

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Drawing Program - Saving and Loading

PDF generated at 16:09 on Wednesday 4th October, 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                          newCircle.X = SplashKit.MouseX();
43                          newCircle.Y = SplashKit.MouseY();
44                          newShape = newCircle;
45                      }
46                      else if (kindToAdd == ShapeKind.Rectangle)
47                      {
48                          MyRectangle newRect = new MyRectangle();
49                          newRect.X = SplashKit.MouseX();
50                          newRect.Y = SplashKit.MouseY();
51                          newShape = newRect;
52                      }
53                  }
```

```

54         else
55         {
56             MyLine newLine = new MyLine();
57             newLine.X = SplashKit.MouseX();
58             newLine.Y = SplashKit.MouseY();
59             newShape = newLine;
60         }
61         myDrawing.AddShape(newShape);
62     }
63     if (SplashKit.KeyTyped(KeyCode.SKey))
64     {
65         string folder =
↪ "C:/Users/lequa/OneDrive/Documents/COS20007/ShapeDrawer 5.3/TestDrawing.txt";
66         try
67         {
68             myDrawing.Save(folder);
69             Console.WriteLine($"Drawing saved {folder}");
70         }
71         catch (Exception e)
72         {
73             Console.Error.WriteLine("Error saviing file: {0}",
↪ e.Message);
74         }
75         /* string saveFolderPath =
↪ @"C:\Users\lequa\OneDrive\Documents\COS20007\ShapeDrawer 5.3";
76             string saveFileName = "TestDrawing.txt";
77             string savePath = System.IO.Path.Combine(saveFolderPath,
↪ saveFileName);
78             myDrawing.Save(savePath);
79
80             Console.WriteLine($"Drawing saved to {savePath}");*/
81         }
82         if (SplashKit.KeyTyped(KeyCode.OKey))
83         {
84             string folder =
↪ "C:/Users/lequa/OneDrive/Documents/COS20007/ShapeDrawer 5.3/TestDrawing.txt";
85
86             try
87             {
88                 myDrawing.Load(folder);
89                 Console.WriteLine($"Drawing loaded from {folder} ");
90
91             }
92             catch (Exception e)
93             {
94                 Console.Error.WriteLine("Error loadding file: {0}",
↪ e.Message);
95             }
96
97         }
98
99         if (SplashKit.MouseClicked(MouseButton.RightButton))
100        {

```

```
101         myDrawing.SelectedShapeAt(SplashKit.MousePosition());
102     }
103
104     if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
↪ SplashKit.KeyTyped(KeyCode.DeleteKey))
105     {
106         myDrawing.RemoveShape();
107     }
108
109     if (SplashKit.KeyTyped(KeyCode.SpaceKey))
110     {
111         myDrawing.Background = SplashKit.RandomRGBColor(255);
112     }
113
114     myDrawing.Draw();
115
116     SplashKit.RefreshScreen();
117
118     } while (!SplashKit.WindowCloseRequested("Drawing Shape"));
119 }
120 }
121 }
```

```
1  using System;
2  using System.IO;
3  using SplashKitSDK;
4  namespace ShapeDrawer
5  {
6      public static class ExtensionMethods
7      {
8          public static int ReadInteger(this StreamReader reader)
9          {
10             return Convert.ToInt32(reader.ReadLine());
11          }
12          public static float ReadSingle(this StreamReader reader)
13          {
14             return Convert.ToSingle(reader.ReadLine());
15          }
16          public static Color ReadColor(this StreamReader reader)
17          {
18             return Color.RGBColor(reader.ReadSingle(), reader.ReadSingle(),
19                                   reader.ReadSingle());
20          }
21          public static void WriteColor(this StreamWriter writer, Color clr)
22          {
23             writer.WriteLine("{0}\n{1}\n{2}", clr.R, clr.G, clr.B);
24          }
25      }
26  }
```

```
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 _selectedShapes)
27             {
28                 if (s.Selected)
29                 {
30                     _selectedShapes.Add(s);
31                 }
32             }
33             return _selectedShapes;
34         }
35
36         public int ShapeCount
37         {
38             get
39             {
40                 return _shapes.Count;
41             }
42         }
43
44         public Color Background
45         {
46             get
47             {
48                 return _background;
49             }
50             set
51             {
52                 _background = value;
53             }
54         }
55     }
```

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

```
107     }
108     public void Load (string filename)
109     {
110         StreamReader reader = new StreamReader(filename);
111         Background = reader.ReadColor();
112         int count = reader.ReadInteger();
113         _shapes.Clear();
114         for (int i = 0; i < count; i++)
115         {
116             string kind = reader.ReadLine();
117             Shape s;
118             if (kind == "Rectangle")
119             {
120                 s = new MyRectangle();
121             } else if (kind == "Circle")
122             {
123                 s = new MyCircle();
124             } else
125             {
126                 continue;
127             }
128             s.LoadFrom(reader);
129             _shapes.Add(s);
130         }
131         reader.Close();
132     }
133 }
134 }
135 }
136 }
137 }
```



```
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     public virtual void SaveTo(StreamWriter writer)
70     {
71         writer.WriteColor(Color);
72         writer.WriteLine(X);
73         writer.WriteLine(Y);
74     }
75     public virtual void LoadFrom(StreamReader reader)
76     {
77         Color = reader.ReadColor();
78         X = reader.ReadInteger();
79         Y = reader.ReadInteger();
80     }
81 }
82 }
```

```
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 public override void SaveTo(StreamWriter writer)
58 {
59     writer.WriteLine("Rectangle");
60     base.SaveTo(writer);
61     writer.WriteLine(Width);
62     writer.WriteLine(Height);
63 }
64 public override void LoadFrom(StreamReader reader)
65 {
66     base.LoadFrom(reader);
67     Width = reader.ReadInteger();
68     Height = reader.ReadInteger();
69 }
70 }
71 }
```

```
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; } set { _radius = value; } }
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         public override void SaveTo(StreamWriter writer)
42         {
43             writer.WriteLine("Circle");
44             base.SaveTo(writer);
45             writer.WriteLine(Radius);
46         }
47         public override void LoadFrom(StreamReader reader)
48         {
49             base.LoadFrom(reader);
50             Radius = reader.ReadInteger();
51         }
52     }
53 }
```

```

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         X = startX;
17         Y = startY;
18         _endX = endX;
19         _endY = endY;
20
21
22     }
23     public MyLine() : this(Color.RandomRGB(255), 0, 0, 20, 20) { }
24
25     public float EndX
26     {
27         get { return _endX; }
28         set { _endX = value; }
29     }
30     public float EndY
31     {
32         get { return _endY; }
33         set { _endY = value; }
34     }
35
36     public override void Draw()
37     {
38         if (Selected)
39         {
40             DrawOutline();
41         }
42         SplashKit.DrawLine(Color, X, Y, _endX, _endY);
43     }
44     public override void DrawOutline()
45     {
46         SplashKit.DrawCircle(Color.Black, X, Y, 5);
47         SplashKit.DrawCircle(Color.Black, _endX, _endY, 5);
48     }
49     public override bool IsAt(Point2D p)
50     {
51         // Calculate the distance from the point to the line
52         double distance = Math.Abs((EndY - Y) * p.X - (EndX - X) * p.Y + EndX * Y
↵ - EndY * X)

```

```
53         / Math.Sqrt(Math.Pow(EndY - Y, 2) + Math.Pow(EndX - X,  
↪ 2));  
54  
55         // Define a tolerance value for how close the point can be to the line  
56         double tolerance = 5.0; // Adjust as needed  
57  
58         // Check if the distance is within the tolerance  
59         return distance <= tolerance;  
60     }  
61     public override void SaveTo(StreamWriter writer)  
62     {  
63         writer.WriteLine("Line");  
64         base.SaveTo(writer); // This will write Color, X, and Y  
65         writer.WriteLine(EndX); // Write EndX  
66         writer.WriteLine(EndY); // Write EndY  
67     }  
68  
69     public override void LoadFrom(StreamReader reader)  
70     {  
71         base.LoadFrom(reader); // This reads Color, X, and Y  
72         EndX = reader.ReadSingle(); // Read EndX  
73         EndY = reader.ReadSingle(); // Read EndY  
74     }  
75 }  
76 }
```

