Development Report and Documentation

CAB302 - Assignment 2

Alex Butler, Bradley Fuller & Joshua O'Riordan

June 2, 2019

Contents

1 Statement of Completeness

The solution includes the following features:

Drawing - Line	Complete - Tested & Working
Drawing - Plot	Complete - Tested & Working
Drawing - Rectangle	Complete - Tested & Working
Drawing - Ellipse	Complete - Tested & Working
Drawing - Polygon	Complete - Tested & Working
Drawing - Pen Colour	Complete - Tested & Working
Drawing - Fill Colour	Complete - Tested & Working
File Handling - Read	Complete - Tested & Working
File Handling - Write	Complete - Tested & Working
GUI - Layout	Complete - Tested & Working
GUI - Buttons	Complete - Tested & Working
GUI - Menu Bar	Complete - Tested & Working
GUI - Colour Picker	Complete - Tested & Working
Additional - Undo History	Complete - Tested & Working
Additional - Bitmap Export	Complete - Tested & Working

By the due date, we were able to complete all of the essential requirements and the required amount of two additional requirements - being Undo History, and Bitmap Export.

2 Statement of Contribution

2.1 Alex Butler

Alex implemented all of the Shape classes, as well as the Shape interface and enumerator. He also wrote the unit tests for testing each Shape class, and the first iteration of the GUI (mainly for testing purposes).

2.2 Bradley Fuller

Bradley implemented the file handling and validation classes, as well as the unit testing for said classes. He also wrote and configured the Apache Ant build script, and the TravisCI configuration for continuous integration and delivery to GitHub Releases. He also wrote most of the report, excluding Section 6, and did some general cleanup and changing of the solution.

2.3 Joshua O'Riordan

Joshua implemented the final iteration of the GUI, including the general layout, buttons, menu bar and colour picker. He also wrote the additional requirements functionality (being Undo History and Bitmap Export), as well as their respective unit tests. Joshua wrote the User Guide in its entirety (Section 6 of this document).

3 Software Development Practices

The team utilised Agile's philosophy of iterative and incremental changes, through the use of continuous integration and deployment. This was achieved through the use of TravisCI and GitHub Releases. Whenever a change was made in any branch of the repository, TravisCI would create a temporary Docker container with Ubuntu 16.04 and Oracle JDK 11. All dependencies are then downloaded - in our case, JUnit 5.5.0 and JavaFX, as well as Apache Ant 1.10.6. Ant is used for creating all build folders, compiling the solution, and then running all unit tests (which are stored in the src/tests/ directory). We would then receive a report (via email) of the testing and if any errors occurred. However, if the change was made in the "master" branch of the repository, TravisCI would run Ant's "dist" script, which packages the compiled Java classes in to a JAR file. This JAR file would then be given a version number (which during development was TravisCI's build number) and uploaded to GitHub Releases. Pull requests would also be tested against the "master" branch of the repository to ensure there would be no breaking changes.

Due to other units, assessment deadlines, and other commitments, we were not able to follow Agile's sprint methodology and strict timing requirements exactly. However, each feature of the solution was completed in sprints, each being in their own branch in the repository. Once these were complete, the branch would be merged in to master, and work on the next feature could begin. The only exception to this was the File Handling features, as they required the GUI to be fully implemented before they could be merged - however they were tested thoroughly beforehand without the use of a GUI, so the integration was very simple. We also had a kanban board of features that were to be implemented in the project, as well as an automated kanban board for bug triage and fixing.

Test-driven development was also used where possible. The file reading class had over 20 tests of it's own, to make sure that exceptions were being thrown when they should have. Different test cases were used such as invalid commands, invalid co-ordinates, invalid colours, non-VEC file tests, incorrect file types etc. Even though it would have been possible, we decided against testing the GUI, mostly due to time constraints, and not getting much actual value from testing the GUI.

4 Software Architecture

4.1 Components

The main class here is the Components class (ComponentsClass). It provides the drawing canvas and methods, as well as a shared set of functions that all shapes utilise in order to be drawn. In addition, it includes the Undo list and logic (not to be confused with Undo History) which is used extensively throughout the program. The Ellipse and Rectangle shapes are included in the ShapesComponent class, as each of these shapes only require two sets of co-ordinates to draw. The Line, Plot and Polygon shapes each have their own class as they have a different set of drawing requirements. The Component interface is only implemented by the ShapesComponent class in this iteration of the solution. It is included as future shapes may follow a similar format to the ShapesComponent class, but have slightly different calculations and/or features - in which case the Component class can be utilised. Finally, the ShapesEnum simply includes a list of all shapes.

4.2 File Handling

There are only two classes in the file handling package - FileRead and FileWrite. FileWrite simply writes the ArrayList provided by the logic in the main window to a file using a BufferedWriter. This is because access to the canvas and ComponentClass is required, and is not reachable from the FileWrite package. The FileRead class is much more complex, as most of the validation is performed in this class. The main function, readFile, takes a file path as a string. It then uses a FileInputStream and a Scanner to parse through the file. This was chosen due to an advantage in performance and memory management, as the entire file is never loaded in to memory. Each line is thoroughly validated, checking for valid shapes, colours, co-ordinates and file types. If all checks pass, the line is loaded as a split array in to an ArrayList. Once the file has been completely loaded, the ArrayList is passed through to the MainWindow logic which then creates shape objects for each loaded shape.

4.3 GUI

As with every Java project, there is a Main class - in this instance, the Main class calls the MainWindow class in order to initialise the GUI. The MainWindow class contains all of the graphical and event handler configurations and declarations. When initialised, the GUI class instantiates the ComponentsClass constructor, which in turn instantiates every Shape class and prepares them for drawing. The ShapesEnum enumerator is constructed which includes a list of all shapes. This enumerator is used to aid in drawing shapes for the GUI. Event handlers for mouse input and keyboard shortcuts are initialised in this class, in order for them to be accessible to the user. The FileRead and FileWrite classes are also instantiated as part of the GUI class - this is in order to read to and write from the Components constructor. There is some file logic in this class, however no file reading or writing actually takes place in this class at all. The other class in this package, MenuCommands, is a utility class which contains the actions corresponding to keyboard or graphical commands, such as selecting a different fill colour or a draw shape. It contains only static methods. It also includes logic for file loading and saving dialog boxes, as well as the BMP export and Undo History logic.

4.4 Miscellaneous

The Main class only exists to launch the GUI from MainWindow. All packages and classes, including test classes, are structured according to Maven's Standard Directory Layout, which can be found here.

5 Usage of Advanced Object-Oriented Principles

5.1 Abstraction

Abstraction is utilised to a large extent throughout the project. For example, the FileRead class performs multiple checks on each line to ensure validity of the file being loaded - however all checks are private and cannot be seen or used outside of the class. The Shapes package also utilises abstraction with most shapes having their own class - however each shape is drawn by the ComponentsClass.

5.2 Encapsulation

As with most software projects that incorporate the use of object-oriented practice, encapsulation has been used to a large extent in the development of this solution. As mentioned earlier, the project has structured according to the Maven Standard Directory Layout, making it easy to modify for programmers who have not previously worked on the project. The File Handling classes in particular utilise encapsulation greatly, most notably the FileRead class. It has a multitude of private functions that are only required for file validation, making it substantially easier to use the class elsewhere - for example, if an auto-loading feature was implemented, the only thing that needs to be passed to the class is the path to the file, and the only object returned is an ArrayList of arguments or commands.

5.3 Inheritance

Inheritance is mainly used in the Shapes package. As mentioned previously, the Component interface is included for shapes that may be added in future iterations of the solution. In addition to this, the Component class can be utilised for it's methods and functions to create, draw and destroy shapes, without requiring much modification. The custom File exception also uses inheritance to some respect, by throwing the exception to it's superclass. The custom Mouse Adapter extends from the MouseAdapter class - this is used to provide mouse handling and other mouse event functionality.

5.4 Polymorphism

As mentioned above, polymorphism has been used in this project, notably within the shape classes. Shapes inherit from the Component interface to ensure that each shape can be drawn the same way, and without making modifications to the main Component class. In addition, this ensures that each shape behaves how it is expected, especially when making modifications to the shape. In addition, method overriding was extensively used in the implementation of the event handlers in the GUI main class, overriding the mouse drag, click and release events.

6 Using the Software

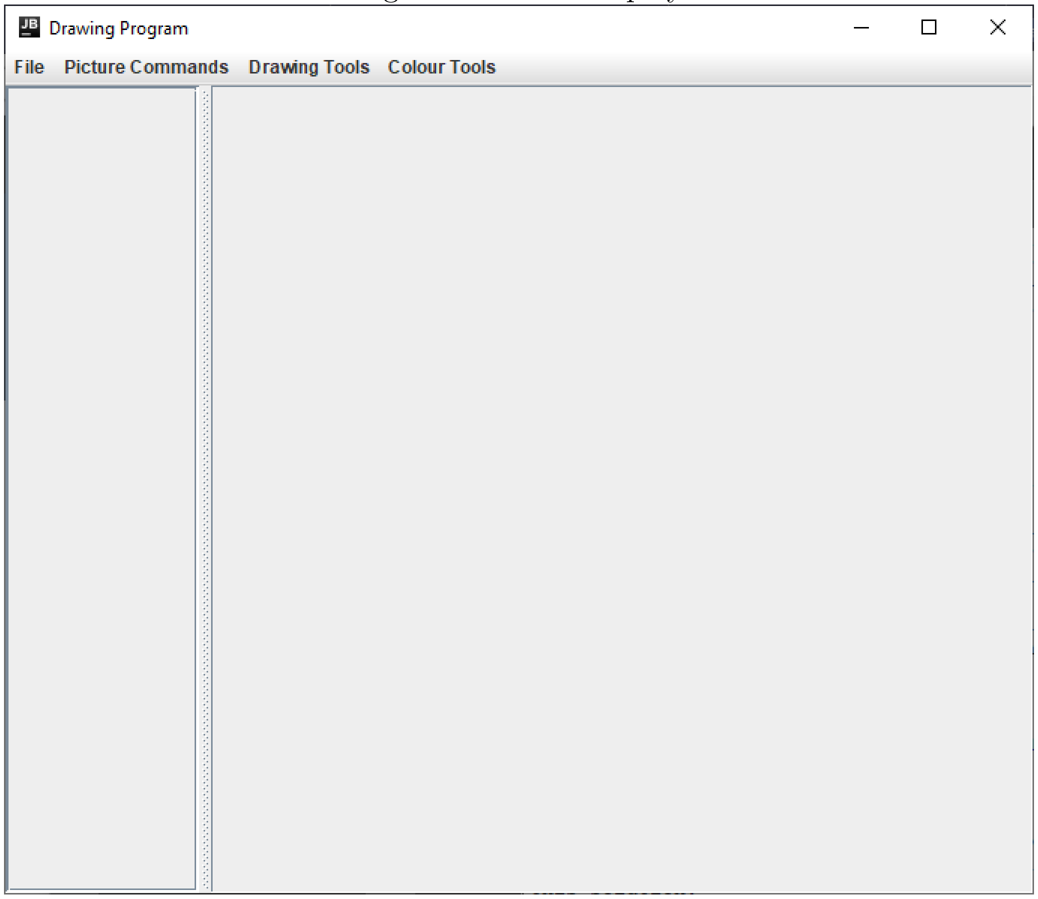
6.1 Preamble

This guide is designed to detail and instruct in the usage of the VEC drawing program. Each section describes related functions, such as file handling or keyboard shortcuts.

6.2 Running the Program

Upon opening the program a display such as that pictured is shown.

Figure 1: Blank Display



The bar on the left hand side is a history bar, and the panel on the right is the location wherein you may draw shapes. Commands are situated in dropdown bars on the top of the program in the menu bar.

6.3 Keyboard Shortcuts

Some commands have keyboard shortcuts, such as Save File. These shortcuts are shown in parentheses after the name of the command. In the case of Save File, pressing ctrl+s (ctrl and s at the same time) will open the dialogue to save the currently displayed picture. Using a keyboard shortcut is the exact same as pressing on the corresponding button.

6.4 File

The options under file handle the saving, opening, clearing, and exporting of drawn shapes. The commands available in the dropdown are: New File, Open, Save, and Export BMP.

6.4.1 New File

New File clears the program of all history and drawn shapes. This wipes all memory of previous shapes and is irreversible. New File will also disable Undo History if it is already active. The following screenshots demonstrate what pressing New File does.

Figure 2: New Fill - Before Command

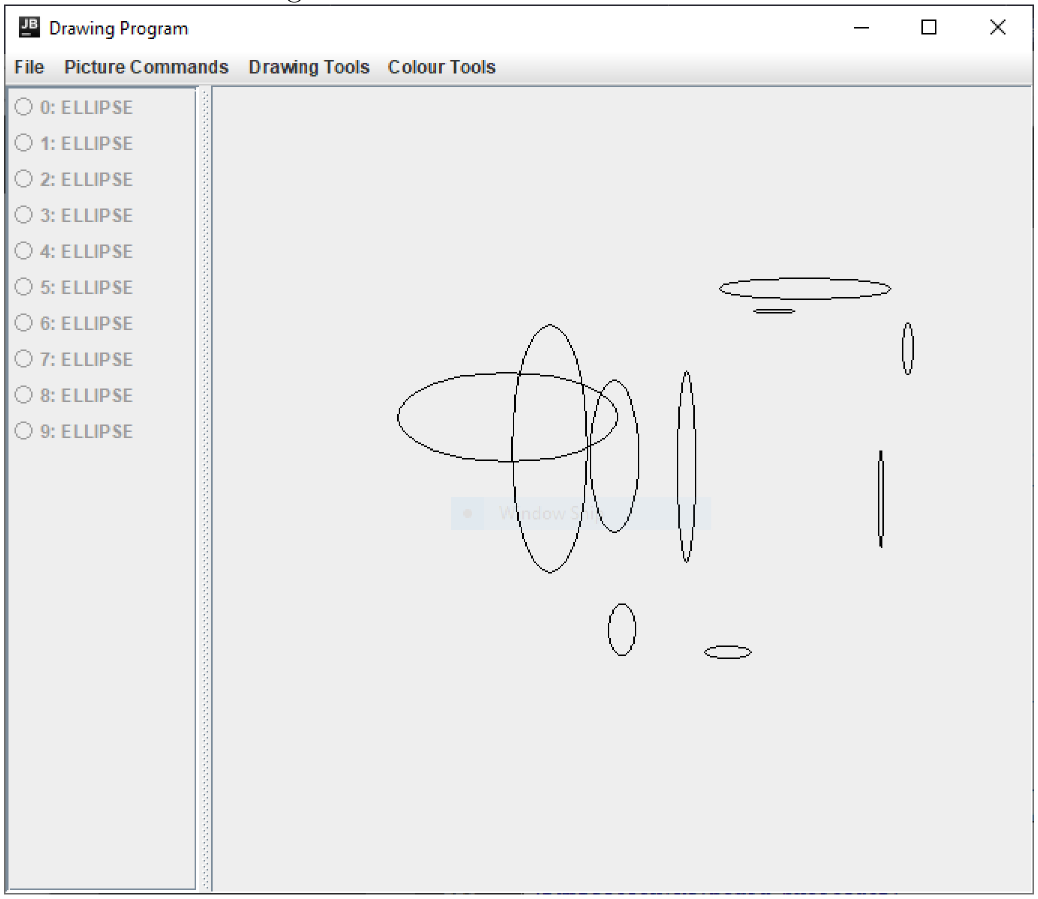
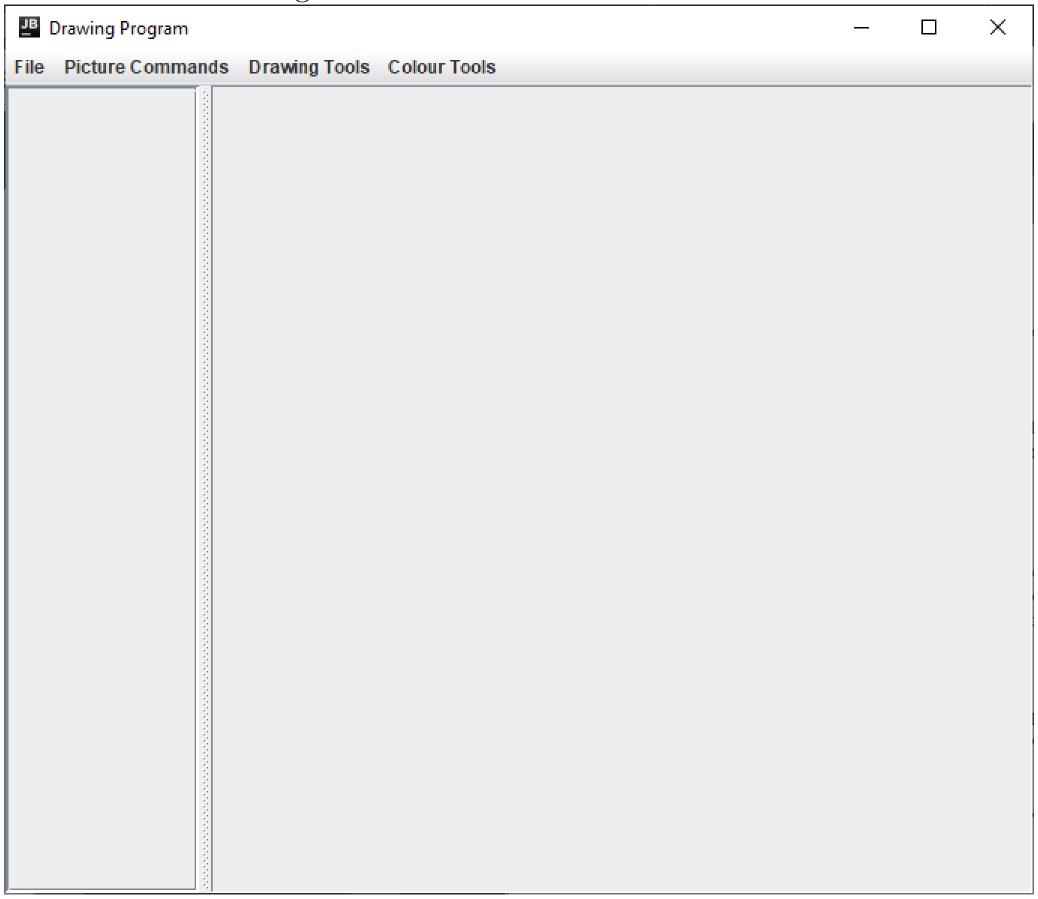


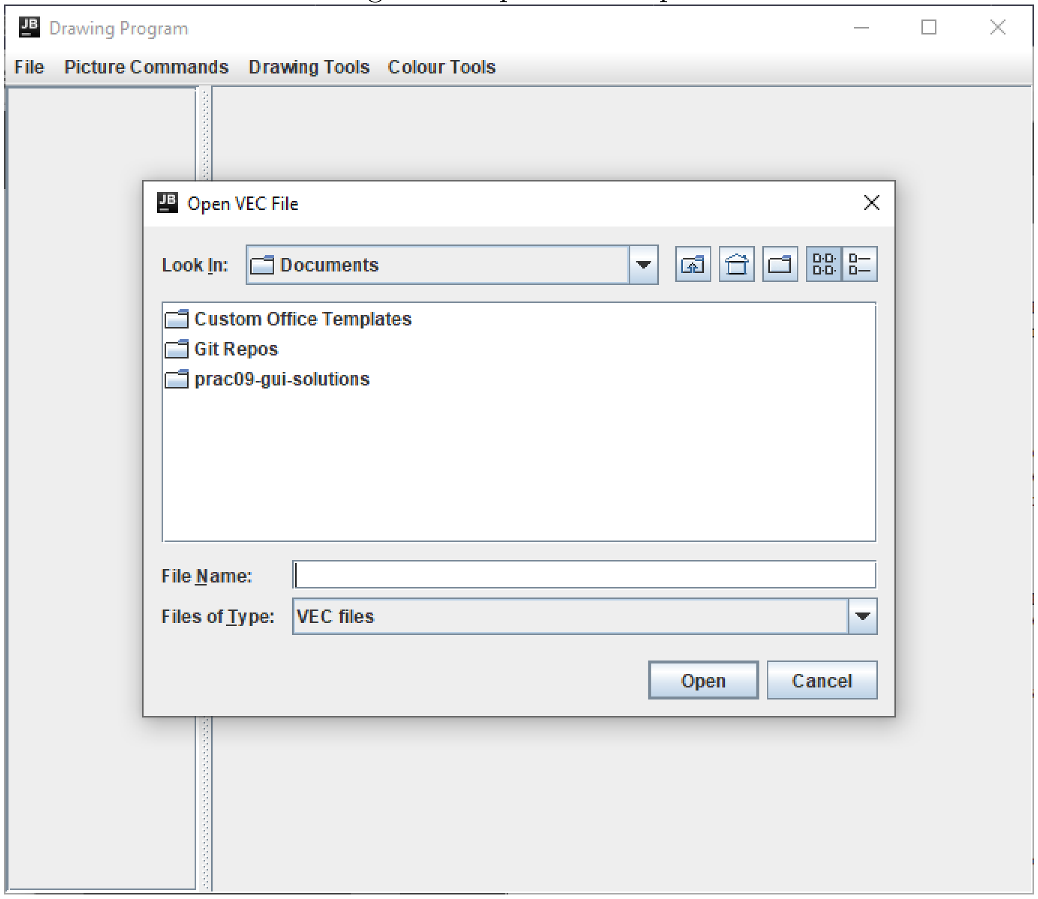
Figure 3: New Fill - After Command



6.4.2 Open

Open allows you to load a VEC file. Loading a VEC file clears the current display of any working progress, and replaces it with that of selected VEC file. Searched file types are filtered by their extension, with the only files shown being those that have the .VEC extension. As Open sets the program to an entirely new image, Undo History is disabled upon the loading of the file.

Figure 4: Open - Example

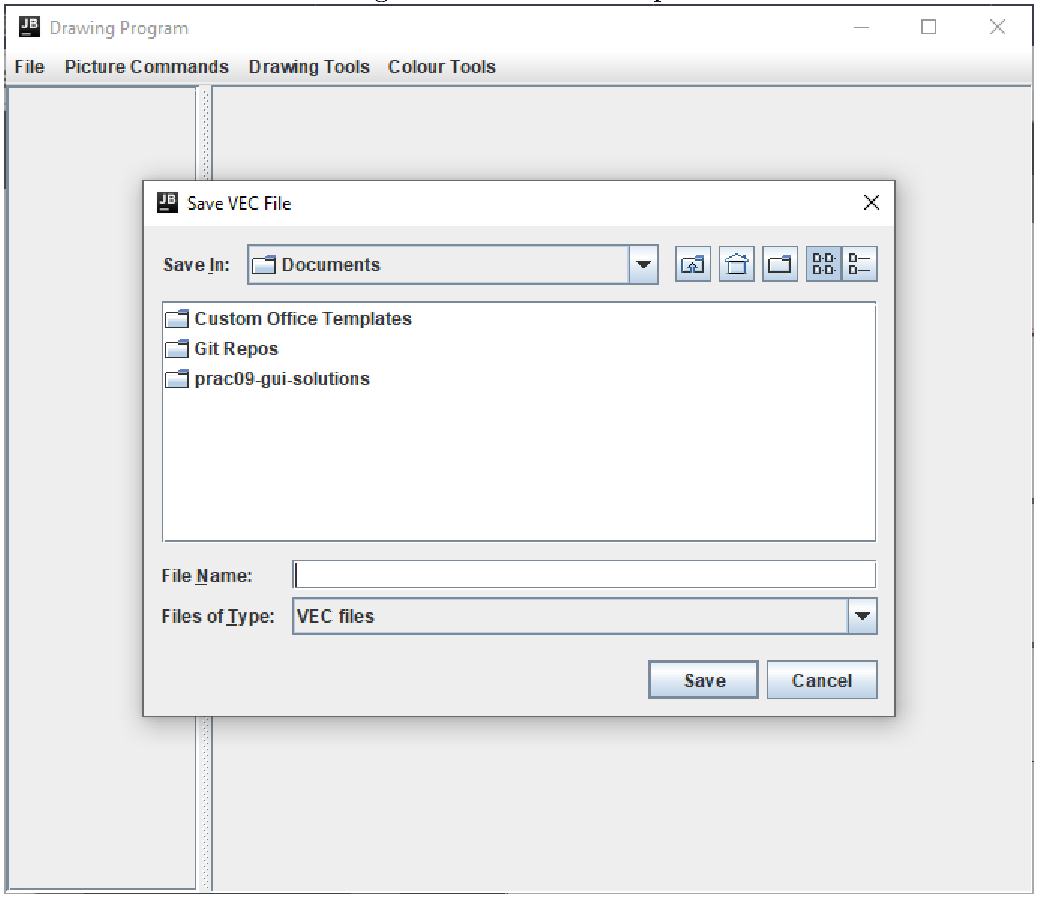


6.4.3 Save

Save is the opposite of Open, in that it allows for the current drawing to be saved to a VEC file. The displayed dialogue allows for the typing of a new file name, however if an existent file is selected that file will be overwritten. Save does not clear the program of previous shapes, allowing you to continuing working without having to open the newly saved file. Like Open, files are filtered so only VEC files are shown.

Save is disabled if Undo History is active. This ensures that what is saved is always identical to that as is being displayed on the drawing panel.

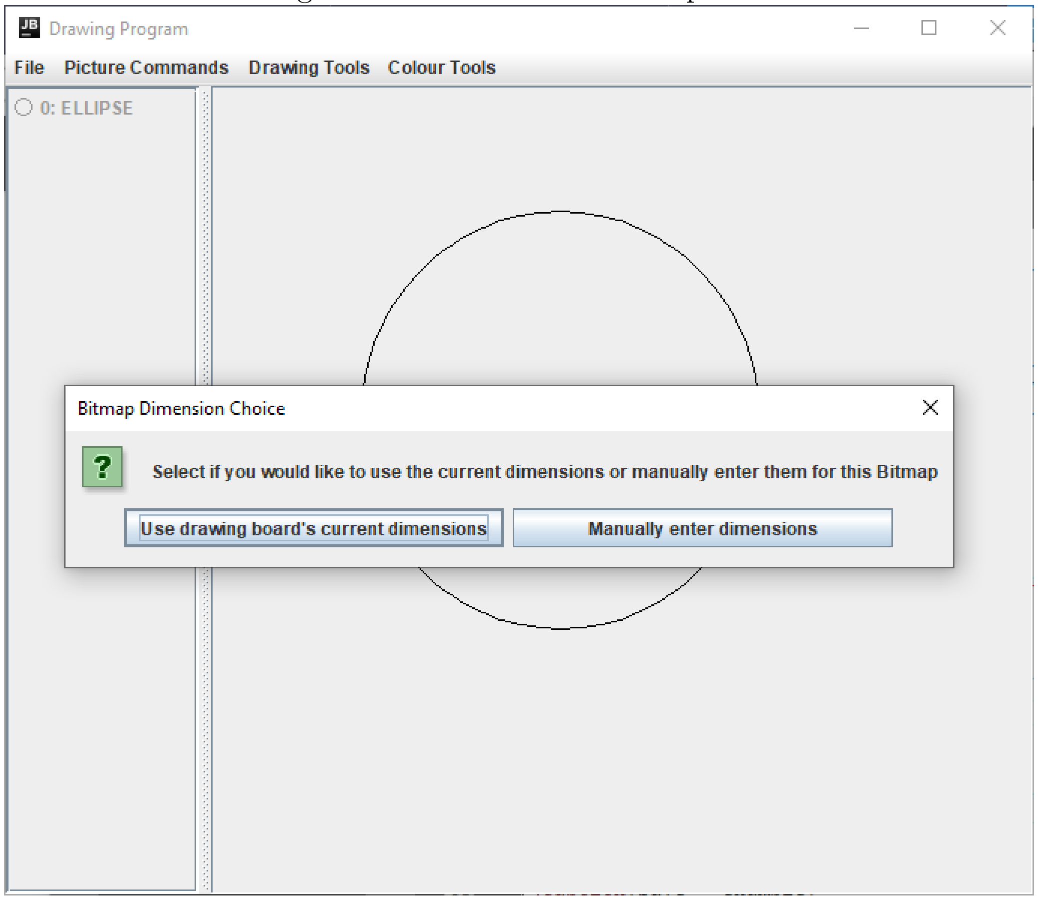
Figure 5: Save - Example



6.4.4 Export BMP

Export BMP allows for the exporting of what is drawn on the display panel to a bitmap file. Upon pressing the command a window (as shown in Figure 6) is displayed.

Figure 6: BMP - Dimension Options

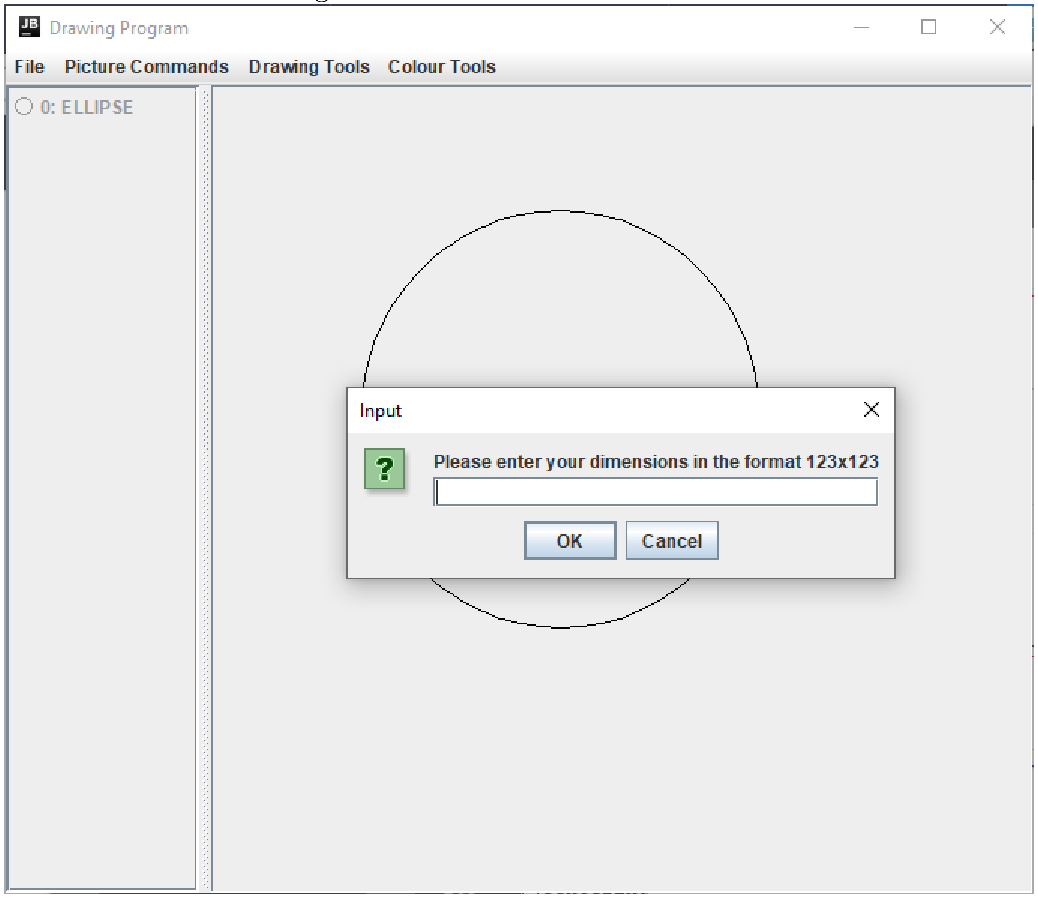


Select if you would like to use the current dimensions or manually enter them for this Bitmap

The option on the left (Use drawing board's current dimensions) which is highlighted by default will use the drawing panel's dimensions (in pixels) for the Bitmap. These are the dimensions of the drawing board at the time the Export BMP functionality was called.

The option on the right allows you to enter the width and height in pixels. If the right hand option (Manually enter dimensions) is pressed, a display such as below is shown.

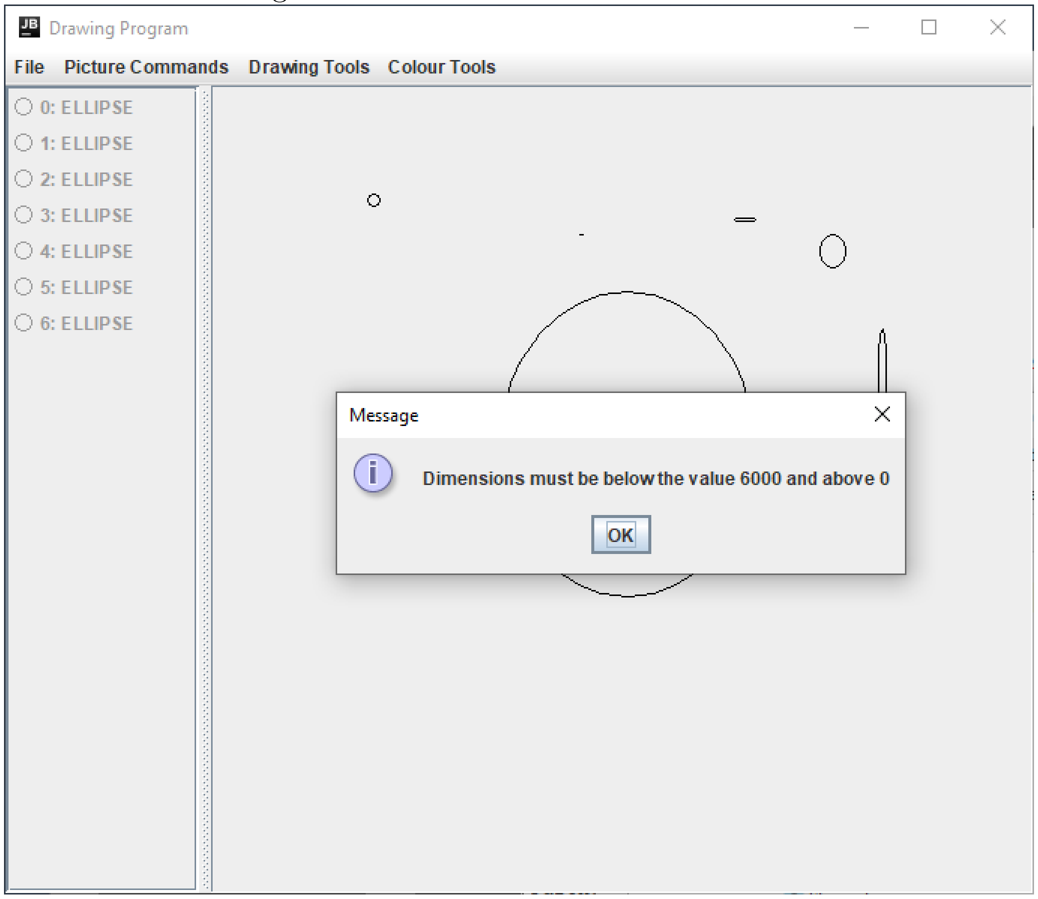
Figure 7: BMP - User Dimensions



Please enter your dimensions in the format 123x123

The input asks for the dimensions in pixels (simply whole numbers) with width before height. These dimensions must be greater than 0 and less than 6000. Furthermore, the dimensions must be in the pattern of (integers)x(integers). Examples of valid data include 127x127, 1027x720, and 4096x2086. Anything that does not match this (such as 127.7x120 or 12ax129) will result in an error message which will detail the error and return to Figure 7. An example message is shown below.

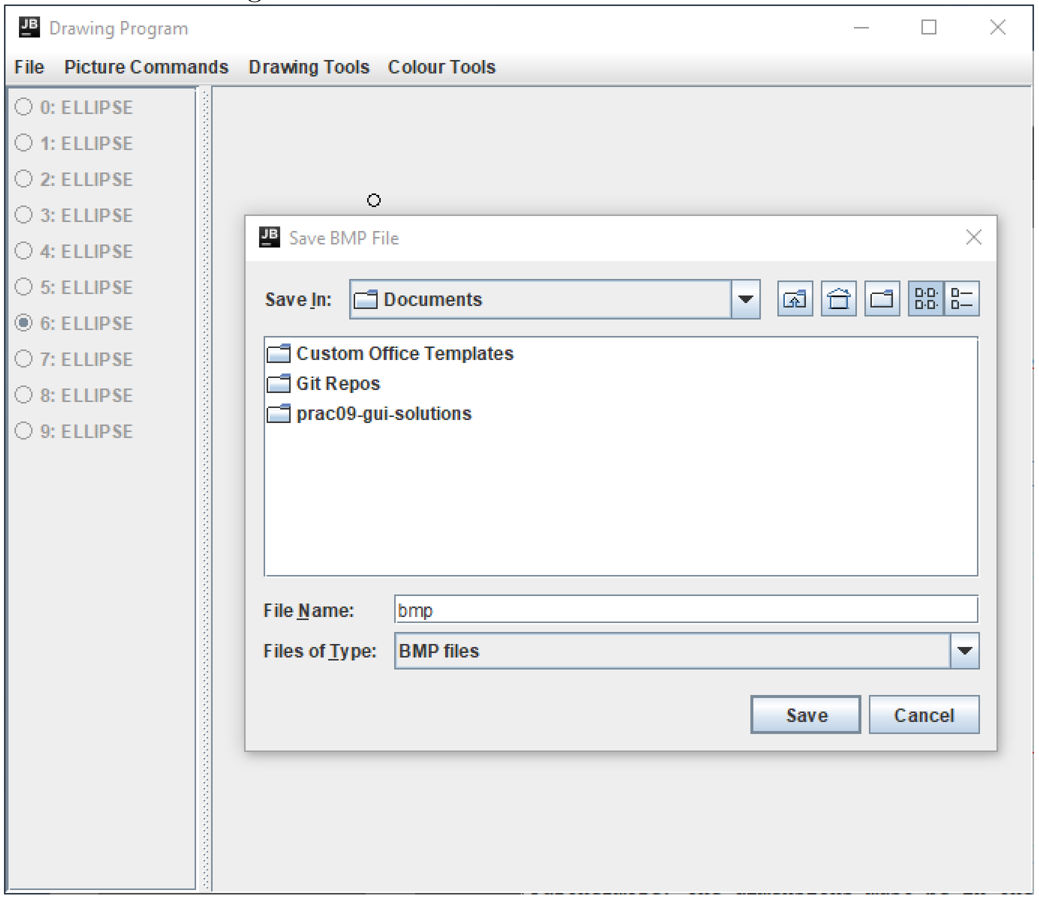
Figure 8: BMP - User Dimensions Error



Dimensions must be below the value 6000 and above 0

If the dimensions are correct, or if the left option (Use drawing board's current dimensions) was selected, then a file saver dialogue will show like below.

Figure 9: BMP - User Dimensions Save File



Akin to saving VEC files, the dialogue filters for only BMP files. A new file name can be typed, or an existing BMP file can be selected, overwriting it. If you do not add the .bmp extension when typing a new file name, it will be added when the file is written.

BMP files cannot be displayed by the VEC drawing software, and must be loaded into separate programs (such as Windows 10 Photos) to be displayed.

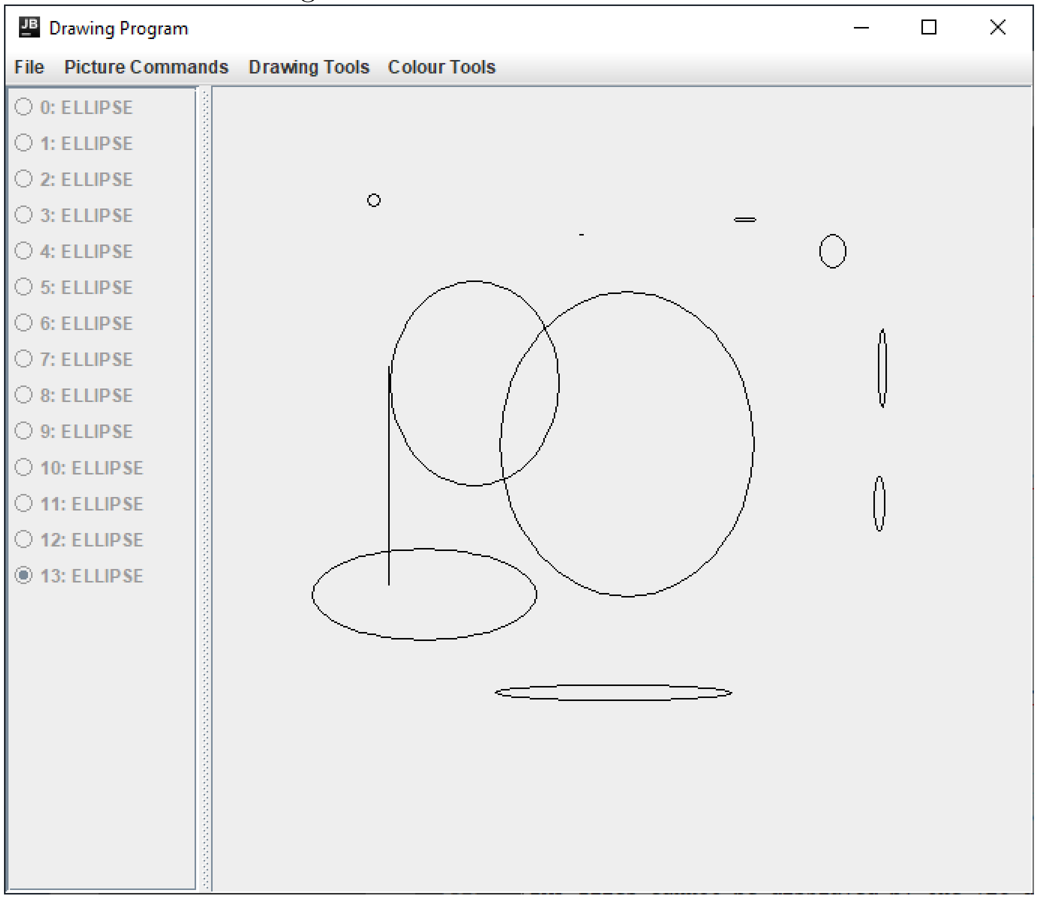
6.5 Picture Commands

Picture Commands handles the navigating of the states of the drawing, or in other words handles how you can view previous iterations of current project. The individual commands are Undo, Show Undo History, and Confirm Selected History.

6.5.1 Undo

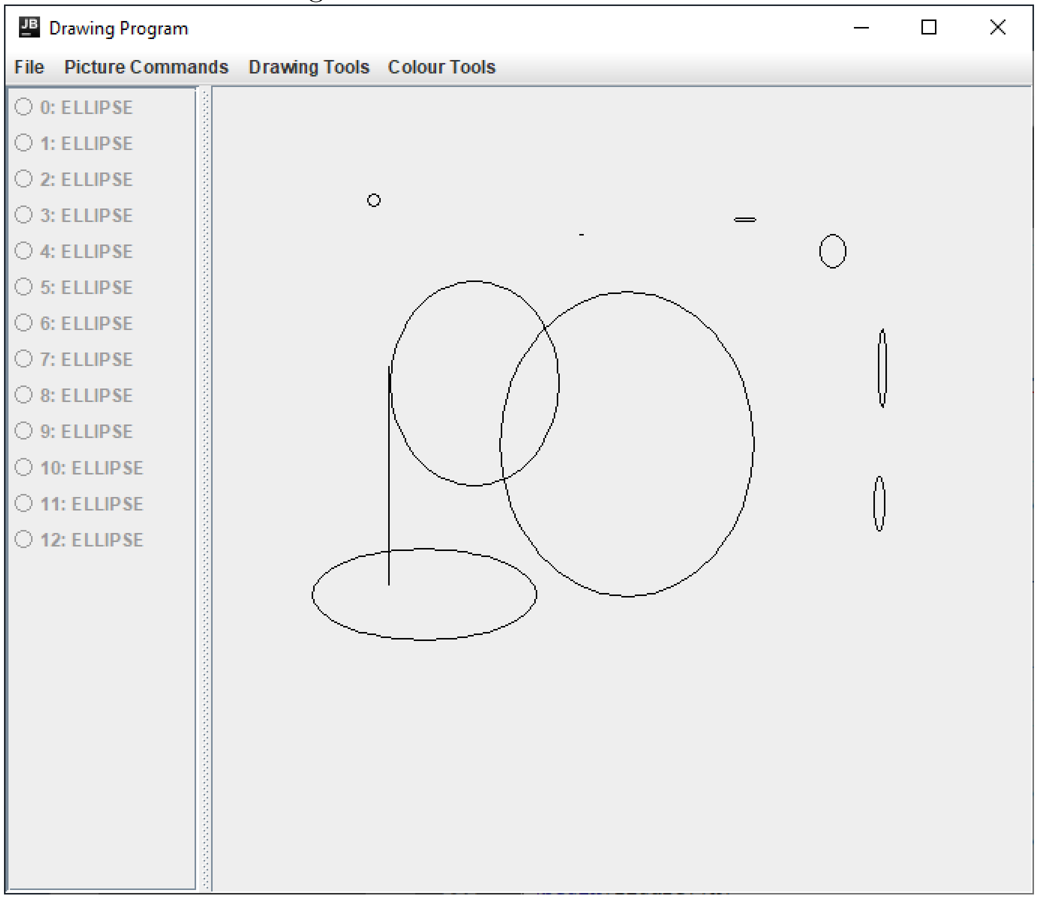
Undo is analgous to how it operates in programs such as Microsoft Word. Pressing it erases the last shape drawn, both from the drawing panel and the history side bar. An example of this with pictures is below.

Figure 10: Undo - Before Command



- ○ 0: ELLIPSE
- ○ 1: ELLIPSE
- ○ 2: ELLIPSE
- ○ 3: ELLIPSE
- ○ 4: ELLIPSE
- ○ 5: ELLIPSE
- ○ 6: ELLIPSE
- ○ 7: ELLIPSE
- ○ 8: ELLIPSE
- ○ 9: ELLIPSE
- ○ 10: ELLIPSE
- ○ 11: ELLIPSE
- ○ 12: ELLIPSE
- ● 13: ELLIPSE

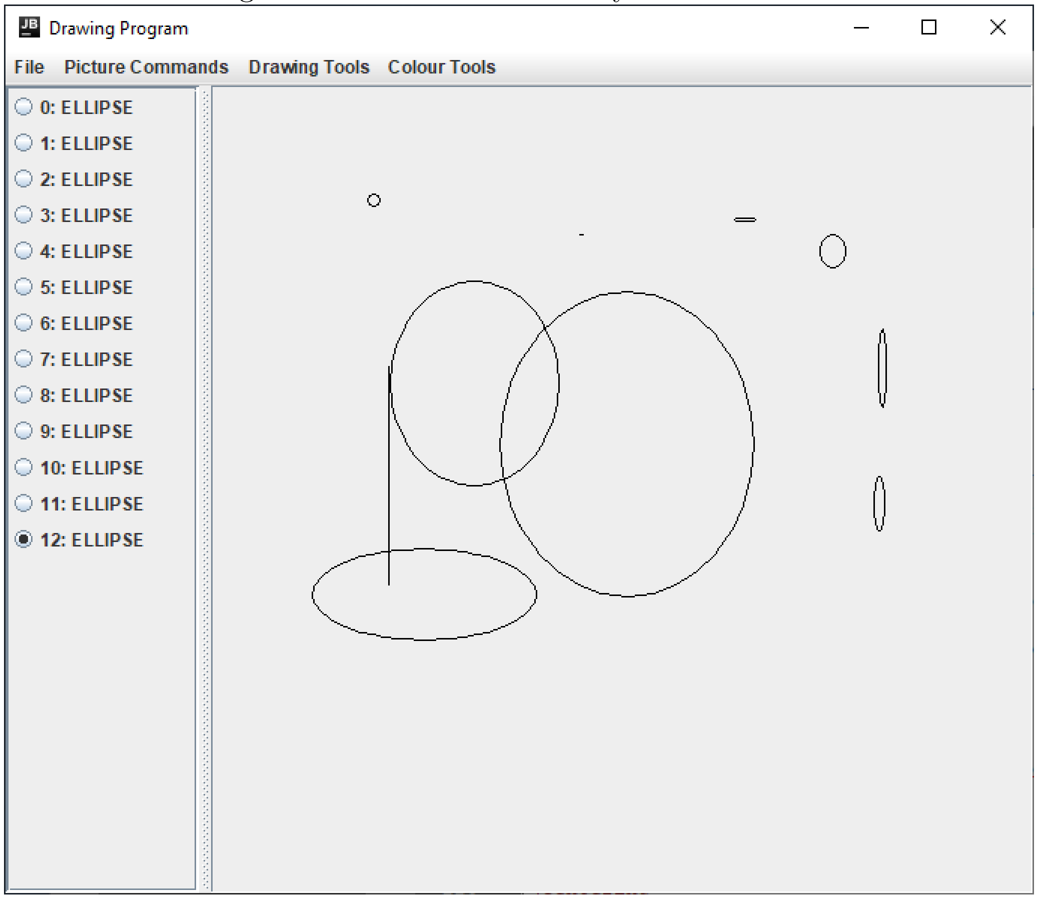
Figure 11: Undo - After Command



6.5.2 Show Undo History

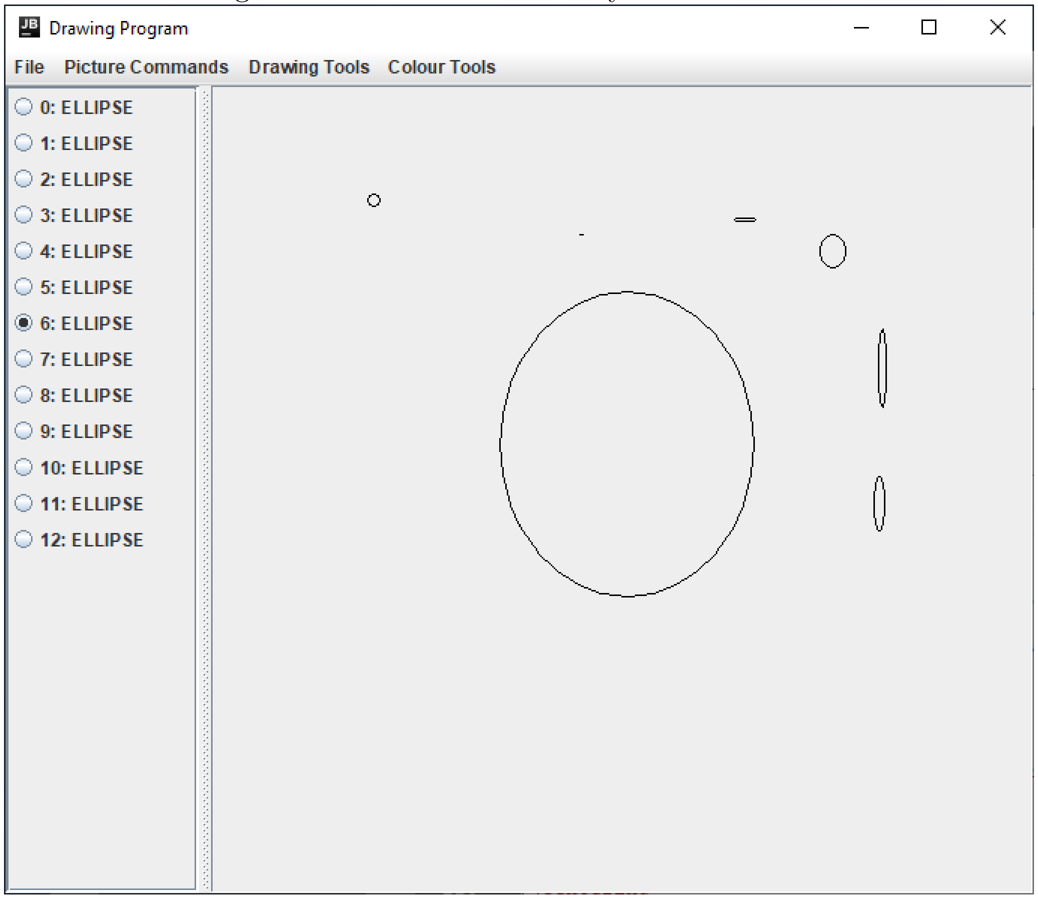
Show Undo History allows you to see previous states of the picture, analogous to Adobe's Photoshop. Pressing the button will activate this mode, preventing any further drawing on the canvas but allowing you step back to earlier iterations. Clicking on a radio button will show the shape that it corresponds to, as well as every shape prior to it, and none of the shapes after it. To demonstrate:

Figure 12: Show Undo History - Latest State



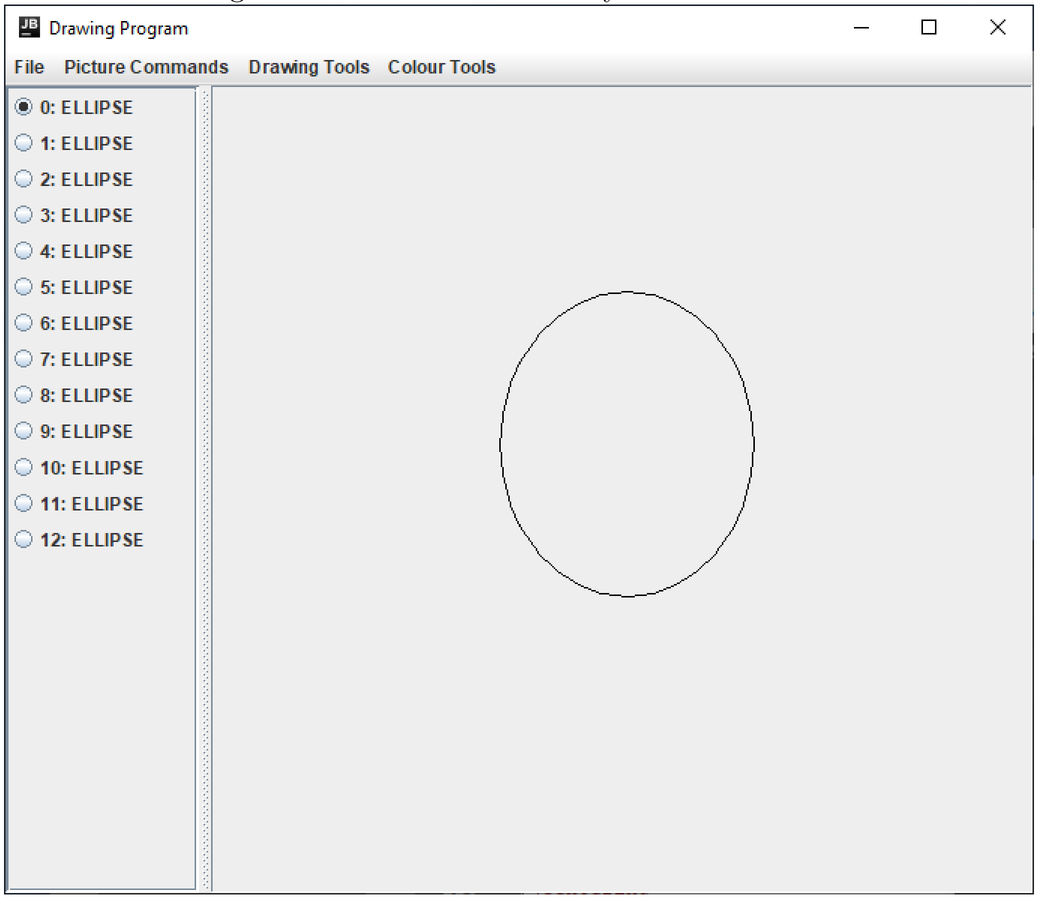
0: ELLIPSE
1: ELLIPSE
2: ELLIPSE
3: ELLIPSE
4: ELLIPSE
5: ELLIPSE
6: ELLIPSE
7: ELLIPSE
8: ELLIPSE
9: ELLIPSE
10: ELLIPSE
11: ELLIPSE
[✓ 12: ELLIPSE]

Figure 13: Show Undo History - Middle State



( ) 0: ELLIPSE
( ) 1: ELLIPSE
( ) 2: ELLIPSE
( ) 3: ELLIPSE
( ) 4: ELLIPSE
( ) 5: ELLIPSE
[✓ 6: ELLIPSE]
( ) 7: ELLIPSE
( ) 8: ELLIPSE
( ) 9: ELLIPSE
( ) 10: ELLIPSE
( ) 11: ELLIPSE
( ) 12: ELLIPSE

Figure 14: Show Undo History - Earliest State

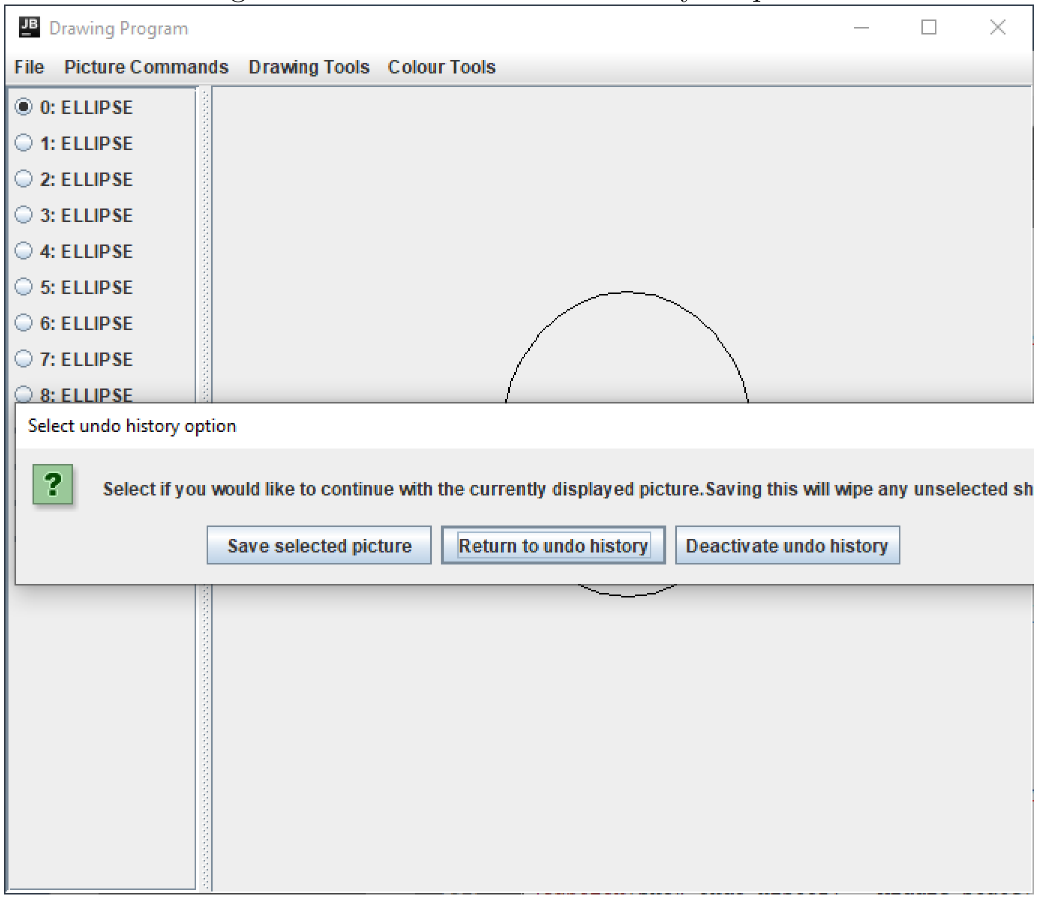


6.5.3 Confirm Selected History

Confirm Selected History, if Undo History has been activated by pressing Show Undo History, shows options in regards to saving, disregarding or continuing with the selected state. If Undo History is not active then a warning appears mentioning this and prevents further actions.

The three options of Confirm Selected History are: Save Selected Picture, Return to Undo History, and Deactivate Undo History. These are shown below.

Figure 15: Confirm Selected History - Options



Select if you would like to continue with the currently displayed picture.Saving this will wipe any unselected sh

Pressing Return to Undo History simply allows you to continue with the Undo History mode.

Pressing Save Selected Picture will re-enable the drawing of new shapes, as well as erasing all states (shapes and associated radio buttons)that are after (or below) the selected shape. This is a permanent action and cannot be undone unless the file was saved with the Save command under File.

Pressing Deactivate Undo History allows for the drawing of more shapes, and sets the drawing canvas to the latest state.

6.6 Drawing Tools

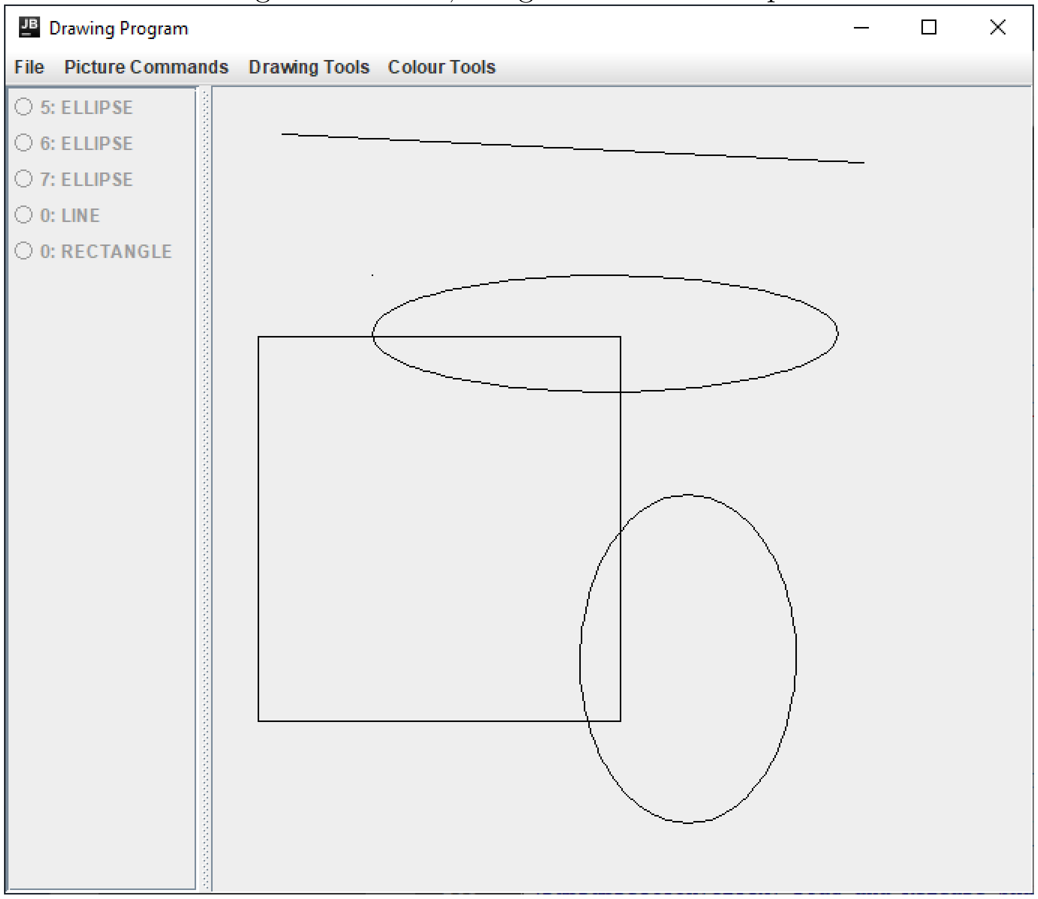
The purpose of this program is drawing, and so these tools are especially important as they allow for a selection of drawing shapes to be chosen. The list of commands is: Plot, Line, Rectangle, Ellipse, Polygon, and Clear Polygon. These commands are divided into sections based upon how they are drawn with the mouse.

6.6.1 Click, Drag and Release Shapes

Line, Rectangle, and Ellipse are all drawn by clicking, dragging and then releasing. After selecting the relevant shape from the dropdown menu, clicking on a point within the drawing canvas will create a new shape of that type (be it line, rectangle or ellipse). Keeping the left mouse button depressed after clicking this spot, dragging the shape will set its orientation, length and if not a line, its size. Releasing the left mouse button will then confirm this shape, and it will be fixed onto the drawing display unless removed with one of the Undo options.

An example of these shapes is below:

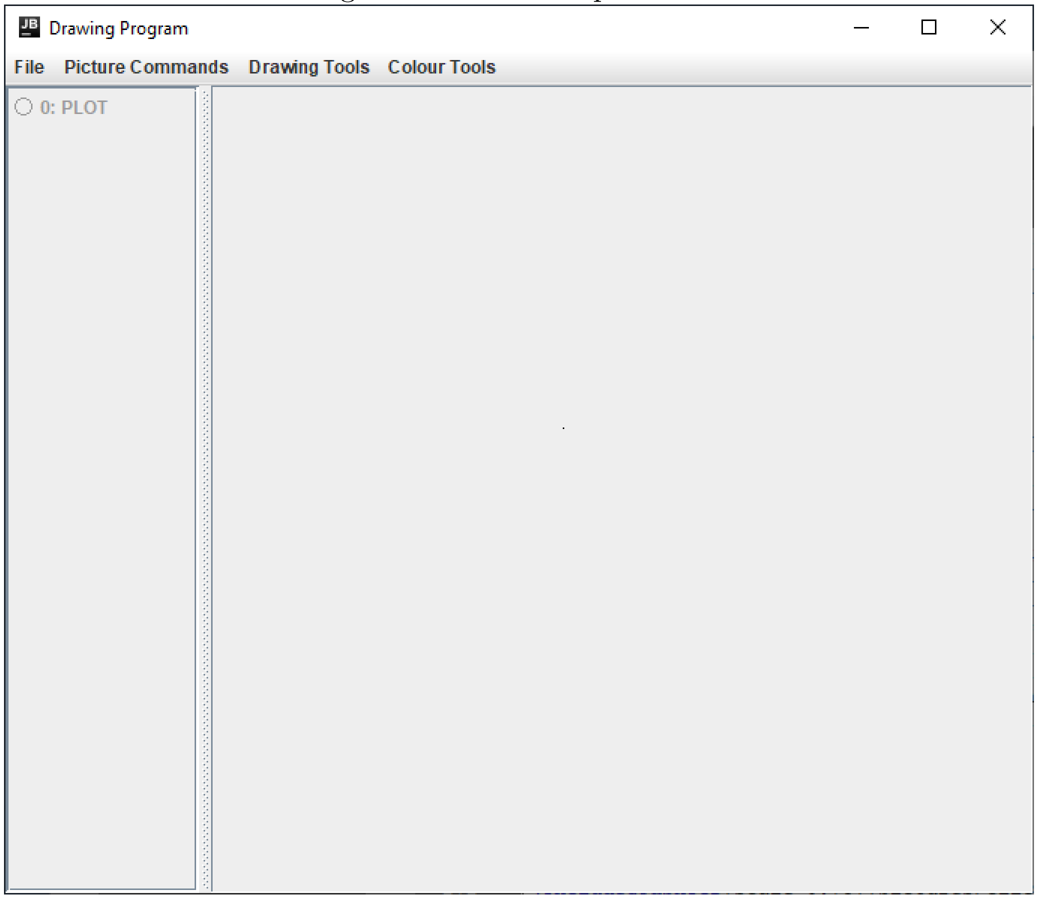
Figure 16: Click, Drag and Release Shapes



6.6.2 Click Shapes

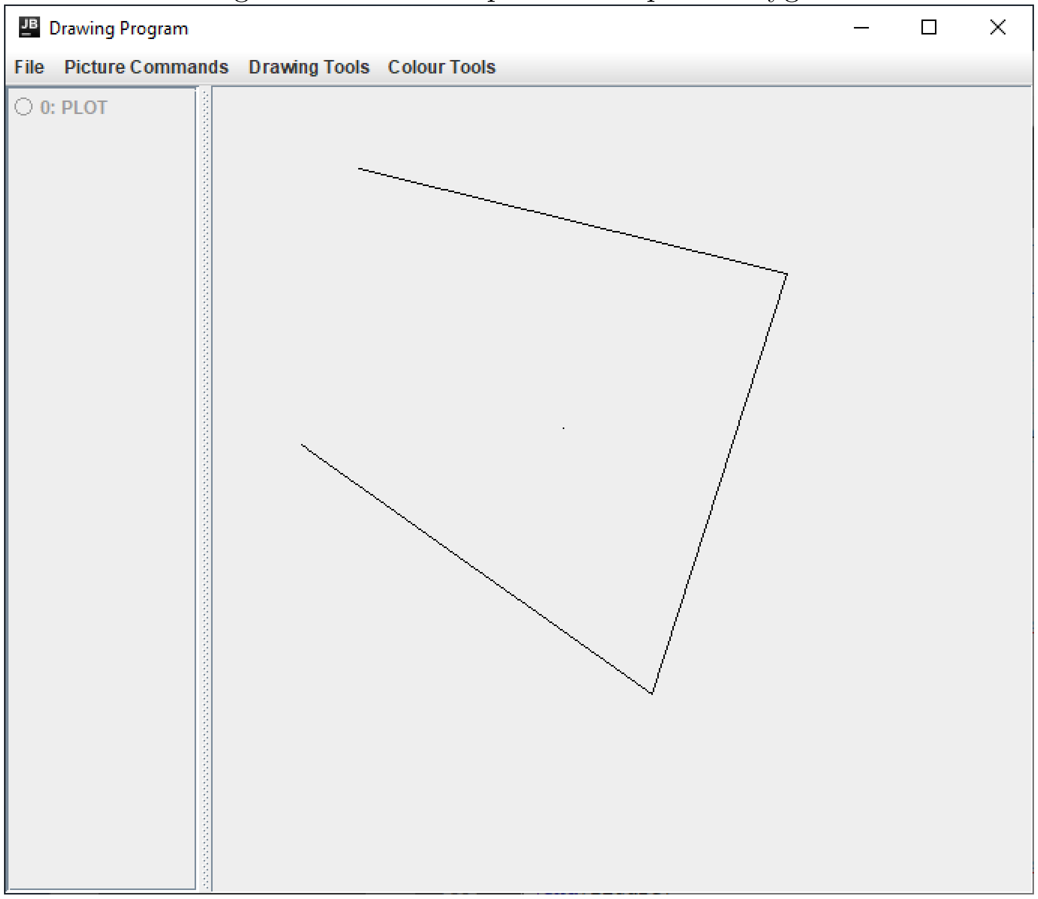
Plot and Polygon are click shapes. That is, the only mouse interaction used in creating the shape is a click. Plot is a simply dot at the spot of the mouse click.

Figure 17: Click Shapes - Plot



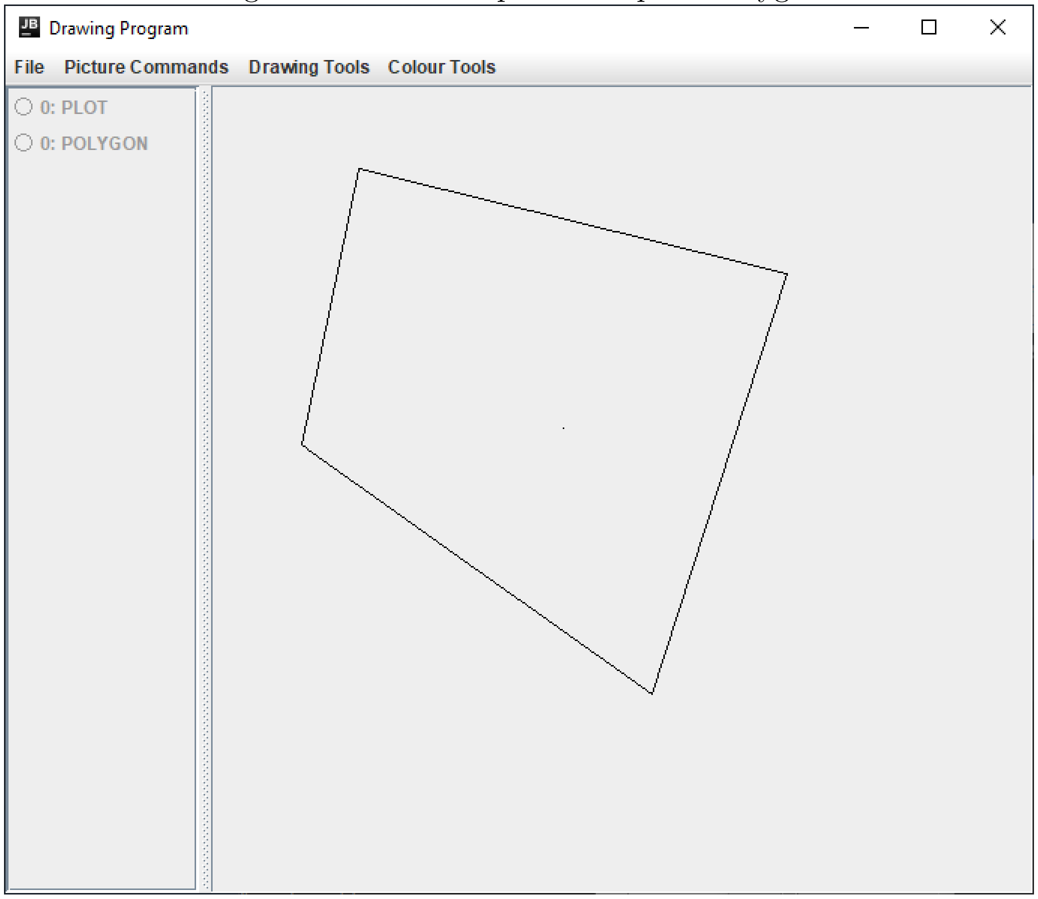
Polygon is more complex, in that whilst the polygon tool is selected, the latest mouse click will create a point that draws a line to the last point that was clicked (assuming at least two points are clicked). An example of an incomplete polygon is shown:

Figure 18: Click Shapes - Incomplete Polygon



To complete a polygon you have to connect back to the starting point. In the above example the left-most point of the highest line is the starting point. To complete the polygon, clicking once more on this point is required. The polygon will not be registered in the history side bar until it is completed.

Figure 19: Click Shapes - Complete Polygon

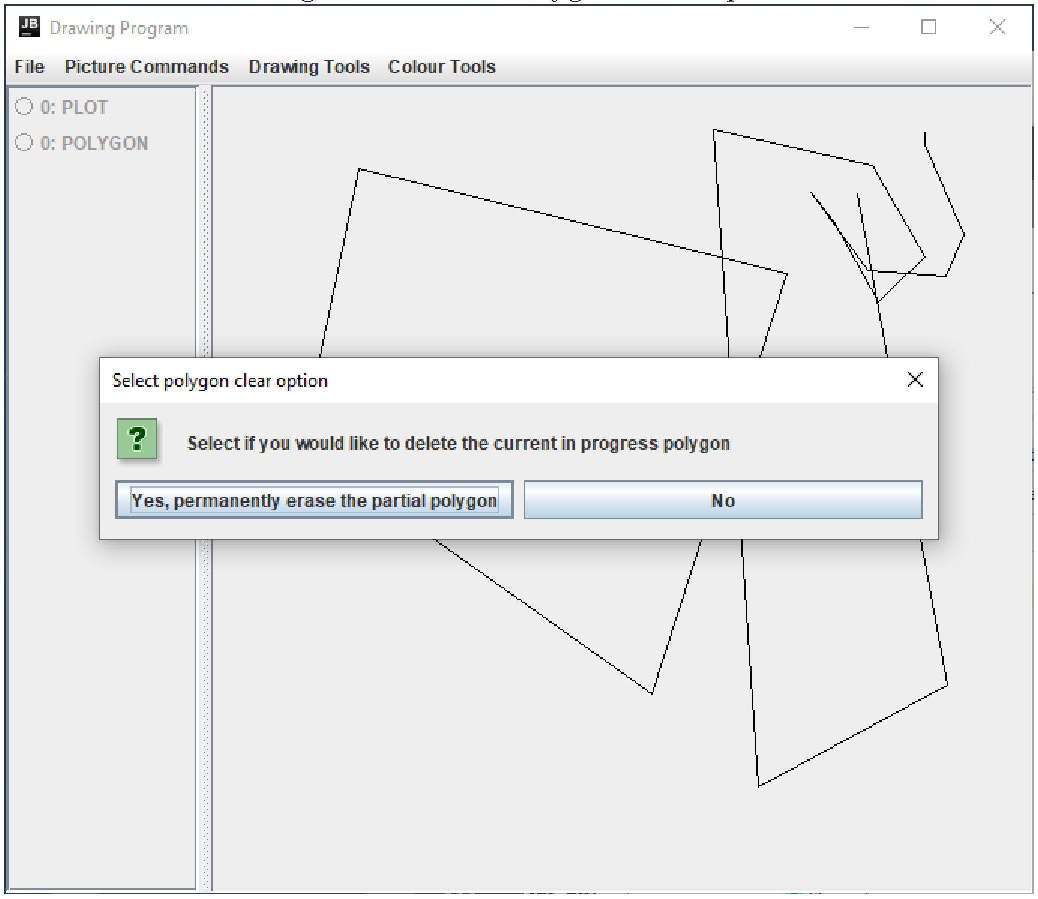


If one switches to another drawing tool whilst there is an incomplete polygon on the screen, you can draw with the newly selected shape, and then reselect the polygon tool to finish the polygon.

6.6.3 Clear Polygon

An incomplete polygon is not registered as a shape, and so as such cannot be removed with the Undo command, saved or affected by Undo History. To remove an incomplete polygon, press Clear Polygon. This will permanently wipe the entire polygon.

Figure 20: Clear Polygon - Example



Select if you would like to delete the current in progress polygon

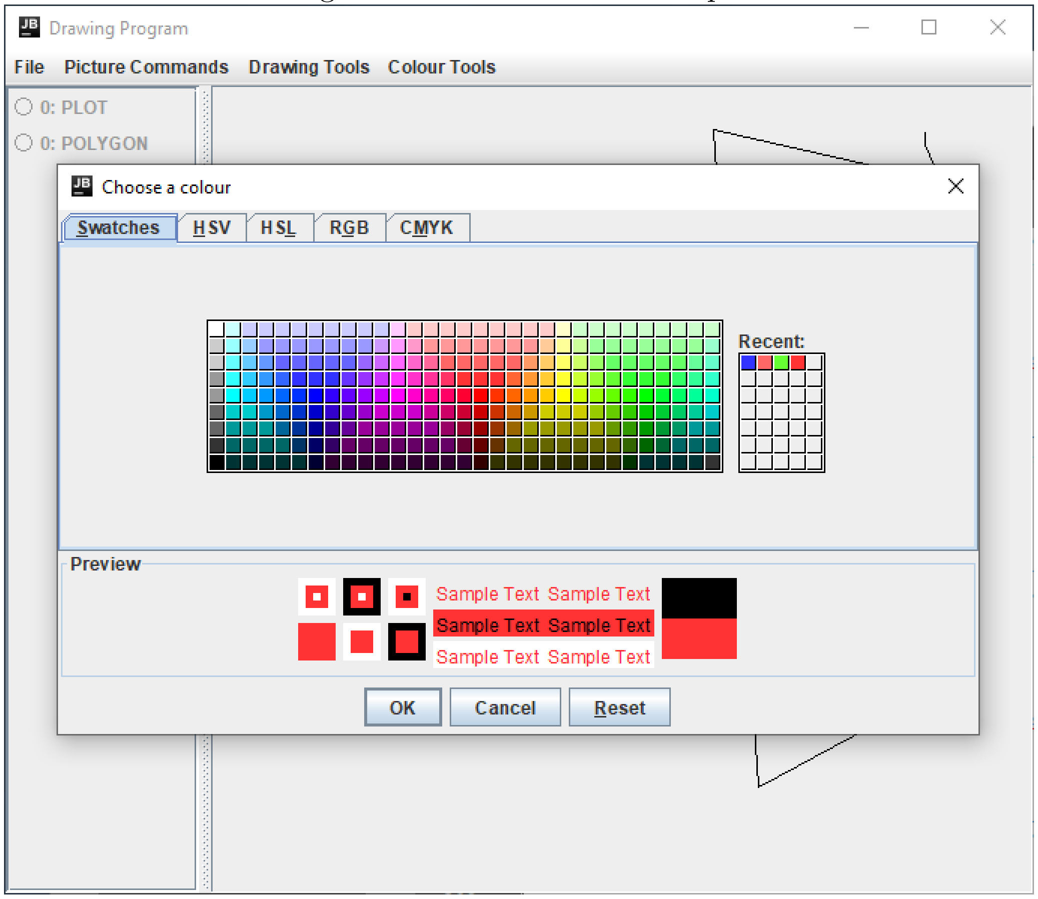
6.7 Colour Tools

Colour Tools allows for changing the colour of both the pen colour (or outline) and the fill colour (which is optional). The options under Colour Tools are Fill Colour, Pen Colour, and Toggle Fill.

6.7.1 Pen Colour

Used on every shape, the pen colour sets the colour of the shape's boundary. Upon pressing Pen Colour, a dialogue is shown allowing for one to choose the colour they desire.

Figure 21: Pen Colour - Example

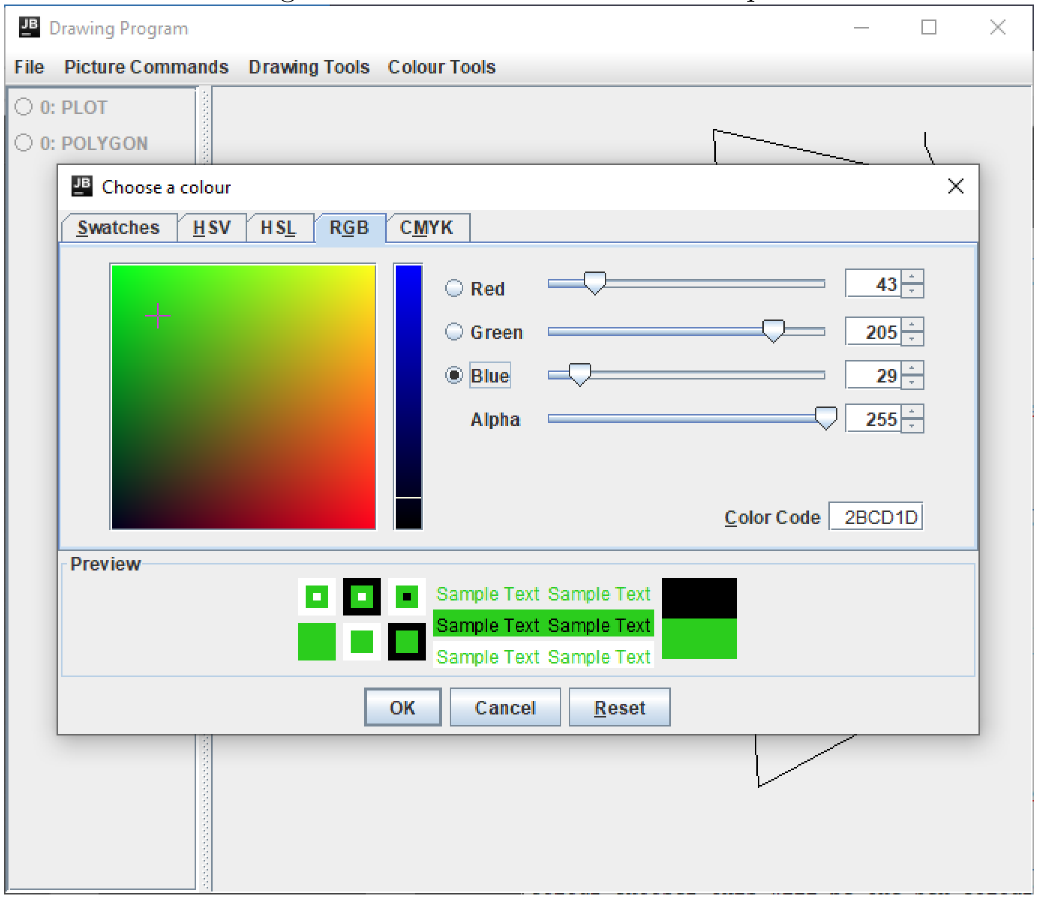


The currently selected pen colour is shown in the example test. Upon loading the colour chooser this will be the pen colour that is being used for drawing on the canvas, but after selecting a different colour the sample text will change to that colour. Hitting Reset on the dialogue will set the colour back to what it initially was while leaving the dialogue open, whereas Cancel will exit the dialogue whilst keeping the pen colour unchanged. Hitting OK will set the pen colour to the currently selected colour in the chooser.

The recent swatches stores previously clicked on colours (not just those selected), across both the Pen Colour and Fill Colour dialogues. This allows for quick access to colour that are of interest.

Finally, the other tabs (such as HSV) allow for other ways of creating a colour, as opposed to relying on the swatches.

Figure 22: Pen Colour - RGB Example



Red 43
Green 205
Blue 29
Alpha 255
Color Code 2BCD1D

6.7.2 Fill Colour

Fill Colour operates in the same way as Pen Colour. The only difference worth noting is that selecting a fill with OK will not result in the colour being applied as a fill if filling has been disabled with Toggle Fill.

6.7.3 Toggle Fill

This simply toggles whether or not shapes are to be filled when drawn or not. By default they are not, so clicking this for the first time will activate colour filling, and then hitting it again will disable filling, and so forth.

Figure 23: Toggle Fill - No Fill

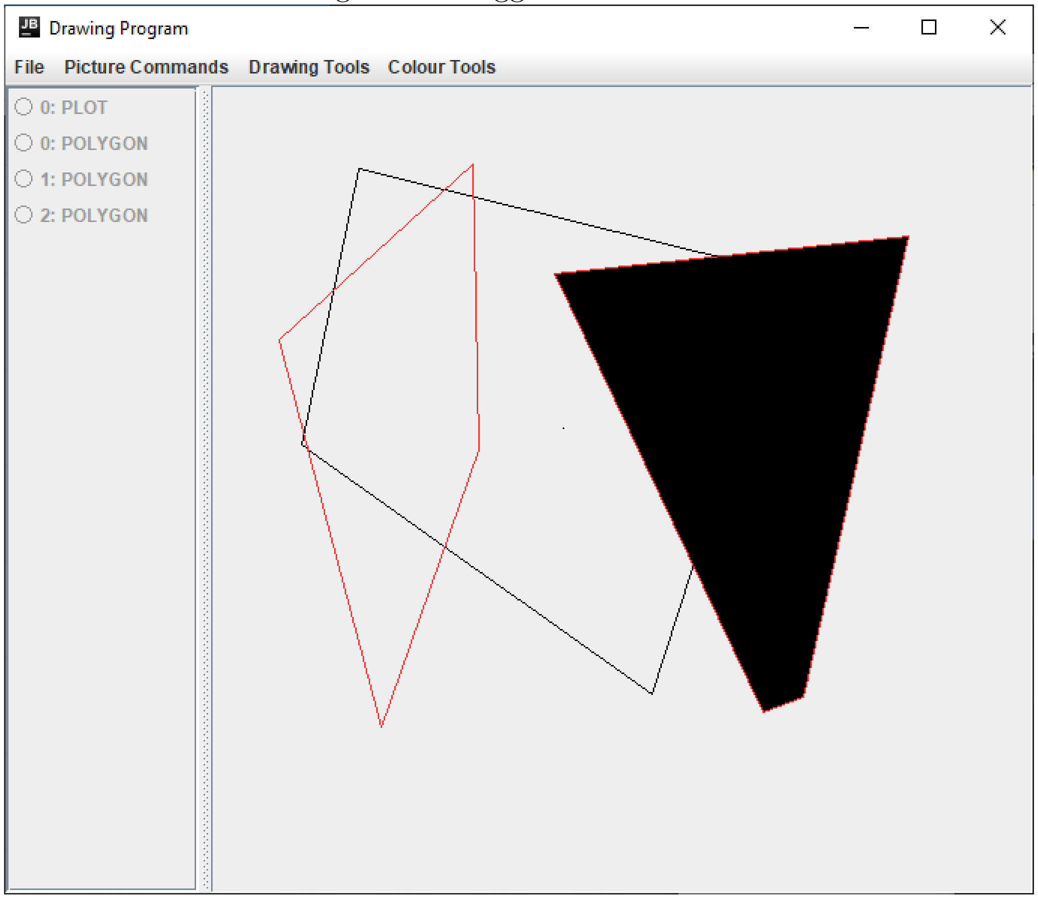
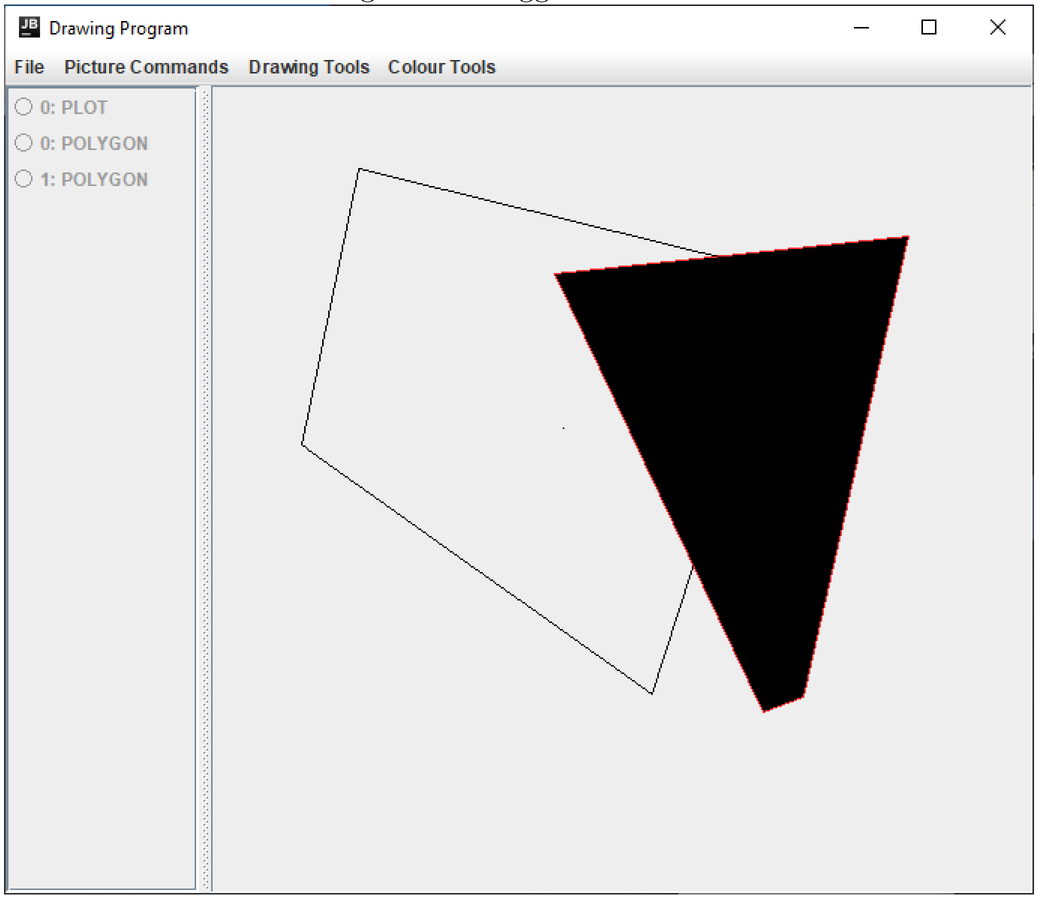


Figure 24: Toggle Fill - Fill


Drawing Program
File
Picture Commands
Drawing Tools
Colour Tools
0: PLOT
0: POLYGON
1: POLYGON