4.1 shapedrawer3
source code
MyCircles.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using shapedrawerV3;
using SplashKitSDK;
using static SplashKitSDK.SplashKit;

namespace shapedrawerV3
{
   public class MyCircles : Shape
   {
      private int _radius;

      public MyCircles(SplashKitSDK.Color clr, float x, float y, int radius) : base(clr)
      {
         X = x;
         Y = y;
         _radius = radius;

      }

      public int Radius
      {
```

```
        get { return _radius; }
        set { _radius = value; }
    }
    public override void Draw()
    {
        if (Selected)
        {
            DrawOutline();
        }
        FillCircle(Color, X, Y, Radius );

    }

    public override void DrawOutline()
    {
        FillCircle(SplashKitSDK.Color.Black, X - 2, Y - 2, Radius + 4); // No need for
SplashKit prefix
    }

    public override bool IsAt(Point2D point)
    {

        double a = (double)(point.X - X);
        double b = (double)(point.Y - Y);

        if (Math.Sqrt(a * a + b * b) < _radius)
        {
            return true;
```

```
        }
        return false;

    }



  }
}
```

MyLine.cs

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

namespace shapedrawerV3
{
  public class MyLine : Shape
  {
    private float _endY;
    private float _endX;

    public MyLine(Color clr, float startX, float startY, float endY, float endX) : base(clr)
    {
      X = startX;
      Y = startY;
```

```
    _endX = endX;
    _endY = endY;
}



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, EndY, EndX);

    }

    public override void DrawOutline()
    {
      SplashKit.DrawRectangle(SplashKitSDK.Color.Black, X - 2, Y - 2, EndY + 4, EndX +
4); // No need for SplashKit prefix
    }

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



  }
}
```

MyRectangle.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using shapedrawerV3;
using SplashKitSDK;
using static SplashKitSDK.SplashKit;

namespace shapedrawerV3
{
   public class MyRectangle : Shape
   {
      private int _width;
      private int _height;




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


      }
```

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

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



}
public override void Draw()
{
   if (Selected)
```

```
        {
            DrawOutline();
        }
        FillRectangle(Color, X, Y, _width, _height);

    }

    public override void DrawOutline()
    {
        FillRectangle(SplashKitSDK.Color.Black, X - 2, Y - 2, _width + 4, _height + 4); // No need for SplashKit prefix
    }

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



  }
}
```

Shape.cs

```
using SplashKitSDK;
using static SplashKitSDK.SplashKit;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace shapedrawerV3
{
   public abstract class Shape
   {
      // Private fields
      private Color _color;
      private float _x, _y;
      private float _width, _height;
      private bool _selected;
      private int x_pos;
      private int y_pos;
      private Color clr;

      public Shape(int x_pos, int y_pos)
      {
         this.x_pos = x_pos;
         this.y_pos = y_pos;
      }

      public Shape(Color clr)
      {
         _color = clr;
      }
```

```
// Properties
public Color Color //call and intialize the variable
{
   get { return _color; }
   set { _color = value; }
}

public float X //call and intialize the variable
{
   get { return _x; } //get and store the x value
   set { _x = value; }
}

public float Y //call and intialize the variable
{
   get { return _y; } //get and store the y value
   set { _y = value; }
}

public float Width //call and intialize the variable
{
   get { return _width; } //get and store the width
   set { _width = value; }
}

public float Height //call and intialize the variable
{
```

```
        get { return _height; } //get and store the height
        set { _height = value; }
    }

    // Method to draw the shape
    public abstract void Draw(); //abstract to override


    // Method to check if a point is within the shape's area
    public abstract bool IsAt(Point2D point); // we set this as abstract so all classes can override
    // {
      // return (point.X >= _x && point.X <= _x + _width) &&
            //(point.Y >= _y && point.Y <= _y + _height); // to find the coord after or below a specify point
    //}//if x is more than equal to x and x is less than equal to x + width is to find the whole width dimension of the box


    public bool Selected //determine selected shapes value and return
    {
        get
        {
            return _selected;
        }
        set
        {
            _selected = value;
        }
    }
```

```
        public abstract void DrawOutline(); //draw outline to show that its highlighted
        //{
          // SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, _width + 4, _height + 4);
        //}
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using shapedrawerV3;
using SplashKitSDK;



namespace shapedrawerV3
{
    public class Program //this program.cs main job is to draw the canvas and display the varibles
    {



        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.Rectangle;

    do
    {
        SplashKit.ProcessEvents();
        SplashKit.ClearScreen();
        if (SplashKit.KeyTyped(KeyCode.RKey)) // for  rectangle shape
        {
            kindToAdd = Shapekind.Rectangle;
        }
        if (SplashKit.KeyTyped(KeyCode.CKey))// for Circle shape
        {
            kindToAdd = Shapekind.Circle;
        }
        if (SplashKit.KeyTyped(KeyCode.LKey)) // for Line
        {
            kindToAdd = Shapekind.Line;
        }
```

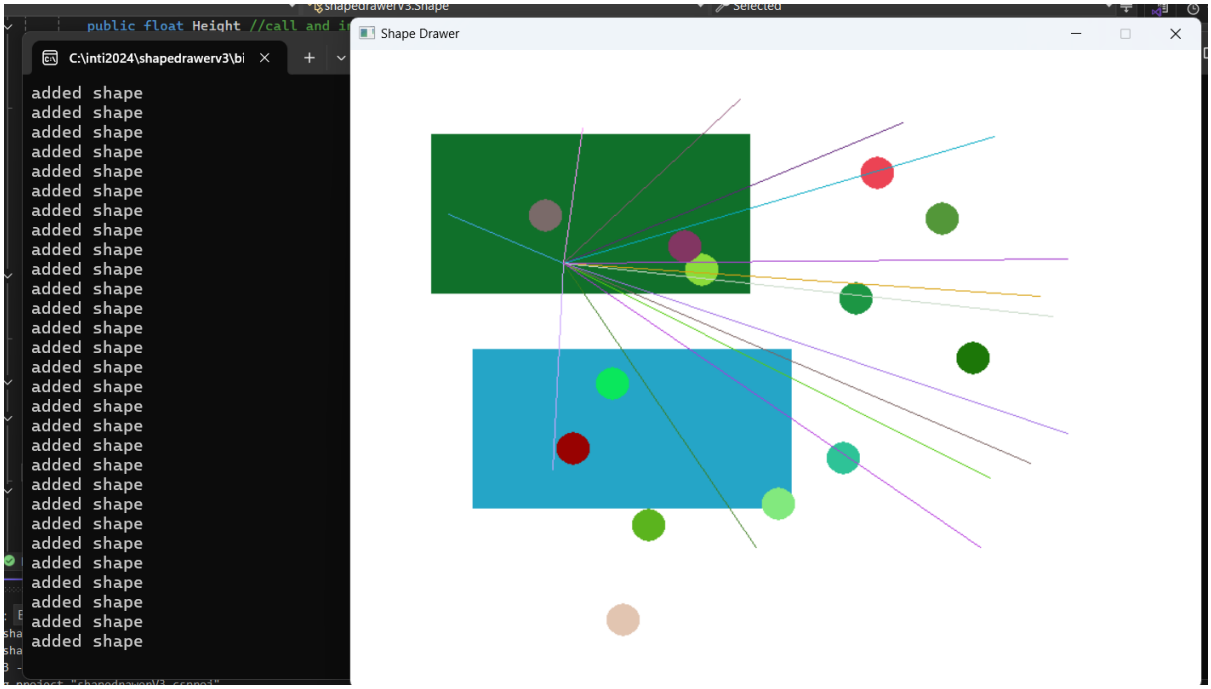
```
        if (SplashKit.MouseClicked(MouseButton.LeftButton)) //the left click to add the shapes according to what we set previously
        {
            Shape newShape;

            switch (kindToAdd) // intiliazing the switch case
            {
                case Shapekind.Circle: // the shapekind will help determind what key are we on
                    newShape = new MyCircles(Color.Blue, 20, 20, 15); //case switch to Circles
                    newShape.X = SplashKit.MouseX();
                    newShape.Y = SplashKit.MouseY();
                    newShape.Color = SplashKit.RandomRGBColor(255);
                    break;

                case Shapekind.Line: // Shapekind will help determine what key are we on
                    newShape = new MyLine(Color.Black, 165, 165, 200, 200); // if switch is in Line then executed
                    newShape.X = SplashKit.MouseX();
                    newShape.Y = SplashKit.MouseY();
                    newShape.Color = SplashKit.RandomRGBColor(255);
                    break;

                default:
                    newShape = new MyRectangle(Color.Red, 456, 234, 300, 150); //set default to rectangle so that the first thing is the rectangle
                    newShape.X = SplashKit.MouseX();
                    newShape.Y = SplashKit.MouseY();
```

```
                newShape.Color = SplashKit.RandomRGBColor(255);
                break;

                // Add the new shape to the Drawing object


            }
            myDrawing.AddShape(newShape); //call addShape function from drawing.cs
            Console.WriteLine("added shape");


        }
        if (SplashKit.KeyTyped(KeyCode.SpaceKey)) //spacekey to change different
background color
        {
            myDrawing.Background = SplashKit.RandomRGBColor(255);

        }
        if (SplashKit.MouseClicked(MouseButton.RightButton)) //right click to highlight
outline of the shape
        {
            myDrawing.SelectShapesAt(SplashKit.MousePosition());

        }

        if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
SplashKit.KeyTyped(KeyCode.DeleteKey))//to delete shapes drawn
        {
            if (SplashKit.KeyTyped(KeyCode.BackspaceKey)) //backspce key to delete
```

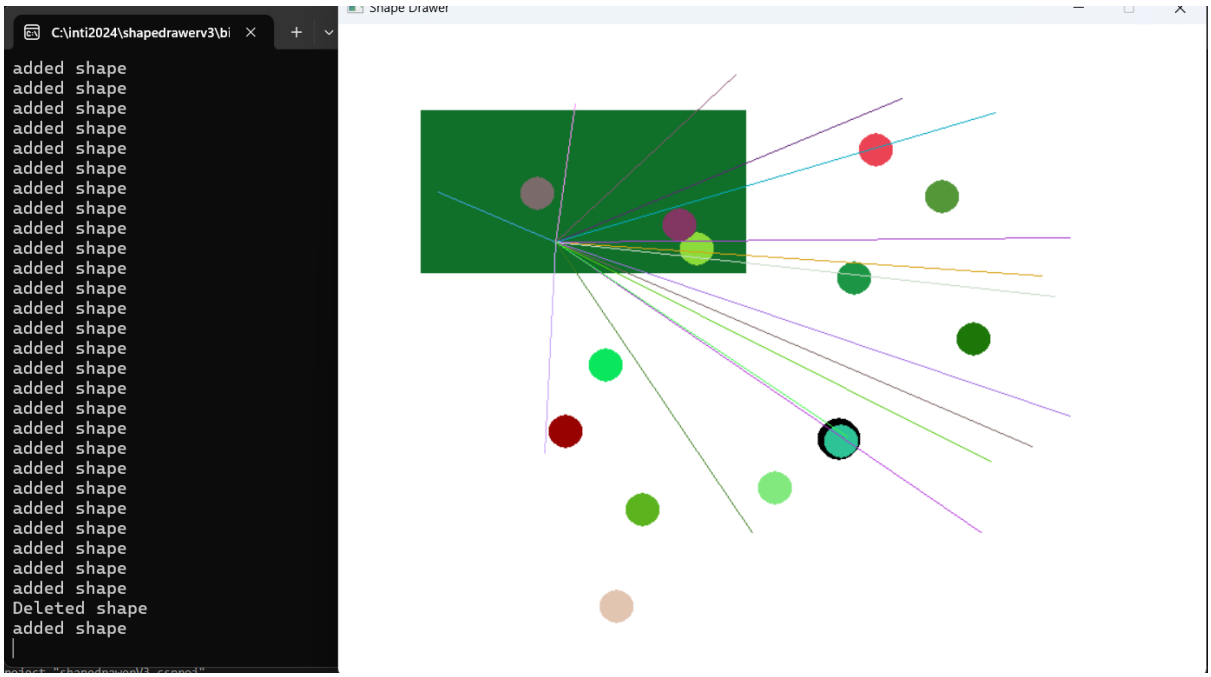
```
            {
                Console.WriteLine("Deleted shape");
            }
            if (SplashKit.KeyTyped(KeyCode.DeleteKey)) //delete key to delete
            {
                Console.WriteLine("Deleted shape");
            }
            myDrawing.RemoveShape(); //call the remove shape function frim drawing.cs
        }

        myDrawing.Draw();

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

//when declare a new function do NOT place it in the same function as prev or else the function wouldnt as its not the main father function

OUTPUT



With all 3 shapes



Some shapes deleted