VECTOR DESIGN TOOL

CAB302 – Software Development

Assignment 2
Group Report
Semester 1, 2019

Abstract

A unique vector design tool has been created in reference to specifications to fast track the design process. Originally the process involved was laborious and required the manual construction of a .vec document used for plotters. The software that has been created to aide in this process allows the user to quickly draw up a model and produce .vec files using an easy to use canvas. Alongside this, additional features have been added to help make the designers life easy. All team members input and management styles have been included in this report to show the process of design, as well as documentation of the code used.

Jarryd Stringfellow (n9734074)
Azeez Bodija (n10010319)
Corey Robinson (n10196587)
Sreya Singh (n9942777)

Contents

Statement of Completeness

Main Program

The project has provided the main functionality required by allowing users to both preview and edit .VEC files. This has been provided on a clean easy to use GUI with other functionalities. The software allows manipulation of plot, line, rectangle, ellipse and polygon vector types. This was required by the designers. The ability to create, save and upload VEC files was also implemented to allow the designers to work on existing files of this type. The ability to change colour for both a pen and fill option were also provided at the designers request. The main design was based off the Microsoft program Paint but was heavily modified to achieve a different result.

Additional Functionality

Additional functionality such as undo history support, BMP file type export, a grid and line width were all added to help improve the users' experience. The undo feature was implemented into the main program, but a history display allowed the user more precise control by selecting a specific implemented feature to remove. The BMP file type allowed for the user to input their own desired dimensions to save the image as, as the BMP format allows for a much higher resolution to be saved as a file - additionally, however, this functionality was also achieved for the PNG format; this may qualify as a minor additionally functionality.

Statement of Contribution

Team Member	Contribution
Jarryd Stringfellow n9734074	- JavaFX initialisation - GUI design and build - GUI functionality - Code quality - Report - Grid feature - Line width feature - Javadoc
Azeez Bodija n10010319	- JavaFX initialisation - Resize canvas - Report - ControllerTests - ShapeTests - Javadoc - PNG export - BMP export - Code quality
Corey Robinson n10196587	- JavaFX initialisation - Code quality - Functionality saving VEC images - GUI functionality - History functionality - Undo/Redo
Sreya Singh n9942777	- JavaFX initialisation - Code quality - Resize canvas - Undo/Redo - Grid

	- Functionality (plot, line, rectangle and polygon) - Functionality loading VEC images - GUI functionality - Javadoc

Statement of Team Management

For the project, Agile team management was used alongside a basic version of test-driven development to maintain constant progress within the team. As the team comprised of a group of four, a solid structure had to be derived for the team to work together. Agile team management was chosen as it breaks the project down into more manageable tasks which can be evenly distributed. "Agile is a project management methodology that uses short development cycles called "sprints" to focus on continuous improvement in the development of a product or service." (Alexander, 2018) At the beginning the team met and broke down the criteria into different functionalities of the application. These were then converted to 'sprints' and a timeline was created.

As there is no customer or stakeholder, the CRA was given this position. All members in the team shared the role of team members and scrum master to share the workload. At different stages of the process, a member would complete their sprint, and another would confirm this by checking its functionality. The use of GitHub was crucial to this role, as the whole process was very iterative.

To confirm that the project was on track, weekly scrum meetings and regular updates were made. The updates correlated to what sprints different members were working on and when they were completed. Weekly meetings were utilised to confirm these changes, bounce ideas around and to plan the next steps. Throughout the process, different sprints were estimated incorrectly. This did not affect the progress however as some sprints proved to be far quicker than estimated allowing multiple tasks to be completed. Whilst other sprints took longer or were interrupted by another task. The progress of the task was not graphed, however use of GitHub's repository and a google doc of tasks was used to keep track of progress.

Initially test-driven development was going to be used but was decided against as it can be a slow process. Rather than use this method, when a specific task was completed, before pushing the update to GitHub, a set of checks was made and confirmed by another group member. At later stages in the project, test cases were generated, and small errors were fixed respectively.

Software Architecture

The project has been implemented using JavaFX with the basic architecture behind the project being based on a model-view-controller (MVC). The three components in the program are *App, UI.fxml* and *Controller* respectively. Other classes have been added to benefit the code quality and legibility and separate different features.

The first class that is looked at is *App*. This class sets up and launches the JavaFX application. The first thing that happens is a stage (window) is created where the scene (contents in window) can be generated from the *UI.fxml* file. This file is an XML-based markup language used is what the UI is generated from. It is broken up into different regions in the scene where the buttons, menu items and tools for the program lie. To add functionality to these controls, a *controller* class has been created to handle all requests.

The *controller* class is where most of the external classes ‘talk’ to as it has the most functionality. This initialises a lot of the GUI functionality the user can see when the tool is launched as well as well as implemented methods for use of the GUI. This class handles majority of the functions that can be see on the interface of the main application. Processes such as the Undo, Redo, Brush Inputs and Save options are all initialised using the controller class. The additional classes that are discussed below are mainly initialised through the Controller class, or have their outputs used in a Controller class.

Alerts class is responsible for giving informative advice on: how to approach a task, what errors may have occurred or specifics required of certain functionalities.

DisplayFile is the class responsible for processing lines (passed in from the current file variable and translating them into drawable shapes. The *DrawPolygon* class specifically designed for drawing shapes of type “Polygon” as other classes use the Shapes class.

DrawShape class passes in the type of shape to be drawn and inherits the Shape methods from the Shape classes dependent on the shape type.

Grid class draws a grid to the canvas and allows the user to modify the cell-size of the grid from input from the GUI, this class is called in the *Controller* class.

ReadFile class chooses the current file selected and reads for usable lines that can be drawn to the canvas, this class contains methods from DisplayFile that allow the file to be displayed once opened.

ResizeCanvas class inherits the current canvas size and scales vectors and the window to display the vectors at the desired size.

SaveFile class records the updated list of vectors and commands associated (including Fill, Pen etc.) and allows it to be exported for use dependant on the type of file the user wishes to export.

Shapes class contains all the methods required for drawing particular shapes based on the users input coordinate values and type of shape wish to be drawn.

UndoRedo class handles undo and redo functionality so the user is able to step in the history of their current design, then displays the changes.

Software Documentation

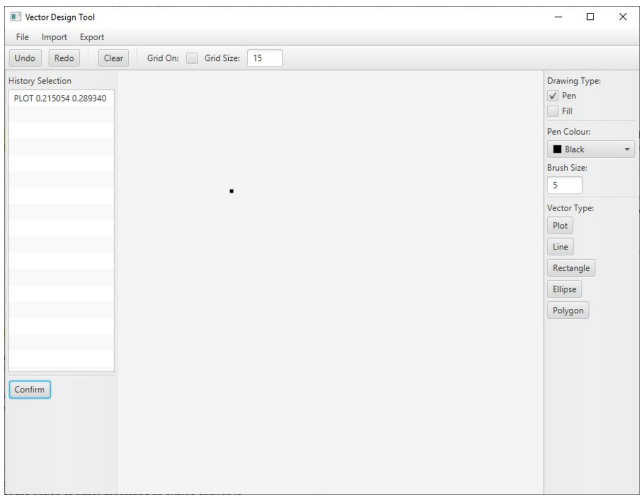
Upon opening of the software there are some tools available to use. Along the top of the window, there are two drop downs, file and export. File lets you either exit the program or open a previously generated .vec file. The export tab gives you the options of exporting the designed vector as either a .vec filetype, a .png or a .bmp image file. The saving of a .png or a .bmp file involves choosing your own dimensions for the saved file, upon a second dialog window being opened.
These tools are often only used at the end of the program and hidden from immediate view. The toolbar along the top of the page has three different buttons, undo, redo and clear. These are used to adjust any changes made to a previous state or to completely remove the vector designed. It also has a checkbox and combobox to select the Grid functionality and choose the grid square size.

The Toolbar on the right-hand side are the main features of the program and are what is used for vector creation and manipulation. Read from top to bottom, the drawing and vector type as well as pen colour and width are available for manipulation. Either the pen or fill box must be checked to begin drawing on the canvas. Starting from the top, the user has the option of either generating an outline or filled in image by checking the Pen or Fill box respectively. The pen colour is then defaulted to black and a size 5 pen. This can be easily changed using the colour picking tool or the change-brush-width text box.

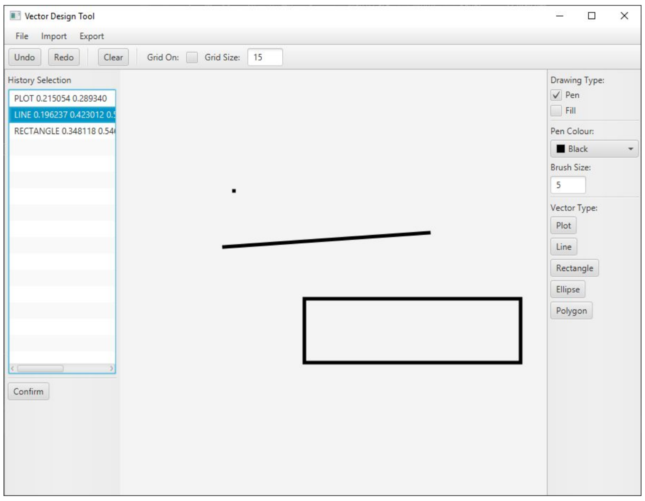
Finally, there are 5 different vector types that can be used varying from a single point to a curved or straight edged shape. All these vector types can be used immediately except for the polygon. If the polygon is selected, an additional dialogue popup is generated. This popup asks the user how many points they would like to draw the shape before letting them continue. From this point, all the user has to do is click or drag their mouse on the central canvas to draw the vector. They can then save their creation as advised earlier.

Undo/Redo and Shape History

Pressing the undo button will remove the last vector and the redo will replace the last vector. There are alerts that will prompt you whether there are no extra shapes to undo/redo. To use history, select in the ListView window (History Selection) the vector you wish to roll back your canvas to.



Then press the confirm button to confirm your choice (this will delete all the shapes up to and including the selected vector).



Grid On: ☐ Grid Size: 15

PLOT 0.215054 0.289340
LINE 0.196237 0.423012 0.5
RECTANGLE 0.348118 0.54

Drawing Type:
✓ Pen
☐ Fill

Pen Colour: ■ Black

Brush Size: 5

If you wish to redo the shapes you've just cleared, pressing REDO button after you've cleared the vectors will bring back the vectors previously cleared.

View Tools
Grid:

Grid checkbox applies a grid to the current sized window when selected, removes once unselected. If window is resized, unchecking and checking the Grid checkbox will re-apply the grid to the sized window.

Advanced OOP Principles

Abstraction

Programming-wise, abstraction involves the selection of data from a larger pool of information, while only showing relevant details to the object. It is generally viewed as the most object-oriented principle, as it greatly reduces the coding intensity required, as well as the complexity of the code written.

An example of this principle can be seen _______

Encapsulation

Encapsulation is the second of the four object-oriented concepts, and involves variables. This concept involves the variables of a single class 'wrapped' as a single unit, along with the code written to affect them. Otherwise known as data hiding, encapsulation is performed through the creation of private fields and variables. While variables are generally the only things being made private, the same can also apply for methods.

Allowing entire classes and several variables to be private means their values, parameters and/or contents can be protected from outside access - this advantage is important when considering programs used in banks, or holding personal information. Because several parts of the code can be remade as each iteration is its own system, encapsulation allows for a more flexible use of values.

An example of this principle can be seen _______

Inheritance

Inheritance involves the derivation of classes through the use of an existing class. This process allows for greater reusability when considering a class, whilst leaving the orignal class unchanged. Reusability is always an advantage when programming, as the development time of applications heavily using inheritance is greatly reduced.

As the derived classes tend downwards - from the base class, that is - the properties in the base classes, more dominant objects are generated. The derived classes that are generated through the use of this method can make use of the the objects and variables above them in the hierarchy. This is achieved via the use of subclasses.

Polymorphism

An example of the final principle can be seen _______

Unit Tests

Unit tests are applied in Test Classes to test the methods in the classes called by the Application. There are two test classes implemented, ***ControllerTest*** and ***<u>ShapesTest</u>***, with six test methods total.

ControllerTests

0. setUpController

The setUpController method throws an instantiation error if the user creates an instance of a class through the ***newInstance*** method but cannot be instantiated because the interface is an interface or an abstract class.

1. testBrushInput

The testBrushInput test class involves the assertion of brush input values. The application requires an integer value between 1 and 200 in order to continue to the next stage of increasing the brush's size. This test created in the ControllerTests class checks, via assertion and the user's inputted test values, whether or not the test input values are valid.

They are useful to the testing process as they can be used to test intended values used for the brushInput processes that affect a large amount of the drawing process. This is achieved using boolean outputs from the inputBrushValue() method in the Controller class.

2. testCanvasHistory

The testCanvasHistory test class involves the assertion of inputted choices regarding shape creation. Throughout the code used, the append command is used to attach the user's instructions into a **savefile**. The testCanvasHistory test class reviews this process on a smaller scale, allowing for a more analytical approach to be used when finding errors on the larger scale.

3. testEmptySaves

The application uses a test class call testEmptySaves to check for instructions when the save command is used. The previous test class went through the append command for instructions on the smaller scale, while this test operate on the same scale regardless. It uses booleans and the **savefile** in the Controller class to check how many instructions have been written. Obviously, there are none initially, and the user's inputted variable should always be greater than zero. if the value is smaller, the error is raised.

When the singular error is raised in the testEmptySaves class, a line of text is generated in the Test window, explaining that the number of edits is invalid and must be greater than zero in order to save the file.

ShapesTest

0. setUpShapes

The setUpController method throws an instantiation error if the user creates an instance of a class through the ***newInstance*** method but cannot be instantiated because the interface is an interface or an abstract class.

1. testCoordinates

An array inside of an array is used when initialising the shapes for the coordinates used throughout the application. These coordinated must be numerical in form and always equal to or greater than zero. This is tested in the ShapesTest class, and allows the user to input any value that they have chosen.

This test is marginally important as the coordinates play a vital role in every process in the program, including saving, uploading, drawing and clearing. The user is given the role of choosing the affected value, and the text output of the error shows which value is incorrect in the text window.

Conclusion

The initial software was designed to provide aide to a group of designers that required to manipulate vector drawings. The program provided has not only met these needs but has also added extra functionality designed to improve the designers experience. Throughout the project, different OOP principles have been used to demonstrate the effectiveness of using Java. Along with using the JavaFX GUI to provide the user with a clean manageable interface. To prove functionality, test cases have been created and run and all stages of progress have been tracked on GitHub. This project has been a success and a great learning experience along the way.

References

Alexander, M. (2018, 6 19). *Agile project management: A comprehensive guide.* Retrieved from CIO: https://www.cio.com/article/3156998/agile-project-management-a-beginners-guide.html

Appendix