Swinburne University of Technology

COS20007 Object Oriented Programming

4.1P - Drawing Program - Multiple Shape Kinds

PDF generated at 12:22 on Tuesday 4th April, 2023

```
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(Color.White);
            ShapeKind kindToAdd = ShapeKind.Circle;

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

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

                    if (kindToAdd == ShapeKind.Circle)
                    {
                        newShape = new MyCircle();
                    }
                    else if (kindToAdd == ShapeKind.Rectangle)
                    {
                        newShape = new MyRectangle();
                    }
                    else
                    {
                        newShape = new MyLine();
                    }

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

                    myDrawing.AddShape(newShape);
                }

                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;
```

```
54                  }
55
56                  if (SplashKit.KeyDown(KeyCode.SpaceKey))
57                  {
58                      myDrawing.Background = SplashKit.RandomRGBColor(255);
59                  }
60
61                  if (SplashKit.MouseClicked(MouseButton.RightButton))
62                  {
63                      myDrawing.SelectedShapesAt(SplashKit.MousePosition());
64                  }
65
66                  if ((SplashKit.KeyTyped(KeyCode.BackspaceKey)) ||
    ↪  (SplashKit.KeyTyped(KeyCode.DeleteKey)))
67                  {
68                      foreach (Shape shape in myDrawing.SelectedShapes)
69                          myDrawing.RemoveShape(shape);
70                  }
71                  myDrawing.Draw();
72
73                  SplashKit.RefreshScreen();
74
75                  } while (!window.CloseRequested);
76          }
77      }
78  }
```

```
using SplashKitSDK;
using System;
using System.Collections.Generic;

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)
        {
        }

        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> selectedShapes = new List<Shape>();
                foreach (Shape shape in _shapes)
                {
                    if (shape.Selected)
                        selectedShapes.Add(shape);
                }
                return selectedShapes;
            }
        }
        public int ShapeCount
        {
            get
            {
                return _shapes.Count;
            }
        }
        public Color Background
        {
            get
            {
                return _background;
            }
            set
            {
                _background = value;
            }
        }

        public void Draw()
```

```
54          {
55              SplashKit.ClearScreen(_background);
56
57              foreach (Shape shape in _shapes)
58                  shape.Draw();
59          }
60          public void SelectedShapesAt(Point2D point)
61          {
62              foreach (Shape shape in _shapes)
63              {
64                  if (shape.IsAt(point))
65                      shape.Selected = true;
66                  else
67                      shape.Selected = false;
68              }
69          }
70
71          public void AddShape(Shape shape)
72          {
73              _shapes.Add(shape);
74          }
75          public void RemoveShape(Shape shape)
76          {
77              _shapes.Remove(shape);
78          }
79      }
80  }
```

```
using SplashKitSDK;
using System;

namespace ShapeDrawer
{
    public abstract class Shape
    {
        private Color _color;

        private float _x;
        private float _y;

        private bool _selected;
        public Shape(Color color, float x, float y)
        {
            _color = color;

            _x = x;
            _y = y;
        }
        public Shape() : this(SplashKit.RandomRGBColor(255), 0, 0)
        {
        }

        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
```

```
54              {
55                  _y = value;
56              }
57          }
58
59          public abstract void Draw();
60
61          public abstract bool IsAt(Point2D point2D);
62
63          public bool Selected
64          {
65              get
66              {
67                  return _selected;
68              }
69              set
70              {
71                  _selected = value;
72              }
73          }
74
75          public abstract void DrawOutline();
76      }
77  }
```

```
using SplashKitSDK;
using System;

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

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

        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);
        }

```

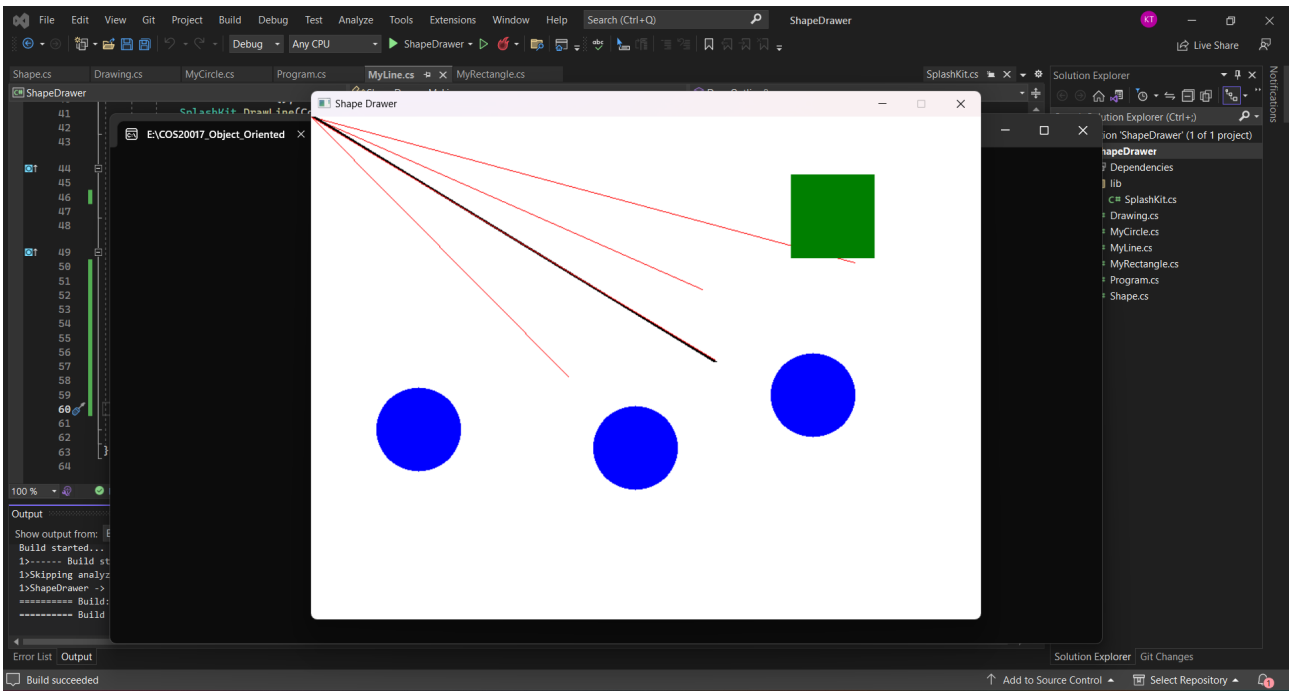
```
54          public override bool IsAt(Point2D point2D)
55          {
56              if (point2D.X >= X && point2D.X < X + Width && point2D.Y >= Y &&
    ↪  point2D.Y < Y + Height)
57                  return true;
58              else
59                  return false;
60          }
61
62          public override void DrawOutline()
63          {
64              SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, Width + 4, Height +
    ↪  4);
65          }
66      }
67  }
```

```
using SplashKitSDK;
using System;

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

        public MyCircle(Color color, float x, float y, int radius)
        {
            Color = color;
            X = x;
            Y = y;
            Radius = radius;
        }
        public MyCircle() : this(Color.Blue, 0, 0, 50)
        {
        }

        public int Radius
        {
            get
            {
                return _radius;
            }
            set
            {
                _radius = value;
            }
        }
        public override void Draw()
        {
            if (Selected)
                DrawOutline();
            SplashKit.FillCircle(Color, X, Y, _radius);
        }

        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color.Black, X, Y, _radius + 2);
        }

        public override bool IsAt(Point2D point2D)
        {
            float a = (float)(point2D.X - X);
            float b = (float)(point2D.Y - Y);

            if (Math.Sqrt(a * a + b * b) < _radius)
                return true;
            else
                return false;
        }
```

```
54      }
55  }
```

```
using SplashKitSDK;
using System;

namespace ShapeDrawer
{
    public class MyLine : Shape
    {
        // Endpoint(_x2, _y2)
        private float _x2;
        private float _y2;

        public MyLine(Color color, float startX, float startY, float endX, float
↪   endY)
        {
            Color = color;
            X = startX;
            Y = startY;
            X2 = endX;
            Y2 = endY;
        }

        public MyLine() : this(Color.Red, 100, 100, 0, 0)
        {
        }

        public float X2
        {
            get
            {
                return _x2;
            }
            set
            {
                _x2 = value;
            }
        }

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

        public override void Draw()
        {
            if (Selected)
                DrawOutline();
```

```
53
54              // By default, the start point (X, Y) will be the coordinates of the
    ↪  mouse when clicking, and the end point (X2, Y2) will be (0, 0)
55              SplashKit.DrawLine(Color, X, Y, X2, Y2);
56          }
57
58          public override bool IsAt(Point2D point2D)
59          {
60              return SplashKit.PointOnLine(point2D, SplashKit.LineFrom(X, Y, X2, Y2));
61          }
62
63          public override void DrawOutline()
64          {
65              Point2D[] points = new Point2D[4];
66              points[0] = SplashKit.PointAt(X - 2, Y - 2);
67              points[1] = SplashKit.PointAt(X + 2, Y - 2);
68              points[2] = SplashKit.PointAt(X2 - 2, Y2 + 2);
69              points[3] = SplashKit.PointAt(X2 + 2, Y2 + 2);
70
71              Quad outlineQuad = new Quad();
72              outlineQuad.Points = points;
73
74              SplashKit.FillQuad(Color.Black, outlineQuad);
75          }
76      }
77  }
```


Shape Drawer
E:\COS20017_Object_Oriented