|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **По домашнему заданию №** | 1 |

**Название:**

Программирование на Object Pascal с использованием классов

**Дисциплина:** Объектно-ориентированное программирование

Вариант 16.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-21Б |  |  | Т.Е.Старжевский |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | О.А.Веселовская |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

Часть 1.

**Текст задания:**

***Графический редактор***

Разработать иерархию классов. Поместить определение классов в отдельном модуле.

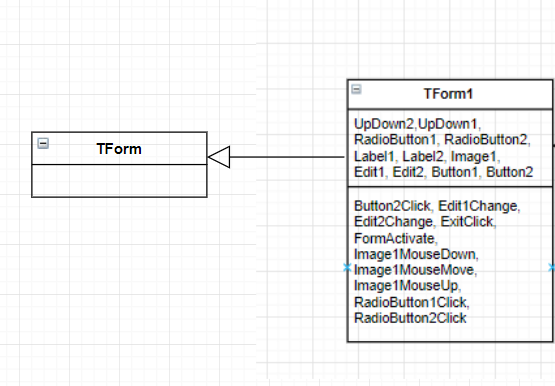
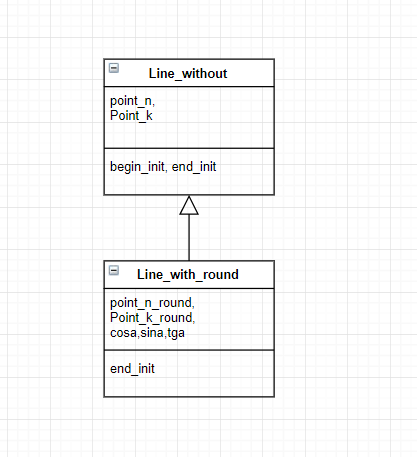
Класс, позволяющий рисовать линию, длина которой определяется нажатием и отпусканием левой клавиши мыши.

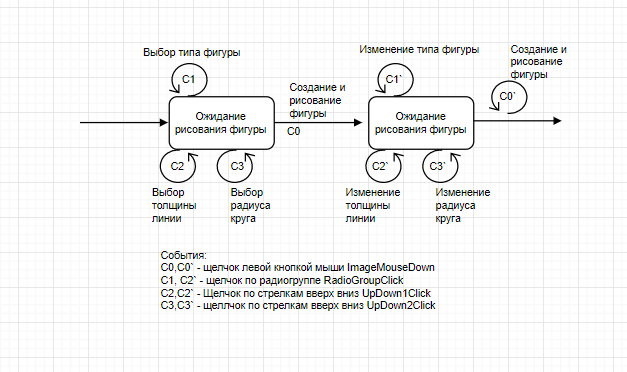
Класс, позволяющий рисовать такую же линию, но с окружностью на конце.

Вид линии и размер окружности задавать с использованием интерфейсных элементов.

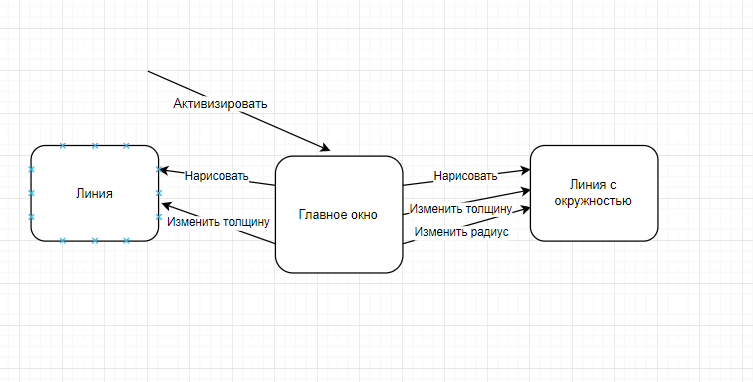
В отчете показать иерархии используемых классов VCL и разработанных классов, граф состояния пользовательского интерфейса и объектную декомпозицию.

**Иерархия классов:**

****

**Графическое состояние интерфейса:**

**Объектная декомпозиция:**

****

**Код программы:**

unit Part1\_unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls,

ComCtrls,my\_math;

type

{ TForm1 }

TForm1 = class(TForm)

Button1: TButton;

Button2: TButton;

Edit1: TEdit;

Edit2: TEdit;

Image1: TImage;

Label1: TLabel;

Label2: TLabel;

RadioButton1: TRadioButton;

RadioButton2: TRadioButton;

UpDown1: TUpDown;

UpDown2: TUpDown;

procedure Button2Click(Sender: TObject);

procedure Edit1Change(Sender: TObject);

procedure Edit2Change(Sender: TObject);

procedure ExitClick(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure Image1MouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer

);

procedure Image1MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure RadioButton1Click(Sender: TObject);

procedure RadioButton2Click(Sender: TObject);

end;

var

Form1: TForm1;

a:integer=1;

implementation

var

choose:integer=1;

key : boolean = false;

point\_n,point\_k:tpoint;

w:integer;

Line\_round:line\_with\_round;

{$R \*.lfm}

{ TForm1 }

procedure TForm1.ExitClick(Sender: TObject);

begin

Close;

end;

procedure TForm1.Edit1Change(Sender: TObject);

var code:integer;

begin

val(edit1.text,w,code);

image1.canvas.Pen.Width:=w;

end;

procedure TForm1.Edit2Change(Sender: TObject);

var code:integer;

begin

val(edit2.Text,w,code);

a:=w;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

image1.canvas.Clear;

end;

procedure TForm1.FormActivate(Sender: TObject);

begin

image1.Canvas.Brush.Color:=clwhite;

image1.canvas.Rectangle(-1,-1,10000,10000);

image1.canvas.pen.Width:=10;

// image1.Color:=clwhite;

// image1.Canvas.brush.Color:=clgreen;

end;

procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if button = mbleft then begin

key:=true;

point\_n.x:=x;

point\_n.y:=y;

end;

end;

procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

if ssleft in shift then begin

// image1.Canvas.Pen.Color:=clblack;

if key then key:=false

else begin

image1.canvas.pen.color:=clwhite;

//Image1.canvas.lineto(point\_k);

image1.canvas.line(point\_n,point\_k);

end;

point\_k.x:=x;

point\_k.y:=y;

image1.canvas.pen.color:=clred;

//image1.Canvas.LineTo(point\_k);

image1.Canvas.line(point\_n,point\_k);

end;

end;

procedure TForm1.Image1MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var code:integer=0;

begin

case choose of

1:

if button = mbleft then begin

point\_k.x:=x;

point\_k.y:=y;

image1.canvas.pen.color:=clAqua;

image1.canvas.line(point\_n,point\_k);

end;

2:

if button = mbleft then begin

point\_k.x:=x;

point\_k.y:=y;

line\_round.begin\_init(point\_n);

line\_round.end\_init(a,point\_k);

//

image1.canvas.pen.color:=clAqua;

image1.canvas.line(point\_n,point\_k);

image1.canvas.Brush.Color:=clwhite;

//image1.canvas.Pen.Width:=2;

Image1.canvas.Ellipse(line\_round.point\_n\_round.x,line\_round.point\_n\_round.y,

line\_round.point\_k\_round.x,line\_round.point\_k\_round.y);

val(edit1.text,w,code);

image1.canvas.Pen.Width:=w;

end;

end;

end;

procedure TForm1.RadioButton1Click(Sender: TObject);

begin

choose:=1;

end;

procedure TForm1.RadioButton2Click(Sender: TObject);

begin

choose:=2;

end;

end.

unit my\_math;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls,

ComCtrls, math;

Type line\_without = object

point\_n,point\_k:Tpoint;

procedure begin\_init (ipoint\_n:Tpoint);

procedure end\_init(ipoint\_k:Tpoint);

end;

line\_with\_round = object (line\_without)

point\_n\_round,point\_k\_round:Tpoint;

cosa,sina,tga:double;

procedure end\_init (a:integer;ipoint\_k:Tpoint);

end;

function situation (xn,xk,yn,yk:integer):integer;

implementation

procedure line\_without.begin\_init (ipoint\_n:Tpoint);

begin

point\_n.x:=ipoint\_n.x;

point\_n.y:=ipoint\_n.y;

end;

procedure line\_without.end\_init (ipoint\_k:Tpoint);

begin

point\_k.x:=ipoint\_k.x;

point\_k.y:=ipoint\_k.y;

end;

procedure line\_with\_round.end\_init (a:integer;ipoint\_k:Tpoint);

var b:integer = 7;

begin

inherited end\_init(ipoint\_k);

if (point\_k.x-point\_n.x) = 0 then tga := 0

else begin

tga:= abs((point\_k.Y-point\_n.y)/(point\_k.x-point\_n.x));

cosa:= sqrt(1/(1+sqr(tga)));

sina:= sqrt(1-sqr(cosa));

end;

case situation (point\_n.x,point\_k.x,point\_n.y,point\_k.y) of

1: begin point\_n\_round.x := point\_k.x - a;

point\_n\_round.y := point\_k.y + a;

point\_k\_round.x := round (point\_k.x + 5 \* a);

point\_k\_round.y := round (point\_k.y - 5 \* a);

end;

2: begin point\_n\_round.x := round (point\_k.x + a);

point\_n\_round.y := round (point\_k.y + a );

point\_k\_round.x := round (point\_k.x - 5 \* a);

point\_k\_round.y := round (point\_k.y - 5 \* a);

end;

3: begin point\_n\_round.x := round (point\_k.x + a);

point\_n\_round.y := round (point\_k.y - a);

point\_k\_round.x := round (point\_k.x - 5 \* a);

point\_k\_round.y := round (point\_k.y + 5 \* a);

end;

4: begin point\_n\_round.x := round (point\_k.x - a);

point\_n\_round.y := round (point\_k.y - a);

point\_k\_round.x := round (point\_k.x + 5 \* a);

point\_k\_round.y := round (point\_k.y + 5 \* a);

end;

5: begin point\_n\_round.x := point\_k.x - a;

point\_n\_round.y := point\_k.y + a;

point\_k\_round.x := point\_k.x + 5 \* a;

point\_k\_round.y := point\_k.y - 5 \* a;

end;

6: begin point\_n\_round.x := point\_k.x + 3 \* a;

point\_n\_round.y := point\_k.y - a;

point\_k\_round.x := point\_k.x - 3 \* a;

point\_k\_round.y := point\_k.y + 5 \* a;

end;

7: begin point\_n\_round.x := point\_k.x + 3 \* a;

point\_n\_round.y := point\_k.y + a;

point\_k\_round.x := point\_k.x - 3 \* a;

point\_k\_round.y := point\_k.y - 5 \* a;

end;

8: begin point\_n\_round.x := point\_k.x + a;

point\_n\_round.y := point\_k.y - 3 \* a;

point\_k\_round.x := point\_k.x - 5 \* a;

point\_k\_round.y := point\_k.y + 3 \* a;

end;

9: begin point\_n\_round.x := point\_k.x - a;

point\_n\_round.y := point\_k.y - 3 \* a;

point\_k\_round.x := point\_k.x + 5 \* a;

point\_k\_round.y := point\_k.y + 3 \* a;

end;

end;

end;

function situation (xn,xk,yn,yk:integer):integer;

begin

if (xk-xn>0) and (yk-yn<0) then result := 1

else if (xk-xn<0) and (yk-yn<0) then result := 2

else if (xk-xn<0) and (yk-yn>0) then result := 3

else if (xk-xn>0) and (yk-yn>0) then result := 4

else if (xk-xn=0) and (yk-yn=0) then result := 5

else if (xk-xn=0) and (yk-yn>0) then result := 6

else if (xk-xn=0) and (yk-yn<0) then result := 7

else if (xk-xn<0) and (yk-yn=0) then result := 8

else if (xk-xn>0) and (yk-yn=0) then result := 9;

end;

end.

program Part\_1;

{$mode objfpc}{$H+}

uses

{$IFDEF UNIX}{$IFDEF UseCThreads}

cthreads,

{$ENDIF}{$ENDIF}

Interfaces, // this includes the LCL widgetset

Forms, Part1\_unit1, my\_math

{ you can add units after this };

{$R \*.res}

begin

RequireDerivedFormResource:=True;

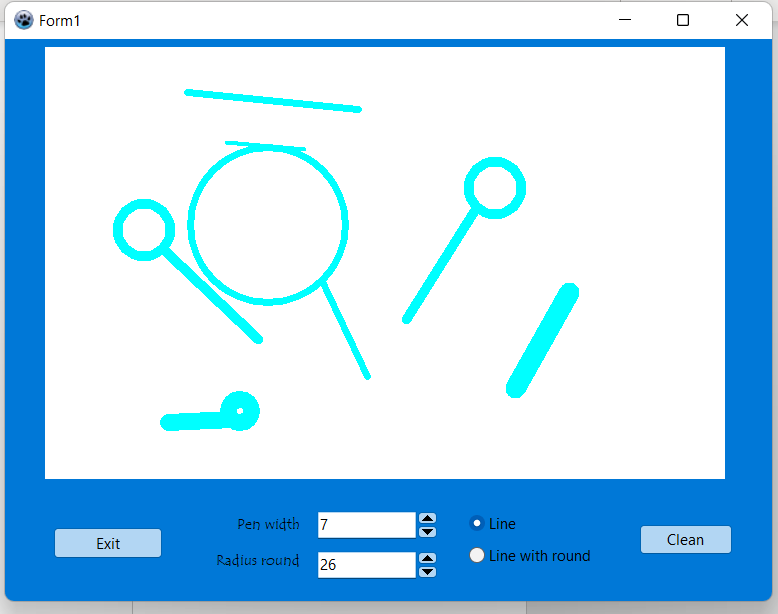
Application.Scaled:=True;

Application.Initialize;

Application.CreateForm(TForm1, Form1);

Application.Run;

end.

****

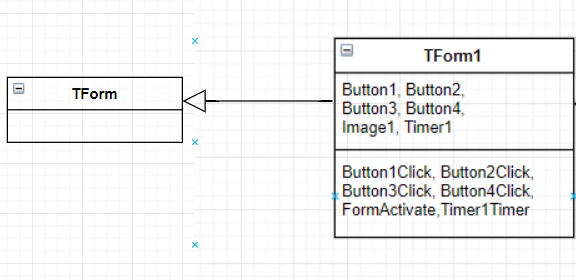
***Часть 1.2. Полиморфное наследование***

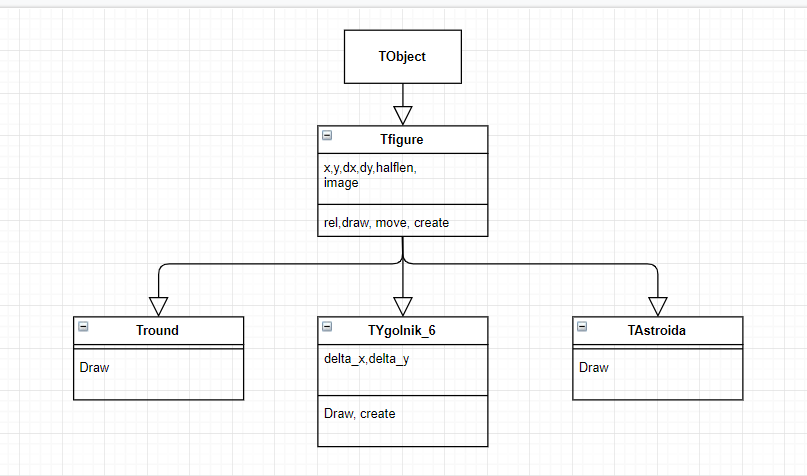
Разработать программу, содержащую описание трех графических объектов:

окружность, правильный шестиугольник, астроида.

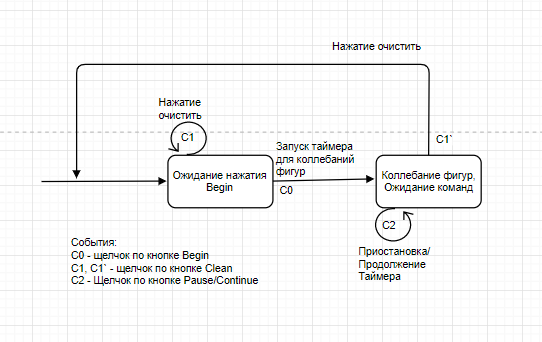
Реализуя механизм полиморфизма, привести объекты в одновременное колебательное движение вдоль горизонтальной оси с различными периодами колебаний.

**Иерархия классов:**

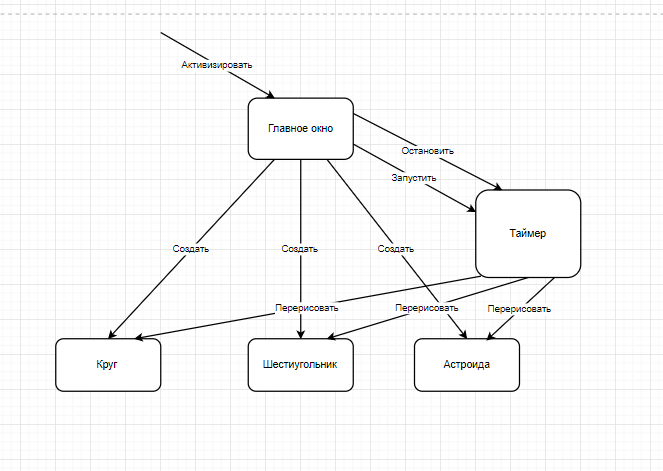




**Графическое состояние интерфейса:**



**Объектная декомпозиция:**



**Код программы:**

unit unit1\_part\_2\_true;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls,

figures, math;

type

{ TForm1 }

TForm1 = class(TForm)

Button1: TButton;

Button2: TButton;

Button3: TButton;

Button4: TButton;

Image1: TImage;

Timer1: TTimer;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

private

public

end;

var

Form1: TForm1;

implementation

{$R \*.lfm}

var

key:boolean = false;

pause:boolean = false;

smena1,smena2,smena3: boolean;

i:integer = 0;

t1,t2,t3:single;

Round:Tround;

Ygolnik\_6:Tygolnik\_6;

Astroida:Tastroida;

{ TForm1 }

procedure TForm1.FormActivate(Sender: TObject);

begin

button2.Enabled:=false;

timer1.Enabled:=false;

image1.Canvas.Brush.Color:=clhighlight;

image1.canvas.Rectangle(-2,-2,1000,1000);

end;

procedure TForm1.Timer1Timer(Sender: TObject);

begin

Round.move(t1);

Ygolnik\_6.move (t2);

// {For astroida}image1.canvas.Pen.color:=clhighlight;

// {For astroida}image1.canvas.Rectangle(570,300,720,100);

// {For astroida}image1.canvas.Pen.color:=claqua;

Astroida.move(t3);

if smena1 then begin

t1:= t1 - 1;

if ( trunc (t1) = -80 ) then smena1 := false;

end

else begin

t1 := t1 + 1;

if (trunc (t1) = 80 ) then smena1 := true;

end;

if smena2 then begin

t2:= t2 - 2;

if ( trunc (t2) = -120 ) then smena2 := false;

end

else begin

t2 := t2 + 2;

if (trunc (t2) = 120 ) then smena2 := true;

end;

if smena3 then begin

t3:= t3 - 8;

if ( trunc (t3) = -120 ) then smena3 := false;

end

else begin

t3 := t3 + 8;

if (trunc (t3) = 120 ) then smena3 := true;

end;

end;

procedure TForm1.Button3Click(Sender: TObject);

begin

close;

end;

procedure TForm1.Button1Click(Sender: TObject);

begin

if key then begin

Round.free;

ygolnik\_6.free;

Astroida.free;

key:=false;

end;

Round := TRound.create ( 120,200,50,Image1 );

Ygolnik\_6 := TYgolnik\_6.create( 410,200,50,Image1 );

Astroida := TAstroida.create( 570,200,50,Image1 );

Timer1.Enabled := True;

button1.Enabled:=false;

button2.Enabled:=true;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

if not(pause) then begin

timer1.Enabled:=false;

button2.Caption:='Continue';

pause:=true;

end

else begin

timer1.Enabled:=true;

button2.Caption:='Pause';

pause:=false;

end;

end;

procedure TForm1.Button4Click(Sender: TObject);

begin

timer1.Enabled:=false;

Image1.canvas.Clear;

button1.Enabled:=True;

button2.Enabled:=false;

button2.Caption:='Pause';

pause:=false;

key:=true;

end;

initialization

smena1:=false; t1 := 0.0;

smena2:=false; t2 := 0.0;

smena3:=false; t3 := 0.0;

finalization

Round.free;

ygolnik\_6.free;

Astroida.free;

end.

unit figures;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils,Forms,Controls,Graphics,Dialogs, ExtCtrls, math;

type

Tfigure=class

x,y,dx,dy,halflen:integer;

Image:timage;

constructor create (ix,iy,ihalflen:integer; iImage:TImage);

procedure move(t:single);

procedure draw ();virtual;abstract;

procedure rel (t:single);

end;

Tround = class (tfigure)

procedure draw();override;

end;

TYgolnik\_6 = class (tfigure)

delta\_x,delta\_y:integer;

constructor create (ix,iy,ihalflen:integer; iImage:TImage);

procedure draw();override;

end;

TAstroida = class (tfigure)

procedure draw (); override;

end;

implementation

constructor Tfigure.create(ix,iy,ihalflen:integer; iImage:TImage);

begin

x:=ix; y:=iy; halflen := ihalflen; Image := iImage;

end;

procedure tfigure.move(t:single);

begin

image.Canvas.Pen.Width:=5;

Image.canvas.pen.color := clhighlight;

Draw;

Rel(t);

Image.canvas.pen.color := clAqua;

Draw;

end;

procedure tfigure.Rel(t:single);

begin

dy := round(t);

{ dx := round(halflen \* cos(t) );

dy := round (halflen \* sin(t) ); }

end;

// Figures

procedure Tround.Draw();

begin

//Если рисуется не ровно, добавить радиус в поля

Image.canvas.ellipse(x+dx+halflen, y+dy+halflen, x+dx-halflen, y+dy-halflen);

end;

constructor Tygolnik\_6.create (ix,iy,ihalflen:integer; iImage:TImage);

begin

inherited create (ix,iy,ihalflen, iImage);

delta\_x := round ( ihalflen \* sqrt (2) / 2 );

delta\_y := round( ihalflen \* sqrt(3) / 2 );

end;

procedure Tygolnik\_6.Draw();

begin

// image.canvas.moveto (x+dx-100,y+dy-100);

// image.canvas.lineto(x+dx+100,y+dy+100);

// image.canvas.brush.color:=clblack;

// image.canvas.rectangle(350,300,800,100);

// image.canvas.brush.color:=clhighlight;

image.canvas.moveto (x+dx,y+dy);

image.canvas.lineto ( x + dx - delta\_x, y + dy - delta\_y );

image.canvas.lineto ( x + dx - delta\_x - halflen , y + dy - delta\_y );

image.canvas.lineto ( x + dx - 2 \* delta\_x - halflen, y + dy );

image.canvas.lineto ( x + dx - delta\_x - halflen, y + dy + delta\_y );

image.canvas.lineto ( x + dx - delta\_x, y + dy + delta\_y);

image.canvas.lineto ( x + dx, y + dy);

end;

procedure TAstroida.Draw ();

var i : single = 0.0;

first:boolean=true;

nx,ny:integer;

begin

while i<=6.28 do

begin

nx:=x+dx+trunc(halflen\*cos(i)\*cos(i)\*cos(i));

ny:=y+dy+trunc(halflen\*sin(i)\*sin(i)\*sin(i));

if first then begin

image.canvas.moveto(nx,ny);

first:=false;

end;

image.Canvas.lineto(nx,ny);

i:=i+0.01;

end;

end;

end.

program Part\_2\_true;

{$mode objfpc}{$H+}

uses

{$IFDEF UNIX}{$IFDEF UseCThreads}

cthreads,

{$ENDIF}{$ENDIF}

Interfaces, // this includes the LCL widgetset

Forms, unit1\_part\_2\_true

{ you can add units after this };

{$R \*.res}

begin

RequireDerivedFormResource:=True;

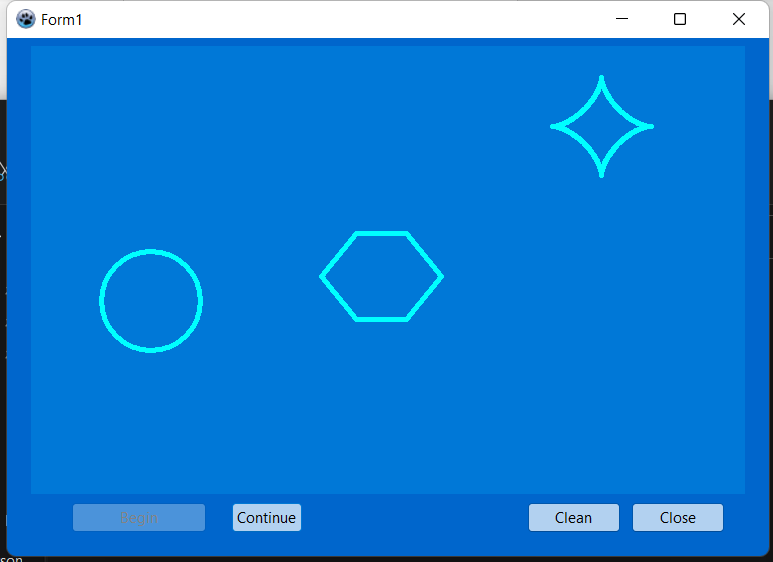
Application.Scaled:=True;

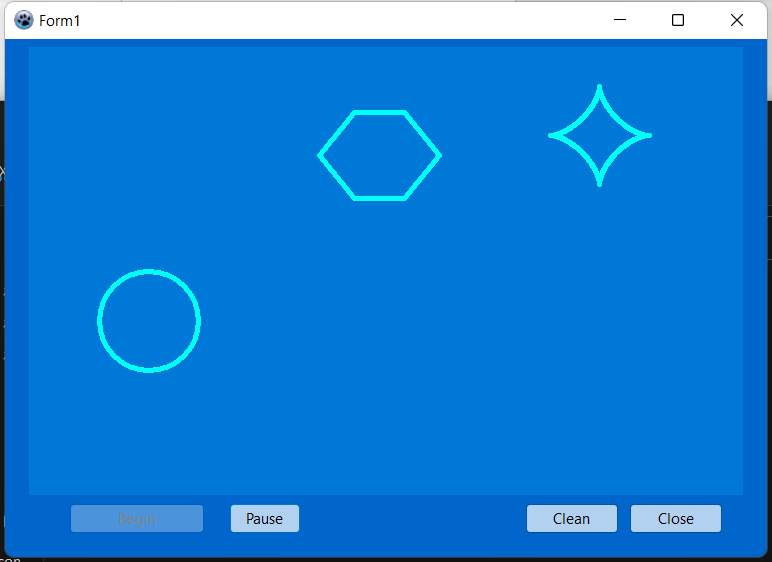
Application.Initialize;

Application.CreateForm(TForm1, Form1);

Application.Run;

end.





**Вывод:** Научился программировать на Object Pascal с использованием классов.