Лабораторна робота №2

Тема: Двовимірне векторне зображення

Мета: Написати програму виведення на екран векторних малюнків з використанням перетворень переносу, масштабування та повороту.

ПЛАН

1. Пласке векторне зображення з відрізків.

2. Доповнення виводу малюнка масштабуванням.

3. Доповнення виводу малюнка обертанням.

1. Пласке векторне зображення з відрізків

Для конкретизації викладення матеріалу визначимося метою створення зображення песика, який зображений на наступному рисунку:

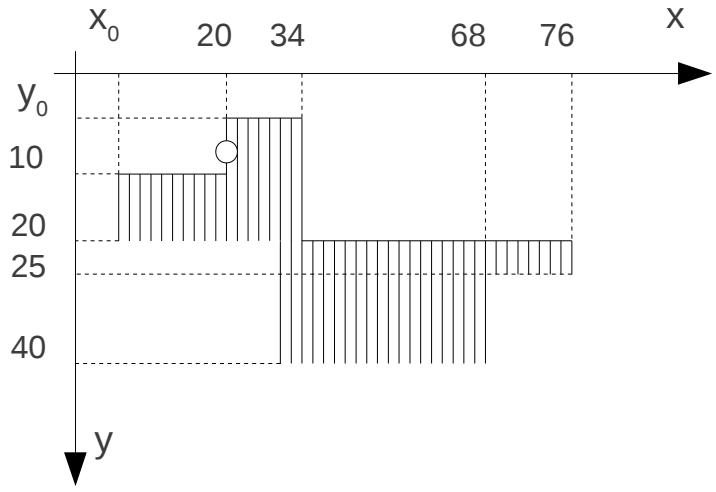

Рис. 1 — Векторне зображення собаки

Код для малювання такого песика матиме вигляд:

```
  Image1.Canvas.Pen.Color:=clBlack;
//Малюємо мордочку
```

```
  for i:=0 to 7 do begin
    Image1.Canvas.Line(i*2,10,i*2,20);
  end;
//Голова:
  for i:=0 to 7 do begin
    Image1.Canvas.Line(14+i*2,0,14+i*2,20);
  end;
//Тулуп
  for i:=0 to 20 do begin
    Image1.Canvas.Line(24+i*2,20,24+i*2,40);
  end;
//Хвіст
  for i:=0 to 8 do begin
    Image1.Canvas.Line(62+i*2,20,62+i*2,25);
  end;
//Око
  Image1.Canvas.Brush.Color:=clWhite;
  Image1.Canvas.Ellipse(12,3,17,8);
```

Однак, нам потрібно малювати песика в довільному місці, яке задається початковою точкою (x;y). Для цього достатньо змістити малюнок на задану величину:

```
//Малюємо мордочку
  Image1.Canvas.Line(x+i*2,y+10,x+i*2,y+20);
//Голова:
  Image1.Canvas.Line(x+14+i*2,y,x+14+i*2,y+20);
//Тулуп
  Image1.Canvas.Line(x+24+i*2,y+20,x+24+i*2,y+40);
//Хвіст
  Image1.Canvas.Line(x+62+i*2,y+20,x+62+i*2,y+25);
//Око
  Image1.Canvas.Ellipse(x+12,y+3,x+17,y+8);
```

Тепер код можна оформити окремою процедурою, яка прийматиме значення зміщення (x,y :Integer):

```
procedure TForm1.DrawPesik(x,y: Integer);
var i: Integer;
begin
//Малюємо песика
...
end;
```

Наступним кроком створимо процедуру, яка викликатиметься при натиснені кнопки та, після очищення зони малювання білим прямокутником, можемо малювати песиків. Результат показано на рис. 2.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Brush.Color:=clWhite;
  Image1.Canvas.FillRect(0,0,Image1.Width,Image1.Height);
  DrawPesik(100,100);
  DrawPesik(150,150);
  DrawPesik(200,200);
  DrawPesik(250,100);
  DrawPesik(300,150);
end;
```

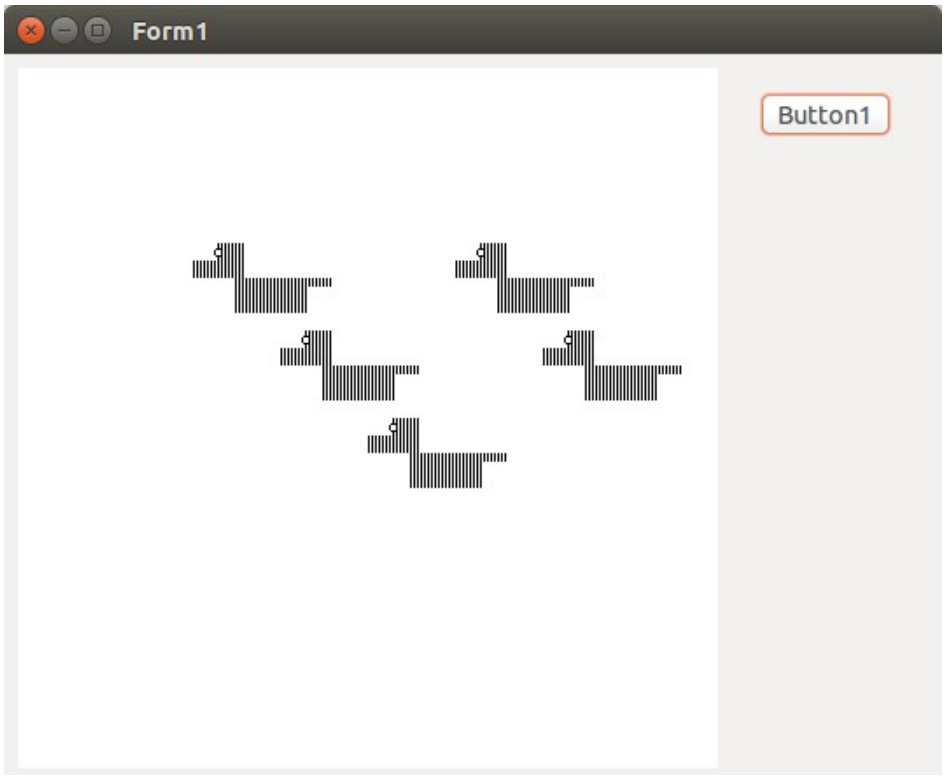

Рис. 2 — Розмноження зображення собак

Не зовсім зручно використовувати малювання ліній вказаним чином. До координат, що визначають лінії потрібно додавати x та y, це робить запис досить громіздким. Надалі, при додаванні інших перетворень, такі записи будуть ускладнені ще сильніше. Змінимо запис малювання песика й порівняємо з попередніми записами:

```
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;
```

Такий код є більш звичним і є ідентичним малюванню одиничного об’єкту. Зміщення песику тепер відбувається в середини власної процедури малювання лінії, код якої показаний

в наступному лістингу:

```
procedure TForm1.MLine(x1,y1,x2,y2:Integer);
begin
  Image1.Canvas.Line(x1+Tra.x, y1+Tra.y,
                     x2+Tra.x, y2+Tra.y);
end;
```

Тепер початкову точку малювання будь-якого об’єкту можна задати за допомогою глобального об’єкту, опис якого показано нижче:

```
  TTransformer = class
    public
      x, y: Integer;
      procedure SetXY(dx,dy: Integer);
  end;
procedure TTransformer.SetXY(dx,dy: Integer);
begin   x:=dx; y:=dy; end;
```

Також змінюється й код виводу на екран й зображень собак:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Brush.Color:=clWhite;
  Image1.Canvas.FillRect(0,0,Image1.Width,Image1.Height);
  Tra.SetXY(100,100);   DrawPesik;
  Tra.SetXY(150,150);   DrawPesik;
  Tra.SetXY(200,200);   DrawPesik;
  Tra.SetXY(250,100);   DrawPesik;
  Tra.SetXY(300,150);   DrawPesik;
end;
```

Ці зміни роблять код виклику малювання самої собаки дещо більш обтяжливим, код зміни положення відірвано від коду малювання, але це з лишком компенсувалося в інших процедурах і, як побачимо далі, виграш буде лише збільшуватися з додаванням функціональності. Запустивши змінену програму ми не побачимо змін, що означає адекватність змін координат що одним та іншим методом. Остаточний код малювання песиків наведено в додатку А.

2. Доповнення виводу малюнка масштабуванням

Наступним кроком є додавання коефіцієнтів масштабування окремо по кожній з координат. Тоді для малювання потрібно перераховувати координати за такими співвідношеннями: $x_{new}=x\cdot m_x+d_x$, $y_{new}=y\cdot m_y+d_y$, де m позначає коефіцієнт масштабування, d — зміщення малюнку від початку координат. Потрібно також врахувати особливості мови програмування — PASCAL забороняє до цілочислових змінних записувати дійсні числа безпосередньо, для цього потрібно використати функцію округлення. Такі розрахунки зручно оформити окремими функціями класу перетворення:

```
procedure TTransformer.SetMXY(masX, masY: Double);
begin   mx:=masX; my:=masY; end;

function TTransformer.GetX(oldX:Integer):Integer;
begin   GetX := Round( mx*oldX + x); end;

function TTransformer.GetY(oldY:Integer):Integer;
begin   GetY := Round( my*oldY + y); end;
```

Відповідно змінам, потрібно вивести обчислення координат з малювання лінії та еліпсу:

```
procedure TForm1.MLine(x1,y1,x2,y2:Integer);
begin
  Image1.Canvas.Line(Tra.GetX(x1), Tra.GetY(y1),
                     Tra.GetX(x2), Tra.GetY(y2));
end;

procedure TForm1.MEllipse(x1,y1,x2,y2:Integer);
begin
  Image1.Canvas.Ellipse(Tra.GetX(x1), Tra.GetY(y1),
                     Tra.GetX(x2), Tra.GetY(y2));
end;
```

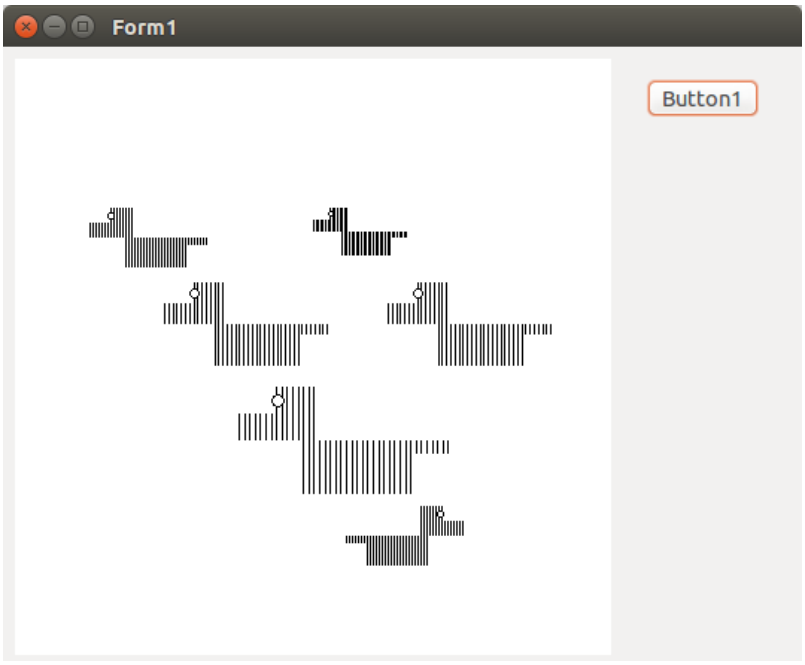

Рис. 3 — Масштабовані собаки

Завдяки введенню додаткового класу, який відповідає за перетворення координат, можна виконати малювання собак різного розміру:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Brush.Color:=clWhite;
  Image1.Canvas.FillRect(0,0,Image1.Width,Image1.Height);
  Tra.SetXY(50,100); Tra.SetMXY(1.0,1.0);
  DrawPesik;
  Tra.SetXY(100,150); Tra.SetMXY(1.4,1.4);
  DrawPesik;
  Tra.SetXY(150,220); Tra.SetMXY(1.8,1.8);
  DrawPesik;
  Tra.SetXY(200,100); Tra.SetMXY(0.8,0.8);
  DrawPesik;
  Tra.SetXY(250,150); Tra.SetMXY(1.4,1.4);
  DrawPesik;
  Tra.SetXY(300,300); Tra.SetMXY(-1.0,1.0);
  DrawPesik;
end;
```

Як бонус сумлінного виконання математичних правил, ми отримали можливість

малювати собак не лише різного розміру, але й їх віддзеркалення: для нижньої собаки використано масштабний коефіцієнт по горизонталі -1, що й призвело до її вимальовування в зворотньому напрямку.

Повний лістинг варіанту програмного коду з масштабуванням векторної собачки наведено в додатку Б.

3. Доповнення виводу малюнка обертанням

Останнім кроком в перетворенні двовимірних координат є обертання зображення на певний кут. Для цього потрібно знайти формули переходу від однієї системи координат до іншої при відомому куті обертання. Для цього зобразимо ці дві системи координат, що повернуті одна відносно одної:

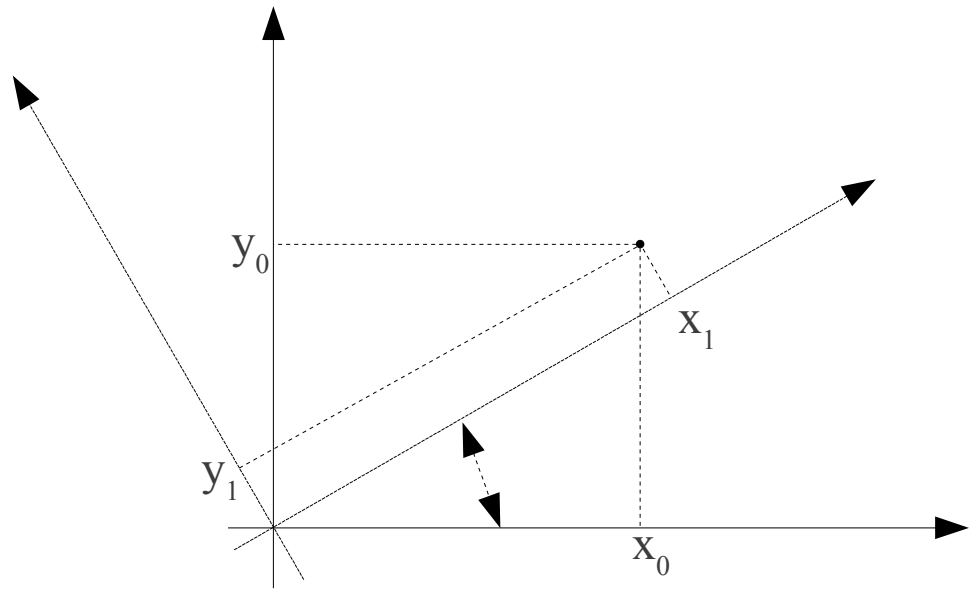

Рис. 4 — Поворот координатних осей

Кут повороту позначено як α =30°, при цьому на рис. 4 видно, що координати $(x_0; y_0)$ переходять до координат $(x_1; y_1)$. Розв'язувати задачу пошуку формул для знаходження $(x_1; y_1)$ з відомих $(x_0; y_0)$ можна декількома способами. На цей раз можна скористатися аналітичною геометрією.

Визначимо рівняння прямої, яка відповідатиме повернутій вісі OX. Ми знаємо, що коефіцієнт нахилу прямої $y = kx + c$ має залежність $k = \tan(\alpha)$, або $k = \sin(\alpha)/\cos(\alpha)$. Так як нова координатна вісь проходить через початок координат, вільний коефіцієнт $c = 0$.

В цілому рівняння прямої матиме вигляд $y = x \cdot \sin(\alpha)/\cos(\alpha) \Rightarrow x \cdot \sin(\alpha) - y \cdot \cos(\alpha) = 0$. Тоді координатою y_1 буде відстань від точки $(x_1; y_1)$ до прямої, що збігається новою віссю: $y_1 = x_0 \cdot \sin(\alpha) - y_0 \cdot \cos(\alpha)$.

Аналогічно, вертикальна вісь матиме коефіцієнт нахилу $k = -ctan(\alpha)$, або $k = -\cos(\alpha)/\sin(\alpha)$. Цьому коефіцієнту нахилу відповідає рівняння прямої $y = -x \cdot \cos(\alpha)/\sin(\alpha) \Rightarrow x \cdot \cos(\alpha) + y \cdot \sin(\alpha) = 0$. Тоді координатою x_1 буде відстань від точки $(x_0; y_0)$ до прямої, що збігається новою вертикальною віссю: $x_1 = x_0 \cdot \cos(\alpha) + y_0 \cdot \sin(\alpha)$.

Остаточно, для знаходження координат точки $(x_0; y_0)$ в системі координат, що повернута на кут α $(x_1; y_1)$, потрібно скористатися виразами:

$$\begin{cases} x_1 = x_0 \cdot \cos(\alpha) + y_0 \cdot \sin(\alpha), \\ y_1 = x_0 \cdot \sin(\alpha) - y_0 \cdot \cos(\alpha). \end{cases}$$

Комбінація повороту з переміщенням на вектор $(dx; dy)$ дає повну формулу переміщення точки з її поворотом:

$$\begin{cases} x_1 = x_0 \cdot \cos(\alpha) + y_0 \cdot \sin(\alpha) + dx, \\ y_1 = x_0 \cdot \sin(\alpha) - y_0 \cdot \cos(\alpha) + dy. \end{cases}$$

Для врахування обертання в програмному коді потрібно необхідно модифікувати трансформер додаванням кута повороту, функції отримання нових координат повинні приймати вже не одну координату а дві:

```
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;
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;
```

Відповідно й малювання відбувається трохи по іншому:

```
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(x1,y2));
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);
  DrawPesik;
  Tra.SetXY(100,150); Tra.SetMXY(1.4,1.4);
  DrawPesik;
  Tra.SetXY(150,220); Tra.SetMXY(1.8,1.8);
  DrawPesik;
  Tra.SetXY(200,100); Tra.SetMXY(0.8,0.8);
  DrawPesik;
  Tra.SetAlpha(PI/4.0); //Тут змінено кут
  Tra.SetXY(250,150); Tra.SetMXY(1.4,1.4);
  DrawPesik;
  Tra.SetXY(300,350); Tra.SetMXY(-1.0,1.0);
```

```
  DrawPesik;
end;
```

Тепер песик може змінювати не лише свій розмір та напрям, але й обертатися на довільний кут. Результат малювання можна побачити на наступному малюку:

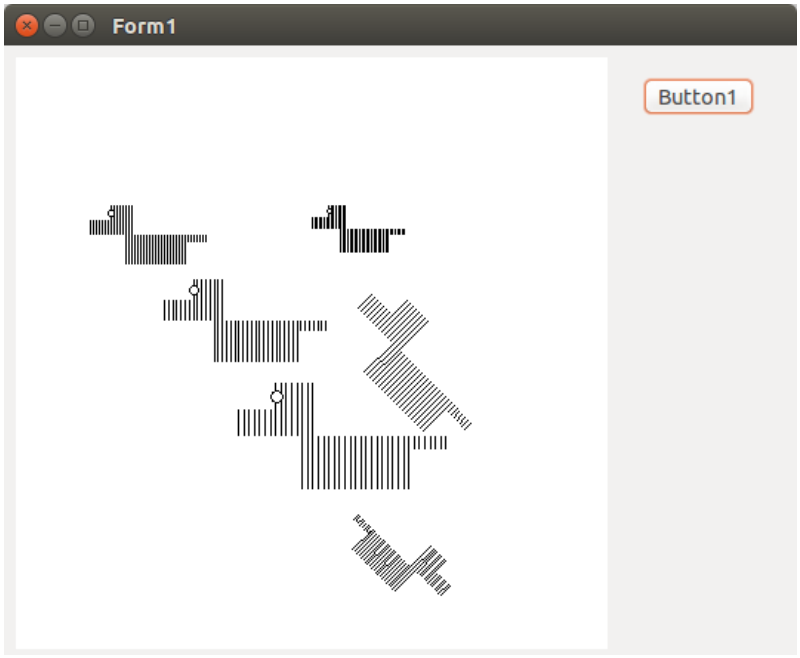

Рис. 5 — Комбінація переміщення масштабування та обертання

Додаток А. Песики.

```
unit Unit1;
{$mode objfpc}{$H+}
interface
uses
    Classes, SysUtils, FileUtil, Forms, Controls, Graphics,
Dialogs, ExtCtrls,   StdCtrls;

type
  { TForm1 }
  TTransformer = class
    public
      x, y: Integer;
      procedure SetXY(dx,dy: Integer);
  end;

  TForm1 = class(TForm)
    Button1: TButton;
    Image1: TImage;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
    Tra: TTransformer;
    procedure MLine(x1,y1,x2,y2:Integer);
    procedure MEllipse(x1,y1,x2,y2:Integer);
    procedure DrawPesik;
  end;

var
```

```
  Form1: TForm1;

implementation

{$R *.lfm}
{ TForm1 }
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.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Brush.Color:=clWhite;
  Image1.Canvas.FillRect(0,0,Image1.Width,Image1.Height);
  Tra.SetXY(100,100);   DrawPesik;
  Tra.SetXY(150,150);   DrawPesik;
  Tra.SetXY(200,200);   DrawPesik;
  Tra.SetXY(250,100);   DrawPesik;
  Tra.SetXY(300,150);   DrawPesik;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin   Tra := TTransformer.Create; end;

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

```
procedure TForm1.MLine(x1,y1,x2,y2:Integer);
begin
  Image1.Canvas.Line(x1+Tra.x,
                     y1+Tra.y,
                     x2+Tra.x,
                     y2+Tra.y);
end;

procedure TForm1.MEllipse(x1,y1,x2,y2:Integer);
begin
  Image1.Canvas.Ellipse(x1+Tra.x,
                        y1+Tra.y,
                        x2+Tra.x,
                        y2+Tra.y);
end;

procedure TTransformer.SetXY(dx,dy: Integer);
begin   x:=dx; y:=dy; end;

end.
```

Додаток Б. Масштабовані песики.

```
unit Unit1;

{$mode objfpc}{$H+}

interface

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

type

  { TForm1 }

  TTransformer = class
    public
      x, y: Integer;
      mx, my: Double;
      procedure SetXY(dx,dy: Integer);
      procedure SetMXY(masX, masY: Double);
      function GetX(oldX:Integer):Integer;
      function GetY(oldY:Integer):Integer;
  end;

  TForm1 = class(TForm)
    Button1: TButton;
    Image1: TImage;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
```

```
    { private declarations }
  public
    { public declarations }
    Tra: TTransformer;
    procedure MLine(x1,y1,x2,y2:Integer);
    procedure MEllipse(x1,y1,x2,y2:Integer);
    procedure DrawPesik;
  end;

var
  Form1: TForm1;

implementation

{$R *.lfm}

{ TForm1 }

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.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Brush.Color:=clWhite;
  Image1.Canvas.FillRect(0,0,Image1.Width,Image1.Height);
  Tra.SetXY(50,100); Tra.SetMXY(1.0,1.0);
  DrawPesik;
  Tra.SetXY(100,150); Tra.SetMXY(1.4,1.4);
  DrawPesik;
  Tra.SetXY(150,220); Tra.SetMXY(1.8,1.8);
  DrawPesik;
  Tra.SetXY(200,100); Tra.SetMXY(0.8,0.8);
  DrawPesik;
  Tra.SetXY(250,150); Tra.SetMXY(1.4,1.4);
  DrawPesik;
  Tra.SetXY(300,300); Tra.SetMXY(-1.0,1.0);
  DrawPesik;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Tra := TTransformer.Create;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  Tra.Destroy;
end;

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

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

procedure TForm1.MEllipse(x1,y1,x2,y2:Integer);
begin
  Image1.Canvas.Ellipse(Tra.GetX(x1),
                     Tra.GetY(y1),
                     Tra.GetX(x2),
                     Tra.GetY(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:Integer):Integer;
begin
  GetX := Round( mx*oldX + x);
end;

function TTransformer.GetY(oldY:Integer):Integer;
begin
  GetY := Round( my*oldY + y);
```

```
end;

end.
```

Додаток В. Собаки з поворотом

```
unit Unit1;

{$mode objfpc}{$H+}

interface

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

type

  { 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;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
```

```
    procedure FormDestroy(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
    Tra: TTransformer;
    procedure MLine(x1,y1,x2,y2:Integer);
    procedure MEllipse(x1,y1,x2,y2:Integer);
    procedure DrawPesik;
  end;

var
  Form1: TForm1;

implementation

{$R *.lfm}

{ TForm1 }

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.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);
  DrawPesik;
  Tra.SetXY(100,150); Tra.SetMXY(1.4,1.4);
  DrawPesik;
  Tra.SetXY(150,220); Tra.SetMXY(1.8,1.8);
  DrawPesik;
  Tra.SetXY(200,100); Tra.SetMXY(0.8,0.8);
  DrawPesik;
  Tra.SetAlpha(PI/4.0); //Тут змінено кут
  Tra.SetXY(250,150); Tra.SetMXY(1.4,1.4);
  DrawPesik;
  Tra.SetXY(300,350); Tra.SetMXY(-1.0,1.0);
  DrawPesik;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Tra := TTransformer.Create;
end;

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

```
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(x1,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.
```