

Individual Report

CS2JA17: Programming in Java

University of Reading

Jason Jay Dookarun
Student Number ° 26017434
Date of Submission: December 8, 2020
Time Spent: 35 Hour(s)

Assignment Evaluation:

This project permitted me to learn Java and JavaFX and practice theory content towards a real-time project. Furthermore, I was able to execute methods from various modules (Software Engineering) and combine the mentioned as part of my development process. Finally, by delivering a video demonstration and report, I was able to intensify my communication skills to warrant a professional tone was maintained, with consistency and coherency.

Table of Contents

Introduction	3
Object Oriented Programming	4
User Manual	5
Testing	8
Conclusion	10
Gitlab Repository	10
GUI Demonstration Mark Form	11
CS2JA16 Coursework JAVA: GUI DEMONSTRATION MARK FORM	11

Introduction

As part of the CS2JA syllabus studied at the University of Reading (2020-21), the task focused on developing a two-step drone simulation, using coding languages, namely Java and JavaFX. The first stage involved creating a programme whereby a user could view and interact with the drones through a terminal execution process. Phase two focused upon taking this approach and further developing the project to ensure the user could now visually see the project's run-time, conducted through the use of JavaFX. JavaFX is a coding language used to animate any program, via a view pane such as a new window. It focuses on delivering the developer's a package where techniques applied result in "design, create, test, and debug, and deploy rich client applications that operate consistently across diverse applications". ¹ Such includes 2D and 3D application support for graphical content, useful for designing applications for all platforms, including different user operating systems, such as Windows, iOS, macOS, Linux, to name a few. It is a JDK installed upon any platform and can apply to a Java project ².

The GUI concept of the project titled "Drone Simulation: GUI Edition" consisted of numerous modules that were not previously in the first edition of the project, namely "Drone Simulation: Terminal Edition". Such involved the application of a new graphical user interface, a file handling mechanism, designed to focus upon loading a pre-saved custom project, a toolbar, providing the client with additional tools such as exiting the program or support, and an information pane to review the content such as existing drones in the system and the current arena size, attached to version control to trace changes. These modular components are discussed, with relation to the methodology and structures applied to ensure the success rate of this project.

As part of the version control procedure, I decided to use CSGitLab, a sub-platform of GitHub, to upload my work. Such was useful as it allowed me to track my progress and view changes made to the project. Moreover, to ensure that both versions of the project were active, I was able to create a branch tag release, whereby existing work would not be affected, resulting in multiple products under the same tree structure. Furthermore, by using GitHub as a tracking and storage mechanism, it provided me with the ability to revert to previous commits made to the project in the scenario that the code was unable to run due to catastrophic errors, such as environment setup.

¹ <https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm>

² <https://github.com/openjdk/jfx/blob/jfx15/doc-files/release-notes-15.md>

Object Oriented Programming

Firstly, it was suitable to examine all methodology types, to grasp the flow of the project and understand all modules involved. To assure the project's success, I originated by apportioning the components involved into modules, implementing a bottom-up approach as well as a scrum approach³ to maintain consistency during development phases. Further, as this project used Java, an object-oriented programming language, it was suitable to converge upon a methodology adhering to such principles, namely, modular approaches. Object-Oriented Programming (OOP) centres on the objects that developers wish to manipulate, rather than the logic involved.⁴ Numerous positives are associated with such methodologies, such as code reusability, scalability and efficiency, to name a few. The first steps often involved as part of OOP, is to collect all objects involved, and identify how they correlate to one another. To ensure such was examined, I initiated by creating a UML Class Diagram to consider all the components/objects involved and discern the interactions made between each class. Such was appropriate as this project focused on being a stage two to an existing project, namely Drone Simulation: Terminal Edition.

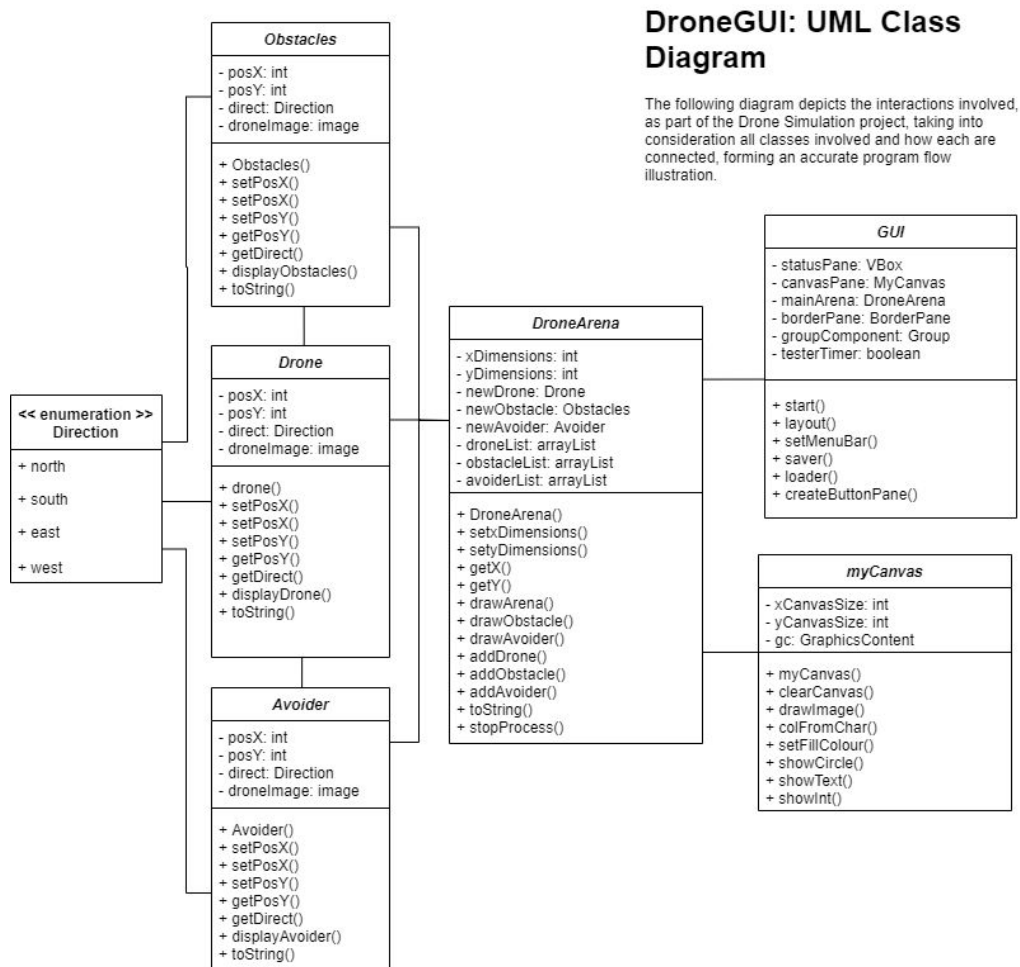


Figure A: UML Class Diagram for DroneGUI Project

³ <https://www.atlassian.com/agile/scrum>

⁴ <https://searcharchitecture.techtarget.com/definition/object-oriented-programming-OOP>

As demonstrated, the enumeration *Direction* focuses on providing an additional sensor to the *Drone* class. Both the *Avoider* class and *Obstacles* class are extensions of the *Drone* class through inheritance. *Drone()*, *Avoider()* and *Obstacles()* focus on creating a drone. These three classes then interact with the *DroneArena* class, the platform whereby objects are added, leading to the main arena's design and implementation coming to fruition. *DroneArena()* focuses upon the development of the arena dimensions, adding objects to the arena and drawing the component, (with interactions of other classes). Previously, as part of Phase One, the *DroneArena* class would interact with a *DroneInterface*, the class designed to focus upon presenting the content to the user, via the terminal. However, as mentioned, this phase focuses upon a graphical user interface, and thus, a GUI class is created, to illustrate visually the status of the system and progress involved. Such interacts with both *DroneArena* and *myCanvas*, a class designed to handle calculations, provided and created by Professor Richard Mitchell. The *GUI()* class focuses on providing the user with the opportunity to interact with the program, through creations of custom configurations, adding objects and viewing their position(s) in real-time.

User Manual

The GUI project focuses on providing the user with a platform whereby the user is granted additional control upon the program, visually. Once the user runs the program, they are firstly welcomed, with a welcome message before commencing the project.

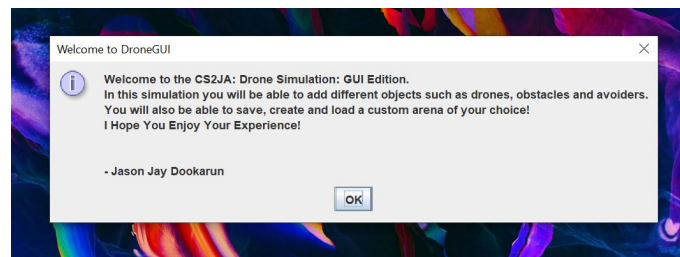


Figure B: DroneGUI: Introductory Welcome Message

The user will then be able to click OK, to load the program. Once the user clicks the button, the main pane will be loaded as shown. The pane will provide the user with a default 512 by 512 canvas to utilise. As illustrated, the pane is on the left-hand side, with a status pane on the right-hand side, to state any content or update deemed necessary. A toolbar is also provided to the user as shown, on the top of the window. The pane also includes a set of buttons, positioned at the bottom for the user's interactions.

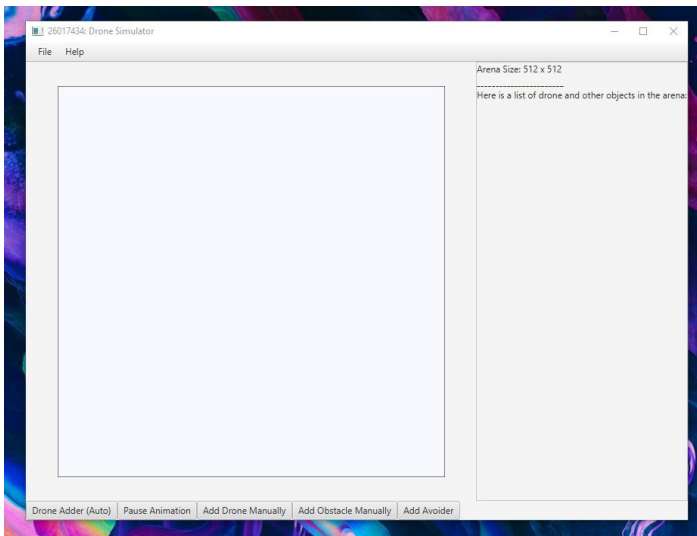


Figure B: DroneGUI Main Pane

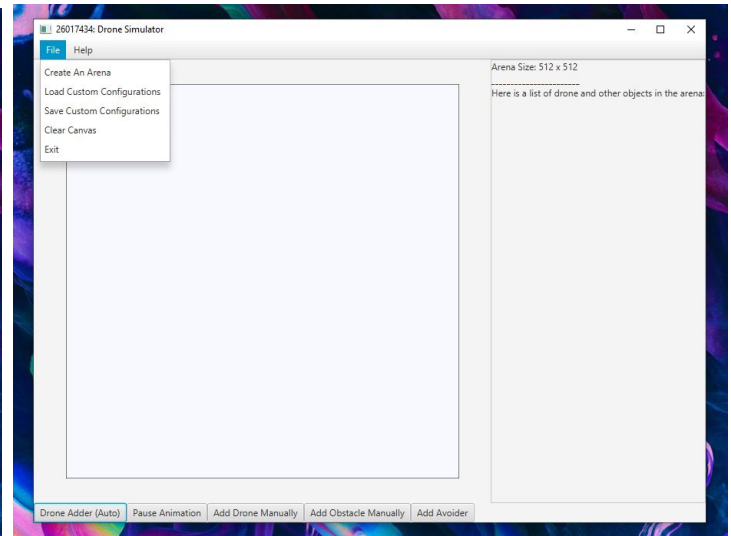


Figure C: DroneGUI File Toggle

The user is provided with additional features as demonstrated in **Figure C**, whereby under the File banner, the user can Create An Arena, Load Custom Configurations, Save Custom Configurations, Clear Canvas and Exit the program. These features were kept under the "File" header to ensure the program maintained a minimal design with reduced complications. Moreover, having these features resemble a similar structure to one of a word-processor such as Microsoft Word whereby New, Open and Save functionalities are stored under the mentioned header.

With loading and saving arena files, any extension(s) can be accepted, for user preferences. "Creating an Arena" allows the user to create a brand new canvas pane with custom dimensions. Finally, clearing canvas deletes any existing content visible on the user's screen, similar to a Restart Project mechanism.

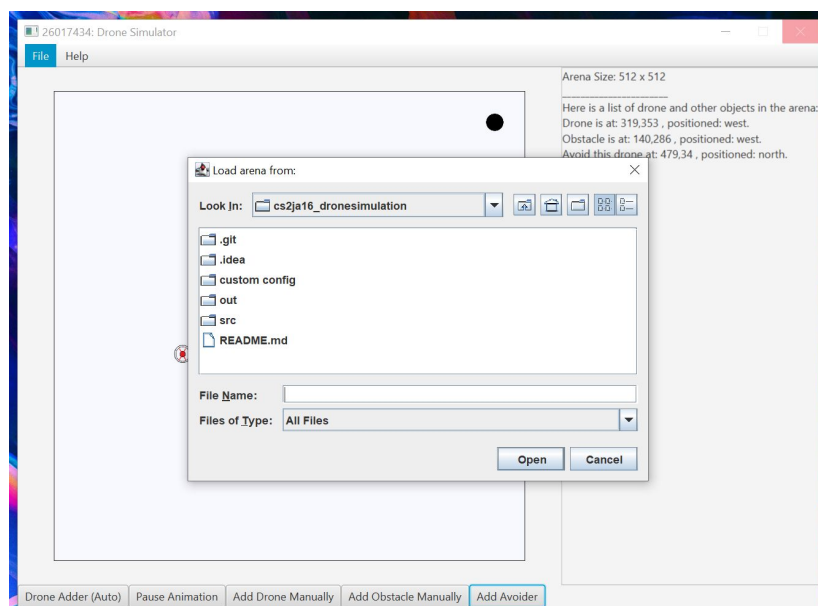


Figure D: DroneGUI File Loader

Moreover, a help panel is displayed (as shown in **Figure E**). Such provides the user with the ability to understand the concept involved as part of the project. Such is supported by the help section and the about pane. The help section, as shown, focuses upon providing the user with a message depicting how to interact with the system. The about sub-section focuses on the creator of the system via a message box.

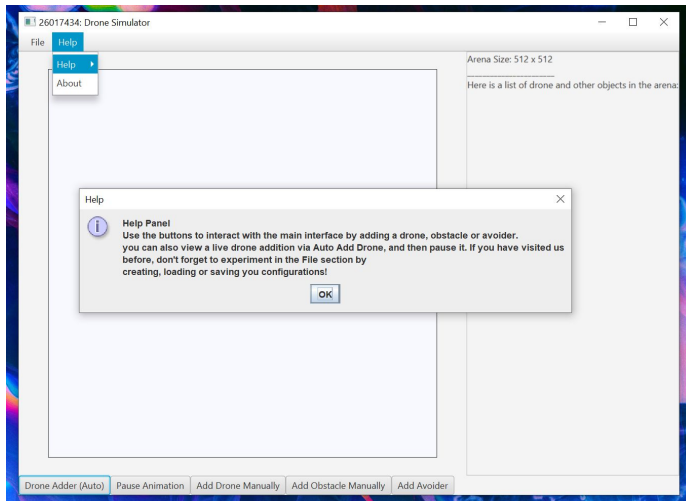


Figure E: DroneGUI Help Panel

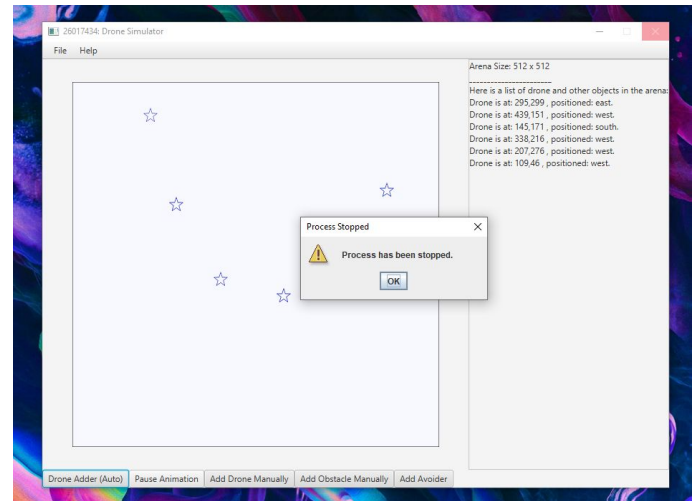


Figure F: DroneGUI Drone Adder, Pause

The user is provided with the opportunity to add drone(s) automatically, via a button press (see **Figure F**). This function focuses upon drawing a drone upon the canvas, with a minor second delay to ensure that the progression can be viewed. If the user wishes to stop this process, this can be achieved by clicking Pause Animation, whereby the user will be presented with a message box, saying the process has been stopped. An identical message would also be presented in the terminal and the status pane as a validation setup. Similarly, as shown in **Figure G**, the user can add an avoider, an obstacle and a drone as part of the canvas. These objects inherit characteristics from the Drone class as mentioned, but focuses upon illustrating different types. Finally, if the user wishes to exit the program, such can be achieved by going to File, then clicking Exit, or pressing the Close button on the window.

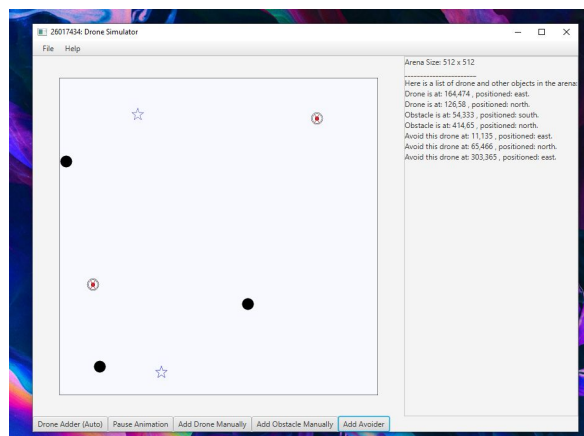
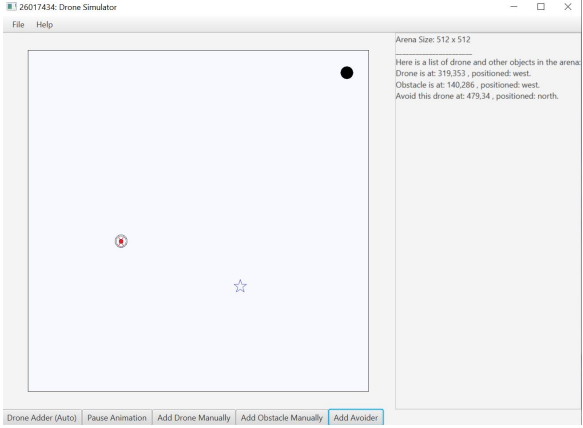
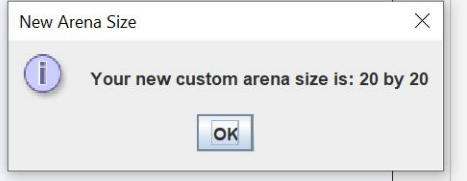
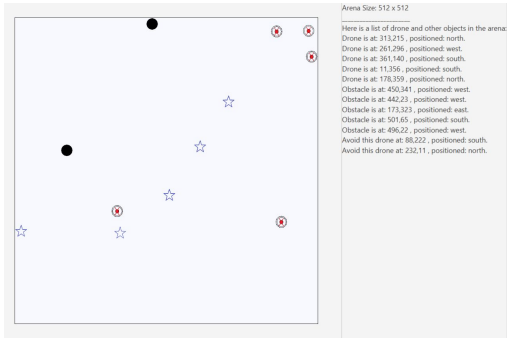
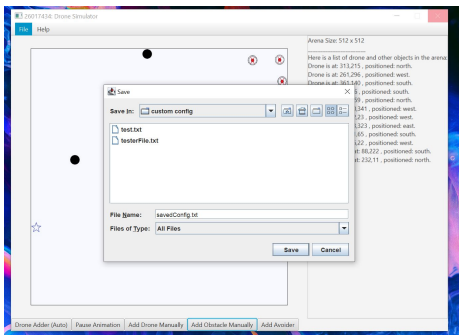
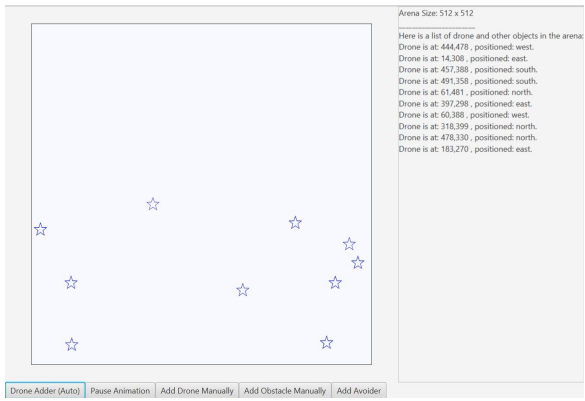
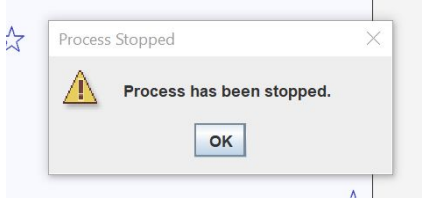


Figure G: DroneGUI Obstacle, Drone and Avoider Addition

Testing

To ensure the product works to the set expectations, methods were coded and attached as part of the main() method. Such focused upon utilising all methods involved as part of the class. This ranged from toString() methods to custom methods designed to focus on adding elements. Find a table below listing all tests completed.

Tested Class	Details of Test	Expected Outcome	Actual Outcome	Success ?
Drone	Creates new drone, prints drone details	Drone is at: (x,y) , positioned: (direction).	Drone is at: 5,6 , positioned: south.	✓
Obstacles	Creates new obstacle, prints obstacle details	Obstacle is at: (x,y) , positioned: (direction).	Obstacle is at: 15,50 , positioned: west.	✓
Avoider	Creates new avoider, prints avoider details	Avoid this drone at: (x,y) , positioned: (direction).	Avoid this drone at: 10,20 , positioned: north.	✓
DroneArena	Adds a drone, avoider and obstacle to the arrayList and prints the contents, and the arena size.	Arena Size: 20 x 10 Here is a list of drone and other objects in the arena: Drone is at: (x,y) , positioned: (direction). Obstacle is at: (x,y) , positioned: (direction). Avoid this drone at: (x,y), positioned: (direction).	Arena Size: 20 x 10 Here is a list of drone and other objects in the arena: Drone is at: 14,3 , positioned: west. Obstacle is at: 6,9 , positioned: south. Avoid this drone at: 5,9 , positioned: north.	✓
Direction	A random direction is picked	Direction, either: north or south or east or west	east	✓
GUI	Adding a Drone	The expected outcome is for the screen to show all of the three object types on the canvas.		✓
	Adding an Obstacle			✓
	Adding an Avoider			✓

	Creating a Custom Arena	Expected result is: user should be able to enter a custom set of coordinates, and then be presented with a message of confirmation.		✓
	Loading a Custom Arena	User(x) should be presented with a load pane and once loading, the details should be presented on the screen. Tester file loaded: custom config/TesterFile.txt		✓
	Saving a Custom Arena	If the user wishes to save the file, they should be able to access File > Save Custom Configurations. They should then be provided with a save panel with the ability to save.		✓
	Starting Auto Drone Adder	When the user clicks the Drone Adder(Auto) button, drones should be automatically added to the panel, with a delay of 1000ms.		✓
	Pausing Drone Adder	Once the Drone Adder(Auto) is underway, the user should be able to click the Pause Animation button, with a message appearing to notify the user.		✓

Conclusion

To conclude, following the principles of OOP, the project was successful in the capacity whereby objectives were accomplished, as demonstrated in the testing table. Utilising a UML Class diagram supported the development of the project whereby I, as a developer, was able to comprehend the flow involved between objects, namely classes and enumerations. Moreover, by applying a bottom-up approach, I was able to focus on individual modules, and as a result, once each module was completed and tested, I was able to develop the next components. Despite this project's development commencing in September 2020 through Phase One, several components were kept, and as a result, became a foundation piece to the Phase Two: Drone GUI setup.

As a developer, utilising JavaFX provided me with the opportunity to experiment with a graphical pane, whereby the content I created could be appropriately demonstrated. Applying certain techniques were difficult as this was my first opportunity to work with JavaFX, however, with determination, I was able to incorporate a range of techniques, studied through practical contents and during phase one of the Drone Simulation project, allowing me to achieve a high-quality product.

Throughout the course of this project, I believe that by implementing a scrum structure ⁵, I was able to not only meet my targets on this project but, I was able to ensure that deadlines were met with space to make further improvements if deemed necessary. Using tools such as a weekly task manager and employing a sprint setup, whereby I ensured tasks were completed on a weekly basis, allowed me to maintain a level of consistency to ensure my work met my expectations. Finally, by utilising a version control management platform like CSGitLab, a sub-platform of Git, I was able to easily maintain version control, ensuring that, if any errors were to occur, I would be able to easily analyse the necessary, and if required, be able to revert to the previous content. Likewise, by using version control, I was able to review my progress and recent changes, ensuring that any later commits increased the product value.

If I was granted the opportunity to work on this project again, I would attempt to change the design further, by creating a menu pane whereby the user could interact more with the program. Moreover, if granted a similar timescale, I would experiment and study other techniques and principles used in JavaFX to enhance the final product, meeting the expectations of any future clientele.

Gitlab Repository

To view the full repository, please visit: https://csgitlab.reading.ac.uk/qt017434/cs2ja16_dronesimulation

To view Phase *One* of this project (Drone Simulation: Terminal Edition), please view the **Master branch**.

To view Phase *Two* of this project (Drone Simulation: GUI Edition) view the **GUI branch**. Please do note that the **repository is private** and request to access may be required unless previously authorised by Jason Jay Dookarun.

⁵ <https://www.mountingoatsoftware.com/agile/scrum>

GUI Demonstration Mark Form

CS2JA16 Coursework JAVA: GUI DEMONSTRATION MARK FORM

Print this out, fill in your assessment in the first column **BEFORE** the demonstration and be set up ready to demonstrate your program and explain your code to the marker

STUDENT NAME: JASON JAY DOOKARUN			
	Student's own assessment		Mark range
1.	<p>Overall OOP design, API and code style: following Java code conventions:</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> variable and class names <input checked="" type="checkbox"/> method names <input checked="" type="checkbox"/> indentation rules <input checked="" type="checkbox"/> using inline developer comments, <input checked="" type="checkbox"/> using Javadoc comments, <input checked="" type="checkbox"/> abstract class <input checked="" type="checkbox"/> good hierarchy for items in building <p>Overall OOP approach must be described in the report under the OOP design including abstraction, encapsulation, and information hiding choices. Appropriate use of inheritance, access modifiers for classes, attributes and methods (Information Hiding).</p>	<p>Number of Classes:</p> <div style="border: 1px solid black; width: 100px; height: 30px; margin: 5px 0; text-align: center; line-height: 30px;">6</div> <p><input checked="" type="checkbox"/> Inheritance used</p> <p>Access modifiers check:</p> <p><input checked="" type="checkbox"/> Correct</p> <p><input type="checkbox"/> Less than 5 incorrect</p> <p><input type="checkbox"/> More than 5 incorrect</p> <p><input checked="" type="checkbox"/> Static variables and methods used correctly</p>	0-25
2.	<p>JavaFX GUI design and implementation: usability, robustness, quality, user-friendliness.</p> <ul style="list-style-type: none"> <input type="checkbox"/> Crashes <input type="checkbox"/> Feedback to user instead of crashing or recover <input checked="" type="checkbox"/> Runs smoothly without interruptions <input type="checkbox"/> Lagging functionality <p>This must be described in the report</p>	<p>Design quality:</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Professional looking <input checked="" type="checkbox"/> Understandable <input checked="" type="checkbox"/> Toolbar buttons <input checked="" type="checkbox"/> Menu <input checked="" type="checkbox"/> File handling 	0-10
3.	<p>Animation: and control (tick all that applies)</p> <ul style="list-style-type: none"> <input type="checkbox"/> Animation attempted but not fully working <input checked="" type="checkbox"/> Animation works with Start and Stop <input checked="" type="checkbox"/> Controlled using Menu control <input checked="" type="checkbox"/> Has Toolbar buttons 	<p>Comments:</p> <p>User is provided with a toolbar at the bottom of the pane to automatically add Drones() to the pane.</p>	0-10
4	<p>Arena: does it support different entities</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Drones which turn when meet walls/drones/objects <input type="checkbox"/> Objects which don't move <input checked="" type="checkbox"/> Drones with additional sensors <input checked="" type="checkbox"/> Panel listing arena contents <p>Design must be described in the report</p>	<p>Total number of different types in arena:</p> <div style="border: 1px solid black; width: 100px; height: 30px; margin: 5px 0; text-align: center; line-height: 30px;">3</div> <p>Any other feature(s):</p>	0-20

5.	Quality of the submitted report: Tests and discussion in report Executable jar archive + embedded javadoc submitted.	Evaluation based on submitted work:	0-20 0-5
6.	Bonus marks for innovation / outstanding features	Academic Evaluation:	0-10
7	<input checked="" type="checkbox"/> Have you submitted Video [Mandatory]		Yes/No
8	<input checked="" type="checkbox"/> Have you provided CSGitLab Link here [Mandatory]: Paste your CSGitLab link: https://csgitlab.reading.ac.uk/qt017434/cs2ja16_drone_simulation.git Note: view branch titled “GUI” for GUI Content		Yes/No