МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Казанский национальный исследовательский технический университет

им. А.Н. Туполева – КАИ»

Институт компьютерных технологий и защиты информации

Отделение СПО ИКТЗИ (Колледж информационных технологий)

ЛАБОРАТОРНАЯ РАБОТА №8

по дисциплине

Основы алгоритмизации и программирования

Тема: «Создание и использование библиотеки классов для графических примитивов»

Работу выполнил

Студент гр.4238

Бусов В.Р.

Принял

Преподаватель Шмидт И.Р.

Казань 2024

**ВАРИАНТ 4**

**Цель работы**

Приобрести умения и практические навыки для разработки приложения по созданию иерархии классов графических примитивов.

**Задание на лабораторную работу**

Требуется создать небольшую иерархию классов, описывающих основные графические примитивы: эллипс, окружность, прямоугольник, квадрат.

**Результат выполнения работы**

Для начала нарисуем все предлагаемые фигуры (Рисунок 1).

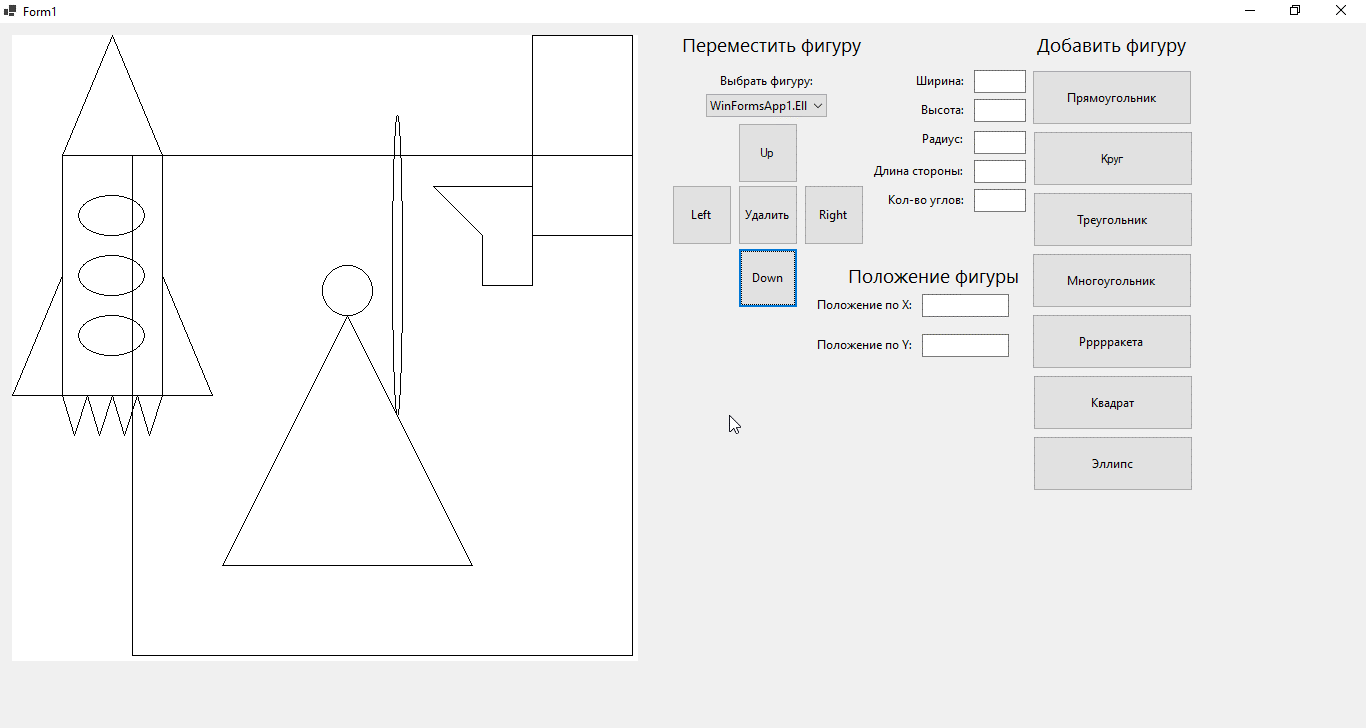


Рисунок 1 - Все фигуры

Затем удалим любую фигуру, например прямоугольник (Рисунок 2).

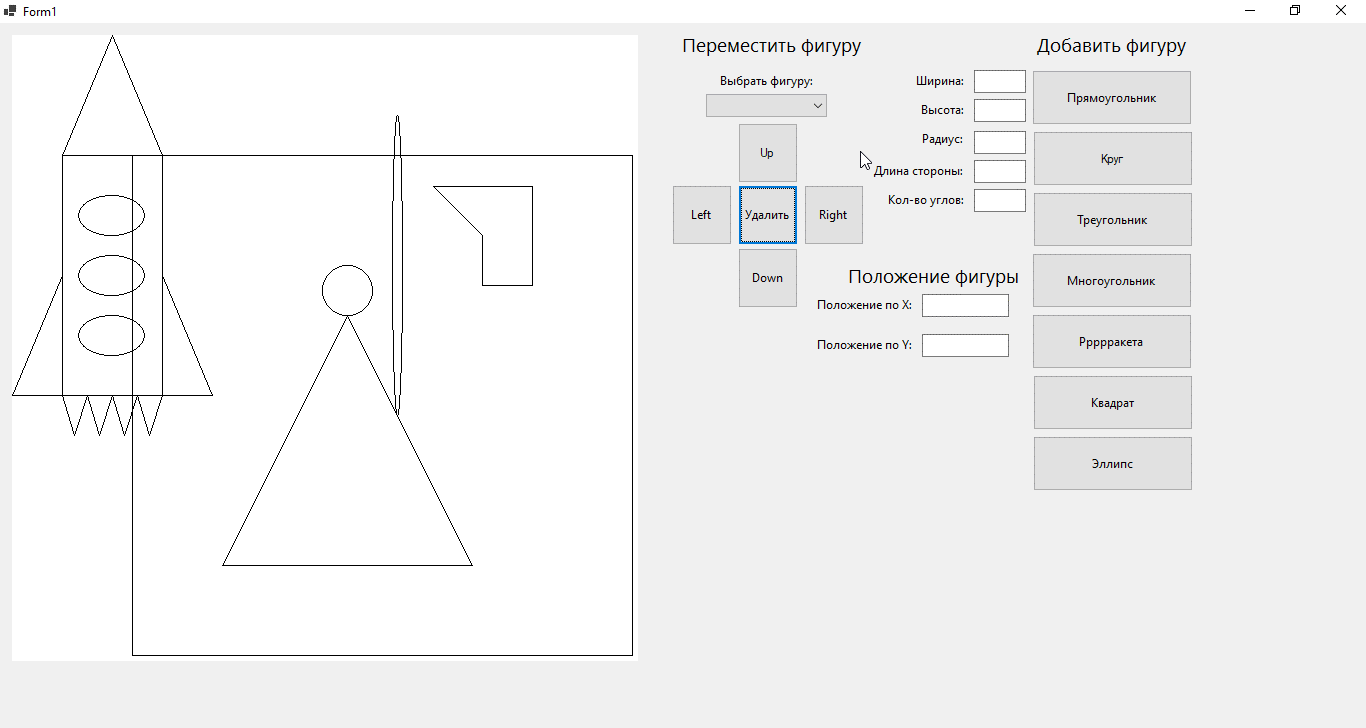


Рисунок 2 - Удаление прямоугольника

Произведём очистку холста от всех нарисованных фигур (Рисунок 3).

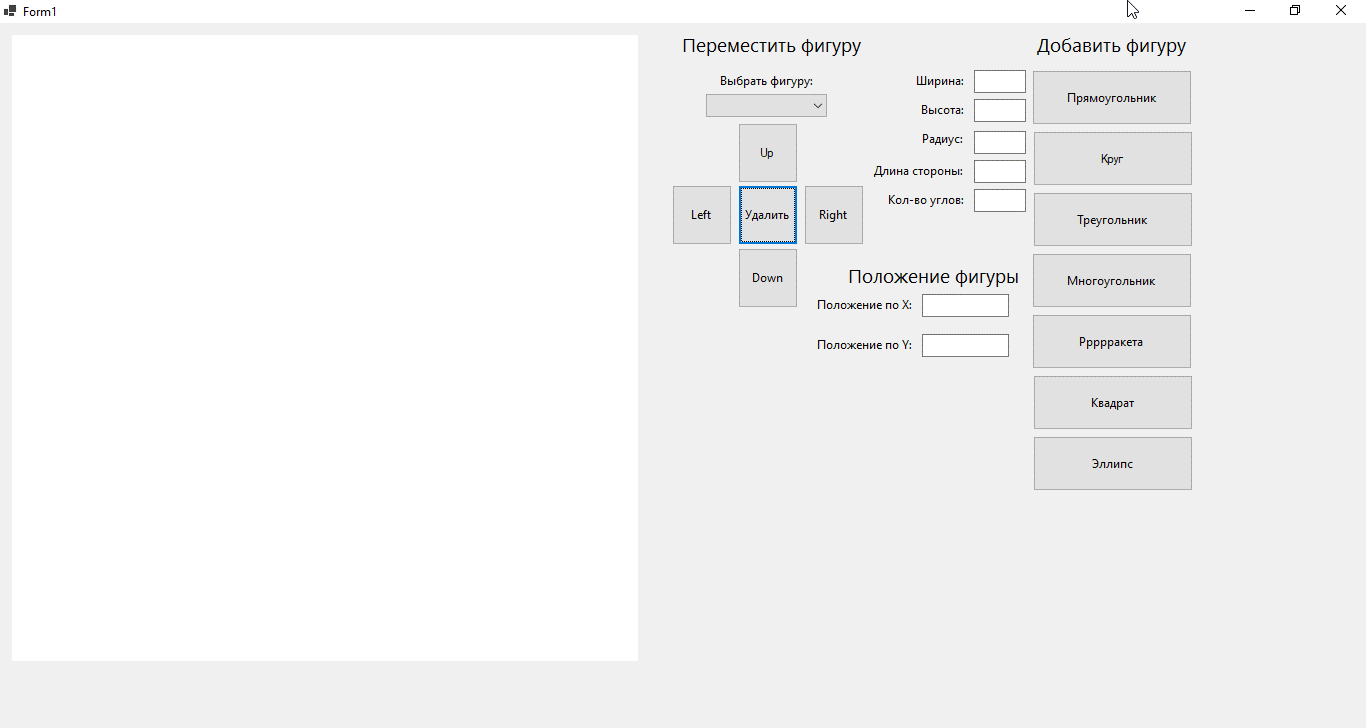


Рисунок 3 - Очистка от всех фигур

Снова нарисуем пару фигур и покажем, как они будут выглядеть в списке фигур (Рисунок 4).

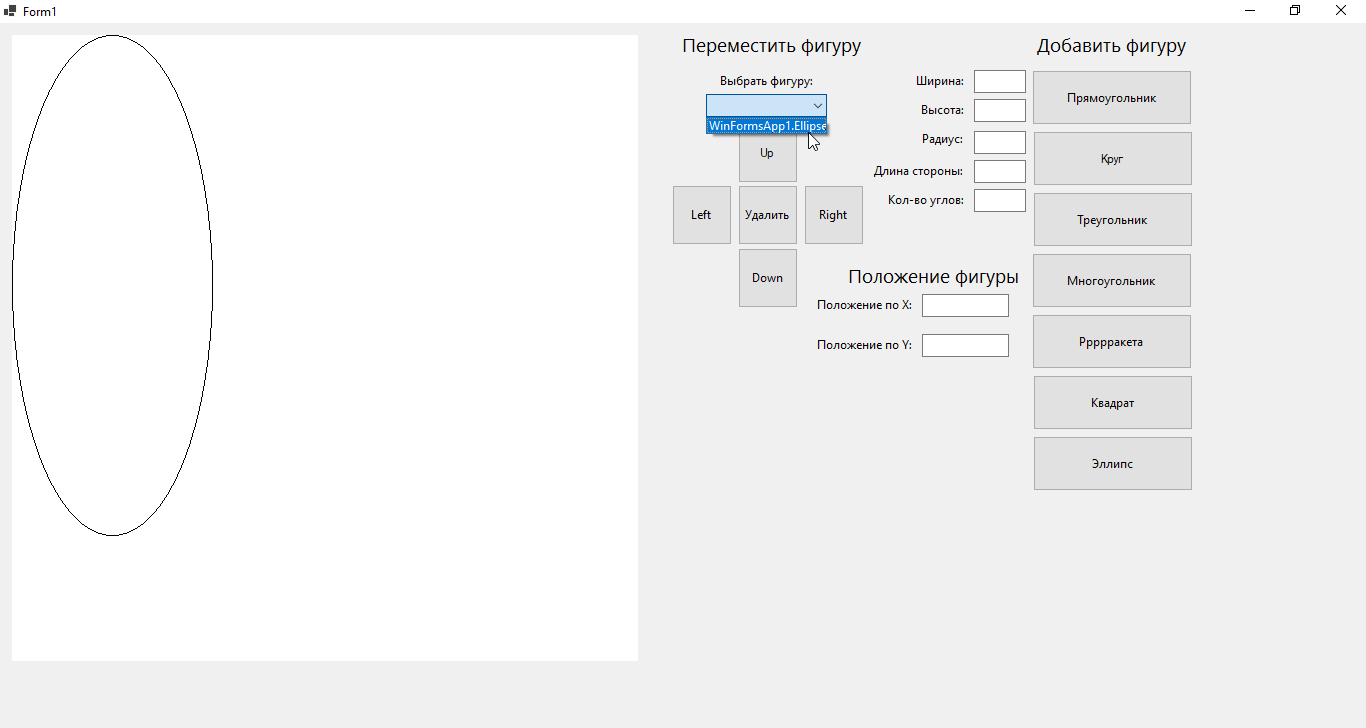


Рисунок 4 - Список всех нарисованных фигур

Далее переместим выбранную фигуру (Рисунок 5).

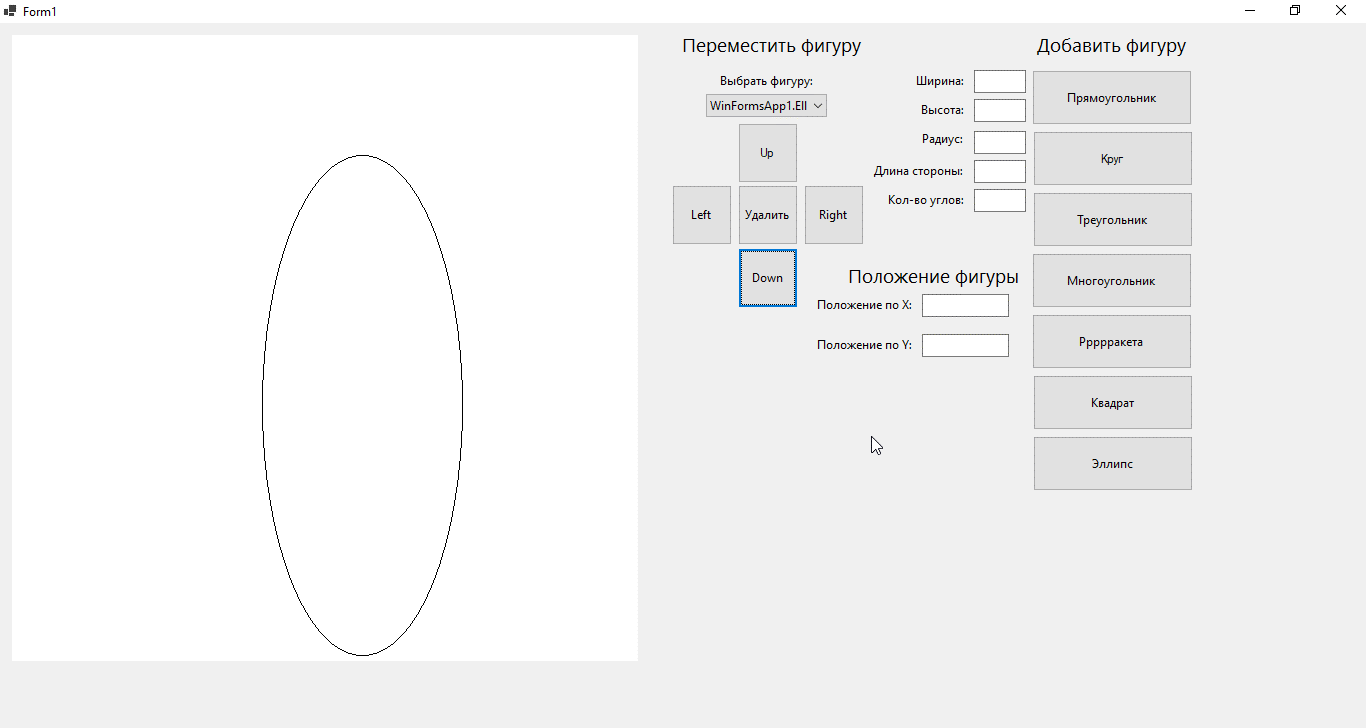


Рисунок 5 - Перемещение фигуры

**Листинг**

**Rocket.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http.Headers;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

public class Rocket:Figure

{

private Triangle nose; // нос

private Polygon fire; // огонек

private Polygon left; // левое крыло

private Polygon right; // правое крыло

private Rectangle body; // тело

private Ellipse c1; // иллюминатор 1

private Ellipse c2; // иллюминатор 2

private Ellipse c3; // иллюминатор 3

private PointF[] pointsl;

private PointF[] pointsr;

private PointF[] pointst;

private PointF[] pointsf;

public Rocket(int x = 100, int y = 100, int w = 50, int h = 150)

{

this.x = x;

this.y = y;

this.w = w;

this.h = h;

this.pointst = new PointF[3];

this.pointst[0] = new PointF((float)(x + w \* 0.25), (float)(y + h \* 0.3));

this.pointst[1] = new PointF((float)(x + w \* 0.5), y);

this.pointst[2] = new PointF((float)(x + w \* 0.75), (float)(y + h \* 0.3));

this.nose = new Triangle(pointst);

this.body = new Rectangle((int)(x + w \* 0.25), (int)(h \* 0.3), w / 2, (int)(h \* 0.6));

this.pointsl = new PointF[3];

this.pointsl[0] = new PointF((float)(x + w \* 0.25), (float)(y + h \* 0.6));

this.pointsl[1] = new PointF((float)(x + w \* 0.25), (float)(y + h \* 0.9));

this.pointsl[2] = new PointF(x, (float)(y + h \* 0.9));

this.left = new Polygon(this.pointsl);

this.pointsr = new PointF[3];

this.pointsr[0] = new PointF((float)(x + w \* 0.75), (float)(y + h \* 0.6));

this.pointsr[1] = new PointF((float)(x + w \* 0.75), (float)(y + h \* 0.9));

this.pointsr[2] = new PointF((float)(x + w), (float)(y + h \* 0.9));

this.right = new Polygon(this.pointsr);

this.c1 = new Ellipse((int)(x + w \* 0.33), (int)(y + h \* 0.4), w / 3, (int)(h \* 0.1));

this.c2 = new Ellipse((int)(x + w \* 0.33), (int)(y + h \* 0.55), w / 3, (int)(h \* 0.1));

this.c3 = new Ellipse((int)(x + w \* 0.33), (int)(y + h \* 0.7), w / 3, (int)(h \* 0.1));

this.pointsf = new PointF[9];

for (int i = 0; i < 9; i++)

{

if (i % 2 == 0) this.pointsf[i] = new PointF((float)(x + w \* 0.25 + i \* w / 16), (float)(y + h \* 0.9));

else this.pointsf[i] = new PointF((float)(x + w \* 0.25 + i \* w / 16), y + h);

}

this.fire = new Polygon(pointsf);

}

public override void draw()

{

this.nose.draw();

this.body.draw();

this.left.draw();

this.right.draw();

this.c1.draw();

this.c2.draw();

this.c3.draw();

this.fire.draw();

}

public override void move\_to(int x, int y)

{

if (this.move\_check(x, y))

{

this.nose.move\_to(x, y);

this.body.move\_to(x, y);

this.left.move\_to(x, y);

this.right.move\_to(x, y);

this.c1.move\_to(x, y);

this.c2.move\_to(x, y);

this.c3.move\_to(x, y);

this.fire.move\_to(x, y);

this.x += x;

this.y += y;

}

}

public override bool move\_check(int x, int y)

{

if (this.x + x < 0 || this.x + w + x > Init.pbw) return false;

else if (this.y + h > Init.pbh || this.y + y < 0) return false;

else return true;

}

}

}

**Rectangle.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

public class Rectangle:Figure

{

public Rectangle(int x=0, int y=0, int w=20, int h=20)

{

this.x = x;

this.y = y;

this.w = w;

this.h = h;

}

public override void draw()

{

Graphics g = Graphics.FromImage(Init.bitmap);

g.DrawRectangle(Init.pen, this.x, this.y, this.w, this.h);

Init.pb.Image = Init.bitmap;

}

public override bool move\_check(int x, int y)

{

// функция проверяет, можно ли переместить фигуру на заданные координаты

// в качестве ответа идет булевое значение. true - можно переместить, false - нельзя переместить

bool lls = this.x + x < 0; // выход за границу левой стороной

bool lts = this.y + y < 0; // выход за границу верхней стороной

bool lrs = this.x + this.w + x > Init.pbw; // выход за границу правой стороной

bool lbs = this.y + this.h + y > Init.pbh; // выход за границу нижней стороной

return !(lls || lts || lrs || lbs);

}

public override void move\_to(int x, int y)

{

if (this.move\_check(x, y))

{

this.x += x;

this.y += y;

this.drop\_figure(this, true);

this.draw();

}

}

}

}

**Polygon.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

public class Polygon : Figure

{

public int count;

public int a;

public PointF[] points;

public Polygon(PointF[] points = null)

{

this.points = points;

if (points == null) this.count = 3;

else this.count = points.Length;

}

public override void draw()

{

Graphics g = Graphics.FromImage(Init.bitmap);

g.DrawPolygon(Init.pen, this.points);

Init.pb.Image = Init.bitmap;

}

public override bool move\_check(int x, int y)

{

for (int i = 0; i < this.count; i++)

{

if (!(points[i].X + x <= Init.pbw && points[i].Y + y <= Init.pbh && points[i].X + x >= 0 && points[i].Y + y >= 0))

{

return false;

}

}

return true;

}

public override void move\_to(int x, int y)

{

if (this.move\_check(x, y))

{

for (int i = 0; i < this.count; i++)

{

this.points[i].X += x;

this.points[i].Y += y;

}

this.drop\_figure(this, true);

this.draw();

}

}

}

}

**Form2.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WinFormsApp1

{

public partial class Form2 : Form

{

private PointF[] points;

private int active\_angle = 0;

private int n\_angles;

public Form1 form1;

public Form2(int n\_angles, Form1 form1)

{

this.form1 = form1;

InitializeComponent();

this.n\_angles = n\_angles;

this.points = new PointF[n\_angles];

this.progressBar1.Maximum = n\_angles;

this.progressBar1.Minimum = 0;

this.progressBar1.Value = 0;

}

private void button1\_Click(object sender, EventArgs e)

{

int.TryParse(this.textBox1.Text, out int x);

int.TryParse(this.textBox2.Text, out int y);

if (x > 0 && x <= Init.pbw && y > 0 && y <= Init.pbh)

{

textBox1.Text = "";

textBox2.Text = "";

this.listBox1.Items.Add($"{x}, {y}");

this.points[this.active\_angle] = new PointF(x, y);

this.progressBar1.Value += 1;

this.active\_angle += 1;

} else {

MessageBox.Show("Неверный координаты точки!");

}

if (this.active\_angle == this.n\_angles)

{

MessageBox.Show("Многоугольник успешно создан!");

Polygon pol = new Polygon(this.points);

pol.draw();

ShapeContainer.AddFigure(pol);

this.form1.comboBox1.Items.Add(pol.get\_name());

this.Close();

}

}

}

}

**Form1.cs**

using System.Drawing;

using System.Reflection;

using System.Runtime.InteropServices;

using System.Security.Cryptography.X509Certificates;

namespace WinFormsApp1

{

public partial class Form1 : Form

{

private int w; // øèðèíà ôèãóðû

private int h; // âûñîòà ôèãóðû

private int r; // ðàäèóñ îêðóæíîñòè

private int a; // äëèíà ñòîðîíû

private int n; // êîëè÷åñòâî óãëîâ ìíîãîóãîëüíèêà

private int x\_pos; // ïîëîæåíèå ôèãóðû ïðè ñîçäàíèè ïî x

private int y\_pos; // ïîëîæåíèå ôèãóðû ïðè ñîçäàíèè ïî y

public Form1()

{

InitializeComponent();

Init.bitmap = new Bitmap(this.pictureBox1.ClientSize.Width, pictureBox1.ClientSize.Height); ;

Init.pen = new Pen(Color.Black, 1);

Init.pb = this.pictureBox1;

Init.pbw = Init.pb.Width;

Init.pbh = Init.pb.Height;

comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;

}

private bool get\_data(

bool rect = false,

bool sq = false,

bool el = false,

bool circ = false,

bool pol = false,

bool tri = false,

bool rocket = false

)

{

bool state = true;

if (rect || sq || rocket || el)

{

if (int.TryParse(this.textBox1.Text, out int w) && w > 0) this.w = w;

else

{

MessageBox.Show("Ââåäåí íåâåðíûé ôîðìàò äàííûõ â ïîëå øèðèíû ôèãóðû");

state = false;

}

}

if (rect || el || rocket)

{

if (int.TryParse(this.textBox2.Text, out int h) && h > 0) this.h = h;

else

{

MessageBox.Show("Ââåäåí íåâåðíûé ôîðìàò äàííûõ â ïîëå âûñîòû ôèãóðû");

state = false;

}

}

if (circ)

{

if (int.TryParse(this.textBox3.Text, out int r)) this.r = r;

else

{

MessageBox.Show("Ââåäåí íåâåðíûé ôîðìàò äàííûõ â ïîëå ðàäèóñà");

state = false;

}

}

if (tri)

{

if (int.TryParse(this.textBox4.Text, out int a)) this.a = a;

else

{

MessageBox.Show("Ââåäåí íåâåðíûé ôîðìàò äàííûõ â ïîëå äëèíû ñòîðîíû");

state = false;

}

}

if (pol)

{

if (int.TryParse(this.textBox5.Text, out int n)) this.n = n;

else

{

MessageBox.Show("Ââåäåí íåâåðíûé ôîðìàò äàííûõ â ïîëå êîëè÷åñòâà óãëîâ");

state = false;

}

}

if (int.TryParse(this.textBox8.Text, out int x\_pos))

{

if (0 <= x\_pos && x\_pos + this.w <= Init.pbw) this.x\_pos = x\_pos;

else

{

MessageBox.Show("Óêàçàííîå çíà÷åíèå x âûõîäèò çà ãðàíèöû ïîëÿ");

state = false;

}

}

else this.x\_pos = 0;

if (int.TryParse(this.textBox7.Text, out int y\_pos))

{

if (0 <= y\_pos && y\_pos + this.h <= Init.pbw) this.y\_pos = y\_pos;

else

{

MessageBox.Show("Óêàçàííîå çíà÷åíèå y âûõîäèò çà ãðàíèöû ïîëÿ");

state = false;

}

}

else this.y\_pos = 0;

if (state) return true;

else return false;

}

private void clear\_boxes()

{

this.textBox1.Text = "";

this.textBox2.Text = "";

this.textBox3.Text = "";

this.textBox4.Text = "";

this.textBox5.Text = "";

this.textBox7.Text = "";

this.textBox8.Text = "";

}

private void Form1\_Load(object sender, EventArgs e)

{

this.clear\_boxes();

}

private void button1\_Click(object sender, EventArgs e) // ïðÿìîóãîëüíèê

{

if (this.get\_data(rect: true))

{

Rectangle rectangle = new Rectangle(this.x\_pos, this.y\_pos, this.w, this.h);

if (rectangle.move\_check(0, 0))

{

rectangle.draw();

this.clear\_boxes();

ShapeContainer.AddFigure(rectangle);

this.comboBox1.Items.Add(rectangle.get\_name());

}

else MessageBox.Show("Íåâåðíûå ðàçìåðû ôèãóðû, âûõîäÿùèå çà ãðàíèöû ïîëÿ");

}

}

private void button2\_Click(object sender, EventArgs e) // êðóã

{

if (this.get\_data(circ: true))

{

Circle circle = new Circle(this.x\_pos, this.y\_pos, this.r);

if (circle.move\_check(0, 0))

{

circle.draw();

clear\_boxes();

ShapeContainer.AddFigure(circle);

this.comboBox1.Items.Add(circle.get\_name());

}

else MessageBox.Show("Íåâåðíûå ðàçìåðû ôèãóðû, âûõîäÿùèå çà ãðàíèöû ïîëÿ");

}

}

private void button3\_Click(object sender, EventArgs e) // òðåóãîëüíèê

{

if (this.get\_data(tri: true))

{

PointF[] points = new PointF[3];

points[0] = new PointF(this.x\_pos + this.a / 2, this.y\_pos);

points[1] = new PointF(this.x\_pos + this.a, this.y\_pos + this.a);

points[2] = new PointF(this.x\_pos, this.y\_pos + this.a);

Triangle triangle = new Triangle(points);

if (triangle.move\_check(0, 0))

{

triangle.draw();

clear\_boxes();

ShapeContainer.AddFigure(triangle);

this.comboBox1.Items.Add(triangle.get\_name());

}

else MessageBox.Show("Íåâåðíûå ðàçìåðû ôèãóðû, âûõîäÿùèå çà ãðàíèöû ïîëÿ");

}

}

private void button4\_Click(object sender, EventArgs e) // ìíîãîóãîëüíèê

{

if (this.get\_data(pol: true))

{

clear\_boxes();

Form2 frm = new Form2(this.n, this);

frm.ShowDialog();

}

}

private void button5\_Click(object sender, EventArgs e) // ðàêåòà

{

if (this.get\_data(rocket: true))

{

Rocket rocket = new Rocket(this.x\_pos, this.y\_pos, this.w, this.h);

if (rocket.move\_check(0, 0))

{

clear\_boxes();

rocket.draw();

ShapeContainer.AddFigure(rocket);

this.comboBox1.Items.Add(rocket.get\_name());

}

else MessageBox.Show("Íåâåðíûå ðàçìåðû ôèãóðû, âûõîäÿùèå çà ðàìêè ýêðàíà");

}

}

private void button10\_Click(object sender, EventArgs e) // êâàäðàò

{

if (this.get\_data(sq: true))

{

Square sq = new Square(this.x\_pos, this.y\_pos, this.w);

if (sq.move\_check(0, 0))

{

clear\_boxes();

sq.draw();

ShapeContainer.AddFigure(sq);

this.comboBox1.Items.Add(sq.get\_name());

}

else MessageBox.Show("Íåâåðíûå ðàçìåðû ôèãóðû, âûõîäÿùèå çà ãðàíèöû ïîëÿ");

}

}

private void button6\_Click(object sender, EventArgs e) // ýëëèïñ

{

if (this.get\_data(el: true))

{

Ellipse el = new Ellipse(this.x\_pos, this.y\_pos, this.w, this.h);

if (el.move\_check(0, 0))

{

this.clear\_boxes();

el.draw();

ShapeContainer.AddFigure(el);

this.comboBox1.Items.Add(el.get\_name());

}

else MessageBox.Show("Íåâåðíûå ðàçìåðû ôèãóðû, âûõîäÿùèå çà ãðàíèöû ïîëÿ");

}

}

private void left\_btn\_Click(object sender, EventArgs e)

{

string figure\_name = this.comboBox1.Text;

for (int i = 0; i < ShapeContainer.length; i++)

{

if (ShapeContainer.figureList[i].get\_name() == figure\_name)

{

Figure figure = ShapeContainer.figureList[i];

figure.move\_to(-10, 0);

break;

}

}

}

private void up\_btn\_Click(object sender, EventArgs e)

{

string figure\_name = this.comboBox1.Text;

for (int i = 0; i < ShapeContainer.length; i++)

{

if (ShapeContainer.figureList[i].get\_name() == figure\_name)

{

Figure figure = ShapeContainer.figureList[i];

figure.move\_to(0, -10);

break;

}

}

}

private void right\_btn\_Click(object sender, EventArgs e)

{

string figure\_name = this.comboBox1.Text;

for (int i = 0; i < ShapeContainer.length; i++)

{

if (ShapeContainer.figureList[i].get\_name() == figure\_name)

{

Figure figure = ShapeContainer.figureList[i];

figure.move\_to(10, 0);

break;

}

}

}

private void down\_btn\_Click(object sender, EventArgs e)

{

string figure\_name = this.comboBox1.Text;

for (int i = 0; i < ShapeContainer.length; i++)

{

if (ShapeContainer.figureList[i].get\_name() == figure\_name)

{

Figure figure = ShapeContainer.figureList[i];

figure.move\_to(0, 10);

break;

}

}

}

private void dropFigure\_Click(object sender, EventArgs e)

{

string figure\_name = this.comboBox1.Text;

for (int i = 0; i < ShapeContainer.length; i++)

{

if (ShapeContainer.figureList[i].get\_name() == figure\_name)

{

this.comboBox1.Items.Remove(figure\_name);

Figure figure = ShapeContainer.figureList[i];

figure.drop\_figure(figure);

break;

}

}

}

}

}

**Figure.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

abstract public class Figure

{

public int x, y, h, w; // объявляем переменные, характеризующие фигуру

private string name;

abstract public void draw();

abstract public void move\_to(int x, int y);

abstract public bool move\_check(int x, int y);

public Figure()

{

string t = DateTime.Now.Subtract(new DateTime(1970, 1, 1)).TotalSeconds.ToString();

this.name = this.ToString() + t;

}

public void drop\_figure(Figure f, bool redraw = false)

{

Graphics g = Graphics.FromImage(Init.bitmap);

if (!redraw) ShapeContainer.RemoveFigure(f);

this.clear();

Init.pb.Image = Init.bitmap;

for (int i = 0; i < ShapeContainer.length; i++)

{

ShapeContainer.figureList[i].draw();

}

}

public void clear()

{

Graphics g = Graphics.FromImage(Init.bitmap);

g.Clear(Color.White);

}

public string get\_name()

{

return this.name;

}

}

}

**Ellipse.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

public class Ellipse : Figure

{

public int r;

public Ellipse(int x=0, int y=0, int w=40, int h=40)

{

this.x = x;

this.y = y;

this.w = w;

this.h = h;

}

public override void draw()

{

Graphics g = Graphics.FromImage(Init.bitmap);

g.DrawEllipse(

Init.pen,

new RectangleF(

this.x,

this.y,

this.w,

this.h)

);

Init.pb.Image = Init.bitmap;

}

public override bool move\_check(int x, int y)

{

// функция проверяет, можно ли переместить фигуру на заданные координаты

// в качестве ответа идет булевое значение. true - можно переместить, false - нельзя переместить

bool lls = this.x + x < 0; // выход за границу левой стороной

bool lts = this.y + y < 0; // выход за границу верхней стороной

bool lrs = this.x + this.w + x > Init.pbw; // выход за границу правой стороной

bool lbs = this.y + this.h + y > Init.pbh; // выход за границу нижней стороной

return !(lls || lts || lrs || lbs);

}

public override void move\_to(int x, int y)

{

if (this.move\_check(x, y))

{

this.x += x;

this.y += y;

this.drop\_figure(this, true);

this.draw();

}

}

}

}

**Circle.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

public class Circle : Ellipse

{

public Circle(int x=0, int y=0, int r=50)

{

this.x = x;

this.y = y;

this.w = r;

this.h = r;

}

}

}

**Triangle.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

public class Triangle:Polygon

{

public Triangle(PointF[] points)

{

this.points = points;

this.count = 3;

}

}

}

**ShapeContainer.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

public static class ShapeContainer

{

public static List<Figure> figureList;

public static int length;

static ShapeContainer()

{

figureList = new List<Figure>();

length = 0;

}

public static void AddFigure(Figure figure)

{

figureList.Add(figure);

length += 1;

}

public static void RemoveFigure(Figure figure)

{

figureList.Remove(figure);

length -= 1;

}

}

}

**Square.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

public class Square : Rectangle

{

public Square(int x, int y, int w)

{

this.x = x;

this.y = y;

this.w = w;

this.h = w;

}

}

}