Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Multiple Shape Kinds

PDF generated at 20:01 on Tuesday 26th September, 2023

```
using SplashKitSDK;
using System.Runtime.Intrinsics.X86;

namespace ShapeDrawer
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }
        public static void Main()
        {

            ShapeKind kindToAdd = ShapeKind.Circle;

            Drawing _drawing = new Drawing();


            Window window = new Window("Shape Drawer", 800, 600);
            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();
                _drawing.Draw();

                ///
                if (SplashKit.KeyDown(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
                }
                else if (SplashKit.KeyDown(KeyCode.CKey)) { kindToAdd =
↪   ShapeKind.Circle; }
                else if (SplashKit.KeyDown(KeyCode.LKey)) { kindToAdd =
↪   ShapeKind.Line; }

                    ///
                    if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape newShape;

                    if (kindToAdd == ShapeKind.Circle)
                    {
                        MyCircle newCircle = new MyCircle();
                        newShape = newCircle;
                    }
                    else if (kindToAdd == ShapeKind.Line)
                    {
                        MyLine newLine = new MyLine();
                        newShape = newLine;
                        newLine.Y2 = SplashKit.MouseY();
                    }
                    else
                    {
                        MyRectangle newRect = new MyRectangle();
                        newShape = newRect;
                    }
                    newShape.X = SplashKit.MouseX();
                    newShape.Y = SplashKit.MouseY();
                    _drawing.AddShape(newShape);

                }

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


                if (SplashKit.KeyTyped(KeyCode.DeleteKey))
                    {
                    foreach (Shape s in _drawing.SelectedShapes)
                    {
                        _drawing.RemoveShape(s);
                    }
                }




                    if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                {
                    _drawing.Background = SplashKit.RandomRGBColor(255);
                }



                SplashKit.RefreshScreen();


            } while (!window.CloseRequested);
        }

    }


}
```

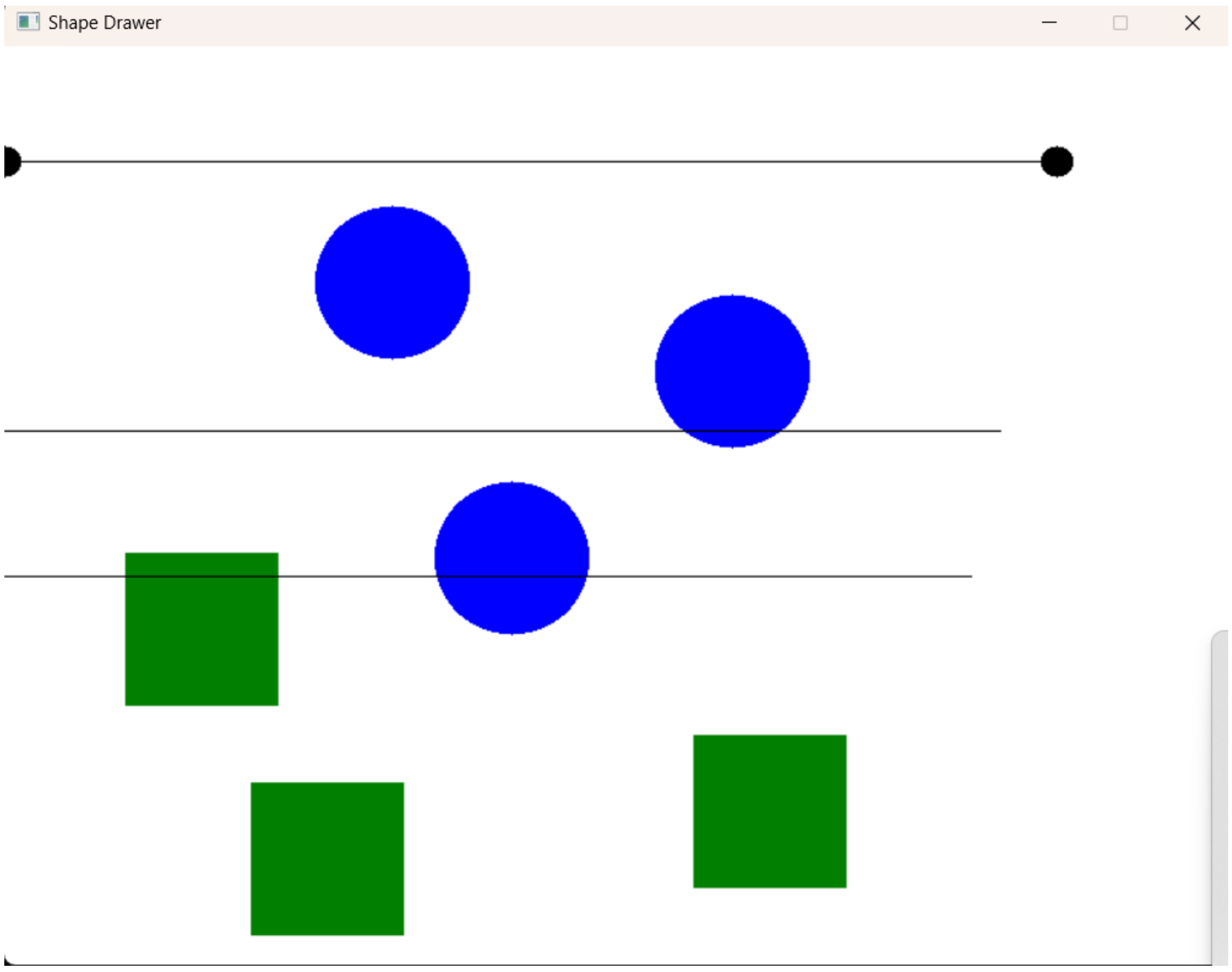
```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ShapeDrawer
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;


        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;
        }
        public Drawing() :this(Color.White) { }

        int ShapeCount { get { return _shapes.Count; } }
        public Color Background { get { return _background; } set { _background =
 ↪  value; } }

        public void Draw()
            {
            SplashKit.ClearScreen(_background);

            foreach (Shape shape in _shapes) { }

            foreach (Shape i in _shapes)
            {
                i.Draw();
            }


            }
        public void SelectShapesAt( Point2D pt)
        {
            foreach (Shape i in _shapes)
            {
               i.Selected = i.IsAt(pt);

            }

        }


        public List<Shape> SelectedShapes
        {


            get
            {
                List<Shape> result = new List<Shape>();
                foreach (Shape i in _shapes)
                {
                    if (i.Selected == true)
                        result.Add(i);
                }
                return result;
            }
        }




        public void AddShape(Shape s)
        {
            _shapes.Add(s);
        }

        public void RemoveShape(Shape s)
        {
            _shapes.Remove(s);
        }

    }
}

/*
 * using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
*/
```

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

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

        public Shape()
        {
            _color = Color.Yellow;
            _x = 0;
            _y = 0;
            _selected = false;
        }

        public Color Color
        {
            get { return _color; }
            set { _color = value; }
        }
        public float X
        {
            get { return _x; }
            set { _x = value; }
        }
        public float Y
        {
            get { return _y; }
            set { _y = value; }
        }


        public abstract void Draw();

        public abstract void DrawOutline();

        public abstract bool IsAt(Point2D pt);


        public bool Selected
        {
            get { return _selected; }
            set { _selected = value; }
        }



    }
}
```

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

namespace ShapeDrawer
{
    public class MyRectangle : Shape
    {
        private int _width, _height;


        public MyRectangle() :this(Color.Green,0,0,100,100)
        {
        }
        public MyRectangle(Color color, float x, float y, int width, int height)
        {
            Color = color;
            X = x;
            Y = y;
            Width = width;
            Height = height;
        }


        public int Width { get { return _width; } set { _width = value; } }
        public int Height { get { return _height; } set { _height = value; } }

        public override void Draw()
        {
            if (Selected) { DrawOutline(); }
            SplashKit.FillRectangle( Color, X, Y, Width, Height);
        }
        public override void DrawOutline()
        {
            SplashKit.FillRectangle(Color.Black, X-2, Y-2, Width+2, Height + 2);
        }

        public override bool IsAt(Point2D pt)
        {
            if (pt.X < X + Width && pt.X > X)
                {
                 if (pt.Y < Y + Height && pt.Y > Y)
                    {
                    return true;
                    }
                }
            return false;
        }

    }
}
```

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

namespace ShapeDrawer
{
    public class MyCircle : Shape
    {
        int _radius;

        public int Radius { get { return _radius; } set { _radius = value; } }

        public MyCircle() :this(50,Color.Blue,0,0)
        {

        }

        public MyCircle(int radius, Color color, float x, float y)
        {
            Color = color;
            _radius = radius;
            X= x;
            Y= y;
        }

        public override void Draw()
        {
            if (Selected)
                DrawOutline();
            SplashKit.FillCircle(Color, X, Y, _radius);
        }
        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color.Black, X-2,Y-2, _radius+2);
        }
        public override bool IsAt(Point2D pt)
        {
            if (pt.X < X + _radius/2 && pt.X > X - _radius / 2)
            {
                if (pt.Y < Y + _radius/2 && pt.Y > Y - _radius / 2)
                {
                    return true;
                }
            }
            return false;
        }

    }
}
```

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

namespace ShapeDrawer
{
    public class MyLine :Shape
    {
        private float _x2;
        private float _y2;


        public float Y2 { get { return _y2; } set { _y2 = value; } }



        public MyLine() : this(Color.Black, 0, 0,0,0)
        {

        }

        public MyLine( Color color, float x, float y,float x2, float y2)
        {
            X=x; Y=y;
            _x2=x2;
            _y2=y2;

            Color = color;

        }

        public override void Draw()
        {
            if (Selected) { DrawOutline(); }
            SplashKit.DrawLine(Color, X, Y, _x2, _y2);
        }
        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color, X, Y, 10);
            SplashKit.FillCircle(Color, _x2, _y2, 10);
        }
        public override bool IsAt(Point2D pt)
        {
            if (pt.X < X)
            {
                if (pt.Y < Y+5 && pt.Y > Y - 5)
                    return true;
            }
            return false;
            }

        }
}
```


Shape Drawer