

GUI Design Report

CAB302 – Software Development
Semester 1 - 2019

Group 002

Kevin Duong – N9934731

Waldo Fouche – N9950095

Due Date: 02/06/19

Contents

Statement of Completeness	2
Statement of Contribution	2
Agile Software Development	3
Software Architecture	4
Folder: Backend.....	5
Folder: userInterface.....	7
Advanced Object-Oriented Programming Principles	8
Abstraction.....	8
Encapsulation	8
Inheritance.....	8
Polymorphism	8
Software Demonstration	8
Launching the Program.....	8
Shape Drawing Functionalities.....	12
.....	16
Revert Action using Undo	16
Colour Usage	17
Saving a VEC Image.....	19
Opening a VEC Image	21
Creating a new VEC File.....	23
Zooming in on Canvas	24
Image Code & History Log	25

Statement of Completeness

Basic Functionality	Status
Loading VEC Images	Completed
PLOT	Completed
LINE	Completed
RECTANGLE	Completed
ELLIPSE	Completed
POLYGON	Completed
Undo	Completed
Saving VEC Images	Completed
Additional Functionality	Status
Zoom	Incomplete

Statement of Contribution

Criteria List	Kevin Duong	Waldo Fouche
Source Control	✓	✓
Report - Agile Methods	✓	
Report - Architecture	✓	
Report - OOP		✓
Report - Manual	✓	✓
Javadoc	✓	✓
GUI function	✓	✓
GUI design	✓	✓
Functionality - Loading VEC Images		✓
Functionality - PLOT	✓	

Functionality - LINE	✓	✓
Functionality - RECTANGLE	✓	✓
Functionality - ELLIPSE	✓	
Functionality - POLYGON	✓	✓
Functionality - Undo		✓
Functionality - Saving VEC Images	✓	✓
PEN Colour	✓	
FILL Colour		✓
Unit Testing	✓	✓
Additional Functionality - Zoom		

Agile Software Development

A good software requires collaborative effort and team working flexibility, where adaptive planning, evolutionary and continual improvement of development are encouraged. Below are the 12 principles of Agile Methodology based on the Manifesto of Agile Software Development that explores effective work practices:

Satisfy The Customer

It is expected that the deliverance of the software upholds good expectations from the client.

Welcome Change

Changes can occur at anytime during development, which can be improved in quality or even so reach closer to the client's needs.

Deliver Frequently

This involves sending software frequently to the client for regular feedback and by being able to find out any additional needs or miscommunications from the client.

Work Together

At a local level, working together makes good and quick progress of developing the software and is effective overall.

Build Projects

Emphasize the work environment and providing support amongst team members who are involved in the project.

Face-To-Face Time

Convey information and communicate ideally through face-to-face conversations, especially team meetings in co-locations.

Measure of Progress

The primary goal is to prioritize in working on the software, putting other tasks below first.

Sustainable Development

Being able to balance work/life style to maintain consistency in work flow.

Continuous Attention

Always looking out for good design methods which leads to mastering in the agility to improve the software.

Keep It Simple

Knowing how to make a GUI relatively easy and simple to use for everyone and making it minimalistic by keeping functionalities that matter.

Organised Teams

Relying each other as a whole team to organise and set tasks and issues to work on, creating quality software architectures and design.

Reflect for Effectiveness

Learning how to improve one's self to become more effective for the whole team and adjust accordingly in benefits for the project.

Software Architecture

The program's architecture consists of two folders: a "backend" and "userinterface" which both combined help run the GUI. Both folders are mostly made up of classes written in Java that work together to parse information and perform functionalities to produce results. Other extension files also assist in creating the software as well.

Folder: Backend

ConsoleGUI

IntelliJ's console is beneficial for outputting feedback and is used to capture and display information onto a TextArea in the GUI. This feature is important to track what coordinates are needed for drawing shapes, identifying what colours are used, and writing/reading files from the console. This class only interacts with Controller class, handles the text streams being sent from the console to the console on the GUI.

DisableButtons

Ideally for button management that disables all other buttons in the GUI if a button is toggled so that other external functionalities do not conflict with the chosen action. For example, if the 'LINE' button is clicked, the rest of the shape buttons 'PLOT', 'RECTANGLE', 'ELLIPSE' and 'POLYGON' cannot be used and are disabled until the user turns off 'LINE' by clicking its button again. The class alone helps action handling method for every shape in the Controller class.

Draw

Instructions to handle drawing vectors when analysing .VEC files. Coordinates are stored and drawn onto the canvas using `getGraphicsContext2D()` and appended to fill and stroke to complete the drawing. Each 'if' statement has its own instructions on how to draw and colour the shape. The class is called from an action method in Controller class that handles opening files and DrawFromFile class and is ~~required for the~~ required for the undo method to ~~function correctly~~ function correctly.

DrawEllipse

The ellipse is one of the shape functionalities available to use and draw on the canvas. When the mouse is pressed down anywhere on the canvas, the ellipse captures the centre x and y of the mouse's location. By pressing the mouse, dragging it across the canvas and releasing it, an ellipse shape with now registered radius x and y, will be displayed onto the canvas along with its shape classification and coordinates in the GUI's TextArea. The Controller class handles the shape function, specifically the ellipse's action event method.

DrawFromFile

Necessary to load .VEC file images onto the GUI canvas, the class scans each text per line written in the file to identify what shape or colour it needs to draw by calling the 'Draw' class. It is used in the method from Controller class that focuses on opening files.

DrawLine

A line functionality class that assembles line drawing by capturing the x and y coordinates of the mouse. Once the mouse is pressed on the canvas, dragged to a relative distance and released, the line is drawn using the initial coordinate pairs (x,y), and the final coordinates upon releasing the mouse. The class interacts with Controller's action event method for handling line button.

DrawPlot

The plot function is itself a class that allows plot points to be drawn onto a canvas via mouse clicking. It only requires a pair of coordinates (x,y) and is basically a round rectangle with 1 pixel dimensions. The class interacts with an action event method referenced to triggering the plot button in the Controller class.

DrawPolygon

Unique shapes can be created with the help of the polygon functionality, derived from the DrawPolygon class. One mouse click at a time assists in forming a desired shape until a decision is made to deactivate the drawing function, collecting as many coordinates as it can. References of the class is shown in the Controller class' action event method for the polygon button.

DrawRectangle

When drawing on the canvas, a rectangle can be formed as the mouse is pressed, dragged along the canvas, and released when satisfied. It takes at an initial point a pair of coordinates from the mouse's location and eventually the width and height of the rectangle after releasing the mouse at a final location. The DrawRectangle class interacts with the Controller class from a method that handles clicking the rectangle button on the GUI.

fileChooser

A support class which assists methods in the Controller class directing decisions for opening and saving files via file chooser. For opening files, the class will use the 'Open' method to make sure files are able to be open with the correct file chooser dialog and vice versa. It also returns the file path and filename for each method in the class, which is passed through the Controller class ready to process and fulfil the function.

fileReader

A class that contributes to loading .VEC files by reading its data and storing it onto the GUI's TextArea. It is referenced in the opening file method from Controller Class.

ReEnableButtons

Contrary to disabling buttons, the class re-enables all buttons when a toggled button is turned off, allowing full access to all buttons once again. Until a shape button is toggled, the class can be activated again, and is referenced in all methods that implement shape drawing in Controller class.

Save

The primary role of the Save class is to capture and save vector drawings, which grabs the console GUI texts and either appends it to an existing `___.VEC` file's data or writes the information

onto a new established file. This class interacts with Controller class' file save and save as methods.

Tool

The Tool class acts as a connectivity between the backend and the user interface, more commonly in canvas and colours for arguments.

Folder: userInterface

icons Folder

A set of .PNG images used to import and illustrate the GUI's design, stored in a menu bar and tool folder.

Controller

An essential class which holds together the GUI and all classes listed in the report the Controller handles conditional settings throughout the GUI, such as button events for all existing buttons, a key event to undo a drawing using CTRL + Z and interacting the file menu for action events of making new files, opening, saving and save as, or closing the application. It's connected to the GUI through interactions from 'ui_layout.fxml'.

Main

As the entry point of the program, this class is able to launch the application and demonstrate the functionalities from all classes and files. It is also where the GUI is setup, as it imports all the fundamental files together which are the 'ui_layout.fxml' for the GUI structure, and cascading stylesheets for buttons and menu files to make the GUI look better.

menuBarStyleSheet.css

In help of styling the menu bar, this CSS stylesheet locates each image in the menu bar folder where each file menu is assigned with an icon via style class labelling, which is instructed so in the stylesheet.

stylesheet.css

Icons for each button are imported to visually show the distinctive features based off an image. The buttons are labelled in the .fxml file where the stylesheet is able to locate all style classes and implement icons into each one of them.

ui_layout.fxml

The mark-up language file that incorporates every button, screen, size, colour, and layout of the GUI; the structural design of how the GUI is made.

Advanced Object-Oriented Programming Principles

Abstraction

Encapsulation

Inheritance

Each drawing tool inherits from a super class called tool, The classes inherit include the getting of the Current canvas being used, the current state of the fill button and the colour from the ColorPicker

Polymorphism

The classes fileChooser(), Save() and Tool() incorporate encapsulation. Filechooser has two child classes Open() and Save(). Save has the child class of SaveAs().

Software Demonstration

Launching the Program

The vector drawing tool can be opened in IntelliJ, by importing the files and folders of the app first to ensure it can run the program. You will need:

- JDK 11
- JUnit 5.3
- JavaFX 12 SDK

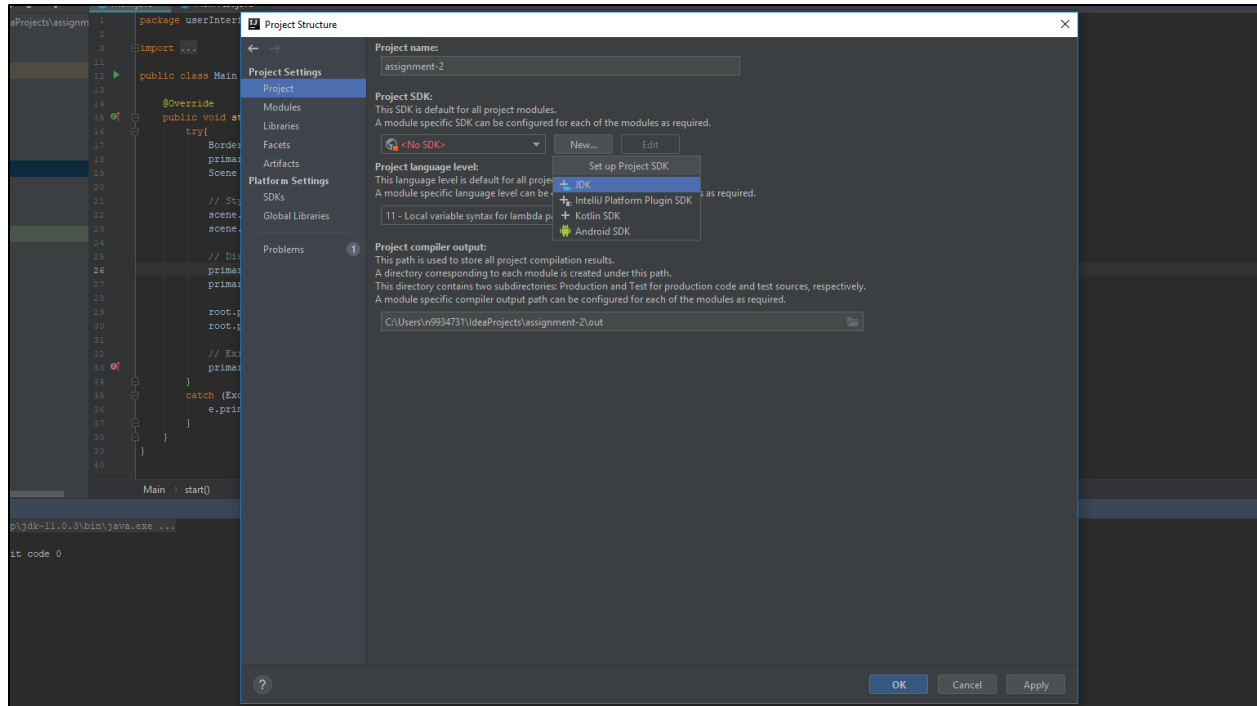
JDK 11 Guide

The software is created with JDK 11, so it's important to use this version for this demonstration. It is recommended to download the .zip file of the JDK:

<https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html>

Once the .zip file is extracted, the JDK folder can placed anywhere on the computer. IntelliJ will try to look for the files when launched. To do this, click on 'File' and 'Project Structure'. Right into the 'Project' tab, you can insert a new JDK and locate the JDK 11 folder that was downloaded by clicking the 'New...' button and then from there find the folder by clicking 'JDK'. After that, finish off by clicking 'Apply'.

CAB302 Assignment 2: Group 002



JUnit 5.3 Guide

Tests were created to showcase potential loopholes and bugs that could cause errors to the program. IntelliJ requires the latest version of JUnit which in this case is 5.3. To download this, go to the 'test' folder and open any class that is in the folder. The `@Test` annotation will be marked as an error (red), so to fix this, press 'Alt Enter' to get a lightbulb icon and then choose 'Add junit.jar to the classpath'. Now download the latest JUnit version which allows testing to happen.

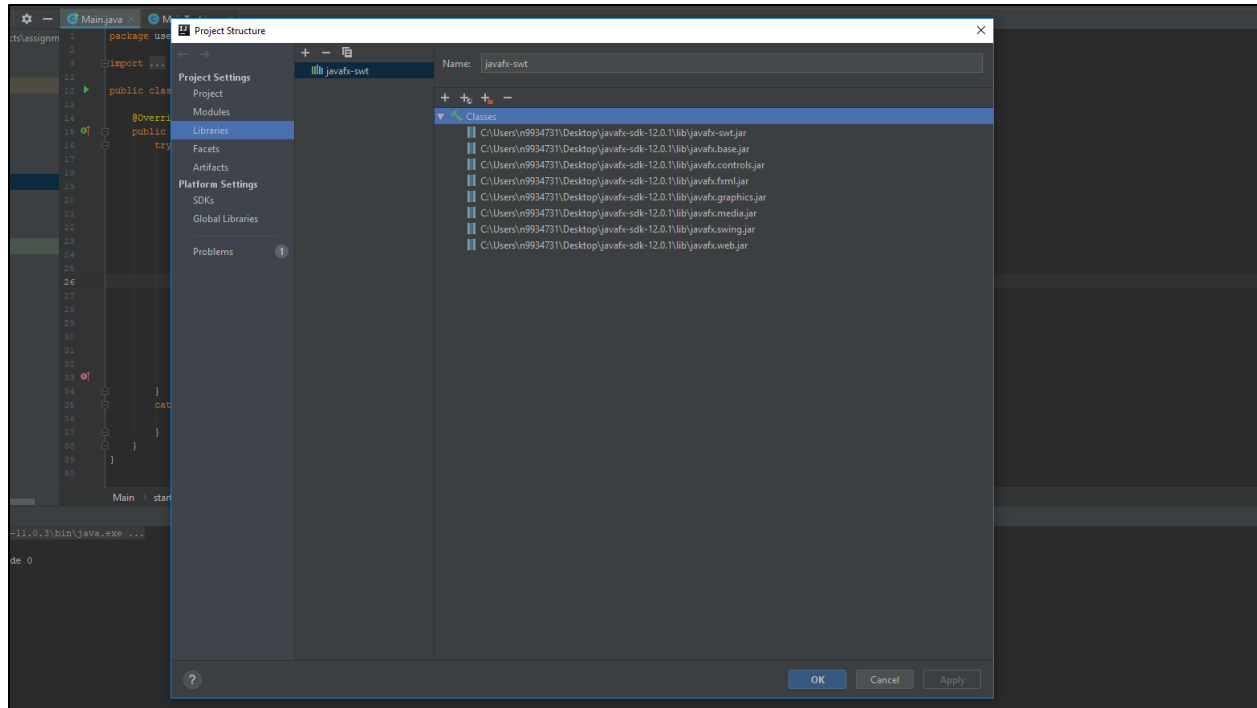
JavaFX 12 SDK Guide

To set up the program, you must download and import JavaFX 12 SDK onto your computer:

<https://gluonhq.com/products/javafx/>

To configure JavaFX in IntelliJ as a library, click on 'File, Project Structure, Libraries' and locate the JavaFX's library path by adding it as a library. Include only the '.jar' files under the 'lib' folder.

CAB302 Assignment 2: Group 002

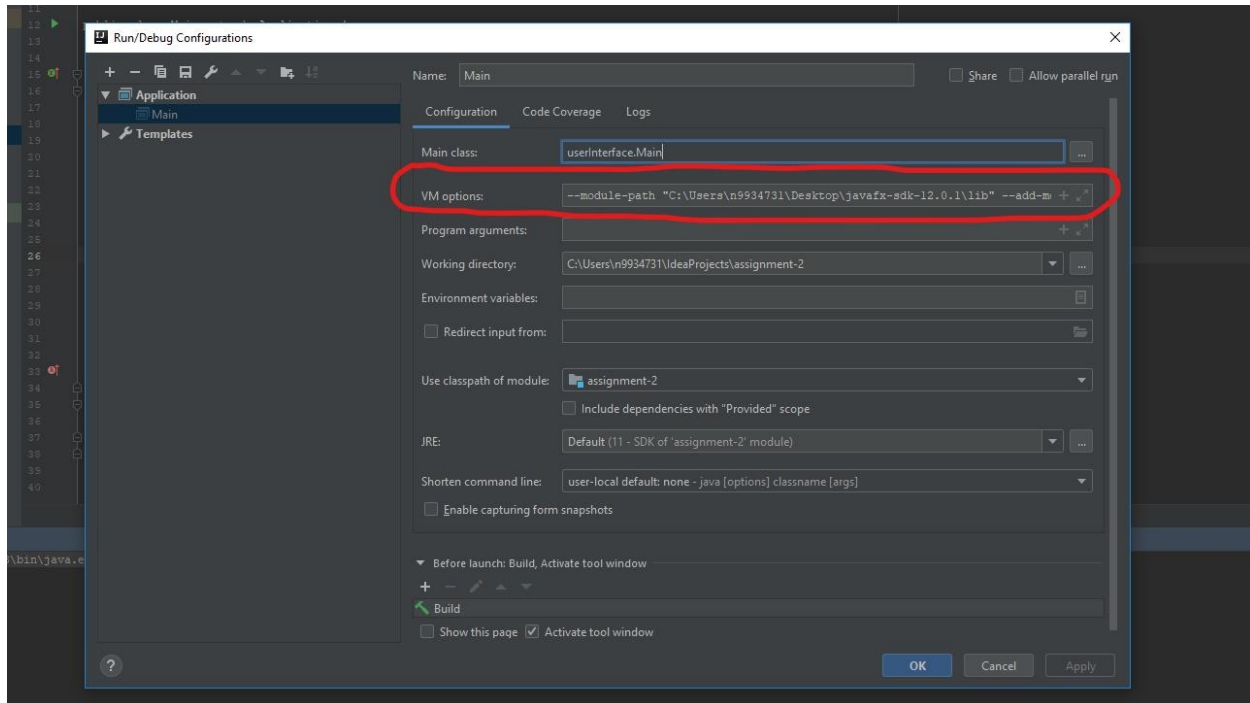


The IDE will recognize the library, but will send an error regarding to missing runtime components. To fix this, click on 'Run' then 'Edit Configurations'. Add the following VM options to the path:

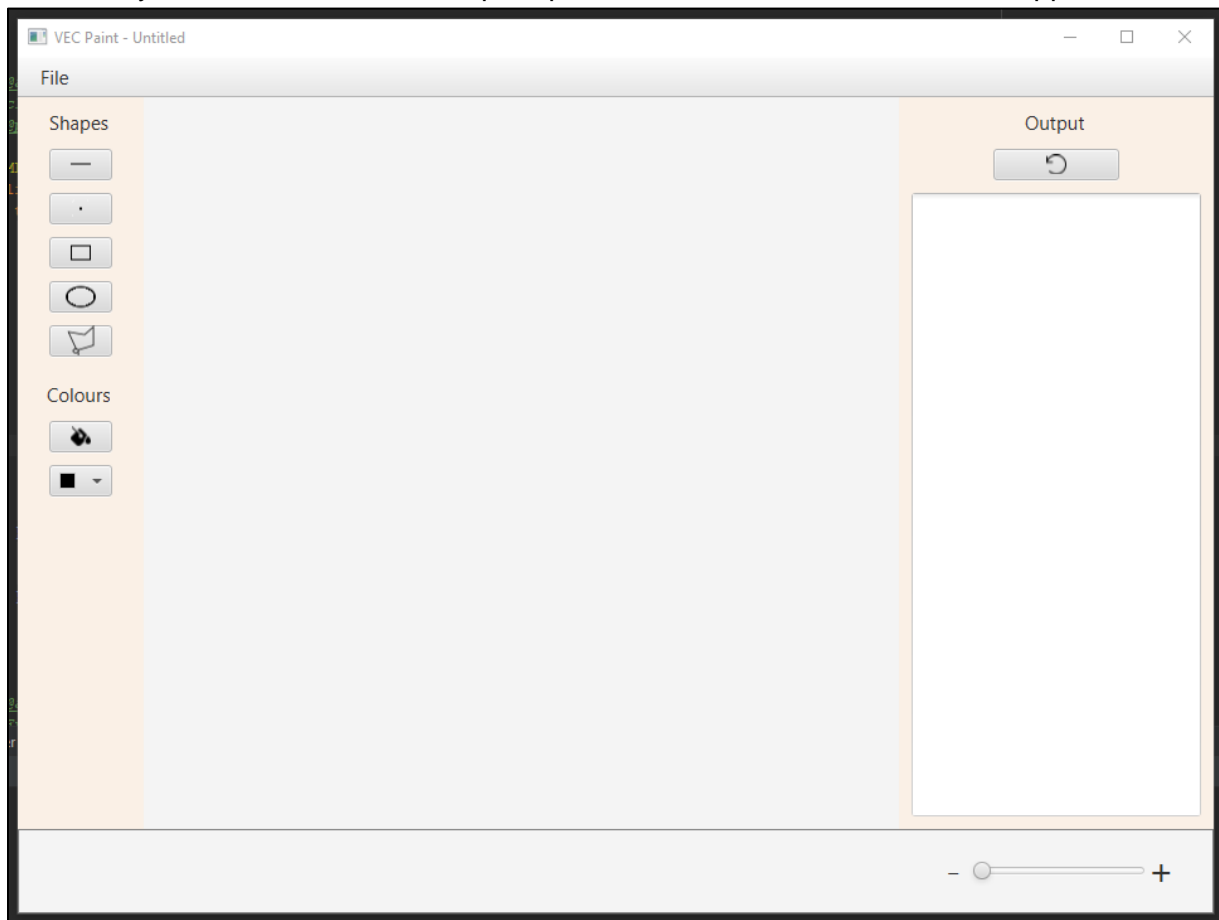
```
--module-path "\path\to\javafx-sdk-12.0.1\lib" --add-modules  
javafx.controls,javafx.fxml
```

The 'path to javafx-sdk-12' is where your file directory for JavaFX's library folder is.

CAB302 Assignment 2: Group 002



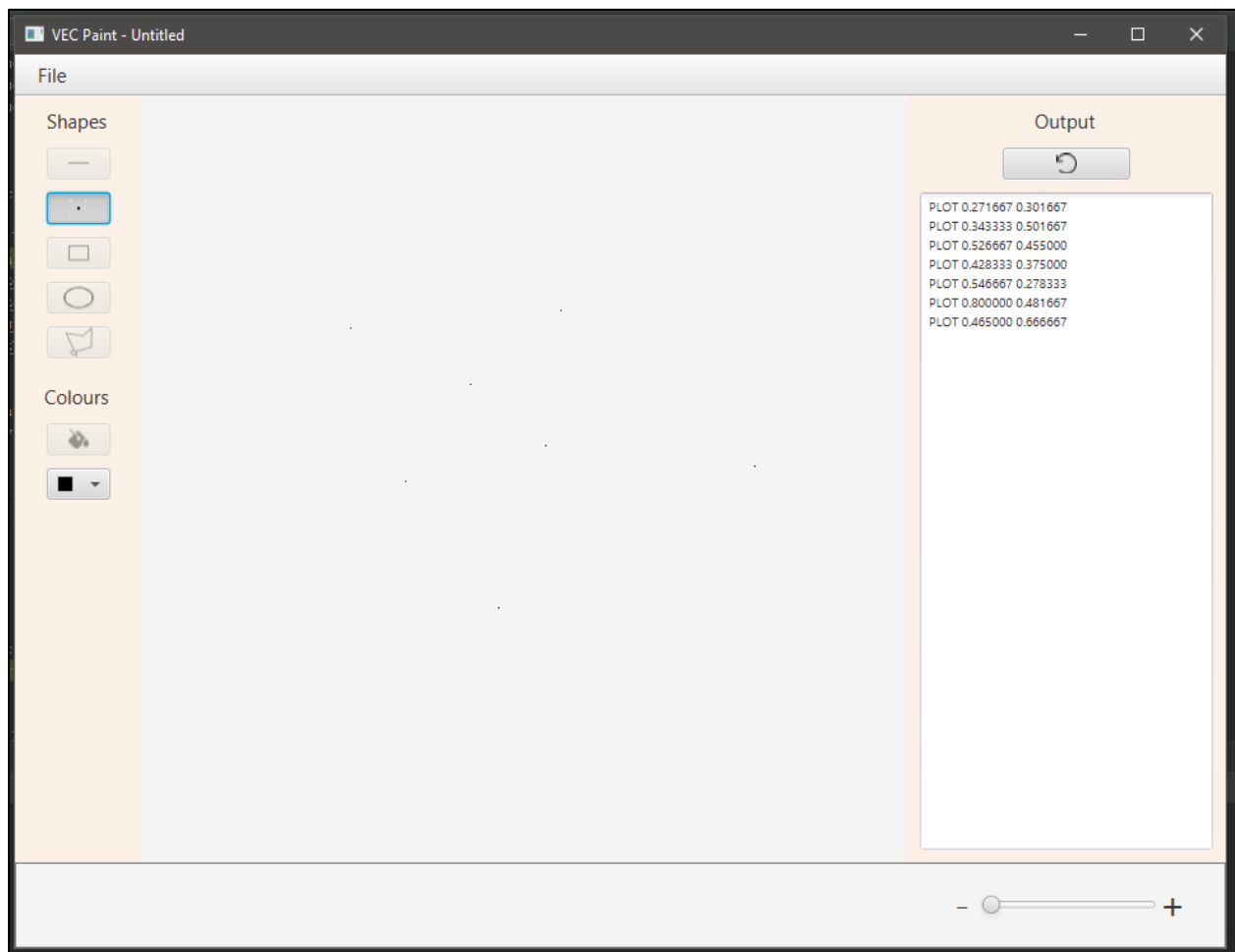
Finally, a successful launch will prompt the user the main interface of the application.



Shape Drawing Functionalities

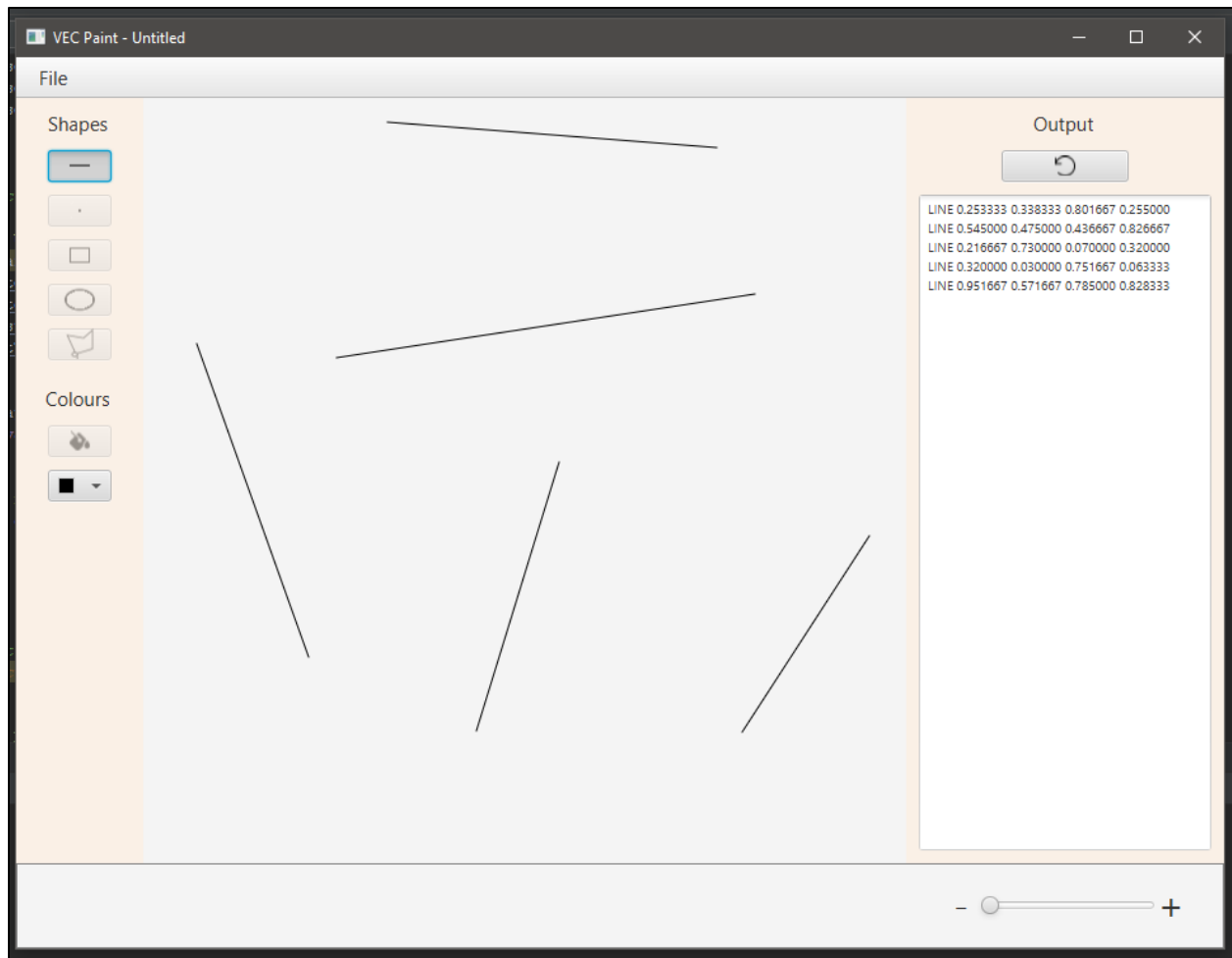
Using the PLOT Function

The plot shape allows the user to create dots on the canvas, by clicking on the dot icon in the left panel of the program. Once this function is selected, the user may now click anywhere on the canvas to draw the plot.



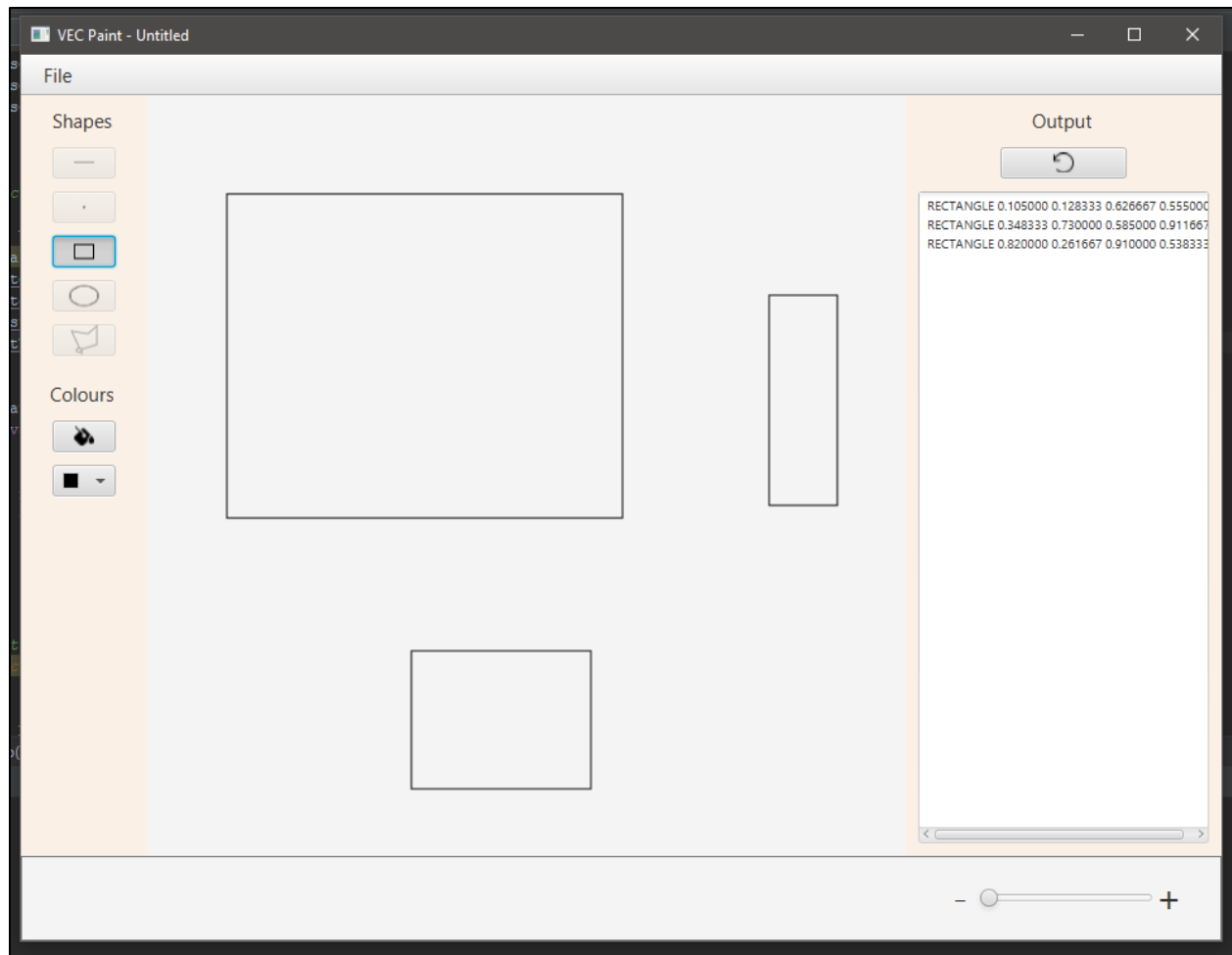
Using the LINE Function

Creating lines in the program is done by clicking the icon that displays a straight line in the left panel. Activating this function gives the user the ability to now draw lines on the canvas. The click and drag method is carried out to create the line, starting on the initial position at any area of the board, and dragging the line towards any end far from the initial contact.



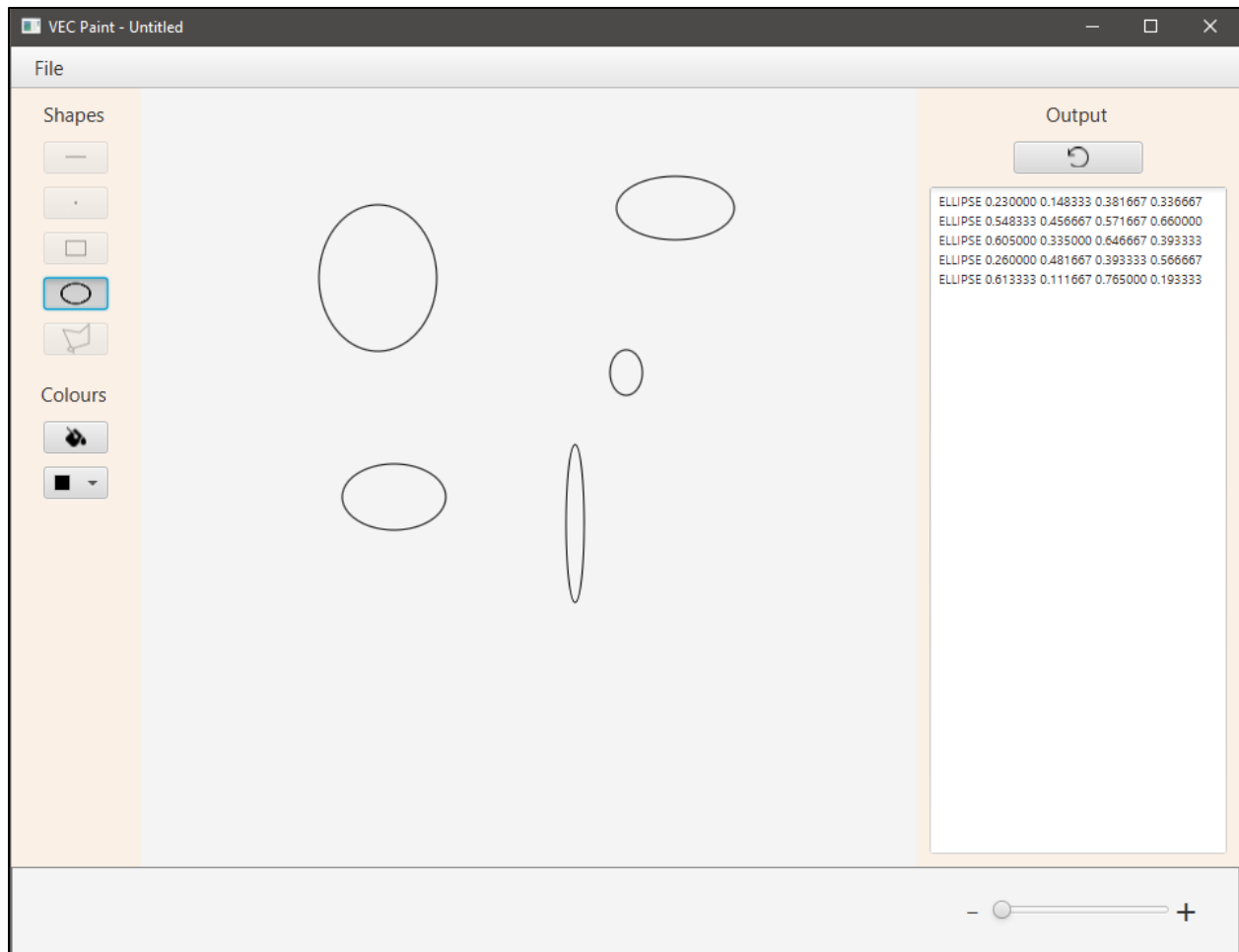
Using the RECTANGLE Function

A rectangle shape is implemented as one of the functions available in the program's panel. Toggling this function lets the user draw rectangle shapes of any sort on the canvas, by clicking and dragging the shape to its desired form, outputting the shape at the end once the user finishes holding down the left click on the mouse.



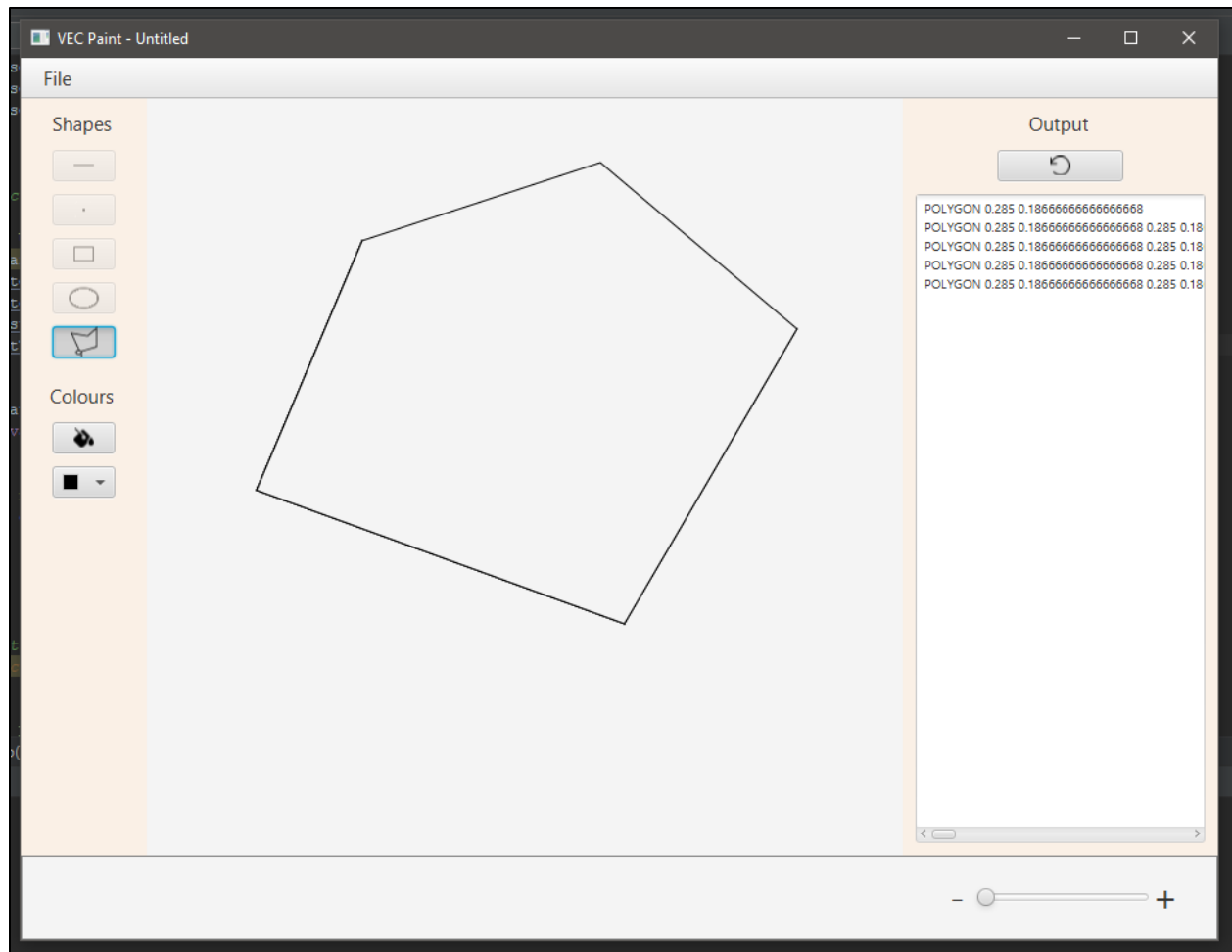
Using the ELLIPSE Function

Circles can be created when the ellipse function is toggled on the left panel. To achieve the shape of any form, users must click and hold down an initial point at a certain location of the canvas and proceed to drag the shape until they release the left click of the mouse.



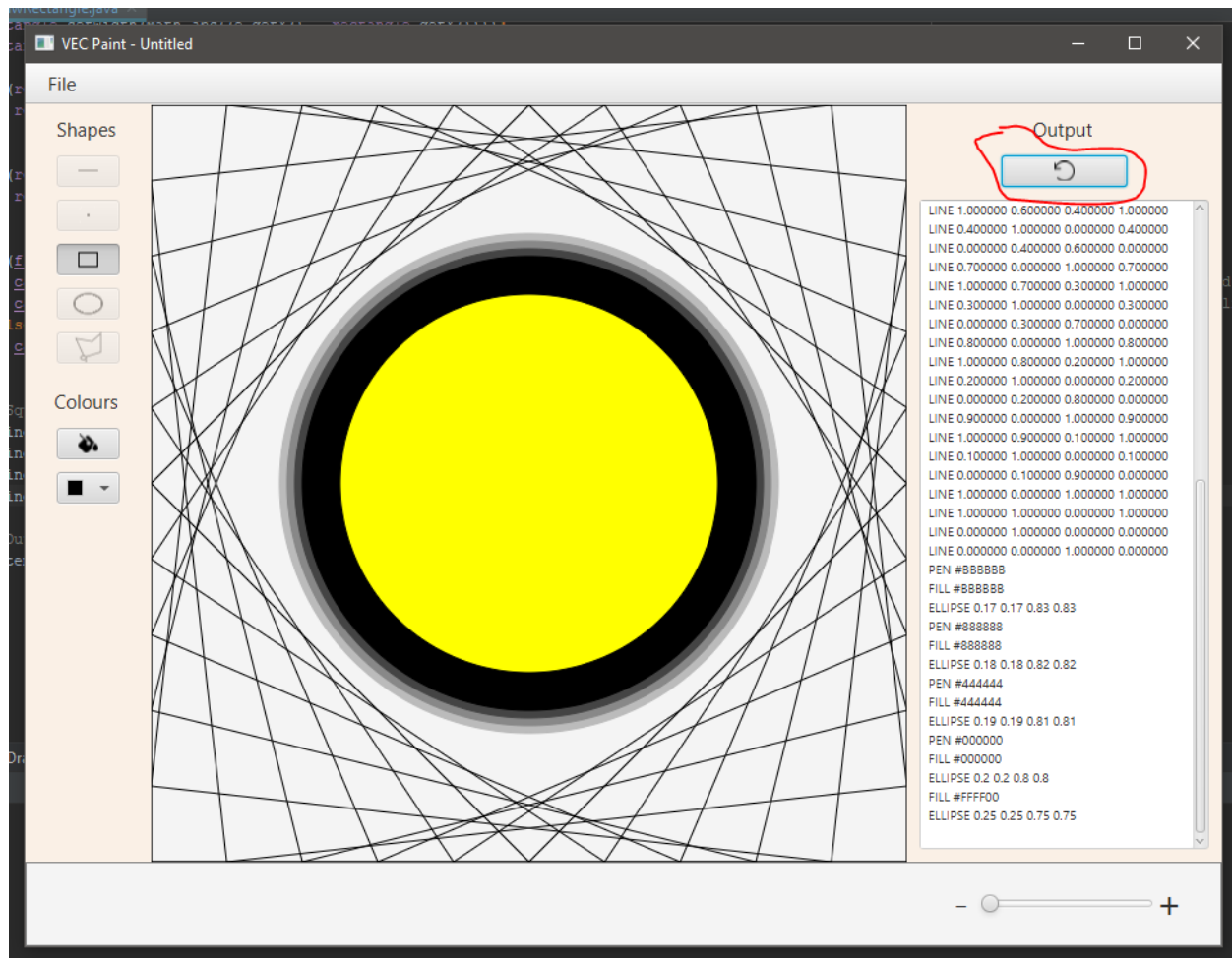
Using the POLYGON Function

The polygon function, once toggled, can create shapes on the canvas with more than 4 points. The user can create any form of shape by simply clicking at each location of the canvas, which subsequently draws up lines connecting each coordinate.



Revert Action using Undo

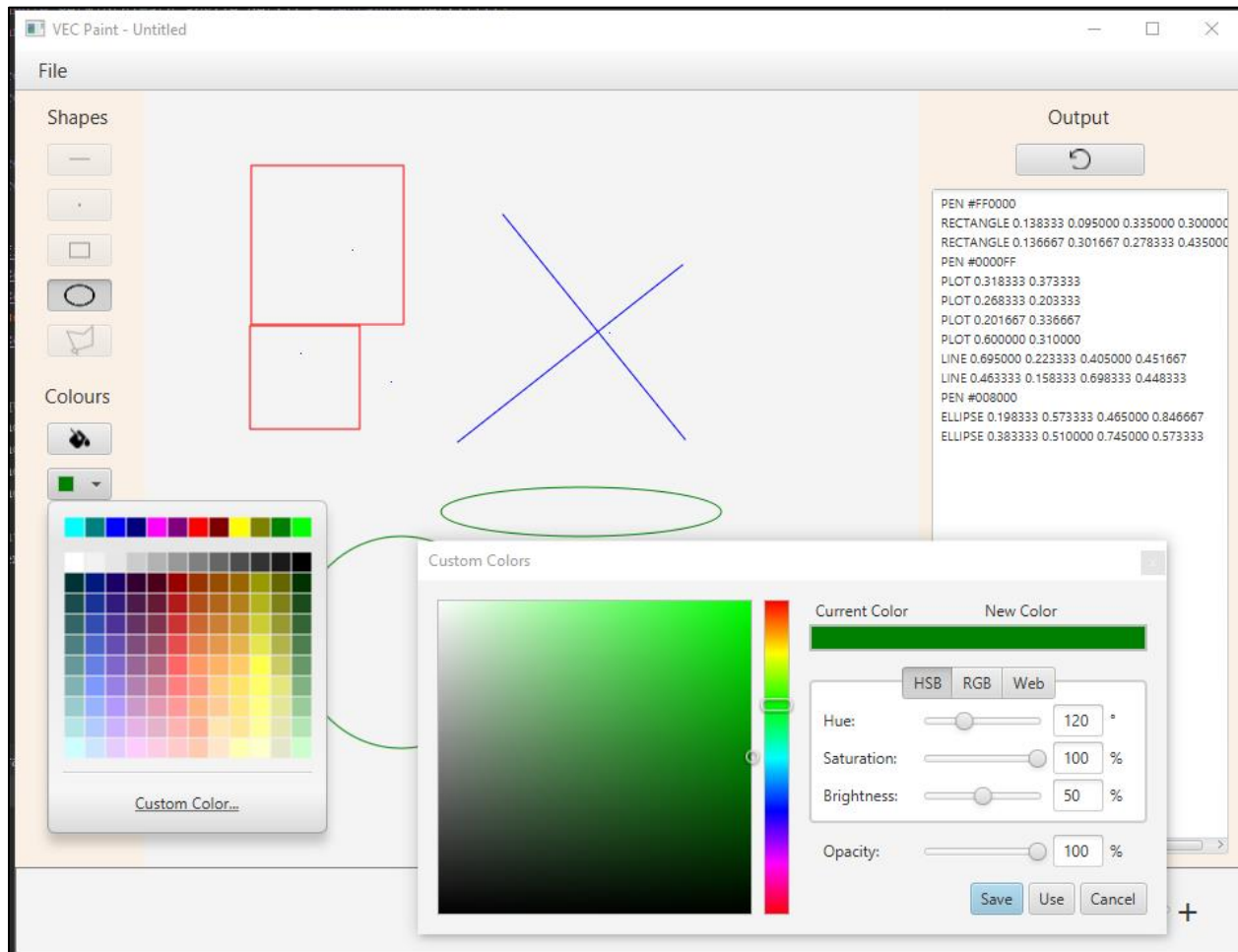
If the user wishes to undo the latest action that is committed to the drawing, they can simply click the undo button that is on the right panel above the console output. This feature can be repeated at a point where the VEC file is first created and is blank.



Colour Usage

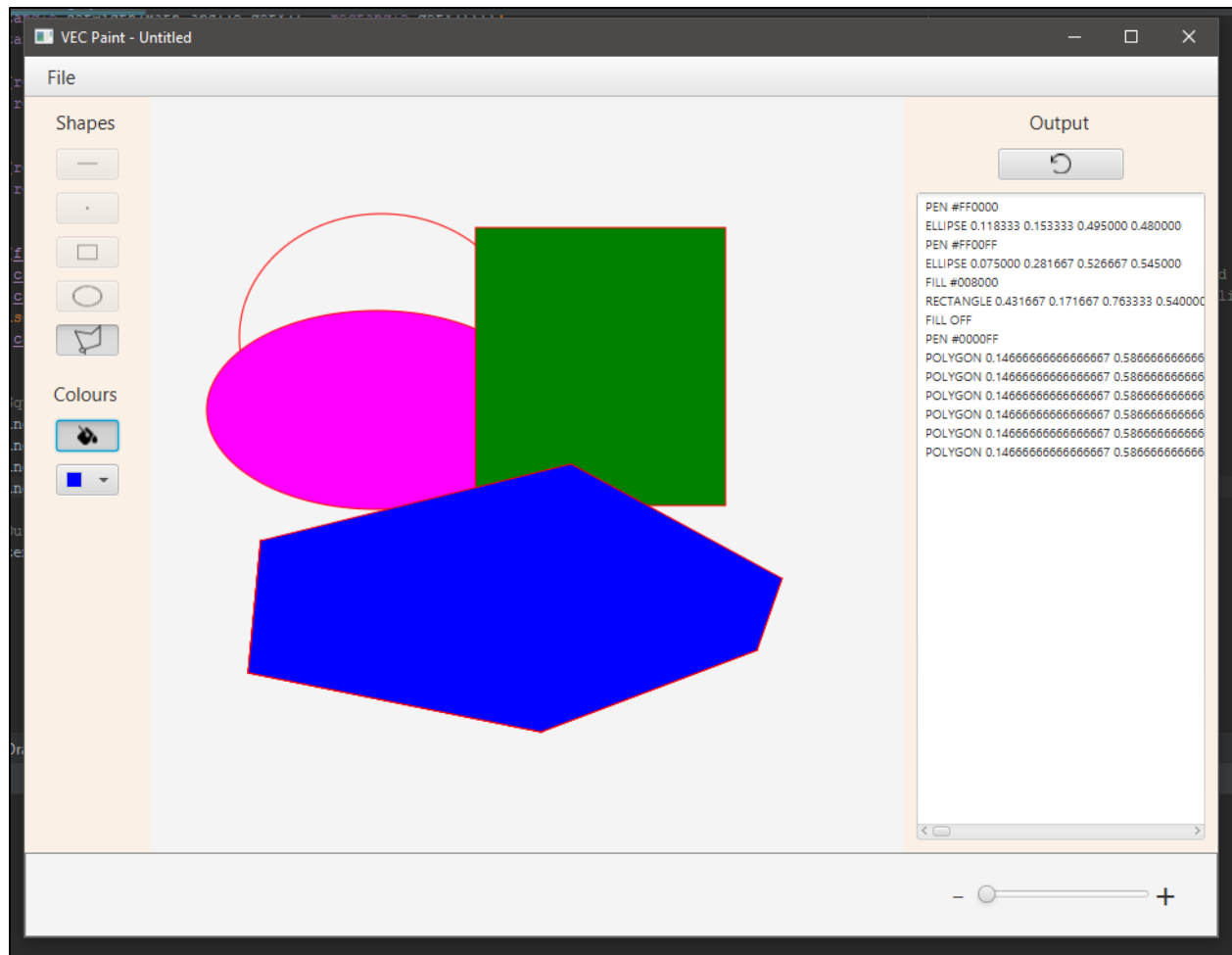
Selecting a Colour

Shape drawings can have different colours when choosing a colour in the palette below the canvas window. Once the user has selected a colour, the pen is updated with a new colour to draw with. Users can also select a variety of colours next to the colour palette. This gives the user a choice to pick out a custom colour based on the colour gradient and hue, or inputting numbers and RGB values for precise preferences.



Fill Bucket Tool

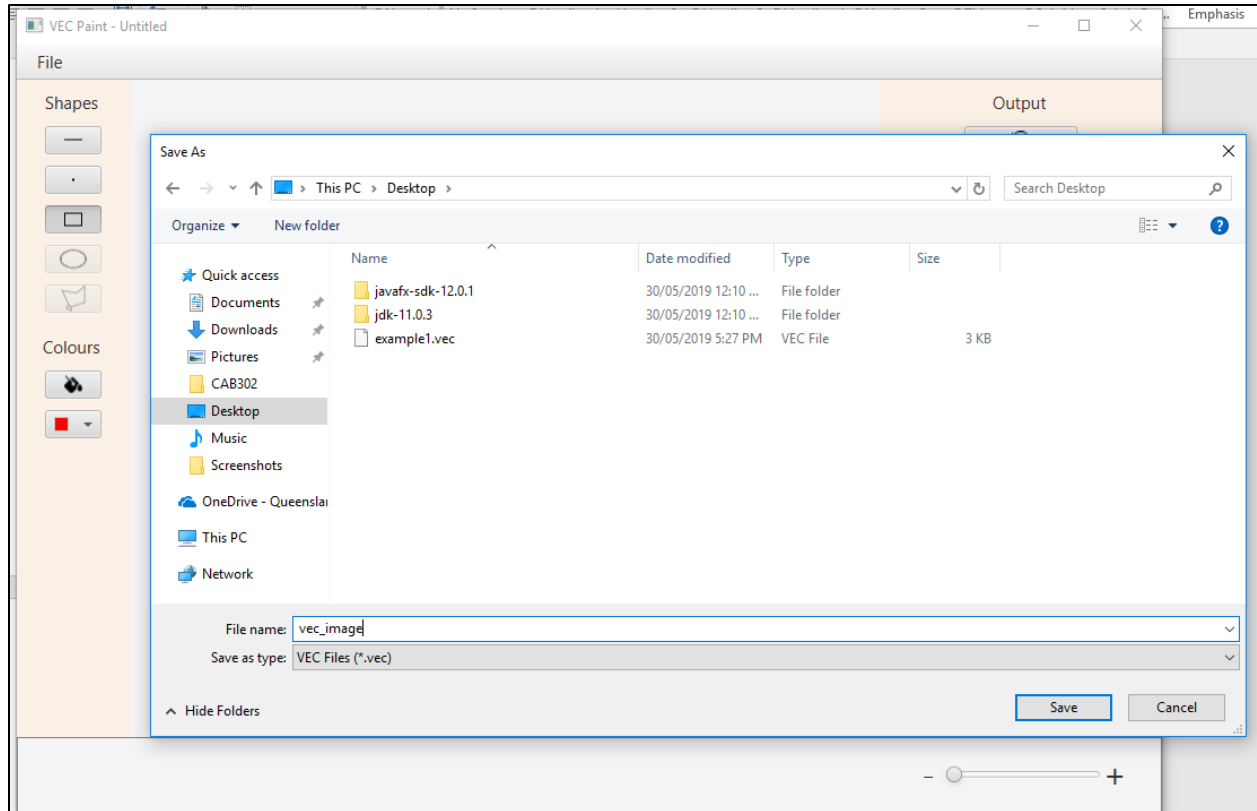
One tool available to use in the software is the fill bucket function, available to toggle on the left panel beneath the shapes functions. Toggling the feature along with any colour allows the user to create shapes that are filled with their chosen colour. This also does not interfere with pen outlines where the user picks two distinctive colours.



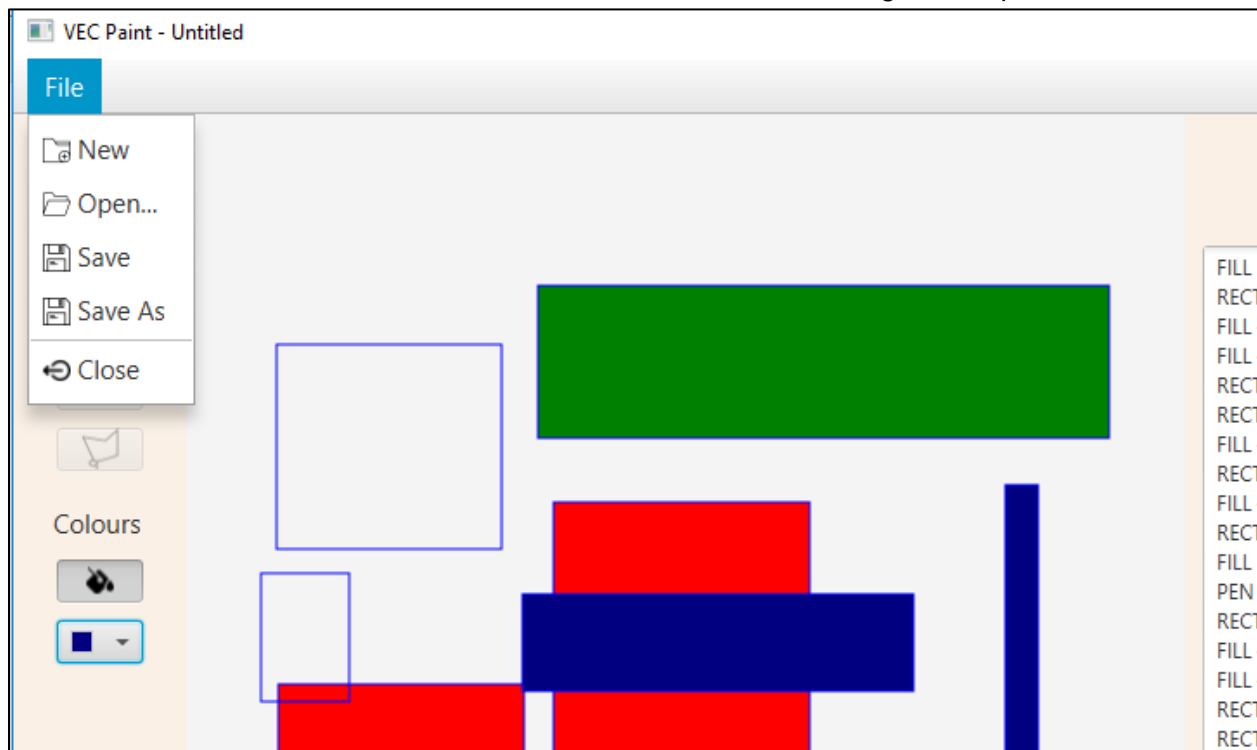
Saving a VEC Image

When users are satisfied with the drawing, they can save their work by clicking the save icon on the menu bar next to the undo button. Initially, users will have to save their work in a local directory for the program to discover and give it a name in the end.

CAB302 Assignment 2: Group 002

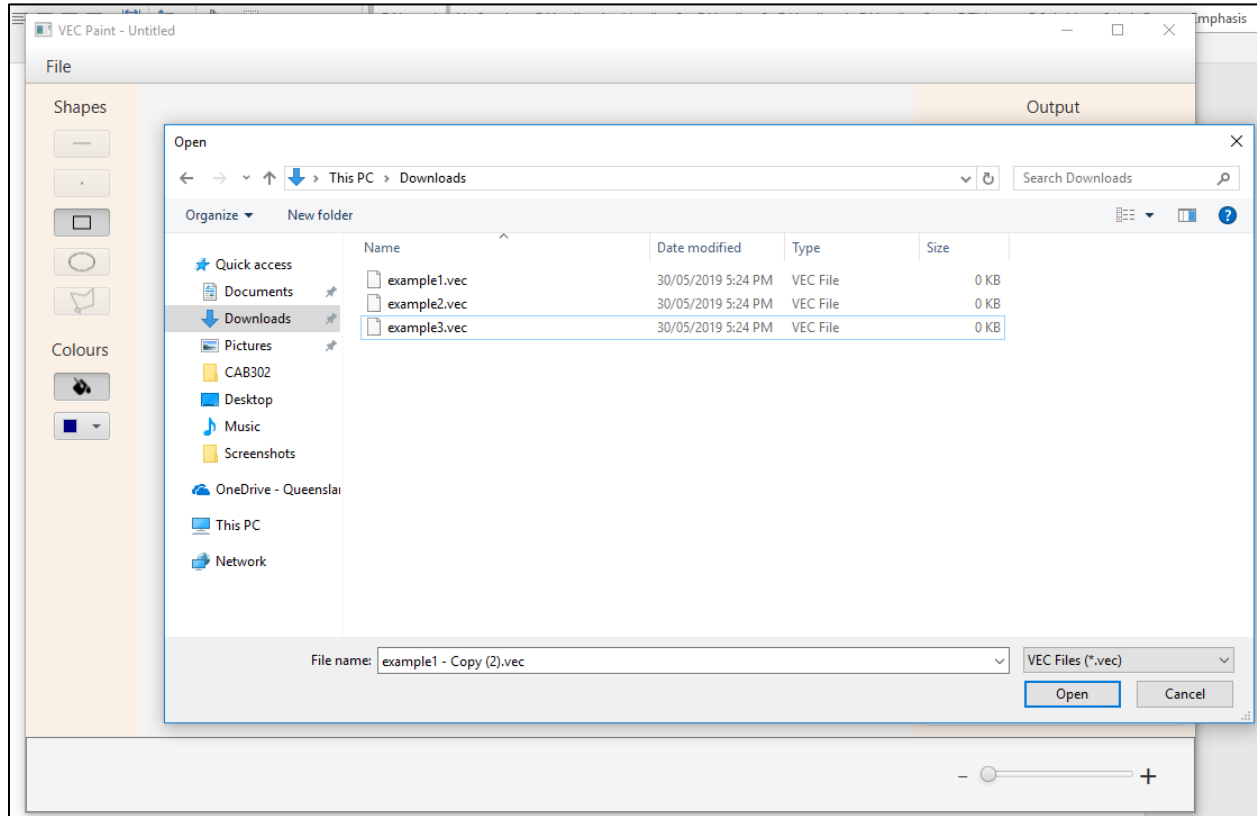


After this, the program will constantly update the file when asked to save it. Another way to save the vector image is by clicking on 'File' and 'Save'. The 'Save As' as mentioned earlier, allows the user to save the file based on a different name and location, creating an independent file.



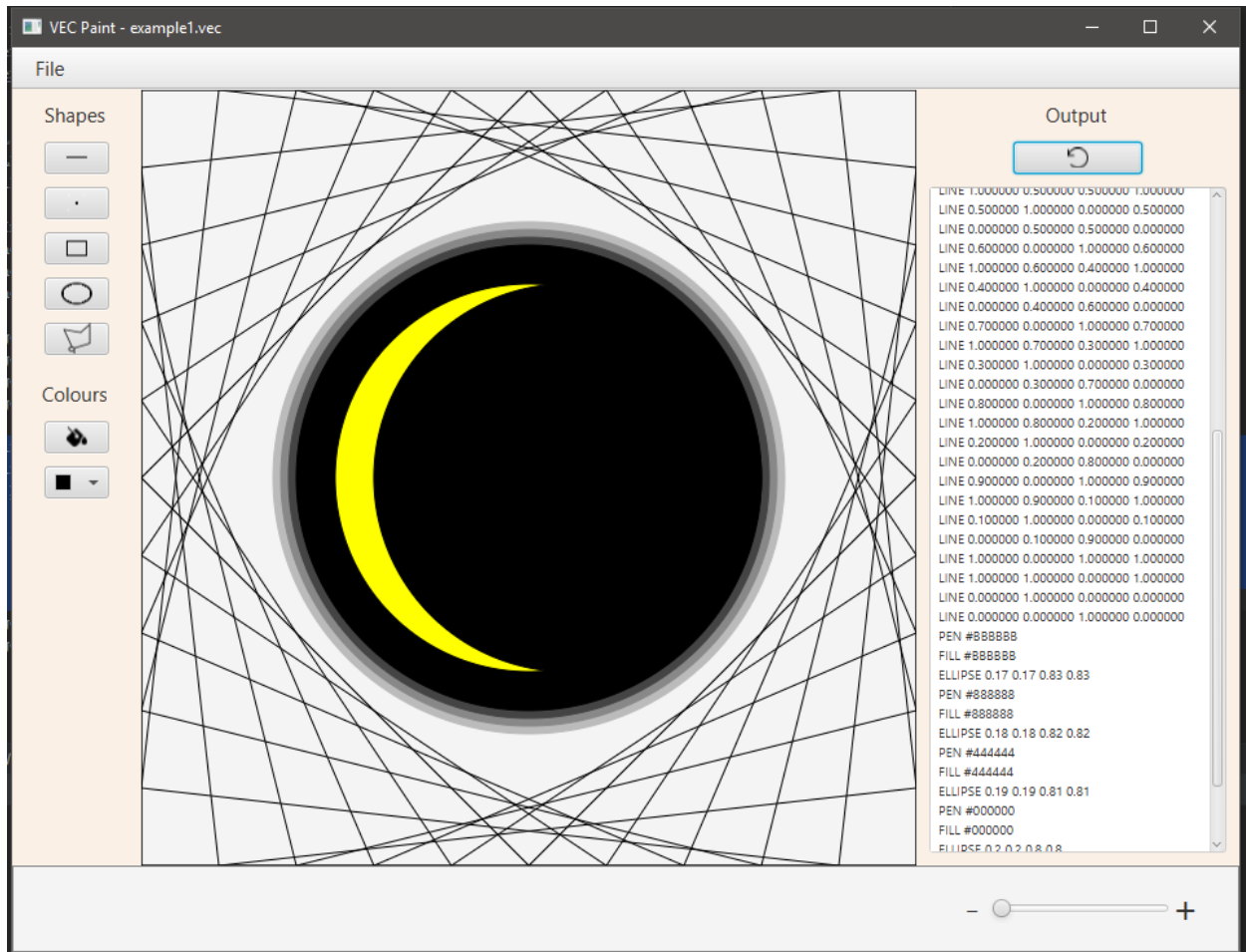
Opening a VEC Image

A file with the .vec extension can be opened with the program tool. To open files, click on 'File' and 'Open', which opens the file explorer to find and choose a compatible file to open. Successful selection of a file will launch the vector image into the canvas.



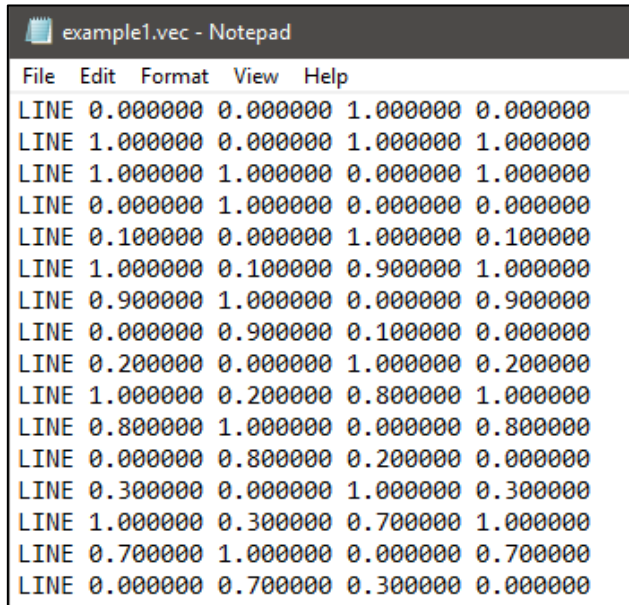
CAB302 Assignment 2: Group 002

Upon Successful File load:

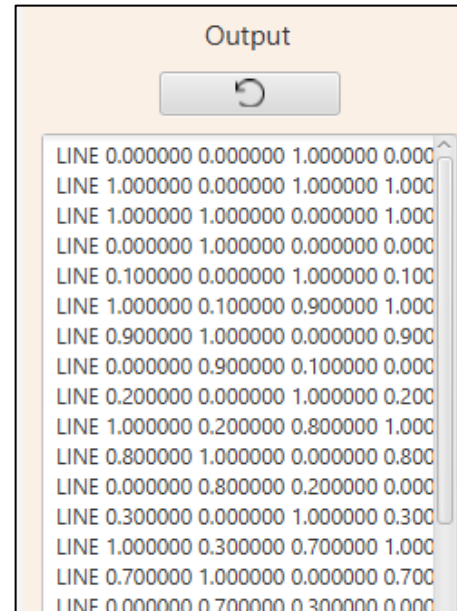


CAB302 Assignment 2: Group 002

Furthermore, a .vec file can be opened with a text editor which allows the user to preview each line of code the image requires to create, or simply preview the console which outputs the loaded code.



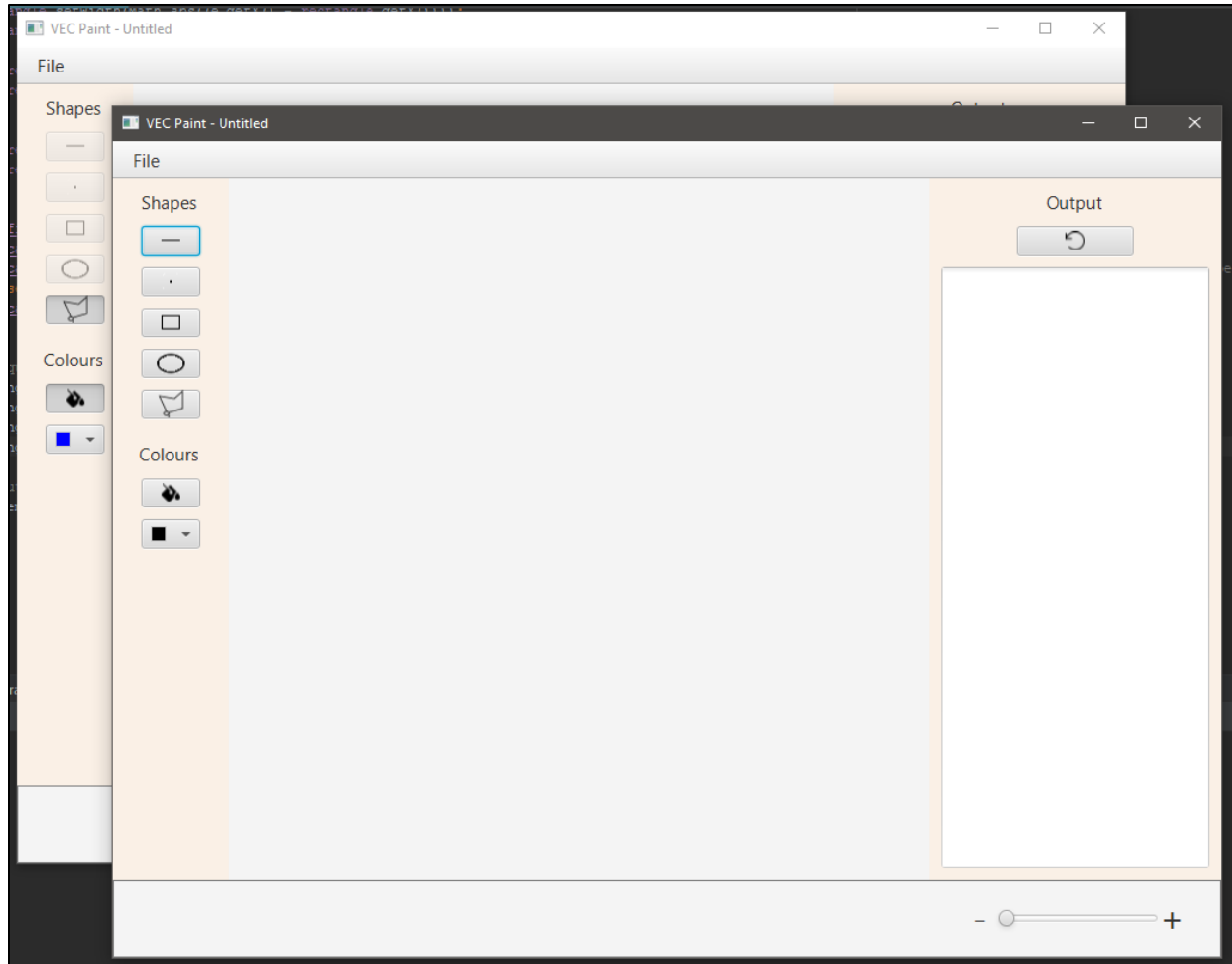
```
example1.vec - Notepad
File Edit Format View Help
LINE 0.000000 0.000000 1.000000 0.000000
LINE 1.000000 0.000000 1.000000 1.000000
LINE 1.000000 1.000000 0.000000 1.000000
LINE 0.000000 1.000000 0.000000 0.000000
LINE 0.100000 0.000000 1.000000 0.100000
LINE 1.000000 0.100000 0.900000 1.000000
LINE 0.900000 1.000000 0.000000 0.900000
LINE 0.000000 0.900000 0.100000 0.000000
LINE 0.200000 0.000000 1.000000 0.200000
LINE 1.000000 0.200000 0.800000 1.000000
LINE 0.800000 1.000000 0.000000 0.800000
LINE 0.000000 0.800000 0.200000 0.000000
LINE 0.300000 0.000000 1.000000 0.300000
LINE 1.000000 0.300000 0.700000 1.000000
LINE 0.700000 1.000000 0.000000 0.700000
LINE 0.000000 0.700000 0.300000 0.000000
```



```
Output
[Refresh Button]
LINE 0.000000 0.000000 1.000000 0.000
LINE 1.000000 0.000000 1.000000 1.000
LINE 1.000000 1.000000 0.000000 1.000
LINE 0.000000 1.000000 0.000000 0.000
LINE 0.100000 0.000000 1.000000 0.100
LINE 1.000000 0.100000 0.900000 1.000
LINE 0.900000 1.000000 0.000000 0.900
LINE 0.000000 0.900000 0.100000 0.000
LINE 0.200000 0.000000 1.000000 0.200
LINE 1.000000 0.200000 0.800000 1.000
LINE 0.800000 1.000000 0.000000 0.800
LINE 0.000000 0.800000 0.200000 0.000
LINE 0.300000 0.000000 1.000000 0.300
LINE 1.000000 0.300000 0.700000 1.000
LINE 0.700000 1.000000 0.000000 0.700
LINE 0.000000 0.700000 0.300000 0.000
```

Creating a new VEC File

A new file can be created by clicking 'File' and then 'New'. After saving the image, the program creates a blank canvas and launches a new window.



Zooming in on Canvas

A canvas can be zoomed in and out to closely inspect the drawing. On the bottom right corner of the program, there is a feature which allows the user to increase or decrease the magnification of the canvas, by also reflecting the action with a percentage value from magnifying the image, with 100% being the default.

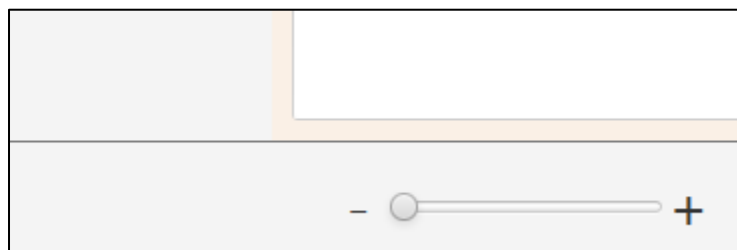


Image Code & History Log

A vector image is made possible with lines of code generated from the user. These codes are based off the creation of the shapes and updates of tools, outputting the type of action performed and the coordinates of the shapes or #RRGGBB values of colours used. Below is a list of functions and their syntax for displaying the drawing:

- LINE with arguments [x1] [y1] [x2] [y2]
- PLOT with arguments [x] and [y]
- RECTANGLE with arguments [x1] [y1] [x2] [y2]
- PEN with argument #RRGGBB
- FILL with argument #RRGGBB
- POLYGON with arguments [x1] [y1] [x2] [y2]... [x_n] [y_n]

```
LINE 0.113333 0.256667 0.376667 0.121667
PLOT 0.208333 0.333333
RECTANGLE 0.381667 0.335000 0.505000 0.438333
ELLIPSE 0.131667 0.541667 0.175000 0.641667
POLYGON 0.3633333333333334 0.555
POLYGON 0.3633333333333334 0.555 0.3633333
PEN #FF00FF
FILL #008080
```