|  |  |
| --- | --- |
| 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 **09.03.03 Прикладная информатика**

**Отчет**

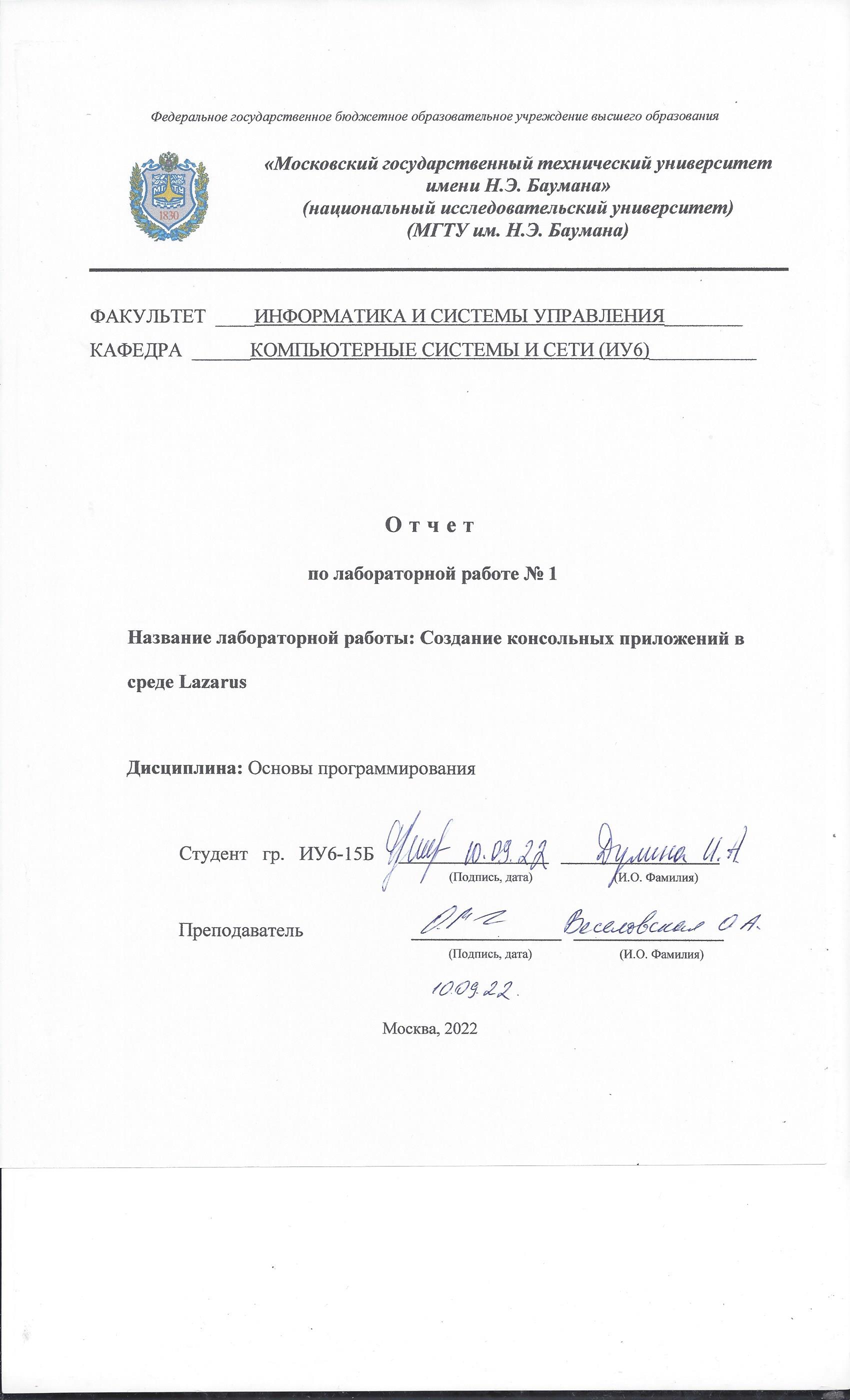
|  |  |
| --- | --- |
| **по домашней работе №** | 1 |

**Название:**

Программирование на Object Pascal с использованием классов

**Дисциплина:** Объектно-ориентированное программирование





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-25 Б |  |  |  |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  | 20.02.2023 |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

20.02.2023

Москва, 2023

**Вариант 8**

**Часть 1.1. Графический редактор**

**Задание:** Разработать иерархию классов. Поместить определение классов в отдельном модуле.

Класс, позволяющий рисовать окружность некоторого радиуса с центром в точке, определенной нажатием правой клавиши мыши.

Класс, позволяющий рисовать отрезок под заданным углом некоторой длины из точки, определенной нажатием левой клавиши мыши.

Длину и угол наклона отрезка, радиус окружности задавать с использованием интерфейсных элементов.

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

*Код модуля main:*

unit main;

{$mode objfpc}{$H+}

interface

uses

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

type

{ TForm1 }

TForm1 = class(TForm)

exitbutton: TButton;

redit: TEdit;

dedit: TEdit;

Image1: TImage;

rlabel: TLabel;

dlabel: TLabel;

procedure exitbuttonClick(Sender: TObject);

procedure formactivate(sender: tobject);

procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

end;

var

Form1: TForm1;

implementation

uses figure;

{$R \*.lfm}

{ TForm1 }

var f:byte=1;

procedure TForm1.formactivate(sender: tobject);

begin

image1.canvas.brush.color:=clwhite;

end;

procedure TForm1.exitbuttonClick(Sender: TObject);

begin close; end;

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

Shift: TShiftState; X, Y: Integer);

begin

if f=1 then begin

Image1.Canvas.FillRect(Rect(0,0,Width,Height));

f:=2; end;

if button=mbleft then

tmycircle.create(image1, x, y, strtoint(redit.text), strtoint(dedit.text));

if button=mbright then

tmyline.create(image1, x, y, strtoint(redit.text), strtoint(dedit.text));

end;

end.

*Код модуля figure:*

unit figure;

{$mode ObjFPC}{$H+}

interface

uses

Classes, SysUtils, variants, graphics, controls,

forms, dialogs, comctrls, stdctrls, extctrls;

type tmyfigure=class

public

x, y, radius, degr: word;

image:TImage;

constructor create(aimage:TImage; ax, ay, ar, ad: word);

procedure draw; virtual; abstract;

end;

tmycircle = class (tmyfigure)

public procedure draw; override; end;

tmyline = class (tmyfigure)

public procedure draw; override; end;

implementation

//var f:byte=1;

constructor tmyfigure.create(aimage:TImage; ax, ay, ar, ad: word);

begin

inherited create;

image:=aimage;

x:=ax;

y:=ay;

radius:=ar;

degr:=ad;

draw;

end;

procedure tmycircle.draw;

begin

image.canvas.pen.color:=clblue;

image.canvas.ellipse(x-radius, y-radius, x+radius, y+radius);

end;

procedure tmyline.draw;

var p:real;

begin

image.canvas.pen.color:=clblue;

image.canvas.MoveTo(x, y);

p:=degr\*pi/180;

image.canvas.lineto(x+round(radius\*cos(p)), y-round(radius\*sin(p)));

end;

end.

*Код основной программы:*

program proj1;

{$mode objfpc}{$H+}

uses

{$IFDEF UNIX}

cthreads,

{$ENDIF}

{$IFDEF HASAMIGA}

athreads,

{$ENDIF}

Interfaces, // this includes the LCL widgetset

Forms, main, figure

{ you can add units after this };

{$R \*.res}

begin

RequireDerivedFormResource:=True;

Application.Scaled:=True;

Application.Initialize;

Application.CreateForm(TForm1, Form1);

Application.Run;

end.

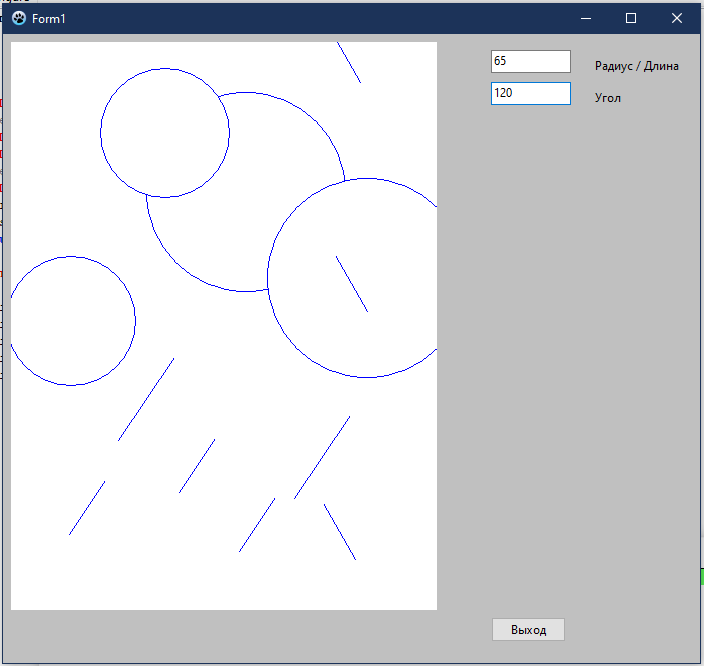


Рисунок 1 – работающая версия программы



Рисунок 2 – диаграмма классов



Рисунок 3 – диаграмма состояния пользовательского интерфейса



Рисунок 4 – объектная декомпозиция

**Вывод:** мы научились при помощи среды Lazarus рисовать фигуры определенного размера (длины), а также указывать угол, под которым будет проведена прямая.

**Часть 1.2. Полиморфное наследование**

**Задание:** Разработать программу, содержащую описание трех графических объектов:

квадрат, ромб, два одинаковых взаимно ортогональных ромба с общим центром.

Реализуя механизм полиморфизма, привести объекты в горизонтальное движение по экрану с различными скоростями с отражением от границ экрана.

В отчете привести диаграмму используемых классов VCL и разработанных классов, граф состояний пользовательского интерфейса и объектную декомпозицию.

*Код модуля main:*

unit main;

{$mode objfpc}{$H+}

interface

uses

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

type

{ TForm1 }

TForm1 = class(TForm)

beginbutton: TButton;

exitbutton: TButton;

Image1: TImage;

Timer1: TTimer;

procedure FormActivate(Sender: TObject);

procedure beginbuttonClick(Sender: TObject);

procedure exitbuttonClick(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

end;

var

Form1: TForm1;

implementation

uses Figure;

{$R \*.lfm}

Var

t:single=0.0;

R:Tromb;

S:TSquare;

RR:Trromb;

{ TForm1 }

procedure TForm1.FormActivate(Sender: TObject);

begin

Image1.Canvas.Brush.Color:=clWhite;

Image1.Canvas.FillRect(Rect(0,0,Width,Height));

end;

procedure TForm1.Timer1Timer(Sender: TObject);

begin

R.Move(5\*t, 5);

S.Move(3\*t, 3);

RR.Move(10\*t, 10);

t:=t+0.5;

end;

procedure TForm1.BeginButtonClick(Sender: TObject);

begin

Image1.Canvas.FillRect(Rect(0,0,Width,Height));

S:=TSquare.Create(90,60,50,Image1);

R:=Tromb.Create(200,197,40,Image1);

RR:=Trromb.Create(100,350,100,Image1);

Timer1.Enabled:=true;

end;

procedure TForm1.ExitButtonClick(Sender: TObject);

begin

Close;

end;

initialization

finalization

R.Free;

S.Free;

RR.Free;

end.

*Код модуля figure:*

unit figure;

{$mode ObjFPC}{$H+}

interface

uses

Classes, SysUtils, graphics,ExtCtrls;

Type

TFigure=Class

private

x,y,halflen, dx:integer; t2, tt:single; f, k:smallint;

Image:TImage;

procedure Draw;virtual;abstract;

procedure Rel(t:real);virtual;

public

constructor Create(ax,ay,ah:integer;aImage:TImage);

procedure Move(t:single; n:byte);

end;

Tromb=Class(TFigure)

private procedure Draw;override;

end;

TSquare=Class(TFigure)

private procedure Draw;override;

end;

TRRomb=Class(TFigure)

private procedure Draw;override;

end;

implementation

Constructor TFigure.Create(ax,ay,ah:integer;aImage:TImage);

Begin

inherited Create;

x:=ax; y:=ay; halflen:=ah; Image:=aImage; f:=1; k:=1; tt:=0;

End;

Procedure TFigure.Rel(t:real);

Begin

dx:=round(10\*t);

End;

Procedure TFigure.Move(t:single; n:byte);

Begin

if (x+dx+halflen>image.width-5\*n) and (f=1) //0.5\*10\*3(n)

or (x+dx-halflen<5\*n) and (f=-1) then

begin

if k=1 then begin t2:=t-t2; k:=2; end;

if f=1 then tt:=t2;

f:=-f;

end;

if (f=1) then tt:=tt+0.5\*n

else begin tt:=tt-0.5\*n; end;

Image.Canvas.Pen.Color:=clWhite;

Draw;

Image.Canvas.Pen.Color:=clBlack;

Rel(tt);

Draw;

End;

Procedure Tromb.Draw;

Begin

image.canvas.MoveTo(x+dx+halflen, y);

Image.Canvas.LineTo(x+dx,y+2\*halflen);

Image.Canvas.LineTo(x+dx-halflen,y);

Image.Canvas.LineTo(x+dx,y-2\*halflen);

Image.Canvas.LineTo(x+dx+halflen,y);

End;

Procedure TSquare.Draw;

Begin

image.canvas.MoveTo(x+dx+halflen, y+halflen);

Image.Canvas.LineTo(x+dx-halflen,y+halflen);

Image.Canvas.LineTo(x+dx-halflen,y-halflen);

Image.Canvas.LineTo(x+dx+halflen,y-halflen);

Image.Canvas.LineTo(x+dx+halflen,y+halflen);

End;

Procedure Trromb.Draw;

var a, b:byte;

Begin

a:=halflen div 3;

image.canvas.MoveTo(x+dx+a, y);

Image.Canvas.LineTo(x+dx,y+2\*a);

Image.Canvas.LineTo(x+dx-a,y);

Image.Canvas.LineTo(x+dx,y-2\*a);

Image.Canvas.LineTo(x+dx+a,y);

b:=halflen div 2;

image.canvas.MoveTo(x+dx+halflen, y);

Image.Canvas.LineTo(x+dx,y+b);

Image.Canvas.LineTo(x+dx-halflen,y);

Image.Canvas.LineTo(x+dx,y-b);

Image.Canvas.LineTo(x+dx+halflen,y);

End;

end.

*Код основной программы:*

program proj;

{$mode objfpc}{$H+}

uses

{$IFDEF UNIX}

cthreads,

{$ENDIF}

{$IFDEF HASAMIGA}

athreads,

{$ENDIF}

Interfaces, // this includes the LCL widgetset

Forms, main, figure

{ you can add units after this };

{$R \*.res}

begin

RequireDerivedFormResource:=True;

Application.Scaled:=True;

Application.Initialize;

Application.CreateForm(TForm1, Form1);

Application.Run;

end.

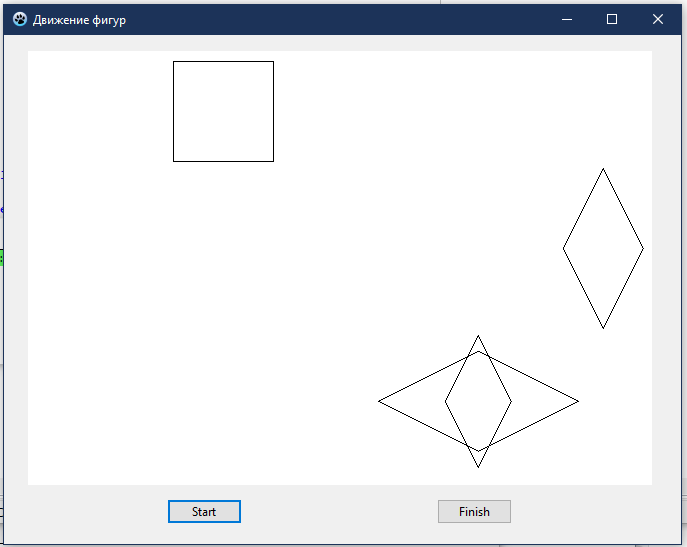


Рисунок 1 – работающая версия программы



Рисунок 2 – диаграмма классов



Рисунок 3 – диаграмма состояния пользовательского интерфейса



Рисунок 4 – объектная декомпозиция

**Вывод:** мы научились создавать при помощи среды Lazarus несколько движущихся по определенной траектории фигур, отличающихся друг от друга скоростью движения.