Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Multiple Shape Kinds

PDF generated at 11:56 on Tuesday 29th August, 2023

```
using System;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }
        public static void Main()
        {
            Drawing myDrawing = new Drawing();
            ShapeKind kindToAdd = ShapeKind.Circle;

            new Window("Drawing Shape", 800, 600);
            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();
                if (SplashKit.KeyTyped(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
                }
                if (SplashKit.KeyTyped(KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line;
                }
                if (SplashKit.KeyTyped(KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle;
                }
                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape newShape;
                    if (kindToAdd == ShapeKind.Circle)
                    {
                        MyCircle newCircle = new MyCircle();
                        newCircle.X = SplashKit.MouseX();
                        newCircle.Y = SplashKit.MouseY();
                        newShape = newCircle;

                    }
                    else if (kindToAdd == ShapeKind.Rectangle)
                    {
                        MyRectangle newRect = new MyRectangle();
                        newRect.X = SplashKit.MouseX();
                        newRect.Y = SplashKit.MouseY();
                        newShape = newRect;
                    }
```

```
54                      else
55                      {
56                          MyLine newLine = new MyLine();
57                          newLine.X = SplashKit.MouseX();
58                          newLine.Y = SplashKit.MouseY();
59                          newShape = newLine;
60                      }
61                      myDrawing.AddShape(newShape);
62                  }
63
64                  if (SplashKit.MouseClicked(MouseButton.RightButton))
65                  {
66                      myDrawing.SelectedShapeAt(SplashKit.MousePosition());
67                  }
68
69                  if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
    ↪  SplashKit.KeyTyped(KeyCode.DeleteKey))
70                  {
71                      myDrawing.RemoveShape();
72                  }
73
74                  if (SplashKit.KeyTyped(KeyCode.SpaceKey))
75                  {
76                      myDrawing.Background = SplashKit.RandomRGBColor(255);
77                  }
78
79                  myDrawing.Draw();
80
81                  SplashKit.RefreshScreen();
82
83              } while (!SplashKit.WindowCloseRequested("Drawing Shape"));
84          }
85      }
86  }
```

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

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> SelectedShapes()
        {
            List<Shape> _selectedShapes = new List<Shape>();
            foreach (Shape s in _selectedShapes)
            {
                if (s.Selected)
                {
                    _selectedShapes.Add(s);
                }
            }
            return _selectedShapes;
        }

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

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

```
54          }
55
56          public void Draw()
57          {
58              SplashKit.ClearScreen(_background);
59
60              foreach (Shape s in _shapes)
61              {
62                  s.Draw();
63              }
64          }
65
66          public void SelectedShapeAt(Point2D pt)
67          {
68              foreach (Shape s in _shapes)
69              {
70                  if (s.IsAt(pt))
71                  {
72                      s.Selected = true;
73                  }
74                  else
75                  {
76                      s.Selected = false;
77                  }
78              }
79          }
80
81          public void AddShape(Shape s)
82          {
83              _shapes.Add(s);
84          }
85
86          public void RemoveShape()
87          {
88              foreach (Shape s in _shapes.ToList())
89              {
90                  if (s.Selected)
91                  {
92                      _shapes.Remove(s);
93                  }
94              }
95          }
96      }
97  }
```

```
using System;
using SplashKitSDK;

namespace ShapeDrawer
{
    public abstract class Shape
    {
        private Color _color;
        private float _x, _y;
        private bool _selected;
        private int _witdh, _height;
        public Shape(Color clr)
        {
            _color = clr;
        }

        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 abstract void Draw();
        public abstract bool IsAt(Point2D p);
```

```
54
55
56          public bool Selected
57          {
58              get
59              {
60                  return _selected;
61              }
62              set
63              {
64                  _selected = value;
65              }
66          }
67
68          public abstract void DrawOutline();
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 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 // Corrected typo
        {
            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.FillRectangle(Color, X - 2, Y - 2, Width + 4, Height + 4);
        }
        public override bool IsAt(Point2D p)
        {
            return SplashKit.PointInRectangle(p, SplashKit.RectangleFrom(X, Y, Width,
    Height));
        }
    }
}
```

```
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 MyCircle(Color clr, int radius) : base(clr)
        {
            _radius = radius;
        }
        public MyCircle() : this(Color.Blue, 50) { }
        public int Radius { get { return _radius; } }
        public override void Draw()
        {
            if (Selected)
                DrawOutline();
            SplashKit.FillCircle(Color, X, Y, _radius);
        }
        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color, X - 2, Y - 2, _radius + 2);
        }
        public override bool IsAt(Point2D p)
        {
            double a = (double)(p.X - X);
            double b = (double)(p.Y - Y);
            if (Math.Sqrt(a * a + b * b) < _radius)
            {
                return true;
            }
            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 int _length;

        public MyLine(Color clr, int lenght) : base(clr)
        {
            _length = lenght;
        }
        public MyLine() : this(Color.RandomRGB(255), 100) { }

        public int Length
        {
            get { return _length; }
            set { _length = value; }
        }

        public override void Draw()
        {
            if (Selected)
            {
                DrawOutline();
            }
            SplashKit.DrawLine(Color, X, Y, X + _length + 3, Y);
        }
        public override void DrawOutline()
        {
            SplashKit.DrawLine(Color, X + 1, Y + 1 , _length + 5, Y);
        }
        public override bool IsAt(Point2D p)
        {
            return SplashKit.PointOnLine(p, SplashKit.LineFrom(X, Y, X + _length,
↪   Y));
        }
    }
}
```


Drawing Shape