Лекція №3

Тема: Двовимірне векторне зображення. Списки елементів

Мета: Створити формат збереження векторного зображення. Додати рухомі елементи.

ПЛАН

1. Зображення “страус”.

2. Задання зображення у вигляді списку точок та полігонів.

3. Реалізація анімації.

1. Зображення страус

В минулих роботах в якості прикладів використовувались прості зображення. Для більш складних зображень, таких як “страус” на рис. 1, малювання є більш складним. Більш складною проблемою є задання довільного векторного малюнку, який може зберігатися у файлі і в композиціях змінювати один одного.

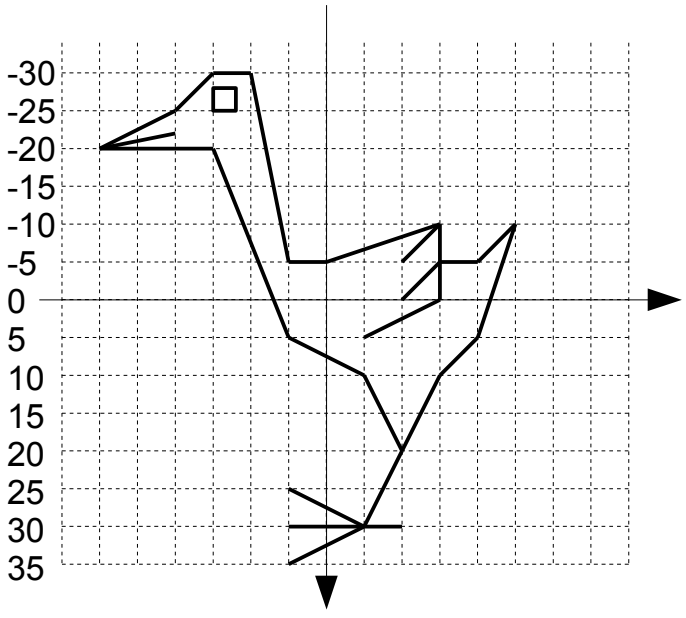

Рис. 1 — Векторне зображення “страус”

Щоб намалювати такого страуса потрібно накреслити 28 ліній, при цьому використати цикли для спрощення малювання та зменшення строчок коду не можна. Тому код для малювання буде занадто довгим і наводиться тут лише частина коду без налаштування:

```
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);
```

Зрозуміло, кожен об'єкт для малювання таким чином задавати досить обтяжливо, тому пропонується записувати послідовність пар точок у файл, а потім читати з нього дані.

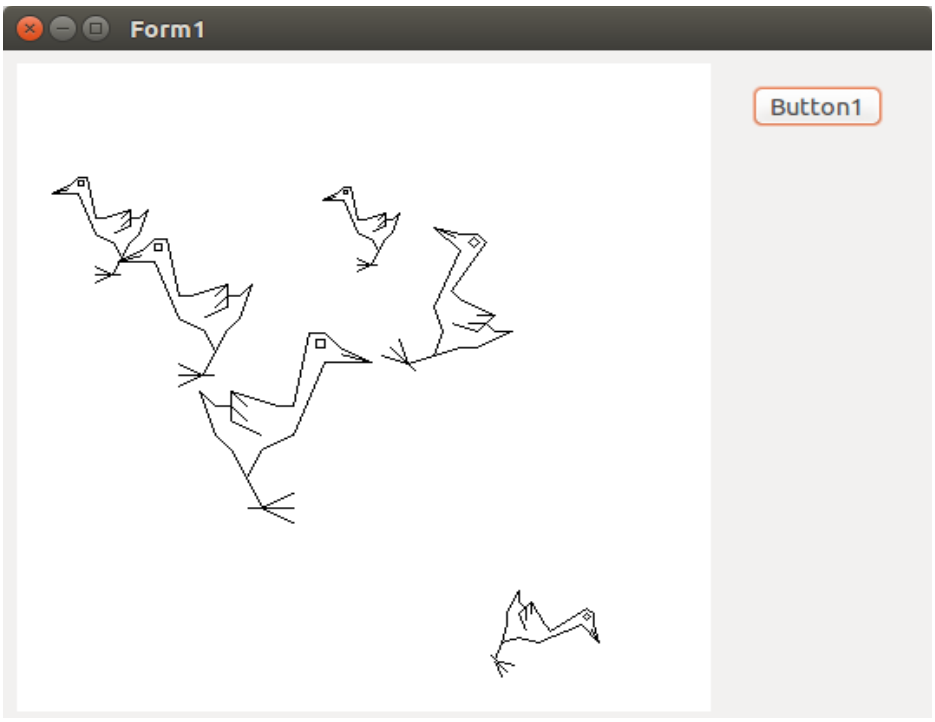

Рис. 2 — Перетворення зображення “страус”

Також легко помітити — більшість точок при малювання зустрічається декілька раз. Це явище досить розповсюджене і його можна використати для економії розміру файлу, зменшення параметрів виклику процедур та функцій та зменшенню обсягу розрахунків. Для такої економії дані зображення розбиваються на два масиви: 1) масив координат точок, які отримують свій порядковий номер; 2) послідовності номерів точок, які утворюють ламану лінію (кожна ламана завершується кодовим номером -1). Для “страуса” ці таблиці матимуть вигляд:

```
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 5
-15 -20
-20 -23
5 30
-5 30
10 30
-5 25
-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
```

Ці числа можна зберігати в різних форматах у файлі і найбільш економним та швидким методом для завантаження є бінарний вигляд. Але для навчальних цілей більш наочно буде використання звичайних текстових файлів. Скопіюйте ці дані в текстовий файл straus.txt й розташуйте його в теці проекту, де малювався песик.

2. Задання зображення у вигляді списку точок та полігонів

Відтепер не є зручним зберігання процедур малювання одного предмету в одній функції. Для цього тепер безперечно є більш істотним використання структури:

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

Для такого запису необхідно долучити процедури зчитування з файлу та малювання:

```
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,p.x,p.y);
    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;
```

Наведена процедура приймає в параметрах об'єкт для малювання та назву файла. Після відкриття файлу як текстового на читання, спершу зчитується кількість точок в фігурі. За кількістю точок корегується довжина масиву точок. В циклі зчитуєтються координатні пари й заносяться до масиву.

По завершенню зчитування точок, зчитується кількість цілих чисел, які задають всі лінії майбутнього малюнка. Також за допомогою циклу всі ці значення заносяться до масиву. Тепер можна створити функцію для малювання зчитаної фігури:

```
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 результаті виконання такої процедури, матимемо зображення страуса, яке масштабується, переноситься та обертається (рис. 2).

3. Реалізація анімації

В більшості випадків комп'ютерна анімація проводиться за допомогою зміни розрахованих окремо кадрів. В часи малої потужності персональних комп'ютерів такий підхід не давав змоги робити анімацію з частотою зміни кадрів для приємного сприйняття руху, тому широко використовувалися методи часткового перемалювання кадру. Сьогодні такої проблеми не має, тому розглянемо метод повного перемалювання кадру.

Для того щоб викликати процедуру малювання декілька раз за секунду, використаємо компонент з вкладки System → TTimer. Цей компонент має властивість Interval, яка визначає кількість мілісекунд між викликами процедури onTimer. Цю процедуру легко створити подвійним кліком на значку компонента, який вже розташований на довільному місці форми.

Для рахування кадрів додамо до TForm1 властивість frameCount:Integer;. В залежності від показника таймеру будемо змінювати горизонтальне положення "страусу" та його масштабування:

```
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;
```

Якщо використати змінені функції, ми побачимо, що страус весело почав скакати від краю області малювання до краю. Завдяки умовному оператору з вибором знаку масштабування, страус змінює напрям малювання і його дзьоб направлений в сторону руху.

Додаток. Рухомий 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,p.x,p.y);
    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.
```