

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

4.1P - Drawing Program - Multiple Shape Kinds

PDF generated at 16:43 on Thursday 20th April, 2023

```
1  using System;
2  using SplashKitSDK;
3
4  namespace ShapeDrawer
5  {
6      public class Program
7      {
8          private enum ShapeKind
9          {
10              Rectangle, Circle, Line
11          }
12
13          private static Drawing myDrawing = new Drawing();
14          public static void Main()
15          {
16              ShapeKind kindToAdd = ShapeKind.Circle;
17
18              Window window = new Window("Shape Drawer", 800, 600);
19
20              do {
21
22                  SplashKit.ProcessEvents();
23                  SplashKit.ClearScreen();
24                  myDrawing.Draw();
25                  if (SplashKit.MouseClicked(SplashKitSDK.MouseButton.LeftButton))
26                  {
27                      Shape newShape;
28
29                      if (kindToAdd == ShapeKind.Rectangle)
30                      {
31                          MyRectangle newRect = new MyRectangle();
32                          newShape = newRect;
33                      } else if (kindToAdd == ShapeKind.Circle)
34                      {
35                          MyCircle newCircle = new MyCircle();
36                          newShape = newCircle;
37                      } else
38                      {
39                          MyLine newLine = new MyLine();
40                          newShape = newLine;
41                      }
42                      newShape.X = SplashKit.MouseX();
43                      newShape.Y = SplashKit.MouseY();
44                      myDrawing.AddShape(newShape);
45                  }
46                  if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.SpaceKey)) {
47                      myDrawing.Background = SplashKit.RandomRGBColor(255);
48                  }
49                  if (SplashKit.MouseClicked(SplashKitSDK.MouseButton.RightButton))
50                  {
51                      myDrawing.SelectShapesAt(SplashKit.MousePosition());
52                  }
53                  if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.BackspaceKey) ||
54                      ↪ SplashKit.KeyTyped(SplashKitSDK.KeyCode.DeleteKey))
```

```
54         {
55             foreach (Shape s in myDrawing.Selected_Shapes)
56             {
57                 myDrawing.RemoveShape(s);
58             }
59         }
60         if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.RKey))
61         {
62             kindToAdd = ShapeKind.Rectangle;
63         }
64         if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.CKey))
65         {
66             kindToAdd = ShapeKind.Circle;
67         }
68         if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.LKey))
69         {
70             kindToAdd = ShapeKind.Line;
71         }
72         if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.SKey))
73         {
74             myDrawing.Save("C:/users/charl/desktop");
75         }
76         if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.OKey))
77         {
78             try
79             {
80                 myDrawing.Load("C:/users/charl/desktop/test_save.txt");
81             } catch (Exception e)
82             {
83                 Console.Error.WriteLine("Error loading file: {0}",
↵ e.Message);
84             }
85         }
86         SplashKit.RefreshScreen();
87     } while (!window.CloseRequested);
88 }
89 }
90 }
```

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

```
54
55     public void Draw()
56     {
57         SplashKit.ClearScreen();
58         SplashKit.FillRectangle(_background, 0, 0, 800, 600);
59         foreach (Shape shape in _shapes)
60         {
61             shape.Draw();
62         }
63     }
64
65     public void SelectShapesAt(Point2D pt)
66     {
67         foreach (Shape shape in _shapes)
68         {
69             if (shape.IsAt(pt))
70             {
71                 shape.Selected = true;
72             } else
73             {
74                 shape.Selected = false;
75             }
76         }
77     }
78
79     public void RemoveShape(Shape shape)
80     {
81         foreach (Shape s in _shapes)
82         {
83             if (shape == s)
84             {
85                 _shapes.Remove(s);
86             }
87         }
88     }
89
90     public void Save(string file_path)
91     {
92         StreamWriter writer = null;
93
94         writer = new StreamWriter(file_path);
95
96         writer.WriteColor(_background);
97
98         writer.WriteLine(_shapes.Count());
99
100        foreach(Shape s in _shapes)
101        {
102            s.SaveTo(writer);
103        }
104
105        writer.Close();
106    }
```

```
107
108     public void Load(string file_path)
109     {
110         StreamReader reader = new StreamReader(file_path);
111         try {
112             _background = reader.ReadColor();
113             int count = reader.ReadInteger();
114             _shapes.Clear();
115             Shape s;
116
117             for (int i = 0; i < count; i++)
118             {
119                 string kind = reader.ReadLine();
120                 switch (kind)
121                 {
122                     case "Rectangle":
123                         s = new MyRectangle();
124                         break;
125                     case "Circle":
126                         s = new MyCircle();
127                         break;
128                     default:
129                         throw new InvalidDataException("Unknown shape kind: " +
↵ kind);
130                 }
131
132                 s.LoadFrom(reader);
133
134                 _shapes.Add(s);
135             }
136         } finally {
137
138             reader.Close();
139
140         }
141     }
142 }
143 }
```

```
1  using System;
2  using SplashKitSDK;
3
4  namespace ShapeDrawer
5  {
6
7      public abstract class Shape {
8
9          private Color _color = Color.Green;
10
11         private float _x = 0;
12
13         private float _y = 0;
14
15         private int _width = 100;
16
17         private int _height = 100;
18
19         private bool _selected = false;
20
21         public Shape(Color clr) { _color = clr; }
22         public Shape() : this(Color.Yellow) { }
23         public Color color { set { _color = value; } get { return _color; } }
24         public float X { set { _x = value; } get { return _x; } }
25         public float Y { set { _y = value; } get { return _y; } }
26
27         public bool Selected { set { _selected = value; } get { return
↵ _selected; } }
28
29         public abstract void DrawOutline();
30
31         public abstract void Draw();
32
33         public abstract bool IsAt(Point2D pt);
34
35         public virtual void SaveTo(StreamWriter writer)
36         {
37             writer.WriteColor(_color);
38             writer.WriteLine(_x);
39             writer.WriteLine(_y);
40
41         }
42
43         public virtual void LoadFrom(StreamReader reader)
44         {
45             _color = reader.ReadColor();
46             _x = reader.ReadInteger();
47             _y = reader.ReadInteger();
48         }
49     }
50
51 }
```

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



```
52         }
53     }
54
55     public override void SaveTo(StreamWriter writer)
56     {
57         writer.WriteLine("Rectangle");
58         base.SaveTo(writer);
59         writer.WriteLine(_width);
60         writer.WriteLine(_height);
61     }
62
63     public override void LoadFrom(StreamReader reader)
64     {
65         base.LoadFrom(reader);
66         _width = reader.ReadInteger();
67         _height = reader.ReadInteger();
68     }
69 }
70 }
```

```
1  using System;
2  using SplashKitSDK;
3  using System.Collections.Generic;
4
5  namespace ShapeDrawer
6  {
7      internal class MyCircle : Shape
8      {
9
10         int _radius;
11
12         Color _color;
13
14         public MyCircle(Color clr, int radius) : base(clr)
15         {
16             _radius = 50;
17             _color = clr;
18         }
19
20         public MyCircle() : this(Color.Blue, 50) { }
21
22         public override void DrawOutline()
23         {
24             SplashKit.FillCircle(Color.Black, X, Y, (_radius + 2));
25         }
26         public override void Draw()
27         {
28             if (Selected) { DrawOutline(); }
29             SplashKit.FillCircle(color, X, Y, _radius);
30         }
31
32         public override bool IsAt(Point2D pt)
33         {
34             Point2D circle_origin = SplashKit.PointAt(X, Y);
35             return SplashKit.PointInCircle(pt, SplashKit.CircleAt(circle_origin,
↵ _radius));
36         }
37
38         public override void SaveTo(StreamWriter writer)
39         {
40             writer.WriteLine("Circle");
41             base.SaveTo(writer);
42             writer.WriteLine(_radius);
43         }
44
45         public override void LoadFrom(StreamReader reader)
46         {
47             base.LoadFrom(reader);
48             _radius = reader.ReadInteger();
49         }
50     }
51 }
```

```
1  using System;
2  using SplashKitSDK;
3  using System.Collections.Generic;
4
5  namespace ShapeDrawer
6  {
7      internal class MyLine : Shape
8      {
9          double _X_start = 150;
10
11         double _Y_start = 150;
12
13         Color _color;
14
15         public MyLine(Color clr) : base(clr)
16         {
17             _color = clr;
18         }
19
20         public MyLine() : this(Color.Green) { }
21
22         public override void DrawOutline()
23         {
24             SplashKit.FillCircle(Color.Black, X, Y, 10);
25             SplashKit.FillCircle(Color.Black, _X_start, _Y_start, 5);
26         }
27
28         public override void Draw()
29         {
30             if (Selected) { DrawOutline(); }
31             SplashKit.DrawLine(_color, _X_start, _Y_start, X, Y);
32         }
33
34         public override bool IsAt(Point2D pt)
35         {
36             Point2D line_start = SplashKit.PointAt(_X_start, _Y_start);
37             Point2D line_end = SplashKit.PointAt(X, Y);
38             return SplashKit.PointOnLine(pt, SplashKit.LineFrom(line_start,
↵ line_end));
39         }
40
41         public override void SaveTo(StreamWriter writer)
42         {
43             writer.WriteLine("Line");
44             base.SaveTo(writer);
45         }
46     }
47 }
```

