Group D2

Robo-Sport 370

A CMPT 370 Project

Michael Graham, Arianne Butler, Kristof Mercier, Chris Mykota-Reid, Samuel Horovatin

October 2, 2016

Contents

Glossary

Computer player: a computer AI that is controlling one of the teams in the game

Human player: a human player that is controlling one of the teams in the game

Player: a human player or a computer player

Robot: a moveable game object of which there are three per team

Scout: the robot player with the highest range of movement

Sniper: the robot player with the highest visual/firing range

Tank: the robot player with the highest attack strength and health

Team: a team of three robots, all of which are controlled by either a human player or a computer player

Tile: a hexagon shaped unit that makes up the game map

Introduction

Our software engineering team is Arianne Butler, Kristof Mercier, Samuel Horvatin, Micahel Graham, and Christopher Mykota-Reid. We have come together to complete the Software Engineering project assigned to us by Dr. Christopher Dutchyn for Computer Science 370. All of our members are either in their third or fourth year of university at the University of Saskatchewan. Our entire project will be overseen by Jordan Ubbens, the teacher's assistant assigned to our group.

Our group's vision is to create user-friendly game that is both enjoyable and challenging. This will require an interface that is both visually pleasing and discoverable. We will achieve this by using easy-to-use controls and simple menuing systems. In order to create a game which challenges a human player, we will implement a computer controlled AI with the capability of beating both human players and other computer controlled players. Human players may also compete against other human players in a hot-seat, pass-the-controller like fashion. Finally, we wish to provide the player with useful information to better improve their skills in the game. We will achieve this through various in-game statistics and an end-of-game breakdown of each team's failures and achievements.

Game Description

Our task is to design and create a version of the pre-existing, computer based board game called Robo-Sport. The game can be played with two, three, or six teams of robots; each team with their own color. Any of the teams may be controlled by either a computer player or a human player. The game map takes the form of a large hexagon, the shape of which is made-up of a collection of smaller hexagon tiles (see figure 1). The size of the game board (or the amount of tiles) depends on the number of teams in the game. Each team begins at the corner of the hexagon which matches their team's color.

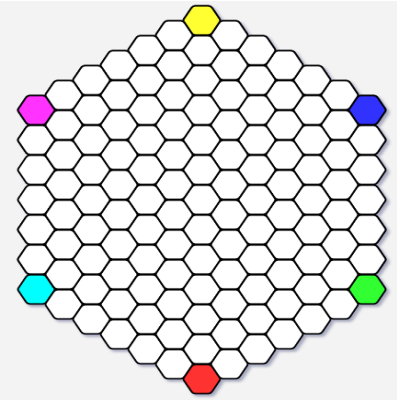


Figure 1: The game map is a large hexagon made up of smaller hexagons. The size of the game board is dependant on the number of players in the game. Each team's starting point is located in one of the hexagons corners.

Each player controls a team of three robots: a scout, a sniper, and a tank. Each of the three robots has its own unique statistics regarding health, movement, range, and severity of attack. Each of the robots are limited by a set visual field, enabling them to see either one, two, or three hexagons in any direction. This visual field is equal to the individual robot's range of fire. Although each of the robots has its own set of behaviors and statistics, they share the primary goal of finding and attacking the other team's robots.

Each type of robot can withstand a different number of attacks. Once a robot has been attacked to its limit of health, it is removed from the game board and the game continues until only one team is left. The last remaining team on the game board is the winner. If the game continues for a certain predetermined amount of time, the game ends in a draw.

Game Rules

The game rules and control flow will vary slightly depending on the amount of players in the game. If more than one human player is participating the game will support a shared-controller multiplayer format for two, three, or six players. A team can consist of all human players, all computer players, or a mixture of both. For a two or three player game each side of the hexagon will be five tiles in length. For a six player game, each side will be seven tiles in length.

On the game board, each team begins with all of their robots on their own starting tile. The starting tiles are located at the corners of the game board. Turns commence with the red team and move in a clockwise direction until all teams have taken a turn. For every turn, only one of the three robots may move. Depending on which robot is in play, it may move up to three spaces in any direction. It is possible for more than one robot to occupy the same space at the same time. When a robot is within its shooting range from another robot, it may shoot. The range for each robot is specified by its range statistic. When a shot is fired at a given tile, damage occurs to all units residing on that tile. A robot is capable of both moving and attacking in the same turn. If a robot attacks before it has used all of its moves, it may move again after the attack.

System View

The system view will begin with the main start-up screen (see figure 2). This screen will contain the game's title, a background image, a display of high scores, and three buttons. The buttons are the *Start* button, the *Instructions* button, and the *Exit* button. The *Instructions* button will bring up the set of game rules and instructions, the *Exit* button will close the program, and the *Start* button will bring up the Player Selection screen (see figure 3): a small window where the user may select the number of players in the game. Once the user has selected a two, three, or six player game, they may click *Start*. Clicking *Start* will prompt the Team Profile Screen (see figure 4).

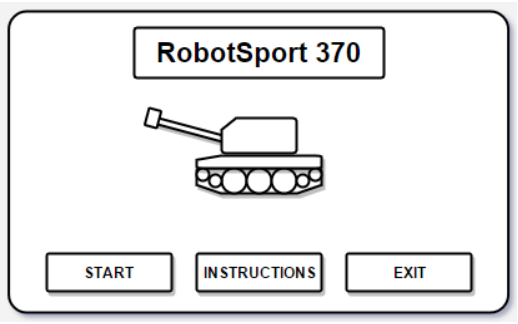


Figure 2: The Main Start-Up Screen contains the game title, a background image, and three buttons. The *Start* button starts the game, the *Instructions* button opens a set of game rules and instructions, and the *Exit* button closes the program.

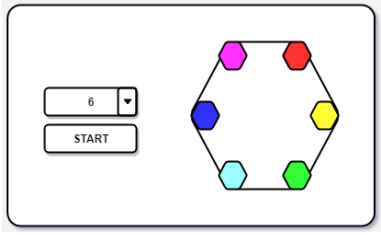


Figure 3: The Player Selection Screen allows the user to select the amount of players. Based on the amount of players that are selected, the Player Selection screen will show where the players will be placed on the game map. This gives the user an idea of how the game map will look.

The Team Profile Screen will contain a box for every player (i.e. three boxes for a three player game). At the top of each box will be two mutually exclusive options, where the user will indicate whether they want that player to be human or computer. Below this selection will be a list of all of the previously created teams. Hovering over a team will produce the

game statistics for that team (see Figure 5). The user may edit a team name if they choose. The user will also be given the option of adding a new team. If they decide to add a new team, they will be prompted to choose a unique team name. When they save the new team, the game statistics for that team will be set to zero. Once team selection is complete, the user may click either *Start* or *Back* to either start the game or return to the start-up screen, respectively.

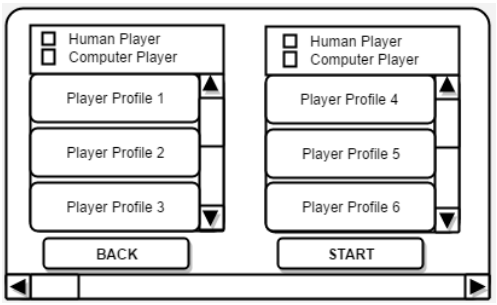


Figure 4: The Team Profile Screen contains a box for every player. The user will select whether the player will be controlled by a human or by the computer. They will have the option of selecting a previously created team with a saved set of statistics, or making a new team with all statistics set to 0.

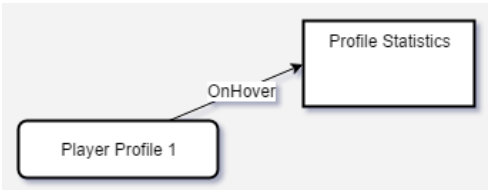


Figure 5: Game statistics are produced by hovering over an existing team in the Team Profile Screen.

Once the user starts the game, the size of the game map will adjust according to the number of players. The game board consists of two main sections (see figure 6). The first section is the restricted view of the current player's game map. The second section consists of two buttons for move and attack, as well as game statistics for the current game (i.e. robots left in the game, remaining health, range, etc.). As the turn moves from one player to the next, the game board will display a waiting screen which waits for confirmation that the next player has taken their place at the monitor. Once confirmation has been given, the window will change to the view of the current player.

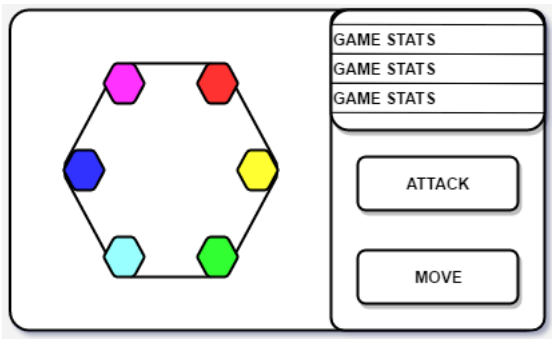


Figure 6: The game board has two sections. The first is the game map and the second is the current game statistics and the Move/Attack buttons.

When the game ends, a display of the updated game statistics will appear overtop of the game map. It will contain updated statistics for all of the teams, as well as the individual robots (i.e. shots fired, incurred damage, etc.). The user may choose to save and/or exit at any point throughout the game.

User Input

User-input will be handled exclusively by the mouse. The game will be equipped with buttons and drop down menus to make interaction with the system both intuitive and discoverable. All buttons will contain descriptive names, as well as descriptive labels where more information is required. To move or attack, the user will click the aforementioned *Move* and *Attack* buttons on the right hand side of the game board. The user will then click the tile on the game board that they wish to perform the *Move* or *Attack* action on. This will result in either the player's robot moving to or attacking the selected tile, or an error message informing the user that they are attempting to make an illegal action. If a tile is within

the current robots range of vision, the user may hover over it to instantiate a dialogue box containing statistics on the robot(s) currently residing on that tile. If there is only one human player in the game, the *Move* and *Attack* buttons will fade to grey once they have finished their turn. They will remain faded until their next turn.

Data Output

Data will be displayed in several ways throughout the system. On the main screen, the teams holding high scores will be displayed. Statistics on various robots in range of the current player are made available by hovering the mouse over a given tile. Statistics (other than current location) for all robots currently in play will be visible at all times above the *Move* and *Attack* buttons in the game-view. At the end of the game, updated statistics will be displayed. Here, the players will be able to see the overall breakdown of each of the team's behaviours.

Must-Have Priority Queue

- Multiplayer functionality
- AI to control computer players
- Forth Interpreter
- Mapping system
- Win conditions
- Menuing System
- Input/Output

System Architecture

The architecture style that our system will use is Model-View-Controller (see Figure 7), because it provides the best trade-off between functionality and simplicity of design. Each player in Robo-Sport must be able to view the board, control their robots, and access the robot database, all of which are encompassed by the main facets of the Model-View-Controller architecture. The strength of the architecture is in its ability to separate and objectify complex sections of the system. This allows for the editing of one section without affecting the rest of the system. The challenges that will need to be addressed in order to achieve successful implementation will revolve around database access. Multiple parts of the system will need access to the information stored in the database.

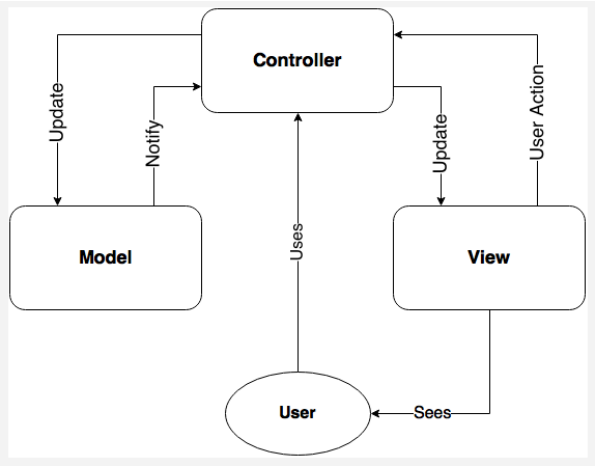


Figure 7: The Model-View-Controller Architecture for our system. The player uses the controller to play the game in the model. The view gives the human player a graphical interface of the model. The computer player uses the controller but doesn't require the view. The robot librarian can receive data and upload libraries from the model via the controller.

Platform Requirements

The minimum hardware required is as follows:

- OS: Windows 7 or higher
- Processor: Intel® Xeon® Processor E5-2690 v3 (30M Cache, 2.60 GHz)
- Memory: 2 GB Ram
- Storage: 500 MB available space

The Recommended hardware required is as follows:
- OS: Windows 7 or higher
- Processor: Intel® Xeon® Processor E5-2690 v3 (30M Cache, 2.60 GHz)
- Memory: 4 GB Ram
- Storage: 1 GB available space

Software requirements are as follows, but not limited to:
- Git
- Intellij IDEA
- TBD Language Specific Compiler

The above platform description is based on minimum requirements without outlining implementation specifics or programming language details. All of the above are subject to change when more specific implementation is decided upon.

Why a Computer Game?

At first glance, Robo-Sport might seem like a board game that doesn't need a computer, and while this is mechanically true, there are many benefits to having a digital version of Robo-sport. The first and most important is the fog of war. If implemented as a physical board game, fog of war would be impossible. On the computer however, opponents game locations can be hidden, adding another layer of depth. Secondly the AI could not be implemented for a board game. Thirdly the stats would be very difficult to record. By creating the game on a computer, we can automatize the collection and distribution of stats on each individual robot quickly and efficiently. No scribe or record keeper (besides the computer!) is needed. Lastly, rules can be strictly enforced. This could be seen as a double edged sword, since there will be methods in place actively preventing people from playing the game any differently than it is designed. For consistency sake however, a computer implementation is far superior to the physical one.

Actors

Actors include all individuals and objects that interact with the system. This would include things like people as well as computerized opponents. There will be three actors in our system:

Robot Librarian's

The robot librarian is responsible for managing teams as well as computer players. It communicates with a server filled with robots so that new ones may be added to the game. It can retire old robots, and also revise their code to make them more challenging.

Computer Player

The AI of our game will be a computer player that autonomously controls its units. Because it's not a human it needs its own set of functions as it will be interacting with the game in a much different way as opposed to how a human would. The computer will need to be able to SCAN for information about tiles surrounding its robot, as well as have primitive commands like TURN, MOVE and ATTACK. By providing these actions, the computer player should be able to function at an adverse level and provide a real challenge for human and robot players alike.

Human Player

The human player is person who can control their robots via a game board. They can view the full game board at any point in time, but tiles outside of their robot's range will not be visible. Because of this, they can interact with the game in a different manner than that of the computer player. More specifically, they can hover their mouse over tiles which will provide them information about the units that are currently on it (if they are on it). From there, the player can either move their robots to a specific tile or attack that tile.

Actions

Actions are specifically what the object/person can do to interact with the system. The system will respond according to what action an object/person has performed.

Computer Player

Computer Actions	**System Response**
MOVE	Moves the robot forward
TURN	Changes which direction the robot is facing
SCAN	Tells the robot information about the viewable tiles (ie. What tiles have robots on them). Will not provide information on tiles that robots cannot move to.
SHOOT	Damages all robots on the tile that the robot attacked

Robot Librarian

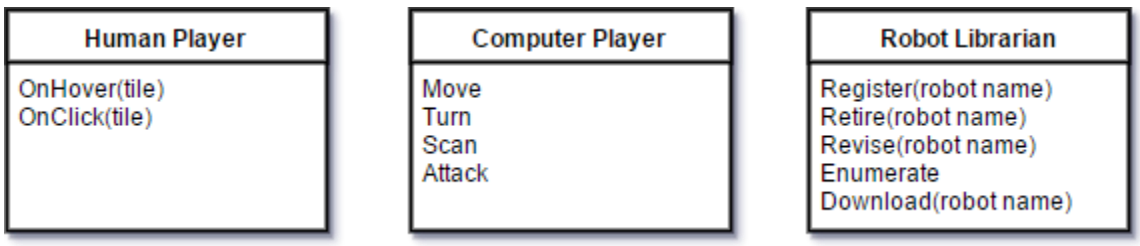
Robot Librarian Actions	**System Response**
REGISTER	Adds a new robot to the store of robots
RETIRE	Freezes a robot so you can reuse the name
REVISE	Replace an existing robot with new code
ENUMERATE	Displays all robots that are currently stored
DOWNLOAD	Gets a robot from the server so that it can be used in game Ex: Download the robot named “Megatron” to use in the game.

Human Player

Player Actions	**System Response**
HOVER ON TILE/ROBOT(S)	Provides player with information on the selected tile/robot(s) on a window that appears upon mouse over of the tile/robot(s)
ATTACK/MOVE	Moves the robot to the selected tile or attacks it based on which button (MOVE/ATTACK) is selected

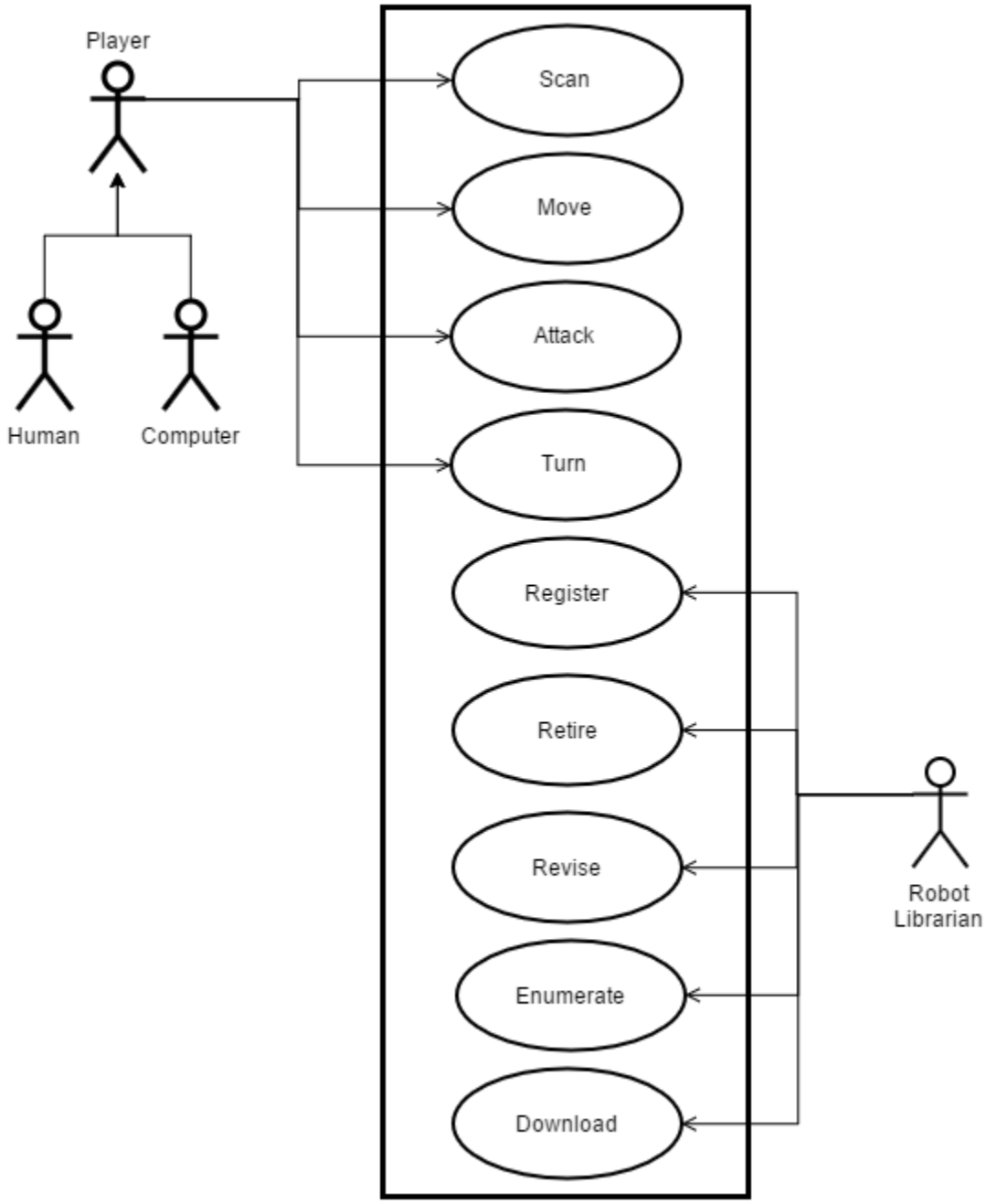
Interfaces

Interfaces are used to show what sort of actions an object/person can do.



Pictured above: The actions that each object can do, and the information required for each action to occur. These are also outlined in the action section

Actor-Actions Diagram



Pictured above: The actors and their actions

Scenarios

KEY: [A] = Alternate (Sub) flow, [E] = Error conditions [SS] = See sub/secondary scenario

Primary Scenarios

1.1 One Complete Game

A game is completed once there is only one team remaining or when the time left in the game reaches zero. At the end of the game the statistics are uploaded to a server so that the overall statistics can be analyzed.

Preconditions:

- There are at least two players on opposing teams who still have robots alive

Main Flow:

1. While there is at least two teams remaining
    a. A round of the game is completed [SS]1.3
    b. [A1]
2. [A2]
3. Statistics are recorded and uploaded to the server

Alternate Event Flow:

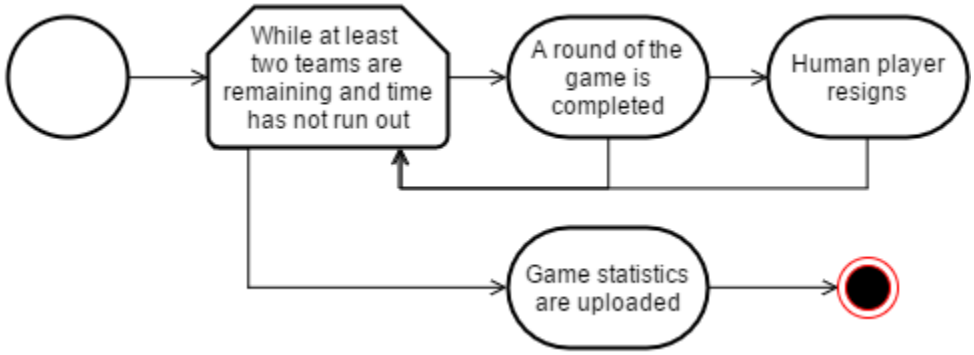
[A1]The human player(s) resign. This doesn't have to happen, but the rest of the game finishes without them

[A2]A player killed the last robot on the last opposing team. That player's team wins

Postconditions:

- Only one team remains **or** five minutes has passed (TIE)

Activity Diagram (see the Complete Game scenario for an explanation):



1.2 Setting up a Game

At the start of the game the player decides that they would like to select a new computer opponent. They can either create one through the robot librarian, revise one or download one from online.

Preconditions:

- The game has not started yet
- There may not be profiles stored online

Main Flow:

1. Player selects the number of players from the Player's Screen
2. The player opens the Team Profile Screen
3. While the player still has to select player profiles
    a. The player hovers over profiles from online to see the player's past stats
    b. [A1][A2][A3]

Alternate Flow:

[A1]They select a profile[E1]

[A2]They create a new player profile[E2]

[A3]They revise an existing profile[E3]

Error Condition:

[E1]Player tries to download a computer player that doesn't exist

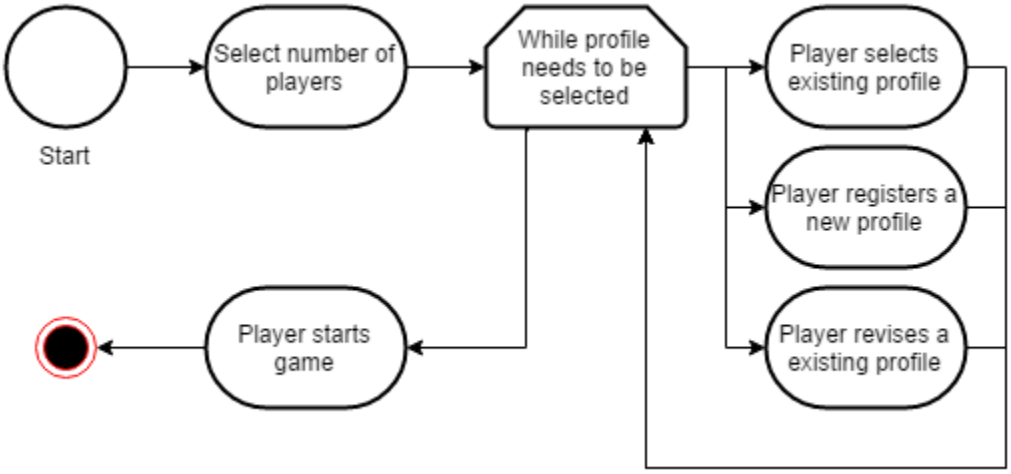
[E2]Player tries to create a computer player that already exists

[E3]Player tries to revise a profile that doesn't exist

Postconditions:

- Player profile(s) are added to the game and the game is now ready to start

Activity Diagram:



Secondary Scenarios

1.1 One Complete Round

As this game is a turn based strategy game each player will have the ability to move only one robot per round. The robot that is to be moved each round is in the order of: SCOUT, SNIPER, TANK where the scout is the fastest moving robot and the tank robot is slowest. The players are always iterated through in the same order each round.

Preconditions:

- No player has moved a robot on this round yet
- There are at least two players on opposing teams who still have robots alive

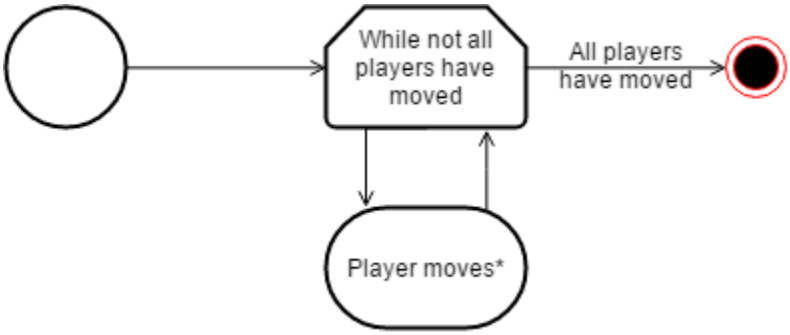
Main Flow:

1. While there are still players that have to move their robots and at least two teams remaining
2. The next player makes their move [SS]1.2

Postconditions:

- All players that are still alive have moved one of their robots

Activity Diagram (See One Complete Round scenario for an explanation):



1.2 Human Player's Turn

When the human player is told that it is its turn, the player decides on what tile to attack or move to based on the information that the game board shows. The player then selects which tile they would like to move to or attack and the system automatically updates their unit's position and/or the damage done to the tile that the player selected.

Preconditions:

- It's currently the player's turn
- The player still has robots that are alive
- There are robots on the enemy's team that are still alive
- The player has not moved a robot or attacked another robot yet

Main Flow:

1. The system tells the player that it is their turn. The game automatically highlights the robot that is to be moved and outlines the adjacent tiles that are within movement distance in one color m, and within range in another color r.
2. While the player's robot still has moves remaining the player:
    a. Decides what tile they want to move to or attack based on statistics provided by hovering their mouse over the tiles in range
    b. [A1][A2][E1]
3. The turn ends when the player hits the END TURN button or when their robot is out of moves.

Alternate Event Flow:

[A1]Player clicks on ATTACK then click on a tile within r to attack that tile

a. The robot is turned to face the direction that it should automatically
b. All units on that tile are damaged proportionally to the attack of the attacking robot.

[A2]Player clicks on MOVE then clicks on a tile within m to move to that tile

a. The robot is turned to face the direction that it should automatically
b. The robot's position is updated to the tile that was selected

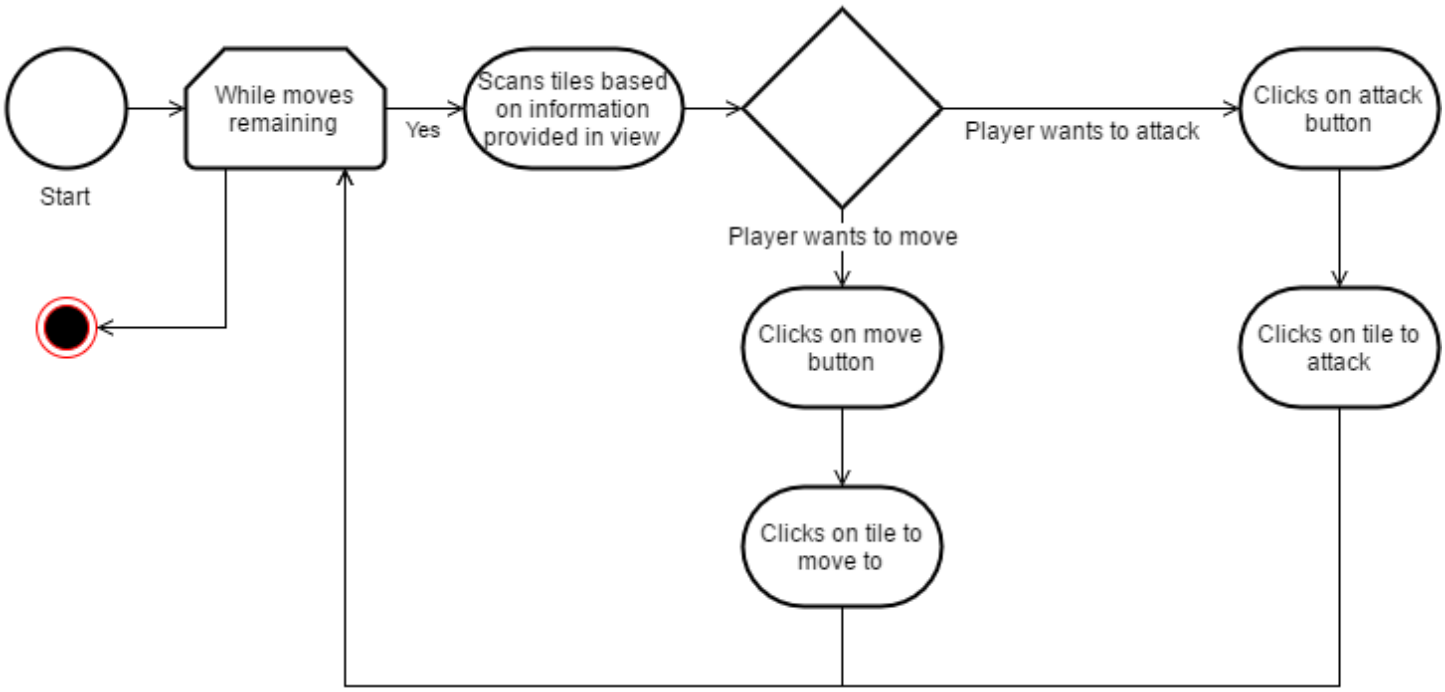
Error Conditions:

[E1]Player attempts to attack or move to a tile that is out of bounds. In this case, nothing occurs

Postconditions:

- The player's turn is over
- The player's robot has been moved and/or has attacked another robot
- The Human player's robot may be dead if they attacked themselves.

Activity Diagram (see the Human Player's Turn scenario for an explanation):



1.3 Computer Player's Turn

When the computer player is told that it is its turn, the computer decides on what tile to attack or move to based on the information that the system provides it. The robot then tells the system what tile it moved to or attacked and the tiles are updated. Once the robot is out of moves the system automatically moves onto the next player.

Preconditions:

- It's currently the computer player's turn
- The computer still has robots that are alive
- There are robots on the enemy's team that are still alive
- The computer has not moved a robot or attacked another robot yet

Main Flow:

1. The system tells the computer that it is their turn.
2. While the computer's robot still has moves remaining the computer:
3. Decides what tile they want to move to or attack based on what information is provided by the SCAN action
4. The computer turns in the direction that they want to move/shoot
5. [A1][A2]
6. The turn ends when the computer's robot is out of moves.

Alternate Event Flow:

[A1]The computer uses the MOVE action and moves forward one tile in the direction that they are facing

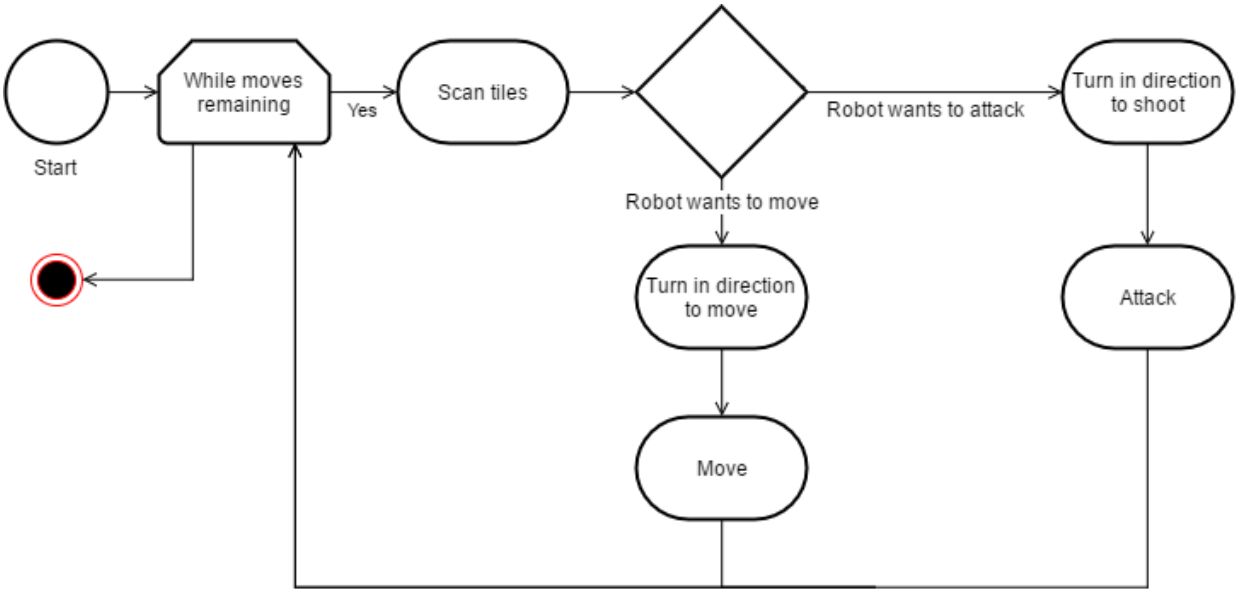
[A2]The computer uses the ATTACK action and attacks in the direction that they are facing hitting the tile that they selected

Error Conditions:

Postconditions:

- The computer's turn is over
- The computer's robot has been moved and/or has attacked another robot
- The Computer player's robot may be dead if they attacked themselves

Activity Diagram (see the Computer Player's Turn scenario for an explanation):



1.4 Navigating the Menu

When the program starts, a window comes up with some options for the user to choose from. They can start a game, view the instructions, or quit the program.

Preconditions:

- The program has just been started

Main Flow:

7. The start window is brought up and the user chooses an option
8. [A1][A2][A3]
9. If the start button was selected, open the window for selecting the number of players
10. Begin primary scenario 1.2

Alternate Event Flow:

[A3]The user chooses the "Start" button and the window to select the number of players

[A4]The user chooses the "instructions" button

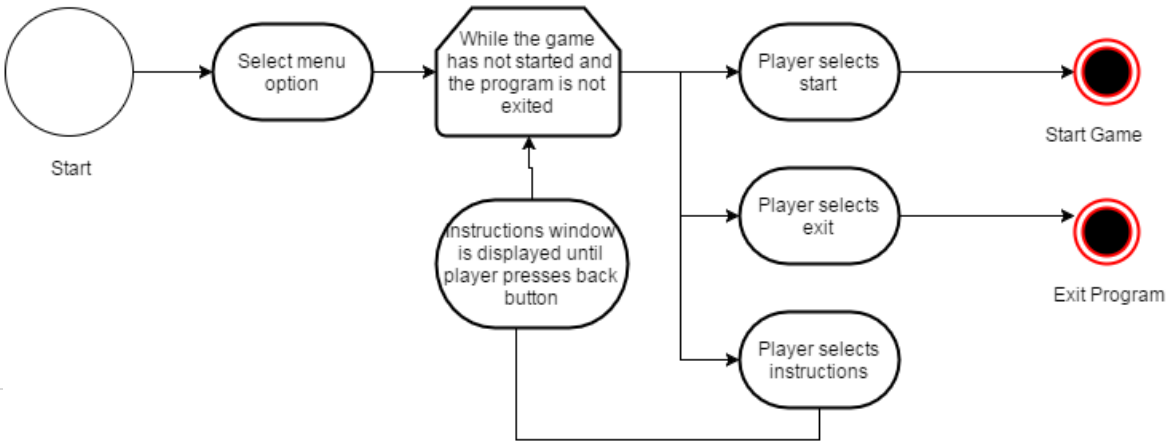
a. A text box comes up explaining the game with a single button to go back to the main menu(step 1)

[A3]The user chooses Exit and the program exits

Postconditions:

- The program either started a game, or exited.

Activity Diagram:



Complete Scenario View

This is a diagram showing the flow of the entire system from the execution of the program, to its exit. For more detail, go to the individual sections described above.