CAB302 Software Development Assignment 2 Report

Student 1 Name: Daryl Tan
Student 1 ID: n9930141

Student 2 Name: Wesley Kok
Student 2 ID: n9972676

Date: 02/06/19

Assignment Implementation Summary

Item Number	Item Description	Implementation Level
1	PLOT function	Fully implemented
2	LINE function	Fully implemented
3	RECTANGLE function	Fully implemented
4	ELLIPSE function	Fully implemented
5	POLYGON function	Fully implemented
6	Pen colour	Fully implemented
7	Fill colour	Fully implemented
8	Save file in VEC	Fully implemented
9	Open VEC files	Fully implemented
10	Toolbar	Fully implemented
11	Menu bar	Fully implemented
12	Undo option	Fully implemented
13	1 additional functionality (Undo part 2)	Fully implemented
14	1 addtional functionality (Save as BMP)	Fully implemented

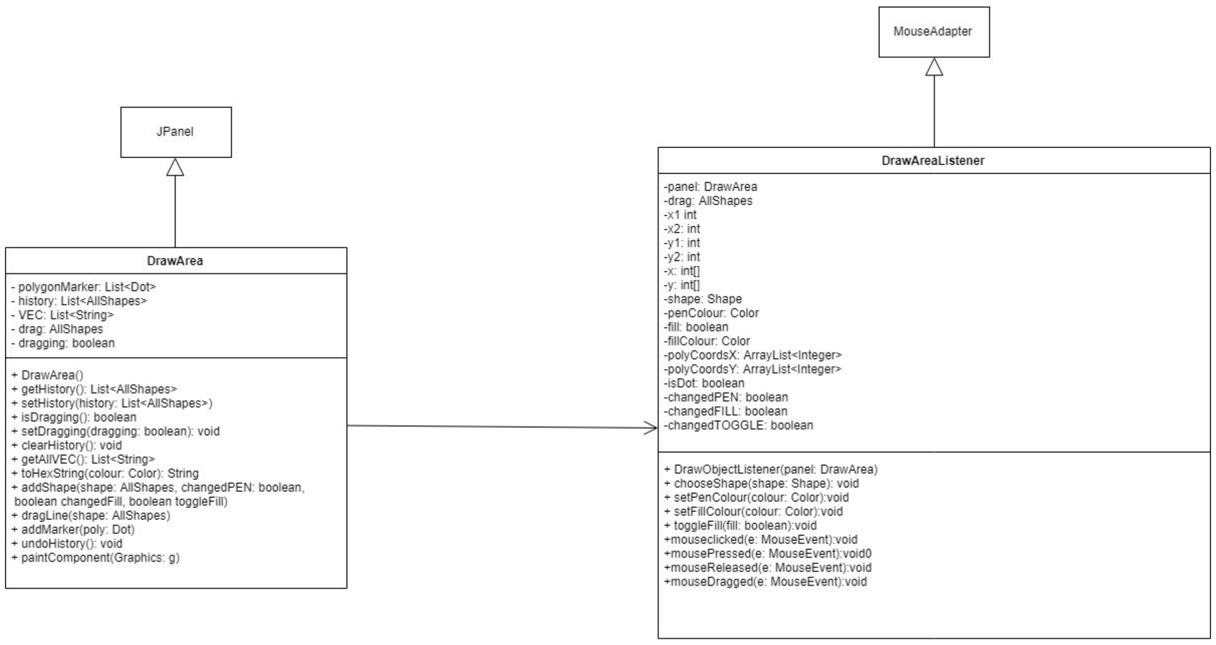
Statement of Contribution

Report	Wesley Kok, n997267 contributed 50% Daryl Tan, n9930141 contributed 50%
Code	Daryl Tan, n9930141 contributed 50% Wesley Kok, n997267 contributed 50%
Testing	Wesley Kok, n997267 contributed 50% Daryl Tan, n9930141 contributed 50%

Development Process

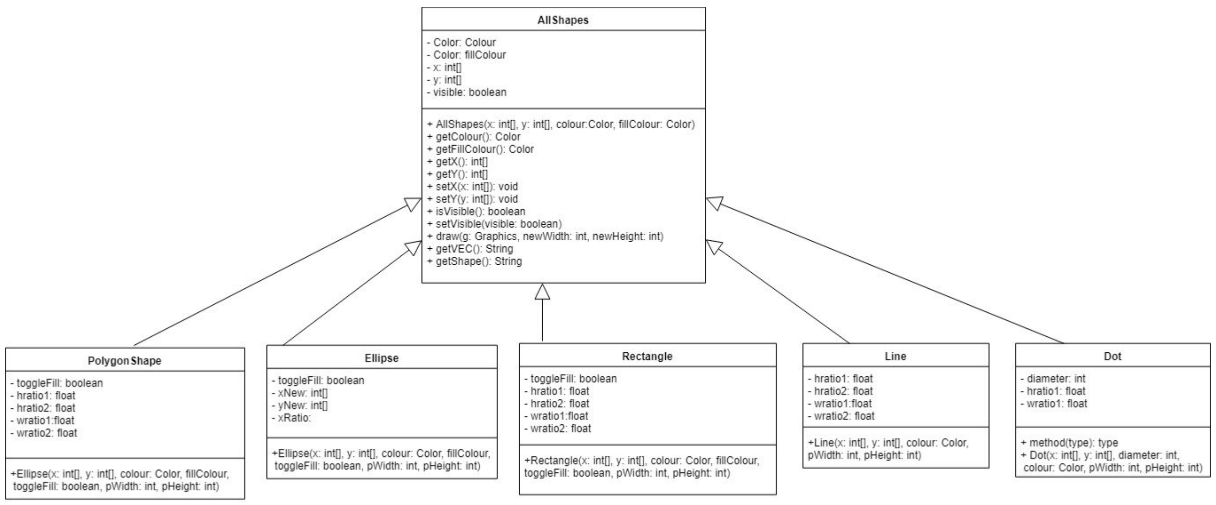
This project development follows the Agile methodology closely, especially for delivering working software frequently. At the start of the development process, basic and small scale features were constantly added, which were mainly done on the repository's master branch. However, as features got more complex and larger in scale, development started on separate branches to experiment with different methods used to implement various ideas. This may sometimes lead to clash in other parts of the software, which was kept separate from the master branch which was mostly bug free and completely working. When it is fully functional, the branch is then merged into the master branch by creating a pull request. With a design in mind, each class was developed with plans to develop into a clean and elegant design to maintain value through various version. Agile methodology also values response to quick changes over following a strict plan. This is strongly valued over choosing additional functionalities for the software as there may be considerations that were not fully thought out properly. Instead of sticking to the waterfall development process, the team can re-consider our options and switch our focus.

Architecture


MouseAdapter
JPanel
DrawArea
- polygonMarker: List<Dot>
- history: List<AllShapes>
- VEC: List<String>
- drag: AllShapes
- dragging: boolean
+ DrawArea()
+ getHistory(): List<AllShapes>
+ setHistory(history: List<AllShapes>)
+ isDragging(): boolean
+ setDragging(dragging: boolean): void
+ clearHistory(): void
+ getAllVEC(): List<String>
+ toHexString(colour: Color): String
+ addShape(shape: AllShapes, changedPEN: boolean, boolean changedFill, boolean toggleFill)
+ dragLine(shape: AllShapes)
+ addMarker(poly: Dot)
+ undoHistory(): void
+ paintComponent(Graphics: g)
DrawAreaListener
-panel: DrawArea
-drag: AllShapes
-x1 int
-x2: int
-y1: int
-y2: int
-x: int[]
-y: int[]
-shape: Shape
-penColour: Color
-fill: boolean
-fillColour: Color
-polyCoordsX: ArrayList<Integer>
-polyCoordsY: ArrayList<Integer>
-isDot: boolean
-changedPEN: boolean
-changedFILL: boolean
-changedTOGGLE: boolean
+ DrawObjectListener(panel: DrawArea)
+ chooseShape(shape: Shape): void
+ setPenColour(colour: Color):void
+ setFillColour(colour: Color):void
+ toggleFill(fill: boolean):void
+mouseclicked(e: MouseEvent):void
+mousePressed(e: MouseEvent):void0
+mouseReleased(e: MouseEvent):void
+mouseDragged(e: MouseEvent):void

https://imgur.com/5Sw6GeH

GUIForm instantiates *JFrame*, *DrawArea*, *JMenus* and *JButtons* and their respective *ActionListener* which acts as Observers. These Observers for buttons wait for the buttons to be clicked, allowing the user to choose shapes, pen colour, fill colour, fill off and undo. The menus' *ActionListener* wait for the menus to be clicked and display the relevant *MenuItem*, allowing user to create new file, open and save file, undo, exit and use history support. *GUIForm* also instantiates *DrawObjectListener* which gets the shape, pen and fill color from the buttons' *ActionListener*. It listens to the mouse activity and draws the relevant shape and colours on *DrawArea*, which is a subclass of *JPanel*.



https://imgur.com/dseLX0m

Abstraction/ Inheritance

An abstract parent class, *AllShapes* is designed which contains methods that the subclasses ("*Dot*", "*Line*", "*Ellipse*", "*Rectangle*", "*PolygonShape*") have in common. The abstract class cannot be implemented, only inherited by the subclasses. These methods, *getVEC*, *draw*, *getShape* are declared abstract because the subclasses have different implementations for these methods. Methods *isVisible* and *setVisible* are not abstract because the implementations are the same for all subclasses of *AllShapes*. The inherited classes then override "*draw*" to draw the respective shapes, and "*getVEC*" to get the VEC command.

Encapsulation

Subclasses of *AllShapes* inherited getters *getShape* and *getVEC* which returns the shape and VEC command respectively. As the shape and coordinates of the drawing are finalised when the mouse button is released, setters are not needed. The *DrawArea* class stores a LinkedList of *AllShapes* as drawing history, temporary polygon markers, VEC commands and keeps track of drag occurrences as private variables. These variables may have getters or setters which will be used by *DrawAreaListener*. The listener listens for events which then calls certain getters or setters to update the value of the variable.

Polymorphism

In *DrawArea* class, there is the method *addShape* that adds *AllShapes* objects into the LinkedList *history*. However, it is actually referring to its child class objects because *AllShapes* is an abstract class that cannot be instantiated, thus utilizing polymorphism and allowing extensibility for future child classes . In the method *paintComponent*, the same concept is utilized to call the *draw* method for each object in the *history* list, where the *draw* method differs slightly for each object of type subclasses of *AllShapes*.

Manual

This software allow users to draw shapes and save the image and VEC commands. Users are able to select 3 standard shapes (line, rectangle, ellipse), polygon, pen and fill colour from the toolbar. Shapes can be selected either by clicked or by pressing shortcut keys *Alt + 1, 2, 3 ,4* for line, rectangle, ellipse and polygons respectively.

Toolbar

- To plot a dot, left click while one of the standard shapes (Line, Rectangle, Ellipse) is selected.
- To draw any standard shape, select it on the toolbar, press left mouse button and drag the cursor on the canvas to determine its size. It will be drawn once the left mouse button is released.

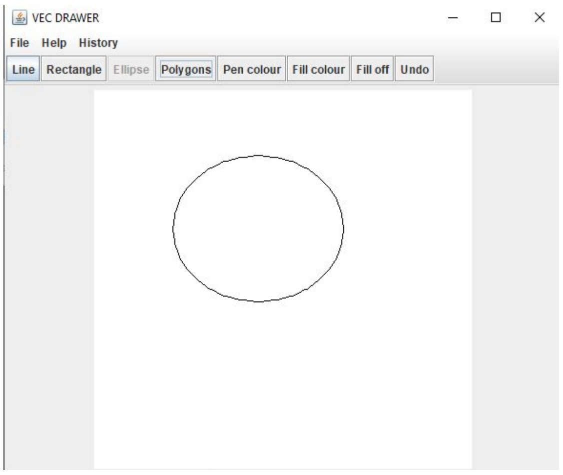


Figure 1. Ellipse tool selected and one ellipse drawn

- To draw a polygon, select it on the toolbar; left click to mark each point of the polygon and right click to draw it.

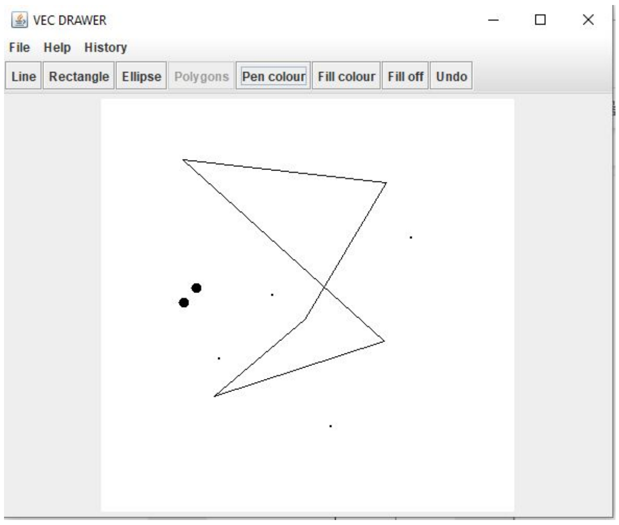


Figure 2. Bigger dots plotted with a 5-edged polygon and smaller dots that are markers for the next polygon.

- To select pen colour, click “Pen colour” button and select it on the toolbar. Pen colour is logged into VEC only when a user draws something with that colour.
- To select fill colour, click “Fill colour” button and select it on the toolbar. Fill colour is logged into VEC only when a user draws something with that colour. It is toggled on when a colour is chosen and can be toggled off by clicking the “Fill off” button

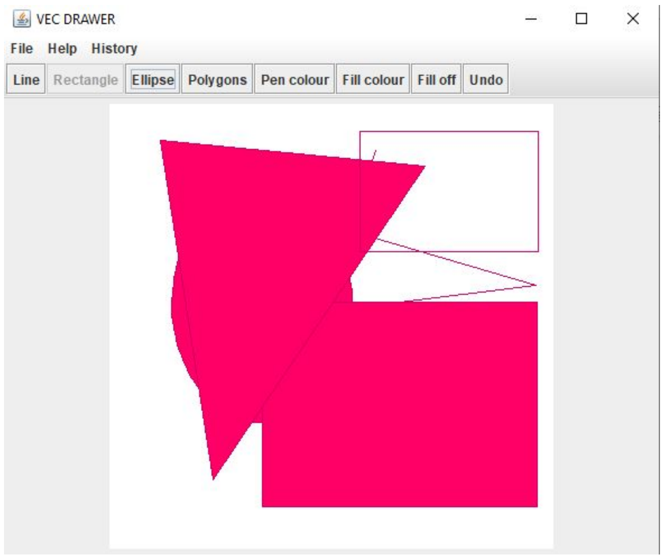
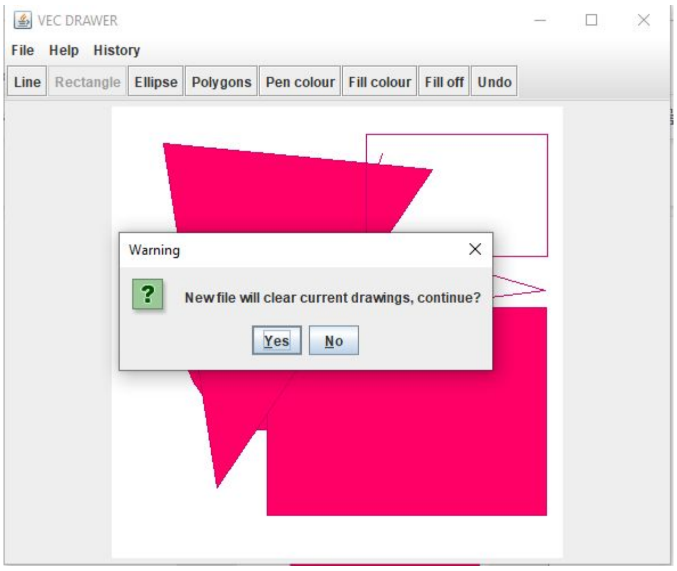

VEC DRAWER
File Help History
Line Rectangle Ellipse Polygons Pen colour Fill colour Fill off Undo

Figure 3. Demonstration of pen colour and fill colour set, with shapes layered on top of one another.

- To remove the last drawing action, click “Undo”

Menu

- File - New File: Clears the canvas. Caution: will override current drawing.


VEC DRAWER
File Help History
Line Rectangle Ellipse Polygons Pen colour Fill colour Fill off Undo
Warning
Yes No

New file will clear current drawings, continue?

Figure 4. Attempt to open new file (blank canvas) while there are drawings.

- File - Open File: Choose a certain VEC file and loads the drawing. Caution: will override current drawing.

- File - Save File: Saves the commands as a .VEC file and saves the drawing as a .bmp file
- File - Exit: Closes application
- History support - This allows you to go back and forth the drawing history. Undo actions will not be part of this history.

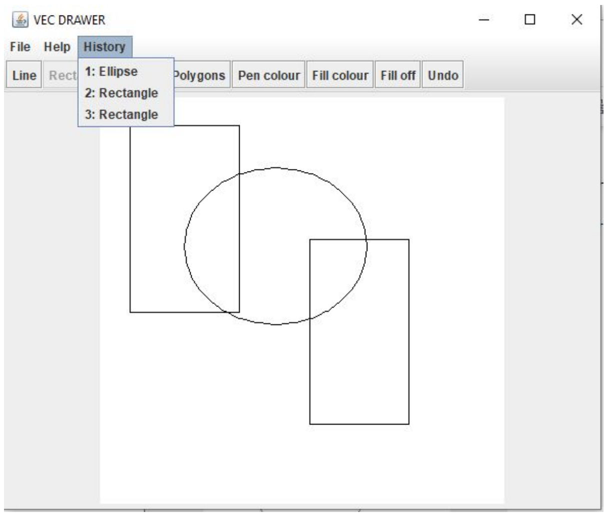


Figure 5. History menu shows order at which shapes are drawn