Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 3.2P: Drawing Program - A Drawing Class

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/04/15 08:54

Tutor:
Jai Cornes

April 15, 2022

```
using System;
using SplashKitSDK;



namespace ShapeDrawer
{
    public class Program
    {
        public static void Main()
        {

            new Window("Shape Drawer", 800, 600);
            //_ = new ShapeDrawer();
            Drawing myDrawing = new Drawing();

            do
            {
                SplashKit.ProcessEvents();
                //SplashKit.ClearScreen();

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape myShape = new Shape();

                    myShape.X = SplashKit.MouseX();
                    myShape.Y = SplashKit.MouseY();


                    myDrawing.AddShapes(myShape);
                }




                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    myDrawing.SelectShapesAt(SplashKit.MousePosition());
                }




                if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
                ↪  (SplashKit.KeyTyped(KeyCode.DeleteKey)))
                {
                    foreach (Shape Shape in myDrawing.SelectedShapes)
                    {
                        myDrawing.RemoveShapes(Shape);
                    }
                }

                myDrawing.Draw();
```

```
53
54                  //SplashKit.ClearScreen();
55                  SplashKit.RefreshScreen();
56              } while (!SplashKit.WindowCloseRequested("Shape Drawer"));
57          }
58      }
59  }
```

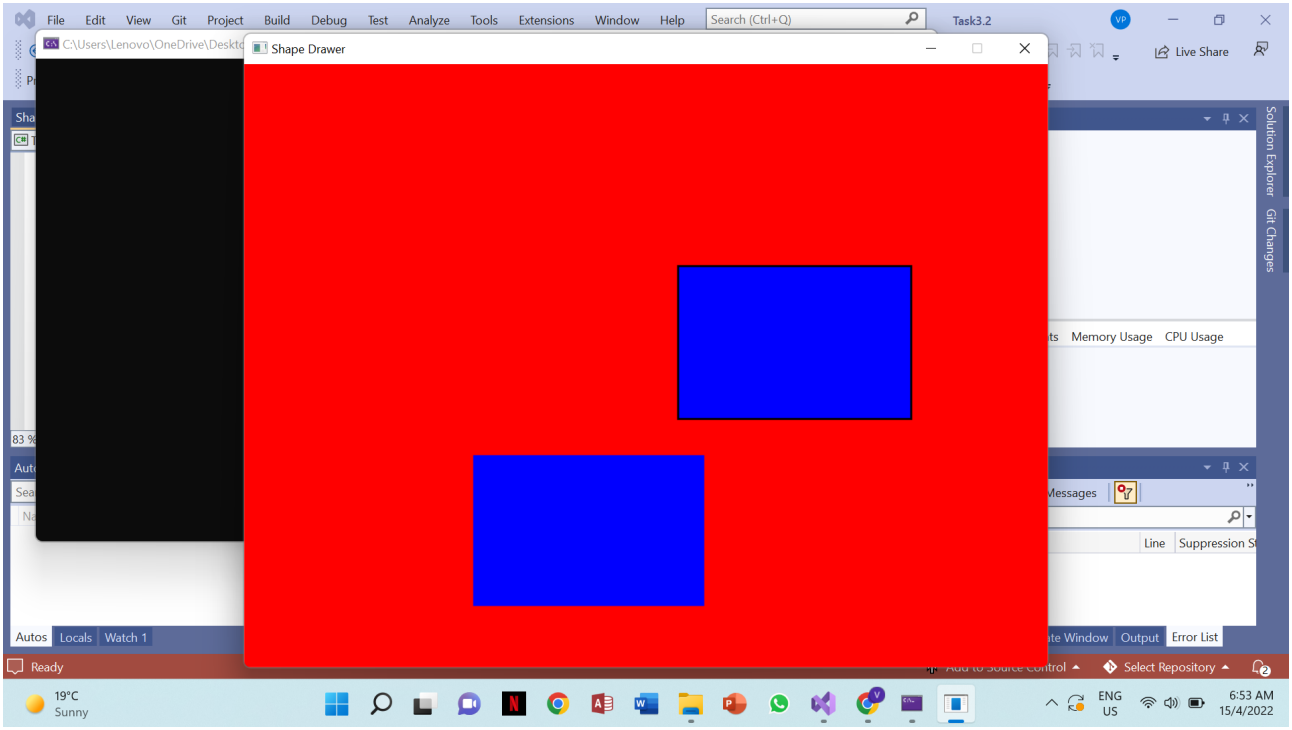
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;



//using System.Collections.Generic.List;

namespace ShapeDrawer
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;

        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> NewShapeSelected = new List<Shape>();

                foreach (Shape shape in _shapes)
                {
                    if (shape.Selected)
                    {
                        NewShapeSelected.Add(shape);
                    }
                }
                return NewShapeSelected;
            }

        }


        public Drawing() : this(Color.Red)
        { }
            //foreach (Shape shape in _shapes)
            //{
              //  shape.Draw();
            //}
        //}



        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;

        }
```

```
54
55
56          public int ShapeCount
57          {
58              get
59              {
60                  return _shapes.Count;
61              }
62          }
63
64
65          public void Draw()
66          {
67              SplashKit.ProcessEvents();
68              //SplashKit.ClearScreen(_background);
69              SplashKit.FillRectangle(_background, 0, 0, SplashKit.ScreenWidth(),
                ↪  SplashKit.ScreenHeight());
70
71              foreach (Shape shape in _shapes) shape.Draw();
72
73          }
74
75
76          public void AddShapes(Shape shape)
77          {
78              _shapes.Add(shape);
79          }
80
81
82          public Color Background
83          {
84              get
85              {
86                  return Background;
87              }
88              set
89              {
90                  Background = value;
91              }
92          }
93
94
95          public void SelectShapesAt(Point2D pt)
96          {
97              foreach (Shape Shapes in _shapes)
98              {
99                  if (Shapes.IsAt(pt))
100                 {
101                     Shapes.Selected = true;
102                 }
103                 else
104                 {
105                     Shapes.Selected = false;
```

```
106                 }
107             }
108         }
109 
110             public void RemoveShapes(Shape shape)
111         {
112             _shapes.Remove(shape);
113         }
114 
115 
116         }
117 
118 
119 
120 
121 
122 
123 
124 }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Shape
    {
        private Color _color;
        private float _x, _y;
        private int _width, _height;
        private bool _selected;

        public Shape() //contructor can only return the reference of the obj
        {
            _color = Color.Blue;
            _x = 0;
            _y = 0;
            _width = 230;
            _height = 150;
        }

        public void Draw()
        {
            if (Selected)
            {
                DrawOutline();
            }

            SplashKit.FillRectangle(_color, _x, _y, _width, _height);
        }

        public void DrawOutline()
        {
            SplashKit.FillRectangle(Color.Black, _x - 2, _y - 2, _width + 4,
            ↪  _height + 4);
        }

        public float X
        {
            get { return _x; }

            set { _x = value; }
        }


        public Color Color
        {
            get { return _color; }
            set { _color = value; }
```

```
53
54          }
55
56
57          public float Y
58          {
59              get { return _y; }
60
61              set { _y = value; }
62          }
63
64          public int Height
65          {
66              get { return _height; }
67              set { _height = value; }
68          }
69          public int Width
70          {
71              get { return _width; }
72              set { _width = value; }
73          }
74
75          public bool Selected
76          {
77              get { return _selected; }
78              set { _selected = value; }
79          }
80
81          public bool IsAt(Point2D point)
82          {
83              if (point.X >= _x && point.X <= _x + _width && point.Y >= _y &&
                ↪  +point.Y <= _y + _height)
84              {
85                  return true;
86              }
87              return false;
88          }
89      }
90
91
92  }
```


Shape Drawer
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Task3.2
Live Share
Memory Usage
CPU Usage
Solution Explorer
Git Changes
Autos
Locals
Watch 1
Error List
Ready
Select Repository
19°C
Sunny
ENG
US
6:53 AM
15/4/2022