CS 319
Object-Oriented Software Engineering

Final Report
World War 3

Group 1-F

Nurefşan Müsevitoğlu

Hüseyin Taşkesen

Halil İbrahim Çavdar

Doğacan Kaynak

1. Implementation:

Initial steps of the game were taken before the second iteration of our game. However, we were able to finish a playable prototype before the second iteration.

As second iteration started, as a group we immediately started to improve our lacking parts in our code as we continued producing reports. We used IntelliJ IDE for developing our game for several reasons, first of which is that it is perfectly compatible with GitHub, where you can pull and push in the IDE environment which was a big plus because of the group project requiring us to act as one. This enables writing code together, as one finished their part and pushed the other could pull and continue on their part with ease.

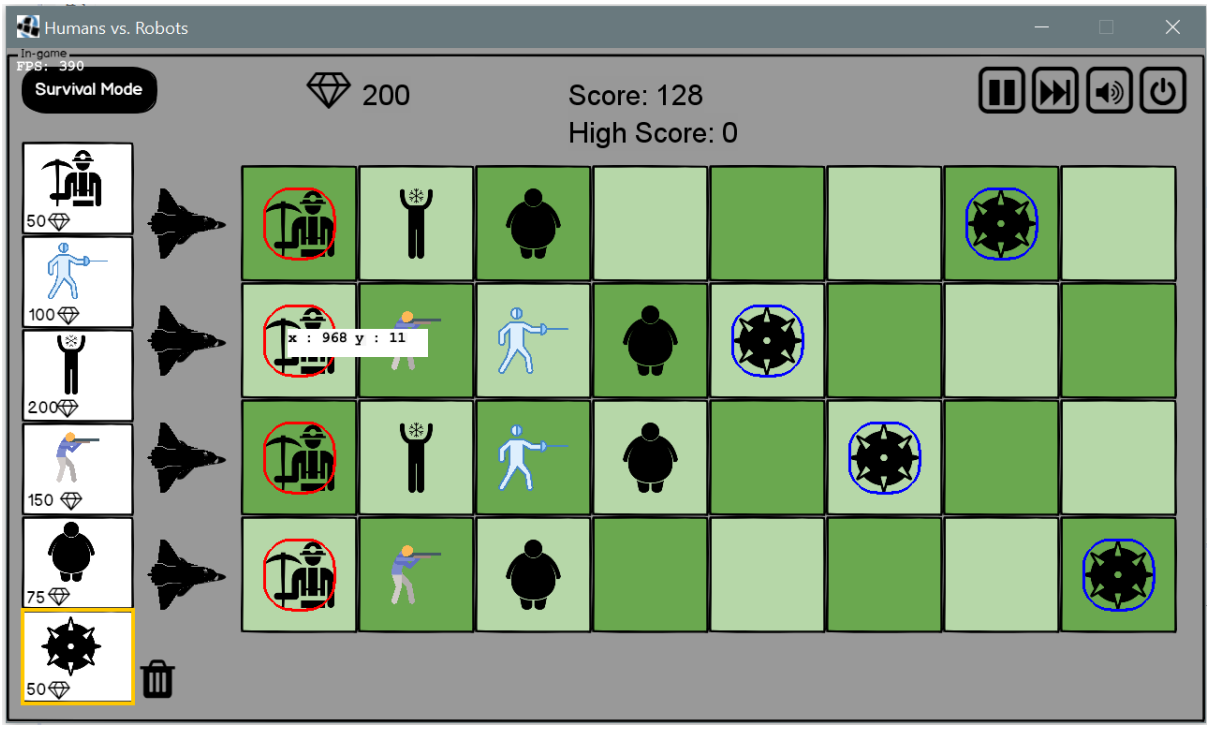
We liked to separate our work as a group because it was easier to handle the complexity that way. Since our design work of our game was very extensive and good, it was not a problem to completely divide the classes because we all knew initially how classes interact with each other and what input or outputs should be. Of course there were small problems with these but they were fixed quickly with the help of teamwork.

We had group meetings and phone calls to achieve continuity and working as a team. Detailed information of whom did which part extensively must be gathered by looking at GitHub contributions.

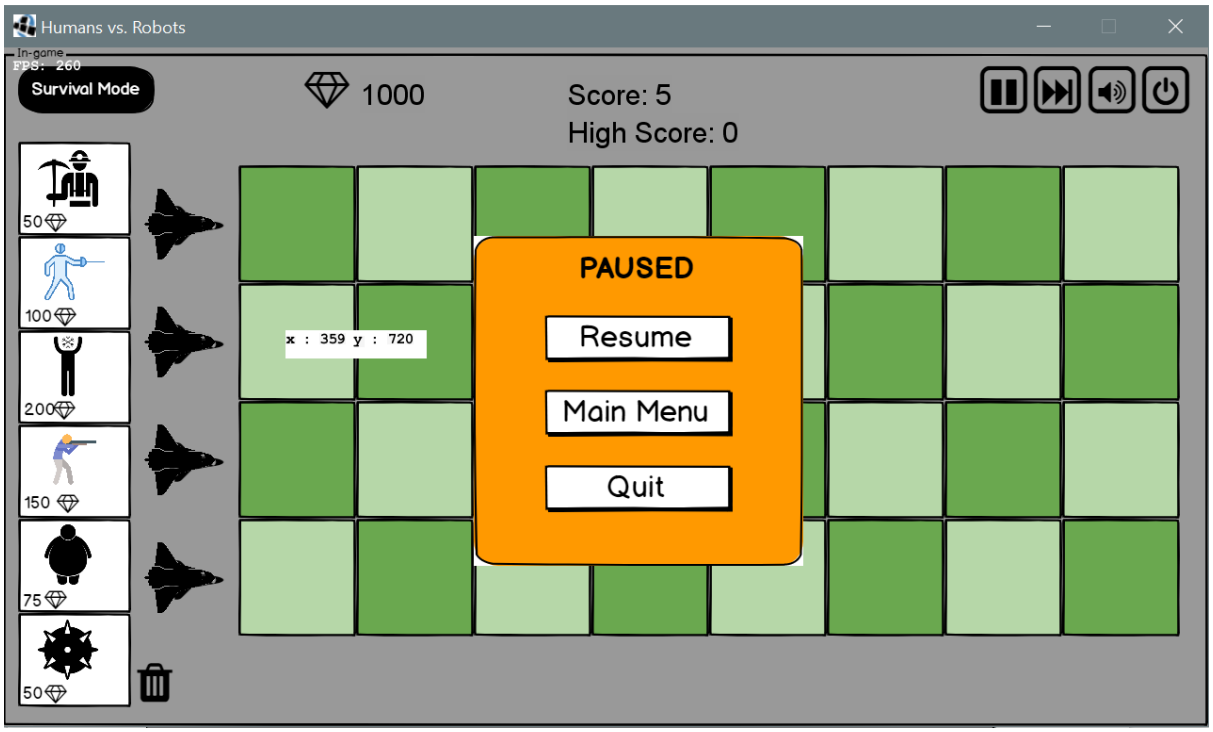
2. Changes in UI

Our game is very similar to what we provide in previous reports. What is changed in UI of our game are map, settings screen, credits screen and representation of humans.

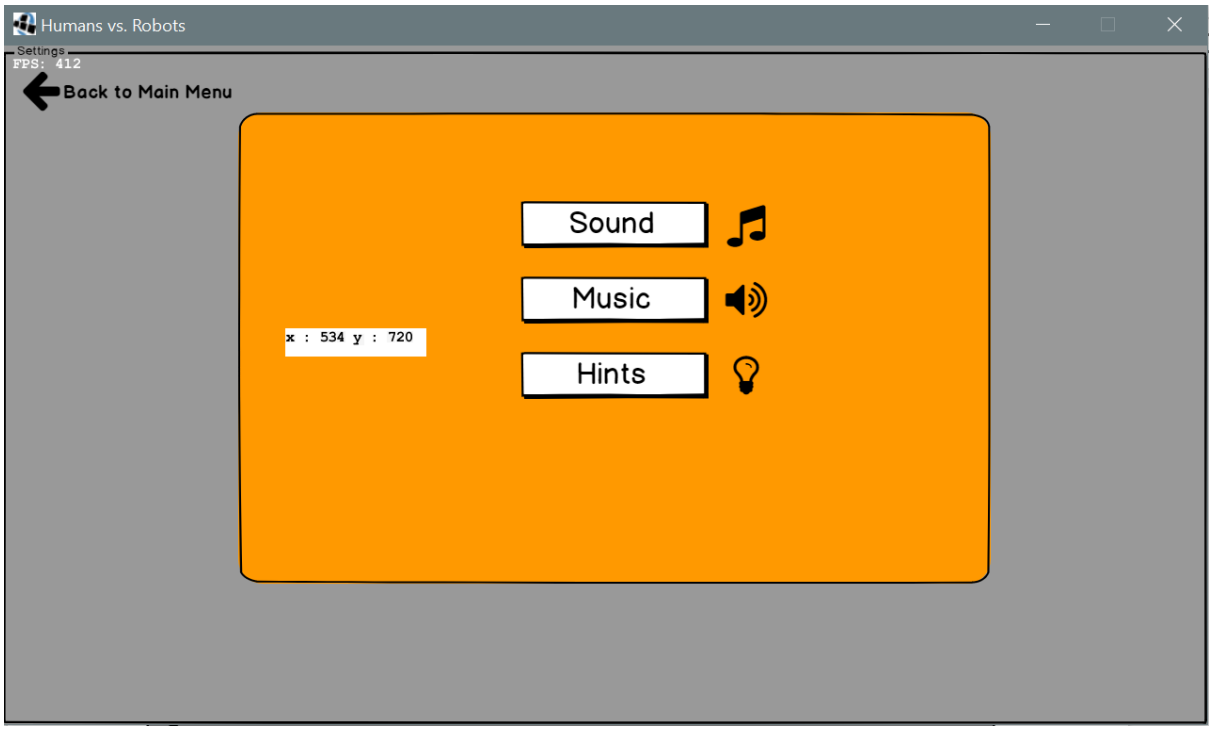
After the player pushes the play game button the game play screen will appears. In our game play screen there are human objects to place which will be used to defense the the home of humans. User can pause the game, turn on or off the background sound and quit to the main menu.



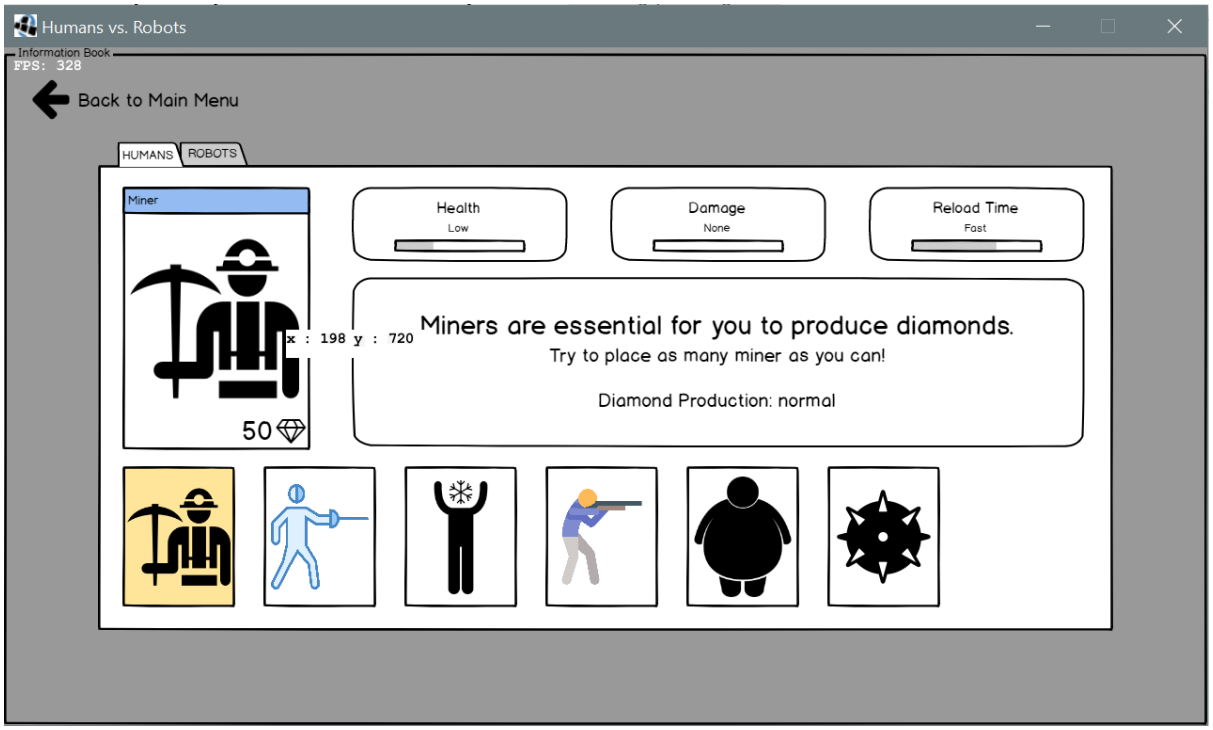
When the player push the pause button pause menu will pop up as shown in the below screenshot. User can resume the game, go back to the main menu or quit from the game.



In settings screen, user can turn on or off the sound or music, and he/she is also able to go to the hints screen.



In Hints screen, there is useful information about the game objects about their health values, damage valus, their reload time and their special skills. It is stronglyl believed that these information will be very useful for the palyer to come up with a good strategy while defending the home.



3. Changes in Design

In the first iteration, our systems were not properly organized. Besides, we were not be able to follow any kind of design pattern due to lack of time. We hurried up and wrote down bunch of code in a single file. In the second iteration we revisited our design in the design report.

First, we have added some model classes to clearly differentiate the classes with respect to their behaviours. To name these, we have added the classes: RangedAttacker, MeleeAttacker and Passive. Ranged attacker class is the super class of human types with a gun, Melee attacker is for melee characters and passive stands for the human objects that do not attack to robots. This way, we made use of inheritance and code reuse between superclasses and subclasses.

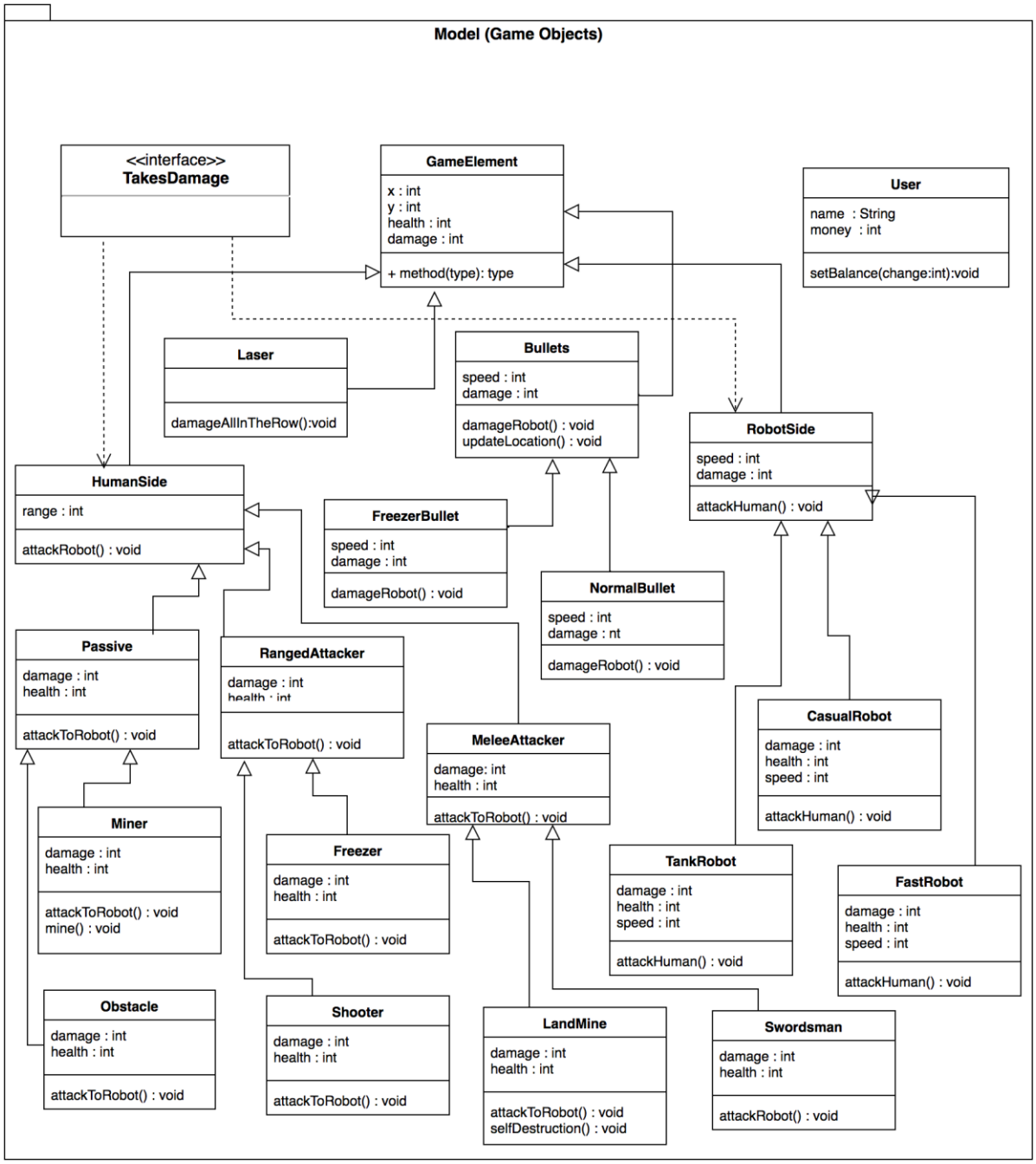


Figure-1 Model classes

Second, we rearrange the classes between packages. Some view related classes were in the controller package due to poor design in the first iteration. We tried to come up with a better system decomposition to support MVC.

Third, we have also followed Façade design pattern for our “manager” classes and we have implemented these classes as Singleton. This way, we put the game logic in a big file and we prevent creating multiple games at the same time. Besides, we could share data between classes as a result of using Singleton.

To sum up, we have made a lot of design changes since first iteration and these changes resulted in a cleaner project overall. These changes provide us the ease of implementation and maintenance of code.

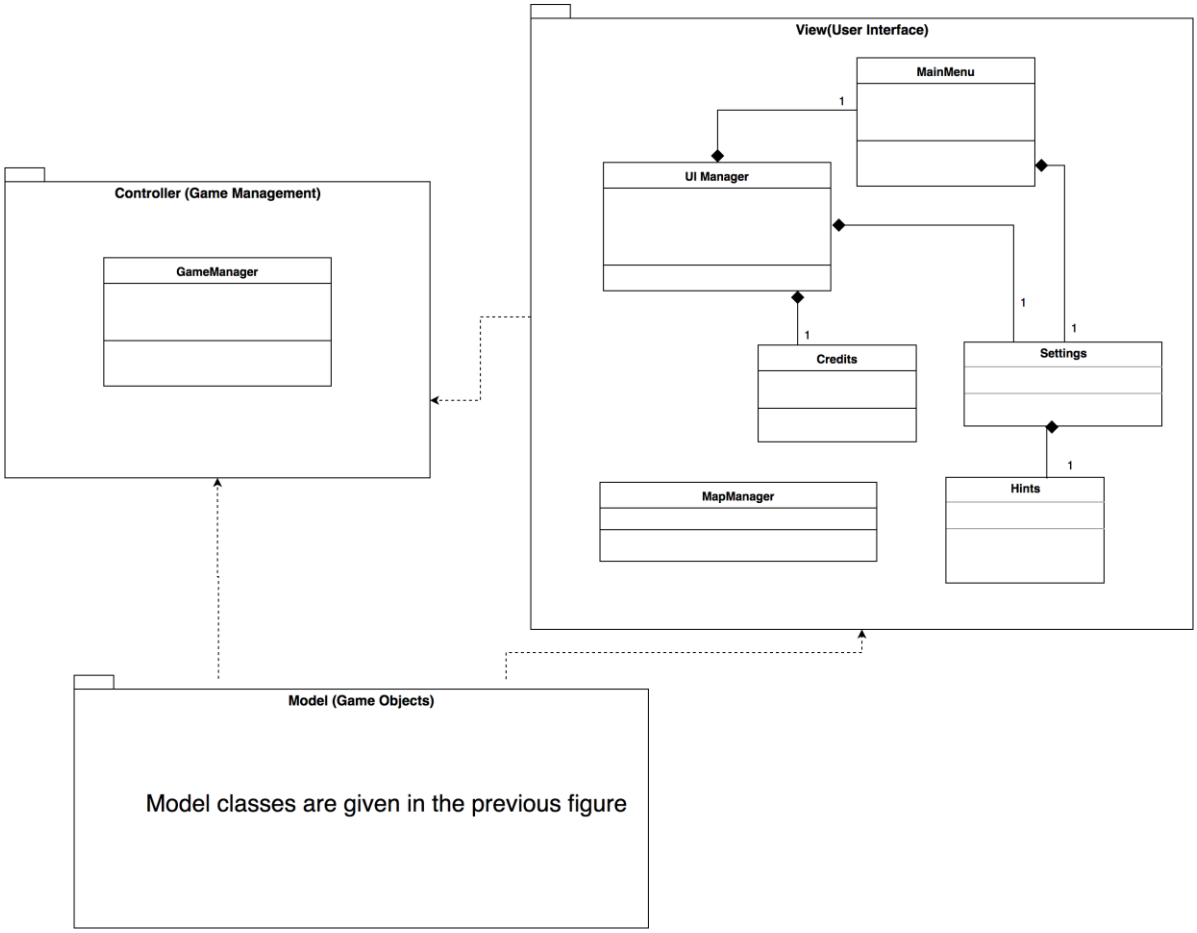


Figure-2 High level system parts

4. Lessons Learnt

We have learned that following some design patterns will be an obligatory when the project becomes larger. Because, when project becomes larger, code becomes more complex and goes out of control easily. Moreover, we have learned some of the advantages of the waterfall approach. Analyzing the domain and thinking about the design before going into implementation is important. It diminishes the number of problems faced during the implementation and helps develop a better quality application.

In our project we have faced more problems in the first iteration where we were writing all of the game logic in a single method. Problem solving and debugging was much more harder as we did not follow design patterns. Besides, the diagrams we have drawn in analysis and design reports facilitate implementing the project. We have looked up to them whenever needed.

5. What is Missing

In previous reports we have said that there will be two different kinds of playing of this game, survival mode or level mode with 3 different levels. Due to the fact that arranging robots with AI is a very hard issue, there was no time left to construct levels.

Also after these parts we have said that there will be a Fast Forward option. We have not been able to implement it as well.

6. User's Guide

6.1. System Requirements

World War 3 is a Java based game. Therefore, JRE (Java Runtime Environment) must be installed in order to run the game. It can be downloaded from Oracle's website from the following link:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Minimum System Requirements:

- Windows XP
- Pentium2 233 MHz CPU or higher.
- 256 MB of RAM or higher
- Screen resolution: 1280x720*

Recommended System Requirements:

- Windows 10
- Intel i5 2 GHz CPU or higher
- 1 GB of RAM or higher.
- Screen Resolution: 1280x720*

* The screen has to be at least 1280x720 as our game has a windowed, 1280x720 fixed size screen.

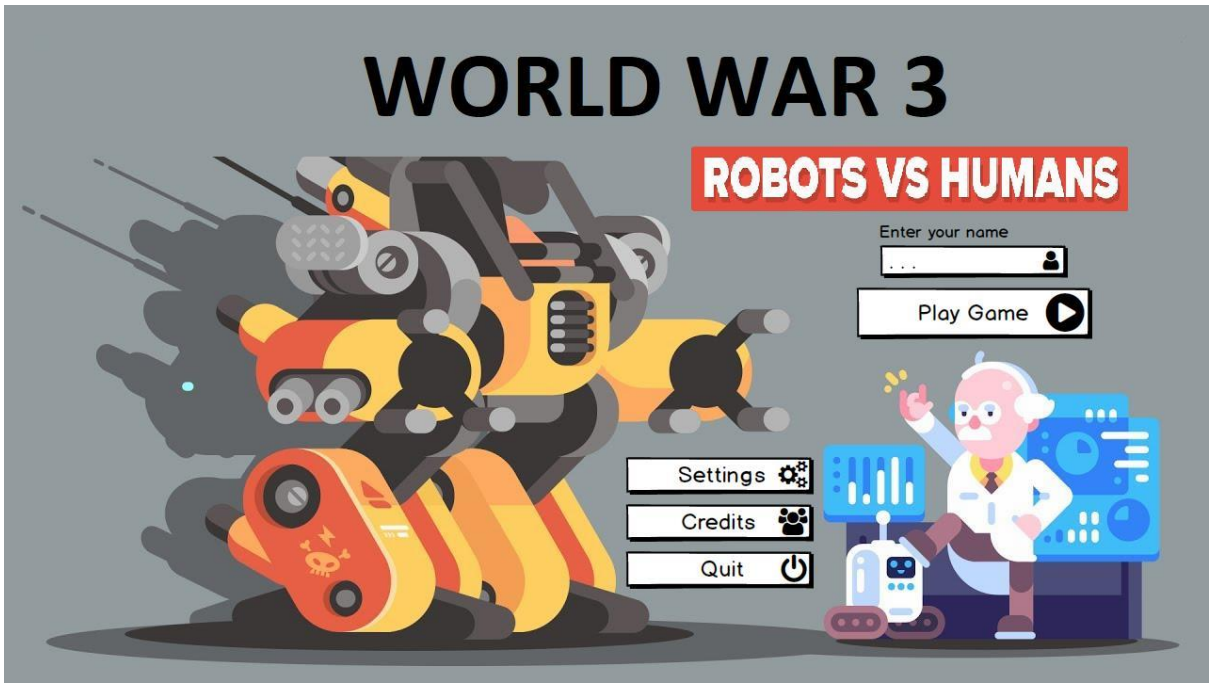
6.2. Installation

Running and compiling our code with a Java IDE is the only way to play our game at the moment. To run it from an IDE, “UIManager.java” should be runned as it is the file that contains our main method. We are planning to make an executable without needing a Java IDE.

6.3. How to Play

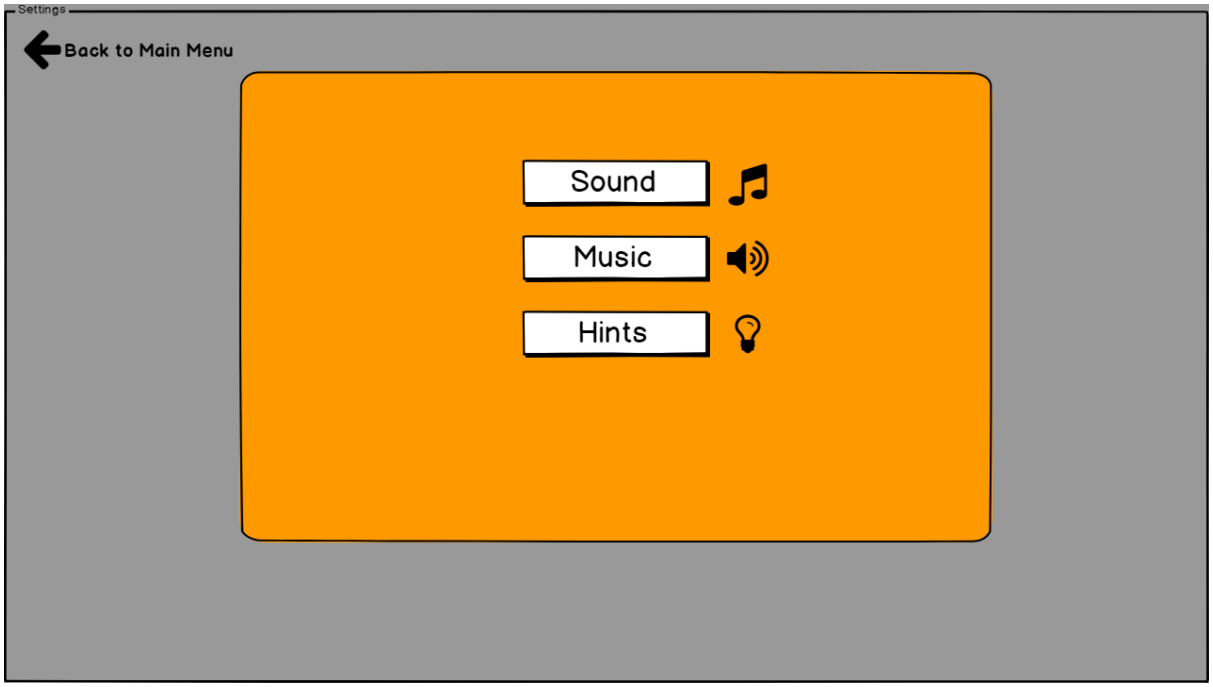
Players can see the instructions of the game from following steps which are detailed for characters;

Inıtially this is the main menu of the game, to reach the instructions of the game hit the settings button:

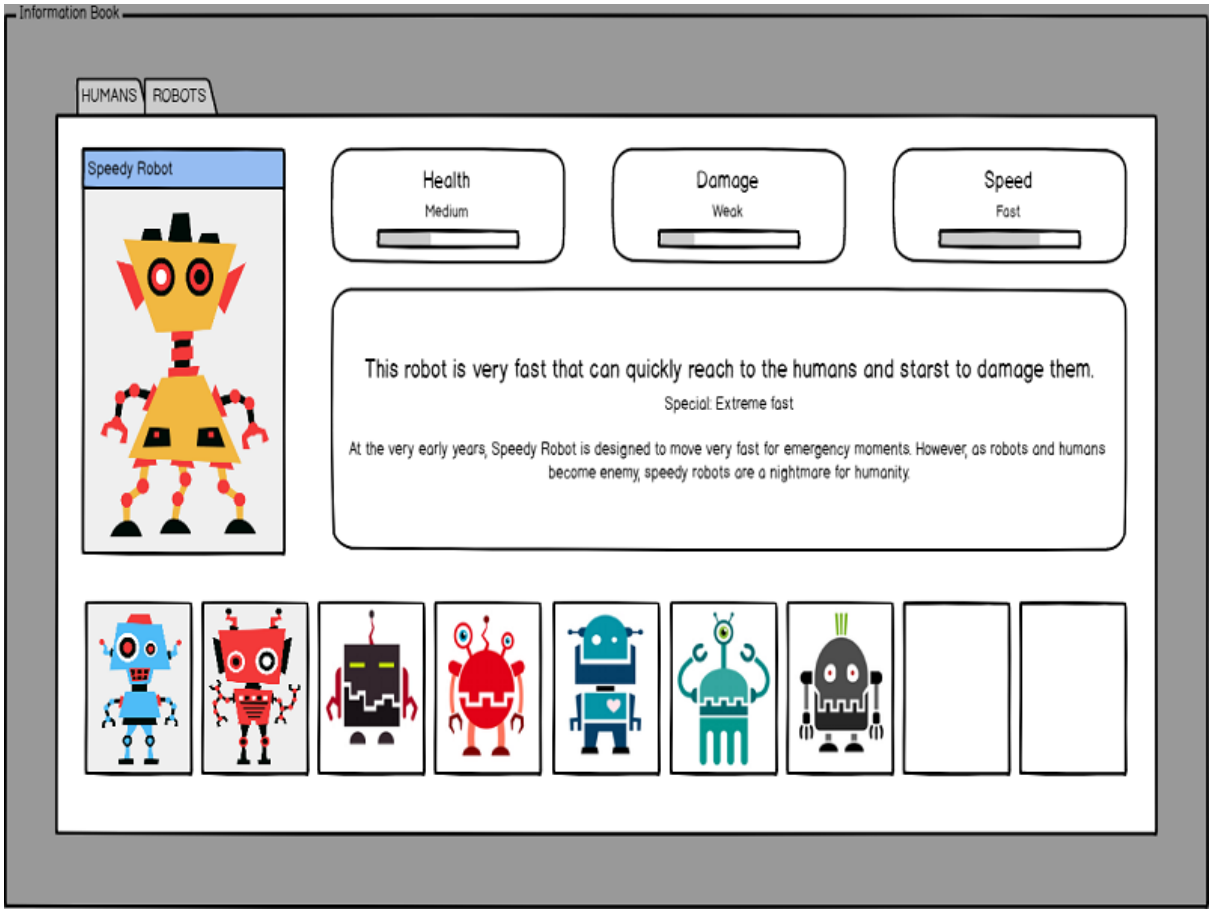


After entering the settings there are 3 buttons which are Sound, Music and Hints. They have functionalities according to their names. Sound and Music are on or off

depending on click. To reach the information about the game, click the Hints button:

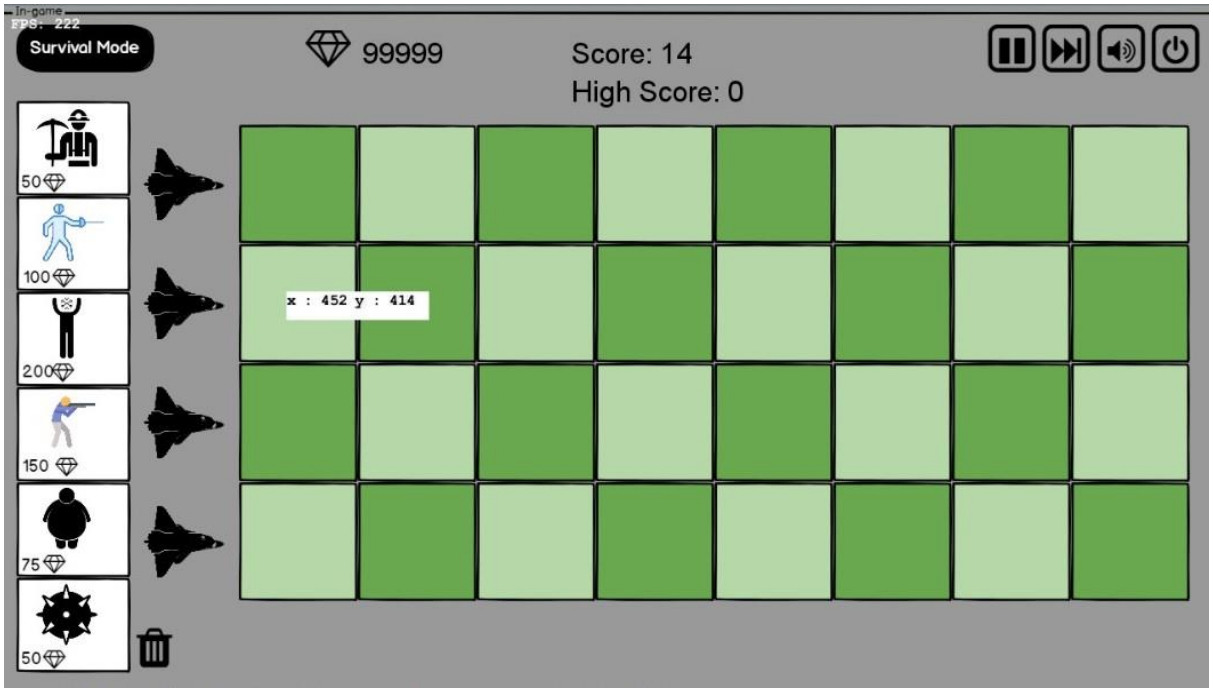


At the end you will reach the guide book which shows the characters of the game you will face with until the end of the game. This screen shows the important abilities of human and robot characters of the game with a brief information about them:



After being informed about characters you can hit the back button on the screens of Hints and Settings screen. Before playing this game don’t forget to write your name on Enter your name section, then click on Play Game button.

- When you start to play the game, you will defend your base by choosing your soldier and place it onto the screen. You will do both selection and placement by pressing the left click of mouse.
- You can also free space by clicking the trash bin icon and then clicking on a slot to remove the human there.
- There are also lasers on the screen that destroys every robot on that row whenever one of the robots steps on them. However, they have limited one-time usage, the next time robot steps onto base the game will be over.
- Depending on every second you survive, your score will increase. You can see the high score under your current score.
- Depending on order, you can stop the game, turn on/off the sound of the game and quit the game.



7. Conclusion:

We think we did a very good job in determining design patterns and obeying them which helped us hugely because it was easier to code and understand each other’s work when talked in the language of UML and after making use of design patterns. We tried to stick to “Principles of Object Oriented Design” as much possible. However, we also see that working with a group of people has both advantages and disadvantages. While splitting the workload is very good, it is very hard to proceed when one or more member fails to do their duty.

As a conclusion, we have tried our best effort to learn from this project and complete our tasks. Also, we were able to achieve many of our promised functional and nonfunctional requirements.