Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Saving and Loading

PDF generated at 00:03 on Monday 4th September, 2023

```
using System;
using SplashKitSDK;
using DrawingProgram;

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

        private const string fPath = "/Users/cobeo/Desktop/Codes/SchoolWork/COS20007
 ↪  - OOP/5.3C - Drawing Program - Saving and Loading/TestDrawing.txt";

        public static void Main(string[] args)
        {
            Drawing drawing = new Drawing();
            ShapeKind kindToAdd = ShapeKind.Rectangle;
            Window window = new Window("Drawing Shape 5.3C", 800, 600);
            while (!window.CloseRequested)
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();
                if (SplashKit.KeyDown(KeyCode.RKey))
                    kindToAdd = ShapeKind.Rectangle;
                if (SplashKit.KeyDown(KeyCode.CKey))
                    kindToAdd = ShapeKind.Circle;
                if (SplashKit.KeyDown(KeyCode.LKey))
                    kindToAdd = ShapeKind.Line;
                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape chosenShape;
                    switch (kindToAdd)
                    {
                        case (ShapeKind.Rectangle):
                            chosenShape = new MyRectangle();
                            break;
                        case (ShapeKind.Circle):
                            chosenShape = new MyCircle();
                            break;
                        case (ShapeKind.Line):
                            chosenShape = new MyLine();
                            break;
                        default:
                            chosenShape = null!;
                            break;
                    }

                    if (chosenShape != null)
                    {
                        chosenShape.X = SplashKit.MouseX();
                        chosenShape.Y = SplashKit.MouseY();
                        drawing.AddShape(chosenShape);
                    }
                }

                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    drawing.SelectShapesAt(SplashKit.MousePosition());
                }

                if (SplashKit.KeyDown(KeyCode.SpaceKey))
                {
                    drawing.Background = SplashKit.RandomRGBColor(255);
                }

                if (SplashKit.KeyDown(KeyCode.EscapeKey))
                {
                    foreach (Shape s in drawing.SelectedShapes)
                    {
                        s.Color = SplashKit.RandomRGBColor(255);
                    }
                }

                if (SplashKit.KeyDown(KeyCode.DeleteKey) || SplashKit.KeyDown(KeyCode.BackspaceKey))
                {
                    foreach (Shape s in drawing.SelectedShapes)
                    {
                        drawing.RemoveShape(s);
                    }
                }

                if (SplashKit.KeyDown(KeyCode.SKey))
                    drawing.Save(fPath);
                if (SplashKit.KeyDown(KeyCode.OKey))
                {
                    try
                    {
                        drawing.Load(fPath);
                    } catch(Exception e)
                    {
                        Console.Write("Error: {0}", e.Message);
                    }
                }

                drawing.Draw();
                SplashKit.RefreshScreen();
            }

        }
    }
```

```
}
```

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

namespace DrawingProgram
{
    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 color)
        {
            writer.WriteLine("{0}\n{1}\n{2}", color.R, color.G, color.B);
        }
    }
}

```

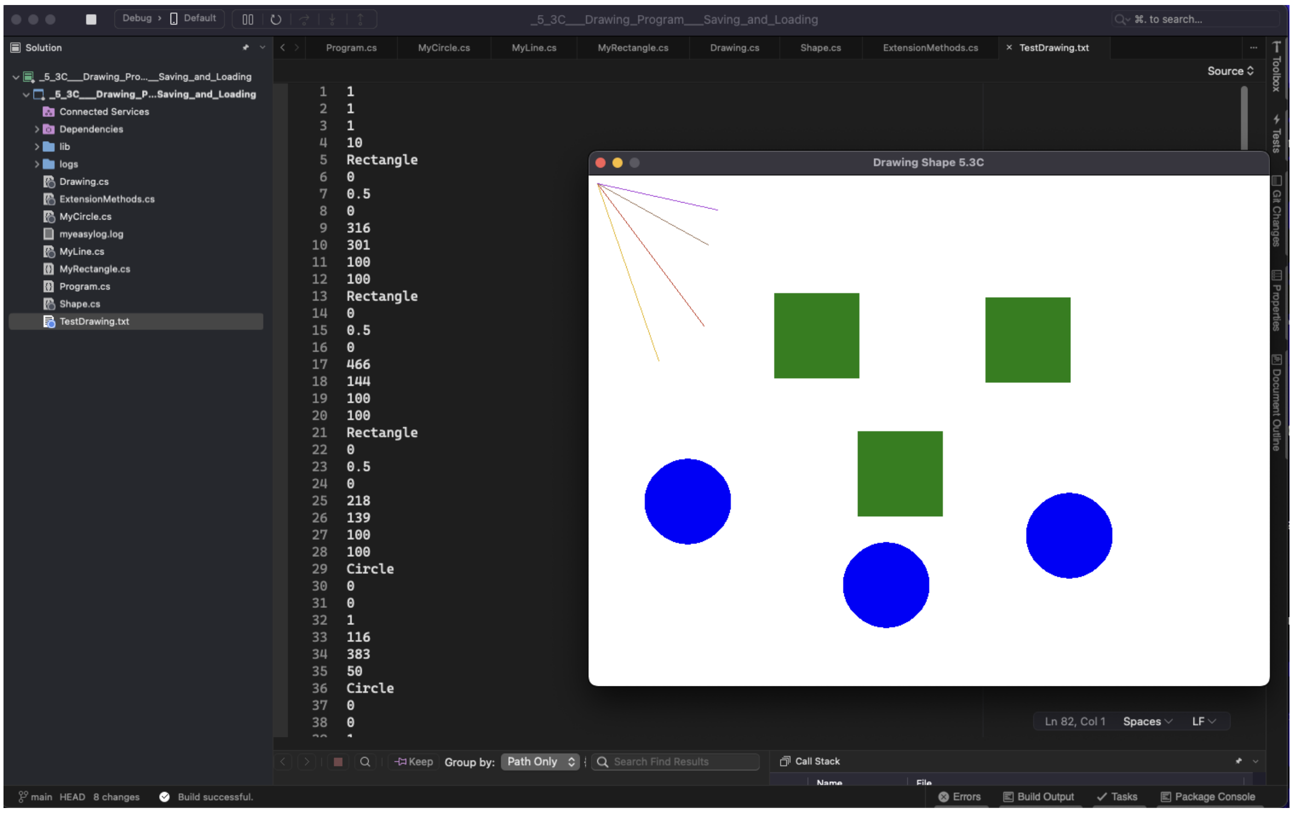
```
using System;
using SplashKitSDK;
using System.Collections.Generic;

namespace DrawingProgram
{
    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) { }

        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;
            }
        }

        public void Draw()
        {
            SplashKit.ClearScreen(_background);
            foreach (Shape s in _shapes)
            {
                s.Draw();
            }

        }
        public void SelectShapesAt(Point2D point)
        {
            foreach (Shape s in _shapes)
            {
                s.Selected = s.IsAt(point);
            }
        }
        public void AddShape(Shape s)
        {
            _shapes.Add(s);
        }
        public void RemoveShape(Shape s)
        {
            _shapes.Remove(s);
        }

        public void Save(string fileName)
        {
            StreamWriter writer = new StreamWriter(fileName);
            try
            {
                writer.WriteColor(_background);
                writer.WriteLine(ShapeCount);

                foreach (Shape s in _shapes)
                {
                    s.SaveTo(writer);

                }
            }
            finally
            {
                writer.Close();
            }
        }

        public void Load(string fileName)
        {
            StreamReader reader = new StreamReader(fileName);
            try
            {
                int count;
                Shape s;
                string kind;

                Background = reader.ReadColor();
                count = reader.ReadInteger();
                _shapes.Clear();
                for (int i = 0; i < count; i++)
                {
                    kind = reader.ReadLine();

                    switch (kind)
                    {
                        case "Rectangle":
                            s = new MyRectangle();
                            break;
                        case "Circle":
                            s = new MyCircle();
                            break;
                        case "Line":
                            s = new MyLine();
                            break;
                        default:
                            throw new InvalidDataException("Unknown Shape Kind: " + kind);
                    }
                    s.LoadFrom(reader);
                    AddShape(s);
                }
            }
            finally
            {
                reader.Close();
            }
        }
    }
}
```

```
using System;
using SplashKitSDK;

namespace DrawingProgram
{
    public abstract class Shape
    {
        private Color _color;
        private float _x, _y;
        private bool _selected;
        private int _width, _height;

        public Shape(Color color)
        {
            _color = color;
            _x = 0;
            _y = 0;
            _width = 100;
            _height = 100;

        }

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

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

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

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

        public abstract void Draw();
        public abstract bool IsAt(Point2D pt);
        public abstract void DrawOutline();

        public virtual void SaveTo(StreamWriter writer)
        {
            writer.WriteColor(_color);
            writer.WriteLine(X);
            writer.WriteLine(Y);
        }
        public virtual void LoadFrom(StreamReader reader)
        {
            Color = reader.ReadColor();
            X = reader.ReadInteger();
            Y = reader.ReadInteger();
        }
    }
}
```

```
using System;
using SplashKitSDK;

namespace DrawingProgram
{
    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 new int Width
        {
            get
            {
                return _width;
            }
            set
            {
                _width = value;
            }
        }

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

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

        public override bool IsAt(Point2D pt)
        {
            return SplashKit.PointInRectangle(pt, SplashKit.RectangleFrom(X, Y,
    ↪  _width, _height));
        }

        public override void DrawOutline()
        {
            SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, _width + 4, _height +
    ↪  4);
        }

        public override void SaveTo(StreamWriter writer)
        {
            writer.WriteLine("Rectangle");
            base.SaveTo(writer);
            writer.WriteLine(Width);
            writer.WriteLine(Height);
        }

        public override void LoadFrom(StreamReader reader)
        {
            base.LoadFrom(reader);
            Width = reader.ReadInteger();
            Height = reader.ReadInteger();
        }
    }
}
```

```csharp
using System;
using SplashKitSDK;

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

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

        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 bool IsAt(Point2D pt)
        {
            //c = sqrt(a^2 + b^2)
            return Math.Sqrt((pt.X - X) * (pt.X - X) + (pt.Y - Y) * (pt.Y - Y)) <
 ↪  _radius;
        }

        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color.Black, X, Y, _radius + 4);
        }

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

        public override void LoadFrom(StreamReader reader)
        {
            base.LoadFrom(reader);
            Radius = reader.ReadInteger();
        }
    }
}

```

```
using System;
using SplashKitSDK;

namespace DrawingProgram
{
    public class MyLine : Shape
    {
        private float _endX, _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, X, Y, _endX, _endY);
        }

        public override void DrawOutline()
        {
            SplashKit.DrawLine(Color, X, Y, _endX + 5, _endY + 5);
        }

        public override bool IsAt(Point2D pt)
        {
            return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y, _endX, _endY));
        }

        public override void SaveTo(StreamWriter writer)
        {
            writer.WriteLine("Line");
            base.SaveTo(writer);
            writer.WriteLine(EndX);
            writer.WriteLine(EndY);
        }

        public override void LoadFrom(StreamReader reader)
        {
            base.LoadFrom(reader);
            EndX = reader.ReadInteger();
            EndY = reader.ReadInteger();
        }
    }
}
```



```
1
1
1
10
Rectangle
0
0.5
0
316
301
100
100
Rectangle
0
0.5
0
466
144
100
100
Rectangle
0
0.5
0
218
139
100
100
Circle
0
0
1
116
383
50
Circle
0
0
```