Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 4.1P: Drawing Program: Multiple Shape Kinds

Submitted By:
Wei Fa Yik
103838398
2022/04/19 10:18

Tutor:
Michael Kenny

April 19, 2022

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

namespace ShapeDrawer
{
    public class Program
    {
        private enum ShapeKind // declare a ShapeKind to findout what type of shape the user want
        {
            Rectangle,
            Circle,
            Line
        }

        public static void Main()
        {
            new Window("Shape Drawer", 800, 600);
            Drawing myDrawing = new Drawing(Color.White);
            ShapeKind kindToAdd = ShapeKind.Circle;

            while (!SplashKit.WindowCloseRequested("Shape Drawer"))
            {
                SplashKit.ProcessEvents();

                // when the left mouse button is pressed by the user it will create a new shape in a new position
                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape newShape;

                    if (kindToAdd == ShapeKind.Circle) // CHecks if Shapekind is a Circle
                    {
                        newShape = new MyCircle();
                    }

                    else if (kindToAdd == ShapeKind.Rectangle) // Checks of the shapekind is a rectangle
                    {
                        newShape = new MyRectangle();
                    }

                    else
                    {
                        newShape = new MyLine(); // else its going to be a line

                    }

                    newShape.X = SplashKit.MouseX();
                    newShape.Y = SplashKit.MouseY();
```

```
50                  myDrawing.AddShape(newShape); // adds new shapes
51              }
52
53              if (SplashKit.KeyTyped(KeyCode.RKey)) // if the user presses the R
                ↪ key it will create a Rectange
54              {
55                  kindToAdd = ShapeKind.Rectangle; // displays the Rectangle
                    ↪ shape
56              }
57
58              else if (SplashKit.KeyTyped(KeyCode.CKey)) // if the user presses
                ↪ the C key it will create a Circle
59              {
60                  kindToAdd = ShapeKind.Circle; // displays new circle shape
61              }
62
63              else if (SplashKit.KeyTyped(KeyCode.LKey)) // if user presses the L
                ↪ key then it will create a Line
64              {
65                  kindToAdd = ShapeKind.Line; // display new line
66              }
67
68              if (SplashKit.KeyTyped(KeyCode.SpaceKey)) //checks if the user
                ↪ presses the spacebar
69              {
70                  myDrawing.Background = SplashKit.RandomColor(); //if user
                    ↪ presses the spacebar the background color will change to a
                    ↪ random color
71              }
72
73              if (SplashKit.MouseClicked(MouseButton.RightButton)) // checks if
                ↪ the user rightclicks on a random shape or line
74              {
75                  myDrawing.SelectShapesAt(SplashKit.MousePosition()); // if user
                    ↪ presses the right click button it will select that specific
                    ↪ shape
76              }
77
78              if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
                ↪ SplashKit.KeyTyped(KeyCode.DeleteKey)) //Checks if the user
                ↪ presses backspacekey or the deletekey
79              {
80                  foreach (Shape s in myDrawing.SelectedShapes) //for loop to
                    ↪ check how many shapes have been selected by the user
81                  {
82                      myDrawing.RemoveShape(s); // Delete those selected shapes
83                  }
84              }
85
86              SplashKit.ClearScreen(myDrawing.Background);
87              myDrawing.Draw();
88              SplashKit.RefreshScreen(60);
89          }
```

```
90              }
91          }
92      }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using SplashKitSDK;
using System.Threading.Tasks;

namespace ShapeDrawer
{
    public class Drawing // Drawing class
    {
        private readonly List<Shape> _shapes; //private read only field to store the list of shapes

        private Color _background; //private background Color field

        public Drawing(Color background) // Constructor takes in background color as parameter
        {
            _shapes = new List<Shape>(); // object stored in _shapes field
            _background = background; // Initialise background to supply background color
        }

        public Drawing() : this(Color.White)
        {
            _shapes = new List<Shape>();
        }

        public List<Shape> SelectedShapes // SelectedShapes property
        {
            get
            {
                List<Shape> result = new List<Shape>();

                foreach (Shape s in _shapes) //for loop
                {
                    if (s.Selected)
                        result.Add(s);
                }

                return result;
            }
        }

        // methods to find the number of shapes
        public int ShapeCount
        {
            get
            {
                return _shapes.Count;
            }
        }
```

```
51
52          public Color Background // baackground property
53          {
54              get
55              {
56                  return _background;
57              }
58
59              set
60              {
61                  _background = value;
62              }
63          }
64
65          public void Draw() //draw method
66          {
67              foreach (Shape s in _shapes) // for loop
68              {
69                  if (s.Selected)
70                      s.DrawOutLine();
71                  s.Draw();
72              }
73          }
74
75          public void SelectShapesAt(Point2D pt) // SelectShapesAt method
76          {
77              foreach (Shape s in _shapes) // for loop
78              {
79                  if (s.IsAt(pt)) // IsAt checks if the mouse click is in the shape
80                      s.Selected = true;
81                  else
82                      s.Selected = false;
83              }
84          }
85
86
87          public void AddShape(Shape s)// method to add more shapes
88          {
89              _shapes.Add(s);
90          }
91          public void RemoveShape(Shape s) // method to remove shapes
92          {
93              _shapes.Remove(s);
94          }
95      }
96  }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

namespace ShapeDrawer
{
    public abstract class Shape
    {
        private Color _color; //variable to set the color of the shape
        private float _x, _y; // variable to set the position of the shape
        private bool _selected = false;

        public Shape(Color color)
        {
            _color = Color.Red; //shape color set to red

            _x = 0; //shape position
            _y = 0;
        }

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

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

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

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

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

        public abstract void Draw();
        public abstract void DrawOutLine();
        public abstract bool IsAt(Point2D pt); // IsAt method

    }
}
```

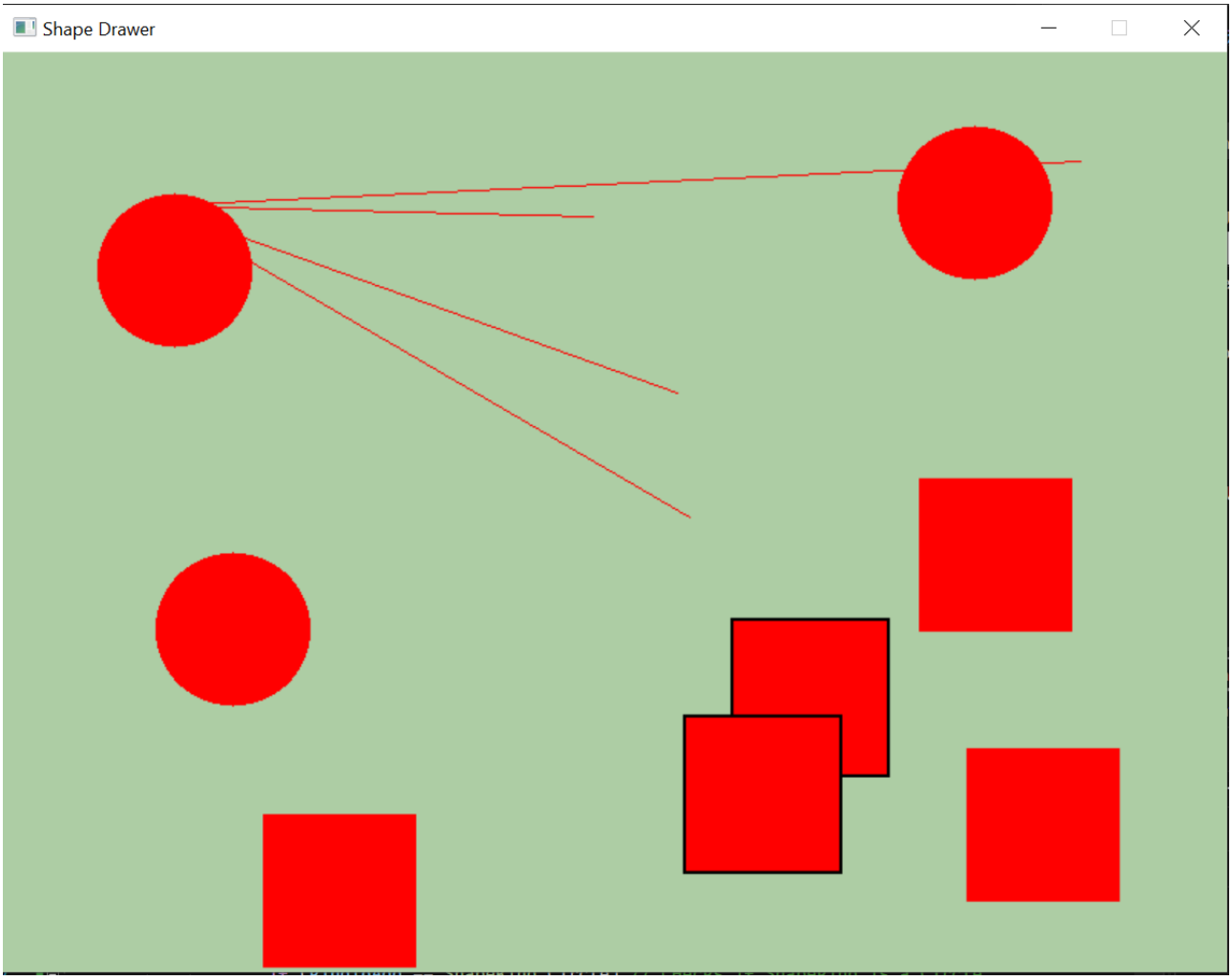
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class MyRectangle : Shape //Set MyRectangle to inherit all features of the Shape class
    {
        private int _width, _height; // variables to set the shape size

        //MyRectanngle Constructor
        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 // Width property
        {
            get
            {
                return _width;
            }

            set
            {
                _width = value;
            }
        }

        public int Height //Height property
        {
            get
            {
                return _height;
            }

            set
            {
                _height = value;
            }
        }
        public override void Draw() //Override the Draw method
        {
```

```
52              SplashKit.FillRectangle(Color, X, Y, _width, _height);
53          }
54
55          public override void DrawOutLine() //Override the DrawOutLine method
56          {
57              SplashKit.FillRectangle(Color.Black, (X - 2), (Y - 2), (_width + 4),
                ↪  (_height + 4));
58          }
59
60          public override bool IsAt(Point2D pt) //Override the IsAt method
61          {
62              // From the given X, Y, width and height values it will return a
                ↪  rectangle
63              Rectangle rectangle = SplashKit.RectangleFrom(X, Y, _width, _height);
64
65              if(SplashKit.PointInRectangle(pt,rectangle))
66                  return true;
67              else
68                  return false;
69          }
70
71      }
72  }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class MyCircle : Shape
    {
        private int _radius;
        public int Radius { get => _radius; set => _radius = value; }

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

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

        public override void Draw() //override Draw method
        {
            if (Selected)
                DrawOutLine();
            SplashKit.FillCircle(Color, X, Y, _radius);

        }

        public override void DrawOutLine() // Override DrawOutLine method
        {
            SplashKit.FillCircle(Color.Black, X, Y, (_radius + 2));

        }

        public override bool IsAt(Point2D pt) // override IsAt method
        {
            // With the given X, Y, radius of the circle it will Return a Circle
            ↪  Shape)
            Circle circle = SplashKit.CircleAt(X, Y, _radius);
            if (SplashKit.PointInCircle(pt, circle))
                return true;
            else
                return false;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

namespace ShapeDrawer

{
   public class MyLine : Shape
    {
        private float _endX; // variable to set X end of the line
        private float _endY; // variable to set Y end of the line

        public MyLine(Color clr, float startX, float startY, float endX, float
        ↪  endY) : base (clr)
        {
            X = startX;
            Y = startY;
            _endX = endX;
            _endY = endY;
        }

        public MyLine(): this(Color.Red, 0, 0, 100, 100) { }

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

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

        public override void Draw() //Override Draw method
        {
            SplashKit.DrawLine(Color, X, Y, _endX, _endY);
        }

        public override void DrawOutLine() //Override the DrawOutLine method
        {
            SplashKit.FillCircle(Color.Black, X, Y, 4);
            SplashKit.FillCircle(Color.Black,_endX, _endY, 4);
        }

        public override bool IsAt(Point2D pt) // override the IsAt method
        {
            Line Line = SplashKit.LineFrom(X, Y, _endX, _endY);

```

```
53              if (SplashKit.PointOnLine(pt,Line))
54                  return true;
55              else
56                  return false;
57          }
58      }
59  }
```


Shape Drawer