Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Уфимский государственный авиационный технический университет»

Факультет информатики и робототехники

Кафедра вычислительной математики и кибернетики

Отчет по лабораторной работе №6

по дисциплине «Объектно-ориентированное программирование»

на тему: «Визуальный редактор»

Выполнил:

Студент группы ПРО-228Б И.И.Гареев

Проверил: В.А. Котельников

Уфа – 2022

Содержание

[1 Цель работы 3](#_Toc93854453)

[2 Программная часть 4](#_Toc93854454)

[2.1 Базовый класс Figure 4](#_Toc93854455)

[2.2 Класс круга CCircle, класс квадрата Square и класс треугольника Triangle 4](#_Toc93854456)

[2.3 Хранилище 4](#_Toc93854457)

[2.4 Событие panel2\_MouseClick 5](#_Toc93854458)

[2.5 Событие panel2\_Paint 5](#_Toc93854459)

[2.6 Событие Form1\_KeyDown 5](#_Toc93854460)

[3 Заключение 8](#_Toc93854461)

[4 Приложение А 9](#_Toc93854462)

# Цель работы

На основе Л.Р.4 реализовать простейший визуальный редактор векторных объектов со следующей функциональностью:

1. Создание графических объектов на экране.
2. Манипуляции объектами в рабочей области.

# Программная часть

## **Базовый класс Figure**

Создаю базовый класс Figure как в Лр 4, в котором есть методы Draw, DoCheckTrue, DoCheckFalse, Move, ChangeSize, ChangeColor, булевые переменные isClicked, isChecked.

## Класс круга CCircle, класс квадрата Square и класс треугольника Triangle

Создаю классы CCircle, Square, Triangle, потомоки класса Figure, в которых есть конструкторы с параметрами, булевые переменные isClicked, отлавливающие клик мыши, методы Draw, рисующие фигуры, если Checked true, то фигуры выделяется, иначе просто рисуются, методы DoCheckTrue и DoCheckFalse, делающие Checked либо true, либо false, также булевые переменные isChecked, которые просто возвращают Checked. Методы Move проверяют нажатие клавиш ←, ↑, →, ↓ и соответственно передвигают фигуры. Методы ChangeSize проверяют нажатие клавиш + или –, соответственно увеличивая, либо уменьшая фигуры. Методы ChangeColor проверяют нажатие на pictureBox и присваивают кисти тот цвет, который изображен на pictureBox.

## Хранилище

Как в Лр 3 создаю класс MyStorage, в котором есть конструктор без параметров, конструктор с параметрами, метод SetObject и AddObject для добавления объектов, метод DeleteObject для удаления объектов и DeleteCheckObject для удаления выделенных объектов. Также в хранилище есть переменная типа bool isCheckedStorage, в которой я проверяю нажатие на круг, и если true, то вызываю у объекта метод DoCheckedTrue, метод NotChecked, который снимает выделение с объектов, метод Draw, который вызывает у объектов метод Draw, метод Move, проверяющий, если объект выделен, то вызывается у него метод Move, метод ChangeSize, проверяющий, если объект выделен, то вызывается у него метод ChangeSize, метод ChangeColor, проверяющий, если объект выделен, то вызывается у него метод ChangeColor.

## Событие panel2\_MouseClick

Проверяется условие, если нажато на пустое место, то с помощью switch выбирается какая фигура будет нарисована, в зависимости на какой pictureBox нажали, иначе если зажата клавиша ctrl, то определенные круги выделяются, иначе снимаю выделение и, проверяю нажатие на pictureBox в котором определены цвета и соответственно меняю цвет.

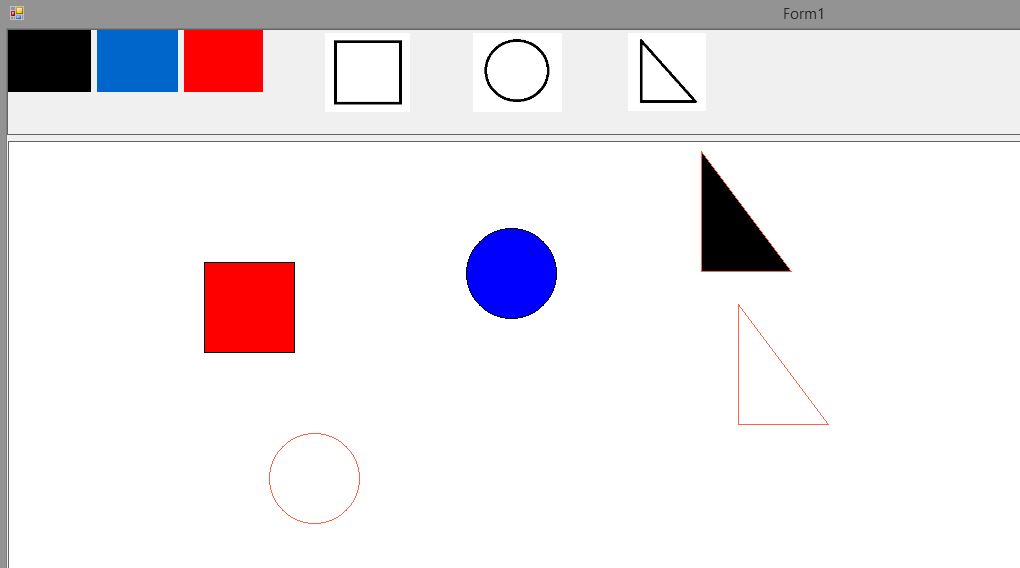


Рисунок 2.1

## Событие panel2\_Paint

Здесь просто рисуются фигуры.

## Событие Form1\_KeyDown

Проверяется условие, если нажата клавиша delete, то выделенный объект удаляется.

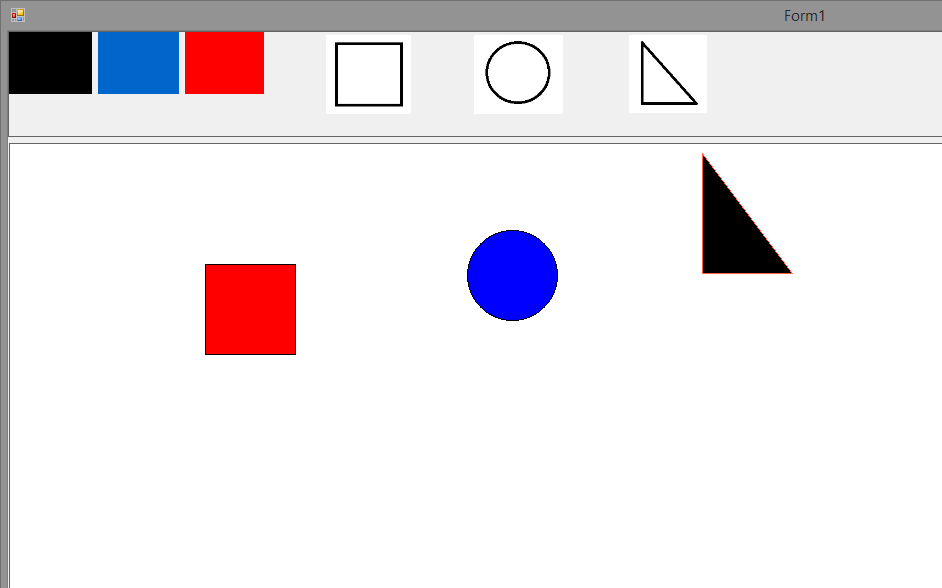


Рисунок 2.

Проверяется условие, если нажаты клавиши ←, ↑, →, ↓, то выделенный объект перемещается.

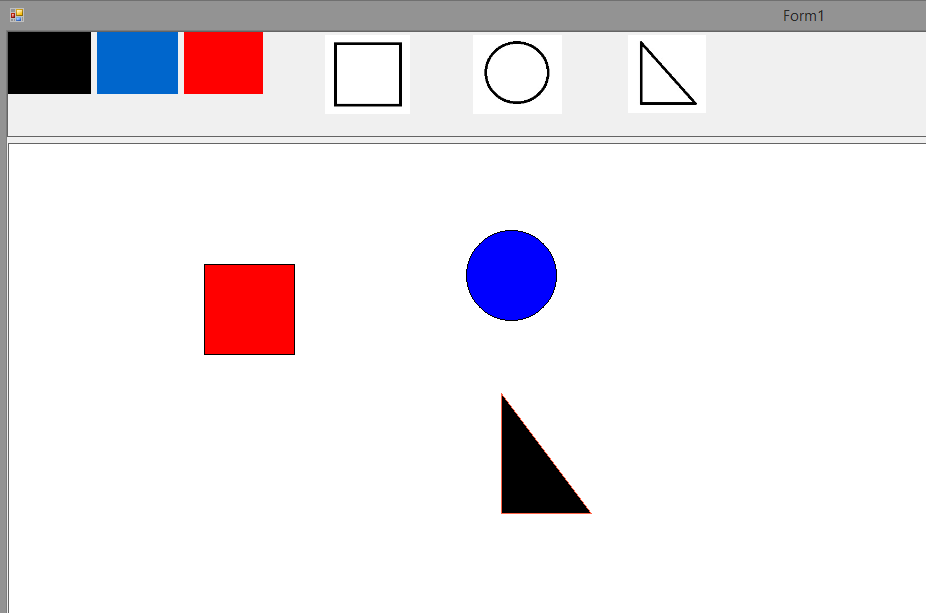


Рисунок 2.

Проверяется условие, если нажаты клавиши + или –, то выделенный объект либо увеличивается, либо уменьшается.

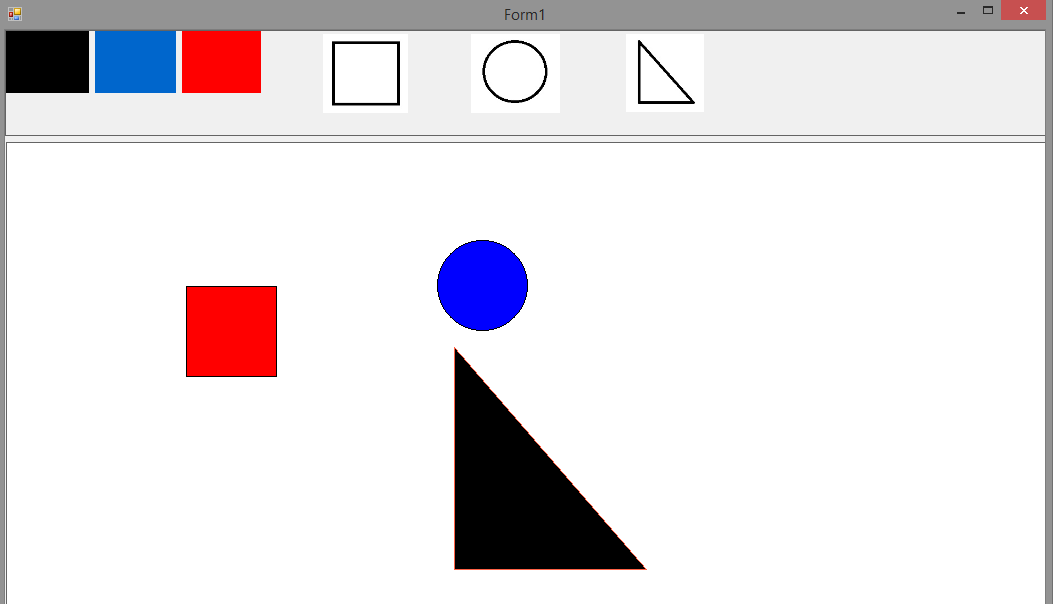


Рисунок 2.

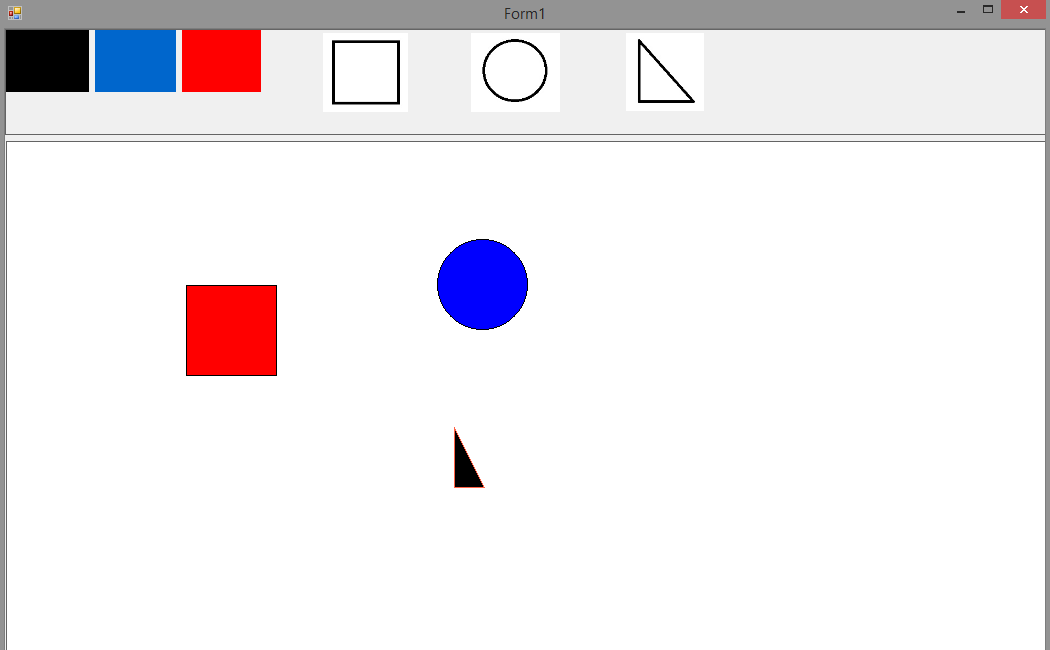


Рисунок 2.5

# Заключение

В ходе данной лабораторной работы я научился манипулировать объектами с помощью клавиатуры. Изменять цвет, размер, положение объектов, удалять их.

<https://github.com/IlyasGareev/LR6>

# Приложение А

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 LR6

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

class Figure //базовый класс

{

virtual public bool isClicked(MouseEventArgs e)

{

return true;

}

virtual public void Draw(Panel panel1, Graphics g)

{

}

virtual public void DoCheckTrue()

{

}

virtual public void DoCheckFalse()

{

}

virtual public bool isChecked()

{

return true;

}

virtual public void Move(KeyEventArgs e)

{

}

virtual public void ChangeSize(KeyEventArgs e)

{

}

virtual public void ChangeColor(int color)

{

}

}

class CCircle : Figure //класс круга

{

private int x;

private int y;

private int rad;

private bool Checked;

private Pen pen;

private SolidBrush brush;

public CCircle(int x, int y, int rad)

{

this.x = x;

this.y = y;

this.rad = rad;

Checked = false;

brush = new SolidBrush(Color.White);

}

override public bool isClicked(MouseEventArgs e)

{

if (((e.X - x) \* (e.X - x) + (e.Y - y) \* (e.Y - y)) <= rad \* rad)

{

return true;

}

else

{

return false;

}

}

public override void Draw(Panel panel1, Graphics g)

{

Rectangle rect = new Rectangle(x - rad, y - rad, rad \* 2, rad \* 2);

if (Checked == true)

{

pen = new Pen(Color.Tomato);

}

else

{

pen = new Pen(Color.Black);

}

g.FillEllipse(brush, rect);

g.DrawEllipse(pen, rect);

}

override public void DoCheckTrue()

{

Checked = true;

}

override public void DoCheckFalse()

{

Checked = false;

}

override public bool isChecked()

{

return Checked;

}

public override void Move(KeyEventArgs e)

{

if (e.KeyValue == 38)

{

y = y - 10;

}

if (e.KeyValue == 40)

{

y = y + 10;

}

if (e.KeyValue == 39)

{

x = x + 10;

}

if (e.KeyValue == 37)

{

x = x - 10;

}

}

public override void ChangeSize(KeyEventArgs e)

{

if (e.KeyValue == 187)

{

rad = rad + 5;

}

if (e.KeyValue == 189)

{

rad = rad - 5;

}

}

public override void ChangeColor(int color)

{

switch (color)

{

case 1:

brush.Color = Color.Black;

break;

case 2:

brush.Color = Color.Blue;

break;

case 3:

brush.Color = Color.Red;

break;

}

}

}

class Square : Figure //класс квадрата

{

private int x;

private int y;

private int rad;

private bool Checked;

private SolidBrush brush;

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

{

this.x = x;

this.y = y;

this.rad = rad;

Checked = false;

brush = new SolidBrush(Color.White);

}

override public bool isClicked(MouseEventArgs e)

{

if (((e.X - x) \* (e.X - x) + (e.Y - y) \* (e.Y - y)) <= rad \* rad \* 2)

{

return true;

}

else

{

return false;

}

}

public override void Draw(Panel panel1, Graphics g)

{

Rectangle rect = new Rectangle(x - rad, y - rad, rad \* 2, rad \* 2);

if (Checked == true)

{

g.FillRectangle(brush, rect);

g.DrawRectangle(Pens.Tomato, rect);

}

else

{

g.FillRectangle(brush, rect);

g.DrawRectangle(Pens.Black, rect);

}

}

override public void DoCheckTrue()

{

Checked = true;

}

override public void DoCheckFalse()

{

Checked = false;

}

override public bool isChecked()

{

return Checked;

}

public override void Move(KeyEventArgs e)

{

if (e.KeyValue == 38)

{

y = y - 10;

}

if (e.KeyValue == 40)

{

y = y + 10;

}

if (e.KeyValue == 39)

{

x = x + 10;

}

if (e.KeyValue == 37)

{

x = x - 10;

}

}

public override void ChangeSize(KeyEventArgs e)

{

if (e.KeyValue == 187)

{

rad = rad + 5;

}

if (e.KeyValue == 189)

{

rad = rad - 5;

}

}

public override void ChangeColor(int color)

{

switch (color)

{

case 1:

brush.Color = Color.Black;

break;

case 2:

brush.Color = Color.Blue;

break;

case 3:

brush.Color = Color.Red;

break;

}

}

}

class Triangle : Figure //класс треугольника

{

private Point[] p;

private bool Checked;

private int side;

private int x, y;

private SolidBrush brush;

public Triangle(int x, int y, int side)

{

p = new Point[3];

p[0].X = x;

p[0].Y = y - 60;

p[1].X = x;

p[1].Y = y + 60;

p[2].X = x + 90;

p[2].Y = y + 60;

Checked = false;

this.side = side;

this.x = x;

this.y = y;

brush = new SolidBrush(Color.White);

}

override public bool isClicked(MouseEventArgs e)

{

if (e.X >= x - side / 2 && e.Y >= y - side / 2 && e.X <= x + side / 2 && e.Y <= y + side / 2)

{

return true;

}

else

{

return false;

}

}

public override void Draw(Panel panel1, Graphics g)

{

if (Checked == true)

{

g.FillPolygon(brush, p);

g.DrawPolygon(Pens.Tomato, p);

}

else

{

g.FillPolygon(brush, p);

g.DrawPolygon(Pens.Black, p);

}

}

override public void DoCheckTrue()

{

Checked = true;

}

override public void DoCheckFalse()

{

Checked = false;

}

override public bool isChecked()

{

return Checked;

}

public override void Move(KeyEventArgs e)

{

if (e.KeyValue == 38)

{

p[0].Y = p[0].Y - 5;

p[1].Y = p[1].Y - 5;

p[2].Y = p[2].Y - 5;

}

if (e.KeyValue == 40)

{

p[0].Y = p[0].Y + 5;

p[1].Y = p[1].Y + 5;

p[2].Y = p[2].Y + 5;

}

if (e.KeyValue == 37)

{

p[0].X = p[0].X - 5;

p[1].X = p[1].X - 5;

p[2].X = p[2].X - 5;

}

if (e.KeyValue == 39)

{

p[0].X = p[0].X + 5;

p[1].X = p[1].X + 5;

p[2].X = p[2].X + 5;

}

}

public override void ChangeSize(KeyEventArgs e)

{

if (e.KeyValue == 187)

{

p[0].X = p[0].X;

p[0].Y = p[0].Y - 3;

p[1].X = p[1].X;

p[1].Y = p[1].Y + 3;

p[2].X = p[2].X + 6;

p[2].Y = p[2].Y + 3;

}

if (e.KeyValue == 189)

{

p[0].X = p[0].X;

p[0].Y = p[0].Y + 3;

p[1].X = p[1].X;

p[1].Y = p[1].Y - 3;

p[2].X = p[2].X - 6;

p[2].Y = p[2].Y - 3;

}

}

public override void ChangeColor(int color)

{

switch (color)

{

case 1:

brush.Color = Color.Black;

break;

case 2:

brush.Color = Color.Blue;

break;

case 3:

brush.Color = Color.Red;

break;

}

}

}

class MyStorage

{

private int size;

Figure[] storage;

public MyStorage()

{

size = 0;

}

public MyStorage(int size)

{

this.size = size;

storage = new Figure[size];

}

public void SetObjects(int index, Figure obj)

{

storage[index] = obj;

}

public void AddObject(Figure obj)

{

Figure[] new\_storage = new Figure[size + 1];

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

{

new\_storage[i] = storage[i];

}

storage = new\_storage;

storage[size] = obj;

size = size + 1;

}

public void DeleteObject(int index)

{

Figure[] new\_storage = new Figure[size - 1];

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

{

new\_storage[i] = storage[i];

}

for (int i = index; i < size - 1; i++)

{

new\_storage[i] = storage[i + 1];

}

size = size - 1;

storage = new\_storage;

}

public bool isCheckedStorage(MouseEventArgs e)

{

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

{

if (storage[i].isClicked(e) == true)

{

storage[i].DoCheckTrue();

return true;

}

}

return false;

}

public void DeleteCheckObject(MyStorage storage)

{

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

{

if (this.storage[i].isChecked() == true)

{

storage.DeleteObject(i);

i = i - 1;

}

}

}

public void NotChecked()

{

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

{

storage[i].DoCheckFalse();

}

}

public void Draw(Panel panel1, Graphics g)

{

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

{

storage[i].Draw(panel1, g);

}

}

public void Move(KeyEventArgs e)

{

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

{

if (storage[i].isChecked() == true)

{

storage[i].Move(e);

}

}

}

public void ChangeSize(KeyEventArgs e)

{

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

{

if (storage[i].isChecked() == true)

{

storage[i].ChangeSize(e);

}

}

}

public void ChangeColor(int color)

{

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

{

if (storage[i].isChecked() == true)

{

storage[i].ChangeColor(color);

}

}

}

}

MyStorage storage = new MyStorage();

int figure;

int color;

private void panel2\_Paint(object sender, PaintEventArgs e)

{

Graphics g = panel2.CreateGraphics();

storage.Draw(panel2, g);

}

private void panel2\_MouseClick(object sender, MouseEventArgs e)

{

if (storage.isCheckedStorage(e) == false)

{

storage.NotChecked();

switch (figure)

{

case 1:

CCircle ellipse = new CCircle(e.X, e.Y, 45);

storage.AddObject(ellipse);

break;

case 2:

Square square = new Square(e.X, e.Y, 45);

storage.AddObject(square);

break;

case 3:

Triangle triangle = new Triangle(e.X, e.Y, 90);

storage.AddObject(triangle);

break;

}

}

else

{

if (Control.ModifierKeys == Keys.Control)

{

storage.isCheckedStorage(e);

}

else

{

storage.NotChecked();

storage.isCheckedStorage(e);

if (color == 1 || color == 2 || color == 3)

{

storage.ChangeColor(color);

}

}

}

Refresh();

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

Graphics g = panel1.CreateGraphics();

if (e.KeyValue == 46)

{

storage.DeleteCheckObject(storage);

g.Clear(Color.White);

}

if (e.KeyValue == 38 || e.KeyValue == 40 || e.KeyValue == 39 || e.KeyValue == 37)

{

g.Clear(Color.White);

storage.Move(e);

}

if (e.KeyValue == 187 || e.KeyValue == 189)

{

g.Clear(Color.White);

storage.ChangeSize(e);

}

Refresh();

}

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

{

figure = 2;

}

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

{

figure = 1;

}

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

{

figure = 3;

}

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

{

color = 1;

}

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

{

color = 2;

}

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

{

color = 3;

}

}

}