

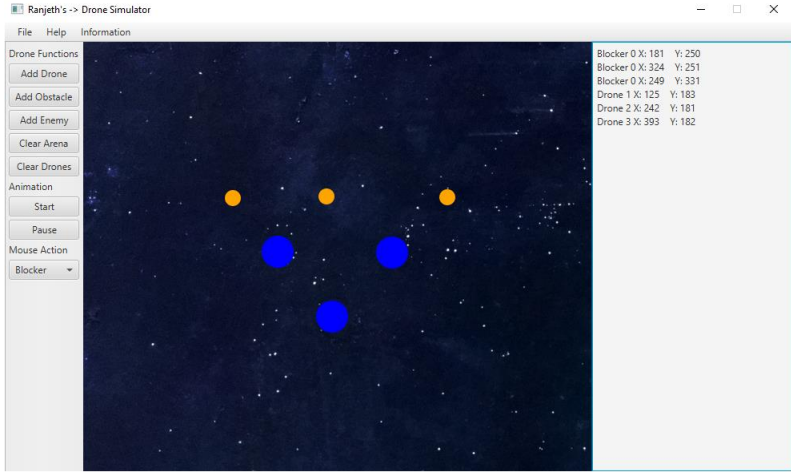
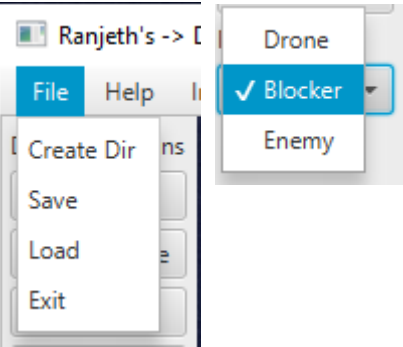
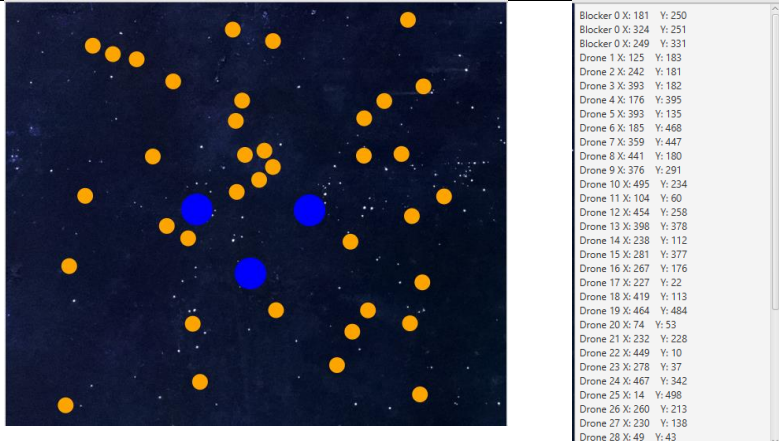
- Module Code: CS2PJ20
- Assignment report Title: Java Programming
- Student Number (e.g., 25098635): 29003671
- Date (when the work completed): 10/12/2021
- Actual hrs spent for the assignment: 23
- Assignment evaluation (3 key points):
 1. Console version of drone completed.
 2. JavaFx version of drone sim.
 3. Unique features to drone sim.

CSGITLAB: <https://gitlab.act.reading.ac.uk/zj003671/cs2pj20-cw>

Drone Simulator

The coursework tested my ability to code proficiently in java and use various libraries, thus allowing myself to import unique features such as a “Level Editor” which the user can test various scenarios and create environments for the drones to traverse along, this also benefits each object’s unique attributes to be tested to the limit.

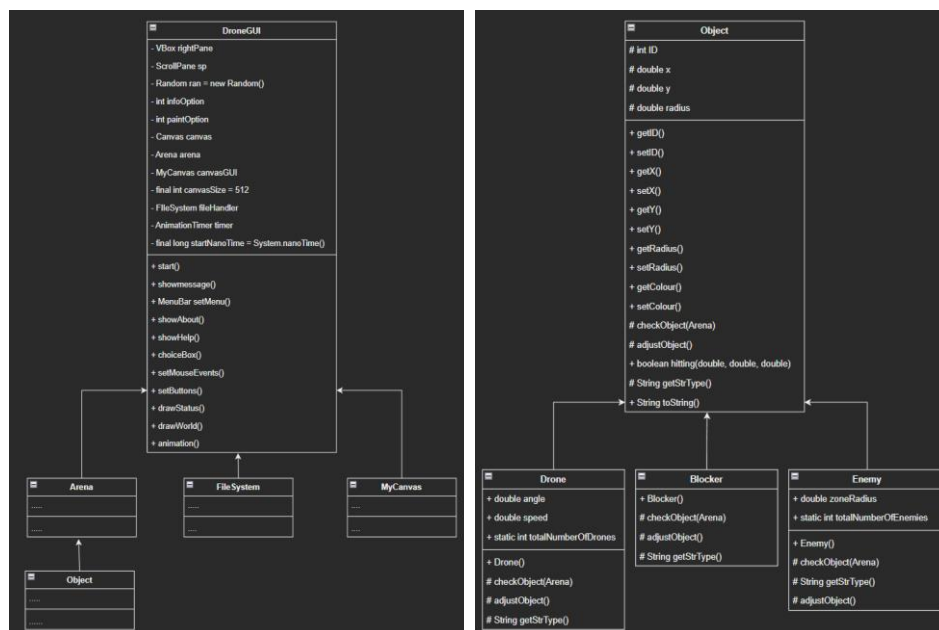
Introduction and Showcase

Screenshot	Description
	<p>This screenshot represents how the program looks overall, the toolbar is located on the left sided, the canvas directly in the middle, and the information panel on the right side filled with data, also at the top is the menu bar which has miscellaneous actions.</p> <p>The toolbar located on the left side of the program holds buttons which all perform actions which can manipulate the canvas. This was an indented design as each button can be presented clearly without miscommunication.</p>
	<p>These two screenshots show how various menu functionality can be utilised throughout the program. The first screenshot illustrates the file handling system, and each sub action handles a specific function related file handling, “Create Dir” declares a directory for the arena files to be located, whereas save and load are self-explanatory therefore the user can easily identify these actions.</p> <p>A choice box was used to hold multiple sets of actions/data as only one can be performed at a time, this is so the user and the program will not be overwhelmed with too much information at once, as well as functionally this choice was reasonable.</p>
	<p>The first screenshot in this scenario is the canvas this is populated with objects, such as Drones and Obstacles. Each object has its own unique attributes and methods. These objects are visualised in the canvas with data accompanying it in the Information panel on the right side of the program. The panel is scrollable therefore there can be a vast quantity of data to be gathered and shown, this data shows each objects positioning while the program is running.</p>

OOP design

A simple layout was created to help with the design process which is shown with the UML diagrams, along the process of coding more classes with several methods and attributes were added into the conclusion. As multiple functionalities were necessary to complete the project, dividing problems into smaller sub problems by creating classes, this was the optimal solution. This can be seen as “DroneGUI” has several relationships which aid in splitting up unique functionalities therefore the code can be processed and edited alternatively. The “FileSystem” was tackled separately as each function/method needed a secure environment to be controlled in therefore a class was created so no other variable can affect this specific class. The file system includes methods such as save, load and directory manipulation, this is all needed when handling files for the program to run smoothly without any errors. Similarly, the “Arena” class also benefits from being a separate class as it holds lots of data specifically for “Objects” and how to manipulate that data within the canvas.

Object is a super class which has many methods and attributes shared to its child classes which are Drone, Blocker and Enemy classes, these benefit from this relationship as inheritance as well as abstraction takes place within this link. Inheritance is shown through the arrow connecting the classes in the UML diagram as well as the protected values and methods which are being used between the child classes. Abstraction is used when those protected methods are being changed within each child class as each class may add a unique twist to the effect of the method changing certain properties, such as Drone using the “adjustObject” method to change its position around in the canvas, also another operation of OOP is encapsulation as all this data is protected and private therefore classes outside of this relationship need to obtain the data through getters and setters.



Theme

The theme for this project is war simulation where the user can create scenarios which benefit the enemy or the drones, as obstacles can be placed throughout the canvas creating intricate mazes which force the drones to move in a constricted environment, whereas the enemy is trailing and capturing the “fleeing” drones.

Improvements/extensions

Level editor was inspired by paint and Mario maker, as the tools being the objects can be used within the canvas with the mouse, by selecting the right tool the user can draw on the canvas enabling them to be creative in their decision for creating unique scenarios. This widens the experience for this simulator as intricate designs are easily accomplished and experimented on when the animation is running. This was achieved with a unique menu decision this was a choice box which allows the user to select one tool from a list containing several, this restricts the user from breaking the program by using multiple tools at once which functionally is unreasonable right.

Enemy Drones needed improvements as the movement was unfinished this was later tackled with vector2D movements as the enemy drone will need to chase the closest drone in its vicinity the enemy will constantly check if there is a drone in its “zoneRadius” otherwise the enemy will continue its path. However, if a drone is close the enemy will lock coordinates of that drone and chase it until “consumed”, this is accomplished by obtaining the x and y position of the drone and the enemy drone will follow a linear path towards the drone.

Sliders will be included so the user is able to change the object’s attributes in real time, this is accomplished with the set values located on the slider this goes from 0-50 in increments of 5 for radius, speed will go from 1-10 with increments of 1. This is also a unique aspect as each drone can be specifically customised while the animation is still running allowing for more versatility in the environment thus unlocking a plethora of combinations and expectations. Eventually these actions will be linked with a mouse listener as each drone can be accessed through a click of the mouse instead of searching through the array list of data.

Background was used to add more unique aspects into the program as the program was simplistic an image was used to spruce up the environment and to alleviate the minimalistic design. This was accomplished by adding an image to the border pane and using javafx’s library which included a “Background” class this in turn processed the image and retrieved from the system to be used within the program.

Conclusions

The program allows the creation of unique arena designs which all perform differently from one another this is all down to the user's creativeness, however this was only due to creating a UML diagram which greatly aided in creating the classes and how to manipulate data efficiently without excessive repetition of code blocks, this was also greatly effective when tackling each subcategory of OOP. By breaking down larger problems into more manageable ones the program was much easier to develop.