Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Saving and Loading

PDF generated at 11:20 on Thursday 26th October, 2023

```
using SplashKitSDK;

namespace ShapeDrawing
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }

        public static void Main()
        {
            Window window = new Window("Shape Drawer", 800, 600);
            Drawing drawing = new Drawing();

            ShapeKind kindToAdd = ShapeKind.Circle;
            //Color[] shapeColors = { Color.Red, Color.Blue, Color.Green };

            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape newShape;

                    if (kindToAdd == ShapeKind.Circle)
                    {
                        newShape = new MyCircle();
                    }
                    else if (kindToAdd == ShapeKind.Rectangle)
                    {
                        newShape = new MyRectangle();
                    }
                    else
                    {
                        newShape = new MyLine();
                    }

                    newShape.X = SplashKit.MouseX();
                    newShape.Y = SplashKit.MouseY();
                    //newShape.Color = SplashKit.RandomRGBColor(255);
                    //newShape.Color = shapeColors[(int)kindToAdd];


                    if (kindToAdd == ShapeKind.Line)
                    {
                        (newShape as MyLine).EndX = 100;
                        (newShape as MyLine).EndY = 100;
```

```
54                      }
55
56                      drawing.AddShape(newShape);
57                  }
58
59                  if (SplashKit.MouseClicked(MouseButton.RightButton))
60                  {
61                      drawing.SelectShapesAt(SplashKit.MousePosition());
62                  }
63
64                  if (SplashKit.KeyTyped(KeyCode.SpaceKey))
65                  {
66                      drawing.Background = SplashKit.RandomRGBColor(255);
67                  }
68
69                  if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
↪   SplashKit.KeyTyped(KeyCode.DeleteKey))
70                  {
71                      foreach (Shape s in drawing.SelectedShapes)
72                      {
73                          drawing.RemoveShape(s);
74                      }
75                  }
76
77                  if (SplashKit.KeyReleased(KeyCode.RKey))
78                  {
79                      kindToAdd = ShapeKind.Rectangle;
80                  }
81                  else if (SplashKit.KeyReleased(KeyCode.CKey))
82                  {
83                      kindToAdd = ShapeKind.Circle;
84                  }
85                  else if (SplashKit.KeyReleased(KeyCode.LKey))
86                  {
87                      kindToAdd = ShapeKind.Line;
88                  }
89
90                  if (SplashKit.KeyReleased(KeyCode.SKey))
91                  {
92                      drawing.Save("TestDrawing.txt");
93                  }
94
95                  if (SplashKit.KeyReleased(KeyCode.OKey))
96                  {
97                      drawing.Load("TestDrawing.txt");
98                  }
99
100                 drawing.Draw();
101
102                 SplashKit.RefreshScreen();
103             } while (!window.CloseRequested);
104         }
105     }
```

```
106 }
```

```
using System;
using System.IO;
using SplashKitSDK;

namespace ShapeDrawing
{
    public static class ExtensionMethods
    {
        public static int ReadInteger(this StreamReader reader)
        {
            return Convert.ToInt32(reader.ReadLine());
        }

        public static float ReadSingle(this StreamReader reader)
        {
            return Convert.ToSingle(reader.ReadLine());
        }

        public static Color ReadColor(this StreamReader reader)
        {
            return Color.RGBColor(reader.ReadSingle(), reader.ReadSingle(),
↪   reader.ReadSingle());
        }

        public static void WriteColor(this StreamWriter writter, Color clr)
        {
            writter.WriteLine("{0}\n{1}\n{2}", clr.R, clr.G, clr.B);
        }

    }
}
```

```
using SplashKitSDK;
using System.IO;

namespace ShapeDrawing
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;

        //Constructor
        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;
        }

        public Drawing() : this(Color.White)
        {
        }

        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> _selectedShapes = new List<Shape>();
                foreach (Shape s in _shapes)
                {
                    if (s.Selected)
                    {
                        _selectedShapes.Add(s);
                    }
                }
                return _selectedShapes;
            }
        }

        public int ShapeCount
        {
            get { return _shapes.Count; }
        }

        public Color Background
        {
            get
            {
                return _background;
            }
            set
            {
                _background = value;
            }
        }
```

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

```
107         public void Load(string filename)
108         {
109             StreamReader reader = new StreamReader(filename);
110 
111             try
112             {
113                 Shape s;
114                 string kind;
115                 Background = reader.ReadColor();
116                 int count = reader.ReadInteger();
117                 _shapes.Clear();
118 
119                 for (int i = 0; i < count; i++)
120                 {
121                     kind = reader.ReadLine();
122                     switch (kind)
123                     {
124                         case "Rectangle":
125                             s = new MyRectangle();
126                             break;
127                         case "Circle":
128                             s = new MyCircle();
129                             break;
130                         case "Line":
131                             s = new MyLine();
132                             break;
133                         default:
134                             throw new InvalidDataException("Error at shape: " +
    ↪  kind);
135                     }
136 
137                     s.LoadFrom(reader);
138                     AddShape(s);
139                 }
140             }
141             finally
142             {
143                 reader.Close();
144             }
145         }
146     }
147 }
148 
```

```
using System;
using SplashKitSDK;

namespace ShapeDrawing
{
    public abstract class Shape
    {
        private Color _color;
        private float _x;
        private float _y;
        private bool _selected;

        public Shape(Color color)
        {
            _color = color;
            _selected = false;
        }

        public Shape() : this(Color.Yellow)
        {

        }

        public bool Selected
        {
            get
            {
                return _selected;
            }
            set
            {
                _selected = value;
            }
        }

        public float X
        {
            get
            {
                return _x;
            }
            set
            {
                _x = value;
            }
        }

        public float Y
        {
            get
            {
                return _y;
            }
```

```
54              set
55              {
56                  _y = value;
57              }
58          }
59
60          public Color Color
61          {
62              get
63              {
64                  return _color;
65              }
66              set
67              {
68                  _color = value;
69              }
70          }
71
72          public virtual void SaveTo(StreamWriter writer)
73          {
74              writer.WriteColor(_color);
75              writer.WriteLine(X);
76              writer.WriteLine(Y);
77          }
78          public virtual void LoadFrom(StreamReader reader)
79          {
80              Color = reader.ReadColor();
81              X = reader.ReadInteger();
82              Y = reader.ReadInteger();
83          }
84
85          public abstract void DrawOutline();
86
87          public abstract void Draw();
88
89          public abstract bool IsAt(Point2D pt);
90      }
91  }
```

```
using SplashKitSDK;

namespace ShapeDrawing
{
    public class MyRectangle : Shape
    {
        private int _width;
        private int _height;

        public MyRectangle(Color color, float x, float y, int width, int height) :
    ↪  base(color)
        {
            X = x;
            Y = y;
            _width = width;
            _height = height;
        }

        public MyRectangle() : this(Color.Green, 0, 0, 100, 100)
        {
        }

        public int Width
        {
            get
            {
                return _width;
            }
            set
            {
                _width = value;
            }
        }

        public int Height
        {
            get
            {
                return _height;
            }
            set
            {
                _height = value;
            }
        }

        public override void Draw()
        {
            if (Selected)
            {
                DrawOutline();
            }
            SplashKit.FillRectangle(Color, X, Y, _width, _height);
```

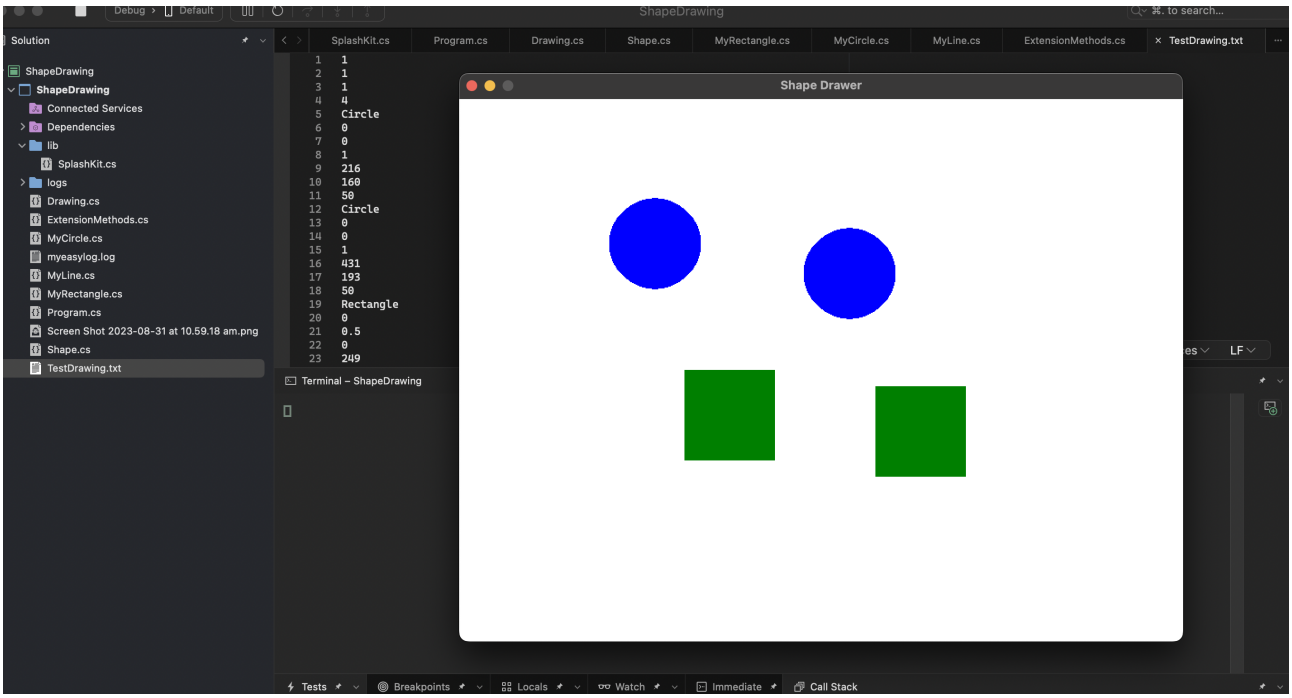
```
53          }
54
55          public override void DrawOutline()
56          {
57              SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, _width + 4, _height +
    ↪  4);
58          }
59
60          public override bool IsAt(Point2D pt)
61          {
62              return pt.X >= X && pt.X <= (X + _width) && pt.Y >= Y && pt.Y <= (Y +
    ↪  _height);
63          }
64
65          public override void SaveTo(StreamWriter writer)
66          {
67              writer.WriteLine("Rectangle");
68              base.SaveTo(writer);
69              writer.WriteLine(Width);
70              writer.WriteLine(Height);
71          }
72
73          public override void LoadFrom(StreamReader reader)
74          {
75              base.LoadFrom(reader);
76              Width = reader.ReadInteger();
77              Height = reader.ReadInteger();
78          }
79      }
80  }
```

```
using SplashKitSDK;

namespace ShapeDrawing
{
    public class MyCircle : Shape
    {
        private int _radius;

        public MyCircle(Color color, float x, float y, int radius) : base(color)
        {
            _radius = radius;
            X = x;
            Y = y;
        }

        public MyCircle() : this(Color.Blue, 0,0,50)
        {

        }

        public int Radius
        {
            get
            {
                return _radius;
            }
            set
            {
                _radius = value;
            }
        }

        public override void Draw()
        {
            if (Selected)
            {
                DrawOutline();
            }
            SplashKit.FillCircle(Color, X, Y, _radius);
        }

        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color.Black, X, Y, (_radius + 2));
        }

        public override bool IsAt(Point2D pt)
        {
            return Math.Sqrt((SplashKit.MouseX() - X) * (SplashKit.MouseX() - X) +
↪   (SplashKit.MouseY() - Y) * (SplashKit.MouseY() - Y)) < _radius;

        }

```

```
53          public override void SaveTo(StreamWriter writer)
54          {
55              writer.WriteLine("Circle");
56              base.SaveTo(writer);
57              writer.WriteLine(Radius);
58          }
59
60          public override void LoadFrom(StreamReader reader)
61          {
62              base.LoadFrom(reader);
63              Radius = reader.ReadInteger();
64          }
65      }
66  }
```

```
using SplashKitSDK;

namespace ShapeDrawing
{
    public class MyLine : Shape
    {
        private float _endX;
        private float _endY;

        public MyLine(Color color, float startX, float startY, float endX, float
↪   endY) : base(color)
        {
            X = startX;
            Y = startY;
            _endX = endX;
            _endY = endY;
        }

        public MyLine() : this(Color.RandomRGB(255), 0, 0, 10, 10)
        {
        }

        public float EndX
        {
            get
            {
                return _endX;
            }
            set
            {
                _endX = value;
            }
        }
        public float EndY
        {
            get
            {
                return _endY;
            }
            set
            {
                _endY = value;
            }
        }

        public override void Draw()
        {
            if (Selected)
                DrawOutline();
            SplashKit.DrawLine(Color.Green, X, Y, EndX, EndY);
        }

        public override void DrawOutline()
```

```
53          {
54              SplashKit.FillCircle(Color.Black, X, Y, 2);
55              SplashKit.FillCircle(Color.Black, EndX, EndY, 2);
56          }
57
58          public override bool IsAt(Point2D pt)
59          {
60              Point2D startPoint = new Point2D();
61              startPoint.X = X;
62              startPoint.Y = Y;
63              Point2D endPoint = new Point2D();
64              endPoint.X = EndX;
65              endPoint.Y = EndY;
66              Line line = new Line();
67              line.StartPoint = startPoint;
68              line.EndPoint = endPoint;
69              return SplashKit.PointOnLine(pt, line);
70          }
71
72          public override void SaveTo(StreamWriter writer)
73          {
74              writer.WriteLine("Line");
75              base.SaveTo(writer);
76              writer.WriteLine(EndX);
77              writer.WriteLine(EndY);
78          }
79
80          public override void LoadFrom(StreamReader reader)
81          {
82              base.LoadFrom(reader);
83              EndX = reader.ReadInteger();
84              EndY = reader.ReadInteger();
85          }
86      }
87  }
```


ShapeDrawing
Solution
ShapeDrawing
Connected Services
Dependencies
lib
SplashKit.cs
logs
Drawing.cs
ExtensionMethods.cs
MyCircle.cs
myeasylog.log
MyLine.cs
MyRectangle.cs
Program.cs
Screen Shot 2023-08-31 at 10.59.18 am.png
Shape.cs
TestDrawing.txt
SplashKit.cs
Program.cs
Drawing.cs
Shape.cs
MyRectangle.cs
MyCircle.cs
MyLine.cs
ExtensionMethods.cs
TestDrawing.txt
1 1 1 4 Circle 0 0 1 216 160 50 Circle 0 0 1 431 193 50 Rectangle 0 0.5 0 249
Shape Drawer
Terminal – ShapeDrawing
Tests
Breakpoints
Locals
Watch
Immediate
Call Stack