COS20007: Object Oriented Programming

Pass Task 4.1: Drawing Program — Multiple Shape Kinds

Show Wai Yan/105293041

Shape.cs

```
using SplashKitSDK;

namespace ShapeDrawer
{
    public abstract class Shape
    {
        // Fields
        private Color _color;
        private float _x;
        private float _y;
        private bool _selected = false;

        // Constructors
        public Shape() : this(Color.Yellow)
        {
        }

        public Shape(Color color)
        {
            Color = color;
            _x = 0.0f; _y = 0.0f;
        }

        // Properties
        public float X
        {
            get { return _x; }
            set { _x = value; }
        }
        public float Y
        {
            get { return _y; }
            set { _y = value; }
        }
        public Color Color
        {
            get { return _color; }
            set { _color = value; }
        }
        public bool Selected
        {
            get { return this._selected; }
            set { _selected = value; }
        }


        // Methods
        public abstract void Draw();

        public abstract bool IsAt(Point2D pt);

        public abstract void DrawOutline();
    }
}
```

MyRectangle.cs

```
using SplashKitSDK;

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

        // Constructor
        public MyRectangle() : this(Color.Green, 0.0f, 0.9f, 100 + 41, 100 + 41)
        {

        }
        public MyRectangle(Color color, float x, float y, int width, int height) : base(color)
        {
            X = x;
            Y = y;
            Width = width;
            Height = height;
        }

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

        // Methods
        public override void Draw()
        {
            if (Selected) this.DrawOutline();
            SplashKit.FillRectangle(Color, X, Y, Width, Height);
        }

        public override void DrawOutline()
        {
            int outlineThickness = 6; //5+1
            SplashKit.FillRectangle(Color.Black, X-outlineThickness, Y-outlineThickness, Width+2*outlineThickness,
Height+2*outlineThickness);
        }

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


    }
}
```

MyCircle.cs

```
using SplashKitSDK;

namespace ShapeDrawer
{
    public class MyCircle : Shape
    {
        // Fields
        private int _radius;

        // Constructor
        public MyCircle() : this(Color.Blue, 100+41, 100+41, 50+41)
        {
        }
        public MyCircle(Color color, int x, int y, int radius) : base(color)
        {
            X = x;
            Y = y;
            Radius = radius;
        }

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

        // Methods
        public override void Draw()
        {
            if (Selected) DrawOutline();
            SplashKit.FillCircle(Color, X, Y, Radius);
        }

        public override void DrawOutline()
        {
            int outlineThickness = 7; //5+2
            SplashKit.FillCircle(Color.Black, X, Y, Radius + outlineThickness);
        }

        public override bool IsAt(Point2D pt)
        {
            // By Distance Formula
            // = √(x2-x1)^2 + (y2-y1)^2
            // And then we get the distnace between mouse click and circle area
            // If that distance is smaller than and equal the circle's radius
            // of course, it is inside the circle
            // return Math.Sqrt(Math.Pow(pt.X - X, 2) + Math.Pow(pt.Y - Y, 2)) <= Radius;

            return SplashKit.PointInCircle(pt,
                    new Circle()
                    {
                        Center =
                        new Point2D() { X = this.X, Y = this.Y },
                        Radius = this.Radius
                    });
        }
    }
}
```

MyLine.cs

```
using SplashKitSDK;

namespace ShapeDrawer
{
    public class MyLine : Shape
    {
        // Fields
        private float _endX;
        private float _endY;

        // Constructor
        public MyLine() : this(Color.Red, SplashKit.MouseX(), SplashKit.MouseY(), SplashKit.MouseX()+new Random().Next(-150, 150), new
Random().Next(0, 601))
        {

        }

        public MyLine(Color color, float startX, float startY, float endX, float endY) : base(color)
        {
            X = startX;
            Y = startY;
            EndX = endX;
            EndY = endY;
        }

        // Properties
        public float EndX
        {
            get { return _endX; }
            set { _endX = value; }
        }
        public float EndY
        {
            get { return _endY; }
            set { _endY = value; }
        }

        // Methods
        public override void Draw()
        {
            if (Selected) DrawOutline();
            SplashKit.DrawLine(Color, X, Y, EndX, EndY);
        }

        public override void DrawOutline()
        {
            int circleRadius = 5;
            SplashKit.FillCircle(Color.Black, X, Y, circleRadius);
            SplashKit.FillCircle(Color.Black, EndX, EndY, circleRadius);
        }

        public override bool IsAt(Point2D pt)
        {
            return SplashKit.PointOnLine(pt,
                new Line()
                {
                    StartPoint = new Point2D() {X = this.X, Y = this.Y},
                    EndPoint = new Point2D() {X = this.EndX, Y = this.EndY},
                });
        }
    }
}
```

Program.cs

```
using System;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }
        public static void Main()
        {
            Window window = new Window("Shape Drawer", 800, 600);
            Drawing myDrawing = new Drawing();

            // ShapeKind Variable
            ShapeKind kindToAdd = ShapeKind.Circle; // First initialization
            int XLineDraw = 1; // Times of line can draw after typed L key

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

                if (SplashKit.KeyTyped(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
                }
                if (SplashKit.KeyTyped(KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle;
                }
                if (SplashKit.KeyTyped(KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line;
                    XLineDraw = 1;
                }

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

                    switch (kindToAdd)
                    {
                        case ShapeKind.Circle:
                            newShape = new MyCircle();
                            break;

                        case ShapeKind.Line:
                            if (XLineDraw == 0) continue;
                            newShape = new MyLine();
                            --XLineDraw;
                            break;

                        default:
                            newShape = new MyRectangle();
                            break;
                    }

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

                    myDrawing.AddShape(newShape);
                }

                if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                {
                    myDrawing.Background = SplashKit.RandomColor();
                }

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

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

                myDrawing.Draw();

                SplashKit.RefreshScreen();
            } while (!window.CloseRequested);
        }
    }
}
```

Screenshot of the Splashkit Window showing your drawing