Assignment 2: Vector Design Tool
CAB302 Semester 1, 2019

Bennett Hardwick
n9803572

June 2, 2019

Contents

1 Statement of Completeness

The assignment was completed in full, with all functionality being implemented. File interactions (loading and saving to a file) were implemented in full. Actions RECTANGLE, FILL, PEN, POLYGON, ELLIPSE, PLOT and LINE were all implemented. Interactions with the canvas - drawing shapes, setting colours - were also implemented. Finally, the ability to undo, as well as binding the action to CTRL-Z, were implemented.

2 Statement of Contribution

As I was the only member of my group, I completed the entire assignment. This includes creating the app, testing as well as writing the report.

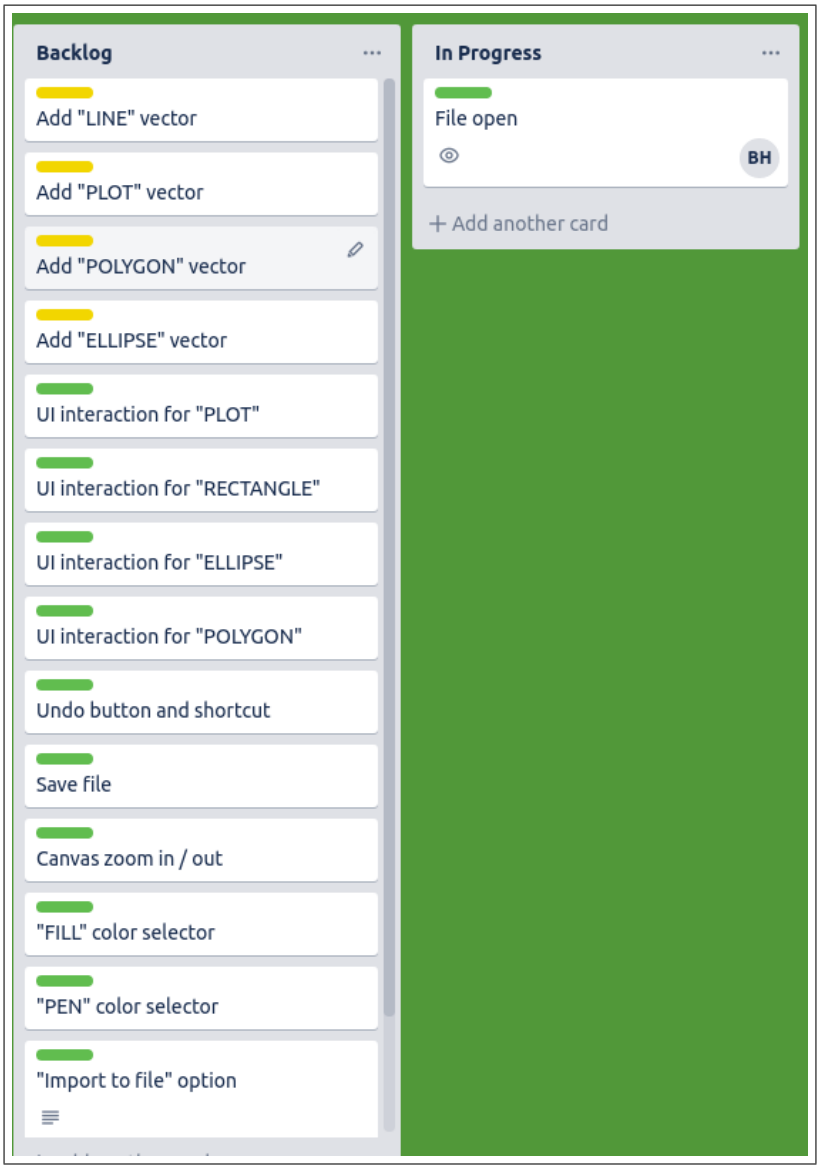
3 Software Development Process

In order to have a working application at all times, Agile Development was followed. Unlike typical Agile sprints however, which last one to four weeks, the project was done as one large sprint spanning the entire time. At the beginning of the project, the task was broken down into two main epics. These epics were the renderer and the GUI. The renderer epic was then broken down into smaller user stories, depicting it's entire functionality. Likewise, the GUI epic was also broken down into user stories outlining how the user would interact with the application.

With all the tasks planned out and added to the backlog, they were prioritised and selected for development. Unlike a normal project, where you would only choose enough stories to fit your capacity, all stories were selected as the project would be completed in one sprint. It was important during the project to have the application in a working state at all times, iterating on that as the project progressed.

This is visible in the way stories were prioritised and the order in which they were completed. For example, as the key function of the application was to draw images based on "VEC" files, the renderer was the first component to be developed. After this, was the canvas - Which could show the shapes depicted by the actions. Next the application was iterated upon by adding the different types of actions. Finally, the ability for the user to interact with the canvas and draw shapes was added.

By using this Agile framework, the application was in a usable state, that was better than the last after every iteration.



4 Software Architecture

The application was created as two separate layers - the presentation-layer and the data-layer. Namely, the tool (GUI) and the renderer.

The “App” class inside the “tool” package is the entry to the application. Instantiating this class will create a self-contained, fully featured “Vector Tool” window. The “App” class is a JFrame which contains; the “MenuBar” class which is a menu bar, the “Toolbar” class which is the tool bar which holds all the options for what vector to draw and such, and finally the “Canvas” class which is responsible for drawing vectors to the screen.

The “Canvas” class acts an abstraction over the “Renderer” class, which is responsible for drawing the actions to the canvas. As the “Canvas” is the main interaction with the renderer, there is a message channel between the “MenuBar” and the “ToolBar”, and the “Canvas”. This message channel follows the “Subject-Observer” pattern and sends an instance of the “UpdateEvent” class from the bars to the “Canvas”.

The “Renderer” holds the state for the application in the form of a list of classes that inherit from “Action”. The “Action” interface is used in three places, the “Fill” class, the “Pen” class and in the “VectorAction” class.

The “VectorAction” class contains an instance of the abstract “Vector” class. The “Vector” class is responsible for

parsing arguments of a “VEC language” statement and converting it into pixels. The “Vector” class is extended by “Ellipse”, “Line”, “Plot”, “Polygon” and “Rectangle” which are responsible for rendering their respective actions.

5 Object Oriented Programming

5.1 Abstraction

Use of abstraction can be seen in the “Vector” abstract class. This class defines how all vectors should work and the interface through which you convert them into a Java AWT shape.

5.2 Encapsulation

An example of Encapsulation can be seen in the “Pen” class. In this class, the Color of the object is hidden. It can only be accessed through the getter method, and can only set through the constructor as an argument and through the setter method.

5.3 Inheritance

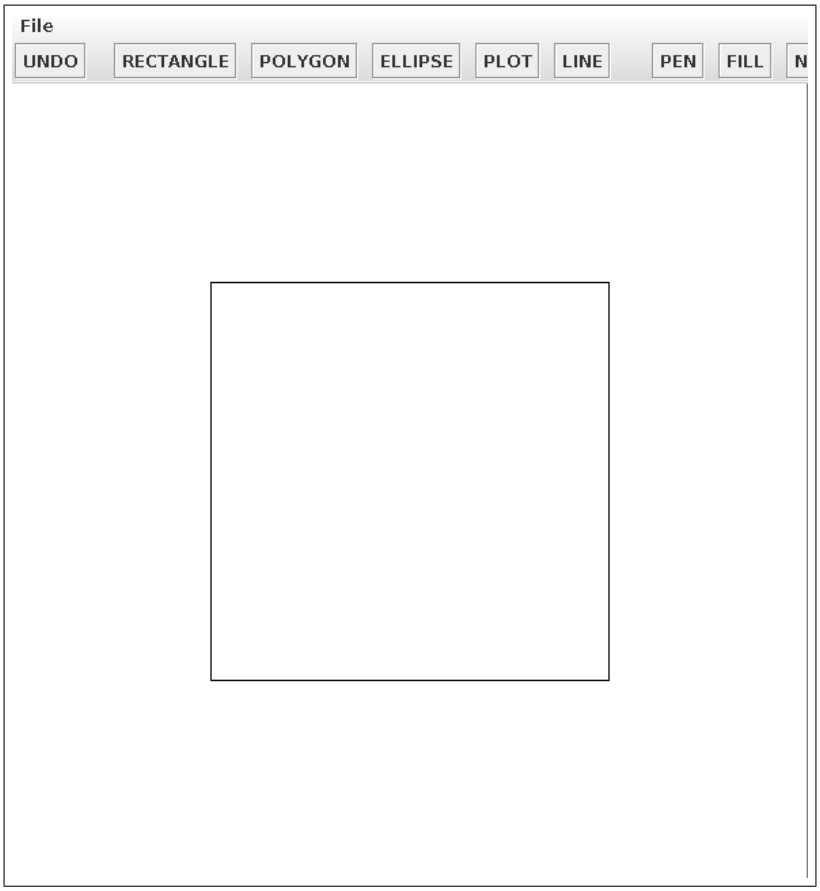
Inheritance is used in the “ToolBar” class. It inherits from the “JToolBar” class.

5.4 Polymorphism

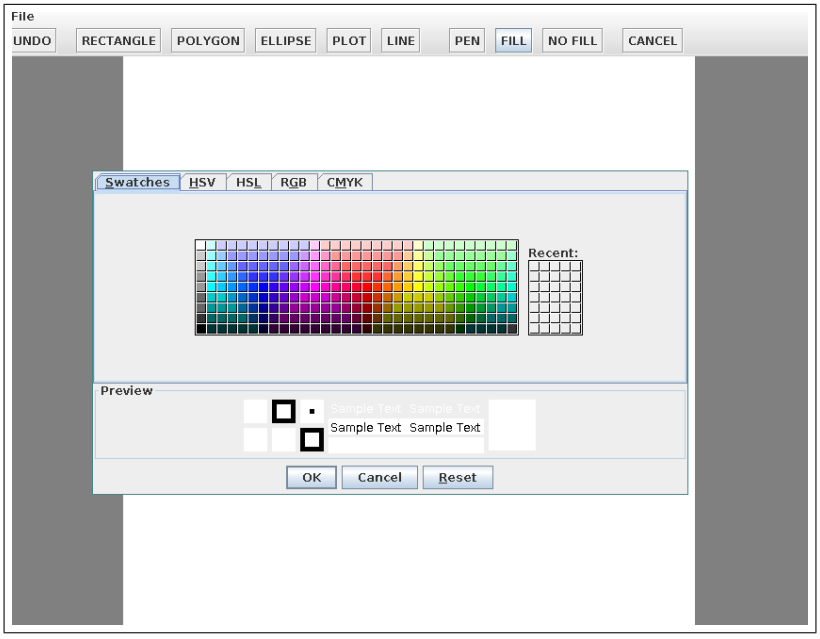
Polymorphism can be seen in the “App” class. It both inherits from a “JFrame” and implements the “KeyListener” interface. As such, it is both a “JFrame” and a “KeyListener”.

6 How to Use

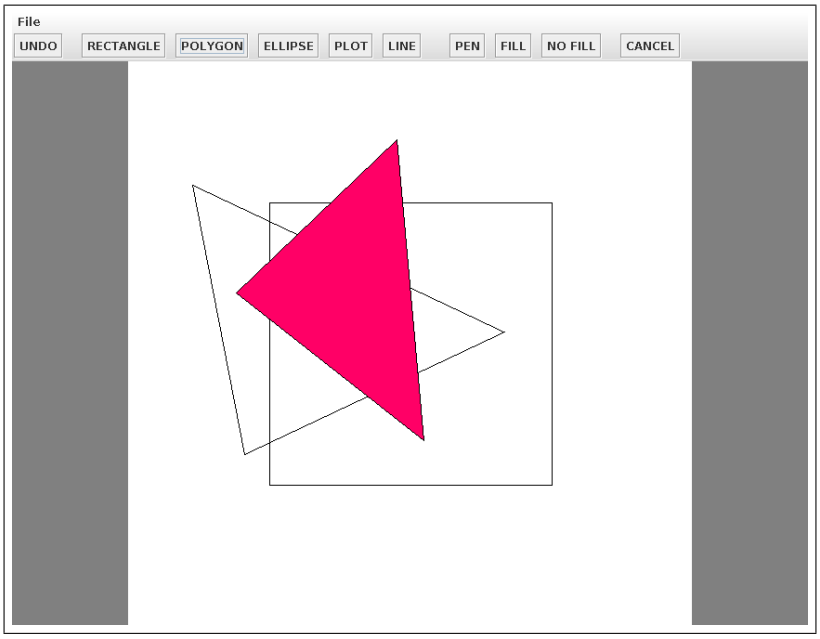
After starting the application, click an option from the tool-bar. If you’ve clicked on an action, you can draw to the canvas by clicking once to start and a second time to complete the shape.



If you've selected the polygon option, left-click to add a point and then right-click to add the final point and complete the polygon. To select a fill or pen colour, click the pen or fill option.



A dialog with a palette will open up, selecting a colour will apply to actions that follow.



Finally, to save the new vector select Save from the File menu.