

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Drawing Program - Multiple Shape Kinds

PDF generated at 22:30 on Tuesday 19th September, 2023

```
1  using System;
2  using SplashKitSDK;
3
4  namespace ShapeDrawer
5  {
6      public class Program
7      {
8          private enum ShapeKind
9          {
10              Rectangle,
11              Circle,
12              Line
13          }
14          public static void Main()
15          {
16              Drawing myDrawing = new Drawing();
17              ShapeKind kindToAdd = ShapeKind.Circle;
18
19              new Window("Drawing Shape", 800, 600);
20              do
21              {
22                  SplashKit.ProcessEvents();
23                  SplashKit.ClearScreen();
24                  if (SplashKit.KeyTyped(KeyCode.RKey))
25                  {
26                      kindToAdd = ShapeKind.Rectangle;
27                  }
28                  if (SplashKit.KeyTyped(KeyCode.LKey))
29                  {
30                      kindToAdd = ShapeKind.Line;
31                  }
32                  if (SplashKit.KeyTyped(KeyCode.CKey))
33                  {
34                      kindToAdd = ShapeKind.Circle;
35                  }
36                  if (SplashKit.MouseClicked(MouseButton.LeftButton))
37                  {
38                      Shape newShape;
39                      if (kindToAdd == ShapeKind.Circle)
40                      {
41                          MyCircle newCircle = new MyCircle();
42
43                          newShape = newCircle;
44
45                      }
46                      else if (kindToAdd == ShapeKind.Rectangle)
47                      {
48                          MyRectangle newRect = new MyRectangle();
49
50                          newShape = newRect;
51                      }
52                      else
53                      {
```

```
54         MyLine newLine = new MyLine();
55
56         newShape = newLine;
57     }
58     newShape.X = SplashKit.MouseX();
59     newShape.Y = SplashKit.MouseY();
60     myDrawing.AddShape(newShape);
61 }
62
63 if (SplashKit.MouseClicked(MouseButton.RightButton))
64 {
65     myDrawing.SelectedShapeAt(SplashKit.MousePosition());
66 }
67
68 if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
↪ SplashKit.KeyTyped(KeyCode.DeleteKey))
69 {
70     List<Shape> selectedShapes = myDrawing.SelectedShapes();
71     foreach (Shape selectedShape in selectedShapes)
72     {
73         myDrawing.RemoveShape(selectedShape);
74     }
75 }
76
77 if (SplashKit.KeyTyped(KeyCode.SpaceKey))
78 {
79     myDrawing.Background = SplashKit.RandomRGBColor(255);
80 }
81
82 myDrawing.Draw();
83
84 SplashKit.RefreshScreen();
85
86 } while (!SplashKit.WindowCloseRequested("Drawing Shape"));
87 }
88 }
89 }
```

```
1  using System;
2  using System.Linq;
3  using System.Collections.Generic;
4  using SplashKitSDK;
5
6  namespace ShapeDrawer
7  {
8      public class Drawing
9      {
10         private readonly List<Shape> _shapes;
11         private Color _background;
12
13         public Drawing(Color background)
14         {
15             _shapes = new List<Shape>();
16             _background = background;
17         }
18
19         public Drawing() : this(Color.White)
20         {
21         }
22
23         public List<Shape> SelectedShapes()
24         {
25             List<Shape> selectedShapes = new List<Shape>();
26             foreach (Shape s in _shapes)
27             {
28                 if (s.Selected)
29                 {
30                     selectedShapes.Add(s);
31                 }
32             }
33             return selectedShapes;
34         }
35         public int ShapeCount
36         {
37             get
38             {
39                 return _shapes.Count;
40             }
41         }
42
43         public Color Background
44         {
45             get
46             {
47                 return _background;
48             }
49             set
50             {
51                 _background = value;
52             }
53         }
54     }
```

```
54
55     public void Draw()
56     {
57         SplashKit.ClearScreen(_background);
58
59         foreach (Shape s in _shapes)
60         {
61             s.Draw();
62         }
63     }
64
65     public void SelectedShapeAt(Point2D pt)
66     {
67         foreach (Shape s in _shapes)
68         {
69             if (s.IsAt(pt))
70             {
71                 s.Selected = true;
72             }
73             else
74             {
75                 s.Selected = false;
76             }
77         }
78     }
79
80     public void AddShape(Shape s)
81     {
82         _shapes.Add(s);
83     }
84
85     public void RemoveShape(Shape s)
86     {
87         _shapes.Remove(s);
88     }
89 }
90 }
```

```
1  using System;
2  using SplashKitSDK;
3
4  namespace ShapeDrawer
5  {
6      public abstract class Shape
7      {
8          private Color _color;
9          private float _x, _y;
10         private bool _selected;
11         private int _width, _height;
12         public Shape(Color clr)
13         {
14             _color = clr;
15         }
16
17         public Color Color
18         {
19             get
20             {
21                 return _color;
22             }
23             set
24             {
25                 _color = value;
26             }
27         }
28
29         public float X
30         {
31             get
32             {
33                 return _x;
34             }
35             set
36             {
37                 _x = value;
38             }
39         }
40
41         public float Y
42         {
43             get
44             {
45                 return _y;
46             }
47             set
48             {
49                 _y = value;
50             }
51         }
52         public abstract void Draw();
53         public abstract bool IsAt(Point2D p);
```

```
54
55
56     public bool Selected
57     {
58         get
59         {
60             return _selected;
61         }
62         set
63         {
64             _selected = value;
65         }
66     }
67
68     public abstract void DrawOutline();
69
70
71 }
72 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using SplashKitSDK;
7
8  namespace ShapeDrawer
9  {
10     public class MyRectangle : Shape
11     {
12         private int _width, _height;
13
14         public MyRectangle(Color clr, float x, float y, int width, int height) :
↪     base(clr)
15         {
16             X = x;
17             Y = y;
18             Width = width;
19             Height = height;
20         }
21
22         public MyRectangle() : this(Color.Green, 0, 0, 100, 100) { }
23
24         public int Width // Corrected typo
25         {
26             get { return _width; }
27             set { _width = value; }
28         }
29         public int Height
30         {
31             get { return _height; }
32             set { _height = value; }
33         }
34         public override void Draw()
35         {
36             if (Selected)
37             {
38                 DrawOutline();
39             }
40             SplashKit.FillRectangle(Color, X, Y, Width, Height);
41         }
42         public override void DrawOutline()
43         {
44             SplashKit.FillRectangle(Color, X - 2, Y - 2, Width + 4, Height + 4);
45         }
46         public override bool IsAt(Point2D p)
47         {
48             if ((p.X > X) && (p.X < (X + _width)))
49             {
50                 if ((p.Y > Y) && (p.Y < (Y + _height)))
51                 {
52                     return true;
53                 }
54             }
55         }
56     }
57 }
```



```
53         }
54     }
55     return false;
56 }
57 }
58 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using SplashKitSDK;
7  namespace ShapeDrawer
8  {
9      public class MyCircle : Shape
10     {
11         private int _radius;
12
13         public MyCircle(Color clr,float x, float y, int radius) : base(clr)
14         {
15             X = x;
16             Y = y;
17             _radius = radius;
18         }
19         public MyCircle() : this(Color.Blue,0,0, 50) { }
20         public int Radius { get { return _radius; } }
21         public override void Draw()
22         {
23             if (Selected)
24                 DrawOutline();
25             SplashKit.FillCircle(Color, X, Y, _radius);
26         }
27         public override void DrawOutline()
28         {
29             SplashKit.FillCircle(Color, X - 2, Y - 2, _radius + 2);
30         }
31         public override bool IsAt(Point2D p)
32         {
33             double a = (double)(p.X - X);
34             double b = (double)(p.Y - Y);
35             if (Math.Sqrt(a * a + b * b) < _radius)
36             {
37                 return true;
38             }
39             return false;
40         }
41     }
42 }
```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using SplashKitSDK;
7
8  namespace ShapeDrawer
9  {
10     public class MyLine : Shape
11     {
12         private float _endX;
13         private float _endY;
14         public MyLine(Color clr, float startX, float startY, float endX, float endY) :
↵ base(clr)
15         {
16             _endX = endX;
17             _endY = endY;
18             X = startX;
19             Y = startY;
20
21         }
22         public MyLine() : this(Color.RandomRGB(255), 0, 0, 20, 20) { }
23
24         public float EndX
25         {
26             get { return _endX; }
27             set { _endX = value; }
28         }
29         public float EndY
30         {
31             get { return _endY; }
32             set { _endY = value; }
33         }
34
35         public override void Draw()
36         {
37             if (Selected)
38             {
39                 DrawOutline();
40             }
41             SplashKit.DrawLine(Color, X, Y, _endX, _endY);
42         }
43         public override void DrawOutline()
44         {
45             SplashKit.DrawCircle(Color.Black, X, Y, 5);
46             SplashKit.DrawCircle(Color.Black, _endX, _endY, 5);
47         }
48         public override bool IsAt(Point2D p)
49         {
50             // Calculate the distance from the point to the line
51             double distance = Math.Abs((EndY - Y) * p.X - (EndX - X) * p.Y + EndX * Y
↵ - EndY * X)

```

```
52         / Math.Sqrt(Math.Pow(EndY - Y, 2) + Math.Pow(EndX - X,  
↪ 2));  
53  
54         // Define a tolerance value for how close the point can be to the line  
55         double tolerance = 5.0; // Adjust as needed  
56  
57         // Check if the distance is within the tolerance  
58         return distance <= tolerance;  
59     }  
60  
61 }  
62 }
```

