Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Saving and Loading

PDF generated at 02:21 on Friday 10th November, 2023

```
using System;
using SplashKitSDK;
using System.Collections.Generic;
using System.IO;

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

        public static void Main()
        {
            Window window = new Window("Shape Drawer", 800, 600);
            ShapeKind kindToAdd = ShapeKind.Circle;
            Drawing draw = new Drawing();
            int clicked = 0;

            List<MyLine> lines = new List<MyLine>();

            do
            {
                SplashKit.ProcessEvents();
                draw.Draw();

                if (SplashKit.KeyTyped(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
                }

                if (SplashKit.KeyTyped(KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle;
                }

                if (SplashKit.KeyTyped(KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line;
                }

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

                    if (kindToAdd == ShapeKind.Rectangle)
                    {
                        newShape = new MyRectangle();
                    }
```

```
54                      else if(kindToAdd == ShapeKind.Circle)
55                      {
56                          newShape = new MyCircle();
57                      }
58                      else
59                      {
60                          clicked++;
61                          newShape = new MyLine();
62                          if (clicked == 1)
63                          {
64                              MyLine line1 = new MyLine();
65                              line1.X = SplashKit.MouseX();
66                              line1.Y = SplashKit.MouseY();
67                              lines.Add(line1);
68                          }
69
70                          if (clicked == 2)
71                          {
72                              MyLine line2 = new MyLine();
73                              line2.X = SplashKit.MouseX();
74                              line2.Y = SplashKit.MouseY();
75                              lines.Add(line2);
76                              clicked = 0;
77                          }
78                      }
79
80                      if (kindToAdd == ShapeKind.Circle || kindToAdd ==
    ↪  ShapeKind.Rectangle)
81                      {
82                          newShape.X = SplashKit.MouseX();
83                          newShape.Y = SplashKit.MouseY();
84                      }
85                      else
86                      {
87                          if (lines.Count == 2)
88                          {
89                              newShape = lines[0] + lines[1];
90                              lines.Clear();
91                          }
92                      }
93                      draw.AddShape(newShape);
94                  }
95
96                  if (SplashKit.MouseClicked(MouseButton.RightButton))
97                  {
98                      draw.SelectShapeAt(SplashKit.MousePosition());
99                  }
100
101                 if (SplashKit.KeyTyped(KeyCode.SpaceKey))
102                 {
103                     draw.Background = SplashKit.RandomRGBColor(255);
104                 }
105
```

```
106                 if (SplashKit.KeyTyped(KeyCode.BackspaceKey))
107                 {
108                     foreach (Shape s in draw.SelectedShapes)
109                     {
110                         draw.RemoveShape(s);
111                     }
112                 }
113 
114                 if (SplashKit.KeyTyped(KeyCode.SKey))
115                 {
116                     //path may vary depends on the chosen directory within each
    ↪ computer
117                     string path =
    ↪ "C:\\msys64\\home\\anlong\\ShapeDrawer\\TestDrawing.txt";
118                     draw.Save(path);
119                     Console.WriteLine("Drawing Saved!");
120                 }
121 
122                 if (SplashKit.KeyTyped(KeyCode.OKey))
123                 {
124 
125                     Console.WriteLine("Drawing Loaded");
126                     try
127                     {
128                         string path =
    ↪ "C:\\msys64\\home\\anlong\\ShapeDrawer\\TestDrawing.txt";
129                         draw.Load(path);
130                     }
131                     catch (Exception e)
132                     {
133                         Console.Error.WriteLine("Error loading file: {0}",
    ↪ e.Message);
134                     }
135                 }
136 
137                 SplashKit.RefreshScreen();
138 
139             } while (!window.CloseRequested);
140         }
141     }
142 }
```

```
using System;
using System.IO;
using SplashKitSDK;
namespace ShapeDrawer
{
    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 writer, Color clr)
        {
            writer.WriteLine("{0}\n{1}\n{2}", clr.R, clr.G, clr.B);
        }
    }
}

```

```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

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

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

        public Drawing() : this(Color.White)
        {
            _shapes = new List<Shape>();
        }

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

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

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

```

```
54          public void AddShape(Shape newShape)
55          {
56              _shapes.Add(newShape);
57          }
58
59          public void Draw()
60          {
61              SplashKit.ClearScreen(_background);
62              foreach (Shape shape in _shapes)
63              {
64                  shape.Draw();
65              }
66          }
67
68          public void SelectShapeAt(Point2D pt)
69          {
70              foreach (Shape s in _shapes)
71              {
72                  if (s.IsAt(pt))
73                  {
74                      s.Selected = true;
75                  }
76              }
77          }
78
79          public void RemoveShape(Shape shape)
80          {
81              _shapes.Remove(shape);
82          }
83
84          public void Save(string filename)
85          {
86              StreamWriter writer = new StreamWriter(filename);
87              writer.WriteColor(Background);
88              writer.WriteLine(ShapeCount);
89              foreach(Shape s in _shapes)
90              {
91                  s.SaveTo(writer);
92              }
93              writer.Close();
94          }
95
96          public void Load(string filename)
97          {
98              StreamReader reader = new StreamReader(filename);
99              Shape s;
100             Background = reader.ReadColor();
101             int count = reader.ReadInteger();
102             _shapes.Clear();
103
104             try
105             {
106                 for (int i = 0; i < count; i++)
```

```
107                 {
108                     string kind = reader.ReadLine();
109 
110                     switch (kind)
111                     {
112                         case "Rectangle":
113                             s = new MyRectangle();
114                             break;
115 
116                         case "Circle":
117                             s = new MyCircle();
118                             break;
119 
120                         case "Line":
121                             s = new MyLine();
122                             break;
123 
124                         default:
125                             throw new InvalidDataException("Unknown shape kind: " +
    ↪  kind);
126                     }
127 
128                     s.LoadFrom(reader);
129                     AddShape(s);
130                 }
131             }
132             finally
133             {
134                 reader.Close();
135             }
136         }
137     }
138 
139 }
```

```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

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

        public Shape(Color color)
        {
            _color = color;
            _x = 0;
            _y = 0;
            _selected = false;
        }

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

        }

        public Color color
        {
            set { _color = value; }
            get { return _color; }
        }

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

        public float Y
        {
            set { _y = value; }
            get { return _y; }
        }

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

```

```
54          public abstract void Draw();
55
56          public abstract bool IsAt(Point2D pt);
57
58          public abstract void DrawOutline();
59
60          public virtual void SaveTo(StreamWriter writer)
61          {
62              writer.WriteColor(color);
63              writer.WriteLine(X);
64              writer.WriteLine(Y);
65          }
66
67          public virtual void LoadFrom(StreamReader reader)
68          {
69              color = reader.ReadColor();
70              X = reader.ReadInteger();
71              Y = reader.ReadInteger();
72          }
73      }
74
75  }
```

```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

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

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

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

        }

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

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

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

        public override void DrawOutline()
        {
            SplashKit.DrawRectangle(Color.Black, (X - 2), (Y - 2), (Width + 4),
↪   (Height + 4));
```

```
52          }
53
54          public override bool IsAt(Point2D pt)
55          {
56              if ((X < pt.X) && (X + _width > pt.X) && (Y < pt.Y) && (Y + _height >
    ↪  pt.Y))
57              {
58                  return true;
59              }
60              else
61              {
62                  return false;
63              }
64          }
65
66          public override void SaveTo(StreamWriter writer)
67          {
68              writer.WriteLine("Rectangle");
69              base.SaveTo(writer);
70              writer.WriteLine(Width);
71              writer.WriteLine(Height);
72          }
73
74          public override void LoadFrom(StreamReader reader)
75          {
76              base.LoadFrom(reader);
77              Width = reader.ReadInteger();
78              Height = reader.ReadInteger();
79          }
80      }
81  }
```

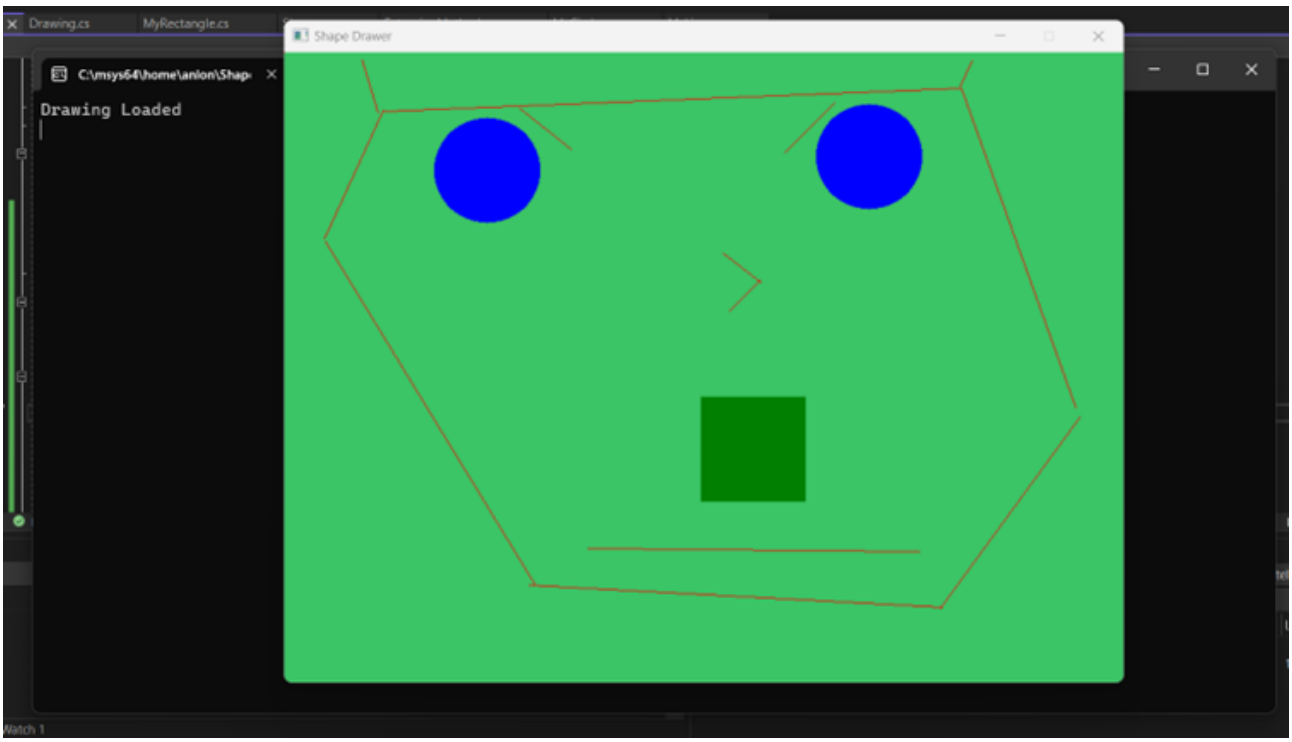
```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace ShapeDrawer
{
    public class MyCircle : Shape
    {
        int _radius;

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

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

        }

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

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

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

        public override bool IsAt(Point2D pt)
        {
            if (((X - pt.X)*(X - pt.X)) + ((Y - pt.Y)*(Y - pt.Y)) <=
↪   (_radius*_radius))
            {
                return true;
            }
            else
            {
```

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

```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace ShapeDrawer
{
    internal class MyLine : Shape
    {
        float _startX, _startY, _endX, _endY;

        public MyLine(Color clr,float startX, float startY, float endX, float endY) :
    base(clr)
        {
            _startX = startX;
            _startY = startY;
            _endX = endX;
            _endY = endY;
        }

        public MyLine() : this(Color.Red, 0, 0, 0, 0)
        {

        }


        public float StartX
        {
            get { return _startX; }
            set { _startX = value; }
        }

        public float StartY
        {
            get { return _startY; }
            set { _startY = value; }
        }

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

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

```

```
53          public override void Draw()
54          {
55              SplashKit.DrawLine(color, _startX, _startY, _endX, _endY);
56              if(Selected == true)
57              {
58                  DrawOutline();
59              }
60          }
61
62          public override void DrawOutline()
63          {
64              SplashKit.DrawCircle(Color.Black, _startX, _startY, 5);
65              SplashKit.DrawCircle(Color.Black, _endX, _endY, 5);
66          }
67
68          public override bool IsAt(Point2D pt)
69          {
70              Point2D start = new Point2D { X = _startX, Y = _startY };
71              Point2D end = new Point2D { X = _endX, Y = _endY };
72
73              Line line = new Line { StartPoint = start, EndPoint = end };
74
75              if (SplashKit.PointOnLine(pt, line, 1000))
76              {
77                  return true;
78              }
79
80              return false;
81
82          }
83
84          public static MyLine operator +(MyLine start, MyLine end)
85          {
86              MyLine newLine = new MyLine();
87              newLine.StartX = start.X;
88              newLine.StartY = start.Y;
89              newLine.EndX = end.X;
90              newLine.EndY = end.Y;
91              return newLine;
92          }
93
94          public override void SaveTo(StreamWriter writer)
95          {
96              writer.WriteLine("Line");
97              base.SaveTo(writer);
98              writer.WriteLine(StartX);
99              writer.WriteLine(StartY);
100             writer.WriteLine(EndX);
101             writer.WriteLine(EndY);
102         }
103
104         public override void LoadFrom(StreamReader reader)
105         {
```

```
106             base.LoadFrom(reader);
107             StartX = reader.ReadInteger();
108             StartY = reader.ReadInteger();
109             EndX = reader.ReadInteger();
110             EndY = reader.ReadInteger();
111         }
112     }
113 }
```


Drawing.cs
MyRectangle.cs
Shape Drawer
Drawing Loaded