Selected files

3 printable files

Week 3\3.2\Program.cs
Week 3\3.2\Shape.cs
Week 3\3.2\Drawing.cs

Week 3\3.2\Program.cs

```
using System;
using SplashKitSDK;

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

            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape myShape = new Shape(50);
                    myShape.X = SplashKit.MouseX();
                    myShape.Y = SplashKit.MouseY();
                    myDrawing.AddShape(myShape);
                }

                if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                {
                    myDrawing.Background = SplashKit.RandomRGBColor(255);
                }

                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    Point2D pt = new Point2D();
                    pt.X = SplashKit.MouseX();
                    pt.Y = SplashKit.MouseY();

                    myDrawing.SelectShapesAt(pt);
                }

                if (SplashKit.KeyTyped(KeyCode.DeleteKey) ||
SplashKit.KeyTyped(KeyCode.BackspaceKey))
```

```
                {
                    myDrawing.RemoveSelectedShapes();
                }

                myDrawing.Draw();
                SplashKit.RefreshScreen();
            } while (!window.CloseRequested);
        }
    }
}

```

Week 3\3.2\Shape.cs

```
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Shape
    {
        // Fields
        private Color _color;
        private float _x, _y;
        private int _width, _height;
        private bool _selected;

        // Constructor
        public Shape(int param)
        {
            _color = Color.Chocolate;
            _x = 0.0f;
            _y = 0.0f;
            _width = param;
            _height = param;
            _selected = false;
        }

        // Methods
        public void Draw()
        {
            SplashKit.FillRectangle(_color, _x, _y, _width, _height);

            if (_selected)
            {
                DrawOutline();
            }
        }

        public bool IsAt(Point2D pt)
        {
            return (pt.X >= _x) && (pt.X <= (_x + _width))
```

```
                && (pt.Y >= _y) && (pt.Y <= (_y + _height));
        }

        public void DrawOutline()
        {
            SplashKit.DrawRectangle(Color.Black, _x - 9, _y - 9, _width + 18, _height + 18);
        }

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

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

        public Color Color
        {
            get { return _color; }
            set { _color = 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; }
        }
    }
}
```

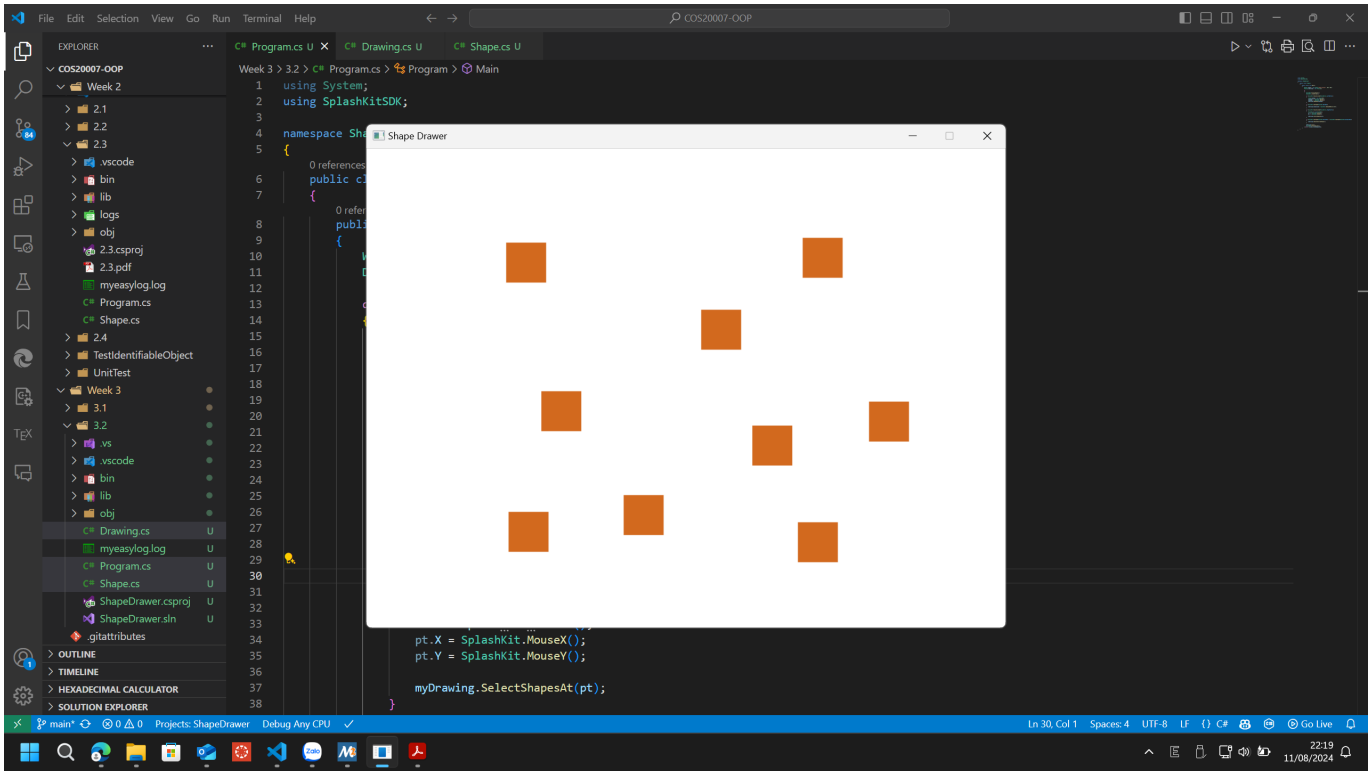
Week 3\3.2\Drawing.cs

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

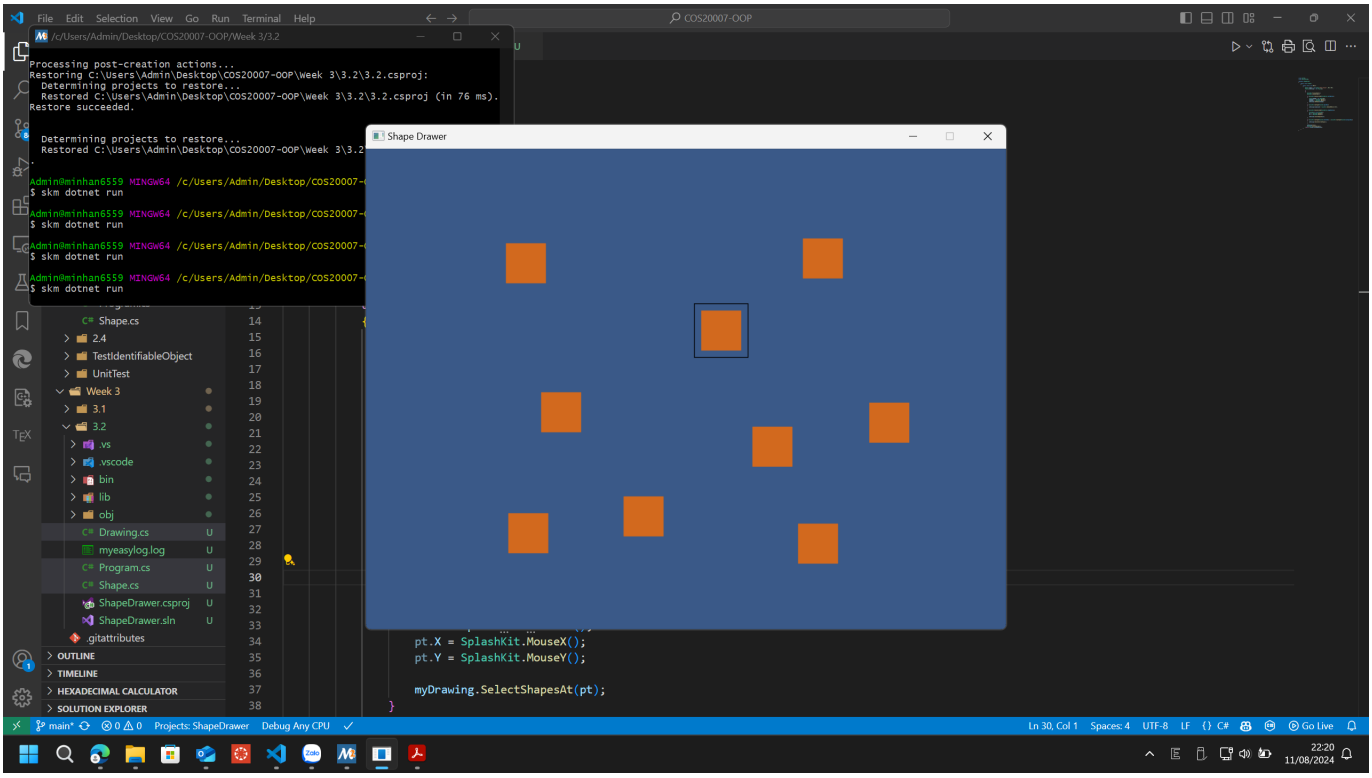
```
using SplashKitSDK;
using System.Linq;
using System.Threading.Tasks;

namespace ShapeDrawer
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;

        // Constructor
        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;
        }

        public Drawing() : this(Color.White)
        {
        }

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

        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)
                    {
                        result.Add(s);
                    }
                }
                return result;
            }
        }

```

```
        // Methods
        public void AddShape(Shape s)
        {
            _shapes.Add(s);
        }

        public void RemoveShape(Shape s)
        {
            _ = _shapes.Remove(s);
        }

        public void Draw()
        {
            SplashKit.ClearScreen(_background);
            foreach (Shape s in _shapes)
            {
                s.Draw();
            }
        }

        public void SelectShapesAt(Point2D pt)
        {
            foreach (Shape s in _shapes)
            {
                s.Selected = s.IsAt(pt);
            }
        }

        public void RemoveSelectedShapes()
        {
            foreach (Shape s in SelectedShapes)
            {
                RemoveShape(s);
            }
        }
    }
}
```

Screenshot of adding shapes:



Screenshot of selecting shapes and changing the background:



Screenshot of removing shape: