

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Drawing Program - Multiple Shape Kinds

PDF generated at 17:16 on Monday 4th September, 2023

```
1  using System;
2  using SplashKitSDK;
3  namespace Task4._1
4  {
5      public class Program
6      {
7          private enum ShapeKind
8          {
9              Rectangle, Circle, Line
10         }
11
12         public static void Main()
13         {
14             Window window = new Window("Shape Drawer", 800, 600);
15
16             Drawing dr = new Drawing();
17             ShapeKind kindToAdd = ShapeKind.Circle;
18             do
19             {
20                 SplashKit.ProcessEvents();
21                 SplashKit.ClearScreen();
22
23                 if (SplashKit.MouseClicked(MouseButton.LeftButton))
24                 {
25                     Shape newShape;
26                     if (kindToAdd == ShapeKind.Circle)
27                     {
28
29                         newShape = new MyCircle();
30
31
32
33                     }
34                     else if (kindToAdd == ShapeKind.Rectangle)
35                     {
36                         newShape = new MyRectangle();
37
38
39
40                     }
41                     else
42                     {
43                         newShape = new MyLine();
44
45
46
47                     }
48
49                     newShape.X = SplashKit.MouseX();
50
51                     newShape.Y = SplashKit.MouseY();
52
53                     dr.AddShape(newShape);
```

```
54         }
55
56         if (SplashKit.KeyTyped(KeyCode.RKey))
57         {
58             kindToAdd = ShapeKind.Rectangle;
59         }
60
61         if (SplashKit.KeyTyped(KeyCode.CKey))
62         {
63             kindToAdd = ShapeKind.Circle;
64         }
65         if (SplashKit.KeyTyped(KeyCode.LKey))
66         {
67             kindToAdd = ShapeKind.Line;
68         }
69         if (SplashKit.MouseClicked(MouseButton.RightButton))
70         {
71
72             dr.SelectShapesAt(SplashKit.MousePosition());
73         }
74
75         if (SplashKit.KeyTyped(KeyCode.SpaceKey))
76         {
77
78             dr.Background1 = SplashKit.RandomRGBColor(255);
79         }
80
81         if (SplashKit.KeyTyped(KeyCode.DeleteKey))
82         {
83             foreach (Shape s in dr.Selectedshapes)
84             {
85                 dr.DeleteShape(s);
86             }
87         }
88
89         if (SplashKit.KeyTyped(KeyCode.BackspaceKey))
90         {
91             foreach (Shape s in dr.Selectedshapes)
92             {
93                 dr.DeleteShape(s);
94             }
95         }
96
97
98         dr.Draw();
99         SplashKit.RefreshScreen();
100     } while (!window.CloseRequested);
101 }
102 }
103
104 }
```

107
108
109

```
1  using System;
2  using SplashKitSDK;
3  using System.Collections.Generic;
4  namespace Task4._1
5
6  {
7      public class Drawing
8      {
9          private readonly List<Shape> _shapes;
10         private Color _background;
11         public Drawing(Color bg)
12         {
13             _shapes = new List<Shape>();
14             _background = bg;
15
16         }
17         public Drawing() : this(Color.White)
18         {
19
20         }
21         public int ShapeCount
22         {
23             get
24             {
25                 return _shapes.Count;
26             }
27         }
28         public void AddShape(Shape s)
29         {
30             _shapes.Add(s);
31         }
32         public void Draw()
33         {
34             SplashKit.ClearScreen(Background1);
35             foreach (Shape s in _shapes)
36             {
37                 s.Draw();
38             }
39
40         }
41
42         public void SelectShapesAt(Point2D pt)
43         {
44             foreach (Shape s in _shapes)
45             {
46                 if (s.IsAt(pt))
47                 {
48                     s.Selected = true;
49                 }
50                 else
51                 {
52                     s.Selected = false;
53                 }
54             }
55         }
56     }
57 }
```

```
54         }
55     }
56     public List<Shape> Selectedshapes
57     {
58         get
59         {
60             List<Shape> _result = new List<Shape>();
61
62
63             foreach (Shape s in _shapes)
64             {
65                 if (s.Selected == true)
66                 {
67                     _result.Add(s);
68                 }
69             }
70             return _result;
71         }
72     }
73     public Color Background1
74     {
75         get
76         {
77             return _background;
78         }
79         set
80         {
81             _background = value;
82         }
83     }
84
85     public void DeleteShape(Shape s)
86     {
87         _shapes.Remove(s);
88     }
89 }
90 }
91
92
93
94
```

```
1  using System;
2
3  using SplashKitSDK;
4
5  namespace Task4._1
6  {
7      public abstract class Shape
8      {
9          protected Color _color;
10         private float _x;
11         private float _y;
12
13         private bool _selected;
14
15         public Shape(Color c)
16         {
17             _color = c;
18             _x = (float)0;
19             _y = (float)0;
20
21         }
22         public Shape():this(Color.Yellow)
23         {
24
25         }
26
27         public abstract void Draw();
28
29         public Color Color
30         {
31             get
32             {
33                 return _color;
34             }
35             set
36             {
37                 _color = value;
38             }
39         }
40
41         public float X
42         {
43             get
44             {
45                 return _x;
46             }
47             set
48             {
49                 _x = value;
50             }
51         }
52         public float Y
53         {
```

```
54         get
55         {
56             return _y;
57         }
58         set
59         {
60             _y = value;
61         }
62     }
63
64     public bool Selected
65     {
66         get
67         {
68             return _selected;
69         }
70         set
71         {
72             _selected = value;
73         }
74     }
75     public abstract bool IsAt(Point2D pt);
76
77     public abstract void Outline();
78
79 }
80 }
81
82
83
84
```



```
1  using System;
2  using SplashKitSDK;
3
4  namespace Task4._1
5  {
6      public class MyRectangle:Shape
7      {
8          private int _width;
9          private int _height;
10         public MyRectangle():this(Color.Green, 0, 0, 100, 100)
11         {
12
13         }
14         public MyRectangle(Color clr, float x, float y, int width, int
↵ height):base(clr)
15         {
16
17             Width = width;
18             Height = height;
19         }
20         public int Width
21         {
22             get
23             {
24                 return _width;
25             }
26             set
27             {
28                 _width = value;
29             }
30         }
31
32         public int Height
33         {
34             get
35             {
36                 return _height;
37             }
38             set
39             {
40                 _height = value;
41             }
42         }
43
44         public override void Draw()
45         {
46             if (Selected == true)
47             {
48                 Outline();
49             }
50             SplashKit.FillRectangle(Color, X, Y, _width, _height);
51
52         }
```

```
53         public override void Outline()
54         {
55             SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, _width + 4, _height +
↵ 4);
56         }
57
58         public override bool IsAt(Point2D pt)
59         {
60
61             if (pt.X >= X && pt.X < (X + _width) && pt.Y >= Y && pt.Y <= (Y +
↵ _height))
62             {
63
64                 return true;
65             }
66             else
67             {
68                 return false;
69             }
70         }
71     }
72 }
73
```

```
1  using System;
2  using SplashKitSDK;
3  namespace Task4._1
4  {
5      public class MyCircle : Shape
6      {
7          private int _radius;
8
9          public MyCircle():this(Color.Blue, 0, 0, 50)
10         {
11
12         }
13         public MyCircle(Color clor,int x, int Y, int radius):base(clor)
14         {
15             _color = clor;
16             _radius = radius;
17         }
18
19         public int Radius
20         {
21             get
22             {
23                 return _radius;
24             }
25             set
26             {
27                 _radius = value;
28             }
29         }
30
31         public override void Draw()
32         {
33             if (Selected)
34                 Outline();
35             SplashKit.FillCircle(Color, X, Y, _radius);
36         }
37         public override void Outline()
38         {
39
40             SplashKit.FillCircle(Color.Black, X , Y , Radius+2);
41
42         }
43
44         public override bool IsAt(Point2D pt)
45         {
46
47             double point1 = (pt.X - X) * (pt.X - X);
48
49             double point2 = (pt.Y - Y) * (pt.Y - Y);
50
51             if(Math.Sqrt(point1+point2)<_radius)
52             {
53                 return true;
```

```
54         }
55         else
56         {
57             return false;
58         }
59     }
60 }
61 }
62 }
63
```

```
1  using System;
2  using System.Numerics;
3  using SplashKitSDK;
4  namespace Task4._1
5  {
6      public class MyLine : Shape
7      {
8          private float _endX;
9          private float _endY;
10
11
12         public MyLine() : this(Color.Orange, 0, 0, 200, 300)
13         {
14
15         }
16         public MyLine(Color clr, float startX, float startY, float endX, float endY)
17         ↪ : base(clr)
18         {
19
20             EndX = endX;
21             EndY = endY;
22
23         }
24
25         public float EndX
26         {
27             get
28             {
29                 return _endX;
30             }
31             set
32             {
33                 _endX = value;
34             }
35         }
36
37         public float EndY
38         {
39             get
40             {
41                 return _endY;
42             }
43             set
44             {
45                 _endY = value;
46             }
47         }
48
49         public override void Draw()
50         {
51             if (Selected)
52                 Outline();
53             SplashKit.DrawLine(Color, X, Y, EndX, EndY,
54             ↪ SplashKit.OptionLineWidth(5));
```

```
53
54     }
55     public override void Outline()
56     {
57
58         SplashKit.FillCircle(Color.Black, X, Y, 4);
59         SplashKit.FillCircle(Color.Black, EndX, EndY, 4);
60
61     }
62     public override bool IsAt(Point2D pt)
63     {
64
65         Line l = SplashKit.LineFrom(X, Y, EndX, EndY);
66
67         return SplashKit.PointOnLine(pt, l, 10);
68     }
69 }
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
```

