Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Уфимский университет науки и технологий»

Факультет информатики и робототехники

Кафедра вычислительной математики и кибернетики

Лабораторная работа №6

“Визуальный редактор”

По дисциплине “Объектно-ориентированное программирование”

Выполнил:

Студент группы ПРО-232Б

Степанов К.А.

Проверил:

доцент кафедры ВМиК

Котельников В. А.

Уфа-2023

# Цель работы

Продолжение изучения основ полиморфизма, создания интерфейса, работы с классами.

# Задание

* Создание графических объектов на экране:
  + меню, панель инструментов с доступными классами объектов
  + добавление нового объекта в рабочую область
  + изменение пользователем размера формы
* Манипуляции объектами в рабочей области:
  + выбор объекта для манипулирования (выделяется цветом или рамкой)
  + несколько одновременно выбранных объектов
  + изменение цвета, размера, положения
  + удаление объекта из рабочей области
  + контроль выхода за рабочую область (при передвижении объект не должен выходить за границы ни одной своей частью)
* Программные требования:
  + манипуляции объектами выполняются с помощью клавиатуры, при этом выбор клавиш управления должен быть как можно ближе к используемым обычно для таких задач в среде Windows
  + «хорошая» иерархия классов
  + использование своего контейнера-хранилища объектов из Л.Р.3
  + отделение взаимодействия с пользователем от логики работы классов

# Ход работы

Лабораторная построена на основе ЛР 4.1, дополняет и расширяет ее возможности.

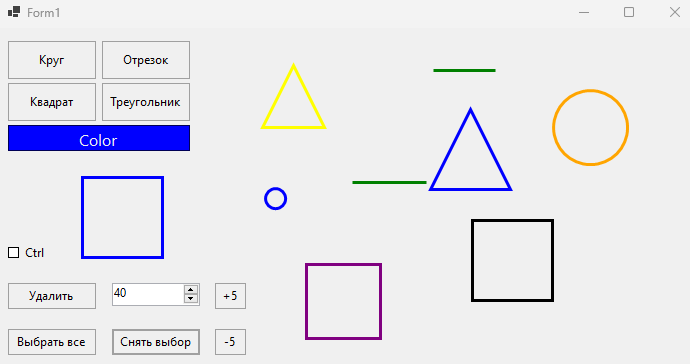


Рис. . Скриншот работы программы

# Выводы

По ходу выполнения работы были закреплены основы полиморфизма, работа с классами и пользовательскими интерфейсами.

Ссылка на GitHub: <https://github.com/stkirill15/OOP-6>.

# Приложение 1

**Form1.cs:**

using Microsoft.VisualBasic.Devices;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Globalization;

using System.Net.Http.Headers;

using System.Reflection;

using System.Windows.Forms;

using System.Windows.Forms.VisualStyles;

using static OOP6.Form1;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;

namespace OOP6

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private List<CShape> Shapes = new List<CShape>();

public int object\_radius = 10;

public bool Ctrl;

Color[] colors = { Color.Black, Color.Blue, Color.Yellow, Color.Green, Color.Orange, Color.Purple };

Color color = Color.Black;

int colorIndex = 0;

int selectedFigure = 0;

private void Form1\_Paint(object sender, PaintEventArgs e) // Отрисовка фигур

{

e.Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias; // Сглаживание

foreach (CShape figure in Shapes)

{

figure.Draw(e.Graphics); // Метод круга для отрисовки самого себя

}

}

private void Form1\_MouseClick(object sender, MouseEventArgs e)

{

if (!Ctrl) // Если не зажат Ctrl

{

foreach (CShape figure in Shapes) // Снятие выделения со всех фигур

{

figure.Select(false);

}

CShape newShape = new CShape();

switch (selectedFigure)

{

case 0:

newShape = new CCircle(e.X, e.Y, object\_radius, color);

newShape.Select(true);

Shapes.Add(newShape);

newShape.observers += new System.EventHandler(this.check\_borders);

newShape.sendShape();

break;

case 1:

newShape = new CSquare(e.X, e.Y, object\_radius, color);

newShape.Select(true);

Shapes.Add(newShape);

newShape.observers += new System.EventHandler(this.check\_borders);

newShape.sendShape();

break;

case 2:

newShape = new CTriangle(e.X, e.Y, object\_radius, color);

newShape.Select(true);

Shapes.Add(newShape);

newShape.observers += new System.EventHandler(this.check\_borders);

newShape.sendShape();

break;

case 3:

newShape = new CSection(e.X, e.Y, object\_radius, color);

newShape.Select(true);

newShape.observers += new System.EventHandler(this.check\_borders);

newShape.sendShape();

Shapes.Add(newShape);

break;

}

Refresh();

}

else if (Ctrl) // Если зажат ctrl

{

foreach (CShape figure in Shapes)

{

if (figure.MouseCheck(e)) // Если попала мышь

{

figure.Select(true); // Выделение кругов

break;

}

}

Refresh();

}

}

private void Form1\_KeyUp(object sender, KeyEventArgs e) // Отжатие кнопки

{

check\_ctrl.Checked = false;

}

private void Form1\_KeyDown(object sender, KeyEventArgs e) // Нажатие кнопок delete и ctrl

{

if (Control.ModifierKeys == Keys.Control)

{

check\_ctrl.Checked = true;

}

else if (e.KeyCode == Keys.Delete)

{

DelFigures();

}

else if (e.KeyCode == Keys.Up)

{

foreach (CShape figure in Shapes)

{

if (figure.selected && ((figure.coords.Y - figure.radius) > 0))

{

figure.coords.Y -= 3;

}

}

Refresh();

}

else if (e.KeyCode == Keys.Down)

{

foreach (CShape figure in Shapes)

{

if (figure.selected && ((figure.coords.Y + figure.radius) < (int)this.ClientSize.Height))

{

figure.coords.Y += 3;

}

}

Refresh();

}

else if (e.KeyCode == Keys.Left)

{

foreach (CShape figure in Shapes)

{

if (figure.selected && ((figure.coords.X - figure.radius) > 0))

{

figure.coords.X -= 3;

}

}

Refresh();

}

else if (e.KeyCode == Keys.Right)

{

foreach (CShape figure in Shapes)

{

if (figure.selected && ((figure.coords.X + figure.radius) < (int)this.ClientSize.Width))

{

figure.coords.X += 3;

}

}

Refresh();

}

else if (e.KeyCode == Keys.Oemplus)

{

Plus5\_button\_Click(sender, e);

}

else if (e.KeyCode == Keys.OemMinus)

{

Minus5\_button\_Click(sender, e);

}

}

public void check\_borders(object sender, EventArgs e)

{

int x = (sender as CShape).coords.X;

int y = (sender as CShape).coords.Y;

int rad = (sender as CShape).radius;

if (x + rad >= this.ClientSize.Width)

(sender as CShape).coords.X = this.ClientSize.Width - rad;

else if (x - rad <= 0)

(sender as CShape).coords.X = rad;

if (y + rad >= this.ClientSize.Height)

(sender as CShape).coords.Y = this.ClientSize.Height - rad;

else if (y - rad <= 0)

(sender as CShape).coords.Y = rad;

}

private void check\_ctrl\_CheckedChanged(object sender, EventArgs e)

{

Ctrl = check\_ctrl.Checked;

foreach (CShape figure in Shapes)

{

figure.fctrl = Ctrl;

}

}

private void Plus5\_button\_Click(object sender, EventArgs e)

{

if (shapeSize\_NumericUpDown.Value <= 95)

shapeSize\_NumericUpDown.Value += 5;

else shapeSize\_NumericUpDown.Value = 100;

foreach (CShape shape in Shapes)

{

if (shape.selected)

{

shape.radius = (int)shapeSize\_NumericUpDown.Value;

shape.sendShape();

}

}

Refresh();

}

private void Minus5\_button\_Click(object sender, EventArgs e)

{

if (shapeSize\_NumericUpDown.Value >= 15)

shapeSize\_NumericUpDown.Value -= 5;

else shapeSize\_NumericUpDown.Value = 10;

foreach (CShape shape in Shapes)

{

if (shape.selected)

{

shape.radius = (int)shapeSize\_NumericUpDown.Value;

shape.sendShape();

}

}

Refresh();

}

private void delete\_button\_Click(object sender, EventArgs e)

{

DelFigures();

}

void DelFigures() // Метод удаления фигур

{

for (int i = 0; i < Shapes.Count; i++)

{

if (Shapes[i].selected == true)

{

Shapes.Remove(Shapes[i]);

i--;

}

}

Refresh();

}

private void removeSelection\_button\_Click(object sender, EventArgs e)

{

foreach (CShape figure in Shapes) // снятие выделения со всех объектов

{

figure.Select(false);

}

Refresh();

}

private void button\_circle\_Click(object sender, EventArgs e)

{

selectedFigure = 0;

}

private void button\_square\_Click(object sender, EventArgs e)

{

selectedFigure = 1;

}

private void button\_triangle\_Click(object sender, EventArgs e)

{

selectedFigure = 2;

}

private void button\_section\_Click(object sender, EventArgs e)

{

selectedFigure = 3;

}

private void Color\_Button\_Click(object sender, EventArgs e)

{

colorIndex = colorIndex < colors.Length - 1 ? colorIndex + 1 : 0;

color = colors[colorIndex];

Color\_Button.BackColor = color;

foreach (CShape figure in Shapes) // Выделенные фигуры меняют цвет

{

if (figure.selected)

figure.shape\_color = color;

}

Refresh();

}

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

{

foreach (Control control in this.Controls)

{

control.PreviewKeyDown += new PreviewKeyDownEventHandler(control\_PreviewKeyDown);

}

}

void control\_PreviewKeyDown(object sender, PreviewKeyDownEventArgs e)

{

if (e.KeyCode == Keys.Up || e.KeyCode == Keys.Down || e.KeyCode == Keys.Left || e.KeyCode == Keys.Right)

{

e.IsInputKey = true;

}

}

private void ChoiceAll\_button\_Click(object sender, EventArgs e)

{

foreach (CShape figure in Shapes) // выделения всех объектов

{

figure.Select(true);

}

Refresh();

}

private void shapeSize\_NumericUpDown\_ValueChanged(object sender, EventArgs e)

{

object\_radius = ((int)shapeSize\_NumericUpDown.Value);

foreach (CShape shape in Shapes)

{

if (shape.selected)

{

shape.radius = (int)shapeSize\_NumericUpDown.Value;

shape.sendShape();

}

}

Refresh();

}

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

{

foreach (CShape shape in Shapes)

{

shape.sendShape();

}

Refresh();

}

}

}

public class CShape

{

public Point coords; // координаты

public int radius; // радиус

public bool selected = false; // отмеченность

public bool fctrl = false; // зажатый ctrl

public System.EventHandler observers;

public Color selected\_color = Color.Red; // Цвет "отметки"

public Color shape\_color = Color.Black; // Цвет фигуры

public void Select(bool condition) // метод переключения выделения

{

selected = condition;

}

public virtual void Draw(Graphics g) // Метод для отрисовки самого себя

{

}

public virtual bool MouseCheck(MouseEventArgs e) // Проверка объекта на попадание в него курсора

{

return false;

}

public void sendShape() // Отправка фигуры обработчику

{

observers.Invoke(this, null);

}

}

public class CCircle : CShape // класс круга

{

public CCircle(int x, int y, int radius, Color color) // конструктор по умолчанию

{

coords.X = x;

coords.Y = y;

this.radius = radius;

shape\_color = color;

}

public override void Draw(Graphics g) // Метод для отрисовки самого себя

{

if (selected == true)

g.DrawEllipse(new Pen(selected\_color, 3), coords.X - radius, coords.Y - radius, radius \* 2, radius \* 2);

else

g.DrawEllipse(new Pen(shape\_color, 3), coords.X - radius, coords.Y - radius, radius \* 2, radius \* 2);

}

public override bool MouseCheck(MouseEventArgs e) // Проверка объекта на попадание в него курсора

{

if (fctrl)

{

if (Math.Pow(e.X - coords.X, 2) + Math.Pow(e.Y - coords.Y, 2) <= Math.Pow(radius, 2) && !selected)

{

selected = true;

return true;

}

}

return false;

}

}

public class CSquare : CShape // класс квадрата

{

public CSquare(int x, int y, int radius, Color color) // конструктор по умолчанию

{

coords.X = x;

coords.Y = y;

this.radius = radius;

shape\_color = color;

}

public override void Draw(Graphics g) // Метод для отрисовки самого себя

{

if (selected == true)

g.DrawRectangle(new Pen(selected\_color, 3), coords.X - radius, coords.Y - radius, radius \* 2, radius \* 2);

else

g.DrawRectangle(new Pen(shape\_color, 3), coords.X - radius, coords.Y - radius, radius \* 2, radius \* 2);

}

public override bool MouseCheck(MouseEventArgs e) // Проверка объекта на попадание в него курсора

{

if (fctrl)

{

if (Math.Pow(e.X - coords.X, 2) + Math.Pow(e.Y - coords.Y, 2) <= Math.Pow(radius, 2) && !selected)

{

selected = true;

return true;

}

}

return false;

}

}

public class CTriangle : CShape // класс треугольника

{

public CTriangle(int x, int y, int radius, Color color) // конструктор по умолчанию

{

coords.X = x;

coords.Y = y;

this.radius = radius;

shape\_color = color;

}

public override void Draw(Graphics g) // Метод для отрисовки самого себя

{

Point point1 = new Point(coords.X, coords.Y - radius);

Point point2 = new Point(coords.X + radius, coords.Y + radius);

Point point3 = new Point(coords.X - radius, coords.Y + radius);

Point[] curvePoints = { point1, point2, point3 };

if (selected == true)

g.DrawPolygon(new Pen(selected\_color, 3), curvePoints);

else

g.DrawPolygon(new Pen(shape\_color, 3), curvePoints);

}

public override bool MouseCheck(MouseEventArgs e) // Проверка объекта на попадание в него курсора

{

if (fctrl)

{

if (Math.Pow(e.X - coords.X, 2) + Math.Pow(e.Y - coords.Y, 2) <= Math.Pow(radius, 2) && !selected)

{

selected = true;

return true;

}

}

return false;

}

}

public class CSection : CShape // класс отрезка

{

public CSection(int x, int y, int radius, Color color) // конструктор по умолчанию

{

coords.X = x;

coords.Y = y;

this.radius = radius;

shape\_color = color;

}

public override void Draw(Graphics g) // Отрисовка отрезка

{

Point point1 = new Point(coords.X - radius, coords.Y);

Point point2 = new Point(coords.X + radius, coords.Y);

Point[] curvePoints = { point1, point2 };

if (selected == true)

g.DrawPolygon(new Pen(selected\_color, 3), curvePoints);

else

g.DrawPolygon(new Pen(shape\_color, 3), curvePoints);

}

public override bool MouseCheck(MouseEventArgs e) // Проверка попадания курсора на объект

{

if (fctrl)

{

if (Math.Pow(e.X - coords.X, 2) + Math.Pow(e.Y - coords.Y, 2) <= Math.Pow(radius, 2) && !selected)

{

selected = true;

return true;

}

}

return false;

}

}