

```
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 DrawingShape
9 {
10     public abstract class Shape
11     {
12         private Color _color;
13         private float _x, _y;
14         private int _width, _height;
15         private bool _selected;
16
17         public Shape(Color color)
18         {
19             _color = Color.Green;
20             _x = 0;
21             _y = 0;
22             _width = 100;
23             _height = 100;
24         }
25         public Color Color
26         {
27             get { return _color; }
28             set { _color = value; }
29         }
30         public float X
31         {
32             get { return _x; }
33             set { _x = value; }
34         }
35         public float Y
36         {
37             get { return _y; }
38             set { _y = value; }
39         }
40
41         public bool Selected { get; internal set; }
42
43         public abstract void Draw();
44
45         public abstract bool IsAt(Point2D pt);
46
47         public abstract void DrawOutline();
48     }
49 }
```

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

```
49         return false;
50     }
51 }
52 }
53
```

```
1 using SplashKitSDK;
2
3 namespace DrawingShape
4 {
5     public class MyLine : Shape
6     {
7         private float _endX, _endY;
8
9         public MyLine() : this(Color.Green)
10        {
11        }
12
13        public MyLine(Color color) : base(color)
14        {
15            Color = color;
16            _endX = 700;
17            _endY = 500;
18        }
19
20        public float EndX
21        {
22            get { return _endX; }
23            set { _endX = value; }
24        }
25
26        public float EndY
27        {
28            get { return _endY; }
29            set { _endY = value; }
30        }
31
32        public override void Draw()
33        {
34            if(Selected)
35            {
36                DrawOutline();
37            }
38            SplashKit.DrawLine(Color, X, Y, _endX, _endY);
39        }
40
41        public override void DrawOutline()
42        {
43            SplashKit.FillCircle(Color.Black, X, Y, 5);
44            SplashKit.FillCircle(Color.Black, _endX, _endY, 5);
45        }
46
47        public override bool IsAt(Point2D pt)
48        {
49            return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y,
```

```
50         }  
51  
52     }  
53 }  
54
```

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

```
48     }
49
50     public override bool IsAt(Point2D pt)
51     {
52         return SplashKit.PointInRectangle(pt, SplashKit.RectangleFrom ↗
            (X, Y, _width, _height));
53     }
54 }
55 }
56
```

```
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 DrawingShape
9 {
10     internal class Drawing
11     {
12         private readonly List<Shape> _shapes;
13         private Color _background;
14
15         public Color Background
16         {
17             get { return _background; }
18             set { _background = value; }
19         }
20
21         public List<Shape> Shapes
22         {
23             get { return _shapes; }
24         }
25
26         public Drawing(Color background)
27         {
28             _shapes = new List<Shape>();
29             _background = background;
30         }
31
32         public Drawing() : this(Color.White)
33         {
34
35         }
36
37         public int ShapeCount
38         {
39             get { return _shapes.Count; }
40         }
41
42         public void AddShape(Shape s)
43         {
44             _shapes.Add(s);
45         }
46
47         public void Draw()
48         {
49             SplashKit.ClearScreen(_background);
```



```
50         foreach (Shape s in _shapes)
51         {
52             s.Draw();
53         }
54         SplashKit.RefreshScreen();
55     }
56
57     public List<Shape> SelectedShapes
58     {
59         get
60         {
61             List<Shape> result = new List<Shape>();
62             foreach (Shape s in _shapes)
63             {
64                 if (s.Selected)
65                 {
66                     result.Add(s);
67                 }
68             }
69             return result;
70         }
71     }
72
73     public void SelectShapesAt(Point2D pt)
74     {
75         foreach (Shape s in _shapes)
76         {
77             if(s.IsAt(pt))
78             {
79                 s.Selected = true;
80             }
81             else
82             {
83                 s.Selected = false;
84             }
85         }
86     }
87 }
88 }
89
```

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

```
50         default:
51             newShape = new MyRectangle();
52             break;
53         }
54         newShape.X = SplashKit.MouseX();
55         newShape.Y = SplashKit.MouseY();
56         myDrawing.AddShape(newShape);
57     }
58     Point2D pt = SplashKit.MousePosition();
59     if(SplashKit.KeyTyped(KeyCode.SpaceKey))
60     {
61         myDrawing.Background = SplashKit.RandomRGBColor(255);
62     }
63     if(SplashKit.MouseClicked(MouseButton.RightButton))
64     {
65         foreach(Shape s in myDrawing.Shapes)
66         {
67             if(s.IsAt(pt))
68             {
69                 s.Selected = !s.Selected;
70             }
71         }
72     }
73     if(SplashKit.KeyTyped(KeyCode.DeleteKey) ||
74        SplashKit.KeyTyped(KeyCode.BackspaceKey))
75     {
76         for (int i = myDrawing.ShapeCount - 1; i >= 0; i--)
77         {
78             if (myDrawing.Shapes[i].Selected)
79             {
80                 myDrawing.Shapes.RemoveAt(i);
81             }
82         }
83         myDrawing.Draw();
84         SplashKit.RefreshScreen();
85     } while (!window.CloseRequested);
86 }
87 }
88 }
89 }
```





