Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Multiple Shape Kinds

PDF generated at 14:41 on Tuesday 15th August, 2023

```
using System;
using SplashKitSDK;
using DrawingProgram;

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

        public static void Main(string[] args)
        {
            Drawing drawing = new Drawing();
            ShapeKind kindToAdd = ShapeKind.Rectangle;
            Window window = new Window("Drawing Shape 4.1P", 800, 600);
            while(!window.CloseRequested)
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();
                if (SplashKit.KeyDown(KeyCode.RKey))
                    kindToAdd = ShapeKind.Rectangle;
                if (SplashKit.KeyDown(KeyCode.CKey))
                    kindToAdd = ShapeKind.Circle;
                if (SplashKit.KeyDown(KeyCode.LKey))
                    kindToAdd = ShapeKind.Line;
                if(SplashKit.MouseDown(MouseButton.LeftButton))
                {
                    switch(kindToAdd)
                    {
                        case(ShapeKind.Rectangle):
                            MyRectangle rectangle = new MyRectangle();
                            rectangle.X = SplashKit.MouseX();
                            rectangle.Y = SplashKit.MouseY();
                            drawing.AddShape(rectangle);
                            break;
                        case (ShapeKind.Circle):
                            MyCircle circle = new MyCircle();
                            circle.X = SplashKit.MouseX();
                            circle.Y = SplashKit.MouseY();
                            drawing.AddShape(circle);
                            break;
                        case (ShapeKind.Line):
                            MyLine line = new MyLine();
                            line.X = SplashKit.MouseX();
                            line.Y = SplashKit.MouseY();
                            drawing.AddShape(line);
                            break;

```

```
54                      }
55                  }
56
57                  if(SplashKit.MouseDown(MouseButton.RightButton))
58                  {
59                      drawing.SelectShapesAt(SplashKit.MousePosition());
60                  }
61
62                  if(SplashKit.KeyDown(KeyCode.SpaceKey))
63                  {
64                      drawing.Background = SplashKit.RandomRGBColor(255);
65                  }
66
67                  if(SplashKit.KeyDown(KeyCode.EscapeKey))
68                  {
69                      foreach(Shape s in drawing.SelectedShapes)
70                      {
71                          s.color = SplashKit.RandomRGBColor(255);
72                      }
73                  }
74
75                  if(SplashKit.KeyDown(KeyCode.DeleteKey) ||
    ↪ SplashKit.KeyDown(KeyCode.BackspaceKey))
76                  {
77                      foreach(Shape s in drawing.SelectedShapes)
78                      {
79                          drawing.RemoveShape(s);
80                      }
81                  }
82                  drawing.Draw();
83                  SplashKit.RefreshScreen();
84              }
85
86          }
87      }
88  }
```

```
using System;
using SplashKitSDK;
using System.Collections.Generic;
namespace DrawingProgram
{
    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> _selectedShapes = new List<Shape>();
                foreach(Shape s in _shapes)
                {
                    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          public void Draw()
56          {
57              SplashKit.ClearScreen(_background);
58              foreach(Shape s in _shapes)
59              {
60                  s.Draw();
61              }
62
63          }
64          public void SelectShapesAt(Point2D point)
65          {
66              foreach(Shape s in _shapes)
67              {
68                  if (s.IsAt(point))
69                      s.Selected = true;
70                  else
71                      s.Selected = false;
72              }
73          }
74          public void AddShape(Shape s)
75          {
76              _shapes.Add(s);
77          }
78          public void RemoveShape(Shape s)
79          {
80              _shapes.Remove(s);
81          }
82      }
83  }
84
```

```
using System;
using SplashKitSDK;

namespace DrawingProgram
{
    public abstract class Shape
    {
        private Color _color;
        private float _x, _y;
        private bool _selected;
        private int _width, _height;

        public Shape(Color color)
        {
            _color = color;
            _x = 0;
            _y = 0;
            _width = 100;
            _height = 100;

        }

        public Shape() : this(Color.RGBAColor(255,0, 255, 255)) { }

        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 int Width
        {
            get
            {
                return _width;
            }
            set
            {
                _width = value;
            }
        }

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

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

        public abstract void Draw();
        public abstract bool IsAt(Point2D pt);
        public abstract void DrawOutline();
    }
}

```

```
using System;
using SplashKitSDK;

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

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

        public MyRectangle() : this(Color.RandomRGB(255), 0, 0, 100, 100) { }

        public new int Width
        {
            get
            {
                return _width;
            }
            set
            {
                _width = value;
            }
        }

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

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

        public override bool IsAt(Point2D pt)
```

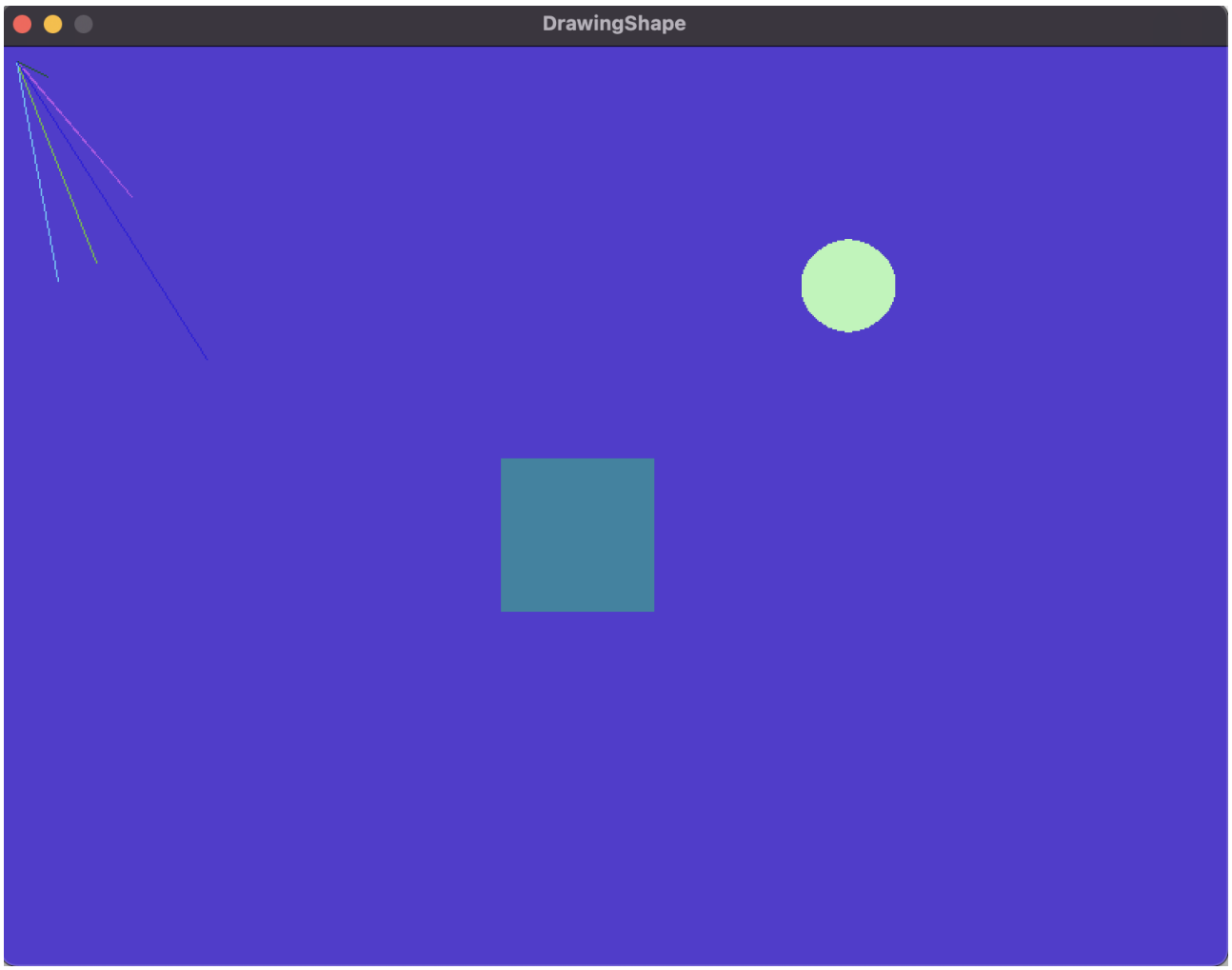
```
53          {
54              return SplashKit.PointInRectangle(pt, SplashKit.RectangleFrom(X, Y,
    ↪  _width, _height));
55          }
56
57          public override void DrawOutline()
58          {
59              SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, _width + 4, _height +
    ↪  4);
60          }
61      }
62  }
63
```

```
using System;
using SplashKitSDK;

namespace DrawingProgram
{
    public class MyCircle : Shape
    {
        private int _radius;

        public MyCircle(Color color, float x, float y, int radius) : base(color)
        {
            X = x;
            Y = y;
            _radius = radius;
        }

        public MyCircle() : this(Color.RandomRGB(255), 0, 0, 30)
        {
        }

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

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

        public override bool IsAt(Point2D pt)
        {
            //c = sqrt(a^2 + b^2)
            double a = pt.X - X;
            double b = pt.Y - Y;
            double c = Math.Sqrt((a * a) + (b * b));
            return c < _radius;
        }

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

```
54            }
55        }
56    }
57
```

```
using System;
using SplashKitSDK;

namespace DrawingProgram
{
    public class MyLine : Shape
    {
        private float _endX, _endY;

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

        public MyLine() : this(Color.RandomRGB(255), 0, 0, 10, 10)
        {
        }

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

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



        public override void Draw()
        {
            if (Selected)
                DrawOutline();
            SplashKit.DrawLine(color, X, Y, _endX, _endY);
```

```
53          }
54
55          public override void DrawOutline()
56          {
57              SplashKit.DrawLine(color, X, Y, _endX + 5, _endY + 5);
58          }
59
60          public override bool IsAt(Point2D pt)
61          {
62              return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y, _endX, _endY));
63          }
64      }
65  }
66
```


DrawingShape