

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

---

# Drawing Program - Multiple Shape Kinds

---

PDF generated at 20:01 on Tuesday 26<sup>th</sup> September, 2023

```
1  using SplashKitSDK;
2  using System.Runtime.Intrinsics.X86;
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
17              ShapeKind kindToAdd = ShapeKind.Circle;
18
19              Drawing _drawing = new Drawing();
20
21
22              Window window = new Window("Shape Drawer", 800, 600);
23              do
24              {
25                  SplashKit.ProcessEvents();
26                  SplashKit.ClearScreen();
27                  _drawing.Draw();
28
29                  ///
30                  if (SplashKit.KeyDown(KeyCode.RKey))
31                  {
32                      kindToAdd = ShapeKind.Rectangle;
33                  }
34                  else if (SplashKit.KeyDown(KeyCode.CKey)) { kindToAdd =
↪ ShapeKind.Circle; }
35                  else if (SplashKit.KeyDown(KeyCode.LKey)) { kindToAdd =
↪ ShapeKind.Line; }
36
37                  ///
38                  if (SplashKit.MouseClicked(MouseButton.LeftButton))
39                  {
40                      Shape newShape;
41
42                      if (kindToAdd == ShapeKind.Circle)
43                      {
44                          MyCircle newCircle = new MyCircle();
45                          newShape = newCircle;
46                      }
47                      else if (kindToAdd == ShapeKind.Line)
48                      {
49                          MyLine newLine = new MyLine();
50                          newShape = newLine;
51                          newLine.Y2 = SplashKit.MouseY();
```

```
52         }
53         else
54         {
55             MyRectangle newRect = new MyRectangle();
56             newShape = newRect;
57         }
58         newShape.X = SplashKit.MouseX();
59         newShape.Y = SplashKit.MouseY();
60         _drawing.AddShape(newShape);
61
62     }
63
64     if (SplashKit.MouseClicked(MouseButton.RightButton))
65     {
66         _drawing.SelectShapesAt(SplashKit.MousePosition());
67     }
68
69
70     if (SplashKit.KeyTyped(KeyCode.DeleteKey))
71     {
72         foreach (Shape s in _drawing.SelectedShapes)
73         {
74             _drawing.RemoveShape(s);
75         }
76     }
77
78
79
80
81     if (SplashKit.KeyTyped(KeyCode.SpaceKey))
82     {
83         _drawing.Background = SplashKit.RandomRGBColor(255);
84     }
85
86
87
88     SplashKit.RefreshScreen();
89
90
91     } while (!window.CloseRequested);
92 }
93
94 }
95
96
97 }
```

```
1  using SplashKitSDK;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7  namespace ShapeDrawer
8  {
9      public class Drawing
10     {
11         private readonly List<Shape> _shapes;
12         private Color _background;
13
14
15         public Drawing(Color background)
16         {
17             _shapes = new List<Shape>();
18             _background = background;
19         }
20         public Drawing() :this(Color.White) { }
21
22         int ShapeCount { get { return _shapes.Count; } }
23         public Color Background { get { return _background; } set { _background =
↵ value; } }
24
25         public void Draw()
26         {
27             SplashKit.ClearScreen(_background);
28
29             foreach (Shape shape in _shapes) { }
30
31             foreach (Shape i in _shapes)
32             {
33                 i.Draw();
34             }
35
36         }
37
38         public void SelectShapesAt( Point2D pt)
39         {
40             foreach (Shape i in _shapes)
41             {
42                 i.Selected = i.IsAt(pt);
43
44             }
45
46         }
47
48
49         public List<Shape> SelectedShapes
50         {
51
52
```

```
53         get
54     {
55         List<Shape> result = new List<Shape>();
56         foreach (Shape i in _shapes)
57         {
58             if (i.Selected == true)
59                 result.Add(i);
60         }
61         return result;
62     }
63 }
64
65
66
67
68
69     public void AddShape(Shape s)
70     {
71         _shapes.Add(s);
72     }
73
74     public void RemoveShape(Shape s)
75     {
76         _shapes.Remove(s);
77     }
78
79 }
80 }
81
82 /*
83  * using SplashKitSDK;
84  using System;
85  using System.Collections.Generic;
86  using System.Drawing;
87  using System.Globalization;
88  using System.Linq;
89  using System.Text;
90  using System.Threading.Tasks;
91  */
```

```
1  using SplashKitSDK;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ShapeDrawer
9  {
10     public abstract class Shape
11     {
12         private Color _color;
13         private float _x, _y;
14         private bool _selected;
15
16         public Shape()
17         {
18             _color = Color.Yellow;
19             _x = 0;
20             _y = 0;
21             _selected = false;
22         }
23
24         public Color Color
25         {
26             get { return _color; }
27             set { _color = value; }
28         }
29         public float X
30         {
31             get { return _x; }
32             set { _x = value; }
33         }
34         public float Y
35         {
36             get { return _y; }
37             set { _y = value; }
38         }
39
40
41         public abstract void Draw();
42
43         public abstract void DrawOutline();
44
45         public abstract bool IsAt(Point2D pt);
46
47
48         public bool Selected
49         {
50             get { return _selected; }
51             set { _selected = value; }
52         }
53     }
```

```
54
55
56     }
57 }
```

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



54 }

```
1  using SplashKitSDK;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ShapeDrawer
9  {
10     public class MyCircle : Shape
11     {
12         int _radius;
13
14         public int Radius { get { return _radius; } set { _radius = value; } }
15
16         public MyCircle() :this(50,Color.Blue,0,0)
17         {
18
19         }
20
21         public MyCircle(int radius, Color color, float x, float y)
22         {
23             Color = color;
24             _radius = radius;
25             X= x;
26             Y= y;
27         }
28
29         public override void Draw()
30         {
31             if (Selected)
32                 DrawOutline();
33             SplashKit.FillCircle(Color, X, Y, _radius);
34         }
35         public override void DrawOutline()
36         {
37             SplashKit.FillCircle(Color.Black, X-2,Y-2, _radius+2);
38         }
39         public override bool IsAt(Point2D pt)
40         {
41             if (pt.X < X + _radius/2 && pt.X > X - _radius / 2)
42             {
43                 if (pt.Y < Y + _radius/2 && pt.Y > Y - _radius / 2)
44                 {
45                     return true;
46                 }
47             }
48             return false;
49         }
50
51     }
52 }
```

```
1  using SplashKitSDK;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Runtime.ConstrainedExecution;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace ShapeDrawer
10 {
11     public class MyLine :Shape
12     {
13         private float _x2;
14         private float _y2;
15
16
17         public float Y2 { get { return _y2; } set { _y2 = value; } }
18
19
20
21         public MyLine() : this(Color.Black, 0, 0,0,0)
22         {
23
24         }
25
26         public MyLine( Color color, float x, float y,float x2, float y2)
27         {
28             X=x; Y=y;
29             _x2=x2;
30             _y2=y2;
31
32             Color = color;
33
34         }
35
36         public override void Draw()
37         {
38             if (Selected) { DrawOutline(); }
39             SplashKit.DrawLine(Color, X, Y, _x2, _y2);
40         }
41         public override void DrawOutline()
42         {
43             SplashKit.FillCircle(Color, X, Y, 10);
44             SplashKit.FillCircle(Color, _x2, _y2, 10);
45         }
46         public override bool IsAt(Point2D pt)
47         {
48             if (pt.X < X)
49             {
50                 if (pt.Y < Y+5 && pt.Y > Y - 5)
51                     return true;
52             }
53             return false;
```

```
54         }  
55  
56     }  
57 }
```

