CMPT 370 Requirements Document for Battle Bots

Mackenzie Power

Haotian Ma

Ryan Tetland

Will Revell

Mitchel Kovacs

**Overview:**

This document is the requirements contract between the student software engineers Mackenzie, Mitch, Will, Haotian, Ryan and our client Chris Dutchyn. We will be developing a robot battle game for our client hereby referred to as Battle Bots. Everything contained within this document is our promise of what we will have in the finished program.

The finished program is called BattleBots, BattleBots will allow users to play against other players, computer controlled robots, or run a simulation of a robotic battle between different robot teams. When the user chooses to play against other players or computers they will be able to visually see a board where they will take turns with their opponents. Each play will involve a series of rounds where each player will be given the ability to move a specific robot and attack anything within that robots’ range. All these movements are limited by the characteristics of the specific robot. As the game goes on, players will be eliminated when they have no robot to make a move. The game ends when only one player has robots remaining. All robots will have their win and loss stats updated based on the outcome of the game. If the user chooses run a game with no human players then a simulation is run instead, showing the result of the simulation and an updated version of each robot team’s stats.

This document covers the main ideas of how the game is going to operate and a detailed description of each of its functions and structures. We will prioritize what is necessary in order for the game to run properly, and also discuss what features we could add to make the game look better and more fun to play. We have included use case diagrams, sequence diagrams and prototype GUI interfaces to help the user gain a better understanding of the game.

BattleBots will be designed on Linux based system, and will be executable using remote access through tuxworld. All team collaboration will be done through a GIT repository accessible to the client. Most major coding will be done in JAVA using eclipse and NetBeans, while AI code will be written in Forth, and will be utilized in JAVA using JSON. The use case diagrams are created in Microsoft PowerPoint and the sequence diagrams are created using LibreOffice Draw. All formal documents will be created using Kate and Microsoft Word.

Must Haves

This section will list the functionality that our game will need in order for it to work properly. It will not include all of the features that we would like to implement, which is outlined in the following section, but rather the features that will be required to allow the game to be playable.

The first user interface must have the following functionality:

-The user must be able to click on a button labelled "Help" which would display a screen that explains all of the rules of the game and comments for troubleshooting. From there, the user should be able to click a “Back” button to return to the previous screen.

-The user must have an option to exit the game at any point in time while the program is running. There will be a button labelled "Quit" which, when pressed, will terminate the program.

Game-play setup screen

-The user will also have the option to select to select the number of players they would like to participate in their game. They will have three options: two players, three players, or six players.

-If three players is chosen, the user will then have to decide whether to play on a board with five spaces on a side or seven spaces on a side.

-The user then must choose how many of the players in the game will be humans, which will determine.

-Once the user has selected these things, they will then have to click a button labelled "Proceed", which will then take them to the game-play screen.

Game-play screen

-Now that the game-play screen is ready to be displayed, the program must be able to display the correct size of game board. There will be five spaces on each side for two players, and seven on each side with six players. If the user has selected three players, they have also chosen the board size to be either five or seven spaces on each side. The program must also randomly assign each player in the game to a colour.

- The program must be able to identify whose team's turn it is, and which robot they should be using for their turn. It must also identify if it is a human team or a computer team.

-If there are no human players in the game, the program will simulate the game behind the scenes and display the end of game screen. If this is taking too long the user has the option to terminate the game. The user must be able to end the game because the game may take a long time to simulate, and may potentially never end.

-If there are humans in the game, then the user will be able to see the entire board. Once their turn is over, they can observe the entire board and where the other players have moved to. If there is more than one human in the game, then they can both see the entire board.

-If it is a human team's turn, they must be able to move the appropriate robot, or fire at a space. They must also be able to move, fire, and then move again if they still have any moves left. These options will be determined by the user supplying input to the keyboard and mouse.

-For teams that are being operated by the computer, the program must have an interpreter to take the moves determined by a certain AI code, and translate that into the computers actions in the game.

-During the game, there will be a section on the side that will display the current statistics of the game. This will include each teams robots that remain, along with how much health they have left. For this to remain accurate, these statistics must be updated every time a robot shoots.

-Once the game is over, a screen will be displayed that shows the final statistics of the game. This means that statistics must be kept track of for each team. This will include how much damage they dealt out, how much damage they received, and how many spaces they moved. After this it will update these statistics to the Robot Librarian.

Future Versions

Due to time constraints we will not be able to include all of the components that we would like to. We have come up with a few ideas that we believe would improve the game, but they are not detrimental to the functionality of the game. We are not promising that these features will be added, they are just some of our ideas that could be added, given more time. The following is a few aspects that could be added in future versions.

- Animation

When a robot is moving to a new position on the map it looks as though it is driving there, instead of just disappearing and reappearing from space to space. Also when a robot is firing at a space on the map, it would show a missile flying through the air to the spot, and an explosion would be shown on collision.

-Sound Effects

It would be nice to add sound effects to our game. This would possibly include sounds for firing missiles, explosions, and driving robots. We could also have some sort of intense background war music playing.

- More advanced AI

Based on testing game-play, we would observe the actions of the CPU's strategy, and devise new strategies that would make the game more difficult for humans to play against. Perhaps, as an example, the CPU's could detect if another player is doing well, and teams up with other CPU's in the game to even things out. We would make an option in the game setup interface to choose a desired difficulty.

- Networking

As of now, our game is designed to be played on a single computer. In a future version, the game would be able to be played on different computers for different human players. Each players screen would only display the area that they are allowed see, and would not be allowed to do anything until it is their turn to move. This would be more desirable for multi-player use.

-Choice of team

It would be nice to add a feature that allows the user to choose their team that they would be playing with.

- Feedback based on user testing

Once we have a working version of our game, we would test it with some people and use their feedback to make improvements based on their opinions.

Walkthrough:

Upon execution of the game, a user friendly screen will be brought up. This screen will display a picture of the game and give the user three options to choose from. The first option to select from will be a ‘rules’ button that will bring the user to a new window and display all the rules of how to play the game. There will be pictures to help show the rules as well. There will be a ‘back’ button to return the user to the initial window shown at execution. The second option will be to quit the game, which will shut the game down. The third option will be a ‘begin’ button that will proceed with the game. Selecting this option will bring the user to a new window which will ask how many players the user wants the new game to involve. The options to choose from will be two, three or six players. The same window will then allow the user to choose how many of the players will be controlled by humans, and then have the other players controlled by computers. Once this is done, the game will initiate with all the users’ chosen parameters.

**Actors:**

Actors:

For the project, there will be four main actors (or roles) that will be employed within this program:

User

The first actor in the system will be the actual user, the individual that will be executing the program and dealing with the various user interfaces. This actor will be able view the rules of the game at any time; terminate the program at any time; choose the number of players; choose the number of human players; also choose the size of the board; view statistics after the game; and play the game again. The only limitations for these actions is the player will only be able to choose 2, 3, and 6 players for the game; and that the size of the board is only an option to choose from if there are 3 players.

Human Player(s)

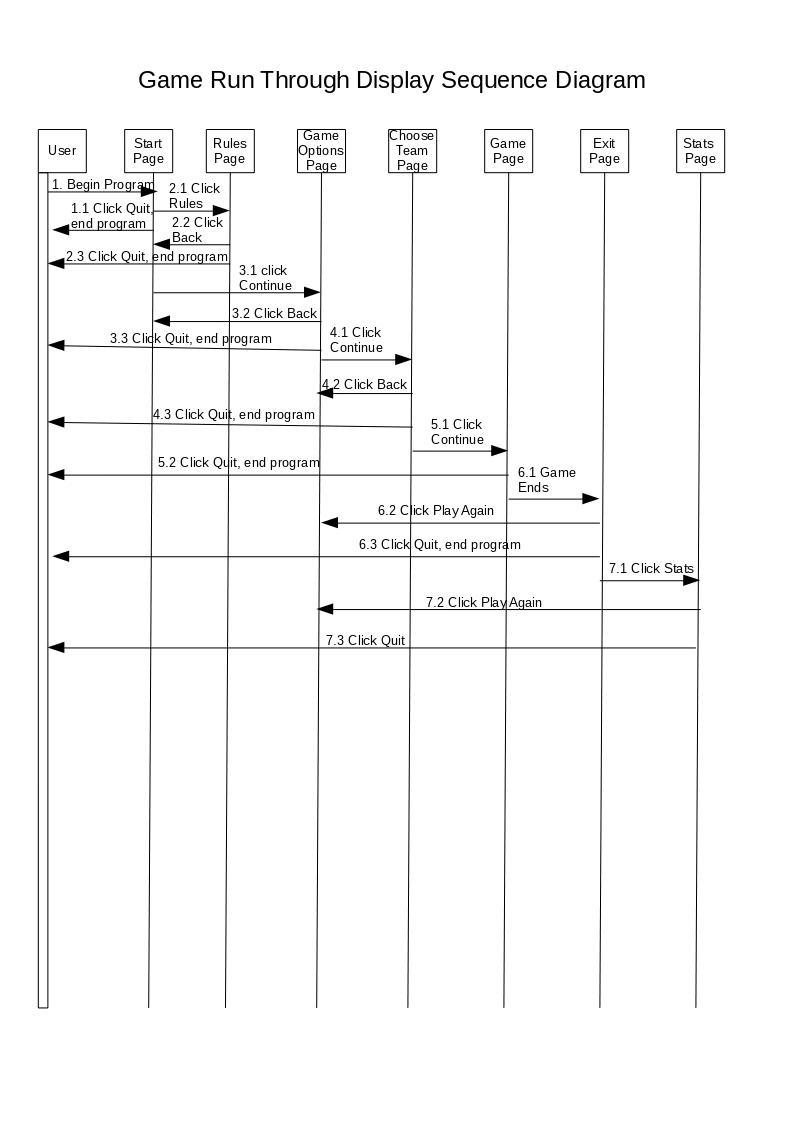
The next actor in the system will be the human players involved in playing the game. Note: any individual who is fulfilling the user role can also assume the role of a human player. Human players have the ability to view the rules section, move a robot, attack a robot, and terminate the game.

Robot Player(s)

Like the human players, the robot players will have a similar role. They will have the option to move (if it is their turn) and attack another robot.

Robot Librarian

The last actor in the system will be the robot librarian whose role will be to query various robot teams and their stats for the user to select from; download the robot AIs for use; register new robots; retire current robots (clears their stats); revise current robots; and finally keep and update the statistics of each robot it has access to after a game.



1. The game initiation sequence diagram describes a play through of a single match of Battlebots, from the various game options, through board initiation until a winner is decided and the end game stats shown. It does not include the numerous possible robot actions during a match.

2. The Robot Turn Sequence Diagram describes in detail the different action paths an AI controlled robot can take from the beginning of their turn until the end.

3. The Game Run Through Display Sequence Diagram shows in detail the different paths a user can take through the games GUI interfaces. Beginning at the Start page and ending at the post-game stats page.

