Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Saving and Loading

PDF generated at 14:39 on Tuesday 3rd October, 2023

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

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);
        }
    }

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

        public static void Main()
        {
            Drawing myDrawing;
            myDrawing = new Drawing();

            ShapeKind kindToAdd = ShapeKind.Circle;


            new Window("Shape Drawer", 800, 600);
            do
            {
                SplashKit.ProcessEvents();


                if (SplashKit.KeyDown(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
```

```
53                  }
54
55                  if (SplashKit.KeyDown(KeyCode.CKey))
56                  {
57                      kindToAdd = ShapeKind.Circle;
58                  }
59
60                  if (SplashKit.KeyDown(KeyCode.LKey))
61                  {
62                      kindToAdd = ShapeKind.Line;
63                  }
64                  if (SplashKit.KeyDown(KeyCode.SKey))
65                  {
66
67                      myDrawing.Save("H:\\Credit/Mydrawing.txt");
68                  }
69
70
71                  if (SplashKit.KeyDown(KeyCode.OKey))
72                  {
73                      try
74                      {
75
76                          myDrawing.Load("H:\\Credit/Mydrawing.txt");
77                      }
78                      catch (Exception e)
79                      {
80                          Console.Error.WriteLine("Error loading file: {0}",
    ↪  e.Message);
81                      }
82                  }
83
84
85                  SplashKit.ClearScreen();
86
87
88
89                  myDrawing.Draw();
90
91
92
93                  if (SplashKit.MouseClicked(MouseButton.LeftButton))
94                  {
95
96
97                      Shape newShape;
98
99                      if (kindToAdd == ShapeKind.Circle)
100                     {
101                         MyCircle newCircle = new MyCircle();
102                         newShape = newCircle;
103                     }
104
```

```
105                     else if (kindToAdd == ShapeKind.Rectangle)
106                     {
107                         MyRectangle newRect = new MyRectangle();
108                         newShape = newRect;
109                     }
110 
111                     else
112                     {
113                         MyLine newLine = new MyLine();
114                         newShape = newLine;
115                     }
116 
117                     newShape.X = SplashKit.MouseX();
118                     newShape.Y = SplashKit.MouseY();
119 
120                     myDrawing.AddShape(newShape);
121                 }
122 
123                 if (SplashKit.KeyDown(KeyCode.SpaceKey))
124                 {
125                     myDrawing.Background = SplashKit.RandomColor();
126                 }
127 
128                 if (SplashKit.MouseClicked(MouseButton.RightButton))
129                 {
130                     myDrawing.SelectShapesAt(SplashKit.MousePosition());
131 
132                 }
133 
134                 if (SplashKit.KeyDown(KeyCode.DeleteKey) ||
    ↪  SplashKit.KeyDown(KeyCode.BackspaceKey))
135                 {
136                     foreach (Shape s in myDrawing.SelectedShapes)
137                     {
138                         myDrawing.RemoveShape(s);
139                     }
140                 }
141 
142                 SplashKit.RefreshScreen();
143             } while (!SplashKit.WindowCloseRequested("Shape Drawer"));
144         }
145     }
146 }
```

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

namespace MyGame
{
    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($"{clr.R}\n{clr.G}\n{clr.B}");
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using SplashKitSDK;
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 = SplashKit.ColorWhite();
        }
        public Drawing() : this(Color.White) { }

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

                return result;
            }
        }

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

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

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

```
107             }
108         }
109 
110         public void Load(string filename)
111         {
112             StreamReader reader = new StreamReader(filename);
113             try
114             {
115                 int count;
116                 Shape s;
117                 string kind;
118 
119                 _background = reader.ReadColor();
120                 count = reader.ReadInteger();
121                 _shapes.Clear();
122 
123                 for (int i = 0; i < count; i++)
124                 {
125                     kind = reader.ReadLine();
126                     switch (kind)
127                     {
128                         case "Rectangle":
129                             s = new MyRectangle();
130                             break;
131                         case "Circle":
132                             s = new MyCircle();
133                             break;
134                         case "Line":
135                             s = new MyLine();
136                             break;
137                         default:
138                             throw new InvalidDataException("Unknown shape kind: " +
    ↪ kind);
139                     }
140 
141                     s.LoadFrom(reader);
142                     AddShape(s);
143                 }
144             }
145             finally
146             {
147                 reader.Close();
148             }
149         }
150 
151     }
152 }
```

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

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

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

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

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



        public Shape(Color color)
        {
            _color = color;
```

```
54              _x = 0;
55              _y = 0;
56              // _width = 100;
57              // _height = 100;
58
59          }
60          public Shape() : this(Color.Yellow) { }
61
62          public bool Selected
63          {
64              get => _selected;
65              set => _selected = value;
66          }
67
68          public abstract void Draw();
69
70
71
72
73          public abstract bool IsAt(Point2D pt);
74
75          public abstract void DrawOutline();
76
77          public virtual void SaveTo(StreamWriter writer)
78          {
79              writer.WriteColor(_color);
80              writer.WriteLine(X);
81              writer.WriteLine(Y);
82          }
83
84          public virtual void LoadFrom(StreamReader reader)
85          {
86              _color = reader.ReadColor();
87              X = reader.ReadInteger();
88              Y = reader.ReadInteger();
89          }
90
91      }
92  }
```

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

namespace ShapeDrawer
{
    public class MyRectangle : Shape
    {
        private int _width, _heigth;
        public MyRectangle(Color clr, int width, int height, float x, float y) :
↪   base(clr)
        {
            _width = width;
            _heigth = height;
            this.X = x;
            this.Y = y;

        }

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

        }

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

        public int Heigth
        {
            get
            {
                return _heigth;
            }
            set
            {
                _heigth = value;
            }
        }

        public override void Draw()
        {
            if (Selected)
            {
```

```
53                 this.DrawOutline();
54             }
55             SplashKit.FillRectangle(this.Color, this.X, this.Y, this.Width,
   ↪  this.Heigth);
56         }
57
58         public override void DrawOutline()
59         {
60             SplashKit.DrawRectangle(SplashKit.ColorBlack(), this.X - 2, this.Y - 2,
   ↪  Width + 4, Heigth + 4);
61         }
62
63         public override bool IsAt(Point2D pt)
64         {
65             if ((pt.X >= this.X) && (pt.X <= (this.X + Width)) && (pt.Y >= this.Y) &&
   ↪  (pt.Y <= (this.Y + Heigth)))
66             {
67                 return true;
68             }
69
70             else
71             {
72                 return false;
73             }
74         }
75
76         public override void SaveTo(StreamWriter writer)
77         {
78             writer.WriteLine("Rectangle");
79             base.SaveTo(writer);
80             writer.WriteLine(Width);
81             writer.WriteLine(Heigth);
82         }
83
84         public override void LoadFrom(StreamReader reader)
85         {
86             base.LoadFrom(reader);
87             Width = reader.ReadInteger();
88             Heigth = reader.ReadInteger();
89         }
90     }
91
92 }
```

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

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

        public MyCircle(Color clr, int radius) : base(clr)
        {
            _radius = radius;
        }
        public MyCircle() : this(Color.Blue, 50) { }

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

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

        public override bool IsAt(Point2D pt)
        {
            return SplashKit.PointInCircle(pt, SplashKit.CircleAt(this.X, this.Y,
↪  _radius));
        }

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

        public override void SaveTo(StreamWriter writer)
        {
            writer.WriteLine("Circle");
            base.SaveTo(writer);
```

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

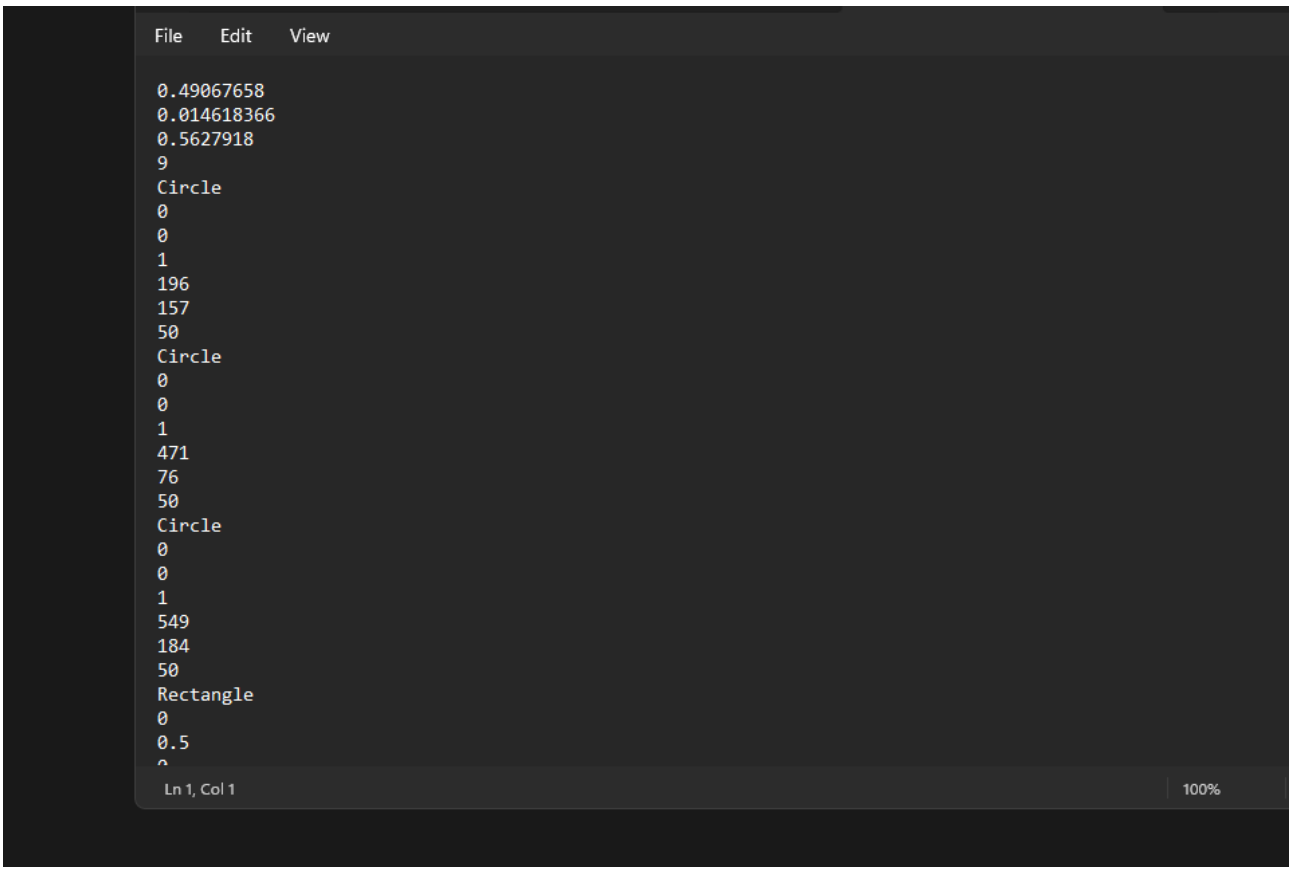
```
using System;
using SplashKitSDK;
using System.IO;

namespace ShapeDrawer
{
    public class MyLine : Shape
    {
        private float _endX, _endY;
        private int _radius;
        public MyLine(Color clr, float startX, float startY, float endX, float endY)
    ↪   : base(clr)
        {
            _endX = endX;
            _endY = endY;
            this.X = startX;
            this.Y = startY;
            _radius = 10;
        }
        public MyLine() : this(Color.Red, 0, 0, 100, 100) { }

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

        public override void Draw()
        {
            if (this.Selected)
            {
                this.DrawOutline();
            }
            SplashKit.DrawLine(this.Color,
    ↪   SplashKit.LineFrom(SplashKit.PointAt(this.X - 100, this.Y),
    ↪   SplashKit.PointAt(this.X + 100, this.Y)));
```

```
51          }
52
53          public override void DrawOutline()
54          {
55              SplashKit.FillCircle(Color.Black, this.X - 100, this.Y, _radius);
56              SplashKit.FillCircle(Color.Black, this.X + 100, this.Y, _radius);
57          }
58
59          public override bool IsAt(Point2D pt)
60          {
61              return SplashKit.PointOnLine(pt,
    ↪ SplashKit.LineFrom(SplashKit.PointAt(this.X - 100, this.Y),
    ↪ SplashKit.PointAt(this.X + 100, this.Y)));
62          }
63
64          public override void SaveTo(StreamWriter writer)
65          {
66              writer.WriteLine("Line");
67              base.SaveTo(writer);
68              writer.WriteLine(this.X);
69              writer.WriteLine(this.Y);
70              writer.WriteLine(_endX);
71              writer.WriteLine(_endY);
72          }
73
74          public override void LoadFrom(StreamReader reader)
75          {
76              base.LoadFrom(reader);
77              //_radius = reader.ReadInteger();
78              this.X = reader.ReadInteger();
79              this.Y = reader.ReadInteger();
80              _endX = reader.ReadInteger();
81              _endY = reader.ReadInteger();
82          }
83
84      }
85  }
```


File Edit View
0.49067658
0.014618366
0.5627918
9
Circle
0
0
1
196
157
50
Circle
0
0
1
471
76
50
Circle
0
0
1
549
184
50
Rectangle
0
0.5
Ln 1, Col 1
100%