Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Multiple Shape Kinds

PDF generated at 22:43 on Thursday 14th September, 2023

```
using SplashKitSDK;
using System;

namespace ShapeDrawer
{
    // the main program class

    public class Program
    {
        // enumerations for the different shapes

        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }

        // the main entry point of the the program

        public static void Main()
        {
            // create a window for drawing

            Window window = new Window("Shape Drawer:                          , 800,
    ↪  600);

            // create a drawing object to manage shapes to draw

            Drawing mydrawing = new Drawing();

            // Initial the kind of shapes to add a cirlce

            ShapeKind kindToAdd = ShapeKind.Circle;

            do
            {
                //Process user input

                SplashKit.ProcessEvents();

                // clear the screen

                SplashKit.ClearScreen();


                // check if the left mouse button was clicked

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


```

```
53                  //Create a new shapes based on the selected kindToAdd
54
55                  if (kindToAdd == ShapeKind.Circle)
56                  {
57                      MyCircle newCirc = new MyCircle();
58                      newCirc.X = SplashKit.MouseX();
59                      newCirc.Y = SplashKit.MouseY();
60                      newShape = newCirc;
61
62                  }
63
64                  else if (kindToAdd == ShapeKind.Line)
65                  {
66                      MyLines newLine = new MyLines();
67                      newLine.X = SplashKit.MouseX();
68                      newLine.Y = SplashKit.MouseY();
69                      newShape = newLine;
70                  }
71                  else
72                  {
73                      MyRectangle newRec = new MyRectangle();
74                      newRec.X = SplashKit.MouseX();
75                      newRec.Y = SplashKit.MouseY();
76                      newShape = newRec;
77                  }
78
79                  //Add the shape to the drawing
80
81                  mydrawing.AddShape(newShape);
82              }
83
84
85              // check for key events to change the kind of shape to add
86
87              if (SplashKit.KeyReleased(KeyCode.RKey))
88              {
89                  kindToAdd = ShapeKind.Rectangle;
90              }
91              else if (SplashKit.KeyReleased(KeyCode.CKey))
92              {
93                  kindToAdd = ShapeKind.Circle;
94              }
95              else if (SplashKit.KeyReleased(KeyCode.LKey))
96              {
97                  kindToAdd = ShapeKind.Line;
98              }
99
100
101             // Change the background color if the space key is pressed
102
103             if (SplashKit.KeyDown(KeyCode.SpaceKey))
104             {
105                 mydrawing.Background = SplashKit.RandomRGBColor(255);
```

```
106                 }
107
108
109                 // Selected shapes at the current mouse position when the right mouse
    ↪  button is clicked
110
111                 if (SplashKit.MouseClicked(MouseButton.RightButton))
112                 {
113                     mydrawing.SelectShapesAt(SplashKit.MousePosition());
114                 }
115
116
117                 // Remove selected shapes when the delete key is pressed
118
119                 if (SplashKit.KeyDown(KeyCode.DeleteKey) ||
    ↪  SplashKit.KeyDown(KeyCode.BackspaceKey))
120                 {
121                     foreach (Shape shape in mydrawing.SelectedShapes)
122                     {
123                         mydrawing.RemoveShape(shape);
124                     }
125                 }
126
127
128                 // draw the shapes and refresh the screen
129
130                 mydrawing.Draw();
131                 SplashKit.RefreshScreen();
132
133                 // Continue the loop until the user close the window
134
135             } while (!window.CloseRequested);
136         }
137     }
138 }
```

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


namespace ShapeDrawer
{
    // represents a drawing
    public class Drawing
    {
        private readonly List<Shape> _shapes; // list to store shapes
        private Color _background; // background color of drawing


        // constructor to create a drawing with a specified background color
        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;
        }


        // constructor to create a drawing with a default white background
        public Drawing() : this(Color.White)
        {

        }


        // gets and sets the background color of the drawing
        public Color Background
        {
            get { return _background; }
            set { _background = value; }
        }


        // gets the number of shapes in the drawing
        public int ShapeCount
        {
            get { return _shapes.Count; }
        }


        // gets a list of selected shapes in the drawing

        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> result = new List<Shape>();
                foreach (Shape shapez in _shapes)
                {
```

```
                    if (shapez.Selected)
                    {
                        result.Add(shapez);
                    }
                }
                return result;
            }
        }


        // adds a shape to the drawing

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


        // clears the screen,sets the background color,and drawing all shapes in the drawing

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

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


        //selects shapes at a given point

        public void SelectShapesAt(Point2D pt)
        {
            foreach (Shape shapez in _shapes)
            {
                // check if shape is at point
                if (shapez.Selected = shapez.IsAt(pt))
                {
                    shapez.Selected = true;
                }
                else
                {
                    shapez.Selected = false;
                }
            }
        }


        // removes all selected shapes from the drawing

```

```
106         public void RemoveShape(Shape shapez)
107         {
108             _shapes.Remove(shapez);
109         }
110     }
111 }
```

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

namespace ShapeDrawer
{
    // The abstract base class for all shapes

    public abstract class Shape
    {
        //color the shape

        private Color _color;

        //X-coordinate of shape

        private float _x;

        //Y-coordinate of shape

        private float _y;

        //whether the shape is selected

        private bool _selected;


        // Default constructor for a shape, sets the color to yellow

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

        //Constructor for a shape with a specified color

        public Shape(Color color)
        {
            _color = color;
        }


        // Gets and sets the color of the shape

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


        //gets and sets the X-coordinate of the shape

        public float X
        {
            get { return _x; }
```

```
54              set { _x = value; }
55          }
56
57
58          // gets and sets the Y-coordinate of the shape
59
60          public float Y
61          {
62              get { return _y; }
63              set { _y = value; }
64          }
65
66          // gets and sets whether the shape is selected
67
68          public bool Selected
69          {
70              get { return _selected; }
71              set { _selected = value; }
72          }
73
74
75          // abstract method to draw the shape on the screen
76
77          public abstract void Draw();
78
79          // abstract method to draw the outline of the shape on the screen
80
81          public abstract void DrawOutline();
82
83          // abstract method for checking if a given point is inside the shape
84
85          public abstract bool IsAt(Point2D pt);
86      }
87  }
```

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



namespace ShapeDrawer
{
    // My rectangle class represents a rectangle

    public class MyRectangle : Shape
    {
        // Width and height of rectangle

        private int _width, _height;

        // constructor to create a rectangle with a specified width, height and color

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

        // constructor to create a rectangle with a specified width, height and color

        public MyRectangle() : this(Color.Green, 0, 0, 100, 100) { }

        // gets and sets the width of the rectangle

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


        // gets and sets the height of the rectangle

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

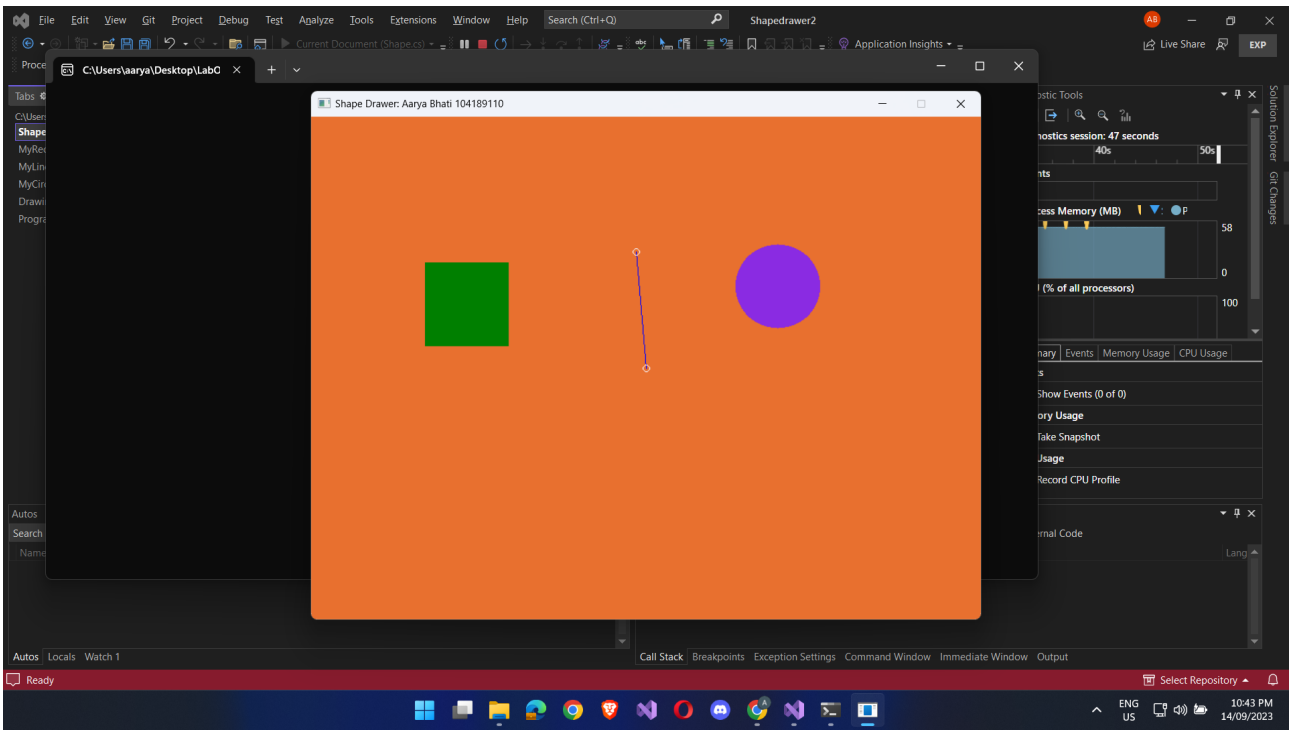
```
53
54          // draws the filled rectangle on the screen and outline if selected
55
56          public override void Draw()
57          {
58              // draw the filled rectangle on the screen
59
60              SplashKit.FillRectangle(Color, X, Y, Width, Height);
61
62              // if the rectangle is selected, draw the outline
63
64              if (Selected)
65              {
66                  DrawOutline();
67              }
68
69          }
70
71
72          // draws the outline of the rectangle on the screen
73
74          public override void DrawOutline()
75          {
76              // draw a sligly bigger black rectangle on the screen
77
78              SplashKit.DrawRectangle(Color.Black, X - 2, Y - 2, Width + 4, Height +
    ↪  4);
79          }
80
81
82          // checks if the rectangle is inside the rectangle
83
84          public override bool IsAt(Point2D mouseLocation)
85          {
86              if (X < mouseLocation.X && mouseLocation.X < (X + Width) && Y <
    ↪  mouseLocation.Y && mouseLocation.Y < (Y + Height))
87              {
88                  // if the point is within the rectangle, return true
89
90                  return true;
91              }
92              else
93              {
94                  // if the point is not within the rectangle, return false
95
96                  return false;
97              }
98          }
99      }
100 }
```

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


namespace ShapeDrawer
{
    // represents a circle

    public class MyCircle : Shape
    {
        private int _radius; // radius of circle

        // constructor to create a circle with a specified radius and color

        public MyCircle() : this(Color.BlueViolet, 50) { }

        // constructor to create a circle with a specified radius and color

        public MyCircle(Color color, int radius) : base(color)
        {
            _radius = radius;
        }

        // gets and sets the radius of the circle

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


        // checks if the circle is at a specified point

        public override bool IsAt(Point2D pt)
        {
            // calculate the distance between the point the circle's center

            double hypotenuse = Math.Sqrt(Math.Pow(X - pt.X, 2) + Math.Pow(Y - pt.Y,
↪  2));

            // if the distance is less than the radius, the point is inside the
↪  circle

            if (hypotenuse <= Radius)
            {
                return true; // point is inside circle
            }
            else
```

```
52              {
53                  return false; // point is outside circle
54              }
55          }
56
57
58          // draws the outline of the circle on the screen
59
60          public override void DrawOutline()
61          {
62              SplashKit.DrawCircle(Color.Black, X, Y, _radius +2 );
63          }
64
65
66          // draws the circle on the screen if selected
67
68          public override void Draw()
69          {
70              SplashKit.FillCircle(Color, X, Y, _radius);
71
72              // if selected, draw the outline of the circle
73
74              if (Selected)
75              {
76                  DrawOutline();
77              }
78          }
79      }
80  }
```

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


namespace ShapeDrawer
{
    // MyLines class represents a line shapes

    public class MyLines : Shape
    {
        // coordinates of lines's end point

        private float _endX, _endY;

        // constructor to create a line with a specified end point and color

        public MyLines() : this(Color.Blue, 100, 100, 400, 300) { }

        // constructor to create a line with a specified end point and color

        public MyLines(Color color, float x, float y, float endx, float endy) :
    ↪  base(color)
        {
            X = x;
            Y = y;
            EndX = endx;
            EndY = endy;
        }

        // gets and sets the X-coordinate of the line's end point

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


        // gets and sets the Y-coordinate of the line's end point

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


        // checks if a given point is within of the line

        public override bool IsAt(Point2D pt)
        {
```

```
            // calculate the gradient and intercept of the line

            float gradient = (EndY - Y) / (EndX - X);
            float intercept = EndY - (gradient * EndX);

            // calculate the distance between the point and the line

            double margin = 10;

            // calculate the distance between the point and the line

            double distance = Math.Abs((gradient * pt.X) + intercept - pt.Y);

            // if the distance is less than the margin, the point is on the line

            return distance <= margin;
        }


        // draws the outline of the line on the screen

        public override void DrawOutline()
        {
            SplashKit.DrawCircle(Color.GhostWhite, X, Y, 4);
            SplashKit.DrawCircle(Color.GhostWhite, EndX, EndY, 4);
        }


        // draws the line on the screen if selected

        public override void Draw()
        {
            // draw the line

            SplashKit.DrawLine(Color, X, Y, EndX, EndY);

            // if selected, draw the outline of the line

            if (Selected)
            {
                DrawOutline();
            }
        }
    }
}
```


Shape Drawer: Aarya Bhati 104189110
C:\Users\aarya\Desktop\LabC
Shapedrawer2
Ready