5.3 shape drawer 4 (with added save and load)
Source code
ExtensionMethods.cs

```
using System;

using System.IO;

using SplashKitSDK;


namespace MyGame

{

   public static class ExtensionMethods

   {

      public static int ReadInteger(this StreamReader reader)

      {

         return Convert.ToInt32(reader.ReadLine());

      }

      public static float ReadSingle(this StreamReader reader)

      {

         return Convert.ToSingle(reader.ReadLine());

      }

      public static Color ReadColor(this StreamReader reader)

      {

         return Color.RGBColor(reader.ReadSingle(), reader.ReadSingle(),
reader.ReadSingle());

      }

      public static void WriteColor(this StreamWriter writer, Color clr)

      {

         writer.WriteLine("{0}\n{1}\n{2}", clr.R, clr.G, clr.B);

      }

   }
```

```
}
shape.cs
using SplashKitSDK;

using static SplashKitSDK.SplashKit;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MyGame;


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();



        // Method to check if a point is within the shape's area
        public abstract bool IsAt(Point2D point);
        // {
        // 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
    public virtual void SaveTo(StreamWriter _writer)
    {
        _writer.WriteColor(_color);
        _writer.WriteLine(X);
        _writer.WriteLine(Y);
    }

    public virtual void LoadFrom(StreamReader _reader)
    {
        Color = _reader.ReadColor();
        X = _reader.ReadInteger();
        Y = _reader.ReadInteger();
    }

    //{
    // SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, _width + 4, _height + 4);
    //}
  }
}
```

Drawing.cs

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

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MyGame;

using SplashKitSDK;


namespace shapedrawerV3

{

  public class Drawing

  {

    private readonly List<Shape> _shapes; //readonly list to store all shapes

    private Color _background;

    StreamWriter _writer;

    StreamReader _reader;


    public Drawing(Color background)

    {

      _shapes = new List<Shape>();

      _background = background;

    }

    public Drawing() : this(Color.White) //"this" to avoid duplication

    {


    }


    public int ShapeCount //property to class Drawing that returns the Count from the
_shapes list collection object
```

```
{
  get { return _shapes.Count; } //get and return from _shapes = new List<shape>
}

public void AddShape(Shape s) //adds shape into the list from shape
{
  _shapes.Add(s);
}
public void RemoveShape()
{
  foreach (Shape s in _shapes.ToList())//each shape s is from AddShape
  {
    if (s.Selected) //if shape is selected
    {
      _shapes.Remove(s); //remove the shape
    }


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

```
{
   get
   {
      return _background;


   }
   set
   {
      _background = value;
   }



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

```
public List<Shape> SelectedShapes()
{
  List<Shape> _Selectedshapes = new List<Shape>();
  foreach (Shape s in _shapes)
  {
    if (s.Selected)
    {
      _Selectedshapes.Add(s);
    }
  }
  return _Selectedshapes;
}
public void Save(string filename)
{
  _writer = new StreamWriter(filename);
  _writer.WriteColor(_background);
  _writer.WriteLine(ShapeCount);

  foreach (Shape s in _shapes)
  {
    s.SaveTo(_writer);
  }
  _writer.Close();
}
public void Load(string filename)
{
  _reader = new StreamReader(filename);
  Shape s;
```

```
        string kind;
        _background = _reader.ReadColor();
        int Count = _reader.ReadInteger();
        _shapes.Clear();
        for (int i = 0; i < Count; i++)
        {
          kind = _reader.ReadLine();
          switch (kind)
          {
            case "Rectangle":
              s = new MyRectangle(Color.Red, 456, 234, 300, 150);
              break;

            case "Circle":
              s = new MyCircles(Color.Blue, 20, 20, 15);
              break;
            case "Line":
              s = new MyLine(Color.Black, 165, 165, 200, 200); // its not stated in question
but i wan to make sure it can load all shapes
              break;
            default:
               s = new MyLine(Color.Black, 165, 165, 200, 200);
              continue;



          }
          s.LoadFrom(_reader);
          AddShape(s);
```

```
            }
            _reader.Close();



        }
    }
}
```

MyRectangle.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MyGame;
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 MyRectangle() : this(Color.RandomRGB(255), 0, 0, 100, 100) { }

        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);
        }
        public override void SaveTo(StreamWriter _writer)
        {
            _writer.WriteLine("Rectangle");
            base.SaveTo(_writer);
            _writer.WriteLine(Width);
            _writer.WriteLine(Height);
            _writer.WriteLine($"{(int)(Color.R * 255)},{(int)(Color.G * 255)},{(int)(Color.B *
255)}");
        }
        public override void LoadFrom(StreamReader _reader)
        {
            base.LoadFrom(_reader);
            Width = _reader.ReadInteger();
            Height = _reader.ReadInteger();
        }






    }
}
```

MyCircles.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MyGame;
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;


        }
        public override void SaveTo(StreamWriter _writer)// overriding from SaveTo in
shape.cs
        {
            _writer.WriteLine("Circle");
            base.SaveTo(_writer);
            _writer.WriteLine(Radius);
            _writer.WriteLine($"{(int)(Color.R * 255)},{(int)(Color.G * 255)},{(int)(Color.B *
255)}");

        }
        public override void LoadFrom(StreamReader _reader) // overriding from LoadFrom
in shape.cs
        {
            base.LoadFrom(_reader);
            Radius = _reader.ReadInteger();

        }



    }
}
```

MyLine.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MyGame;
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));
        }
        public override void SaveTo(StreamWriter _writer)
        {
            _writer.WriteLine("Line");
            base.SaveTo(_writer);
            _writer.WriteLine(EndY);
            _writer.WriteLine(EndX);
            _writer.WriteLine($"{(int)(Color.R * 255)},{(int)(Color.G * 255)},{(int)(Color.B *
255)}");
        }
        public override void LoadFrom(StreamReader _reader)
        {
            base.LoadFrom(_reader);
            EndX = _reader.ReadInteger();
            EndY = _reader.ReadInteger();
        }




    }
}
```

Program.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

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
            }
            if (SplashKit.KeyDown(KeyCode.SKey)) //s key to save
            {
                myDrawing.Save("C:\\inti2024\\shapedrawerv4\\TestDrawing.txt");
                {
                    Console.WriteLine("Shape Saved");
                }
            }

            if (SplashKit.KeyTyped(KeyCode.OKey))
            {
                myDrawing.Load("C:\\inti2024\\shapedrawerv4\\TestDrawing.txt");
                {
                    Console.WriteLine("Loaded shapes");
                }
            }



            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
```

TestDrawing.cs
1

1

1

23

Line

0.9949034

0.8796655

0.7837153

86

104

200

200

253,224,199

Line

0.2677694

0.26804405

0.43549913

261

272

200

200

68,68,111

Line

0.40449232

0.27759635

0.39509264

77

218

200

200

103,70,100

Line

0.76116216

0.91116065

0.4969329

254

144

200

200

194,232,126

Line

0.18341625

0.7705924

0.9307535

155

316

200

200

46,196,237

Line

0.5107272

0.6183966

0.7137669

436

139

200

200

130,157,182

Line

0.38975188

0.6966155

0.23520616

180

290

200

200

99,177,59

Line

0.7253944

0.7744377

0.6326792

272

81

200

200

184,197,161

Line

0.10599078

0.2822657

0.280282

387

387

200

200

27,71,71

Line

0.0675985

0.9032868

0.15704826

338

126

200

200

17,230,40

Line

0.5722831

0.023926511

0.56248665

376

368

200

200

145,6,143

Line

0.5964232

0.8579058

0.31595814

420

182

200

200

152,218,80

Circle

0.27213356

0.3872799

0.7384869

300

191

15

69,98,188

Circle

0.57585377

0.89831233

0.4552446

328

306

15

146,229,116

Circle

0.62843716

0.8550066

0.07144383

375

132

15

160,218,18

Circle

0.6596881

0.07144383

0.63075656

171

274

15

168,18,160

Circle

0.74925995

0.42695394

0.90359205

171

126

15

191,108,230

Circle

0.8180792

0.3541673

0.63203835

190

309

15

208,90,161

Circle

0.6192511

0.35734123

0.81026644

275

125

15

157,91,206

Circle

0.5834834

0.61668754

0.20169683

158

396

15

148,157,51

Circle

0.38340405

0.82940155

0.45179603

311

189

15

97,211,115

Rectangle

0.39136937

0.7318339

0.25815606

404

226

300

150

99,186,65

Rectangle

0.044343393

0.044618063

0.5476546

337

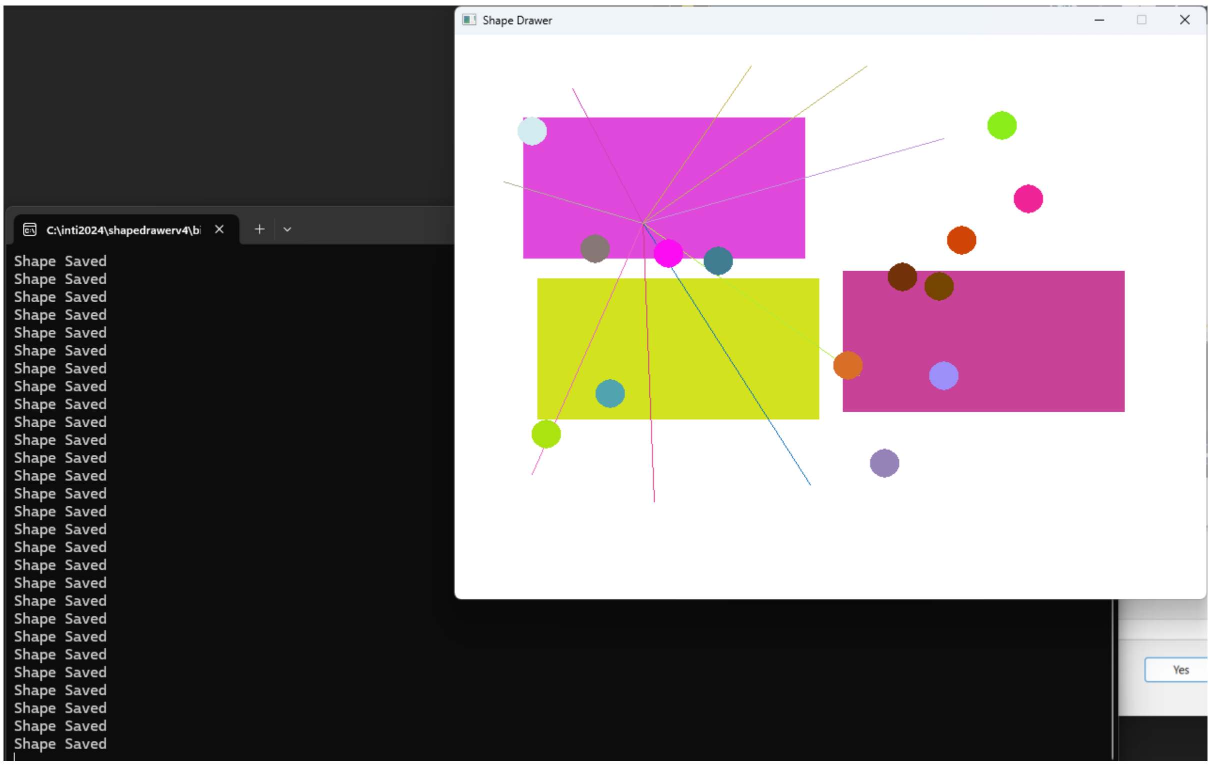
353

300

150

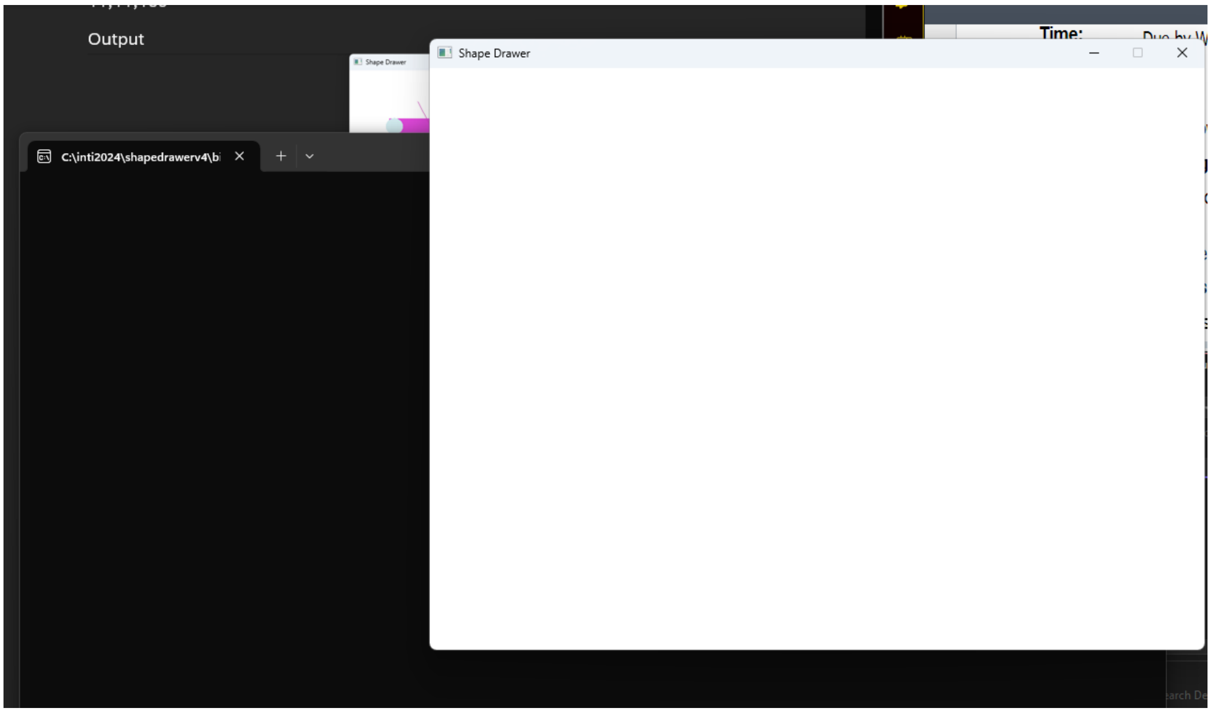
11,11,139

Output

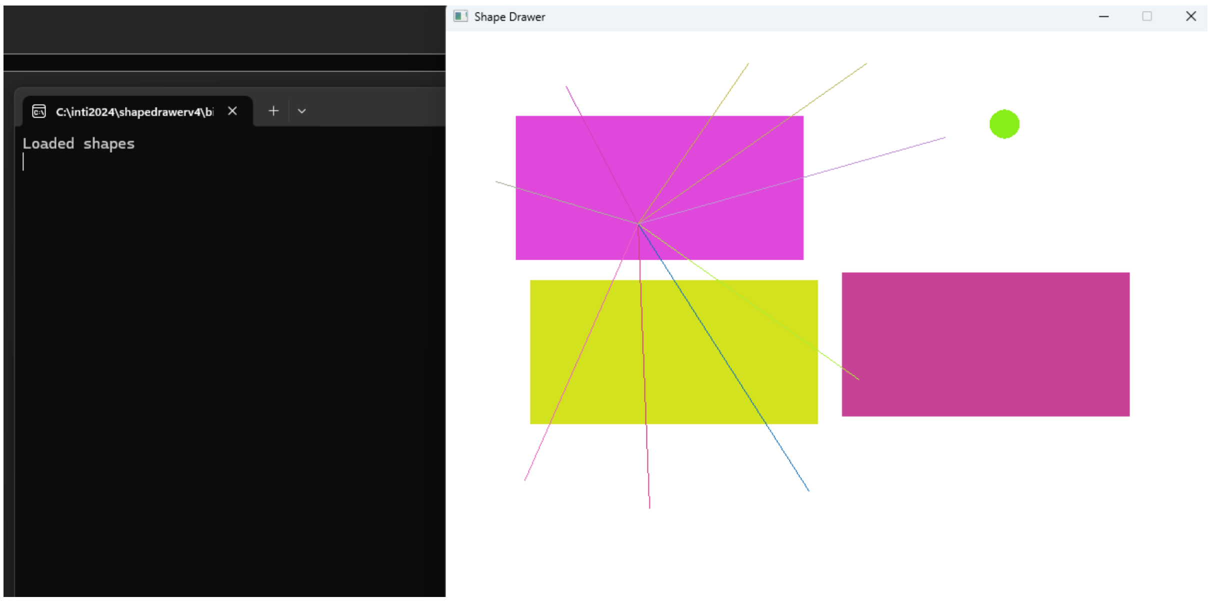


```
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
Shape Saved
```

inserting new shapes and saving it



blank canvas to show it works as intended then last load it back



```
Loaded shapes
```

it will load as the saved as shown in console

- it appear as loaded shapes so which means it’s a successful load from the text file