Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Multiple Shape Kinds

PDF generated at 17:16 on Monday 4th September, 2023

```
using System;
using SplashKitSDK;
namespace Task4._1
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle, Circle, Line
        }

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

            Drawing dr = new Drawing();
            ShapeKind kindToAdd = ShapeKind.Circle;
            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape newShape;
                    if (kindToAdd == ShapeKind.Circle)
                    {

                        newShape = new MyCircle();



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



                    }
                    else
                    {
                        newShape = new MyLine();



                    }

                    newShape.X = SplashKit.MouseX();

                    newShape.Y = SplashKit.MouseY();

                    dr.AddShape(newShape);
                }

                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.RightButton))
                {

                    dr.SelectShapesAt(SplashKit.MousePosition());
                }

                if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                {

                    dr.Background1 = SplashKit.RandomRGBColor(255);
                }

                if (SplashKit.KeyTyped(KeyCode.DeleteKey))
                {
                    foreach (Shape s in dr.Selectedshapes)
                    {
                        dr.DeleteShape(s);
                    }
                }

                if (SplashKit.KeyTyped(KeyCode.BackspaceKey))
                {
                    foreach (Shape s in dr.Selectedshapes)
                    {
                        dr.DeleteShape(s);
                    }
                }


                dr.Draw();
                SplashKit.RefreshScreen();
            } while (!window.CloseRequested);
        }
    }

}


```

```
using System;
using SplashKitSDK;
using System.Collections.Generic;
namespace Task4._1

{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;
        public Drawing(Color bg)
        {
            _shapes = new List<Shape>();
            _background = bg;

        }
        public Drawing() : this(Color.White)
        {

        }
        public int ShapeCount
        {
            get
            {
                return _shapes.Count;
            }
        }
        public void AddShape(Shape s)
        {
            _shapes.Add(s);
        }
        public void Draw()
        {
            SplashKit.ClearScreen(Background1);
            foreach (Shape s in _shapes)
            {
                s.Draw();
            }

        }

        public void SelectShapesAt(Point2D pt)
        {
            foreach (Shape s in _shapes)
            {
                if (s.IsAt(pt))
                {
                    s.Selected = true;
                }
                else
                {
                    s.Selected = false;
                }
            }
        }
        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;
            }
        }
        public Color Background1
        {
            get
            {
                return _background;
            }
            set
            {
                _background = value;
            }
        }

        public void DeleteShape(Shape s)
        {
            _shapes.Remove(s);
        }
    }
}
```

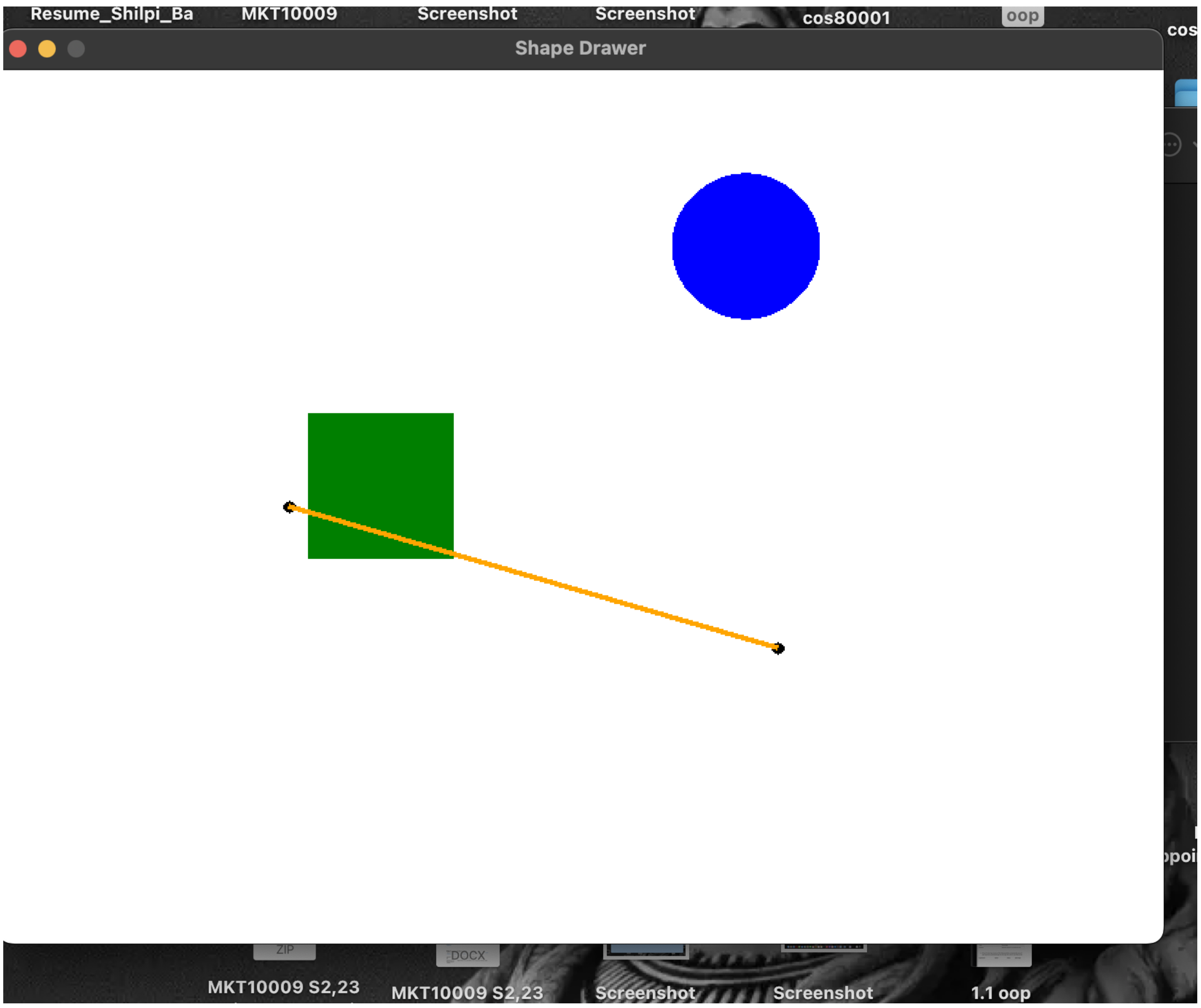
```
using System;

using SplashKitSDK;

namespace Task4._1
{
    public abstract class Shape
    {
        protected Color _color;
        private float _x;
        private float _y;

        private bool _selected;

        public Shape(Color c)
        {
            _color = c;
            _x = (float)0;
            _y = (float)0;

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

        }

        public abstract void Draw();

        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 bool Selected
        {
            get
            {
                return _selected;
            }
            set
            {
                _selected = value;
            }
        }
        public abstract bool IsAt(Point2D pt);

        public abstract void Outline();

}
}
```

```
using System;
using SplashKitSDK;

namespace Task4._1
{
    public class MyRectangle:Shape
    {
        private int _width;
        private int _height;
        public MyRectangle():this(Color.Green, 0, 0, 100, 100)
        {

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

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

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

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

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

        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;
            }
        }
    }
}

```

```
using System;
using SplashKitSDK;
namespace Task4._1
{
    public class MyCircle : Shape
    {
        private int _radius;

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

        }
        public MyCircle(Color clor,int x, int Y, int radius):base(clor)
        {
            _color = clor;
            _radius = radius;
        }

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

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

            SplashKit.FillCircle(Color.Black, X , Y , Radius+2);


        }
        public override bool IsAt(Point2D pt)
        {

            double point1 = (pt.X - X) * (pt.X - X);

            double point2 = (pt.Y - Y) * (pt.Y - Y);

            if(Math.Sqrt(point1+point2)<_radius)
            {
                return true;
            }
            else
            {
                return false;
            }

        }
    }
}

```

```
using System;
using System.Numerics;
using SplashKitSDK;
namespace Task4._1
{
    public class MyLine : Shape
    {
        private float _endX;
        private float _endY;


        public MyLine() : this(Color.Orange, 0, 0, 200, 300)
        {

        }
        public MyLine(Color clr, float startX, float startY, float endX, float endY)
↪   : base(clr)
        {

            EndX = endX;
            EndY = endY;

        }

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

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

        public override void Draw()
        {
            if (Selected)
                Outline();
            SplashKit.DrawLine(Color, X, Y, EndX, EndY,
↪   SplashKit.OptionLineWidth(5));
```

```

        }
        public override void Outline()
        {


            SplashKit.FillCircle(Color.Black, X, Y, 4);
            SplashKit.FillCircle(Color.Black, EndX, EndY, 4);

        }
        public override bool IsAt(Point2D pt)
        {


            Line l = SplashKit.LineFrom(X, Y, EndX, EndY);


            return SplashKit.PointOnLine(pt, l, 10);
        }


    }
}
```


Resume_Shilpi_Ba
MKT10009
Screenshot
Screenshot
cos80001
oop
Shape Drawer
MKT10009 S2,23
MKT10009 S2,23
Screenshot
Screenshot
1.1 oop