Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Multiple Shape Kinds

PDF generated at 01:31 on Friday 10th November, 2023

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

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

        public static void Main()
        {
            Window window = new Window("Shape Drawer", 800, 600);
            ShapeKind kindToAdd = ShapeKind.Circle;
            Drawing draw = new Drawing();
            int clicked = 0;

            List<MyLine> lines = new List<MyLine>();

            do
            {
                SplashKit.ProcessEvents();
                draw.Draw();

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

```
54                      {
55                          newShape = new MyCircle();
56                      }
57                      else
58                      {
59                          clicked++;
60                          newShape = new MyLine();
61                          if (clicked == 1)
62                          {
63                              MyLine line1 = new MyLine();
64                              line1.X = SplashKit.MouseX();
65                              line1.Y = SplashKit.MouseY();
66                              lines.Add(line1);
67                          }
68
69                          if (clicked == 2)
70                          {
71                              MyLine line2 = new MyLine();
72                              line2.X = SplashKit.MouseX();
73                              line2.Y = SplashKit.MouseY();
74                              lines.Add(line2);
75                              clicked = 0;
76                          }
77                      }
78
79                      if (kindToAdd == ShapeKind.Circle || kindToAdd ==
    ↪ ShapeKind.Rectangle)
80                      {
81                          newShape.X = SplashKit.MouseX();
82                          newShape.Y = SplashKit.MouseY();
83                      }
84                      else
85                      {
86                          if (lines.Count == 2)
87                          {
88                              newShape = lines[0] + lines[1];
89                              lines.Clear();
90                          }
91                      }
92                      draw.AddShape(newShape);
93                  }
94
95                  if (SplashKit.MouseClicked(MouseButton.RightButton))
96                  {
97                      draw.SelectShapeAt(SplashKit.MousePosition());
98                  }
99
100                 if (SplashKit.KeyTyped(KeyCode.SpaceKey))
101                 {
102                     draw.Background = SplashKit.RandomRGBColor(255);
103                 }
104
105                 if (SplashKit.KeyTyped(KeyCode.BackspaceKey))
```

```
106                 {
107                     foreach (Shape s in draw.SelectedShapes)
108                     {
109                         draw.RemoveShape(s);
110                     }
111                 }
112 
113 
114                 SplashKit.RefreshScreen();
115 
116             } while (!window.CloseRequested);
117         }
118     }
119 }
```

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

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)
        {
            _shapes = new List<Shape>();
        }

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

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

        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 void AddShape(Shape newShape)
```

```
54          {
55              _shapes.Add(newShape);
56          }
57
58          public void Draw()
59          {
60              SplashKit.ClearScreen(_background);
61              foreach (Shape shape in _shapes)
62              {
63                  shape.Draw();
64              }
65          }
66
67          public void SelectShapeAt(Point2D pt)
68          {
69              foreach (Shape s in _shapes)
70              {
71                  if (s.IsAt(pt))
72                  {
73                      s.Selected = true;
74                  }
75              }
76          }
77
78          public void RemoveShape(Shape shape)
79          {
80              _shapes.Remove(shape);
81          }
82      }
83
84  }
```

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

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

        public Shape(Color color)
        {
            _color = color;
            _x = 0;
            _y = 0;
            _selected = false;
        }

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

        }

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

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

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

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

        public abstract void Draw();
```

```
54
55          public abstract bool IsAt(Point2D pt);
56
57          public abstract void DrawOutline();
58      }
59
60  }
```

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

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

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

        public MyRectangle() : this(Color.Green, 0, 0, 100, 100)
        {

        }

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

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

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

        public override void DrawOutline()
        {
            SplashKit.DrawRectangle(Color.Black, (X - 2), (Y - 2), (Width + 4),
↪   (Height + 4));
        }
```

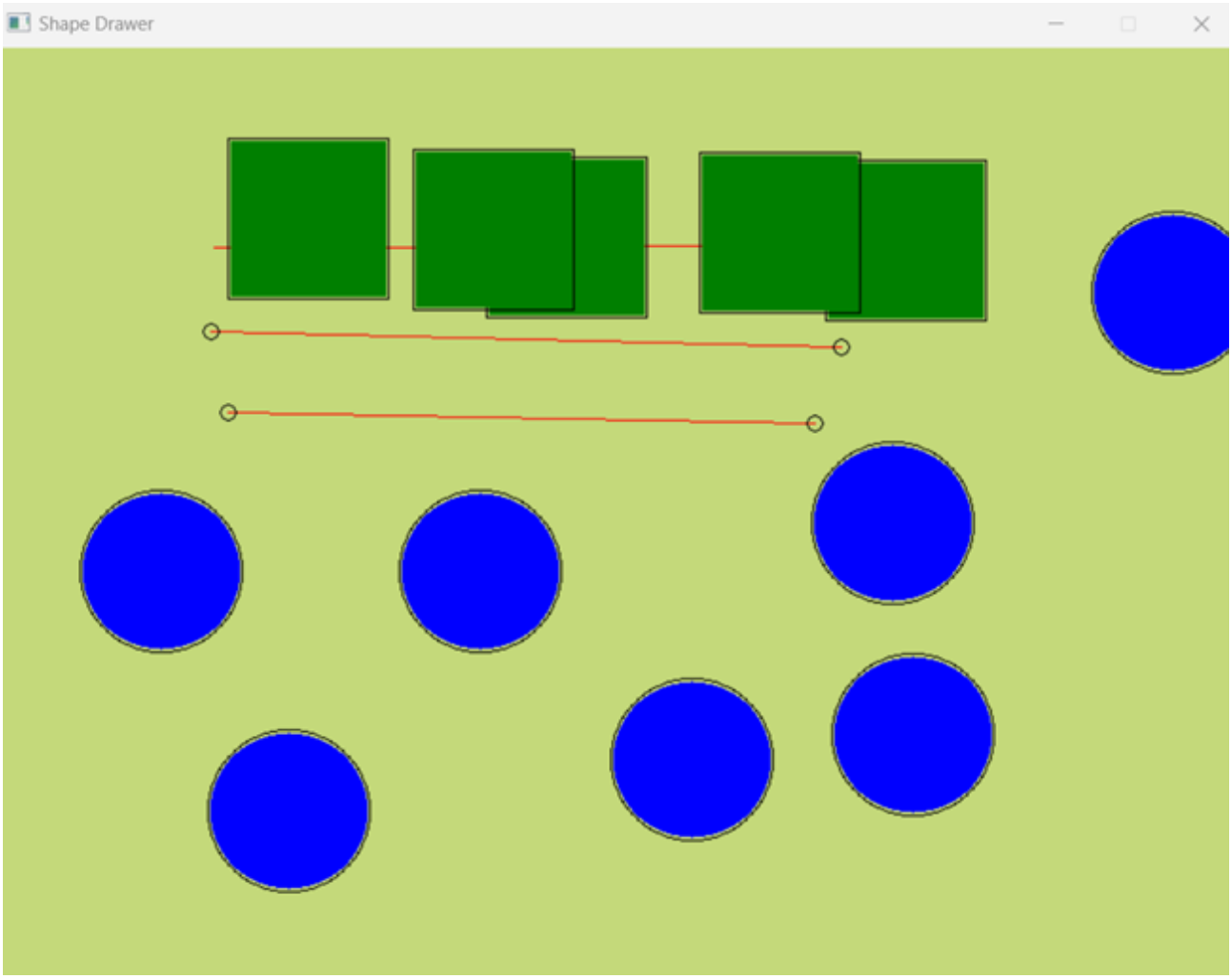
```
52
53          public override bool IsAt(Point2D pt)
54          {
55              if ((X < pt.X) && (X + _width > pt.X) && (Y < pt.Y) && (Y + _height >
    ↪  pt.Y))
56              {
57                  return true;
58              }
59              else
60              {
61                  return false;
62              }
63          }
64      }
65  }
```

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

namespace ShapeDrawer
{
    public class MyCircle : Shape
    {
        int _radius;

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

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

        }

        public int Radius
        {
            get { return _radius; }
        }

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

        public override void DrawOutline()
        {
            SplashKit.DrawCircle(Color.Black, X, Y, Radius + 2);
        }

        public override bool IsAt(Point2D pt)
        {
            if (((X - pt.X)*(X - pt.X)) + ((Y - pt.Y)*(Y - pt.Y)) <=
↪   (_radius*_radius))
            {
                return true;
            }
            else
            {
                return false;
            }
```

```
53            }
54        }
55    }
```

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

namespace ShapeDrawer
{
    internal class MyLine : Shape
    {
        float _startX, _startY, _endX, _endY;

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

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

        }


        public float StartX
        {
            get { return _startX; }
            set { _startX = value; }
        }

        public float StartY
        {
            get { return _startY; }
            set { _startY = value; }
        }

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

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

        public override void Draw()
```

```
53          {
54              SplashKit.DrawLine(color, _startX, _startY, _endX, _endY);
55              if(Selected == true)
56              {
57                  DrawOutline();
58              }
59          }
60
61          public override void DrawOutline()
62          {
63              SplashKit.DrawCircle(Color.Black, _startX, _startY, 5);
64              SplashKit.DrawCircle(Color.Black, _endX, _endY, 5);
65          }
66
67          public override bool IsAt(Point2D pt)
68          {
69              Point2D start = new Point2D { X = _startX, Y = _startY };
70              Point2D end = new Point2D { X = _endX, Y = _endY };
71
72              Line line = new Line { StartPoint = start, EndPoint = end };
73
74              if (SplashKit.PointOnLine(pt, line, 1000))
75              {
76                  return true;
77              }
78
79              return false;
80
81          }
82
83          public static MyLine operator +(MyLine start, MyLine end)
84          {
85              MyLine newLine = new MyLine();
86              newLine.StartX = start.X;
87              newLine.StartY = start.Y;
88              newLine.EndX = end.X;
89              newLine.EndY = end.Y;
90              return newLine;
91          }
92      }
93  }
```


Shape Drawer