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**

О Т Ч Е Т (В А Р И А Н Т 1 2)

по домашней работе № 1

Название: Программирование на Object Pascal с использованием классов

Дисциплина: Объектно-ориентированное программирование

Студент	ИУ6-23Б	01.03.2023	В.К. Залыгин
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			А.М.Минитаева
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

Цель работы

Научиться практически применять полиморфное наследование, разрабатывать приложения с графическим интерфейсом на основе библиотеки LCL на языке Object Pascal.

Часть 1

Задание

Разработать иерархию классов. Поместить определение классов в отдельном модуле.

Класс, позволяющий рисовать линию от точки, определенной нажатием правой клавиши мыши, до точки ее отпускания.

Класс, позволяющий рисовать прямоугольник с диагональю от точки, определенной нажатием левой клавиши мыши, до точки ее отпускания.

Цвет фигур задавать с использованием интерфейсных элементов.

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

Выполнение

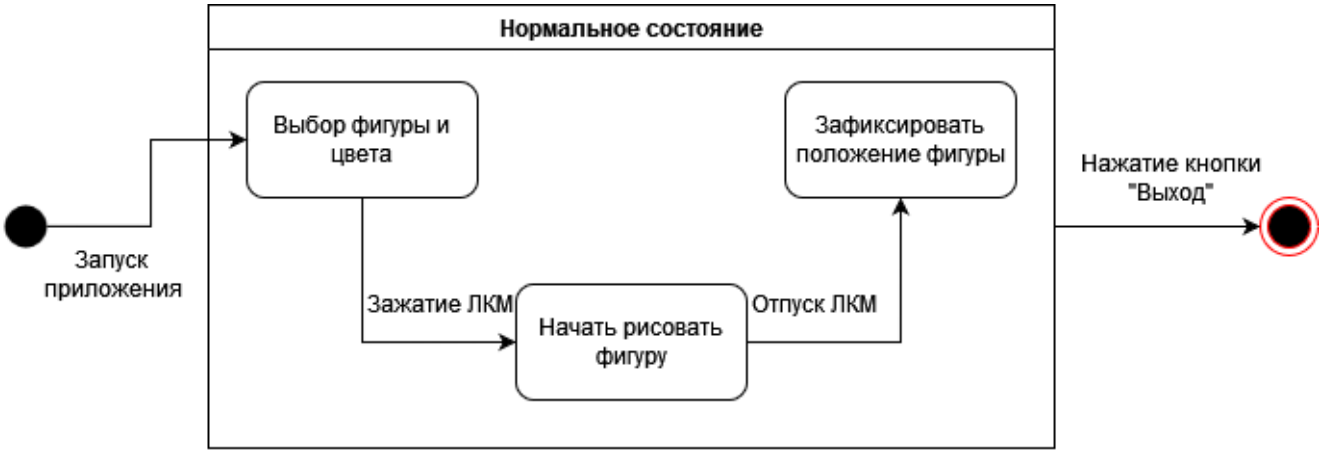


Рисунок 1 - граф состояний

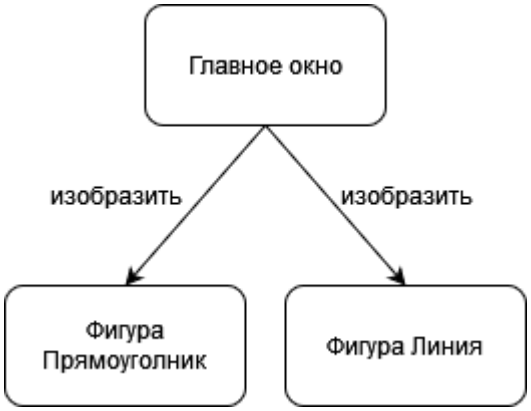


Рисунок 2 - объектная декомпозиция

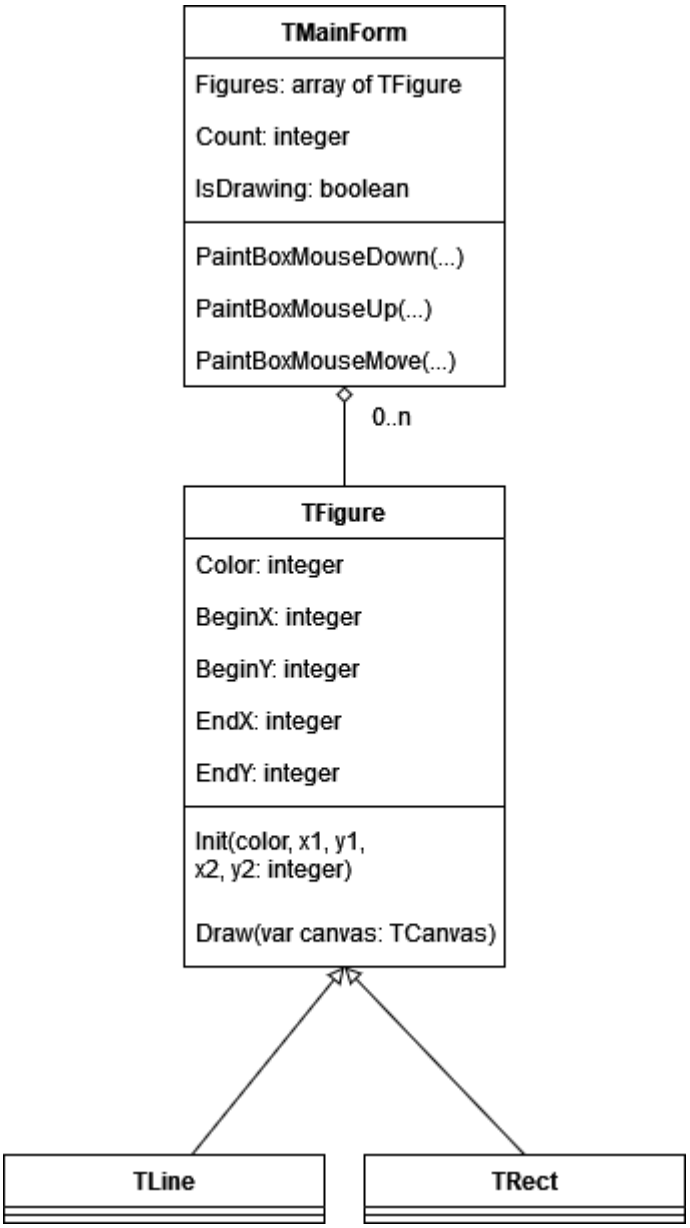


Рисунок 3 - диаграмма классов

Текст программы

Код модуля “Main”

```
unit main;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, Menus,
  StdCtrls, ActnList, Lines, Rectangles, Figure;

type

  { TMainForm }

  TMainForm = class(TForm)
    PaintBox: TPaintBox;
    LineButton: TRadioButton;
    BlueButton: TRadioButton;
    GreenButton: TRadioButton;
    RedButton: TRadioButton;
    RadioGroup2: TRadioGroup;
    RectButton: TRadioButton;
    RadioGroup1: TRadioGroup;
    procedure FormCreate(Sender: TObject);
    procedure PaintBoxMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: integer);
    procedure PaintBoxMouseMove(Sender: TObject; Shift: TShiftState; X, Y: integer);
    procedure PaintBoxMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: integer);
  private
    Figures: array of TFigure;
    Count: integer;
    IsDrawing: boolean;
    function GetColor: integer;
    function GetFigure: integer;
  public

  end;

var
  MainForm: TMainForm;

implementation

{$R *.lfm}

procedure ClearCanvas(canvas: TCanvas);
begin
  canvas.Brush.Color := $FFFFFF;
  canvas.FillRect(0, 0, 400, 400);
end;
```

Рисунок 4 - код модуля Main

```
procedure DrawFigures(canvas: TCanvas; figures: array of TFigure);
var
  i: integer;
begin
  ClearCanvas(canvas);
  for i := 0 to high(figures) do
    figures[i].Draw(canvas);
end;

{ TMainForm }

function TMainForm.GetColor: integer;
begin
  if TRadioButton(RadioGroup2.Controls[0]).Checked then
    Result := $FF0000
  else if TRadioButton(RadioGroup2.Controls[1]).Checked then
    Result := $00FF00
  else if TRadioButton(RadioGroup2.Controls[2]).Checked then
    Result := $0000FF
  else
    Result := $000000;
end;

function TMainForm.GetFigure: integer;
begin
  if TRadioButton(RadioGroup1.Controls[0]).Checked then
    Result := 0
  else
    Result := 1;
end;

procedure TMainForm.FormCreate(Sender: TObject);
begin
  ClearCanvas(PaintBox.Canvas);
end;
```

Рисунок 5 - код модуля Main

```
.  procedure TMainForm.PaintBoxMouseDown(Sender: TObject; Button: TMouseButton;
90   Shift: TShiftState; X, Y: integer);
.  var
.    line: TLine;
.    rect: TRect;
.  begin
95   IsDrawing := True;
.    Count += 1;
.    SetLength(Figures, Count);
.    case GetFigure of
.      0: begin
100      line.Init(GetColor, x, y, x, y);
.        Figures[Count - 1] := line;
.      end;
.      1: begin
.        rect.Init(GetColor, x, y, x, y);
105      Figures[Count - 1] := rect;
.      end;
.    end;
.    DrawFigures(PaintBox.canvas, Figures);
.  end;
110
.  procedure TMainForm.PaintBoxMouseMove(Sender: TObject; Shift: TShiftState;
.    X, Y: integer);
.  begin
.    if IsDrawing then
115  begin
.      Figures[Count - 1].EndX := x;
.      Figures[Count - 1].EndY := y;
.      DrawFigures(PaintBox.canvas, Figures);
.    end;
120  end;
.
.  procedure TMainForm.PaintBoxMouseUp(Sender: TObject; Button: TMouseButton;
.    Shift: TShiftState; X, Y: integer);
.  begin
125  IsDrawing := False;
.  end;
.
128  end.
```

Рисунок 6 - код модуля Main

```
unit figure;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils, Graphics;

type
  TFigure = object
    constructor Init(color, beginX, beginY, endX, endY: integer);
    procedure Draw(var canvas: TCanvas); virtual;

  public
    Color, BeginX, BeginY, EndX, EndY: integer;
  end;

implementation

constructor TFigure.Init(color, beginX, beginY, endX, endY: integer);
begin
  self.Color := color;
  self.BeginX := beginX;
  self.BeginY := BeginY;
  self.EndX := endX;
  self.EndY := endY;
end;

procedure TFigure.Draw(var canvas: TCanvas);
begin
  canvas.Brush.Color := clblue;
  canvas.Pen.Width := 4;
  Canvas.Pen.Color := clBlue;
  Canvas.Pen.Style := psSolid;
  Canvas.MoveTo(10, 10);
  Canvas.LineTo(50, 50);
end;

end.
```

Рисунок 7 - код модуля Figure

```
unit Lines;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils, Figure, Graphics;

type
  TLine = object(TFigure)
    constructor Init(c, x1, y1, x2, y2: integer);
    procedure Draw(var canvas: TCanvas); virtual;
  end;

implementation

constructor TLine.Init(c, x1, y1, x2, y2: integer);
begin
  inherited Init(c, x1, y1, x2, y2);
end;

procedure TLine.Draw(var canvas: TCanvas);
var
  x1, y1, x2, y2: integer;
begin
  x1 := self.BeginX;
  y1 := self.BeginY;
  x2 := self.EndX;
  y2 := self.EndY;

  canvas.Brush.Color:=self.Color;
  canvas.Pen.Color:=self.Color;
  canvas.Pen.Width:=4;
  canvas.Line(x1, y1, x2, y2);
end;

end.
```

Рисунок 8 - код модуля Lines

```
unit rectangles;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils, Figure, Graphics;

type TRect = object(TFigure)
  constructor Init(c, x1, y1, x2, y2: integer);
  procedure Draw(var canvas: TCanvas); virtual;
  end;

implementation

constructor TRect.Init(c, x1, y1, x2, y2: integer);
begin
  inherited Init(c, x1, y1, x2, y2)
end;

procedure TRect.Draw(var canvas: TCanvas);
var x1, y1, x2, y2, tmp: integer;
begin
  x1 := self.BeginX;
  y1 := self.BeginY;
  x2 := self.EndX;
  y2 := self.EndY;
  if x1 < x2 then begin
    tmp := x1;
    x1 := x2;
    x2 := tmp;
  end;
  if y1 < y2 then begin
    tmp := y1;
    y1 := y2;
    y2 := tmp;
  end;
  canvas.Brush.Color:=self.Color;
  canvas.Pen.Color:=self.Color;
  canvas.Pen.Width:=4;
  canvas.Rectangle(x1, y1, x2, y2);
end;

end.
```

Рисунок 9 - код модуля Rectangles

Тестовые данные

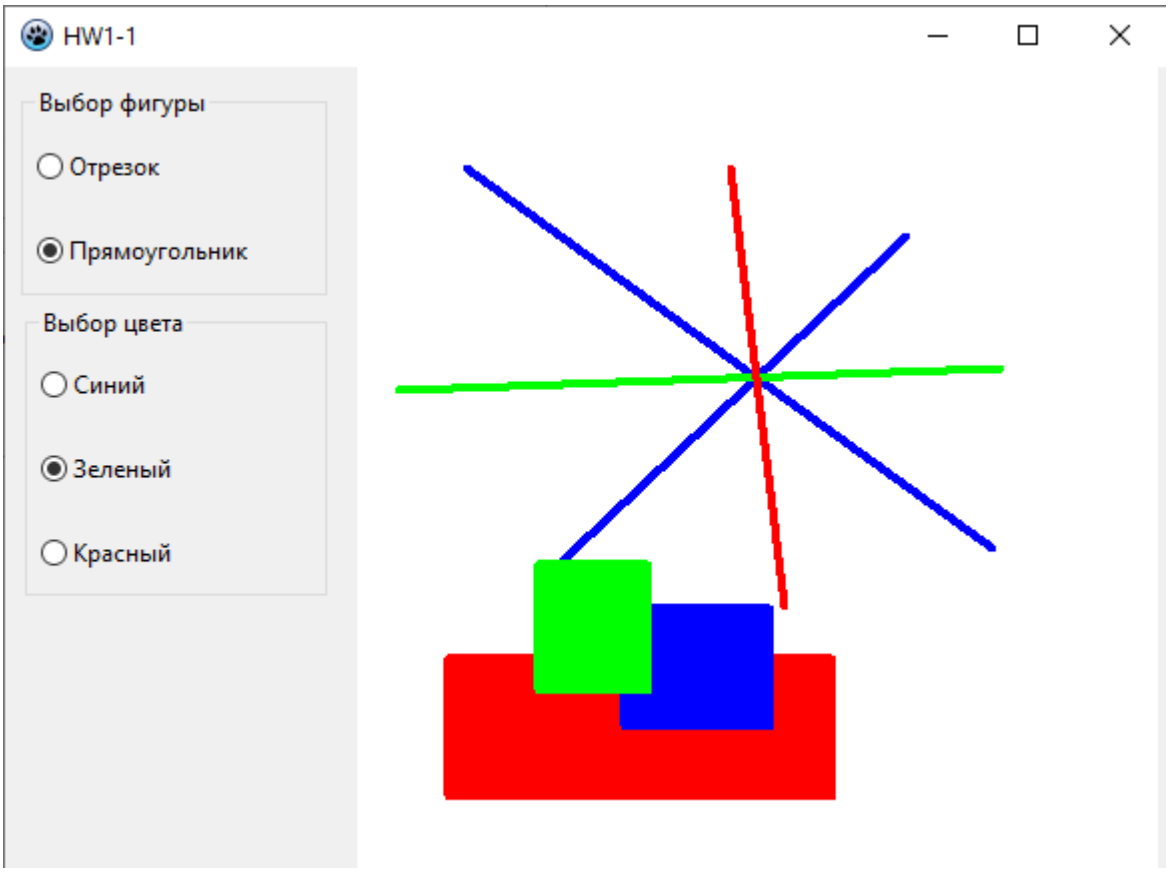
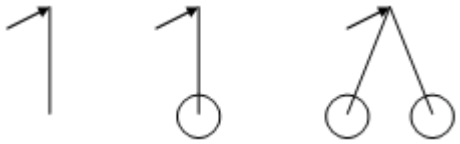


Рисунок 10 - пример работы программы

Часть 2

Задание

Разработать программу, содержащую описание трех графических объектов:



Реализуя механизм полиморфизма, привести объекты в одновременное колебательное движение вокруг указанных точек с разными амплитудами и периодами колебаний.

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

Выполнение

Рисунок 11 - граф состояний

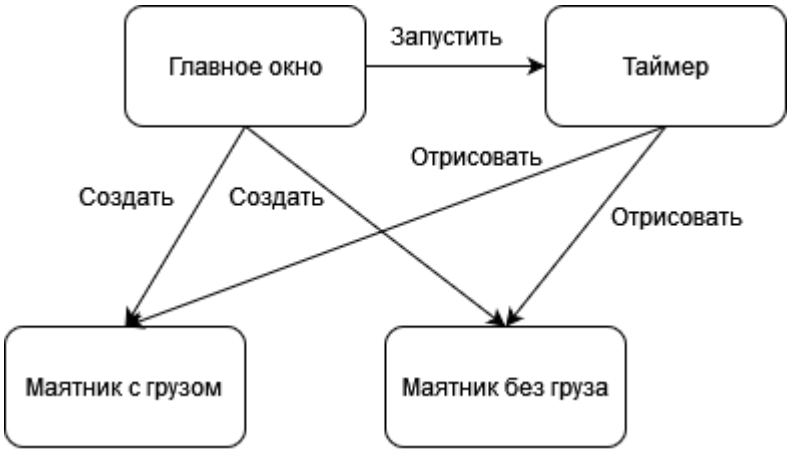


Рисунок 12 - объектная декомпозиция

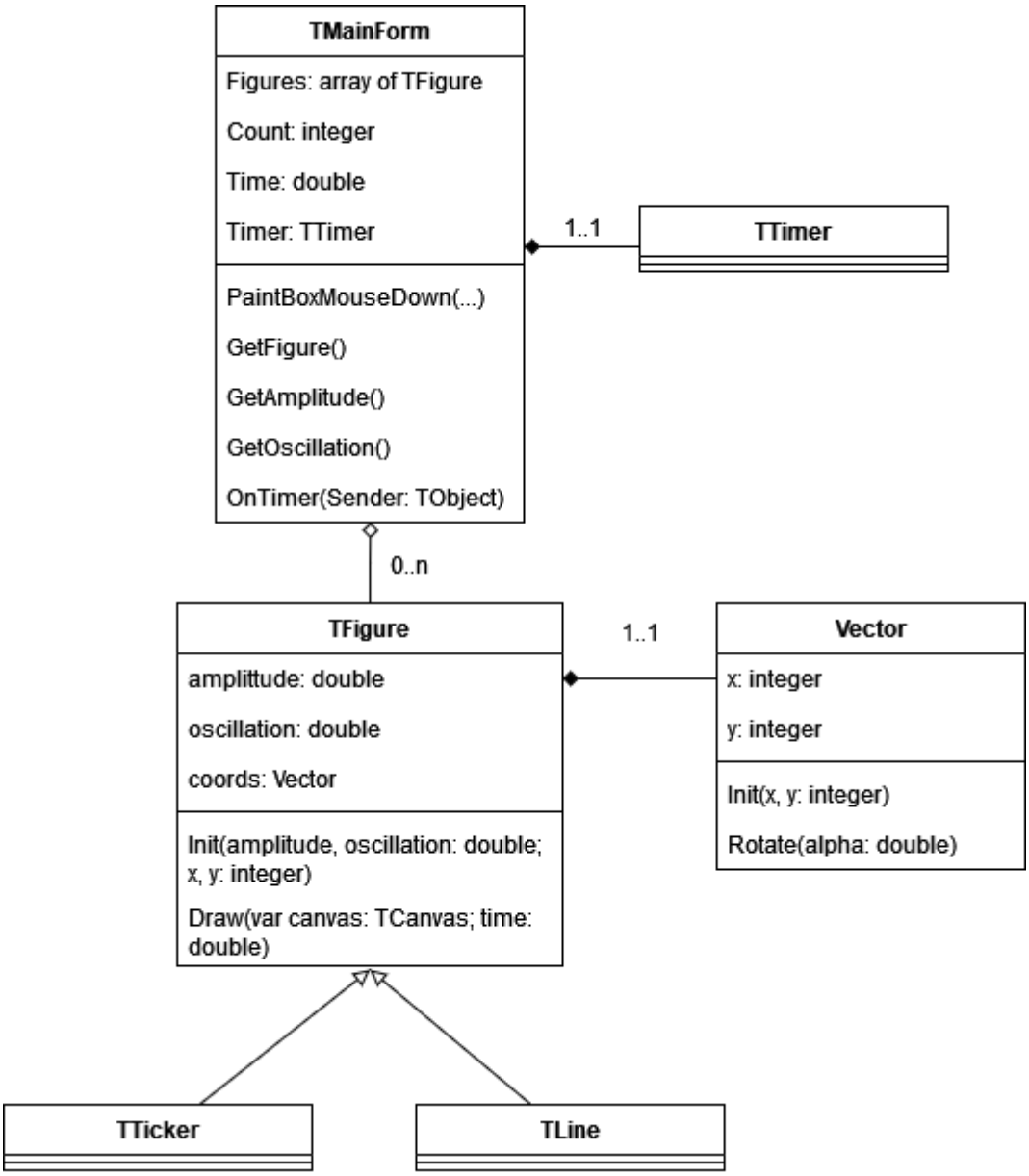


Рисунок 13 - диаграмма классов

Код программы

`

```
unit main;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, Menus,
  StdCtrls, ActnList, Lines, Tickers, Figure;

type

  { TMainForm }

  TMainForm = class(TForm)
    AmplitudeEdit: TEdit;
    OscillationEdit: TEdit;
    PaintBox: TPaintBox;
    LineButton: TRadioButton;
    TickerButton: TRadioButton;
    RadioGroup1: TRadioGroup;
    Timer: TTimer;
    procedure FormCreate(Sender: TObject);
    procedure OnTimer(Sender: TObject);
    procedure PaintBoxMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: integer);
  private
    Figures: array of TFigure;
    Count: integer;
    Time: double;
    function GetFigure: integer;
    function GetAmplitude: double;
    function GetOscillation: double;
  public

  end;

var
  MainForm: TMainForm;
const
  Delta = 50;

implementation

{$R *.lfm}

procedure ClearCanvas(canvas: TCanvas);
begin
  canvas.Brush.Color := $FFFFFF;
  canvas.FillRect(0, 0, 400, 400);
end;
```

Рисунок 14 - код модуля main

```
procedure DrawFigures(canvas: TCanvas; figures: array of TFigure; time: double);
var
  i: integer;
begin
  ClearCanvas(canvas);
  for i := 0 to high(figures) do
    figures[i].Draw(canvas, time);
end;

{ TMainForm }

function TMainForm.GetFigure: integer;
begin
  if TRadioButton(RadioGroup1.Controls[0]).Checked then
    Result := 0
  else
    Result := 1;
end;

function TMainForm.GetAmplitude: double;
begin
  if AmplitudeEdit.Text <> '' then
    Result := StrToFloat(AmplitudeEdit.Text)
  else
    Result := 0.0;
end;

function TMainForm.GetOscillation: double;
begin
  if OscillationEdit.Text <> '' then
    Result := StrToFloat(OscillationEdit.Text)
  else
    Result := 0.0;
end;

procedure TMainForm.FormCreate(Sender: TObject);
begin
  ClearCanvas(PaintBox.Canvas);
  Timer.Interval:=Delta;
end;

procedure TMainForm.OnTimer(Sender: TObject);
begin
  Time += Delta/1000;
  DrawFigures(PaintBox.Canvas, Figures, Time);
end;
```

Рисунок 15 - код модуля main

```
100 procedure TMainForm.PaintBoxMouseDown(Sender: TObject; Button: TMouseButton;
    .   Shift: TShiftState; X, Y: integer);
    . var
    .   line: TLine;
    .   ticker: TTicker;
105 begin
    .   self.Timer.Enabled:=true;
    .   Count += 1;
    .   SetLength(Figures, Count);
    .   case GetFigure of
110     0: begin
    .       line.Init(GetAmplitude, GetOscillation, x, y);
    .       Figures[Count - 1] := line;
    .     end;
    .     1: begin
115       ticker.Init(GetAmplitude, GetOscillation, x, y);
    .       Figures[Count - 1] := ticker;
    .     end;
    .   end;
    . end;
120
121 end.
```

Рисунок 17 - код модуля main

```
  1 unit Figure;
    .
    . {$mode ObjFPC}{$H+}
    .
  5 interface
    .
    . uses
    .   Classes, SysUtils, Graphics, Vec;
    .
 10 type
    .   TFigure = object
    .   public
    .     constructor Init(amplitude, oscillation: double; x, y: integer);
    .     procedure Draw(var canvas: TCanvas; time: double); virtual; abstract;
 15   public
    .     coords: vector;
    .     size: vector;
    .     amplitude, oscillation: double;
    .   end;
 20
    . implementation
    .
    . constructor TFigure.Init(amplitude, oscillation: double; x, y: integer);
    . begin
25    self.coords.Init(x, y);
    .   self.size.Init(100, 0);
    .   self.amplitude := amplitude;
    .   self.oscillation := oscillation;
    . end;
 30
 31 end.
```

Рисунок 16 - код модуля Figure

```
unit vec;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils;

type
  vector = object
    procedure Init(x, y: integer);
    function Rotate(alpha: double): Vector;
  public
    x, y: integer;
  end;

implementation

procedure Vector.Init(x, y: integer);
begin
  self.x := x;
  self.y := y;
end;

function Vector.Rotate(alpha: double): Vector;
var xl, yl: integer;
    res: Vector;
begin
  xl := trunc(+cos(alpha)*x + sin(alpha)*y);
  yl := trunc(-sin(alpha)*x + cos(alpha)*y);
  res.Init(xl, yl);
  Result := res;
end;

end.
```

Рисунок 18 - код модуля Vec

```
unit Tickers;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils, Figure, Graphics, Vec;

type
  TTicker = object(TFigure)
    constructor Init(amplitudeT, oscillationT: double; xT, yT: integer);
    procedure Draw(var canvas: TCanvas; time: double); virtual;
  end;

implementation

constructor TTicker.Init(amplitudeT, oscillationT: double; xT, yT: integer);
begin
  inherited Init(amplitudeT, oscillationT, xT, yT);
end;

procedure TTicker.Draw(var canvas: TCanvas; time: double);
var
  alpha: double;
  shift: Vector;
begin
  alpha := 2 * Pi * (time / oscillation);
  alpha := sin(alpha) * amplitude / 180 * Pi;
  shift := size.Rotate(alpha);
  shift := shift.Rotate(-Pi / 2);

  canvas.Brush.Color := $FFFFFF;
  canvas.Pen.Color := $000000;
  canvas.Pen.Width := 2;
  canvas.Ellipse(coords.x + shift.x - 10, coords.y + shift.y - 10,
    coords.x + shift.x + 10, coords.y + shift.y + 10);
  canvas.Line(coords.x, coords.y, coords.x + shift.x, coords.y + shift.y);
end;

end.
```

Рисунок 19 - код модуля Tickers

```
unit Lines;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils, Figure, Graphics, Crt, Vec;

type
  TLine = object(TFigure)
    constructor Init(amplitudeT, oscillationT: double; xT, yT: integer);
    procedure Draw(var canvas: TCanvas; time: double); virtual;
  end;

implementation

constructor TLine.Init(amplitudeT, oscillationT: double; xT, yT: integer);
begin
  inherited Init(amplitudeT, oscillationT, xT, yT);
end;

procedure TLine.Draw(var canvas: TCanvas; time: double);
var
  alpha: double;
  shift: Vector;
begin
  alpha := 2 * Pi * (time / oscillation);
  alpha := sin(alpha) * amplitude / 180 * Pi;
  shift := size.Rotate(alpha);
  shift := shift.Rotate(-Pi / 2);

  canvas.Brush.Color := $000000;
  canvas.Pen.Color := $000000;
  canvas.Pen.Width := 2;
  canvas.Line(coords.x, coords.y, coords.x + shift.x, coords.y + shift.y);
end;

end.
```

Рисунок 20 - код модуля Lines

Пример работы

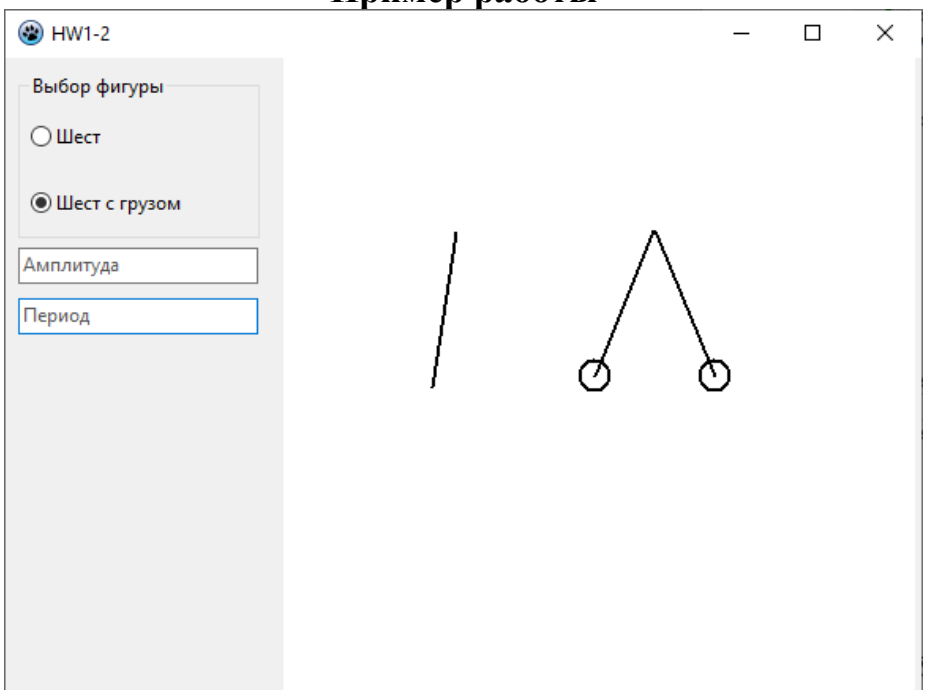


Рисунок 21 - пример работы

Вывод

В результате были изучены принципы применения полиморфного наследования, разработки приложений с графическим интерфейсов на основе библиотеки LCL на языке Object Pascal.