Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Multiple Shape Kinds

PDF generated at 19:10 on Monday 18th September, 2023

```
using System;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }

        public static void Main()
        {
            // Create a window for the shape drawer
            Window window = new Window("Shape Drawer", 800, 600);
            Drawing myDrawing = new Drawing(Color.White);
            ShapeKind kindToAdd = ShapeKind.Circle;

            do
            {
                // Processes events from user input
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();

                Point2D pt = SplashKit.MousePosition();

                // Check for keyboard input to change the kind of shape to add
                if (SplashKit.KeyTyped(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
                }
                else if (SplashKit.KeyTyped(KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle;
                }
                else if (SplashKit.KeyTyped(KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line;
                }

                // Check for left mouse button click to add a new shape
                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape newShape;

                    // Create a new shape based on the selected kind
                    if (kindToAdd == ShapeKind.Circle)
                    {
                        newShape = new MyCircle();
                    }
```

```
54                      else if (kindToAdd == ShapeKind.Rectangle)
55                      {
56                          newShape = new MyRectangle();
57                      }
58                      else
59                      {
60                          newShape = new MyLine();
61                      }
62
63                      newShape.X = SplashKit.MouseX();
64                      newShape.Y = SplashKit.MouseY();
65
66                      // Add the new shape to the drawing
67                      myDrawing.AddShape(newShape);
68                  }
69
70                  // Select shapes at the mouse position when right mouse button is
    ↪  clicked
71                  if (SplashKit.MouseClicked(MouseButton.RightButton))
72                  {
73                      myDrawing.SelectShapesAt(pt);
74                  }
75
76                  // Change the background color to a random color when Space key is
    ↪  typed
77                  if (SplashKit.KeyTyped(KeyCode.SpaceKey))
78                  {
79                      myDrawing.Background = Color.RandomRGB(255);
80                  }
81
82                  // Check for backspace key to remove selected shapes
83                  if (SplashKit.KeyTyped(KeyCode.BackspaceKey))
84                  {
85                      List<Shape> selectedShapes = myDrawing.SelectedShapes;
86                      foreach (Shape shapeToRemove in selectedShapes)
87                      {
88                          myDrawing.RemoveShape(shapeToRemove);
89                      }
90                  }
91
92                  myDrawing.Draw();
93                  SplashKit.RefreshScreen();
94              } while (!window.CloseRequested);
95          }
96      }
97  }
```

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

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

        // public properties
        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;
        }
        // A default constructor
        public Drawing() : this(Color.White)
        {
        }
        // Readonly property to access the shape count
        public int ShapeCount
        {
            get { return _shapes.Count; }
        }
        public Color Background
        {
            get { return _background; }
            set { _background = value; }
        }
        // Select shapes
        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> selectedShapes = new List<Shape>();
                foreach (Shape shape in _shapes)
                {
                    if (shape.Selected)
                    {
                        selectedShapes.Add(shape);
                    }
                }
                return selectedShapes;
            }
        }
        // Tells SplashKit to clear the screen to the background color
        public void Draw()
        {
            SplashKit.ClearScreen(_background);
            foreach (Shape shape in _shapes)
```

```
54              {
55                  shape.Draw();
56              }
57          }
58          public void SelectShapesAt(Point2D pt)
59          {
60              foreach (Shape shape in _shapes)
61              {
62                  shape.Selected = shape.IsAt(pt);
63              }
64          }
65          // Method to add a shape to the list of shapes
66          public void AddShape(Shape shape)
67          {
68              _shapes.Add(shape);
69          }
70          // Removes the shapes
71          public void RemoveShape(Shape shape)
72          {
73              _shapes.Remove(shape);
74          }
75
76      }
77  }
```

```
using System;
using SplashKitSDK;

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

        // Constructor to initialize the shape's properties.
        public Shape(Color color)
        {
            _color = color;
            _x = 0;
            _y = 0;
        }

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

        // Property to get or set the X-coordinate of the shape
        public float X
        {
            get { return _x; }
            set { _x = value; }
        }

        // Property to get or set the Y-coordinate of the shape
        public float Y
        {
            get { return _y; }
            set { _y = value; }
        }

        // Property to get or set the color of the shape
        public Color Color
        {
            get { return _color; }
            set { _color = value; }
        }

        // Property to get or set whether the shape is selected
        public bool Selected
        {
            get { return _selected; }
            set { _selected = value; }
        }

        // Method to draw the shape on the screen using SplashKit
```

```
54          public abstract void Draw();
55
56          // Method to draw a black outline when the shape is selected
57          public abstract void DrawOutline();
58
59          // Method to check if a given point is within the boundaries of the shape
60          public abstract bool IsAt(Point2D pt);
61      }
62  }
```

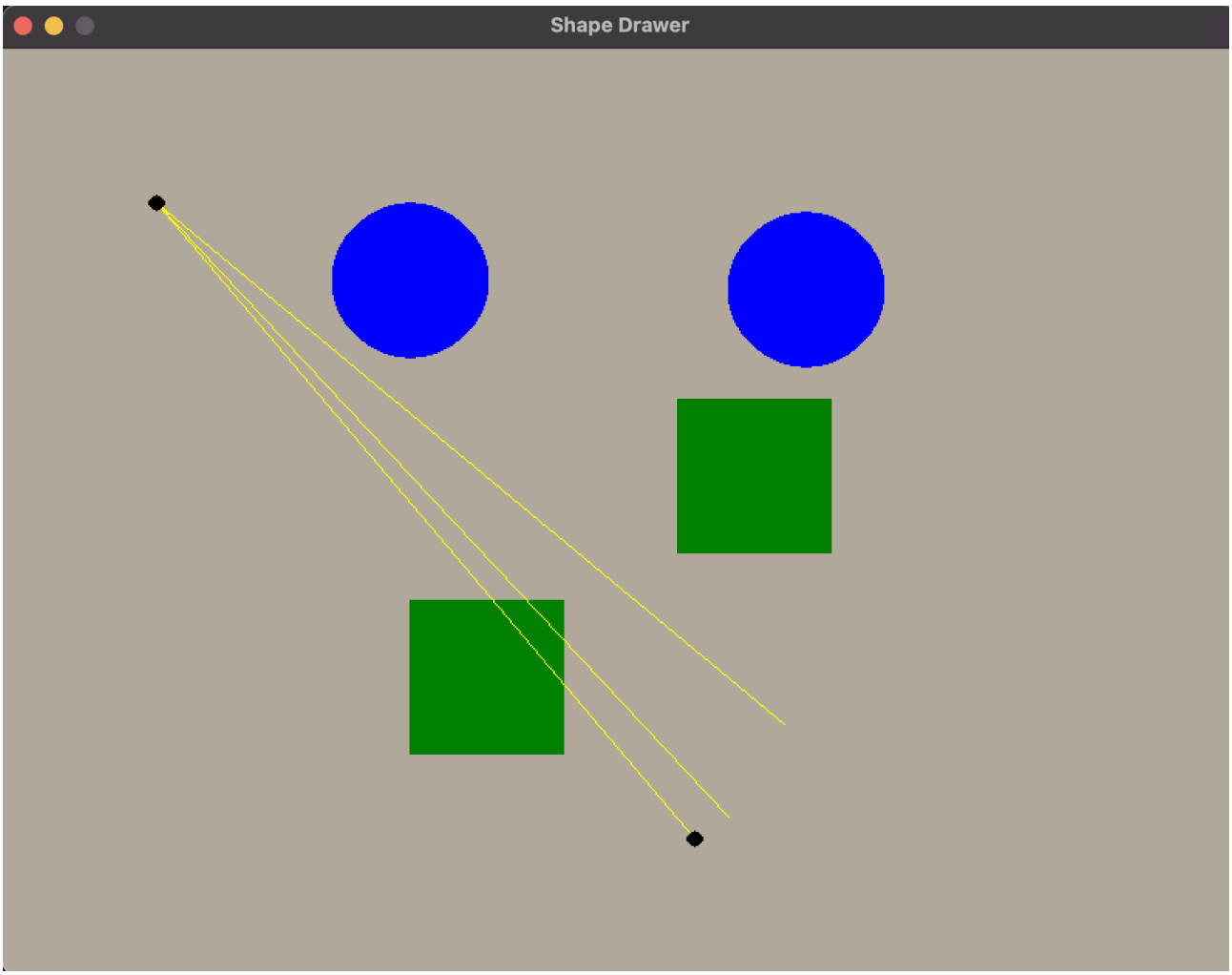
```
using System;
using SplashKitSDK;

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

        // Constructor to create a rectangle with specified color, position, width, and height
        public MyRectangle(Color color, float x, float y, int width, int height) : base(color)
        {
            X = x;
            Y = y;
            Width = width;
            Height = height;
        }

        // Default constructor to create a green rectangle at (0, 0) with width 100 and height 100
        public MyRectangle() : this(Color.Green, 0, 0, 100, 100)
        {
        }

        // Property to get or set the width of the rectangle
        public int Width
        {
            get { return _width; }
            set { _width = value; }
        }

        // Property to get or set the height of the rectangle
        public int Height
        {
            get { return _height; }
            set { _height = value; }
        }

        // Method to draw the rectangle on the screen
        public override void Draw()
        {
            SplashKit.FillRectangle(Color, X, Y, Width, Height);
            // Call the DrawOutline method if the rectangle is selected
            if (Selected) DrawOutline();
        }

        // Method to draw a black outline around the rectangle when it's selected
        public override void DrawOutline()
        {
            SplashKit.DrawRectangle(Color.Black, X - 2, Y - 2, Width + 4, Height + 4);
```

```
51          }
52
53          // Method to check if a given point is within the boundaries of the rectangle
54          public override bool IsAt(Point2D pt)
55          {
56              // Check if the point is within the rectangle's boundaries
57              return (X < pt.X && pt.X < (X + Width) && Y < pt.Y && pt.Y < (Y +
    ↪  Height));
58          }
59      }
60  }
```

```
using System;
using SplashKitSDK;

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

        // Constructor to create a circle with specified color, position, and radius
        public MyCircle(Color color, float x, float y, int radius) : base(color)
        {
            X = x;
            Y = y;
            Radius = radius;
        }

        // Default constructor to create a blue circle at (0, 0) with radius 50
        public MyCircle() : this(Color.Blue, 0, 0, 50)
        {
        }

        // Property to get or set the radius of the circle
        public int Radius
        {
            get { return _radius; }
            set { _radius = value; }
        }

        // Method to draw the circle on the screen
        public override void Draw()
        {
            SplashKit.FillCircle(Color, X, Y, Radius);
            // Call the DrawOutline method if the circle is selected
            if (Selected) DrawOutline();
        }

        // Method to draw a black outline around the circle when it's selected
        public override void DrawOutline()
        {
            SplashKit.DrawCircle(Color.Black, X, Y, Radius + 2);
        }

        // Method to check if a given point is within the boundaries of the circle
        public override bool IsAt(Point2D pt)
        {
            // Check if the point is within the circle's boundaries
            return (pt.X >= X - Radius && pt.X <= X + Radius && pt.Y >= Y - Radius &&
↪   pt.Y <= Y + Radius);
        }
    }
}
```

```
using ShapeDrawer;
using System;
using SplashKitSDK;

public class MyLine : Shape
{
    public MyLine(Color color, float x, float y, float endX, float endY) :
    ↪ base(color)
    {
        X = x; // Setting X and Y as one end of the line
        Y = y;
        _endX = endX;
        _endY = endY;
    }

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

    private float _endX;
    private float _endY;

    public float EndX
    {
        get { return _endX; }
        set { _endX = value; }
    }

    public float EndY
    {
        get { return _endY; }
        set { _endY = value; }
    }

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

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

    public override bool IsAt(Point2D pt)
    {
        const double Tolerance = 5.0;
        double distanceToStart = SplashKit.PointPointDistance(pt, new Point2D() { X =
    ↪ X, Y = Y });
        double distanceToEnd = SplashKit.PointPointDistance(pt, new Point2D() { X =
    ↪ EndX, Y = EndY });
```

```
51          return distanceToStart < Tolerance || distanceToEnd < Tolerance;
52      }
53  }
54
```


Shape Drawer