Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Multiple Shape Kinds

PDF generated at 15:03 on Wednesday 17th May, 2023

```csharp
using System;
using MultipleShapes;
using SplashKitSDK;

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

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

        ShapeKind kindToAdd = ShapeKind.Circle;

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

            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)
                {
                    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();

                drawingshape.AddShape(newShape);
            }

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

            }

            if (SplashKit.KeyReleased(KeyCode.DeleteKey) ||
↪   SplashKit.KeyReleased(KeyCode.BackspaceKey))
            {
                foreach (Shape shape in drawingshape.SelectedShapes)
                {
                    drawingshape.RemoveShape(shape);
                }
            }

            if (SplashKit.KeyTyped(KeyCode.SpaceKey))
            {
                drawingshape.Background = SplashKit.RandomRGBColor(255);
            }

            drawingshape.Draw();
            SplashKit.RefreshScreen();

        }
        while (!SplashKit.WindowCloseRequested("Drawing Shape"));

    }
}
```

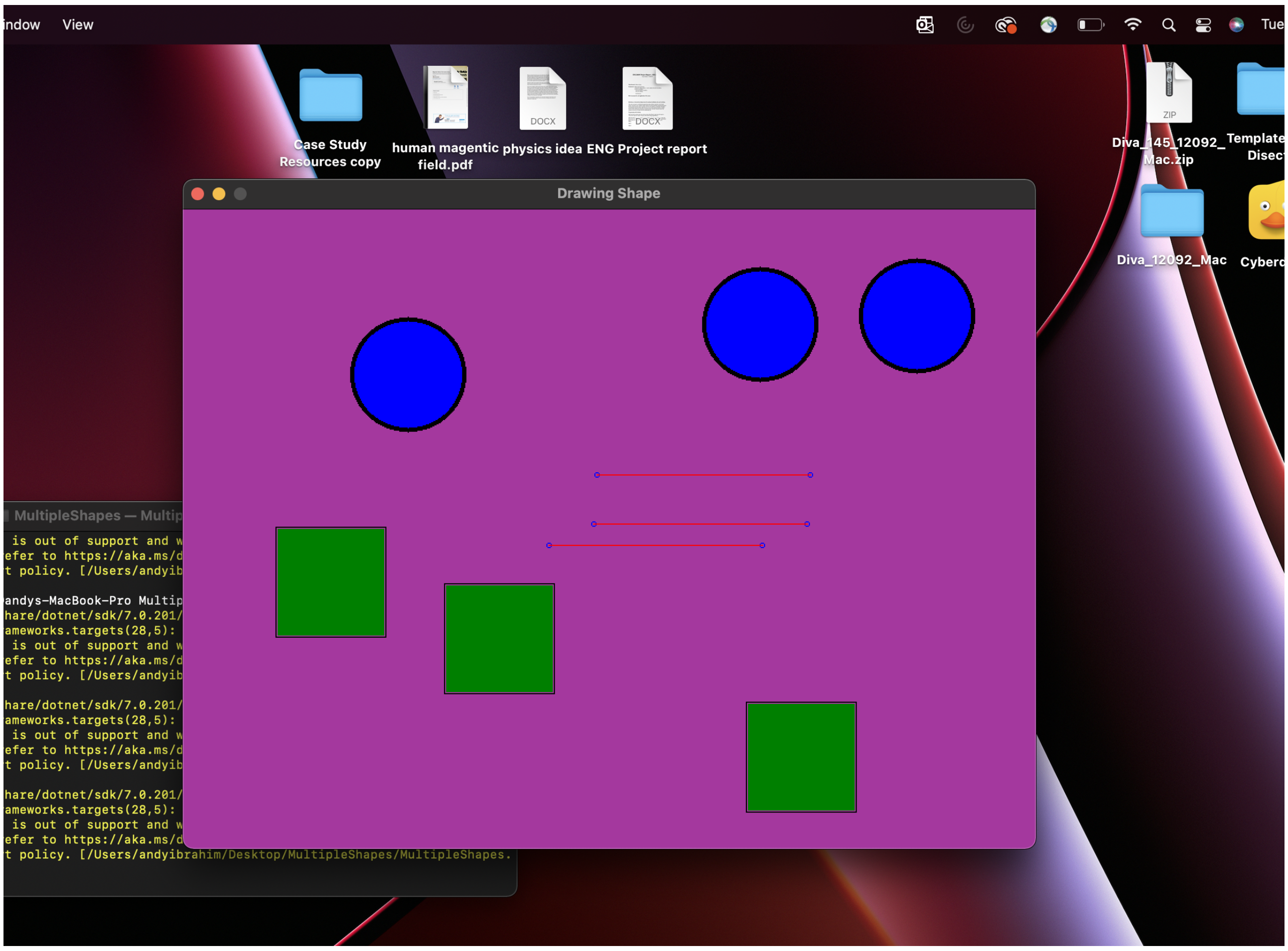
```
using System;
using System.Collections.Generic;
using SplashKitSDK;

namespace MultipleShapes
{
    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> shapes_selected = new List<Shape>();

                foreach (Shape s in _shapes)
                {
                    if (s.Selected == true)
                    {
                        shapes_selected.Add(s);
                    }
                }

                return shapes_selected;
            }
        }


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


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


        public void Draw()
        {
            SplashKit.ClearScreen(Background);

            foreach (Shape shape in _shapes)
            {
                shape.Draw();
            }

        }

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

            }
        }


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

        public void RemoveShape(Shape shape)
        {
            _shapes.Remove(shape);
        }

    }
}

```

```csharp
using System;
using SplashKitSDK;

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


        public Shape(Color clr)
        {
            this._color = clr;
            _x = 0;
            _y = 0;
            _selected = true;
        }


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


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


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

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


        public abstract void Draw();


        public abstract void DrawOutline();


        public abstract bool IsAt(Point2D pt);

    }
}

```

```csharp
using System;
using SplashKitSDK;

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


        public MyRectangle(Color clr, float x, float y, int width, int height): base
        ↪  (clr)
        {
            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);

        }

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

        }

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

        }


    }
}

```

```
using System;
using SplashKitSDK;

namespace MultipleShapes
{
    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)
            {
                DrawOutline();
            }

            SplashKit.FillCircle(Color, X, Y, _radius);

        }

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

        }

        public override bool IsAt(Point2D pt)
        {

            double a = (pt.X - X);
            double b = (pt.Y - Y);

            if (Math.Sqrt(a * a + b * b) < _radius)
            {
                return true;
            }
            return false;
        }

    }
}

```

```csharp
using System;
using SplashKitSDK;

namespace MultipleShapes
{
    public class MyLine: Shape
    {
        private int _length;


        public MyLine(Color clr, int length) : base(clr)
        {
            _length = length;
        }

        public MyLine() : this (Color.Red, 200) { }



        public override void Draw()
        {
            if (Selected)
            {
                DrawOutline();
            }
            SplashKit.DrawLine(Color, X, Y, X + _length, Y);
        }


        public override void DrawOutline()
        {
            SplashKit.DrawCircle(Color.Blue, X, Y, 2);
            SplashKit.DrawCircle(Color.Blue, X + _length, Y, 2);

        }

        public override bool IsAt(Point2D pt)
        {
            return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y, X + _length,
 ↪  Y));
        }

    }
}
```



```
 is out of support and w
efer to https://aka.ms/d
t policy. [/Users/andyib

andys-MacBook-Pro Multip
hare/dotnet/sdk/7.0.201/
ameworks.targets(28,5):
 is out of support and w
efer to https://aka.ms/d
t policy. [/Users/andyib

hare/dotnet/sdk/7.0.201/
ameworks.targets(28,5):
 is out of support and w
efer to https://aka.ms/d
t policy. [/Users/andyib

hare/dotnet/sdk/7.0.201/
ameworks.targets(28,5):
 is out of support and w
efer to https://aka.ms/d
t policy. [/Users/andyibrahim/Desktop/MultipleShapes/MultipleShapes.
```