**Prepared by: Team B5**

Lin, YuCHEN | NELSON, JORDAN | PARK, RYAN | WANG, XINGENG | vAN HEERDE, WILLIE

*RoboChess*

Requirements Document

Version 1.3

October 2, 2016

Table of Contents

[**1. Introduction** 3](#_Toc463164752)

[1.1 Team Members 3](#_Toc463164753)

[1.2 Purpose 3](#_Toc463164754)

[1.3 Intended Audience 3](#_Toc463164755)

[1.4 Platform 3](#_Toc463164756)

[**2. Description** 3](#_Toc463164757)

[2.1 Game Summary 3](#_Toc463164758)

[2.2 Win Conditions 4](#_Toc463164759)

[2.3 Stalemate Conditions 4](#_Toc463164760)

[**3. System Features** 4](#_Toc463164761)

[3.1 Core Features 5](#_Toc463164762)

[3.1.1 Forth Interpreter 5](#_Toc463164763)

[3.1.2 Hot Seat Multiplayer 5](#_Toc463164764)

[3.1.3 Graphical User Interface 5](#_Toc463164765)

[3.1.4 Display Player Statistics 5](#_Toc463164766)

[3.2 Additional Features 5](#_Toc463164767)

[3.2.1 Animation/Enhanced Graphical User Interface 5](#_Toc463164768)

[3.2.2 Sound Effects 5](#_Toc463164769)

[3.2.3 Leaderboard/Point System 6](#_Toc463164770)

[3.2.4 Custom Game Setups 6](#_Toc463164771)

[3.2.5 Turn Timer 6](#_Toc463164772)

[3.2.6 Tutorial 6](#_Toc463164773)

[3.2.7 Difficulty Settings 6](#_Toc463164774)

[3.2.8 Networked Multiplayer 6](#_Toc463164775)

[3.2.9 Hint System 6](#_Toc463164776)

[**4. Actors and Actions** 7](#_Toc463164777)

[4.1 Player 9](#_Toc463164778)

[4.2 Robot Librarian 9](#_Toc463164779)

[4.3 Forth Interpreter 9](#_Toc463164780)

[4.4 New Game 9](#_Toc463164781)

[4.5 Start Game 10](#_Toc463164782)

[4.6 Quit 12](#_Toc463164783)

[4.7 Move 13](#_Toc463164784)

[4.8 Fire 16](#_Toc463164785)

[4.9 End Turn 18](#_Toc463164786)

[4.10 Options 19](#_Toc463164787)

[**5. GUI Storyboards** 21](#_Toc463164788)

[5.1 Main Menu 21](#_Toc463164789)

[5.2 Options 22](#_Toc463164790)

[5.3 Player Transition Screen 23](#_Toc463164791)

[5.4 AI Transition Screen 24](#_Toc463164792)

[5.5 Combat Screen 25](#_Toc463164793)

[5.6 Game Setup Screen 26](#_Toc463164794)

[5.7 End Game Screen 27](#_Toc463164795)

[5.8 Stats Screen 28](#_Toc463164796)

# **1. Introduction**

## Team Members

Our team consists of two computer engineering students, Xingeng Wang and Yuchen Lin, as well as three computer science students: Ryan Park, a fourth year; as well as Jordan Nelson and Willie van Heerde, third years. Our objective is to create a multiplayer turn-based strategy video game.

## Purpose

The purpose of creating such a game is to give us experience with the concept of software-engineering, working as a team to complete a large programming task with a design-first approach. The reason for selecting a game for such a task is a good one, as it is fun and holds our interest while giving us the option and motivation to go above and beyond the scope of the requirements and teach ourselves interesting and/or challenging new things.

## Intended Audience

The intended audience for RoboChess is mainly ourselves, as well as children, as the game is easy to grasp and becomes simple to win once certain strategies are learned.

## Platform

The platform we will design our game for will be tuxworld, the Linux servers based in the computer science department of the University of Saskatchewan. Because of the fact that macOS and Linux are both based on Unix, as well as the fact that several of our team members will be programming and testing on a macOS system, the game will also run on macOS. The game will be designed and implemented using a variety of software systems, including GIT for version control, Java as the programming language, and Eclipse as an integrated development environment. Other software may be used, but is still to be decided.

With hardware, the game is simple enough that almost any semi-modern machine running either Linux or macOS/OS X should be able to run the game perfectly fine.

# **2. Description**

## 2.1 Game Summary

The game consists of a varied number of players and is played on a hexagonal tile-based board. The board’s size is dependent on the number of players playing. Each player starts on an edge tile of the board, evenly spaced from the other players. Each player also starts with three robots: a scout, a sniper, and a tank. The three robots have separate and unique attributes for health, attack damage, movement, and shooting range.

Turns are separated into three categories: a turn, a round, and a play. In a turn, a player plays the robot with the highest movement points that hasn’t yet played that play. In that turn, the player can move that robot anywhere on the board (up to the maximum movement range of that robot), fire anywhere on the board once per turn (up to the maximum range of that robot), or even choose to do nothing at all and end the turn all together.

When the turn is over, the next player begins their turn, by finding their robot with the highest movement points that hasn’t yet been used that play, and so on. When all players have had one turn, the round is over. A new round begins in the same manner, with the next highest movement point robot in each player’s arsenal, and so on. When all players have moved all robots the play is over, and a new play begins. Note that if a player’s robot is killed, and they only have two left for example, they will not have a turn during the third round, as all of their robots will have been used that round.

## 2.2 Win Conditions

The objective of the game is to be the last player with at least one or more robots alive.

## 2.3 Stalemate Conditions

As an extra feature, as well as a precaution against certain AI vs. AI games having an infinite amount of turns, we will implement a stalemate system.

A game shall be declared a stalemate and end if one (1) or more of the following conditions are met:

(a) No players spends movement points for X consecutive turns (X to be determined)

(b) No players issues attack commands for X consecutive turns (X to be determined)

(c) No damage has been dealt to any robot for X consecutive turns (X to be determined)

# **3. System Features**

The nature of the game means we have many options on how to implement it, and the ability to add quite a few additional features. Features are divided into two (2) main categories: core features and additional features. Core features are essential to the game’s operation and are the bare minimum requirements we will implement upon the completion of the game. Additional features are extra functionalities. They will only be implemented as time permits as they are not necessary for the software to function properly.

## 3.1 Core Features

### 3.1.1 Forth Interpreter

We will implement an interpreter to communicate between various artificial intelligence programs and our game system, so that a single human player can play against several unique computer players during a single play session. In the event the user wishes to have a game run with only computer players, features will be available so the user may either view the result as quickly as possible, or enable a ‘spectator mode’ where they may watch the computer players play against each other, at their leisure.

### 3.1.2 Hot Seat Multiplayer

Human vs. Human multiplayer will be implemented using a ‘hot seat’ multiplayer method, meaning the game will be localized on a single computer where the players will take turns playing their turn at the computer. In order to avoid “screen peeking” there will be a transition page in-between turns.

### 3.1.3 Graphical User Interface

The game will be displayed using a very basic graphical user interface, which the player can interact with using a mouse and various buttons displayed on the screen. The game will also have a menu system outside of the game that the player will use to choose from various options, including how many players are playing, how many human players, etc.

### 3.1.4 Display Player Statistics

We will implement a status bar on the made interfaces, allowing players to see the robots they have left alive, their health, as well as the range and movement range of their current robot.

## 3.2 Additional Features

The next few items, as explained above, are things we would like to implement, given enough time.

### 3.2.1 Animation/Enhanced Graphical User Interface

We would like to possibly implement animations and a more advanced graphical user interface to increase the aesthetics and enhance the immersion of the game. Three-dimensional robot/game board models may also be considered but, unlikely.

### 3.2.2 Sound Effects

Implementation of more sounds to increase the appeal of the game and give it more polish is another feature we would like to implement.

### 3.2.3 Leaderboard/Point System

The implementation of a leaderboard and/or a point system would enhance the competitiveness of the game

### 3.2.4 Custom Game Setups

We would like to add extra features to the game to enable customized setups, which would include things such as non-regulated map sizes (2 players on a 6 player board, 6 players on a 2 player board, etc.), power-up abilities, or terrain features to the board.

### 3.2.5 Turn Timer

Another optional feature we would like to implement is a turn timer, to limit the time a player has to make a turn. The length of this would be regulated in the game set-up menu.

### 3.2.6 Tutorial

We would like to implement a tutorial of some kind, accessed through the main menu, for new players to learn the game.

### 3.2.7 Difficulty Settings

The last feature we would like to implement, given the time, is the option for a player to set a difficulty level against a computer player. This could be implemented either by giving the player a boost/deflation to their robots’ stats, or by having a pre-screened subset of AI programs, sorted by their known difficulty level.

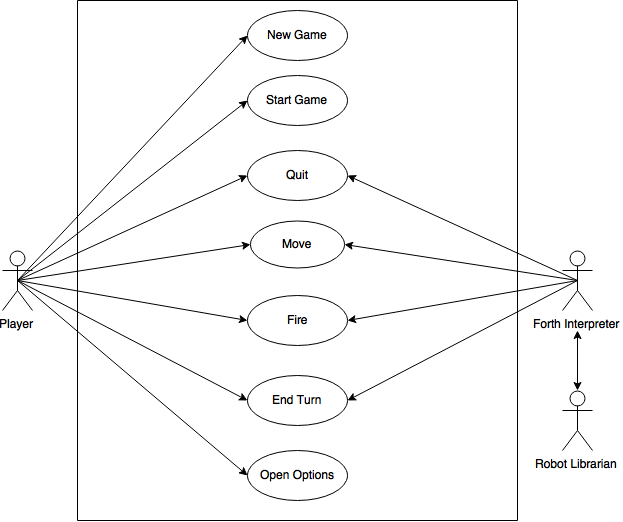
### 3.2.8 Networked Multiplayer

We would like add additional functionality to allow network multiplayer, where each player sits at a separate computer and waits for their turn sequentially.

### 3.2.9 Hint System

The last feature we would like to implement given enough time is an optional hint system for new players, where we make an AI program read the current human player’s positions as well as all known enemy positions and gives the player hints based on what the AI would do.

# **4. Actors and Actions**



The system diagram above shows that interactions with the game are relatively simple. A player can choose a variety of games from a menu screen, including ‘new game’, which opens a set-up screen; ‘start game’, which starts the game after the set-up screen; and ‘open options’, which opens the options from the main menu. In game, a player can ‘move’, ‘fire’, ‘end turn’, and ‘quit’, which are all fairly self explanatory, but are explained in greater detail below.

The robot librarian is a library containing various AI programs capable of playing the game, which communicates with the Forth Interpreter, which is a program that translated the commands of an AI program written in forth so that it can communicate the game, which is written in Java. The interpreter, as a control input for the AI, does not need access to any menu features, and thus simply can ‘move’, ‘fire’, ‘end turn’, and ‘quit’.

* Interfaces for actors:
  + “Player”
    - Move
    - Fire
    - Start game
    - New game
    - End turn
    - Options
    - Quit
  + “Forth Interpreter”
    - Move
    - Fire
    - Start game
    - End turn
    - Quit
  + “Robot Librarian”
    - Register
    - Retire
    - Revise
    - Enumerate
    - Download
* Interfaces for actions:
  + “New Game”
    - Get robot from librarian
    - Start a new game
  + “Start Game”
    - Get the game parameter (# of players & map size) information
    - Initialize the game
  + “Quit”
    - Delete all robots of current player
    - Remove player from turn queue
    - Check turn queue
  + “Move”
    - Get the robot information (move distance)
    - Get valid hexes
    - Update fog of war
    - Update the robot information
  + “Fire”
    - Get the robot information (fire, distance, health)
    - Get valid targetable hexes
    - Update robot information
    - End game
  + “End Turn”
    - Check stalemate
    - Check timer
    - Check turn queue
  + “Options”
    - Get game settings
    - Set game Settings
    - Save game settings

There are a total of three actors on our system. These are:

## 4.1 Player

The player is a human player, playing the game and interacting with it using a traditional computer mouse.

## 4.2 Robot Librarian

The robot librarian is a library which contains a variety of programs (AI) designed to play the game with a variety of strategies and skill.

## 4.3 Forth Interpreter

The forth interpreter acts as a translator between the main system of the game and any program within the robot librarian, as the two are written in two different programming languages.

## 4.4 New Game

‘New Game’ is the action that sets up the game to be played. The primary user will from the main menu choose the “create game” button. From here the user will input the number of human players & the number of AI players. It is possible for the user to create a game where there are: Only AI players, only human player, and a mix of both. No matter how the user chooses to organize the game the user will then press the confirm button which transitions into the board view perspective for the first.

* Pre conditions:
  + None
* Flow of events:
  + Default path:
    1. From the main menu the user presses the play game button.
    2. User selects the number of players in the game (AI or real).
       - If there are 3 players, the user specifies if they want to play on the big or small.
    3. User presses the start game button.
    4. The program transitions from the menu to the gameplay screen.
* Alternate paths:
  + - N/A
* Post conditions:
  + The user is in a game that has the parameters that the user set in the menu.
* Error conditions:
  + The player presses the play game button without specifying the number of players.



## 4.5 Start Game

The ‘Start Game’ action takes place after the start game action. In this action, the game will find out what is the size of the board, populate all the robots needed for the game and determine which players has which color assigned to them.

* Pre Conditions:
  + The user has chosen the number of human players and AI players for this new game.
* Flow of Events:
  + Default Path:
    1. The user clicks the "Start Game" button.
    2. The system determines which board size to use.
    3. The robots are placed in their respective starting tile.
    4. Display the transition screen for the first player.
  + Alternate Path(s):
    - Alternate Path 1 (If all AI players)
      1. If Spector Mode is on
      2. Display game board.
      3. Run game quickly.
* Alternate Path 2 (if all AI players)

1. If Spector Mode is off
2. Don't display game board.
3. Run game quickly.

* Post Conditions:
  + A new game has started.
  + It is player #1’s turn.
* Error Conditions
  + N/A



## 4.6 Quit

The “Quit” action may be invoked by any human or AI player at any time during their turn.

Once invoked it causes the player to withdraw from the game. All of the current players’ robots are removed from play. If there are only 2 players active in the game, then the game ends.

* Pre-Conditions
  + Game must be in progress.
  + The current player is still alive.
  + There is another player on the board still alive.
* Flow of Events
  + Default Path
    1. The player clicks “Quit” button.
    2. Confirm desire to withdraw.
    3. Player’s robots are removed from the board.
    4. Transition to next player’s turn.
  + Alternate Path (2 Active Players Left)
    1. One of the 2 players quits.
    2. Game ends.
    3. Display the end game screen.
* Post-Conditions
  + Player’s robots are removed from the board.
  + Player is removed from the turn queue.
* Error Conditions
  + None



## 4.7 Move

A ‘Move’ action happens when a human or AI player chooses to move their current robot, as long as that robot is able to move, that player is still alive (and thus the robot is also still alive) and there is another player on the board still alive. After the move action is over, the robot is on the hex the player wished, and the movement points the robot has left that turn are decremented by the amount the player moved. The actual move happens in the following way:

1. The player uses the mouse to click the 'move' button on the GUI, and the hex's within the movement range around the robot are 'highlighted' by lighting up/displaying some kind of colour/animation.
2. The player then clicks on which hex they wish the robot to move to.
3. When this happens, the previous hex's that were highlighted return to normal, and the robot moves to that hex (possibly by playing some animation).
4. The old hex the robot was on is un-highlighted from its 'current robot hex' animation, and the new hex the robot is on is highlighted instead.
5. The fog of war is adjusted, so out-of-range hex's are covered and in-range hex's are revealed.
6. The stat showing the movement range the robot has left is then decremented by the amount the robot has moved.
7. After this, the game then waits for the next player input.

* Move:
  + Preconditions:
    - The current robot is able to move.
    - The current player is still alive.
    - There is another player on the board still alive.
  + Flow of Events:
    - Default Path:
      1. The player clicks the move button.
      2. The hex's within the robot's movement range highlight.
      3. The player clicks which hex the robot will move to. This causes the tile to be highlighted.
      4. The player then presses the “Confirm” key.
      5. The movement range hexes un-highlight.
      6. The robot is moved to that hex.
      7. The robot’s current tile is highlighted.
      8. The fog of war is adjusted.
      9. The amount the current robot can still move is decremented by the amount moved.
    - Alternate Path(s):
      * -N/A
  + Post-conditions:
    - The current robot is on a new hex.
    - The amount the robot can move is decremented.
  + Error Conditions
    - The player tries to move to an out-of-range hex
    - The player clicks the move button when they have no movement points.



## 4.8 Fire

The user is able to attack a tile provided that it is within their current robot's range and they have not attacked in the current turn. The user is able to attack a tile if it is within range of the robot, this means the user can choose to attack a tile that is: empty, has a robot occupying it (even your own robot), the tile that the current robot is occupying. Note that during the user's turn they can attack once at any point during their turn so they can shoot before moving and vice versa.

* Preconditions:
  + The player has a robot that is still alive.
  + The player has not already attacked this turn.
* Flow of events:
  + Default Path:
    1. Position cursor over "fire" button and click it.
    2. The tiles that the tank is able to fire onto will highlight.
    3. The user clicks on the desired tile. This causes the tile to be highlighted.
    4. The player presses the “Confirm” key to execute the command.
    5. The tiles become un-highlighted.
    6. Tank fires at target.
    7. Damage gets calculated.
  + Alternate paths:
    - N/A
* Post conditions:
  + Player has attacked a tile within the range of their tank. A tank may have suffered damage or be destroyed. More than 1 tank can also be damaged in the attack.
* Error conditions:
  + The player tries to attack a tile outside the range of their tank
  + The player tries to attack a tile that is in the fog of war
  + The player has already attacked this turn



## 4.9 End Turn

The “end turn” action can be invoked by the current player at any point during their turn. It can also be triggered if the current player turn timer has reached 0. When a player ends their turn the next player in the queue will start their turn. Notice if there is only one player is alive then “end turn” action will not happen; it will transition to the game over screen.

* Pre Conditions:
  + It is currently the player’s turn.
* Flow of Events:
  + Default Path:
    1. The player clicks the "End Turn" button.
    2. The player clicks on the “Confirm” button to proceed.
    3. Check if stalemate.
    4. Take next player from queue.
    5. Find next Robot.
    6. Check if next player is human.
    7. Display the proper transition page for the next player (AI or player screen).
  + Alternate Path(s)
    - N/A
* Post Conditions
  + It is the next player in the queue’s turn.
* Error Conditions
  + Window change to End Game while there is more than one player alive.



## 4.10 Options

This is the action that will allow the user to change the settings of the game. The option that can be changed include sound. After the user changes the options the software will save the current state of it to ensure that the options remain toggled properly.

* Precondition:
  + N/A
* Flow of events:
  + Default path:
    1. From the main menu click the options button.
    2. Toggle the options how the user sees fit.
    3. Press the back button to go back to the main menu.
  + Alternative path(s):
    - N/A
* Post conditions:
  + Options are set and saved.
  + The user is back at the main menu.
* Error conditions:
  + None



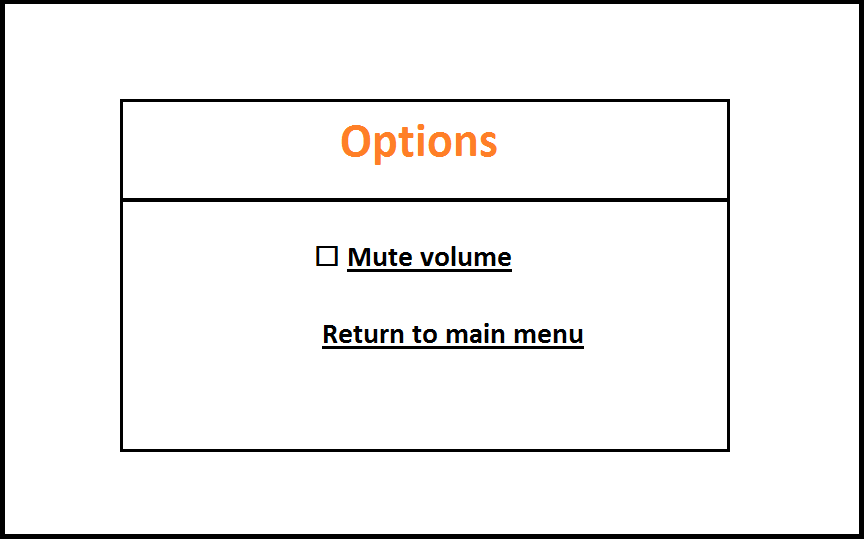
# **5. GUI Storyboards**

## 5.1 Main Menu



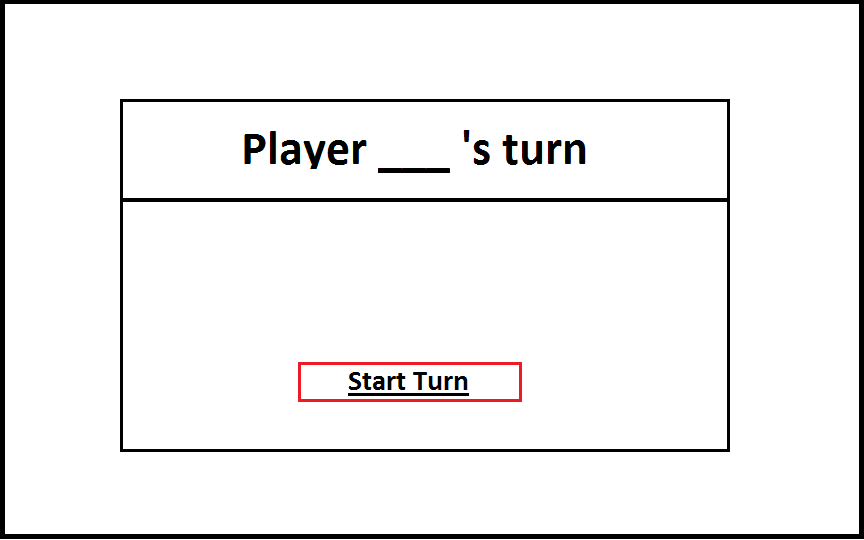
* When the program is started this is the screen it will always boot to.
* If the user presses “Create game” it will transition to the game setup screen.
* If the user clicks the “Options” button they will go to the options menu.
* When the “Exit” button is pressed the program will terminate.

## 5.2 Options



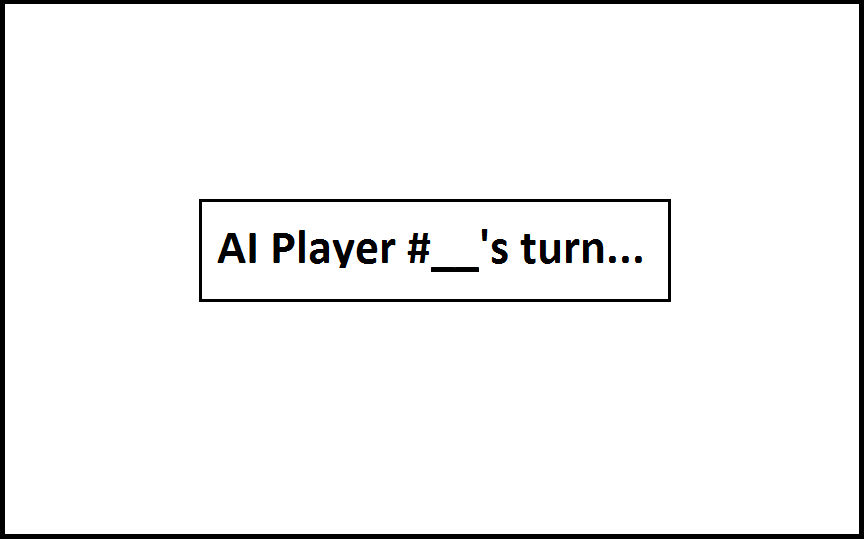
* If the mute box is clicked it will toggle between checked and unchecked. If it is toggled there will be no sounds in the software.
* If the “return to main menu” box is clicked the user will return to the main menu.

## 5.3 Player Transition Screen



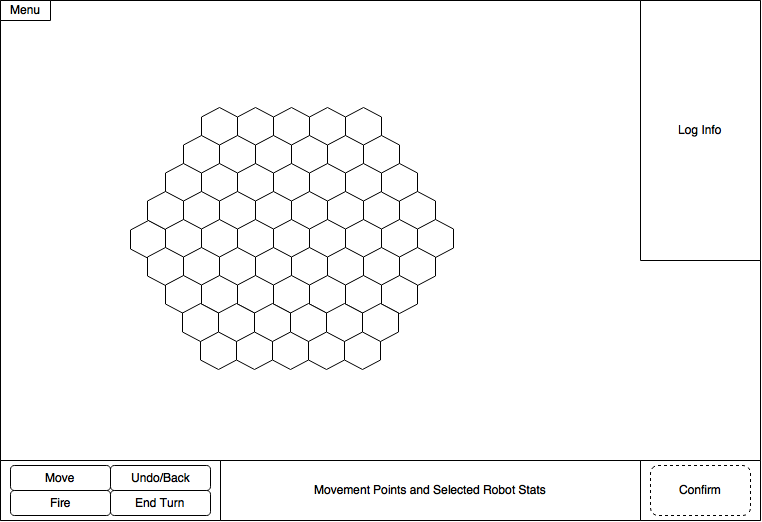
* On this screen the \_\_\_ will have the number of the human player whose turn it currently is.
* When the “Start Turn” button is pressed the software will go to the combat screen and show the player’s perspective.

## 5.4 AI Transition Screen



* This screen shows that it is currently an AI players turn.
* If spectator mode is on, the game will transition to the combat screen from the Ai’s perspective. If it is off this screen will show until the AI’s turn has ended.

## 5.5 Combat Screen



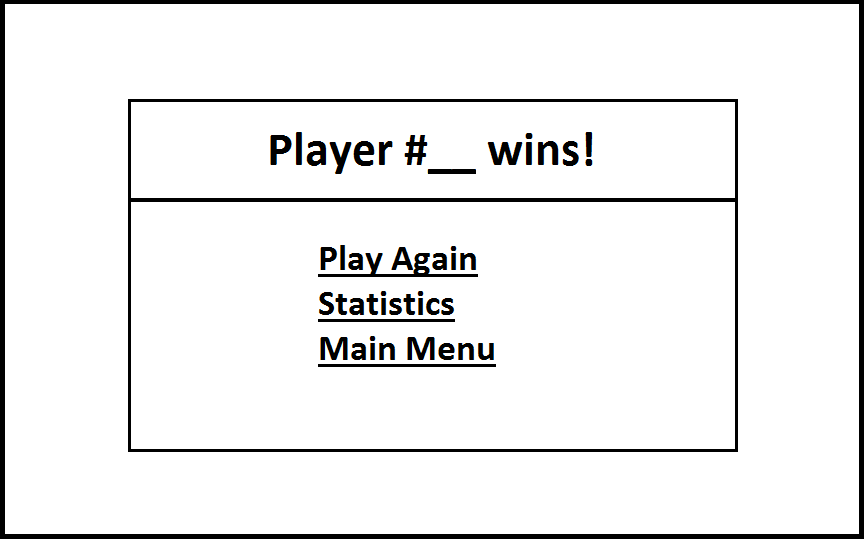
* This screen represents the game board.
* The log displays the events that have happened, this includes things from movement, attacking, and players being removed from the game because they lost all their units or quit.
* The menu button (top left) can be pressed and it will bring up a pop up that has 3 buttons:
  + 1. The first being an options button that brings up the options menu.
    2. The second button labeled “Quit” will allow the player to exit the game if clicked.
    3. Finally, the last button is a “Resume” button that will go close the pop up.
* The move, fire, end turn buttons function as described in the actions sections. For Move and fire the player will click on the button first then click on a highlighted tile then click the “Confirm” button to execute the attack. The “End Turn” button functions similarly except that a tile does not need to be clicked rather just the “End Turn” button then the “Confirm” button.
* Clicking the “Undo/Back” button will revert the last action of the player. This includes moving & attacking. If the player attacks and destroys the final enemy tank it will be impossible to undo that action because the game will be over.
* The section labeled “Movement Points and Selected Robots Stats” will show the status of each of the player’s robots.
* The grid composed of hexagons where the robots from all players move and attack each other. There is a fog of war imposed over the tiles that hide the enemy robots.

## 5.6 Game Setup Screen

\\cabinet\work$\wsv570\cmpt\cswin\Desktop\Create_game.png

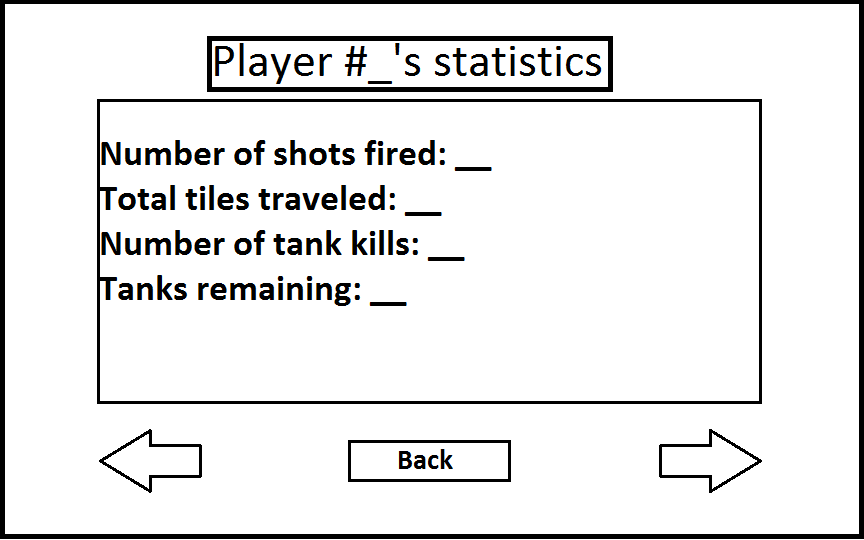
* The user presses on one of the checkboxes to determine if there are 2, 3, or 6 players.
* The user can input the number of human players in the drop down menu found below the checkbox that determines the number of players (must be checked or drop down menu will not work),
* If the player presses the “Spectator mode” checkbox they will enable spectator mode when the game starts.
* If the user has the valid fields filled in they can then press the “Create” button to start the game, if the fields are not correctly filled the button will be grayed out.

## 5.7 End Game Screen



* After the game is over this screen will display who won the game (the \_\_\_ will show what player won the game).
* If the user clicks on the “Play Again” button the system will create a new game that has the same parameters (number of players, AI players, and map size).
* If the “Statistics” button is pressed the software will transition to the statistics screen.
* If the “Main Menu” button is pressed the software will transition to the main menu.

## 5.8 Stats Screen



* This screen displays the statistics of each individual player. When this menu is first opened it will start with the stats of player #1.
  + “Number of shots fired:” will show the recorded amount of shots that were fired by the player.
  + “Total tiles traveled:” will show the total number of tiles that the player traversed in the game.
  + “Number of tank kills:” will show the number of enemy robots destroyed (does not count destroying your own tanks).
  + “Tanks remaining:” this will show the number of robots that the player had left when the game ended.
* The left and right arrows when pressed traverse the player stats in the queue.
* The “Back” button when pressed will return the user to the end game screen.