Swinburne University of Technology

COS20007 Object Oriented Programming

4.1P - Drawing Program - Multiple Shape Kinds

PDF generated at 15:13 on Friday 14th April, 2023

```
using System;
using SplashKitSDK;

namespace DrawingProgram
{
    public class Program
    {

        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }

        public static void Main()
        {
            ShapeKind kindToAdd = ShapeKind.Circle;

            Drawing drawing = new Drawing();

            Window window = new Window("Shape Drawer", 800, 600);

            //event loop
            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();

                //shape depending on key
                if (SplashKit.KeyTyped(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
                }
                if (SplashKit.KeyTyped(KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle;
                }
                if (SplashKit.KeyTyped(KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line;
                }

                //new shape
                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape ShapeDrawn;
                    if (kindToAdd == ShapeKind.Rectangle)
                    {
                        MyRectangle myRectangle = new();
                        ShapeDrawn = myRectangle;
                    }
                    else if (kindToAdd == ShapeKind.Circle)
```

```
54                      {
55                          MyCircle myCircle = new();
56                          ShapeDrawn = myCircle;
57                      }
58                      else
59                      {
60                          MyLine myLine = new();
61                          ShapeDrawn = myLine;
62                      }
63                      ShapeDrawn.X = SplashKit.MouseX();
64                      ShapeDrawn.Y = SplashKit.MouseY();
65                      drawing.AddShape(ShapeDrawn);
66                  }
67                  //delete
68                  if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
    ↪  SplashKit.KeyTyped(KeyCode.DeleteKey))
69                  {
70                      foreach (Shape s in drawing.SelectedShapes())
71                      {
72                          drawing.RemoveShape(s);
73                      }
74                  }
75                  //select
76                  if (SplashKit.MouseClicked(MouseButton.RightButton))
77                  {
78                      drawing.SelectShapesAt(SplashKit.MousePosition());
79                  }
80                  //background color
81                  if (SplashKit.KeyTyped(KeyCode.SpaceKey))
82                  {
83                      drawing.Background = SplashKit.RandomRGBColor(255);
84                  }
85                  drawing.Draw();
86                  SplashKit.RefreshScreen();
87              } while (!window.CloseRequested);
88          }
89      }
90  }
91
92
93
```

```
using System;
using SplashKitSDK;
using System.Linq;
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;
        }

        //default constructor
        public Drawing() : this(Color.White)
        {

        }

        //list of currently selected shapes
        public List<Shape> SelectedShapes()
        {
            List<Shape> result = new List<Shape>();
                foreach (Shape s in _shapes)
                {
                    if (s.Selected == true)
                    {
                        result.Add(s);
                    }
                }
                return result;
        }

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

        //background color
        public Color Background
        {
            get
            {
                return _background;
```

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

```
using System;
using SplashKitSDK;


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

        //constructor
        public Shape(Color clr)
        {
            _color = clr;
        }

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

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

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

        public float Y
        {
            get
            {
                return _y;
```

```
54              }
55              set
56              {
57                  _y = value;
58              }
59          }
60
61          public bool Selected
62          {
63              get
64              {
65                  return _selected;
66              }
67              set
68              {
69                  _selected = value;
70              }
71          }
72
73          //methods
74          public abstract void Draw();
75          public abstract bool IsAt(Point2D pt);
76          public abstract void DrawOutLine();
77      }
78  }
```

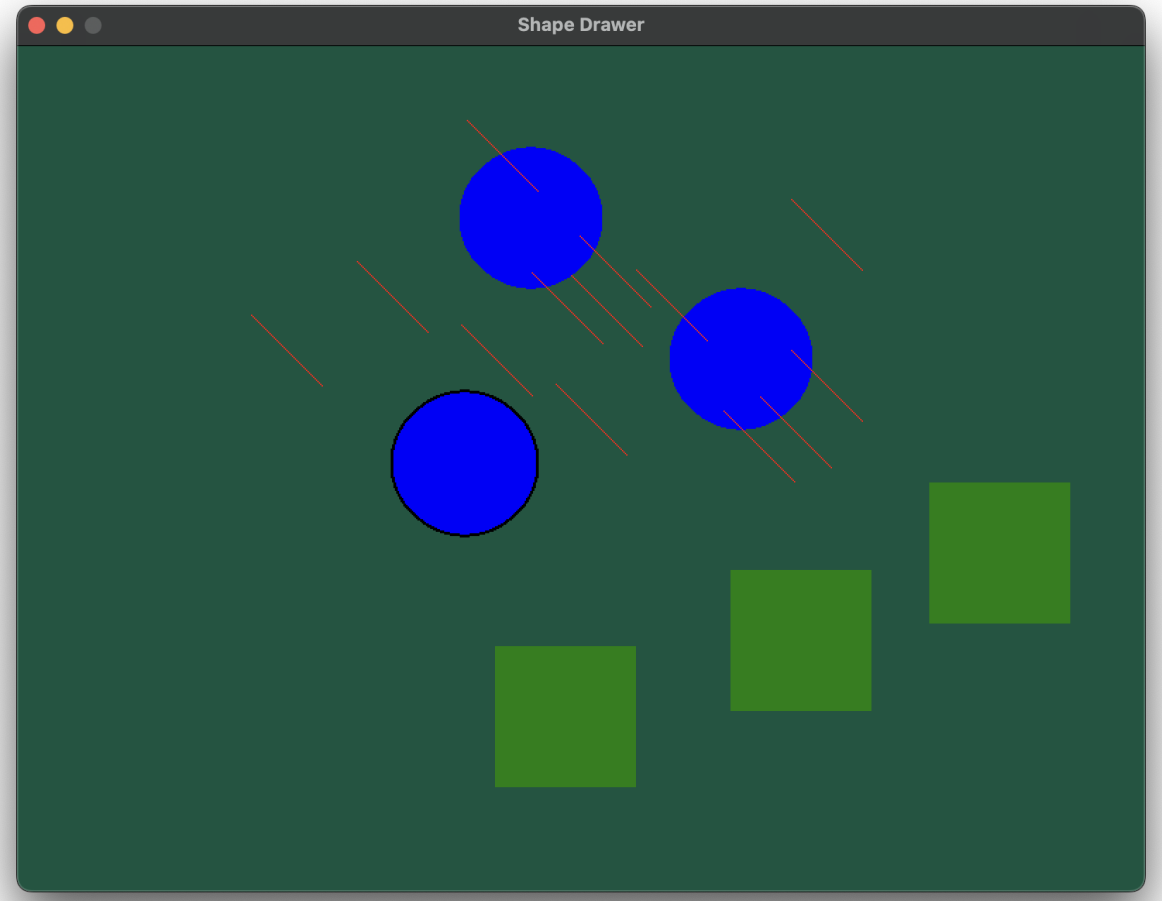
```
using SplashKitSDK;

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

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

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


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


        //default constructor
        public MyRectangle() : this(Color.Green, 0, 0, 100, 100) { }


        //methods
        public override void Draw()
        {
            if (Selected)
            {
```

```
53                  DrawOutLine();
54              }
55              SplashKit.FillRectangle(COLOR, X, Y, Width, Height);
56          }
57
58          public override void DrawOutLine()
59          {
60              SplashKit.FillRectangle(Color.Black, X -2, Y - 2, _width + 4, _height +
    ↪ 4);
61          }
62
63          public override bool IsAt(Point2D pt)
64          {
65              return SplashKit.PointInRectangle(pt, SplashKit.RectangleFrom(X, Y,
    ↪ Width, Height));
66          }
67      }
68  }
69
```

```
using System;
using SplashKitSDK;

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

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

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

        //methods
        public override void Draw()
        {
            if (Selected)
            {
                DrawOutLine();
            }
            SplashKit.FillCircle(COLOR, X, Y, _radius);
        }

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

        public override bool IsAt(Point2D pt)
        {
            Circle circle = new Circle()
            {
                Center = new Point2D()
                {
                    X = X,
                    Y = Y,
                },
                Radius = _radius
            };
            return SplashKit.PointInCircle(pt, circle);
        }
    }
}

```

```
using System;
using SplashKitSDK;

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

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

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

        public MyLine(Color clr, float endX, float endY) : base(clr)
        {
            _endX = SplashKit.MouseX() +50;
            _endY = SplashKit.MouseY() +50;
        }

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


        public override void Draw()
        {
            if (Selected)
            {
                DrawOutLine();
            }
            SplashKit.DrawLine(COLOR, X, Y, EndX, EndY);
        }

        public override void DrawOutLine()
```

```
54          {
55              int radius = 2;
56              SplashKit.FillCircle(Color.Black, X, Y, radius);
57              SplashKit.FillCircle(Color.Black, EndX, EndY, radius);
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
```


Shape Drawer