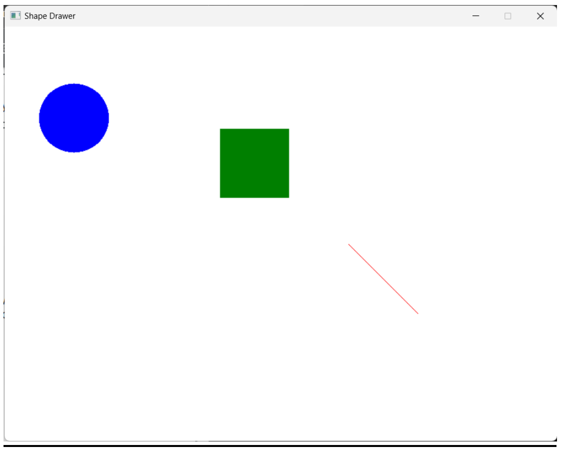
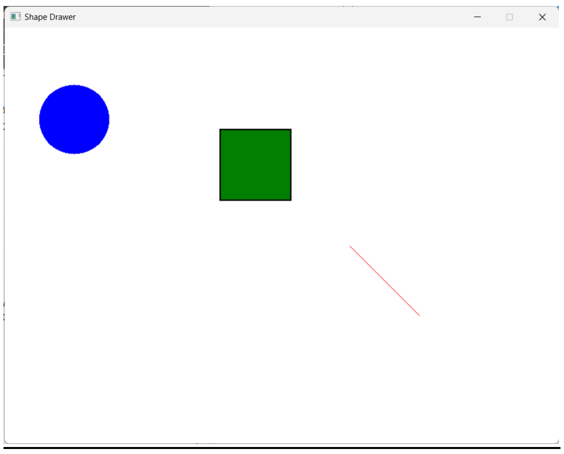
4.1: Drawing Program — Multiple Shape Kinds

Jayden Kong, 104547242

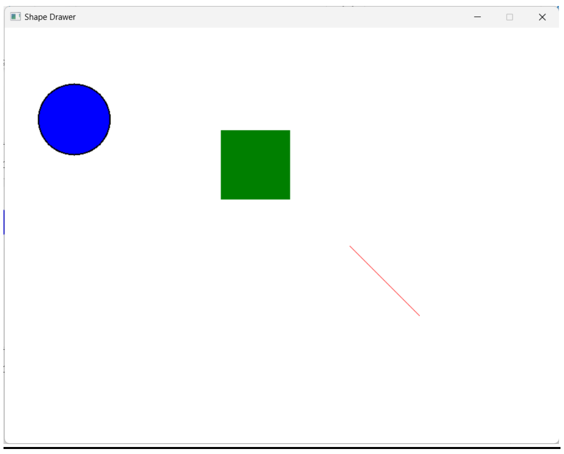
<u>Drawing all shapes:</u>



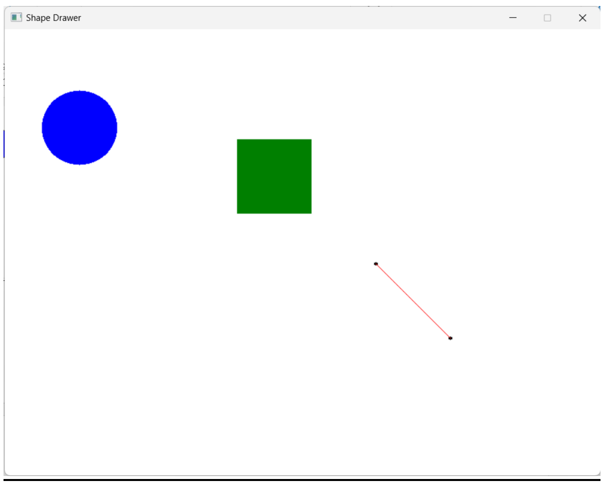
<u>Selecting rectangle:</u>



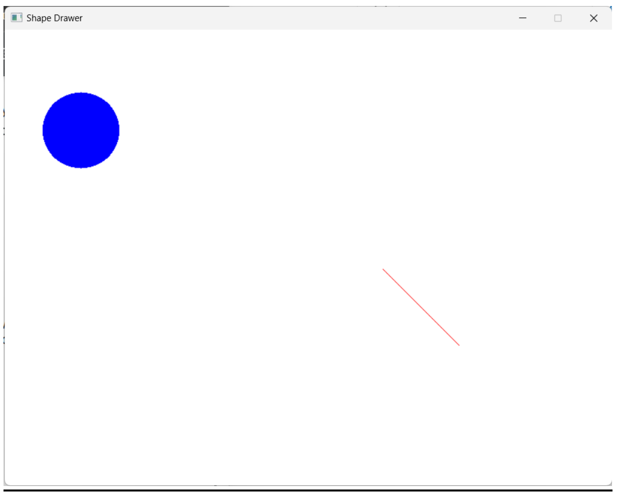
<u>Selecting circle:</u>



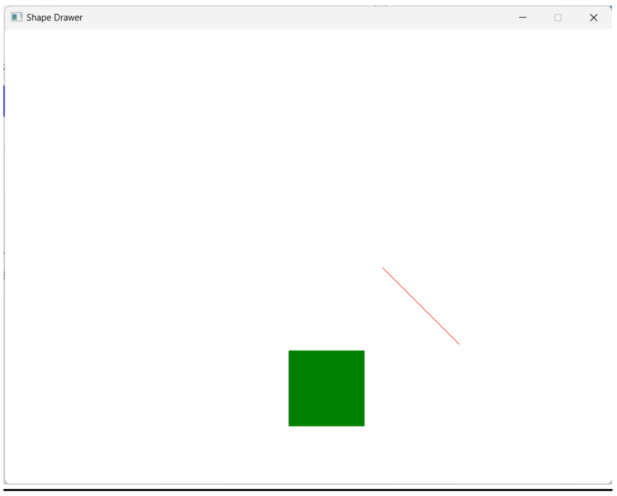
Selecting line:



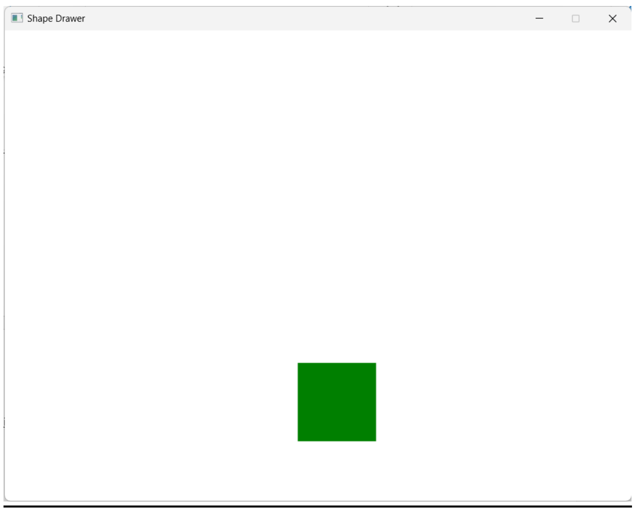
Removing rectangle:



Removing circle (added rectangle to show this is the same window):



Removing line:



```
using System;
using SplashKitSDK;
namespace ShapeDrawer
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }

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

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

                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;

                    switch (kindToAdd)
                    {
                        case ShapeKind.Circle:
                            newShape = new MyCircle();
                            break;
                        case ShapeKind.Line:
                            newShape = new MyLine();
                            break;
                        default:
                            newShape = new MyRectangle();
                            break;
                    }

                    newShape.X = SplashKit.MouseX();
                    newShape.Y = SplashKit.MouseY();
                    myDrawing.AddShape(newShape);
                }

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

                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    myDrawing.SelectShapesAt(SplashKit.MousePosition());
                }

                if (SplashKit.KeyTyped(KeyCode.DeleteKey) || SplashKit.KeyTyped(KeyCode.BackspaceKey))
                {
                    foreach(Shape s in myDrawing.SelectedShapes)
                    {
                        myDrawing.RemoveShape(s);
                    }
                }

                myDrawing.Draw();

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

```

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

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

        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;
            }
        }


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

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

        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);
            }
        }

    }
}

```

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

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

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

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

        public Shape(Color color)
        {
            _color = color;
            _x = 0.0f;
            _y = 0.0f;
        }

        public abstract void Draw();

        public abstract void DrawOutline();

        public abstract bool IsAt(Point2D pt);
    }
}

```

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

namespace ShapeDrawer
{
    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() : this(Color.Green, 0.0f, 0.0f, 100, 100) { }

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

        public override void Draw()
        {
            if (base.Selected)
            {
                DrawOutline();
            }

            SplashKit.FillRectangle(base.Color, X, Y, _width, _height);
        }

        public override void DrawOutline()
        {
            SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, _width + 4,
              _height + 4);
        }

        public override bool IsAt(Point2D pt)
        {
            return ((pt.X >= X) && (pt.X <= X + _width) && (pt.Y >= Y) &&
              (pt.Y <= Y + _height));
        }


    }
}
```

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

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

        public int Radius
        {
            get
            {
                return _radius;
            }
            set
            {
                _radius = value;
            }
        }
        public MyCircle() : this(Color.Blue, 0.0f, 0.0f, 50) { }

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


        public override void Draw()
        {
            if (Selected)
            {
                DrawOutline();
            }

            SplashKit.FillCircle(base.Color, X, Y, _radius);
        }

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

        public override bool IsAt(Point2D pt)
        {
            return SplashKit.PointInCircle(pt, SplashKit.CircleAt(X, Y, 
              _radius));
        }
    }
}

```

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

namespace ShapeDrawer
{
    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() : this(Color.Red, 0.0f, 0.0f, 100.0f, 100.0f) { }

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

        public override void Draw()
        {
            if (base.Selected)
            {
                DrawOutline();
            }

            SplashKit.DrawLine(base.Color, X, Y, X + EndX, Y + EndY);
        }

        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color.Black, X, Y, 2);
            SplashKit.FillCircle(Color.Black, X + EndX, Y + EndY, 2);
        }

        public override bool IsAt(Point2D pt)
        {
            return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y, X +
              EndX, Y + EndY), 5);
        }
    }
}
```