каждая ломаная завершается кодовым номером 1). Для "страуса" эти таблицы будут иметь вид:

30

- 30 -20

- 20 -25

- 15 -30

- 10 -30

- 5 -5 0 -5

15 -10 15

0 5 5 15

-5 20 -5

25 -10 20

5 15 10 10

20 5 10

- 5 мая

- 15 -20

- 20 -23 30

мая

- 30 мая

30 октября

- 25 мая

- 5 35

10 0 10

-5

- 15 -25
- 15 -28
- 12 -28
- 12 -25 45

0 1 2 3 4 5 6 7 8 -1

9 10 11 12 13 14 15 16 17 0 18 -1 20 21 -1 22 19 -1 23

19 -1 24 9 -1 25 6 -1

26 27 28 29 26 -1 14 19 -1

Эти числа можно сохранять в различных форматах в файле и наиболее экономным и быстрым методом для загрузки является бинарный вид. Но для учебных целей более наглядно будет использование обычных текстовых файлов. Скопируйте эти данные в текстовый файл straus.txt и расположите его в папке проекта, где рисовался собачка.

2. Задание изображения в виде списка точек и полигонов

Отныне не является удобным хранения процедур рисования одного предмета в одной функции. Для этого теперь бесспорно является более существенным использования структуры:

```
TMPoint = record
    x, y: Double;
end;
TMyImage = record
    Points: array of TMPoint; PolyLine: array of
  Integer; end;
```

Для такой записи необходимо включить процедуры считывания из файла и рисования:

```
procedure TForm1.LoadPicFromFile (var Image: TMyImage; FileName: String); var
```

```
  i, n: Integer; p:
 TMPoint; F: Text; begin


  system.assign (F, FileName); system.reset
  (F) ReadLn (F, n)

  SetLength (Image.Points, n) for i: = 0 to n-1
  do begin
      ReadLn (F, px, py) Image.Points
    [i]: = p; end;

  ReadLn (F, n)
  SetLength (Image.PolyLine, n) for i: = 0 to n-1
  do begin
      Read (F, Image.PolyLine [i]); end;

  system.close (F) end;
```

Приведенная процедура принимает в параметрах объект для рисования и название файла. После открытия файла в виде текстового на чтение, сначала считывается количество точек в фигуре. По количеству точек корректируется длина массива точек. В цикле зчитуетються координатные пары и заносятся в массив.

По завершению считывания точек, считывается количество целых чисел, задающих все линии будущего рисунка. Также с помощью цикла все эти значения заносятся в массив. Теперь можно создать функцию для рисования считанного фигуры:

```
procedure TForm1.DrawMyImage (var Image: TMyImage) var

  t, i, n: Integer; p: TMPoint;
 begin

  n = Length (Image.PolyLine) t = - 1;

  for i: = 0 to n-1 do begin
      if (t> = 0) and (Image.PolyLine [i]> = 0) then begin
          MLine (Image.Points [t] .x,
                  Image.Points [t] .y,
                  Image.Points [Image.PolyLine [i]]. X, Image.Points
              [Image.PolyLine [i]]. Y) end;
```

```
t = Image.PolyLine [i]; end; end;
```

В результате выполнения такой процедуры, иметь изображение страуса, которое масштабируется, переносится и вращается (рис. 2).

3. Реализация анимации

В большинстве случаев компьютерная анимация производится с помощью изменения рассчитанных отдельно кадров. Во времена малой мощности персональных компьютеров такой подход не позволял делать анимацию с частотой смены кадров для приятного восприятия движения, поэтому широко использовались методы частичного перерисовки кадра. Сегодня такой проблемы нет, поэтому рассмотрим метод полного перерисовки кадра.

Для того чтобы вызвать процедуру рисования несколько раз в секунду, используем компонент вкладке System → TTimer. Этот компонент имеет свойство Interval, которая определяет количество миллисекунд между вызовами процедуры onTimer. Эту процедуру легко создать двойным кликом на значке компонента, который уже находится на произвольном месте формы.

Для подсчета кадров добавим к TForm1 свойство frameCount: Integer ;. В зависимости от показателя таймеру будем менять горизонтальное положение "страусу" и его масштабирования

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

    frameCount: = 0;
    Tra = TTransformer.Create;
    LoadPicFromFile (Straus, "straus.txt '); Timer1.Enabled = true;
  end;


procedure TForm1.FormDestroy (Sender: TObject); begin

    Timer1.Enabled = false; Tra.Destroy;
  end;


procedure TForm1.Timer1Timer (Sender: TObject); begin
```

```
    Image1.Canvas.Brush.Color = clWhite;
    Image1.Canvas.FillRect (0,0, Image1.Width, Image1.Height) Tra.SetXY (200 + Round (150.0
    * sin (0.05 * frameCount)),
                        200 - abs (Round (10.0 * sin (0.4 * frameCount))))
    Tra.SetAlpha (0.0)
    if cos (0.05 * frameCount)> 0.0 then begin
        Tra.SetMXY (-1.0,1.0) end else
    begin
        Tra.SetMXY (1.0,1.0) end;

    DrawMyImage (Straus) frameCount =
  frameCount + 1; end;
```

Если использовать изменены функции, мы увидим, что страус весело начал скакать от края области рисования до предела. Благодаря условному оператору с выбором знака масштабирования, страус меняет направление рисования и его клюв направлен в сторону движения.

Дополнение. Подвижной страус.

```
unit Unit1;

{$ Mode objfpc} {$ H +}

interface

uses
     Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs,
ExtCtrls,
  StdCtrls;

type

  TMPoint = record
    x, y: Double;
  end;
  TMyImage = record
    Points: array of TMPoint; PolyLine: array of
   Integer; end;


  {TForm1}

  TTransformer = class public

    x, y: integer;
    mx, my: double; alpha:
    double;
    procedure SetXY (dx, dy: integer); procedure SetMXY (masX,
    masY: double) procedure SetAlpha (a: double)

    function GetX (oldX, oldY: integer): integer; function GetY (oldX, oldY:
   integer): integer; end;


  TForm1 = class (TForm)
    Button1: TButton; Image1:
    TImage; Timer1: TTimer;
```

```
        procedure Button1Click (Sender: TObject); procedure FormCreate
       (Sender: TObject); procedure FormDestroy (Sender: TObject);
       procedure Timer1Timer (Sender: TObject); private


        {Private declarations} public

        {Public declarations} Tra:
        TTransformer; Straus: TMyImage;
        frameCount: Integer;

        procedure MLine (x1, y1, x2, y2: integer); procedure MEllipse (x1, y1, x2,
        y2: integer); procedure DrawPesik; procedure DrawStraus;


        procedure LoadPicFromFile (var Image: TMyImage; FileName: String); procedure DrawMyImage (var Image:
      TMyImage) end;


var
    Form1: TForm1;

implementation

{$ R * .lfm}

{TForm1}

procedure TForm1.DrawMyImage (var Image: TMyImage) var

    t, i, n: Integer;
begin
    n = Length (Image.PolyLine) t = - 1;

    for i: = 0 to n-1 do begin
        if (t> = 0) and (Image.PolyLine [i]> = 0) then begin
            MLine (Round (Image.Points [t] .x),
                    Round (Image.Points [t] .y),
                    Round (Image.Points [Image.PolyLine [i]]. X), Round (Image.Points
                [Image.PolyLine [i]]. Y)); end;
```

```
        t = Image.PolyLine [i]; end; end;



procedure TForm1.LoadPicFromFile (var Image: TMyImage; FileName: String); var

    i, n: Integer; p:
   TMPoint; F: Text; begin


    system.assign (F, FileName); system.reset
    (F) ReadLn (F, n)

    SetLength (Image.Points, n) for i: = 0 to n-1
    do begin
        ReadLn (F, px, py) Image.Points
      [i]: = p; end;

    ReadLn (F, n)
    SetLength (Image.PolyLine, n) for i: = 0 to n-1
    do begin
        Read (F, Image.PolyLine [i]); end;

    system.close (F) end;


procedure TForm1.DrawPesik; var

    i: integer; begin

    Image1.Canvas.Pen.Color = clBlack; for i: = 0 to 7 do begin


        MLine (i * 2, 10 i * 2, 20); end;

    for i: = 0 to 7 do begin

        MLine (14 + i * 2, 0, 14 + i * 2, 20); end;

    for i: = 0 to 20 do begin
```

```
        MLine (24 + i * 2, 20, 24 + i * 2, 40); end;

    for i: = 0 to 8 do begin

        MLine (62 + i * 2, 20, 62 + i * 2, 25); end;

    Image1.Canvas.Brush.Color = clWhite; MEllipse (12, 3, 17, 8);
  end;


procedure TForm1.DrawStraus; begin

    MLine (-30, -20, -20, -25) MLine (-20, -25,
    -15, -30); MLine (15, 30, 10, -30); MLine
    (-10, -30, -5, -5) MLine (5, 5, 0, -5) MLine (0,
    5, 15, -10) MLine (15 -10, 15, 0); MLine (15
    0, 5, 5); MLine (15 -5, 20, -5) MLine (20 -5,
    25, -10) MLine (25 -10, 20, 5); MLine (20, 5,
    15, 10); MLine (15, 10, 10, 20); MLine (10,
    20, 5, 10); MLine (5, 10, 5, 5); MLine (5, 5,
    15, -20) MLine (15, 20, 30, -20) MLine (-30,
    -20, -20, -23) MLine (10, 20, 5, 30); MLine
    (5, 30, 10, 30); MLine (5, 25, 5, 30); MLine
    (5, 35, 5, 30); MLine (10 0, 15 -5) MLine (10
    -5, 15, -10) MLine (15, 25, 15, -28) MLine
    (15, -28, -12, -28) MLine (12, -28, -12, -25)
    MLine (12, 25, 15, -25) end;
```

```
procedure TForm1.Button1Click (Sender: TObject); begin

    Image1.Canvas.Brush.Color = clWhite;
    Image1.Canvas.FillRect (0, 0, Image1.Width, Image1.Height) Tra.SetAlpha (0.0) Tra.SetXY (50,
    100); Tra.SetMXY (1.0, 1.0); DrawMyImage (Straus) Tra.SetXY (100, 150); Tra.SetMXY (1.4, 1.4);
    DrawMyImage (Straus) Tra.SetXY (150, 220); Tra.SetMXY (-1.8, 1.8); DrawMyImage (Straus)
    Tra.SetXY (200, 100); Tra.SetMXY (0.8, 0.8); DrawMyImage (Straus) Tra.SetAlpha (PI / 4.0);
    Tra.SetXY (250, 150); Tra.SetMXY (1.4, 1.4); DrawMyImage (Straus) Tra.SetXY (300, 350);
    Tra.SetMXY (-1.0, 1.0); DrawMyImage (Straus) end;
```

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

    frameCount: = 0;
    Tra = TTransformer.Create; LoadPicFromFile (Straus,
  "straus.txt '); Timer1.Enabled = true; end;
```

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

    Timer1.Enabled = false; Tra.Destroy;
  end;
```

```
procedure TForm1.Timer1Timer (Sender: TObject); begin

    Image1.Canvas.Brush.Color = clWhite;
    Image1.Canvas.FillRect (0,0, Image1.Width, Image1.Height)
                Tra.SetXY (200 + Round (150.0 * sin (0.05 * frameCount)), 200 -
abs (Round (10.0 * sin (0.4 * frameCount))))
    Tra.SetAlpha (0.0)
    if cos (0.05 * frameCount)> 0.0 then begin
        Tra.SetMXY (-1.0,1.0) end else
    begin
        Tra.SetMXY (1.0,1.0) end;

    DrawMyImage (Straus) frameCount =
  frameCount + 1; end;


procedure TForm1.MLine (x1, y1, x2, y2: integer); begin

    Image1.Canvas.Line (Tra.GetX (x1, y1),
        Tra.GetY (x1, y1), Tra.GetX
     (x2, y2), Tra.GetY (x2, y2)) end;



procedure TForm1.MEllipse (x1, y1, x2, y2: integer); begin

    Image1.Canvas.Ellipse (Tra.GetX (x1, y1),
        Tra.GetY (x1, y1), Tra.GetX
     (x2, y2), Tra.GetY (x2, y2)) end;



{TTransformer}

procedure TTransformer.SetXY (dx, dy: integer); begin

    x = dx; y = dy;
  end;


procedure TTransformer.SetMXY (masX, masY: double)
```

```
begin
    mx = masX; my =
  masY; end;


function TTransformer.GetX (oldX, oldY: integer): integer; begin

    GetX = Round (mx * oldX * cos (alpha) - my * oldY * sin (alpha) + x) end;


function TTransformer.GetY (oldX, oldY: integer): integer; begin

    GetY = Round (mx * oldX * sin (alpha) + my * oldY * cos (alpha) + y) end;


procedure TTransformer.SetAlpha (a: double) begin

    alpha = a; end;


end.
```