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)

Vector Design Tool

CAB302 – Software Development

Assignment 2

Group Report

Semester 1, 2019

Contents

[Statement of Completeness 2](#_Toc9958139)

[Main Program 2](#_Toc9958140)

[Additional Functionality 2](#_Toc9958141)

[Statement of Contribution 2](#_Toc9958142)

[Statement of Team Management 2](#_Toc9958143)

[Software Documentation 3](#_Toc9958144)

[Advanced OOP Principles 3](#_Toc9958145)

[Abstraction 3](#_Toc9958146)

[Encapsulation 3](#_Toc9958147)

[Inheritance 3](#_Toc9958148)

[Polymorphism 3](#_Toc9958149)

[Software Documentation 3](#_Toc9958150)

[Conclusion 3](#_Toc9958151)

[References 4](#_Toc9958152)

[Appendix 5](#_Toc9958153)

# Statement of Completeness

A statement of completeness, describing what functionality your team managed to implement – both the basic functionality and any applicable additional functionality (note that you must implement 1 item of additional functionality for each person in your team beyond the first.)

## Main Program

Draw vectors

* Plot
* Line
* Rectangle
* Ellipse
* Polygon

VEC files

* Create
* Save
* Upload

## Additional Functionality

Grid

Undo history support

BMP Export

Multi image support???

# Statement of Contribution

A statement of contribution, describing who is in your team and what each team member contributed to the project – report writing, code, creation of unit tests etc.

|  |  |
| --- | --- |
| **Team Member** | **Contribution** |
| Jarryd Stringfellow  n9734074 | * JavaFX initialisation * GUI design and build * GUI functionality * Code quality * Report * Grid feature * Javadoc |
| Azeez Bodija  n10010319 | * JavaFX initialisation * Resize canvas * Report * Unit tests * PNG export * BMP export - Dimensions * Code quality |
| Corey Robinson  n | * JavaFX initialisation * Code quality * Functionality saving VEC images |
| Sreya Singh  n9942777 | * JavaFX initialisation * Code quality * Resize canvas * 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](https://www.cio.com/article/2950579/methodology-frameworks/how-to-pick-a-project-management-methodology.html) 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 Documentation

Documentation of your software architecture, describing all the classes that make up your project and how they interact with the other classes to bring your project together.

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 *control* class has been created to handle all requests.

The *control* class is where most of the external classes ‘talk’ to as it has the most functionality.

# Advanced OOP Principles

Documentation of how you made use of advanced object-oriented programming principles in the creation of this software. Have a section for each of:

## Abstraction

## Encapsulation

## Inheritance

## Polymorphism

# Software Documentation

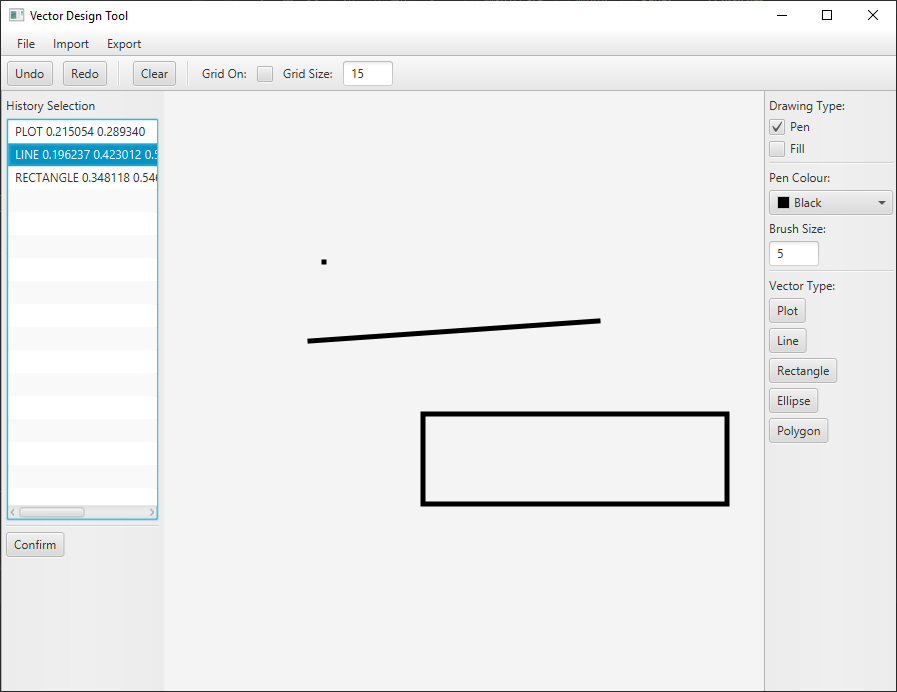
Documentation of how to use your software, with associated screenshots showcasing all the items of functionality

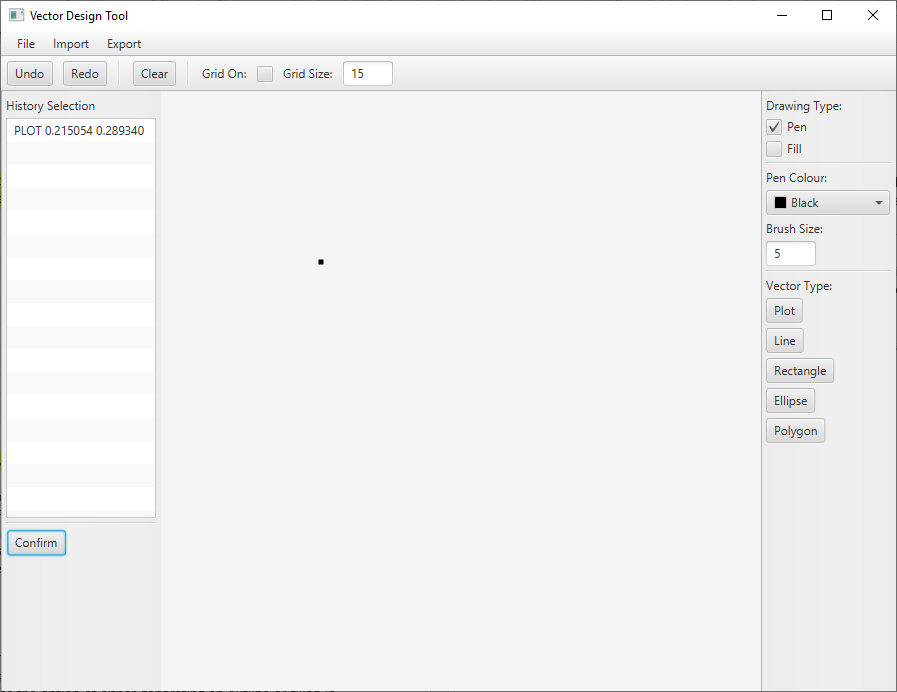
Upon opening of the software there are some tools available to use. Along the top of the window, there are two drop downs, file and export. File lets you either exit the program or open a previously generated .vec file. The export tab gives you the options of exporting the designed vector as either a .vec filetype or a .png image file. 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.

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. All features must be selected to first generate an image. Starting from the top, the user has the option of either generating an outline or filled in image. The pen colour is then defaulted to black and a size 5 pen. This can be easily changed for different objectives. 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).

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.

# 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 ***ShapesTest***, with six test methods total.

## ControllerTests

1. ***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***
2. ***testClearCanvasHistory***
3. ***testEmptySaves***

## ShapesTest

1. ***setUpShapes***
2. ***tesCoordinates***

test input coordinates to be greater than zero

# Conclusion

# 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