REQUIREMENTS

GROUP D1, CMPT370

Nico Dimaano, ned948

Niklaas Neijmeijer, nkn565

Kyle Seidenthal, kts135

Brendon Sterma, bws948

Jiawei Zang, jiz457

INTRODUCTION

This document outlines the requirements for creating a system for the “Robot Arena” game. The “Robot Arena” game is a simple turn-based game, played on a board. There can be 2, 3 or 6 players at a time. Each player begins with three robots, each with different statistics. The players take turns moving their robots, starting with the robot with the highest movement. On their turn, a player may move, shoot or do nothing. The last player with a robot on the board wins the game. The system we are creating will implement this game and its rules on a computer system that allows the game to be played by multiple players at once. This will make the game less tedious to keep track of, and allow a comfortable gameplay experience than playing the game on a traditional tabletop board. One of the main features of the game is a visibility system, where the tank can only see as far as its range, which brings out new strategy and difficulty to the game. Another feature is that you have the ability to face AI opponents. In addition, the game can be played over a network to allow the player to conveniently play from his or her own machine. The rest of this document will outline what such a system will require to be successful.

GIT TAG GOES HERE WHEN COMPLETE ---

SYSTEM DIAGRAM

Below is a high level diagram of the system:

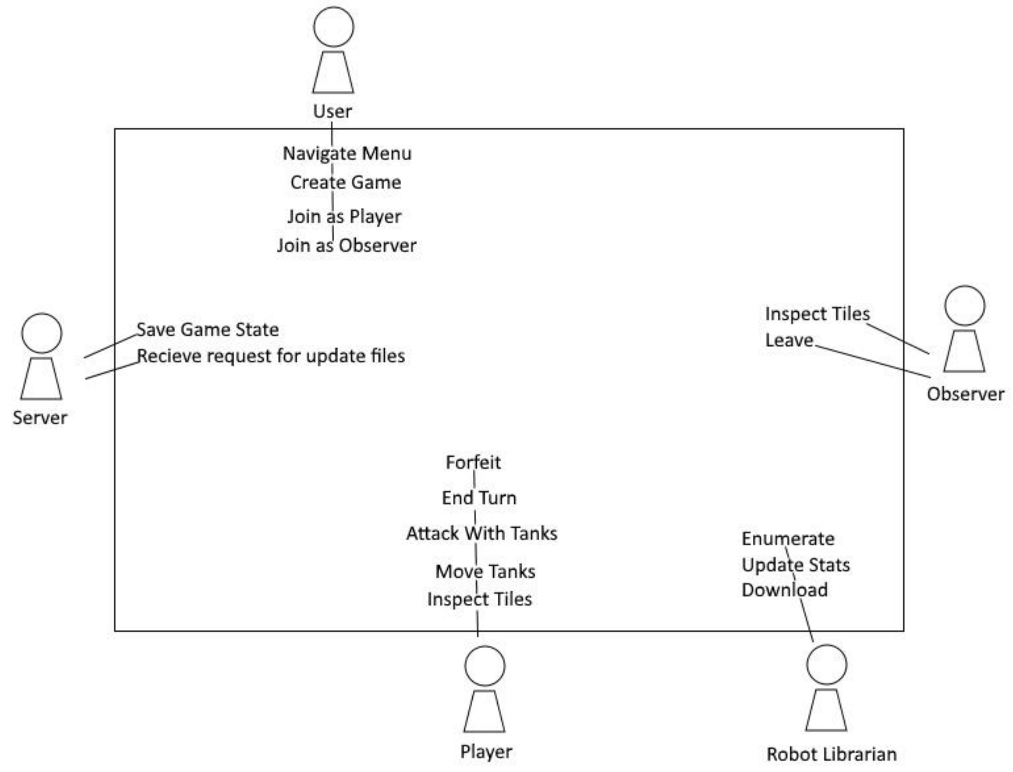


FIG 1.1: *The above diagram shows the actors for the system, as well as their actions. The actors include: User, Server, Player, Robot Librarian, and Observer. Their actions will be defined in detail in the Actors and their Actions section below.*

ACTIONS AND THEIR SCENARIOS

Below is an overview of actions and scenarios compatible with the system.

One of the actors in this system is the user this represents a person who has started the program and is interacting with the system. The user can take the actions: Navigate Menu, Create Game, and Join Game.

User: Create Game Scenario

When the user has chosen to create a game from the menu, they entered into a lobby, where they can choose how many computers players will participate while users connect. Once they have the required number of players, they may initiate a new game and the user becomes a player in the system. Should the player decide to cancel, they are sent back to the main menu.

Preconditions:	-The player has successfully started the program and can navigate the menu screen.
Post-conditions:	-The user is a player
	-A new instance of the game is created
Error-conditions:	-If the player does not have a network connection and chooses two or more human players, the game cannot be created and is not joinable by other human players.

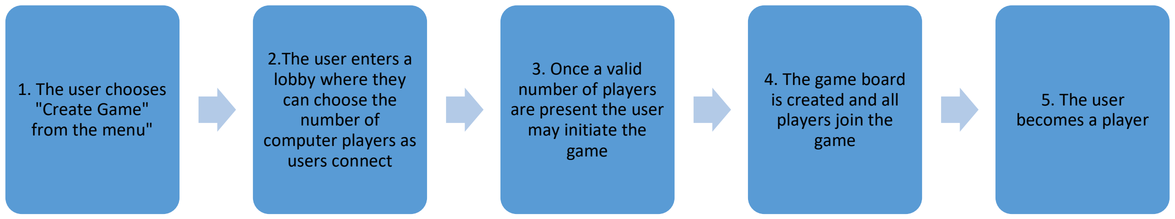

1. The user chooses "Create Game" from the menu"
2.The user enters a lobby where they can choose the number of computer players as users connect
3. Once a valid number of players are present the user may initiate the game
4. The game board is created and all players join the game
5. The user becomes a player

FIG 2.1: *The above figure shows the sequence of events taken by a user who has chosen to create a new game. Note that at any time, should the player decide to cancel this action, they are sent back to the main menu screen.*

User: Join Game

The user may also choose to join a game lobby. In this scenario, the user chooses the "Join Game" option from the menu screen. The user is then prompted to enter a form of match identification for the match they would like to join. If the user has a valid network connection, they enter the lobby as an observer.

Preconditions:	-A valid game must exist
	- The user and game must both have valid network connections
Post-conditions:	-The user becomes an Observer
	-The user has joined a match
Error-conditions:	-The user has no network connection

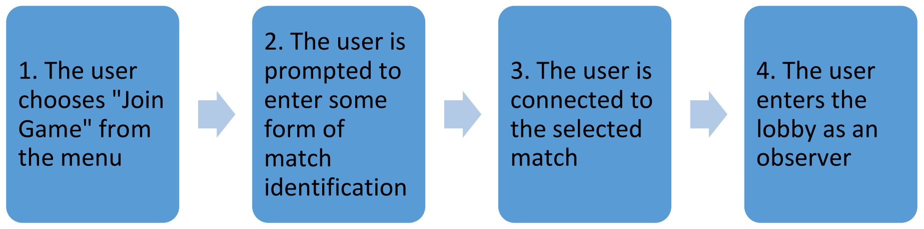

1. The user chooses "Join Game" from the menu
2. The user is prompted to enter some form of match identification
3. The user is connected to the selected match
4. The user enters the lobby as an observer

FIG 2.2: *The above figure shows the sequence of events when the user chooses to enter the lobby.*

When a user has successfully joined the lobby, once the game starts they become an observer. An observer sees the entire board and may take the actions: Inspect Tile, or Leave.

Secondary scenario: User – Join as Player

Should the user wish to be a player, they may click the switch button in the lobby screen to switch between being an observer or player if there is enough room for an extra player.

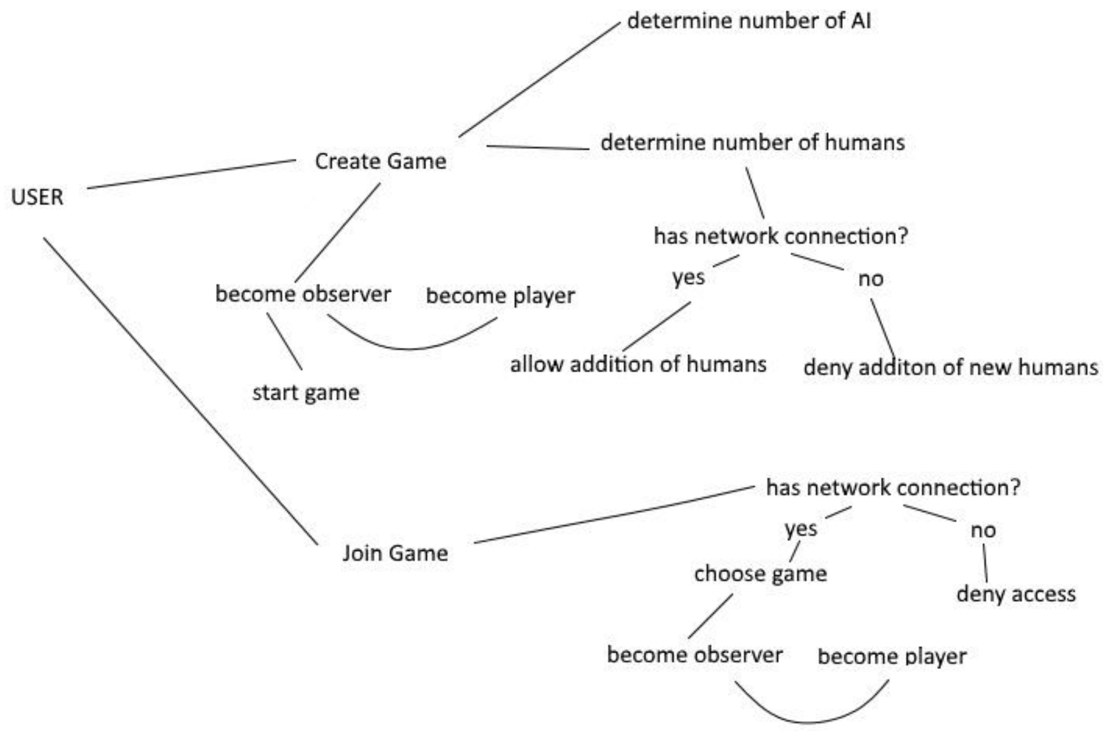


FIG 2.3: *The above diagram shows an overview of the user and the actions they can take. Notice that a user can Create or Join a game. Both actions result in the user becoming player.*

When a user has successfully joined the lobby, and registered as a player, once the game starts they become a player. The player is part of a match and as such can take turns. On their turn, they can take the actions: Forfeit, Inspect Tile, Move, Attack, and End Turn.

Player: Forfeit

When the user is part of a match, they become a player. Should the player desire to quit the game during an active match, they have the option to forfeit the game. They can do so by selecting the forfeit button, at which time they will be prompted with a confirmation dialogue. They may cancel the action and return to the game, or they can confirm the action and become an observer of the match. They can then leave the game at any time.

Preconditions: **-**The player is part of an active match

Post-Conditions: -The player becomes an observer and is no longer part of the match.

-The player's robots are removed from the game

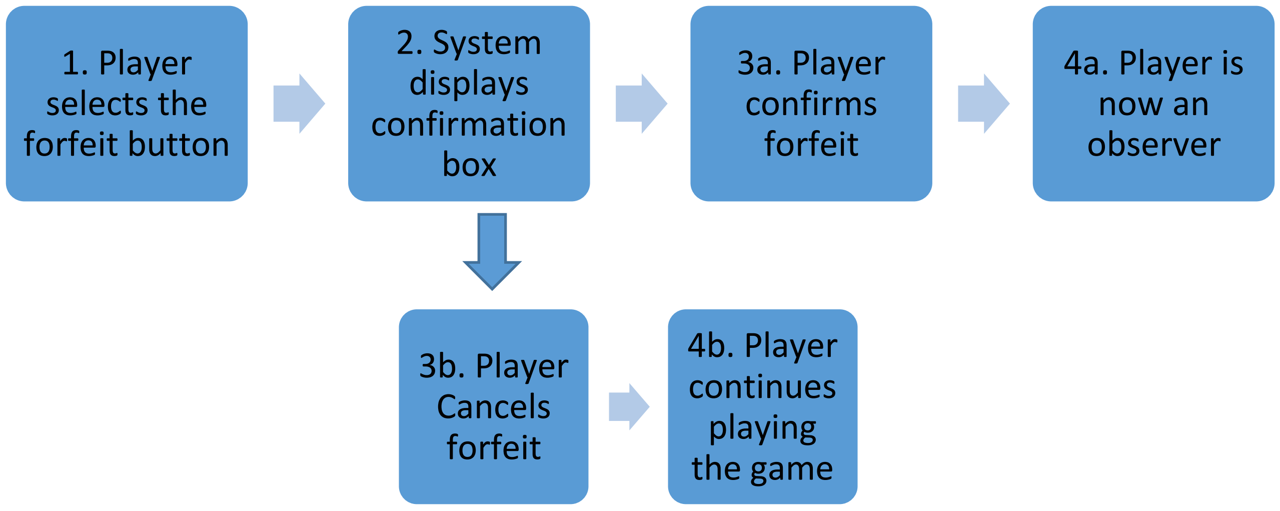

1. Player selects the forfeit button
2. System displays confirmation box
3a. Player confirms forfeit
4a. Player is now an observer
3b. Player Cancels forfeit
4b. Player continues playing the game

FIG 2.4: *The above figure shows the sequence of events taken by a player who considers forfeiting the game. When the player forfeits, they become an observer. They may also cancel their forfeit.*

Player: Inspect Tile

On their turn, a player may inspect a tile. A tile can only be inspected if it is in range of one of the player's robots. If the tile has any tanks on it, a window with a list of all robots on the tile and their stats will be shown to the player. If the tile is empty, the window will not be shown.

Preconditions: -The tile is within range of one of the player's robots

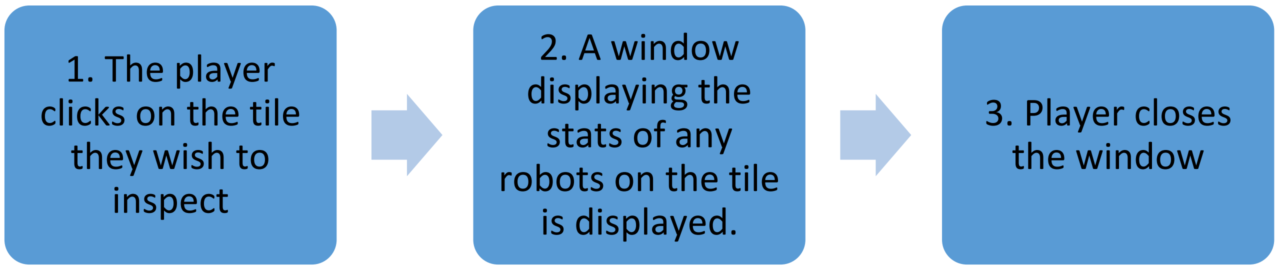

1. The player clicks on the tile they wish to inspect
2. A window displaying the stats of any robots on the tile is displayed.
3. Player closes the window

FIG 2.5: *The above figure shows the sequence of events the player takes when inspecting a tile.*

Player: Move

Along with inspecting a tile, on their turn a player may move one of their robots to a tile within range of the robot. They do so by selecting the move option and selecting a tile in range and confirming their choice via a confirm button. The robot will then move to the selected tile, and the line of sight for the player will be updated. Note that a player can repeat this action as long as the robot has not moved a number of tiles equal to its movement range.

Preconditions: -It is the player's turn

-The robot the player is moving has not moved its movement range

Post-conditions: -The robot has been moved to the tile selected

-The number of tiles traveled this turn is updated

to match the number of tiles the robot has moved

Error-conditions: -The player chooses a tile out of range of the robot

-The player has moved the movement amount for the robot

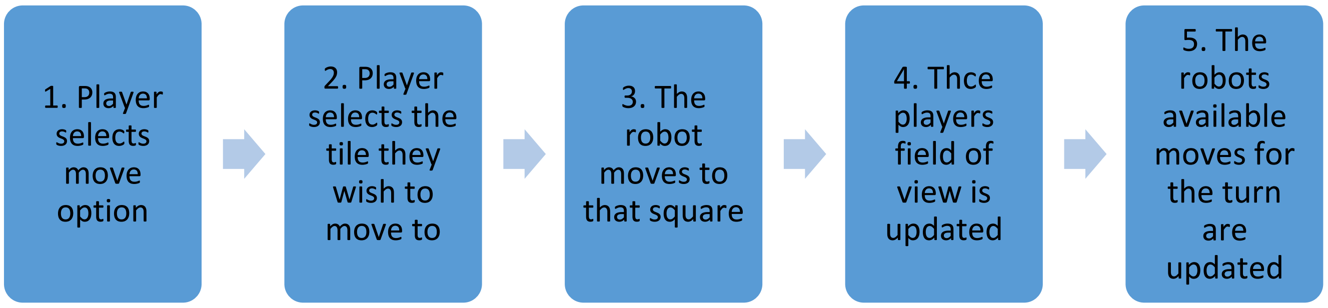

1. Player selects move option
2. Player selects the tile they wish to move to
3. The robot moves to that square
4. Thce players field of view is updated
5. The robots available moves for the turn are updated

FIG 2.6: *The above figure shows the sequence of events for a player who has chosen to move a robot. This action results in a robot being moved to the selected tile, and the player's field of view being updated. Note that if the robot has moved its movement range, the robot can no longer be moved.*

Player: Attack

Along with moving, a player can attack on their turn. The player can attack a tile if it is in the shooting range of one of their robots. The selected tile then takes the amount of damage the player's robot can give, and subtracts it from the health of all robots on the tile. Note that a player can move and shoot in the same turn, and can move after shooting, provided the robot's movement range has not been exceeded. Also, a player can damage their own robot.

Preconditions: -It is the player's turn

-The enemy robot is within shooting range of the player's robot

Post-conditions: -The enemy robot takes the amount of damage the player's robot can give

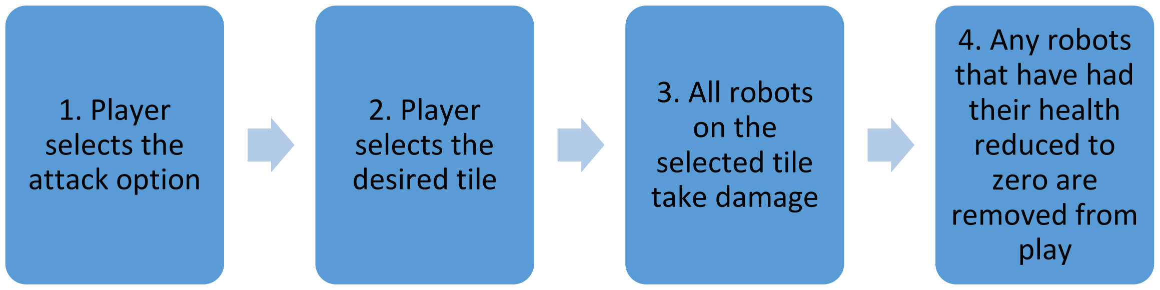

1. Player selects the attack option
2. Player selects the desired tile
3. All robots on the selected tile take damage
4. Any robots that have had their health reduced to zero are removed from play

FIG 2.7: *The above figure shows the sequence of events taken by a player who has chosen to attack. Note that a player can damage their own robot.*

Player: End Turn

Finally, a player can end their turn. This can be done at any time during a player's turn. The player chooses the "End Turn" button and will be prompted with a confirmation prompt. When the player confirms, their turn is over, and the next player takes their turn.

Preconditions: -It is the player's turn

Post-conditions: -The player's turn is over, and the next player takes their turn

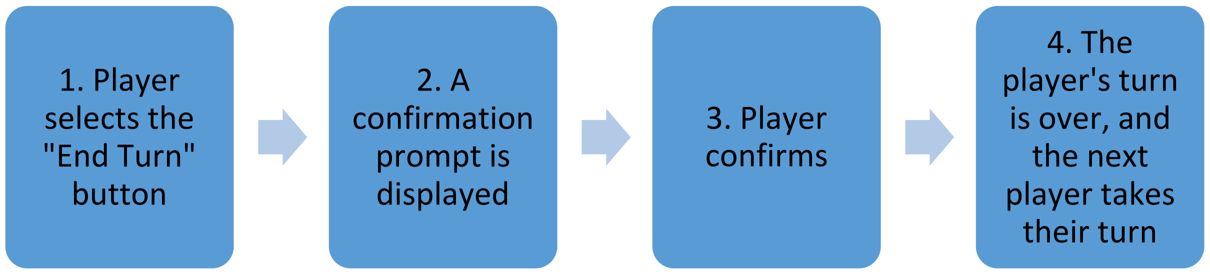

1. Player selects the "End Turn" button
2. A confirmation prompt is displayed
3. Player confirms
4. The player's turn is over, and the next player takes their turn

FIG 2.8: *The above figure shows the sequence of events taken by a player choosing to end their turn. This can be done at any time during the player's turn. The next player gets their turn after this action.*

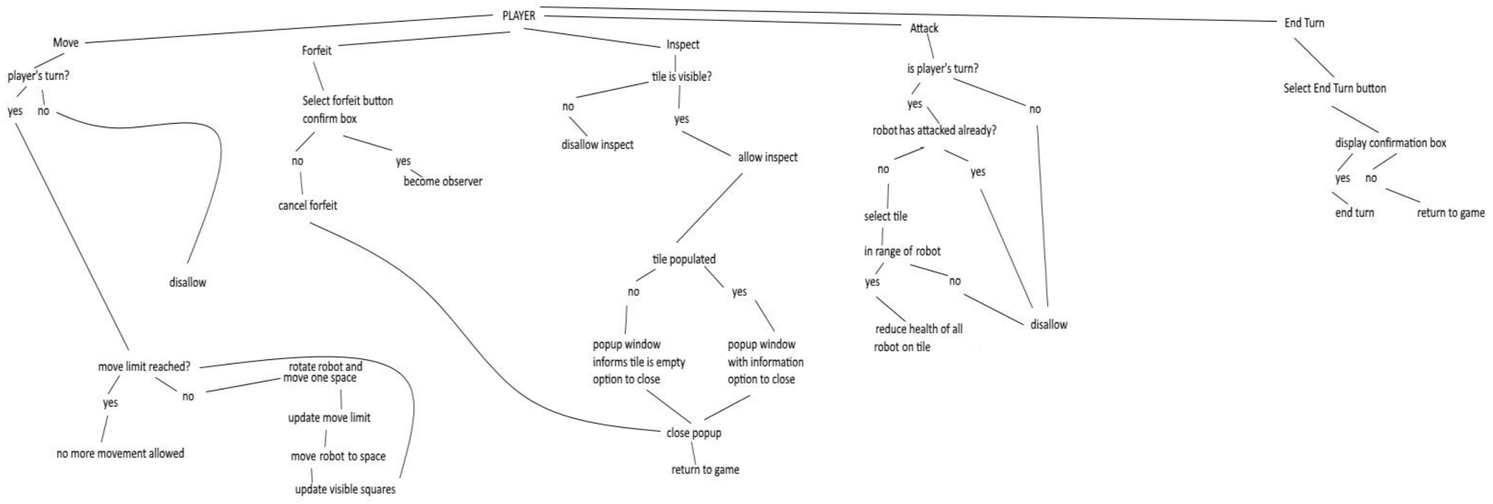


FIG 2.9: *The above diagram shows an overview of the Player actor. The player can take actions that progress the game, such as moving and shooting.*

The Robot librarian is another actor in the system. It is used to manage robot programs. The Robot Librarian can: Enumerate, Download robot programs, and Update Match Statistics after a match.

Robot Librarian: Enumerate

The Robot Librarian can query for robot programs by different fields such as: Team, Name, Wins, Matches Played, Win/Loss Ratio. The Robot Librarian can also display statistics.

Post-conditions: -The query returns the list of robots in a sorted order

Error conditions: -The robot programs cannot be accessed

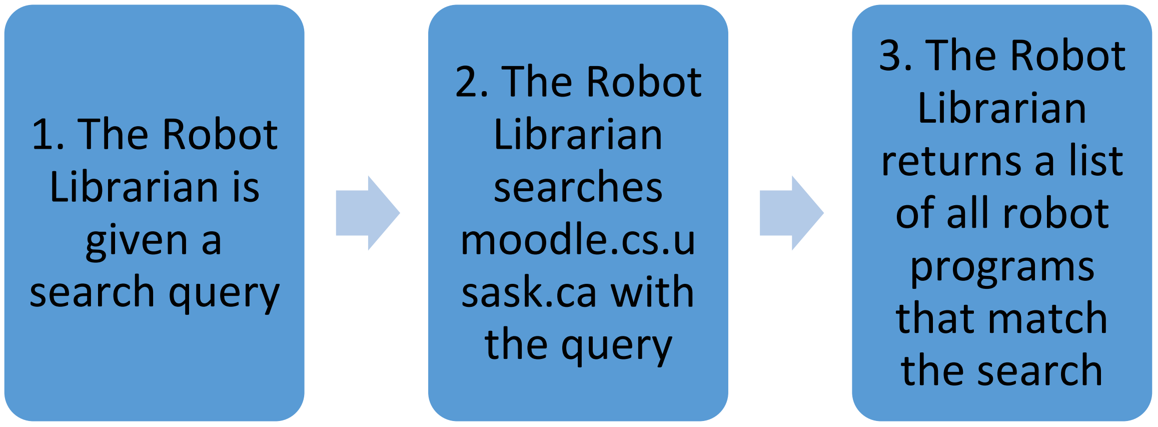

1. The Robot Librarian is given a search query
2. The Robot Librarian searches moodle.cs.usask.ca with the query
3. The Robot Librarian returns a list of all robot programs that match the search

FIG 2.10: *The above sequence diagram illustrates the sequence of events taken by the Robot Librarian when asked to search for robot programs.*

Robot Librarian: Download

The Robot Librarian can download robot programs so they can be run with the system.

Preconditions: -A robot program is selected

Post-conditions: -A robot program is downloaded into the system

Error conditions: -The robot programs are unavailable

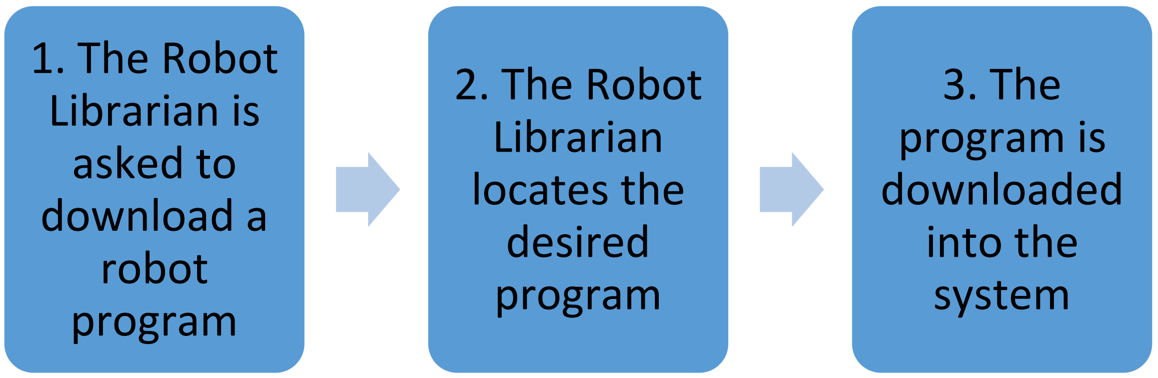

1. The Robot Librarian is asked to download a robot program
2. The Robot Librarian locates the desired program
3. The program is downloaded into the system

FIG 2.11: *The above sequence diagram shows the sequence of events taken by the Robot Librarian when it is required to download a robot program. The program can then be used by the system.*

Robot Librarian: Update Stats

The Robot Librarian is also able to update the stats for a robot program. After a match has finished, the Robot Librarian uploads the new stats for the selected program to moodle.cs.usask.ca.

Preconditions:	-A match has finished with the selected robot program
	-There is a valid network connection
Post-conditions:	-The stats of the robot program are uploaded to moodle.cs.usask.ca
Error Conditions:	-Data is unable to be sent to moodle.cs.usask.ca

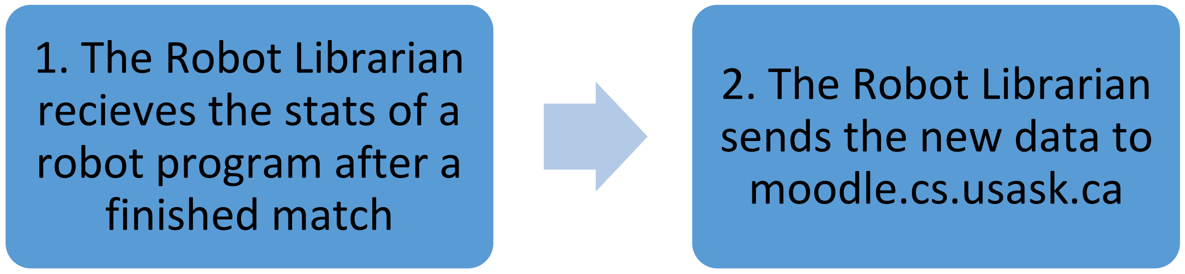

1. The Robot Librarian recieves the stats of a robot program after a finished match
2. The Robot Librarian sends the new data to moodle.cs.usask.ca

FIG 2.12: *The above sequence diagram shows the sequence of events for a Robot Librarian uploading new stats to moodle.cs.usask.ca.*

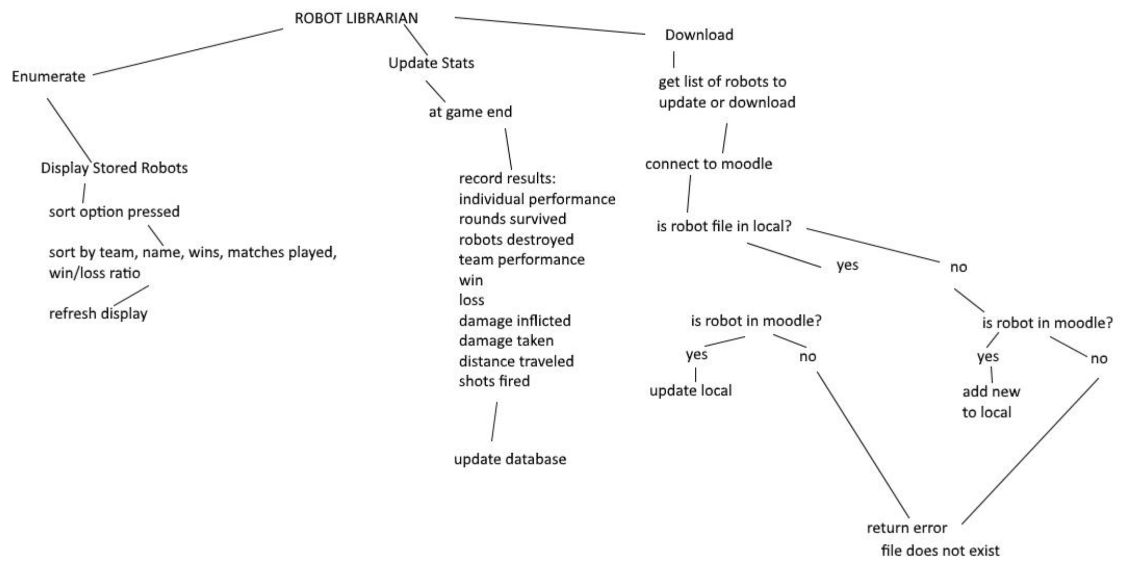


FIG 2.13: *The above diagram shows an overview of the Robot Librarian. The main purpose of the Robot Librarian is to keep track of robot programs for the system, and update their stats when necessary.*

The server is the actor that handles communications between multiple machines for network gameplay. It can: Save the Game State and Receive and Respond to Requests for updates.

Server: Save Game State

The server periodically saves the state of the game. This allows it to send updates to other players on request.

Preconditions:	-There is an active game
Post-conditions:	-The state of the game has been saved to the server.
Error-conditions:	-The server loses connection to the game
	-The game is corrupted upon saving

Server: Receive and Respond to Update Requests

The Server is able to receive and respond to update requests. A player's computer will periodically request updates of the game state from the server. The server can respond to these requests with an updated instance of the current match.

Preconditions:	-The server is connected to a game
Post-conditions:	-The server has sent an updated instance of the game to the player
Error-conditions:	-The game data is corrupted during transmission
	-A player disconnects from the match

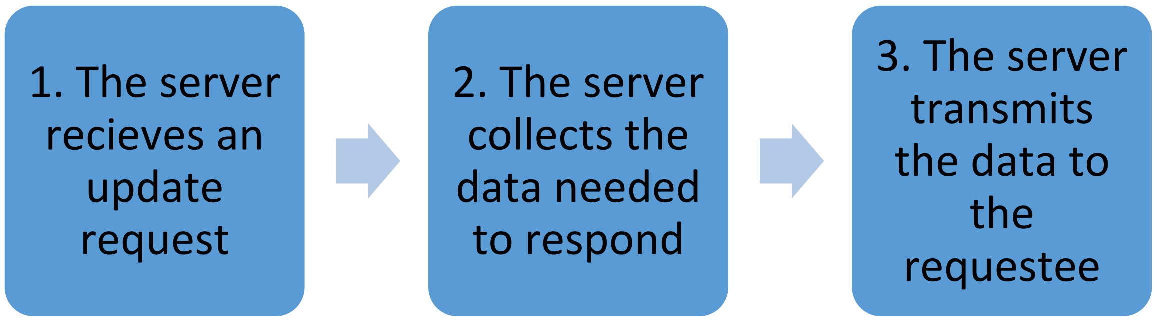

1. The server recieves an update request
2. The server collects the data needed to respond
3. The server transmits the data to the requestee

FIG 2.14: *The above sequence diagram shows the steps taken by the server when it has received an update request from a player's computer.*

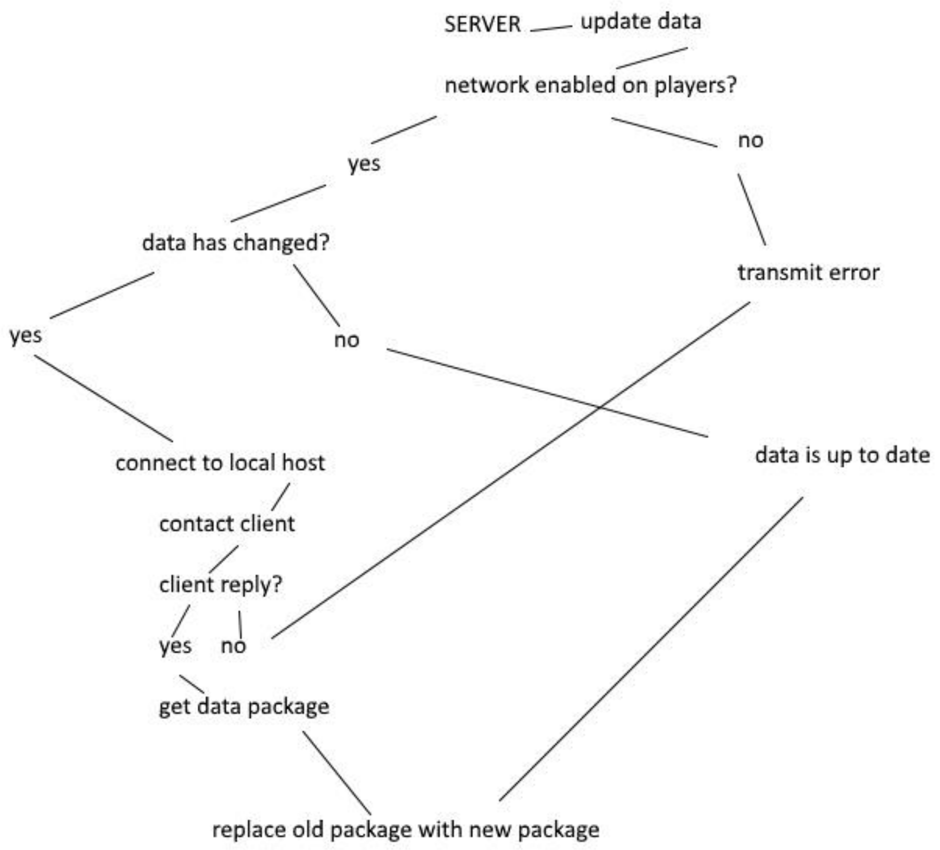


FIG 2.15: *The above diagram shows an overview of the server. The server is in charge of updating all computers with the most recent game states.*

The observer is the last actor in the system. It is similar to a player, but cannot directly influence gameplay. The observer can see all tiles on the board and the robots on them. The observer has the ability to: Inspect Tiles, and Leave.

Observer: Inspect Tiles

An observer may inspect a tile. If the tile has any tanks on it, a window with a list of all robots on the tile and their stats will be shown to the observer. If the tile is empty, the window will not be shown.

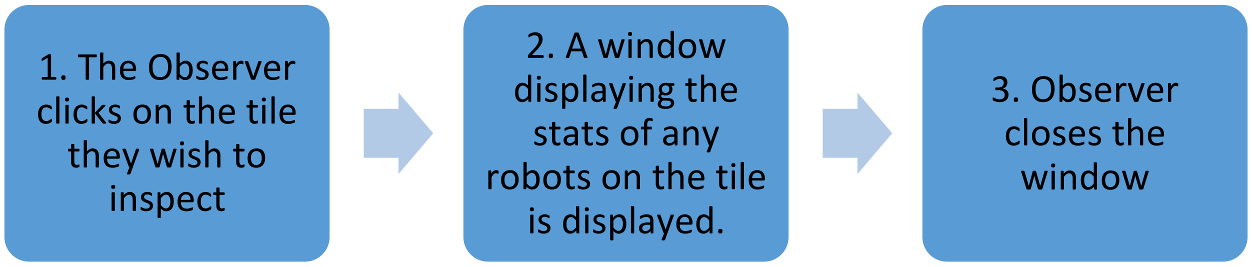

1. The Observer clicks on the tile they wish to inspect
2. A window displaying the stats of any robots on the tile is displayed.
3. Observer closes the window

FIG 2.16: *The above figure shows the sequence of events the Observer takes when inspecting a tile.*

Observer: Leave

When the user is part of a match, but not a player, they become an observer. Should the observer desire to leave the game during an active match, they have the option to leave the game. They can do so by selecting the quit button, at which time they will be prompted with a confirmation dialogue. They may cancel the action and return to the game, or they can confirm the action and be returned to the starting menu. They can then leave the game at any time.

Preconditions: -The observer is watching an active match

Post-Conditions: -The observer is returned to the opening screen

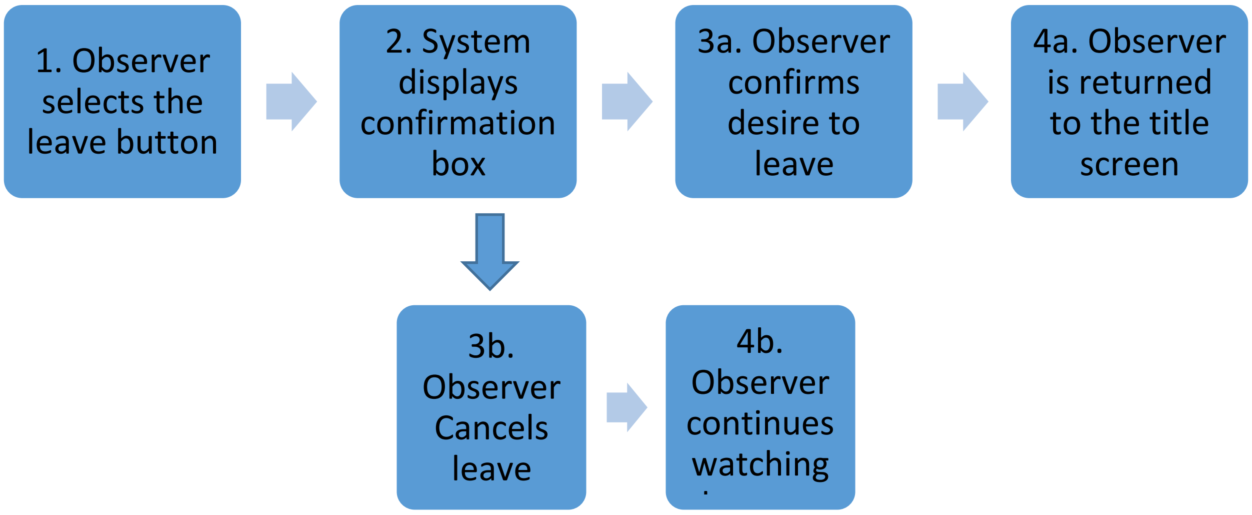

1. Observer selects the leave button
2. System displays confirmation box
3a. Observer confirms desire to leave
4a. Observer is returned to the title screen
3b. Observer Cancels leave
4b. Observer continues watching

FIG 2.17: *The above figure shows the sequence of events taken by an observer who considers leaving the game. When the observer leaves, they return to the opening screen. They may also cancel their leave.*

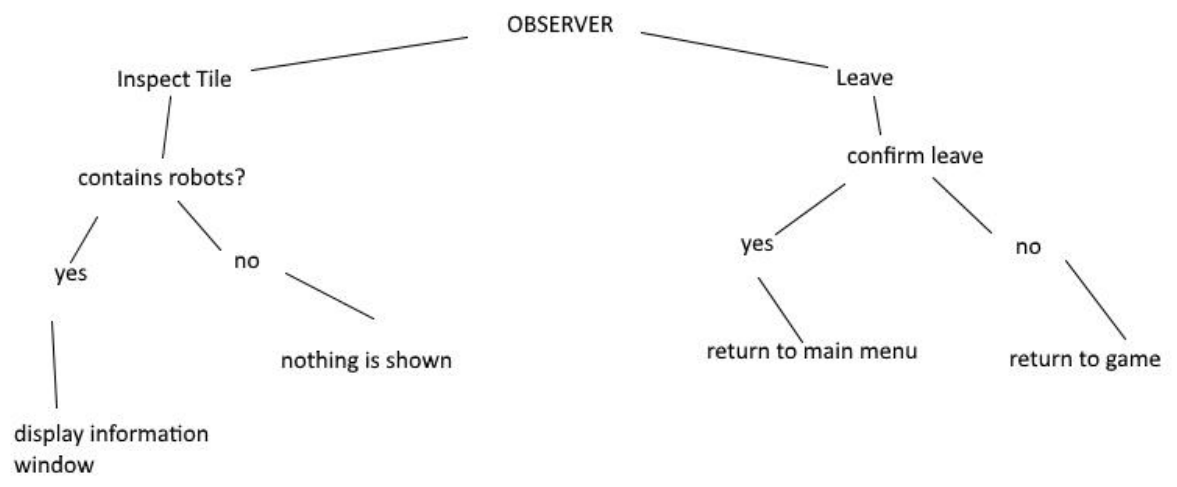


FIG 2.18: *The above diagram shows an overview of the observer. An observer is a human player who does not interact directly with gameplay. They can see the whole board and inspect tiles, but are not on a team and cannot move robots.*

Mock Ups

Mock Up: Start Screen

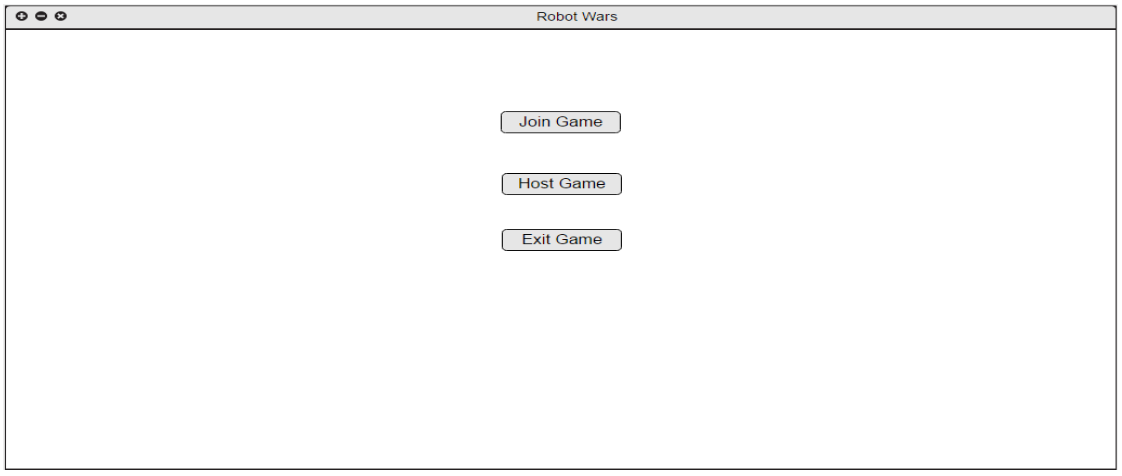
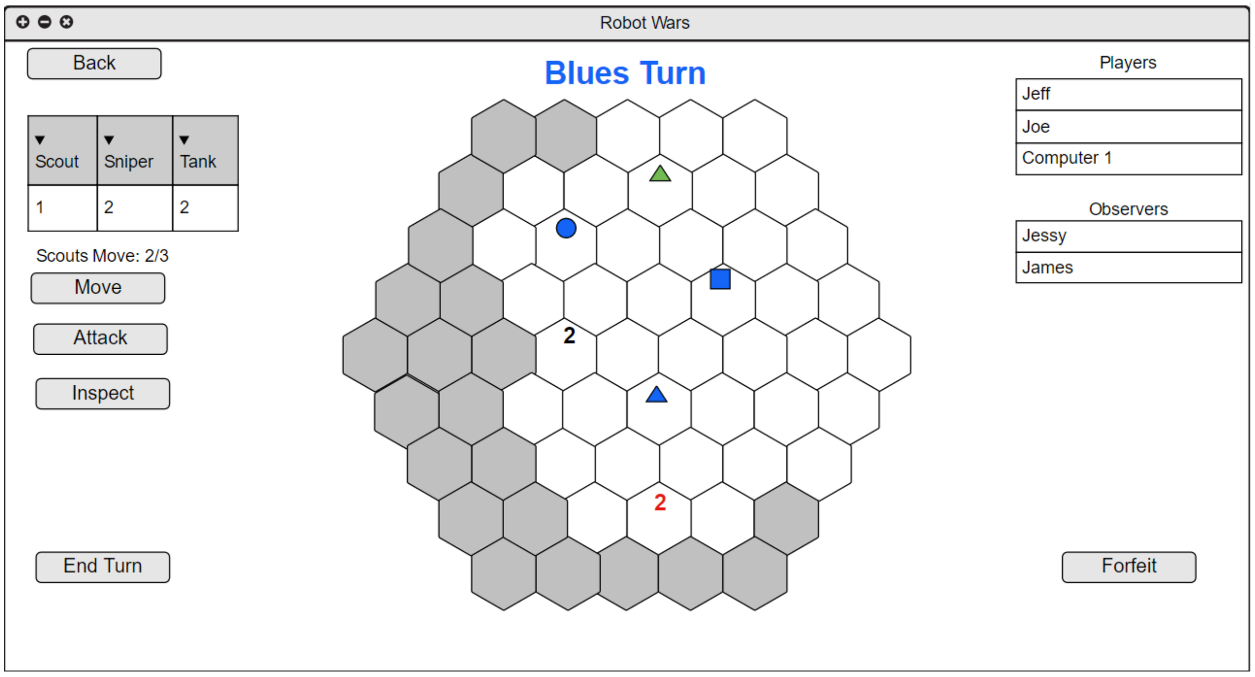


FIG 3.1: *The above diagram shows the Start Screen the user first sees when accessing our Robot Game. They will see three buttons right in the middle which are "Join Game", "Host Game", and "Exit Game".*

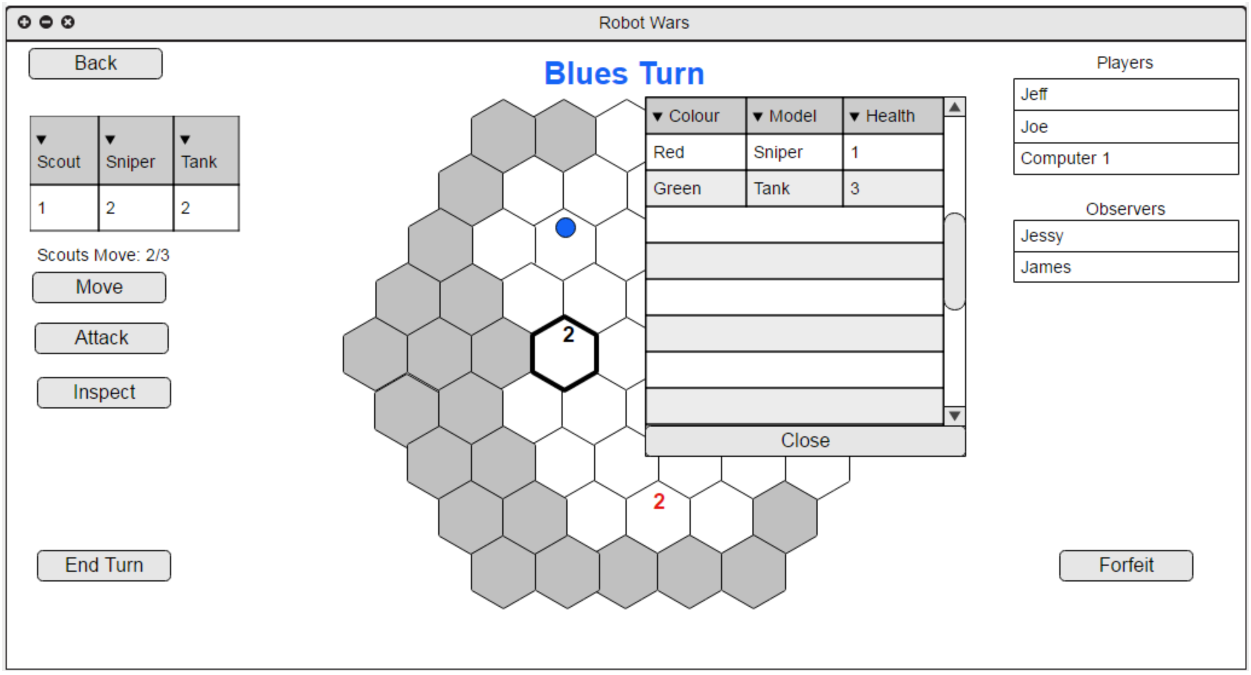
Mock up: Game



▾ Scout	▾ Sniper	▾ Tank
1	2	2

FIG 3.2: *The above diagram shows the current board as seen by the blue player. White hexes are board spaces that are within range of the blue player and grey spaces are out of vision. UI is shown on both sides of the screen with player moves and statuses on the left side and current players on the right side. Blue's robots will always be shown as singular pieces on their position of the board. Enemies will be shown as individual pieces if they are solo on a hex, and shown as a number with their corresponding color if there are more than one. If more than one enemy player is on a square it will be shown as a black Number.*

Mock Up: Inspect Tile



▾ Scout	▾ Sniper	▾ Tank
1	2	2

▾ Colour	▾ Model	▾ Health
Red	Sniper	1
Green	Tank	3

Players
Jeff
Joe
Computer 1

Observers
Jessy
James

FIG 3.3: *The above figure shows what the player sees when they choose to inspect a tile. Once a tile has been chosen, a menu showing the information about the tanks on that tile is shown.*

Mock Up: Lobby Screen

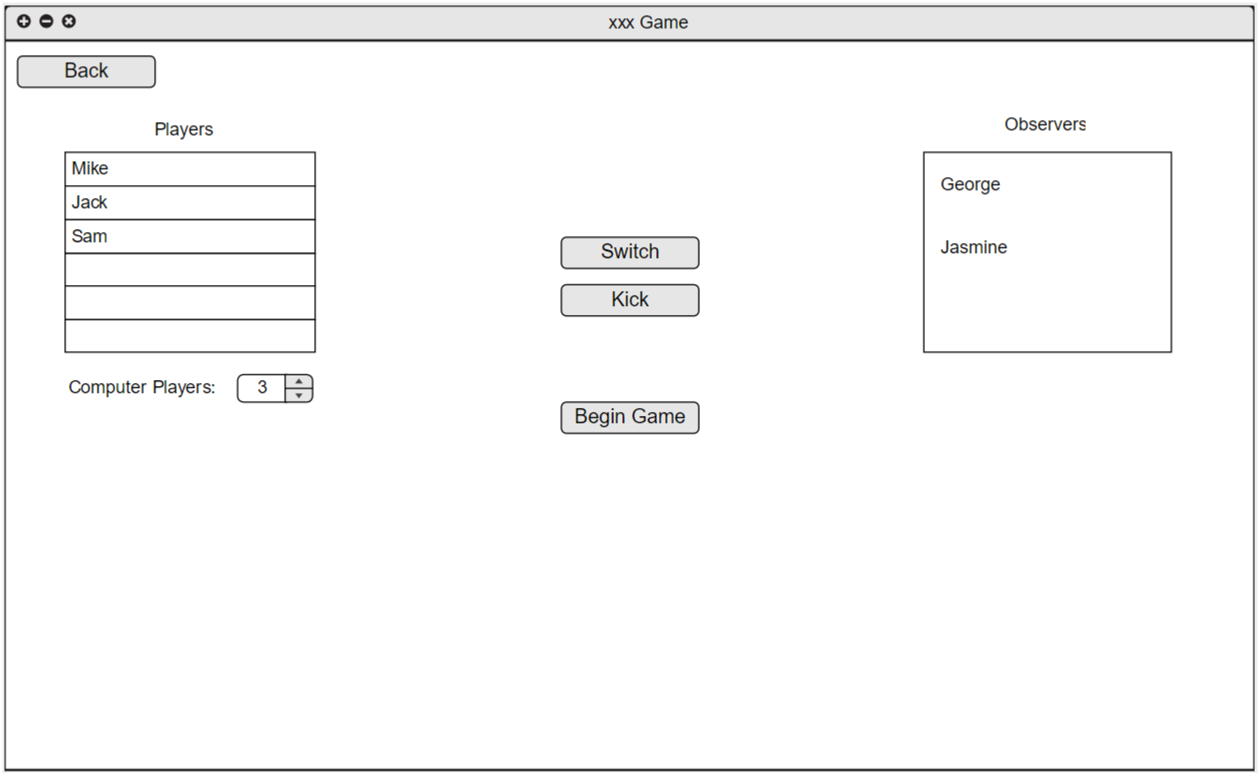


FIG 3.4: *The figure above shows what the user sees when are in the lobby screen. They are able to choose to be an observer or a player. They may also determine the number of AI players are to be in the match. When the begin game button is selected, the match is begun.*

PLATFORM

The available platforms for the Robot game will be on WINDOWS and Linux machines with the benefit of facing each other cross Platform. The Game will be allowed to run in tuxworld as well. We will be programming in Java and using Eclipse for editing our code. Git will be our version control software.

SUMMARY

The Robot game is a turn based multiplayer game with three different robots. The Robot Game provides a more comfortable experience than the usual tabletop experience with features like Visibility System, online battle, AI opponents, Cross platforming, multiple players and strategic robots. To achieve this, we will use teamwork, Java, and the basic Software Engineering Method.

We have important actors such as the User who can Create a game and Join a game, the Player who can Forfeit, inspect a tile, move, attack, and end turn, The Robot Librarian who can download enumerate and update stats, the server which Receives and Responds to update requests and Saves the Game state, and the observer which inspects Tiles, and leaves the game. Each part plays an important role in the use of the Robot Game.