Swinburne University of Technology

COS20007 Object Oriented Programming

4.1P - Drawing Program - Multiple Shape Kinds

PDF generated at 16:43 on Thursday 20th April, 2023

```
using System;
using SplashKitSDK;

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

        private static Drawing myDrawing = new Drawing();
        public static void Main()
        {
            ShapeKind kindToAdd = ShapeKind.Circle;

            Window window = new Window("Shape Drawer", 800, 600);

            do {

                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();
                myDrawing.Draw();
                if (SplashKit.MouseClicked(SplashKitSDK.MouseButton.LeftButton))
                {
                    Shape newShape;

                    if (kindToAdd == ShapeKind.Rectangle)
                    {
                        MyRectangle newRect = new MyRectangle();
                        newShape = newRect;
                    } else if (kindToAdd == ShapeKind.Circle)
                    {
                        MyCircle newCircle = new MyCircle();
                        newShape = newCircle;
                    } else
                    {
                        MyLine newLine = new MyLine();
                        newShape = newLine;
                    }
                    newShape.X = SplashKit.MouseX();
                    newShape.Y = SplashKit.MouseY();
                    myDrawing.AddShape(newShape);
                }
                if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.SpaceKey)) {
                    myDrawing.Background = SplashKit.RandomRGBColor(255);
                }
                if (SplashKit.MouseClicked(SplashKitSDK.MouseButton.RightButton))
                {
                    myDrawing.SelectShapesAt(SplashKit.MousePosition());
                }
                if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.BackspaceKey) ||
↪   SplashKit.KeyTyped(SplashKitSDK.KeyCode.DeleteKey))
```

```
                {
                    foreach (Shape s in myDrawing.Selected_Shapes)
                    {
                        myDrawing.RemoveShape(s);
                    }
                }
                if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
                }
                if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle;
                }
                if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line;
                }
                if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.SKey))
                {
                    myDrawing.Save("C:/users/charl/desktop");
                }
                if (SplashKit.KeyTyped(SplashKitSDK.KeyCode.OKey))
                {
                    try
                    {
                        myDrawing.Load("C:/users/charl/desktop/test_save.txt");
                    } catch (Exception e)
                    {
                        Console.Error.WriteLine("Error loading file: {0}",
  ↪ e.Message);
                    }
                }
                SplashKit.RefreshScreen();
            } while (!window.CloseRequested);
        }
    }
}
```

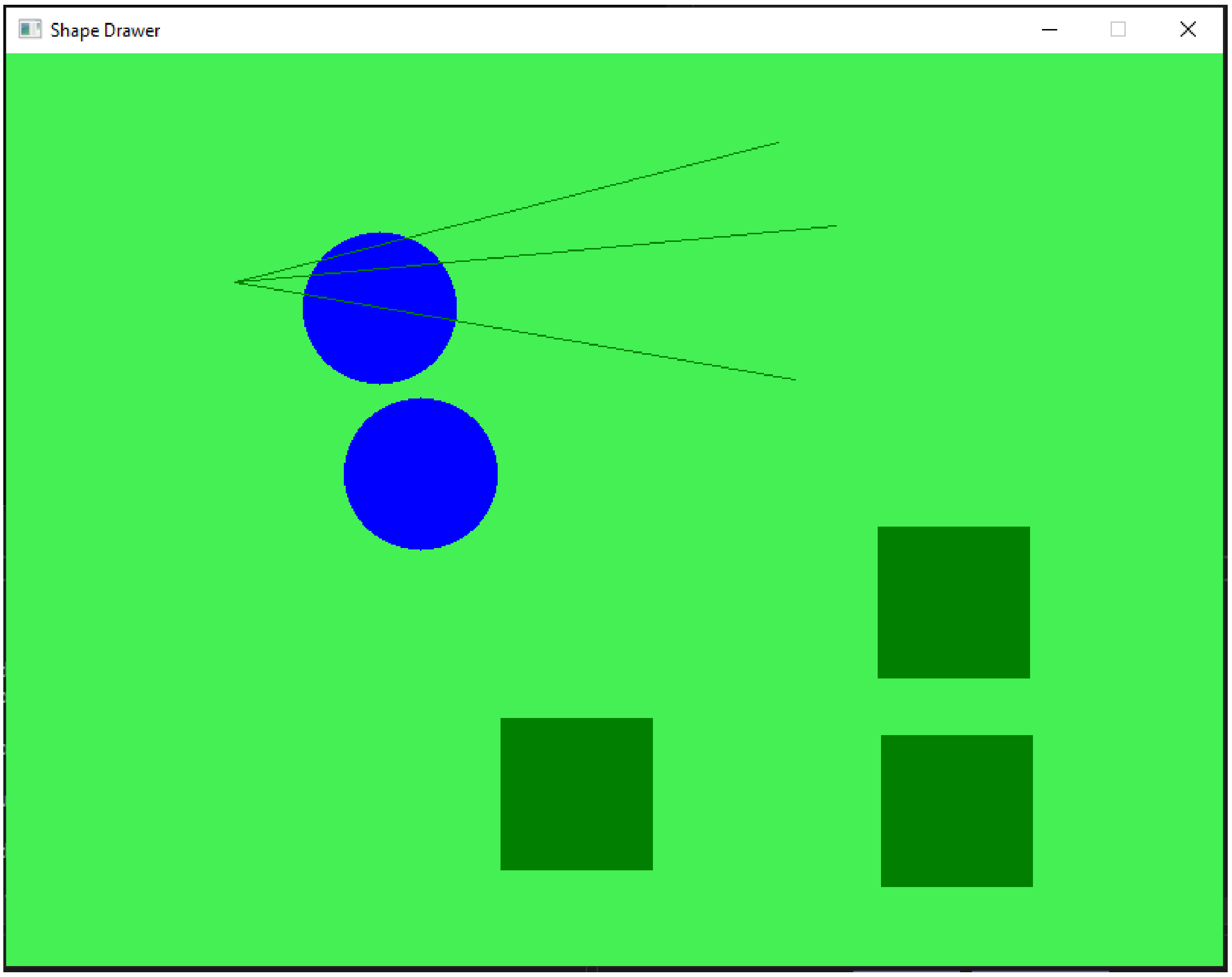
```csharp
using System;
using SplashKitSDK;
using System.Collections.Generic;

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) { }

        public List<Shape> Selected_Shapes
        {
            get
            {
                List<Shape> _selectedShapes = new List<Shape>();

                foreach (Shape shape in _shapes)
                {
                    if (shape.Selected)
                    {
                        _selectedShapes.Add(shape);
                    }
                }

                return _selectedShapes;
            }
        }

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

        }

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

        public void AddShape(Shape shape)
        {
            _shapes.Add(shape);
        }

        public void Draw()
        {
            SplashKit.ClearScreen();
            SplashKit.FillRectangle(_background, 0, 0, 800, 600);
            foreach (Shape shape in _shapes)
            {
                shape.Draw();
            }
        }

        public void SelectShapesAt(Point2D pt)
        {
            foreach (Shape shape in _shapes)
            {
                if (shape.IsAt(pt))
                {
                    shape.Selected = true;
                } else
                {
                    shape.Selected = false;
                }
            }
        }

        public void RemoveShape(Shape shape)
        {
            foreach (Shape s in _shapes)
            {
                if (shape == s)
                {
                    _shapes.Remove(s);
                }
            }
        }

        public void Save(string file_path)
        {
            StreamWriter writer = null;

            writer = new StreamWriter(file_path);

            writer.WriteColor(_background);

            writer.WriteLine(_shapes.Count());

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

            writer.Close();
        }

        public void Load(string file_path)
        {
            StreamReader reader = new StreamReader(file_path);
            try {
                _background = reader.ReadColor();
                int count = reader.ReadInteger();
                _shapes.Clear();
                Shape s;

                for (int i = 0; i < count; i++)
                {
                    string kind = reader.ReadLine();
                    switch (kind)
                    {
                        case "Rectangle":
                            s = new MyRectangle();
                            break;
                        case "Circle":
                            s = new MyCircle();
                            break;
                        default:
                            throw new InvalidDataException("Uknown shape kind: " +
↪   kind);
                    }

                    s.LoadFrom(reader);

                    _shapes.Add(s);
                }
            } finally {

                reader.Close();

            }
        }
    }
}
```

```
using System;
using SplashKitSDK;

namespace ShapeDrawer
{

        public abstract class Shape {

            private Color _color = Color.Green;

            private float _x = 0;

            private float _y = 0;

            private int _width = 100;

            private int _height = 100;

            private bool _selected = false;

            public Shape(Color clr) { _color = clr; }
            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; } }

        public abstract void DrawOutline();

            public abstract void Draw();

            public abstract bool IsAt(Point2D pt);

            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;
using System.Collections.Generic;

namespace ShapeDrawer
{
    public class MyRectangle : Shape
    {

        private int _width = 100;

        private int _height = 100;

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

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

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

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

        public override bool IsAt(Point2D pt)
        {
            if (pt.X > X && pt.X < (X + _width) && pt.Y > Y && pt.Y < (Y + _height))
            {

                return true;

            }
            else
            {

                return false;

            }
        }

        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();
        }
    }
}
```

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

namespace ShapeDrawer
{
    internal class MyCircle : Shape
    {

        int _radius;

        Color _color;

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

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

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

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

        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;
using System.Collections.Generic;

namespace ShapeDrawer
{
    internal class MyLine : Shape
    {
        double _X_start = 150;

        double _Y_start = 150;

        Color _color;

        public MyLine(Color clr) : base(clr)
        {
            _color = clr;
        }

        public MyLine() : this(Color.Green) { }

        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color.Black, X, Y, 10);
            SplashKit.FillCircle(Color.Black, _X_start, _Y_start, 5);
        }

        public override void Draw()
        {
            if (Selected) { DrawOutline(); }
            SplashKit.DrawLine(_color, _X_start, _Y_start, X, Y);
        }

        public override bool IsAt(Point2D pt)
        {
            Point2D line_start = SplashKit.PointAt(_X_start, _Y_start);
            Point2D line_end = SplashKit.PointAt(X, Y);
            return SplashKit.PointOnLine(pt, SplashKit.LineFrom(line_start,
 ↪  line_end));
        }

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


Shape Drawer