Swinburne University of Technology

COS20007 Object Oriented Programming

3.3P - Drawing Program - A Drawing Class

PDF generated at 13:20 on Saturday 18th March, 2023

```
using DrawingProgram.lib;

namespace DrawingProgram
{
    public class Program
    {
        public static void Main()
        {
            Window window = new Window("Shape Drawer", 800, 600);

            Drawing drawing = new Drawing();
            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();

                // add new shape
                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    drawing.AddShape(new Shape(SplashKit.MouseX(),
↪   SplashKit.MouseY()));
                }

                // delete a shape
                if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
↪   SplashKit.KeyTyped(KeyCode.DeleteKey))
                {
                    drawing.DeleteShape();
                }
                // select a shape
                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    drawing.SelectShapesAt(SplashKit.MousePosition());
                }

                // change background color
                if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                {
                    drawing.Background = Color.Random();
                }

                drawing.Draw();
                SplashKit.RefreshScreen();

            } while (!window.CloseRequested);
        }
    }
}
```

```
using DrawingProgram.lib;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DrawingProgram
{
    public class Drawing
    {
        //variables
        private readonly List<Shape> _shapes;
        private Color _background;

        //properties

        //number of shapes in list, readonly
        public int ShapeCount
        {
            get
            {
                return _shapes.Count;
            }
        }

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

        //list of shapes that are currently selected
        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> result = new List<Shape>();

                foreach (Shape s in _shapes)
                {
                    if (s.Selected == true)
                    {
                        result.Add(s);
                    }
                }
```

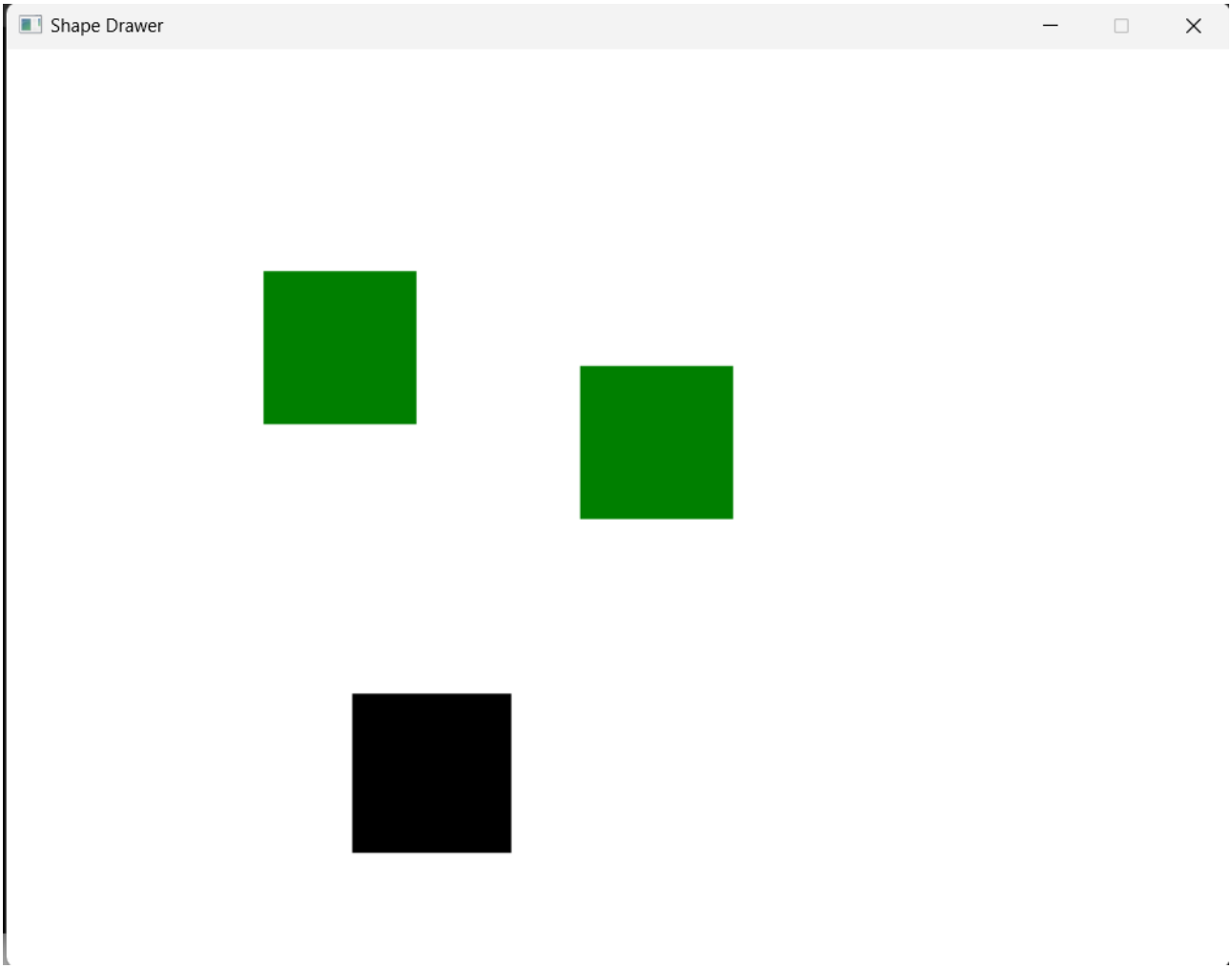
```
54
55                  return result;
56              }
57          }
58
59          //constructer that accepts color as a parameter for the background
60          public Drawing(Color background)
61          {
62              _shapes = new List<Shape>();
63              _background = background;
64          }
65
66          //default constructor
67          public Drawing() : this(Color.White)
68          {
69
70          }
71
72          //methods
73          public void AddShape(Shape shape)
74          {
75              _shapes.Add(shape);
76          }
77
78          public void Draw()
79          {
80              SplashKit.ClearScreen();
81              foreach (Shape shape in _shapes)
82              {
83                  shape.Draw();
84              }
85          }
86
87          public void SelectShapesAt(Point2D pt)
88          {
89              // checks if mouse position is over a shape, if true then its selected
    ↪  property is set to true
90              foreach (Shape s in _shapes)
91              {
92                  if (s.IsAt(pt))
93                  {
94                      s.Selected = true;
95                  }
96                  else
97                  {
98                      s.Selected = false;
99                  }
100             }
101         }
102
103         public void DeleteShape()
104         {
105             // new variable initiated since list cannot be modified while being
    ↪  enumerated
```

```
106             Shape deletedShape = new Shape(0, 0);
107             foreach (Shape s in _shapes)
108             {
109                 if (s.Selected)
110                 {
111                     deletedShape = s;
112                 }
113             }
114             _shapes.Remove(deletedShape);
115         }
116     }
117 }
```

```
using System;
using DrawingProgram.lib;

namespace DrawingProgram
{
    public class Shape
    {
        // local variables
        private Color _color;
        private float _x;
        private float _y;
        private int _width;
        private int _height;
        private bool _selected;

        // constructor
        public Shape(float x, float y)
        {
            _color = Color.Green;
            _x = x;
            _y = y;
            _width = 100;
            _height = 100;
            _selected = false;
        }

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

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

```
54              {
55                  return _y;
56              }
57              set
58              {
59                  _y = value;
60              }
61          }
62
63          public int Width
64          {
65              get
66              {
67                  return _width;
68              }
69              set
70              {
71                  _width = value;
72              }
73          }
74          public int Height
75          {
76              get
77              {
78                  return _height;
79              }
80              set
81              {
82                  _height = value;
83              }
84          }
85
86          public bool Selected
87          {
88              get
89              {
90                  return _selected;
91              }
92              set
93              {
94                  _selected = value;
95              }
96          }
97
98          // methods
99          public void Draw()
100         {
101             SplashKit.FillRectangle(_color, _x, _y, _width, _height);
102             if (_selected)
103             {
104                 DrawOutline();
105             }
106         }
```

```
107
108         public bool IsAt(Point2D pt)
109         {
110             if (pt.X > _x && pt.Y > _y)
111             {
112                 if (pt.X < _x + _width && pt.Y < _y + _height)
113                 {
114                     return true;
115                 }
116                 else
117                 {
118                     return false;
119                 }
120             }
121             else
122             {
123                 return false;
124             }
125
126         }
127
128         public void DrawOutline()
129         {
130             SplashKit.FillRectangle(Color.Black, _x - 2, _y - 2, _width + 4, _height
    ↪  + 4);
131         }
132
133
134     }
135
136 }
```


Shape Drawer