МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Гомельский государственный технический

университет имени П.О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Отчёт по лабораторной работе №5

По дисциплине «Объектно-ориентированное программирование»

**«****События»**

Выполнил:

Студент группы ИТП-21

Коркуц С. И.

Принял:

ассистент Карабчикова Е.А.

Гомель 2020

**Цель работы:** изучить работу с событиями

**Задание:**

Создать графическое приложение для динамического управления над фигурами прямоугольника, круга и звезды. На левый клик мыши фигура должна активироваться и впоследствии её можно будет перемещать клавишами клавиатуры по экрану. На правый клик – добавление фигуры. Двойной клик по любой фигуре – замена фигуры. Наведение мыши – изменение цвета фигуры.

На рисунке 1 изображена структура решения.

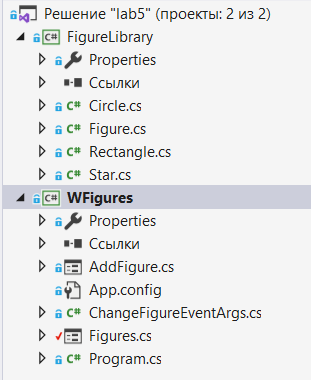


Рисунок 1 – Структура решения

На рисунке 2 изображен внешний вид программы.

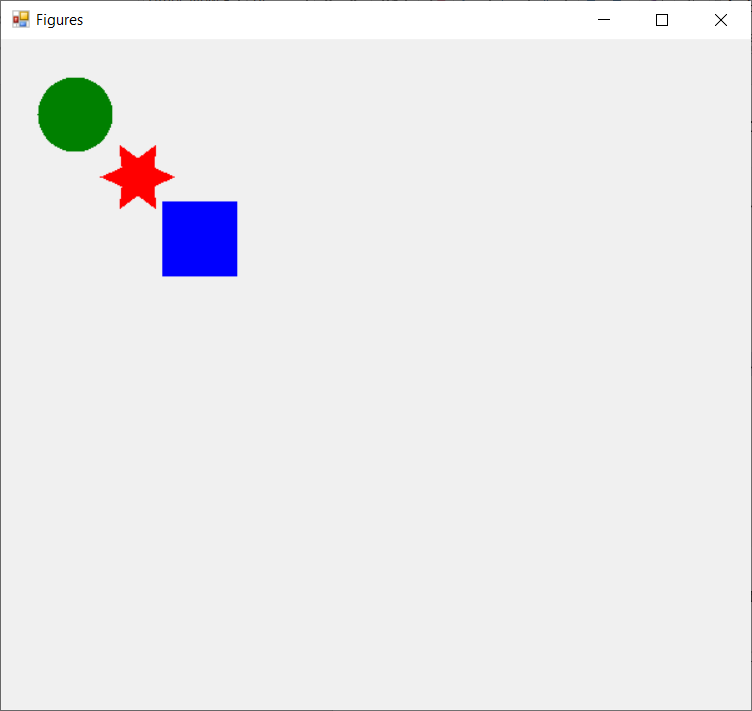


Рисунок 2 – внешний вид программы

При наведении мыши на объект его цвет меняется (рисунок 3).

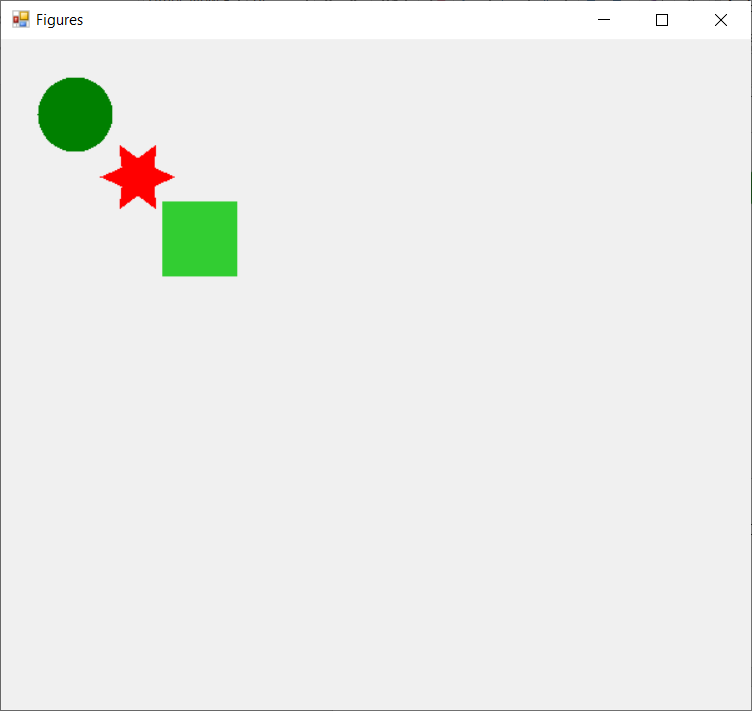


Рисунок 3 – изменение цвета объекта при наведении мыши

При нажатии на правую кнопку мыши выводится меню создания объекта в точке нажатия мыши (рисунок 4).

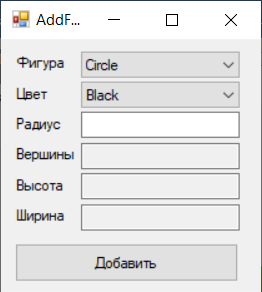


Рисунок 4 – окно создания объекта

При нажатии левой кнопки мыши происходит активация объекта, и его можно перемещать при помощи клавиш клавиатуры (рисунки 5 – 6).

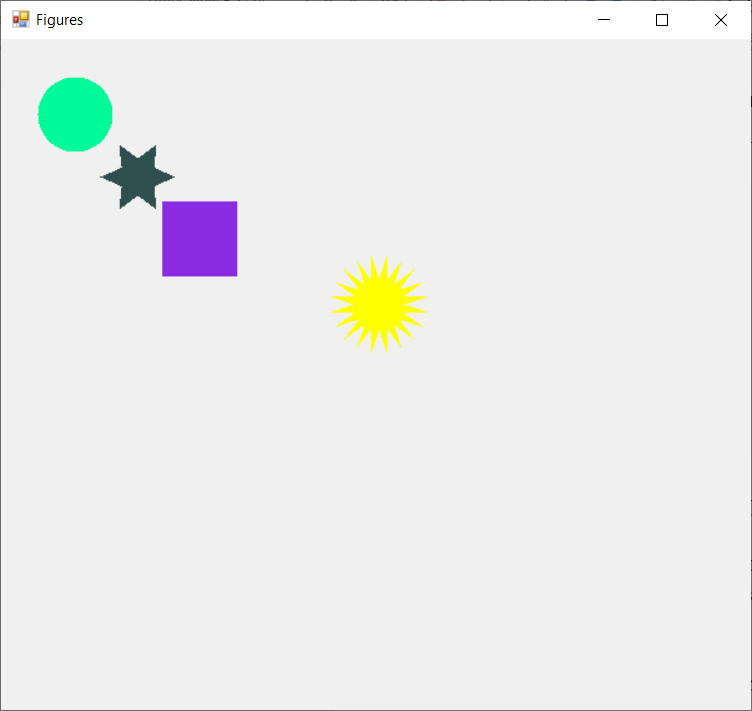


Рисунок 5 – созданный объект на первоначальном месте

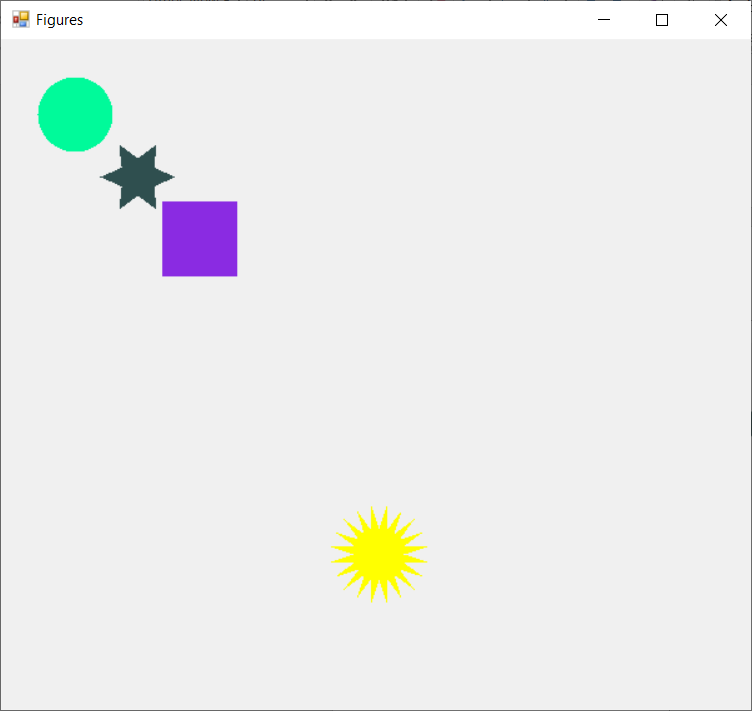


Рисунок 6 – перемещенный объект

**Вывод:** Создано графическое приложение для работы с фигурами. Реализованы пользовательские события.

**ПРИЛОЖЕНИЕ А**

Листинг классов фигур.

using System;

using System.Drawing;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FigureLibrary

{

public class Figure

{

public Figure(double x, double y, Brush brush)

{

X\_center = x;

Y\_center = y;

Brush = brush;

IsSelected = false;

}

public double X\_center { get; protected set; }

public double Y\_center { get; protected set; }

public Brush Brush { get; set; }

public bool IsSelected { get; set; }

public virtual void Draw(PaintEventArgs e)

{

throw new Exception("Нельзя нарисовать абстрактную фигуру");

}

public virtual void Move(double x\_shift, double y\_shift)

{

X\_center += x\_shift;

Y\_center += y\_shift;

}

public virtual bool IsPointIn(double x, double y)

{

throw new Exception("Нельзя проверить на вхождение точку в абстрактной фигуре");

}

}

public class Rectangle : Figure

{

public double Width { get; set; }

public double Height { get; set; }

public Rectangle(double x, double y, Brush brush, double w, double h) : base(x, y, brush)

{

Width = w;

Height = h;

}

public override void Draw(PaintEventArgs e)

{

e.Graphics.FillRectangle(Brush, (float)(X\_center - Width / 2), (float)(Y\_center - Height / 2), (float)Width, (float)Height);

}

public override void Move(double x\_shift, double y\_shift)

{

base.Move(x\_shift, y\_shift);

}

public override bool IsPointIn(double x, double y)

{

if (X\_center - Width / 2 <= x && x <= X\_center + Width / 2 && Y\_center - Height / 2 <= y && y <= Y\_center + Height / 2)

return true;

return false;

}

}

public class Circle : Figure

{

public double Radius { get; set; }

public Circle(double x, double y, Brush brush, double radius) : base(x, y, brush)

{

Radius = radius;

}

public override void Draw(PaintEventArgs e)

{

e.Graphics.FillEllipse(Brush, (float)(X\_center - Radius), (float)(Y\_center - Radius), (float)(Radius \* 2), (float)(Radius \* 2));

}

public override void Move(double x\_shift, double y\_shift)

{

base.Move(x\_shift, y\_shift);

}

public override bool IsPointIn(double x, double y)

{

if (Math.Pow(x - X\_center, 2) + Math.Pow(y - Y\_center, 2) <= Math.Pow(Radius, 2))

return true;

return false;

}

}

public class Star : Circle

{

public int VerticesNumber { get; }

public double Angle { get; }

public PointF[] Points { get; private set; }

public Star(double x, double y, Brush brush, double radius, int verticesNumber, double angle) : base(x, y, brush, radius)

{

VerticesNumber = verticesNumber;

Angle = angle;

InitializePoints();

}

public override void Draw(PaintEventArgs e)

{

e.Graphics.FillPolygon(Brush, Points);

}

public override void Move(double x\_shift, double y\_shift)

{

base.Move(x\_shift, y\_shift);

for (int i = 0; i < Points.Length; i++)

{

Points[i].X += (float)x\_shift;

Points[i].Y += (float)y\_shift;

}

}

private void InitializePoints()

{

Points = new PointF[2 \* VerticesNumber + 1];

double a = Angle, da = Math.PI / VerticesNumber, r = Radius / 2, l;

for(int k = 0; k < 2 \* VerticesNumber + 1; k++)

{

l = k % 2 == 0 ? r : Radius;

Points[k] = new PointF((float)(X\_center + l \* Math.Cos(a)), (float)(Y\_center + l \* Math.Sin(a)));

a += da;

}

}

}

}

**ПРИЛОЖЕНИЕ Б**

Листинг графического приложения

using System;

using fl = FigureLibrary;

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;

using System.Reflection;

namespace WFigures

{

public partial class Figures : Form

{

public List<fl.Figure> figures;

public Figures()

{

InitializeComponent();

figures = new List<fl.Figure>();

StartInitialization();

EditFigureEvent += EditFigure;

}

private void StartInitialization()

{

figures.Add(new fl.Circle(50, 50, Brushes.Green, 30));

figures.Add(new fl.Star(100, 100, Brushes.Red, 30, 6, Math.PI / 2));

figures.Add(new fl.Rectangle(150, 150, Brushes.Blue, 60, 60));

}

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

{

foreach(fl.Figure figure in figures)

{

figure.Draw(e);

}

}

public void panelInvalidate()

{

panel.Invalidate();

}

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

{

if(e.Button == MouseButtons.Left)

{

foreach (fl.Figure figure in figures)

{

figure.IsSelected = false;

}

figures.Reverse();

foreach (fl.Figure figure in figures)

{

if (figure.IsPointIn(e.X, e.Y))

{

figure.IsSelected = true;

break;

}

}

figures.Reverse();

}

else if (e.Button == MouseButtons.Right)

{

AddFigure form = new AddFigure(this, e.X, e.Y);

form.Show();

}

}

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

{

fl.Figure selectedFigure = null;

foreach(fl.Figure figure in figures)

{

if (figure.IsSelected)

{

selectedFigure = figure;

break;

}

}

if(selectedFigure != null)

{

switch (e.KeyData)

{

case Keys.Right:

if(selectedFigure.X\_center < panel.Width - 10)

selectedFigure.Move(10, 0);

break;

case Keys.Left:

if (selectedFigure.X\_center > 10)

selectedFigure.Move(-10, 0);

break;

case Keys.Down:

if (selectedFigure.Y\_center < panel.Height - 10)

selectedFigure.Move(0, 10);

break;

case Keys.Up:

if (selectedFigure.Y\_center > 10)

selectedFigure.Move(0, -10);

break;

}

panelInvalidate();

}

}

delegate void DoubleClickOnFigureHandler(object sender, ChangeFigureEventArgs e);

event DoubleClickOnFigureHandler EditFigureEvent;

void EditFigure(object sender, ChangeFigureEventArgs e)

{

figures.Remove(e.Figure);

AddFigure form = new AddFigure(this, e.X, e.Y);

form.Show();

panelInvalidate();

}

private void panel\_MouseDoubleClick(object sender, MouseEventArgs e)

{

fl.Figure figureToEdit = null;

foreach (fl.Figure figure in figures)

{

if (figure.IsPointIn(e.X, e.Y))

{

figureToEdit = figure;

break;

}

}

if (figureToEdit != null)

{

EditFigureEvent.Invoke(sender, new ChangeFigureEventArgs(figureToEdit, e.X, e.Y));

}

}

private Brush PickBrush()

{

Brush result = Brushes.Transparent;

Random rnd = new Random();

Type brushesType = typeof(Brushes);

PropertyInfo[] properties = brushesType.GetProperties();

int random = rnd.Next(properties.Length);

result = (Brush)properties[random].GetValue(null, null);

return result;

}

fl.Figure prevSelected = null;

private void panel\_MouseMove(object sender, MouseEventArgs e)

{

fl.Figure selected = null;

foreach (fl.Figure figure in figures)

{

if (figure.IsPointIn(e.X, e.Y))

{

selected = figure;

break;

}

}

if(selected != null && !selected.Equals(prevSelected))

{

selected.Brush = PickBrush();

prevSelected = selected;

panelInvalidate();

}

}

}

}