

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

4.1P - Drawing Program - Multiple Shape Kinds

PDF generated at 16:13 on Wednesday 22nd March, 2023

```
1  using DrawingProgram.lib;
2
3  namespace DrawingProgram
4  {
5      public class Program
6      {
7          // enumeration for kinds of shapes
8          private enum ShapeKind
9          {
10              Rectangle,
11              Circle,
12              Line
13          }
14          public static void Main()
15          {
16              //initialize a variable with shapekind enum to keep track of which shape
17              ↪ is currently equipped
18              ShapeKind kindToAdd = ShapeKind.Circle;
19
20              //initialized coordinates for drawing a line, each to hold 2 mouse
21              ↪ clicks
22              float startX = 0;
23              float startY = 0;
24              float endX = 0;
25              float endY = 0;
26
27              Drawing drawing = new Drawing();
28
29              Window window = new Window("Shape Drawer", 800, 600);
30
31              //event loop starts here
32              do
33              {
34                  SplashKit.ProcessEvents();
35                  SplashKit.ClearScreen();
36
37                  //change shape kind depending on key pressed
38                  if (SplashKit.KeyTyped(KeyCode.RKey))
39                  {
40                      kindToAdd = ShapeKind.Rectangle;
41                  }
42                  if (SplashKit.KeyTyped(KeyCode.CKey))
43                  {
44                      kindToAdd = ShapeKind.Circle;
45                  }
46                  if (SplashKit.KeyTyped(KeyCode.LKey))
47                  {
48                      kindToAdd = ShapeKind.Line;
49                  }
50
51                  // add new shape
52                  if (SplashKit.MouseClicked(MouseButton.LeftButton))
53                  {
```

```

52         //new rectangle
53         if (kindToAdd == ShapeKind.Rectangle)
54         {
55             // make the new shape, assign coordinates and add it to the
↪ drawing list
56             MyRectangle newRect = new MyRectangle();
57             newRect.X = SplashKit.MouseX();
58             newRect.Y = SplashKit.MouseY();
59             drawing.AddShape(newRect);
60         }
61         //new circle
62         else if (kindToAdd == ShapeKind.Circle)
63         {
64             MyCircle newCircle = new MyCircle();
65             newCircle.X = SplashKit.MouseX();
66             newCircle.Y = SplashKit.MouseY();
67             drawing.AddShape(newCircle);
68         }
69         //new line
70         else if (kindToAdd == ShapeKind.Line)
71         {
72             // check here if mouse has been clicked once or twice. on
↪ first click assign start coordinates,
73             //on 2nd click assign end values and reset both values so
↪ they can be used for the next line
74             if (startX == 0 && startY == 0)
75             {
76                 startX = SplashKit.MouseX();
77                 startY = SplashKit.MouseY();
78             }
79             else if (endX == 0 && endY == 0)
80             {
81                 endX = SplashKit.MouseX();
82                 endY = SplashKit.MouseY();
83             }
84             MyLine newLine = new MyLine();
85             newLine.X = startX;
86             newLine.Y = startY;
87             newLine.X2 = endX;
88             newLine.Y2 = endY;
89             drawing.AddShape(newLine);
90
91             startX = 0;
92             startY = 0;
93             endX = 0;
94             endY = 0;
95         }
96     }
97 }
98
99     // delete a shape
100    if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
↪ SplashKit.KeyTyped(KeyCode.DeleteKey))

```

```
101         {
102             drawing.DeleteShape();
103         }
104         // select a shape
105         if (SplashKit.MouseClicked(MouseButton.RightButton))
106         {
107             drawing.SelectShapesAt(SplashKit.MousePosition());
108         }
109
110         // change background color
111         if (SplashKit.KeyTyped(KeyCode.SpaceKey))
112         {
113             drawing.Background = Color.Random();
114         }
115
116         drawing.Draw();
117         SplashKit.RefreshScreen();
118
119     } while (!window.CloseRequested);
120 }
121 }
122 }
```

```
1  using DrawingProgram.lib;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace DrawingProgram
9  {
10     public class Drawing
11     {
12         //variables
13         private readonly List<Shape> _shapes;
14         private Color _background;
15
16         //properties
17
18         //number of shapes in list, readonly
19         public int ShapeCount
20         {
21             get
22             {
23                 return _shapes.Count;
24             }
25         }
26
27         // background color
28         public Color Background
29         {
30             get
31             {
32                 return _background;
33             }
34             set
35             {
36                 _background = value;
37             }
38         }
39
40         //list of shapes that are currently selected
41         public List<Shape> SelectedShapes
42         {
43             get
44             {
45                 List<Shape> result = new List<Shape>();
46
47                 foreach (Shape s in _shapes)
48                 {
49                     if (s.Selected == true)
50                     {
51                         result.Add(s);
52                     }
53                 }
54             }
55         }
56     }
57 }
```

```

54
55         return result;
56     }
57 }
58
59 //constructor that accepts color as a parameter for the background
60 public Drawing(Color background)
61 {
62     _shapes = new List<Shape>();
63     _background = background;
64 }
65
66 //default constructor
67 public Drawing() : this(Color.White)
68 {
69
70 }
71
72 //methods
73 public void AddShape(Shape shape)
74 {
75     _shapes.Add(shape);
76 }
77
78 public void Draw()
79 {
80     SplashKit.ClearScreen();
81     foreach (Shape shape in _shapes)
82     {
83         shape.Draw();
84     }
85 }
86
87 public void SelectShapesAt(Point2D pt)
88 {
89     // checks if mouse position is over a shape, if true then its selected
↪ property is set to true
90     foreach (Shape s in _shapes)
91     {
92         if (s.IsAt(pt))
93         {
94             s.Selected = true;
95         }
96         else
97         {
98             s.Selected = false;
99         }
100     }
101 }
102
103 public void DeleteShape()
104 {
105     // duplicated shapes list with ToList method to read that list instead
↪ of the original one

```

```
106         // because a list cannot be modified while being enumerated
107         foreach (Shape s in _shapes.ToList())
108         {
109             if (s.Selected)
110             {
111                 _shapes.Remove(s);
112             }
113         }
114     }
115 }
116 }
```

```
1  using System;
2  using DrawingProgram.lib;
3
4  namespace DrawingProgram
5  {
6      public abstract class Shape
7      {
8          // local variables
9          private Color _color;
10         private float _x;
11         private float _y;
12         private bool _selected;
13
14         // constructor
15         public Shape(Color clr)
16         {
17             _color = clr;
18         }
19         //default constructor
20         public Shape() : this(Color.Yellow)
21         {
22         }
23
24         // properties
25         public Color Color
26         {
27             get
28             {
29                 return _color;
30             }
31             set
32             {
33                 _color = value;
34             }
35         }
36
37         public float X
38         {
39             get
40             {
41                 return _x;
42             }
43             set
44             {
45                 _x = value;
46             }
47         }
48         public float Y
49         {
50             get
51             {
52                 return _y;
53             }
54         }
55     }
```



```
54         set
55         {
56             _y = value;
57         }
58     }
59
60     public bool Selected
61     {
62         get
63         {
64             return _selected;
65         }
66         set
67         {
68             _selected = value;
69         }
70     }
71
72     // methods
73     public abstract void Draw();
74     public abstract bool IsAt(Point2D pt);
75     public abstract void DrawOutline();
76
77
78 }
79
80 }
```

```
1  using DrawingProgram.lib;
2  using System;
3  using System.Collections.Generic;
4  using System.Drawing;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace DrawingProgram
10 {
11     //this class requires common references to have their namespaces explicitly
    ↪ mentioned
12     // such as Color, Rectangle(not entirely sure why). not sure why other classes
    ↪ do not give the same error
13     public class MyRectangle : Shape
14     {
15         //local variables
16         private int _width;
17         private int _height;
18
19         //properties
20         public int Width
21         {
22             get
23             {
24                 return _width;
25             }
26             set
27             {
28                 _width = value;
29             }
30         }
31         public int Height
32         {
33             get
34             {
35                 return _height;
36             }
37             set
38             {
39                 _height = value;
40             }
41         }
42
43         // constructor (not sure yet why this accepts x y coordinates but other
    ↪ shapes dont, but its not being utilised for now)
44         public MyRectangle(lib.Color clr, float x, float y, int width, int height) :
    ↪ base( clr)
45         {
46             X = x;
47             Y = y;
48             _width = width;
49             _height = height;
```

```
50     }
51     //default constructor
52     public MyRectangle() : this(lib.Color.Green, 0, 0, 100, 100) { }
53
54     //methods
55     public override void Draw()
56     {
57         if (Selected)
58         {
59             DrawOutline();
60         }
61         SplashKit.FillRectangle(Color, X, Y, _width, _height);
62     }
63
64     public override void DrawOutline()
65     {
66         SplashKit.FillRectangle(lib.Color.Black, X - 2, Y - 2, _width + 4,
↪ _height + 4);
67     }
68
69     public override bool IsAt(Point2D pt)
70     {
71         //removed previous logic of checking mouse position
72         //create new rectangle shape because it does not accept raw values on
↪ the parameter. will look for better alternatives
73         lib.Rectangle rectangle = new lib.Rectangle()
74         {
75             X = X,
76             Y = Y,
77             Width = _width,
78             Height = _height
79         };
80         return SplashKit.PointInRectangle(pt, rectangle);
81     }
82 }
83 }
```

```
1  using DrawingProgram.lib;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace DrawingProgram
9  {
10     public class MyCircle : Shape
11     {
12         //local variables
13         private int _radius;
14
15         //properties
16         public int Radius
17         {
18             get
19             {
20                 return _radius;
21             }
22             set
23             {
24                 _radius = value;
25             }
26         }
27         //constructor
28         public MyCircle(Color clr, int radius) : base(clr)
29         {
30             _radius = radius;
31         }
32         //default constructor
33         public MyCircle() : this(Color.Blue, 50) { }
34
35         //methods
36         public override void Draw()
37         {
38             if (Selected)
39             {
40                 DrawOutline();
41             }
42             SplashKit.FillCircle(Color, X, Y, _radius);
43         }
44         public override void DrawOutline()
45         {
46             SplashKit.FillCircle(Color.Black, X, Y, _radius + 2);
47         }
48
49         public override bool IsAt(Point2D pt)
50         {
51             //checking mouse point with distance formula
52             //if (Math.Sqrt(Math.Pow(pt.X - X, 2) + Math.Pow(pt.Y - Y, 2)) <=
↪ _radius )
```

```
53         //{
54         //     return true;
55         //}
56         //else
57         //{
58         //     return false;
59         //}
60
61         // -----
62
63         //new circle to pass it on PointInCircle method
64         // because it has no overload methods that take in raw values
65         Circle circle = new Circle()
66         {
67             Center = new Point2D()
68             {
69                 X = X,
70                 Y = Y
71             },
72             Radius = _radius
73         };
74         return SplashKit.PointInCircle(pt, circle);
75     }
76 }
77 }
78 }
```

```
1  using DrawingProgram.lib;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace DrawingProgram
9  {
10     public class MyLine : Shape
11     {
12         //local variables
13         // x2 y2 for end coordinates. start coordinate values are already inherited
14         ↪ from Shape class
15         private float _x2;
16         private float _y2;
17
18         //properties
19         public float X2
20         {
21             get
22             {
23                 return _x2;
24             }
25             set
26             {
27                 _x2 = value;
28             }
29         }
30         public float Y2
31         {
32             get
33             {
34                 return _y2;
35             }
36             set
37             {
38                 _y2 = value;
39             }
40         }
41         //constructor
42         public MyLine(Color clr, float x2, float y2) : base(clr)
43         {
44             _x2 = x2;
45             _y2 = y2;
46         }
47         // default constructor
48         public MyLine() : this(Color.Red, 0, 0) {}
49
50         // methods
51         public override void Draw()
52         {
53             if (Selected)
```

```
53         {
54             DrawOutline();
55         }
56         SplashKit.DrawLine(Color, X, Y, X2, Y2);
57     }
58     public override void DrawOutline()
59     {
60         int radius = 2;
61         SplashKit.FillCircle(Color.Black, X, Y, radius);
62         SplashKit.FillCircle(Color.Black, X2, Y2, radius);
63     }
64     public override bool IsAt(Point2D pt)
65     {
66         return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y, X2, Y2), 5);
67     }
68 }
69 }
```

