

Requirements Document

CMPT 370

**Group C4**

Jack Huang

Brandon Jamieson

Ixabat Lahiji

Daniel Morris

Kevin Noonan

## **Table of Contents**

Page 3: **Executive Statement**

Page 4: **Game Rules**

Page 5: **Programming Task**

Page 5 – 10: **Visual Requirements**

Page 11: **System Diagram**

Page 11 – 14: **Actors and Actions**

Page 15 – 25: **Scenarios**

Page 26: **Hardware, Software, Data Formats and the Team**

## **Executive Statement**

The purpose of this project is to develop a multiplayer board game in Java. The board game will require players or AIs to take turns moving and attack with a team of three robots. Once there are only one player's robots remaining, that team is declared the winner. The system must support one or two human players, as well as up to six AI-controlled players. The board itself is hexagon-based and must have functionality for choosing the number of players. As well, the user must also be able to select a board size of five or seven if there are four players. If there are no human players in game, the user should be able to spectate the AI's gameplay.

The user-interface will be split into two parts, the menu and in-game. The menu must have functionality for starting a game with the user-chosen player number and board size. The menu must also have functionality for downloading and updating robots, as well as configuring robot teams. On top of this, the menu must also contain an option for exiting the application.

In-game, the UI must have an indicator of the current player's turn. As well, it must have indicators for the current player's team of robots and their individual statuses (health, range, movement). In-game, there must also be a button for attacking and moving. It should have an indicator telling the user they are playing, waiting, or spectating.

We have narrowed down the actors for our system to five: human players, AIs, spectators, the menu navigator and the robot librarian. We have identified thirteen primary scenarios that our actors will encounter within the game. For example, the menu navigator has a scenario involving the creation of a game. The navigator will select a player amount, as well as a board size based on it.

The system will be programmed entirely in Java, making use of the AspectJ extension. As per assignment requirements, the system must compile and run on the U of S' *tuxworld*. The system will make use of the JSON file format for downloading and updating robots and their statistics.

This requirement document marks the completion of the first phase in our software-engineering process, and is the first of four deliverables our team will be submitting.

## Game Rules

The game itself can be played with up to 6 people. The game board is composed of hexagons and will have an edge size of five or seven depending on the number of players. Games of two or three players will have a board size of five hexagons per edge, four players will have edge size of five or seven, and six players will have seven.

Each player will represent a colored team and will have three robots on their team, all of which will begin the game in the team's respective corner. Gameplay is divided into turns and rounds; each turn will consist of up to three rounds depending on the amount of tanks in play. On the first round, each team plays their robot that has the highest speed stat. On the second round, their second fastest and on the third, the slowest robot. If a team does not have a robot to play during a round, they will be skipped. Once all rounds are complete, a new turn begins.

Plays can be any combination of movement and shooting as long as they have movement points available. Moving costs one point and will move the robot one hexagon in the specified direction. Shooting also costs one movement point and deals the robot's damage to **all** robots occupying the target hexagon. Robots cannot fire at any hexagons outside of their respective range. Once a robot is out of movement points, their play is over and the next team's play begins. Each robot's movement points are replenished at the beginning of each turn. If a robot's health reaches zero, the robot is dead and removed from play.

Robots can only see as far as many hexagons as their range stat allows. Players will only be able to see hexagons and their contents if they fall within the range of one of their robots. All other hexagons will be blacked out and any occurrences in those hexagons will not be visible to the player: this is called the fog of war. Fog of war will be updated for respective teams as their robots move around the board.

Once there is only one team with robots remaining in play, the game will be over and the team declared the winner.

## **Programming Task**

The task of this project is to translate this game to a computer-based application. The main rules of the game will remain the same for the most part, with slight changes made to better suit a computer version. For instance, players will not be able to see the entire board as they would in a physical version. Instead, hexagons that fall within the player's range will be visible and all others will be blacked out. These blacked out sections are called the fog of war. Players won't know about any occurrences that take place in the fog of war. This gives the game another level of strategy that a physical version would not have.

One benefit a computer version will have is the AI-controlled opponents. This allows for solo players as well as adding replay value to the game. The game will allow for up to two human players on the same computer. To support this, players will simply take turns making their moves on the same shared screen, one after the other. One disadvantage this brings is the possibility for cheating. Player one could watch player two make their move and play based on it.

Another advantage of our computer-version is the ease of access. Users can choose multiple opponents, leading to different board sizes requiring no set-up time. In a physical version, up to six teams may not be possible and board sized may be fixed. As well, robots will be customizable and keep track of their own individual stats over multiple games. The system will also support updating, adding, and removing robots. For a user to do this with a physical game would require a lot of extra book-keeping and attention, while our system does it all with ease.

Being computer-based also allows us to have a spectator mode if the player is defeated by AI, or it is only AIs playing. This mode will allow the player to fast forward through computer moves, see the statuses of their robots, and more. Having this mode in place will give users the ability to run games of only AIs and watch as they play the game out. It also allows games to finish completely if the human is defeated but there are AIs remaining, the human can watch the remainder of the game at their own pace.

## **Visual Requirements**

This project will involve a decent amount of visual requirements. They are composed of two main screens and one selection box on the menu screen. The screens are:

1. Menu Screen
2. Play and Options Selection Screen
3. In-game Screen

## **Menu Screen**

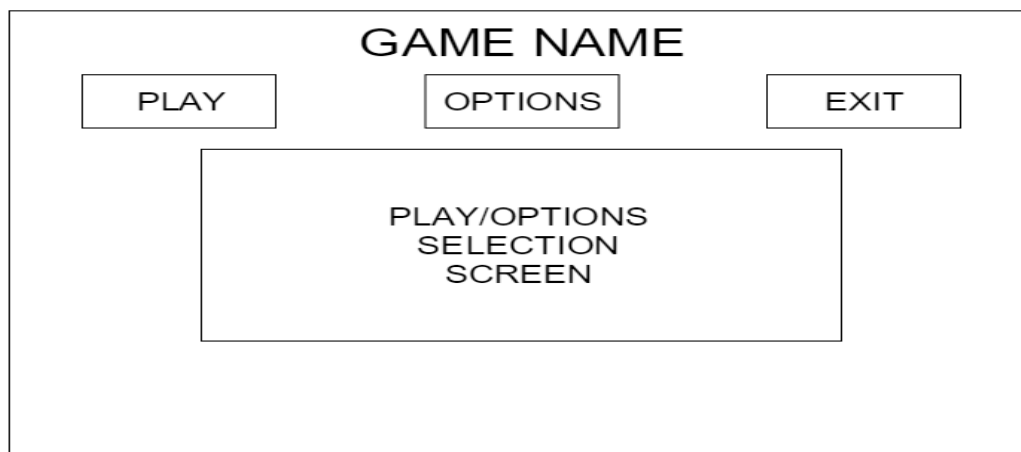
When the user launches the program the menu screen is presented. The menu screen is comprised of three options for the user to select at the top of the screen. A larger selection screen is below the options boxes where the user will select the game options when either “Play” or “Options” are pressed.

### **1. Play**

The “Play” selection box will bring the user to the larger selection box. Inside this box the user has the options to select the number of players in the game, whether the player is an AI or a Human, the board size (between 5 and 7), the colour of the player’s team and the name of the player. Once the user has selected the options there is a “Begin” button at the bottom of the selection box where the user may start the game. Clicking the “Begin” button will bring the user to the in-game screen.

### **2. Options**

The “Options” selection box will bring the user to the larger selection box. Inside the box is the Robot Library where the user can import and delete the team of robots they would like to use in-game. There are three buttons provided for the user and a (originally empty) list. The buttons are “Import from Server”, “Import from Local” and “Delete”. Clicking on the “Import from Server” button brings the user to the server page where they can select a robot team. Clicking on the “Import from Local” button will bring the user to their file directory where they can choose a robot team saved locally. Lastly the user has the option to click on already imported robot team in the scroll list and press the delete button to remove that team from the list.



### 3. Exit



The final selection box is the “Exit” box which closes the program.

## Play and Options Selection Screen

### 1. Play Selection Box Layout

The play selection box will appear once the menu navigator presses the play button.

At the top of the box will be a list of designated lines for the user to input the team names. Directly to the left of the team’s name will be a small circle to designate the team color of that team. To the right of the player’s name will be the option to declare the player’s team either a “Human” or “AI”. To the right of the list will be two images of a size five hexagonal board and a size seven hexagonal board. The boards will be highlighted based on the player size that the user chooses. Two players will highlight the size five board. Three players will highlight both boards and six players will highlight the size seven board. At the bottom of the box will be the “Begin” button to start the game.

Color	Team Name	Number of Players	Board Select
<input type="radio"/>	<input type="text"/>	<input type="text" value="Human"/>	
<input type="radio"/>	<input type="text"/>	<input type="text" value="AI"/>	
<input type="radio"/>	<input type="text"/>	<input type="text" value="AI"/>	
<input type="radio"/>	<input type="text"/>	<input type="text" value="AI"/>	
<input type="radio"/>	<input type="text"/>	<input type="text" value="AI"/>	
<input type="radio"/>	<input type="text"/>	<input type="text" value="AI"/>	
<input type="button" value="BEGIN"/>			

## 2. Options Selection Box Layout

The options selection box will appear once the menu navigator presses the options button.

The user will be able to view and modify robot teams (revise, update stats, etc.), as well as import new robot teams from local JSON files or from a server. They will also be able to delete existing robot teams.

The diagram shows a rectangular container for the 'Robot Library' options. At the top left is the title 'Robot Library'. To its right are three buttons: 'Import from Server', 'Import from Local', and 'Delete'. Below the title and buttons is a vertical list of seven input fields. The first input field is labeled 'RobotTeam Name' on its left side. The other six input fields are unlabeled.

## In-Game Screen Elements

### 1. Hexagonal Playing Field

This is the main playing field located in the centre of the screen. If the user decided on a AI only game the entire field is visible to the user and they are able to see the decisions made by the AI in spectator mode. If the user is playing in the game though they can only see the part of the field that their pieces allow them to see the rest is covered in a fog of war. If a player runs out of robots, they go to spectator mode and can see the rest of the field. Each side of the field is separated into six different colours displaying the different spots where each team starts.

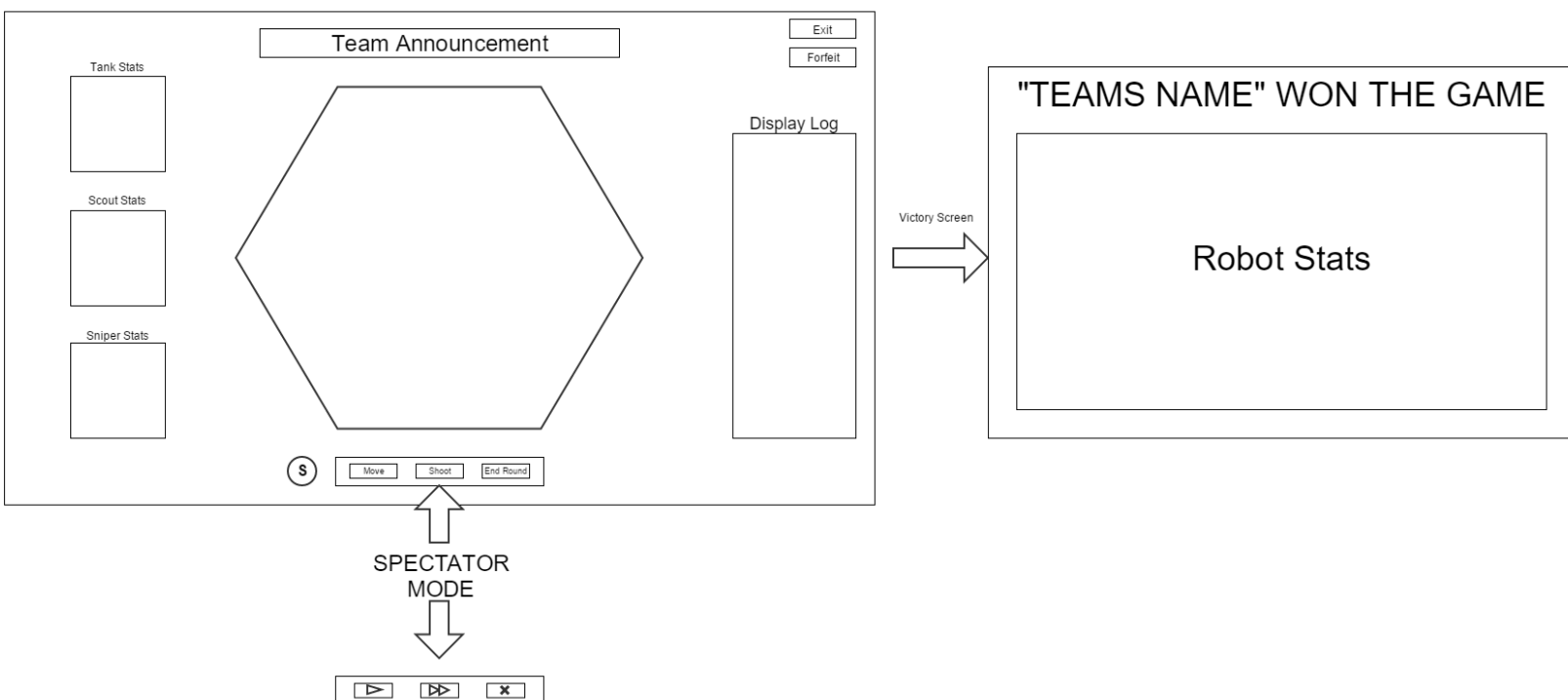


## 2. Announcer Box

The announcer box is used to display which team's turn it is. This horizontal box is located at the top of the screen. This box displays the team's name and is displayed in the colour of the team.

## 3. Display Log

The display log shows the health bar, the damage given of the players pieces that are shown on the board. If the user is playing, the display box will only show the information of the pieces that are outside of the fog of war. If the user is not playing and the only players are AIs, then the display box will show the information of all the players pieces on the board separated between the teams. This box will be a vertical rectangular shaped box to the right of the hexagonal playing field.



## 4. Forfeit Box

If the user would like to end the game early, there is a forfeit box located in the top-right of the screen. Pressing this box with more than one human player left in the game will simply remove the user's player from the game. If they are the last human player left then the user is moved into spectator mode.

## **5. Exit Box**

The “Exit” selection box is located directly below the “Forfeit” selection box. Pressing this button quits the in-game screen and brings the user to the main menu screen.

## **6. Turn Box**

The turn box is located below the playing field. Within the box are three buttons. “Move”, “Shoot” and “End Turn”. When either “Move” or “Shoot” is pressed the option is given to the user to click on a piece on the playing field and execute that given command. If “End Turn” is pressed it simply ends the player's turn and the round goes to the next player.

## **7. Robot Display Log**

The robot display logs are located to the left of the hexagonal playing field. There is one log box for each of the player's robots. These boxes display the health of the robot, damage that robot has given to other player's robots, range stats, as well as movement points. If that robot is destroyed, the log is blackened out showing that the robot is no longer available.

## **8. Spectator Mode Options**

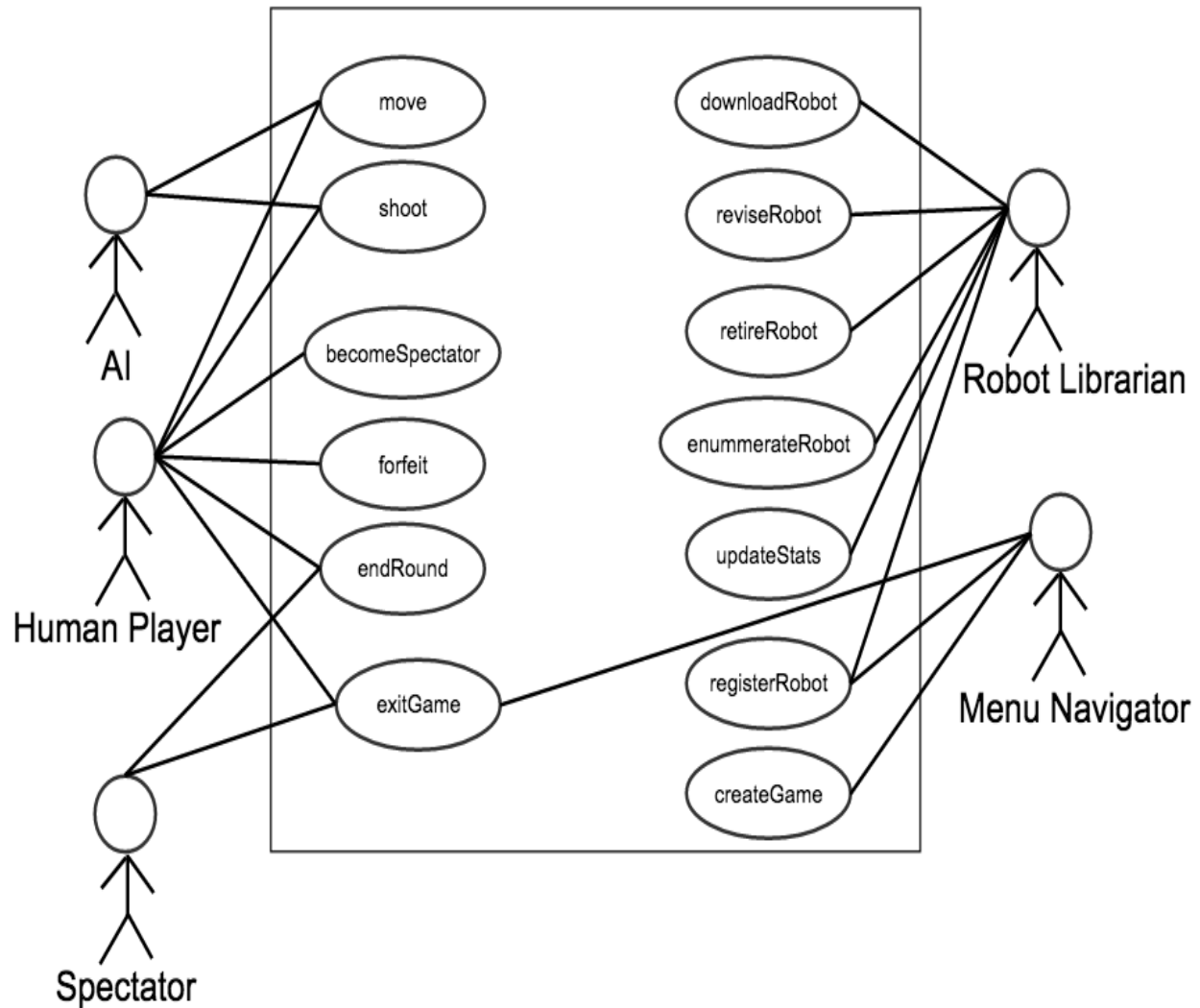
The spectator mode options are displayed in the turn box and are only displayed when the user is in spectator mode. The buttons given to the player are “Pause/Play”, “Fast Forward”, and “End Game”. This gives options to the user to pause the AI's play-through of the game. “Fast Forward” also has the option of times one the speed, times two the speed and times four the speed. End Game will bring the player to the victory box screen. To the left of the turn box is the fog of war toggle that can be used to turn it on and off during spectator mode.

## **9. Victory Box**

The victory box is a display box that only pops up after the game has finished and the player has not exited to the main menu. The box covers most of the screen and is used to display the player that won and the robots stats for each individual team.

While these interfaces are simple mock-ups now, the general look will remain the same, albeit more polished, by the final product. The system will ensure that these functionalities are accessible to their respective actors. Actors are what we view as the primary externals that will be interacting with our system.

## System Diagram



We have identified five actors on our system. Among these actors, thirteen primary actions have been identified. The following points will describe each actor and a brief overview of their respective actions. Scenarios and their use-cases involving these actions will be covered in-depth afterwards.

## **Menu Navigator:**

The menu navigator interacts with the main menu and handles game creation. This actor would also perform operations such as managing robots, game creation, and exiting the game.

### **Actions:**

- Create Game
  - Creates new game instance
    - Navigator will be presented with options to change the number of players and modify the board size.
    - Navigator will also be able to choose which robots are to be used in the match.
- Exit
  - Quits application.

## **Human Player:**

This is the human player in the main game. This actor will perform gameplay tasks on their team of robots, such as moving, turning, or shooting. On top of this, the human player can forfeit the game, move to spectator mode, or exit all-together.

### **Actions:**

- Move
  - Moves robot forward.
- Shoot
  - Applies damage to all robots on a target tile.
- End Round
  - Finds next robot to control on next team.
- End Turn
  - Allows next player to control their robots.
- Forfeit
  - Removes the Human Player from the current match.
    - If no more Human Players are in play, **the option for Spectator mode is given.**
- Exit
  - Quits application.

## **AI:**

The AI serves as the computer player during the main game. Actions performed are similar to the player, though AIs will not be able to forfeit the game, spectate, or exit the application.

### **Actions:**

- Move
  - Moves robot forward.
- Shoot
  - Applies damage to all robots on a target tile.
- End Round
  - Finds next robot to control on next team.
- End Turn
  - Allows next player to control their robots.

## **Spectator:**

The spectator serves as an observer to the main game. While it cannot interact directly with any robots in the game, it has the ability to

An observer to the main game. Cannot interact directly with any of the robots in the game. Has the ability to go to the end of the computer turns after they have completed it.

### **Actions:**

- End Round
  - Finds next robot to control on same team.
- End Turn
  - Allows next player to control their robots.
- Exit
  - Quits application.

## **Robot Librarian:**

Manages the robots in the system, as well as pulls data/updates for robots from a server.

### **Actions:**

- Register
  - Registers a new robot to the system from a JSON file.
- Revise
  - Revises an existing robot with new code.
- Retire
  - Freezes a robot, allowing its name to be reused.
- Enumerate
  - Displays all robots currently stored by the system.
    - Sorted by values
    - Show stats and versions
- Download
  - Downloads a robot record for use in-game from a remote server.
- Update
  - Records and updates details about a robot's gameplay results.
    - Details include
      - Number of games survived or destroyed
      - Number of times robot's team won or lost
      - Amount of damage inflicted or taken
      - Number of hexagons travelled
      - Number of shots fired

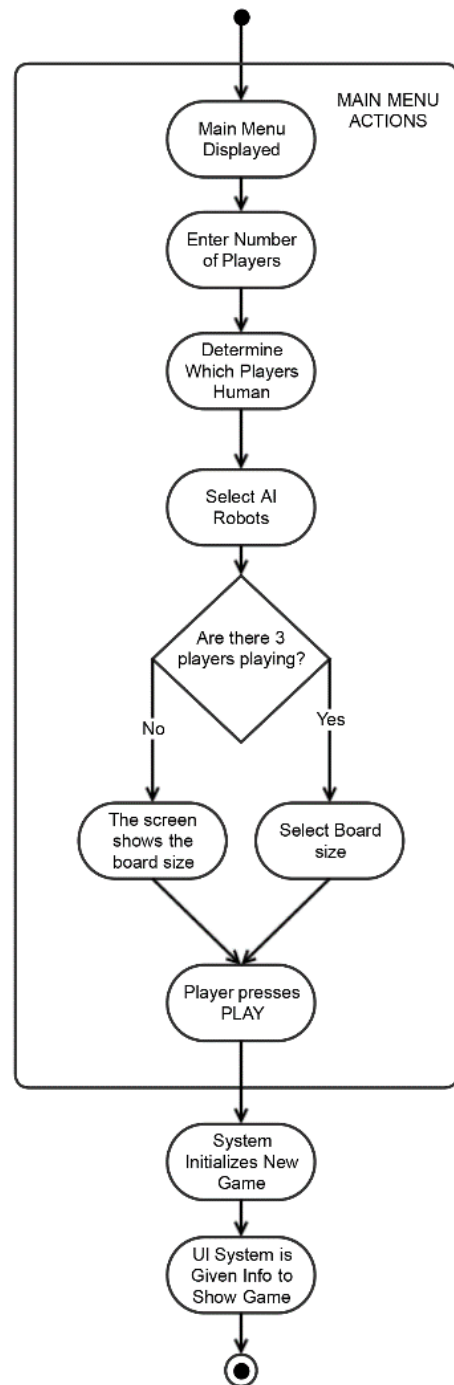
## **Scenarios:**

The following are primary scenarios the actors will encounter while interacting with the system. Use-cases as well as diagrams are present for each one. Some secondary scenarios are present and are pointed out as such in their respective primary scenario's section. It is important to note that this does not cover all technicalities of the system: they are simply what we have identified as primary scenarios and our actors will repeatedly be a part of.

## Create Game:

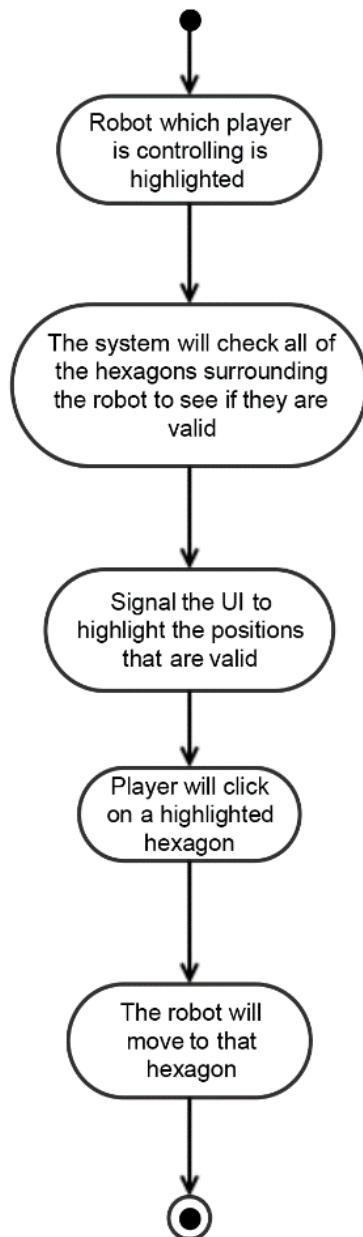
When a menu navigator presses “PLAY” from the main menu they are prompted with a display showing options that they could play the game with. These options are the number of human players, the number of computer players, the robots that the computer players are using. If there are three players, there will also be an option to select a board size. Other player amounts require a fixed board size. The menu navigator will then hit another “PLAY” button to continue to the in game view.

1. The use case begins when the navigator selects play from the main menu.
2. The navigator enters the amount of players (two, three, or six).
3. The navigator selects which players will be human and which will be computer.
4. The navigator will select an AI for each of the three types of robot for every computer player.
5. If the amount of players is three, the navigator may select between a board size with sides of length five or length seven.
6. The navigator presses play
7. The system initializes a new game with the specified parameters
8. The UI system is notified of the parameters and what will be displayed to the screen
9. The use case ends



## Move Robot:

When it is a player or AI's turn and they wish to move a robot they can do so using the mouse. All of the spaces around the robot that it can possibly move to will be highlighted. Clicking on one of these locations will result in the robot moving one space into that location. If the robot has already moved as far as it can for that turn no spaces will be highlighted.



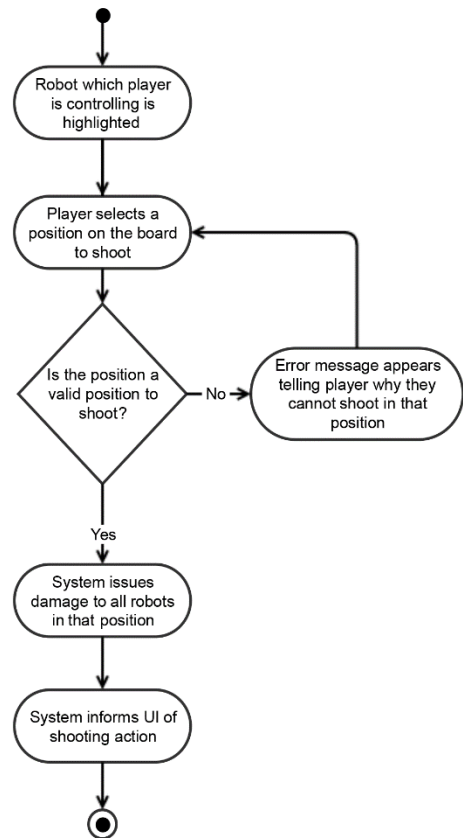
1. The use case begins when it is the player's turn and they are in moving mode
2. The system will determine which of the six positions around the robot are valid locations to move to
3. The player will click on the game board on one of the hexagons adjacent to the robot that is highlighted
4. The system will determine the necessary number and direction of rotations the robot will need to make to face the selected hexagon
5. The system will perform the necessary rotations and move the robot forward into the selected position
6. The use case ends



## Shoot Robot:

When it is a player's turn and they wish to shoot with a robot they can do so using the mouse. When the player is in shooting mode the player will be able to click on a board tile that is in range to fire upon that location. All robots in the location that is being fired upon will receive damage.

1. The use case begins when it is the player's turn and they are in shooting mode.
2. The player will select a position to shoot.
3. The system will check that the board tile selected is within range and is a valid place to shoot.
4. If the board tile is selected is valid the system will issue damage to all robots on that board tile.
5. The system will inform the UI of the shooting action.
6. The use case ends.



### Secondary Scenario

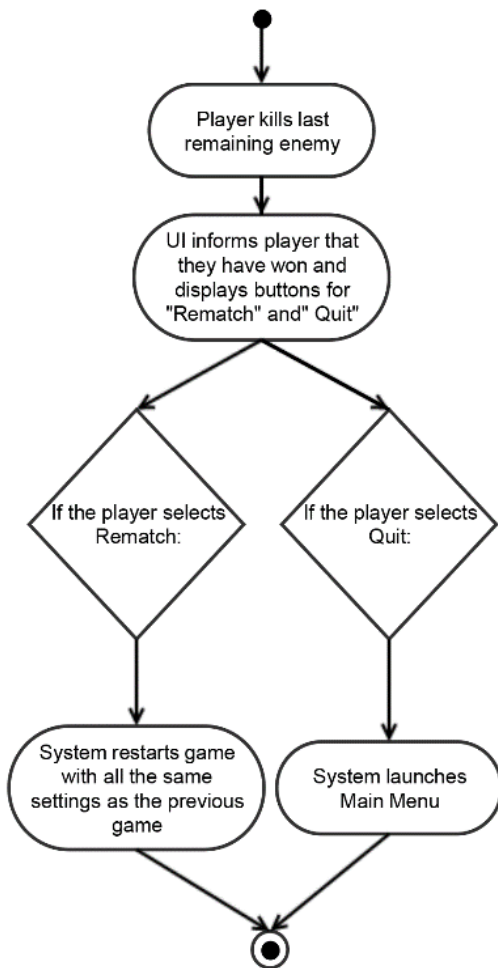
#### Invalid Hexagon:

Use case: The player has selected an invalid hexagon to move fire on (out of range).

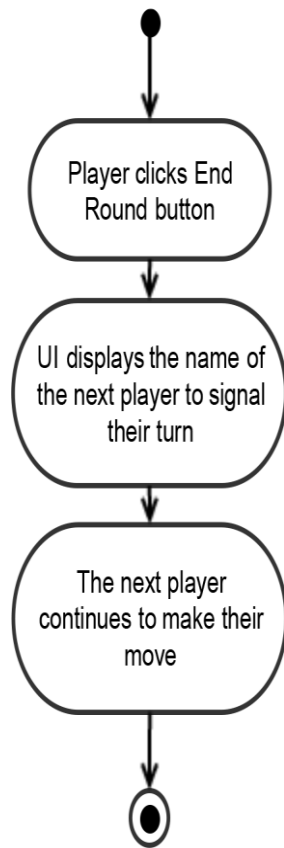
1. Error message pops up explaining why the player can't shoot.
2. Return to selecting a position to shoot.
3. Use case ends.

## Winning:

If a player has killed the last robot of the only other remaining team, the player has won. Gameplay will cease and a message will appear to that player that they have won and will prompt the player to have a rematch or to quit.



1. The use case begins when the player kills the last remaining robot of the only enemy team left alive
2. The system informs the UI of the victory
3. The UI shows on the screen the victory graphic and the two prompts for the player to select from, rematch and quit
4. If the player selects rematch the system will restart the game with all of the same settings as before
5. If the player selects quit the system will launch the main menu
6. The use case ends



## Ending Round:

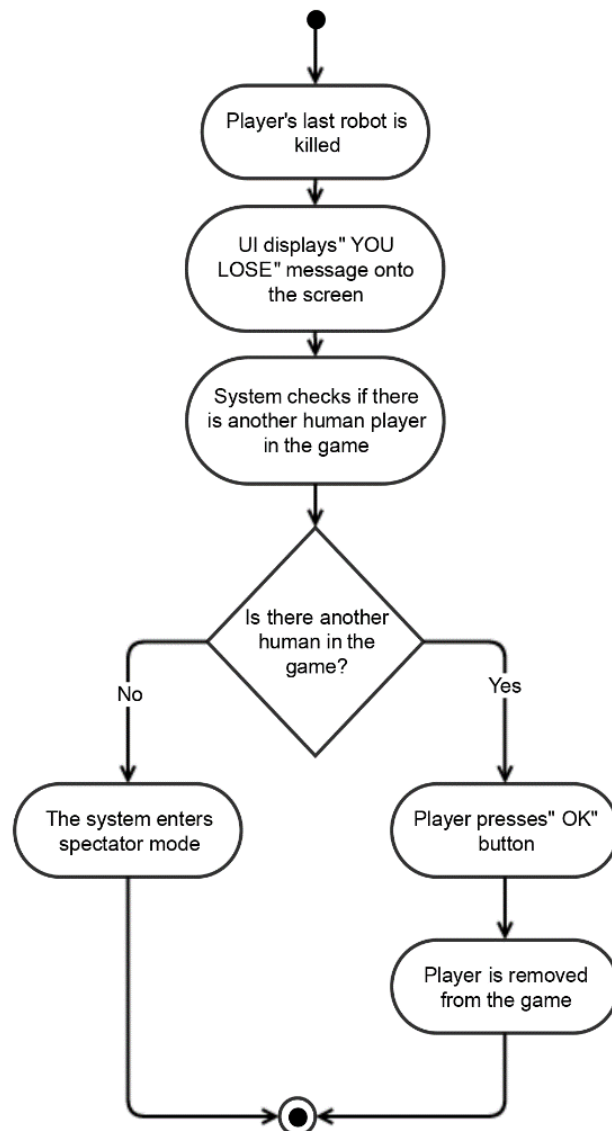
At any point in a player's turn, they can decide that it is the end of their round by pressing the "End Round" button in the main Game UI. The game will then move on to the next player. If the player has not elected to end the round before they have used all of their movement points, then the game will be automatically move on to the next player.

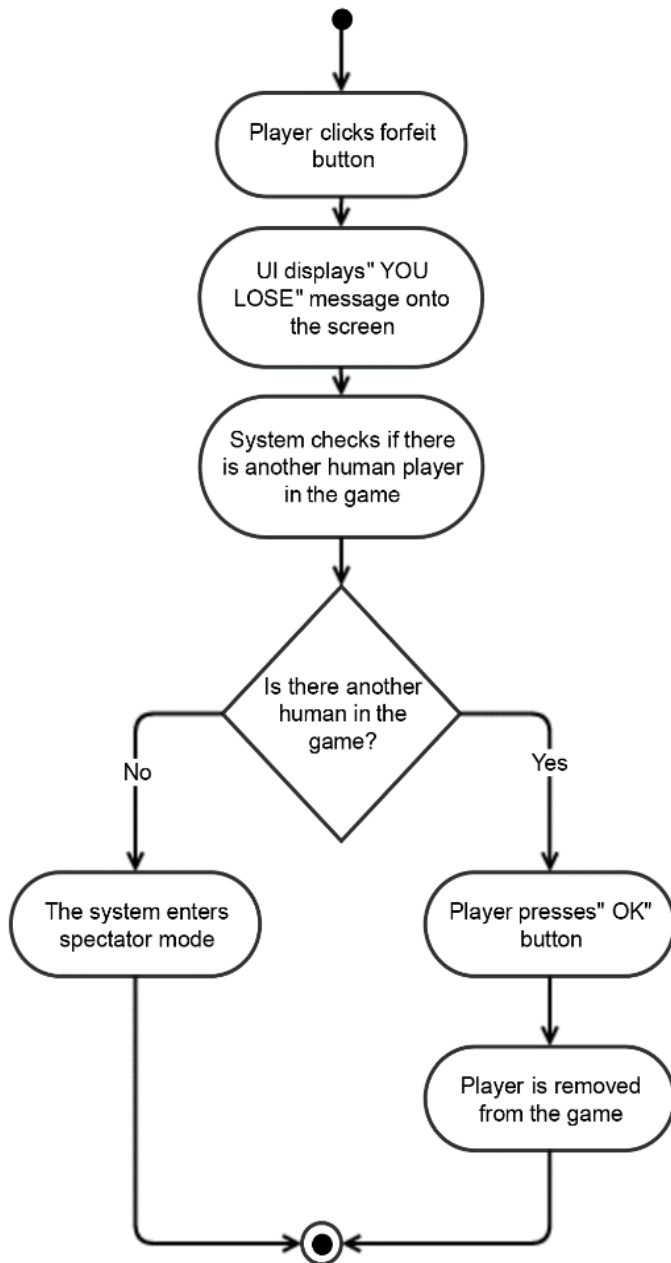
1. The use case begins when the player selects "End Round".
2. The system will then move onto the next player.
3. The next player will be then prompted to play their highest movement distance robot that they have not used yet in that turn.

## Losing:

If a player's last robot is killed they have lost the game. After a player's last robot is killed when it is their turn again a message will show on the screen saying they have lost. If the player is the only human left in the game the game will enter into spectator mode.

1. The use case begins when the last robot of a player's team is killed.
2. The system informs the user they have lost the game.
3. The system determines if there are any other human players in the game.
4. If there are not any other human players in the game the game will enter spectator mode.
5. If there IS another human in the game, the UI will simply display an OK button.
6. The use case ends when the user selects an option.





## Forfeiting Game:

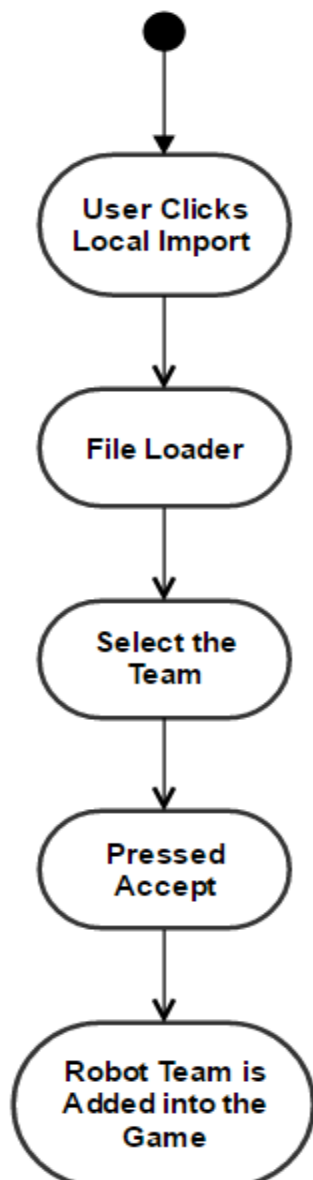
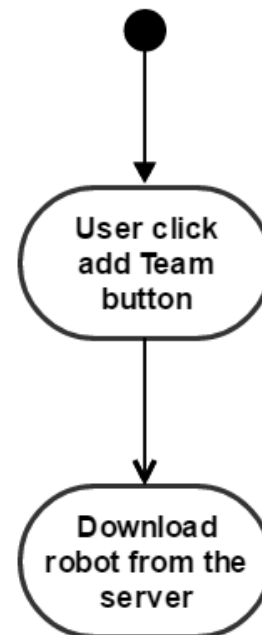
On any turn a player may choose to forfeit the game using the forfeit button in the top right corner of the screen. Upon doing so the player will be directed to the "YOU LOSE" screen which will be the exact same for either death or forfeit. If the player is the only human left in the game the game will enter into spectator mode.

1. Use case begins when the player selects forfeit when it is their turn
2. System changes all of the player's robots to zero health.
3. The system informs the user they have lost the game.
4. The system determines if there are any other human players in the game
5. If there are not any other human players in the game the game will enter spectator mode
6. If there is another human in the game the UI will simply display an OK button
7. The use case ends when the user selects an option

## Download:

Download scenario is for when the user decides to add a robot team, the system would download the robot team and place it into the local robot folder.

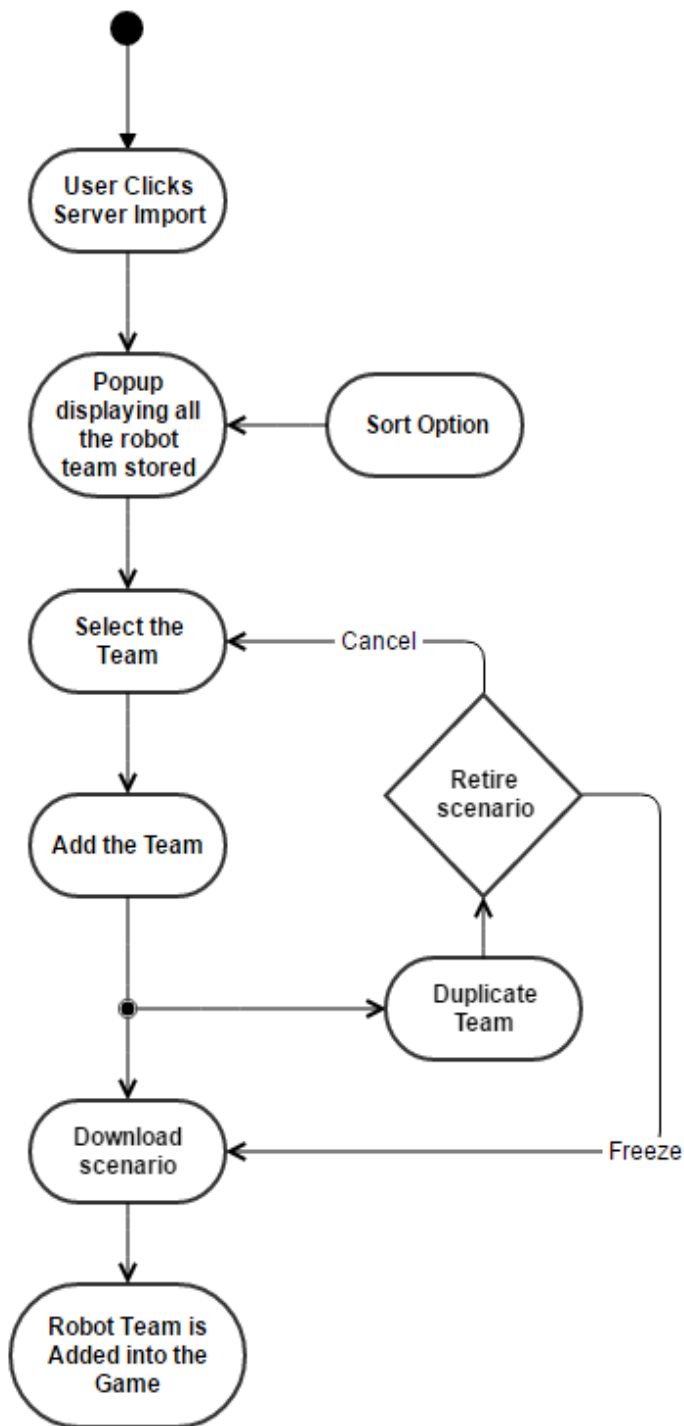
1. Download happens when the user clicks the Add Team
2. The system would download the robot from the server, providing an robot record so the simulator can run it



## Register:

The register scenario is for adding tanks into the game from local file.

1. Register happens when the user clicks Local Import.
2. A file loader would appear, showing all the robot team files that is in the robot team file.
3. User selects the team to register.
4. User presses accept.
5. Add the team in to the game.



## Enumerate:

The enumerate scenario take the robot team from the server, take their statistics and display them in a list for the user to pick from.

1. Enumerate happens when the user clicks Server Imports
2. A popup would appear displaying all the team of robot that is stored on the server, each robot team on the list would show the team name, wins, match played, win/loss ratio, and version.
3. The user is able to sort the list of robot by clicking either Team, Name, Win, Matches Played, or Win/Loss Ratio.

4. User select the Team
5. User pressed Add Team
6. Download Scenarios
7. Add the team in to the game

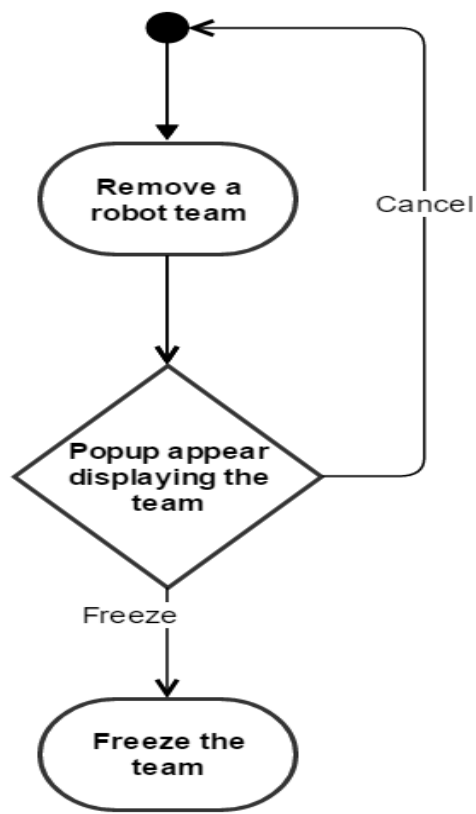
## Secondary Scenario:

**Duplicate:** A retire popup would appear (After Step 5)

1. The duplicate happens when the user tries to add a team with a name that has already been taken.

2. The retire primary scenario happen

3. If the user decides to press freeze team that already has the name, go back to step five of the Primary Scenario. If the user decides to choose a, go back to step four of the Primary Scenario.



## Retire:

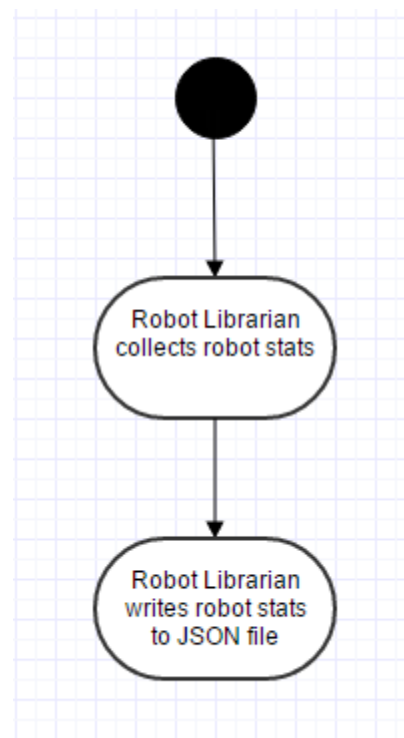
Retire scenario is for when the user decides to remove a robot team from the game, but not delete the team file.

1. Retire happens when the user decides to remove an robot team
2. A popup would appear displaying the robot team that is going to be freeze with its statistics, with two option, Freeze and Cancel
3. The user pressed Freeze
4. The game would freeze the team, making the team's name available for use

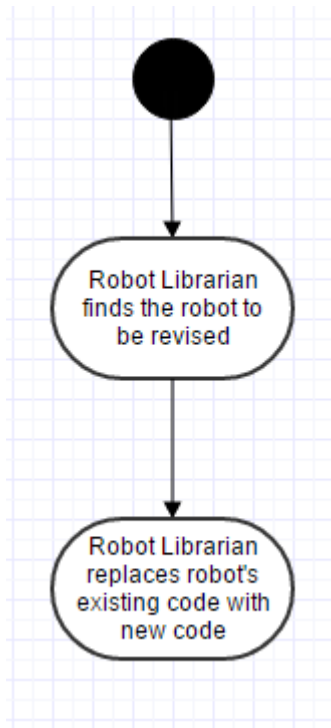
## Update Stats:

Update stats scenario begins when the robot librarian is told to update the stats of a robot

1. Robot Librarian collects stats from the robot itself.
2. Robot Librarian writes stats to the robot's JSON file.
3. Use case ends







## Revise Robot:

Use case begins when the Robot Librarian has been code to revise a selected robot with.

1. Robot Librarian finds the robot to be revised
2. Robot Librarian replaces the robot's existing code with the given new code.

## Spectate:

Use-case begins when a Human Player has been defeated, or if the game has been started with no Human Players.

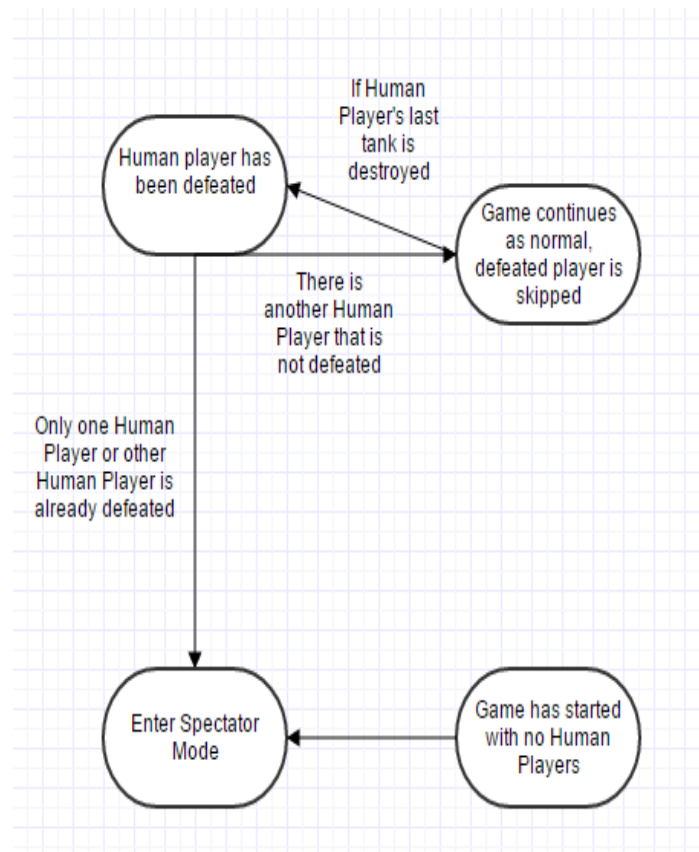
1. Human Player has been defeated or there are no Human Players to begin with
2. User enters spectator mode to observe gameplay

Secondary Scenario:

### Other Human:

Use-case begins when two human players and only one of them has been defeated.

1. The defeated player has their turn skipped every round
2. If the remaining Human Player dies, the primary Spectate scenario takes place.



## Spectator Mode

Once all of the human players in the game have forfeited or died the game will enter into spectator mode. In this mode the player(s) will not be playing but rather watching the remaining AI computer players make their moves. When in this mode the button options for “Move”, “Shoot”, and “End Turn” will be changed to “Pause/Play”, “Fast Forward”, “End Game”, “Fog of War”. During spectator mode, the game will carry out based on time, regulating the speed of the computer player’s movements based on a timer. (Once every second the computer AI will make a move for example). This is why we have the need for speed controls in the spectator mode.

The fast forward control allows for quick viewing of the game and the pause button allows for the spectator time to view the board. The fog of war button is an on/off toggle for the fog of war view of the board. If the toggle is turned on, the spectator will only be able to see what the computer player can see of who’s turn it is. If the toggle is off, then the spectator can see the entire board regardless of which team’s turn it is. To exit spectator mode, the spectator can press “End Game”. This button will skip all remaining turns of the computer players and show the Victory box displaying the stats of all the robots in the winning team.

## Hardware, Software, and Data Formats

This game is being developed to run on Linux, with the U of S’ *tuxworld* in mind. It will be written in Java, making use of the AspectJ extension. This aspect-based extension allows us to improve modularity and will help with testing. We will be doing our development using Oracle Corporation’s Eclipse IDE. The robots in our program operate and communicate using the RoboSports370 control language, modelled in forth. Their statistics and other information will be kept track of using the JSON file format.

## The Team

We are team C4, made up of Jack, Ixabat, Daniel, Kevin, and Brandon. Each member has their own unique skillset that makes us an overall well-rounded team. Decisions are decided in a democratic fashion with emphasis on member-input. Communication is frequent and meetings are organized multiple days a week to ensure members are up-to-date on current objectives. Our team is making use of Git as our form of version-control.

## **Summary**

Our team will be creating a computer version of the board game described prior. It will require players to move and shoot robots on a hexagon-based board. Up to two human players will be possible on the same computer, as well as up to four AI players. Without human players, six AI players will be possible. Robots will have different statistics which will be stored in a JSON file. It will also be possible to import, update, and revise them

The application's user interface will be split into three main sections: the menu, the options, and in-game. The menu will have buttons for creating a game, viewing options, and exiting the application entirely. If the user creates a game, they will be presented with a screen prompting them to choose a number of players and a board sized based upon it. The game will then begin with the specified parameters. If the user views options, they will be able to configure robot teams, import new robots, and delete existing once.

Once the player has entered the game, they will see the game board in the middle, including their tanks and the hexagons visible to them. There will be graphics on the left side indicating the current stats of each robot. There will be an indicator of the current team's turn at the top, and buttons at the bottom for moving, shooting, and ending the current play. On the right side, there will be a log of events and buttons for exiting or forfeiting.

It will also be possible to spectate games, in which the user will watch AI players take turns until the game is over. This will only happen if there are no human players currently alive in the game. In this case, the user will watch at their own pace and exit the application freely whenever they want. When spectating, whichever AI is playing will have their robots displayed to the user and their field of view.

The game will have five actors, including human players and AIs, performing a total of thirteen actions. Various scenarios such as moving and shooting will take place, making use of said actions. These are the primary situations our actors will encounter while interacting with our system. The system itself will be developed in Java, making use of the AspectJ extension. The hardware will be the U of S' *tuxworld*.