```
using System;
using _4._1._1;
using SplashKitSDK;

namespace _4_1_1
{
        public class Program
        {
            private enum ShapeKind
            {
                Rectangle,
                Circle,
                Line
            }
            public static void Main()
            {
                Window window = new Window("Shape Drawer Task 4.1", 800, 
                  600);
                Drawing myDraw = new Drawing();
                ShapeKind kindToAdd = ShapeKind.Rectangle;

                do
                {
                    SplashKit.ProcessEvents();

                    if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                    {
                        myDraw.background = SplashKit.RandomColor();
                    }

                    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 = new MyRectangle();

                        switch (kindToAdd)
                        {
                            case ShapeKind.Rectangle:
                                newShape = new MyRectangle();
                                break;

                            case ShapeKind.Circle:
                                newShape = new MyCircle();
                                break;


                            case ShapeKind.Line:
                                newShape = new MyLine();
                                break;
                        }

                        newShape.X = SplashKit.MouseX();
                        newShape.Y = SplashKit.MouseY();


                        myDraw.AddShape(newShape);
                    }

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

                        myDraw.SelectShapesAt(mousePos);
                    }

                    if (SplashKit.KeyTyped(KeyCode.BackspaceKey)||
                   SplashKit.KeyTyped(KeyCode.DeleteKey))
                    {
                        var selectedShapes = myDraw.SelectedShapes;

                        foreach (var shape in selectedShapes)
                        {
                            myDraw.RemoveShape(shape);
                        }
                    }


                    myDraw.Draw();


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

```

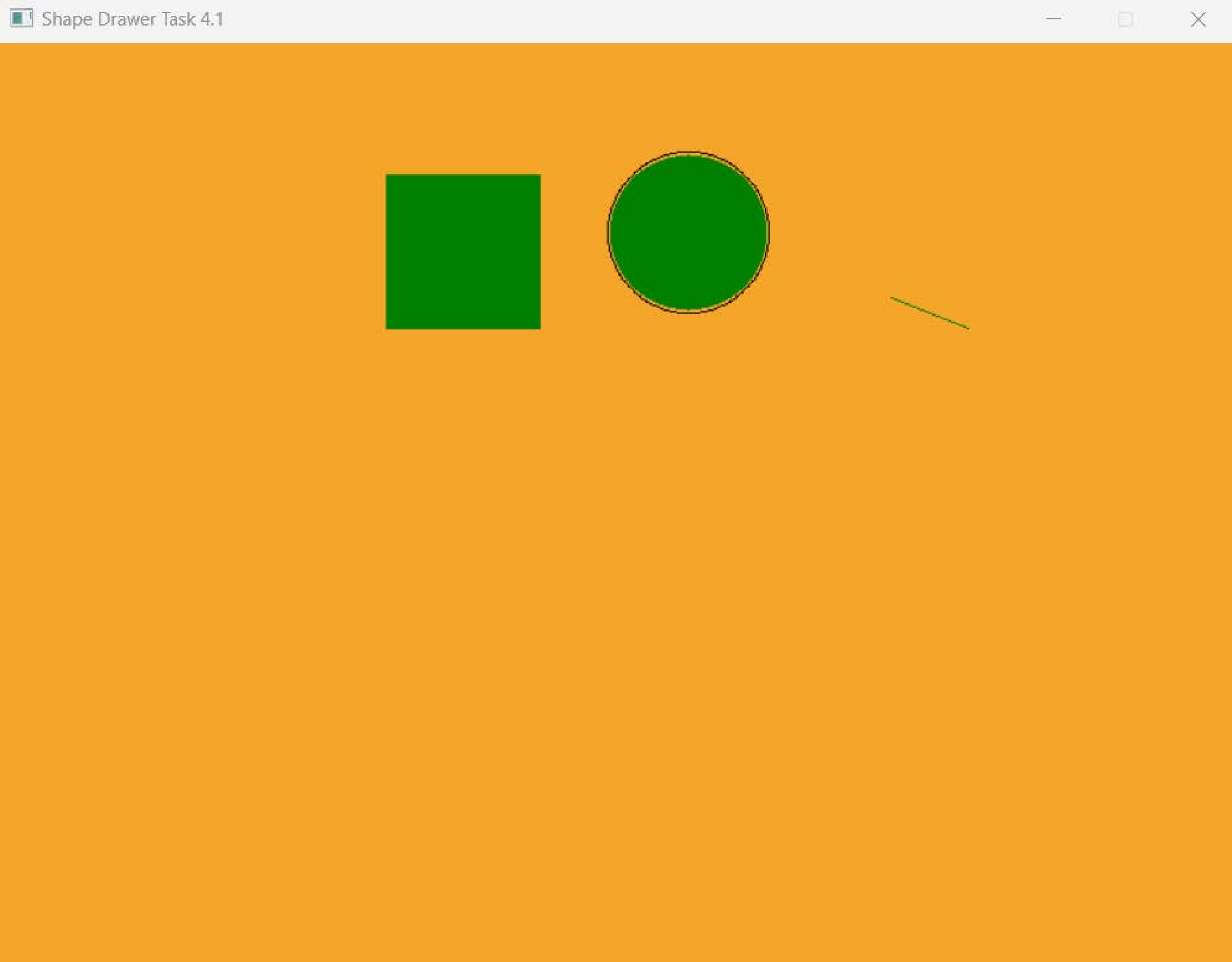
```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _4._1._1
{
    public abstract class Shape
    {
        private Color _color;
        private float _x;
        private float _y;

        private bool _selected;
        public Shape(Color color)
        {
            _color = Color.Green;
            _x = 0.0f;
            _y = 0.0f;
        }
        public Shape(): this(Color.Yellow) { }

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

        public float X
        {
            get { return _x; }
            set { _x = value; }
        }
        public float Y
        {
            get { return _y; }
            set { _y = value; }
        }
        public bool Selected
        {
            get
            {
                return _selected;
            }
            set { _selected = value; }
        }
        public abstract void Draw();
        public abstract void DrawOutline();
        public abstract bool IsAt(Point2D pt);

    }
}
```

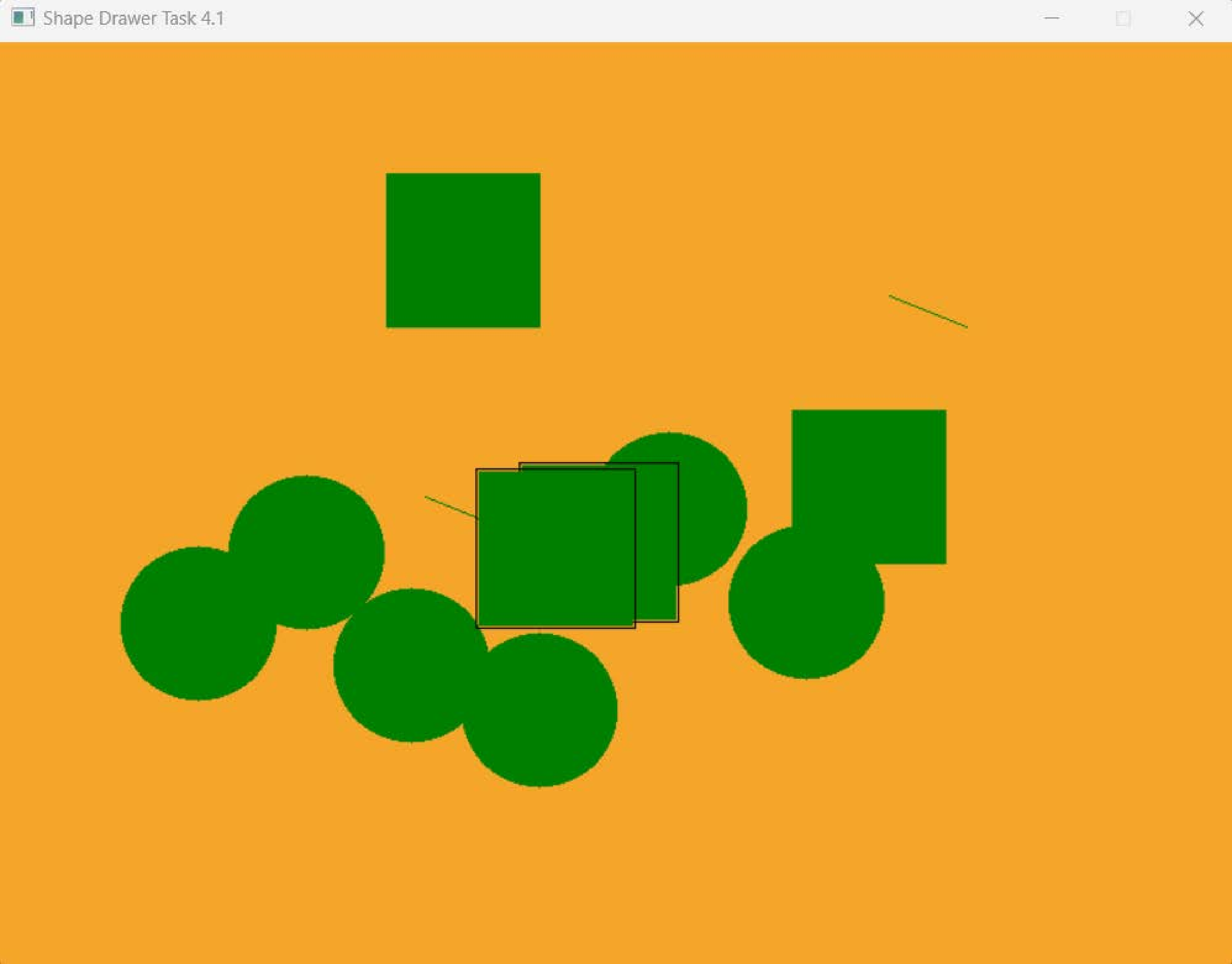
```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _4._1._1
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private SplashKitSDK.Color _background;
        private SplashKitSDK.Color Color;

        public Drawing(SplashKitSDK.Color background)
        {
            _background = background;
            _shapes = new List<Shape>();
        }
        public SplashKitSDK.Color background
        {
            get
            {
                return _background;
            }
            set
            {
                _background = value;
            }
        }
        public Drawing() : this(SplashKitSDK.Color.White)
        {

        }
        public List<Shape> SelectedShapes
        {
            get
            {
                var result = new List<Shape>(); ;
                foreach (var shape in _shapes)
                {
                    if (shape.Selected)
                    {
                        result.Add(shape);
                    }
                }
                return result;
            }
        }
        public int ShapeCount
        {
            get { return _shapes.Count(); }
        }
        public void AddShape(Shape shape)
        {
            _shapes.Add(shape);
        }
        public void RemoveShape(Shape shape)
        {
            _shapes.Remove(shape);
        }
        public void Draw()
        {
            SplashKit.ClearScreen(_background);
            foreach (Shape shape in _shapes)
            {
                shape.Draw();
            }
        }

        public void SelectShapesAt(Point2D pt)
        {
            foreach (var shape in _shapes)
            {
                if (shape.IsAt(pt))
                {
                    shape.Selected = true;
                }
                else { shape.Selected = false; }
            }
        }

    }
}


```

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

namespace _4._1._1
{
    public class MyRectangle : Shape
    {
        private int _width;
        private int _height;

        public MyRectangle(Color color, float x, float y, int width, int 
          height) : base(color)
        {
            X = x;
            Y = y;
            Width = width;
            Height = height;
        }
        public MyRectangle() : this(Color.Green, 0.0f, 0.0f, 100, 100)
        {

        }

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

        public int Height
        {
            get { return _height; }
            set { _height = value; }
        }
        public override void Draw()
        {
            SplashKit.FillRectangle(Color, X, Y, Width, Height);
            if (Selected)
            {
                DrawOutline();
            }
        }

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

        public override bool IsAt(Point2D pt)
        {
            return SplashKit.PointInRectangle(pt, SplashKit.RectangleFrom
              (X, Y, Width, Height));
        }
    }
}


```

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

namespace _4._1._1
{
    public class MyCircle : Shape
    {
        private int _radius;

        public MyCircle(Color color, int radius) : base(color)
        {
            Radius = radius;
        }

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

        public int Radius
        {
            get { return _radius; }
            set { _radius = value; }
        }

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

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

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

        public override bool IsAt(Point2D pt)
        {
            double distanceX = Math.Abs(pt.X - X);
            double distanceY = Math.Abs(pt.Y - Y);

            return (distanceX <= Radius) && (distanceY <= Radius);
        }

    }
}
```

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

namespace _4._1._1
{
    public class MyLine : Shape
    {
        private float _endX;
        private float _endY;

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

        public MyLine() : this(Color.Blue, 0.0f, 0.0f, 50.0f, 20.0f)
        {
        }

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

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

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

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

        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color.Black, X, Y, 3);
            SplashKit.FillCircle(Color.Black, X + EndX, Y + EndY, 3);
        }

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


```


Shape Drawer Task 4.1


Shape Drawer Task 4.1