Requirement Document

Project Name:	RobotGo
Team:	CMPT370-B3
Team Member:	Cheng, Gong Hounjet, Carlin Lan, Shaoxiong Xie, Joey Yue, YiRui
Date:	Oct 2, 2016

Table of Contents

Part 1 Team identification & Game Description.

1.1 Team identification

Team B3 a team of five with three international students and two local students. We agreed on using democratic strategies on resolving conflicts. We meet at least three times a week, apart from the TA meeting.

1.2 Game Description

RobotGo is a two, three or five player's game. The goal of this project is to create a turn based robot battle game. With this program we can use a computer to hide player's moves from each other to increase strategy. Each player has a chance to name and control their own three robots, the options being scout, sniper and tank. Different robots have different attributes like attack power, movement range, field of view range and shooting range value. Each player begins at a different side of the board and take turns searching for each other and conducting thrilling battle on the battle grid!

A turn begins with Red team playing its robot with greatest movement followed by the next team's robot with highest movement value in clockwise rotation. When this round is completed, a second round is played where Red plays its next-highest movement robot, then orange, through all the teams up to blue. The third round where Red's lowest-movement robot plays, and so on until finally blue's lowest-movement robot plays. Basically, every robot gets one play per turn, and fastest robots move first. The game is completed when only one team's robots remain on the game board.

Part 2 System Diagram

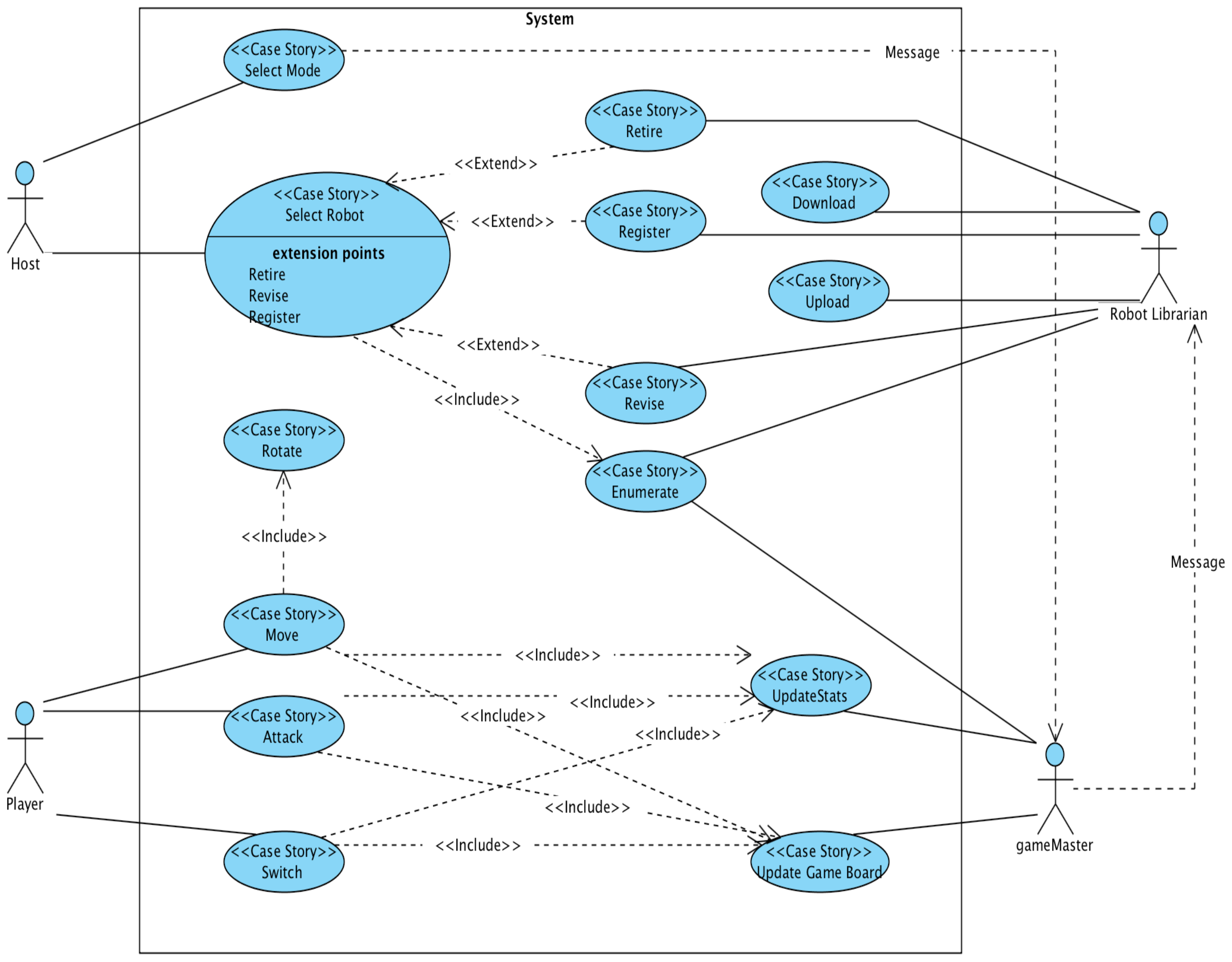


Figure 2.1 System Diagram

This is the system diagram outlining the basic interactions between the actors and actions.

Part 3 Uses Cases

This part mainly introduces the uses cases of the game. Based on the logic sequence of the game, we developed **4** actors and **14** actions. For each action, we provided a few detailed scenarios that might associate with this specific action, as well as an activity diagram for primary scenarios.

3.1 Select Mode Use Case

If the user chooses not to view and edit robots or is finished doing so and wishes to begin a game, they will navigate through the main menu to here. This where the user chooses the number of teams in a game and how many of which are human or AI controlled.

Preconditions:
1. Robots are loaded
2. User has made the choice to begin the game and navigated to this interface

Primary Actor: Host

Primary Scenario:
1. User is prompted to choose between 2, 3 and 6 teams
2. User chooses which teams are human or AI

Alternative Flow:
1. If user chooses 3 team game, the option to select between a 5 or 7 sided board becomes available
2. User chooses which teams are human or AI

Post Conditions
1. Players have their team
2. Game board size is set
3. Number of teams is set

Error Conditions
1. User doesn't set type of players for a team before trying to continue
2. Prompt user to choose and refuse to start game

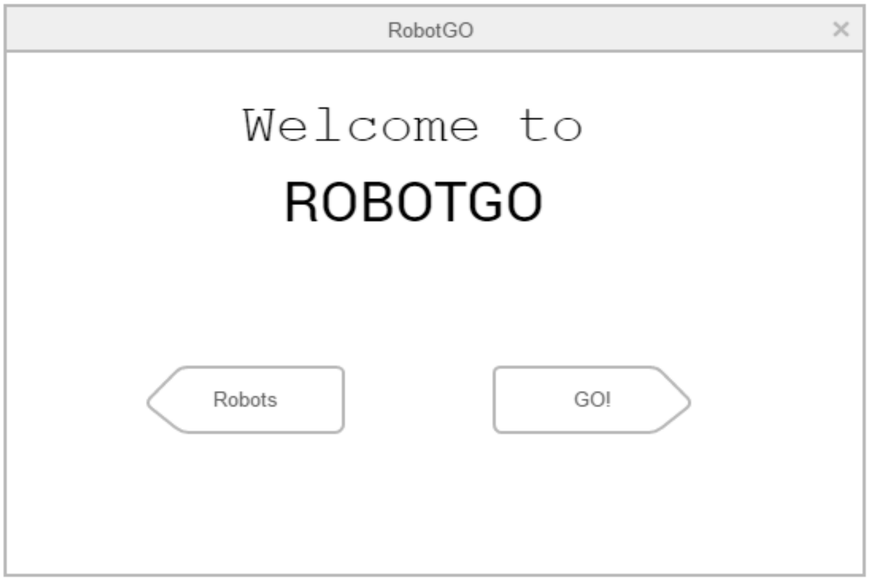


Figure 3.1.1: Welcome Screen

The user will be faced with a welcome interface and two choices: Change/View robot data (Robots) or Choose Mode and initialize game (GO!).

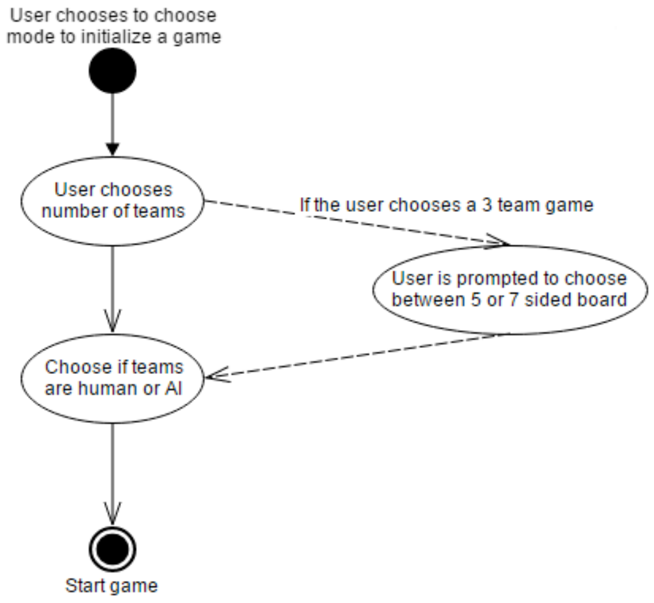

User chooses to choose mode to initialize a game
User chooses number of teams
If the user chooses a 3 team game
User is prompted to choose between 5 or 7 sided board
Choose if teams are human or AI
Start game

Figure 3.1.2: Select Mode Activity Diagram

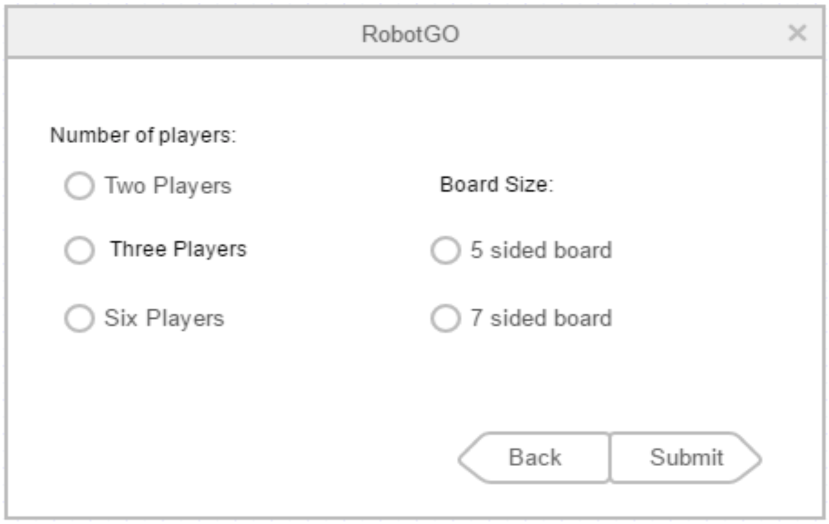


Figure 3.1.3 Interface for Host to Select Game Mode

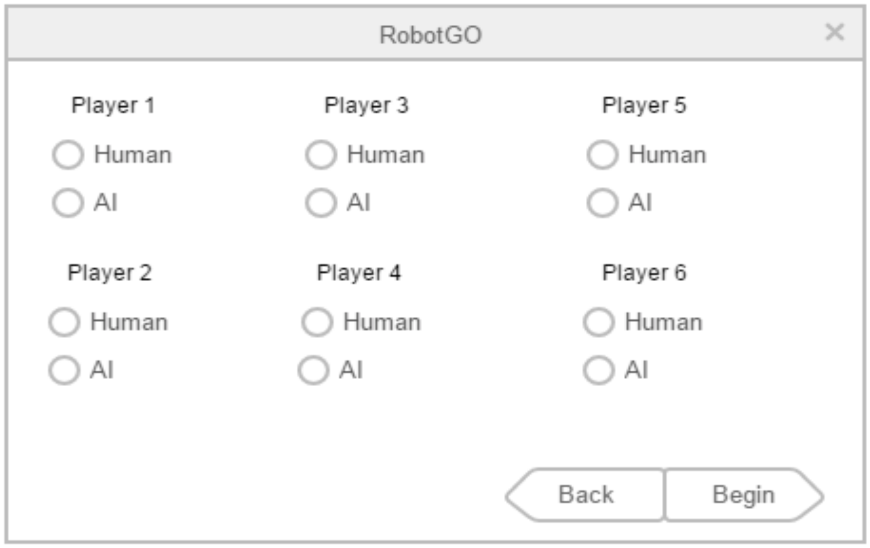
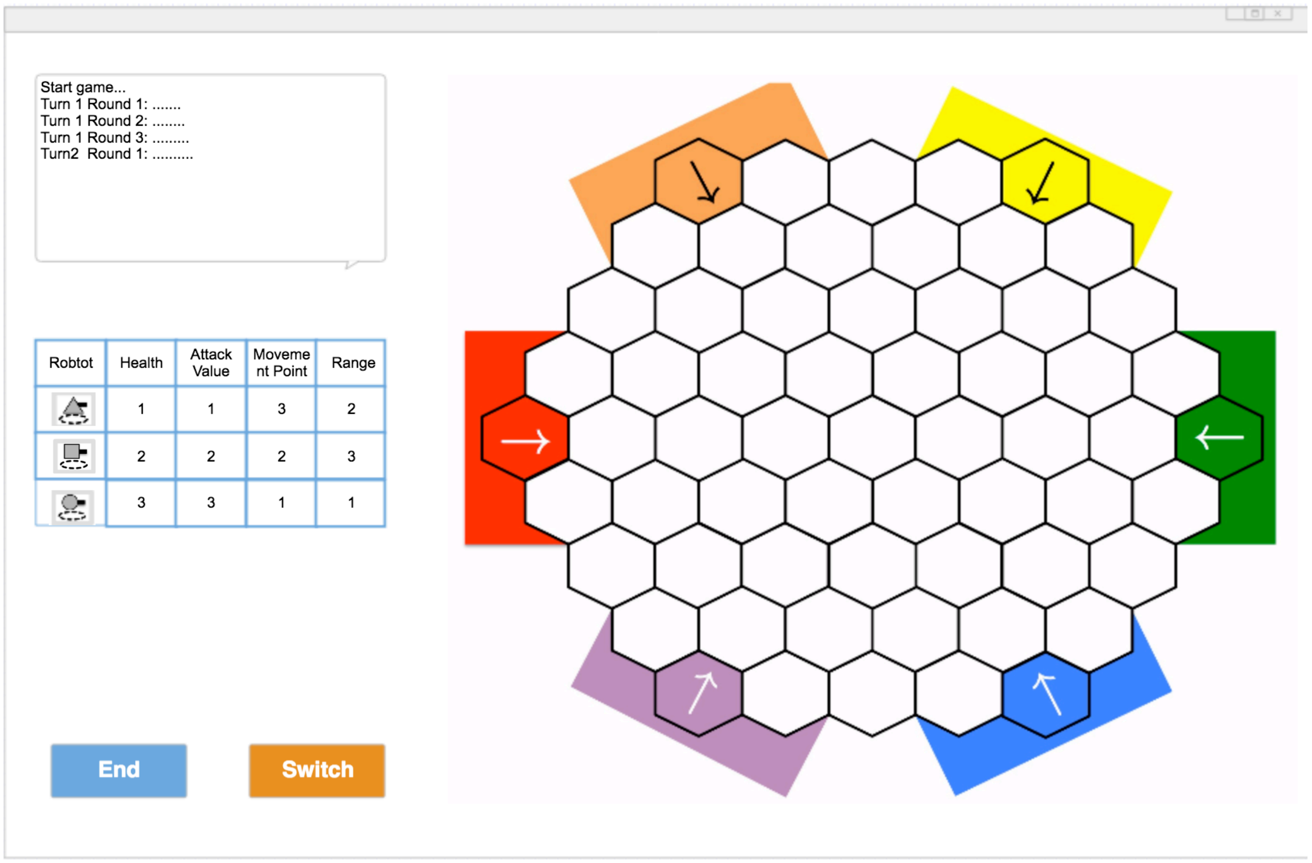


Figure 3.1.4: Interface for choosing AI or Human for teams

The main game board of the game is shown at the next page.

Main Game Board



Start game...
Turn 1 Round 1: ....
Turn 1 Round 2: ....
Turn 1 Round 3: ....
Turn2 Round 1: ....

Robtot	Health	Attack Value	Moveme nt Point	Range
	1	1	3	2
	2	2	2	3
	3	3	1	1

Figure 3.1.5: Interface for Game Board

3.2 Download Use Case:

On system start-up, download the Robot's records from the JSON file so the Robot Librarian has access to it for display and modification.

Primary Actor: Robot Librarian

Pre Conditions:

1. There is a JSON file populated with robots and their stats
2. The Robot Librarian has the address of the file

Primary Scenario

1. Robot librarian is told to download records
2. Robot Librarian downloads records
3. Robot Librarian saves records for as long as the game is run

Post Conditions:

Data is downloaded and stored for use

Error Conditions:

1. If data is corrupt, then display error message and continue
2. If file does not exist, display error message and continue

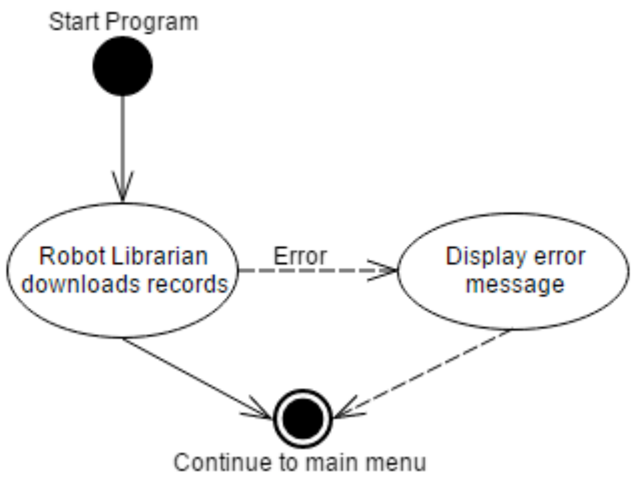


Figure 3.2.1: Download Activity Diagram

3.3 Select Robots Use Case

Primary Actor: Host

Description:
At the beginning of the game, the host chooses the desired robot for the entire game. The host can choose any robot that is displayed on the screen. The host can register a new robot if wanted, and the host can also revise or retire any specific robot.

Preconditions:
- The host have already selected the game mode.

Primary Scenarios:
- The host choose any robot enumerated on the screen.
- The host can revise or retire the robot that’s chosen.
- If the host is not satisfied by the enumerated old robots list, the host can also register a new robot.

Error Conditions:
- If the host choose more that three robots, the game board interface shows “Sorry, you can’t choose more that three robots.”
- If the host attempts to register a name which already exists, the game board interface shows “Sorry, the name already exists.

Postconditions:
- The host finishes choosing the robot and moves to next procedure.

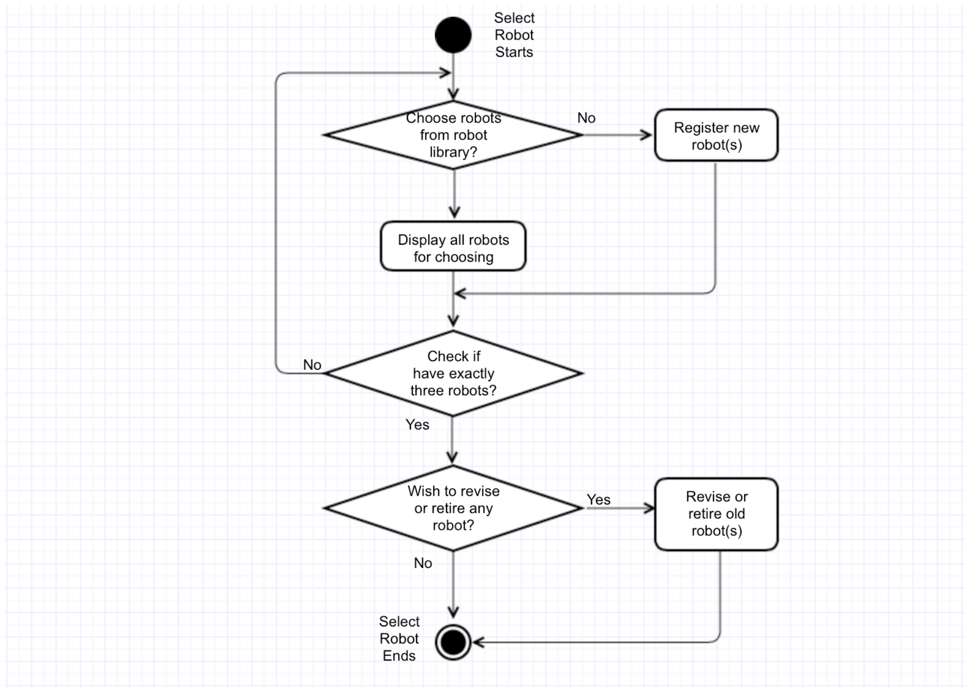

Select Robot Starts
Choose robots from robot library?
No
Register new robot(s)
Display all robots for choosing
No
Check if have exactly three robots?
Yes
Wish to revise or retire any robot?
Yes
Revise or retire old robot(s)
No
Select Robot Ends

Figure 3.3.1: Select Robot Activity Diagram

3.4 Enumerate Use Case:

The game stores all the robots and their stats. If the user chooses to change robots, Robot Librarian will display all information in an interface separate from the game board.

Primary Actor: Robot Librarian

Preconditions:

-The program is in the appropriate interface

-Robot stats have been downloaded

Primary Scenario

1. User chooses to see all the stored robots and their stats.
2. The appropriate interface is opened where all robots are displayed in a table
3. The user chooses to sort the robots in any column
5. The robots are sorted.

Post Condition:

-All the robots are displayed and sorted by what column the user specified

Secondary Scenario

1. If the user never chooses to sort the robots the robots the display does not change
2. If the user chooses to sort the robots again the use case returns to step three of the primary scenario

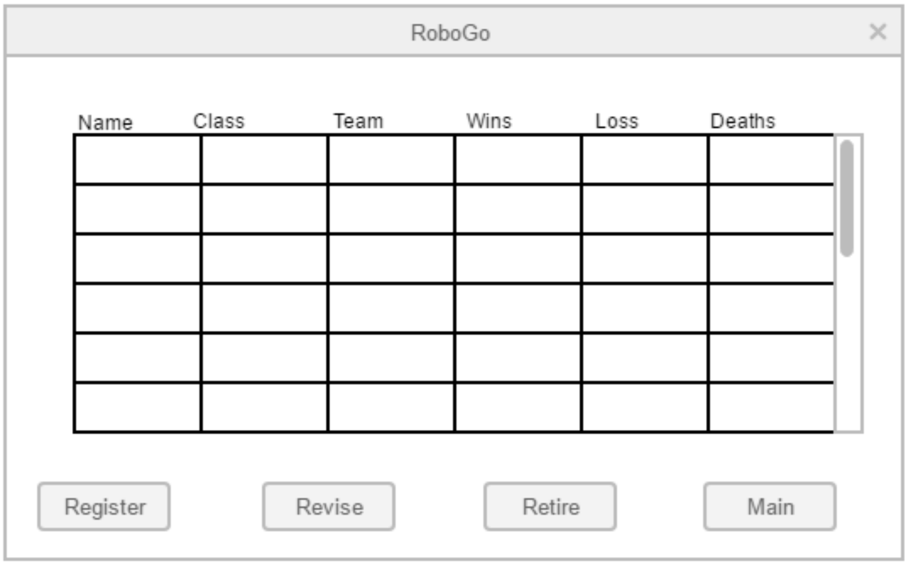


Figure 3.4.1: Choose Robots interface.

Enumerate will display robot information in a scrollable table. If they choose to Register, Revise or Retire any robot, the robot is selected and the desired function's button is pressed.

3.5 Register Use Case

The user can choose to create a new robot and add it to the store of robots.
A new robot needs a name, a class, and a team. A robot's name must be unique within a team

Primary Actor: Robot Librarian

Precondition:

The program is in the appropriate interface

Primary Scenario

1. User chooses to register a new robot.
2. User inputs class,name , team and code for the new robot.
3. The robot is added to the store of robots.
4. All statistics of the robot are initialized.

Post Condition:

A new robot is stored, with a name, team, and class specified by the user and other stats are initialized

Error Conditions:

1. The user inputs a name and team pair that already exists in the store
   -the user must choose a new name (return to step 2) or not register a robot.
2. A team has all 3 classes filled
   -User is prompted to cancel or retire a robot

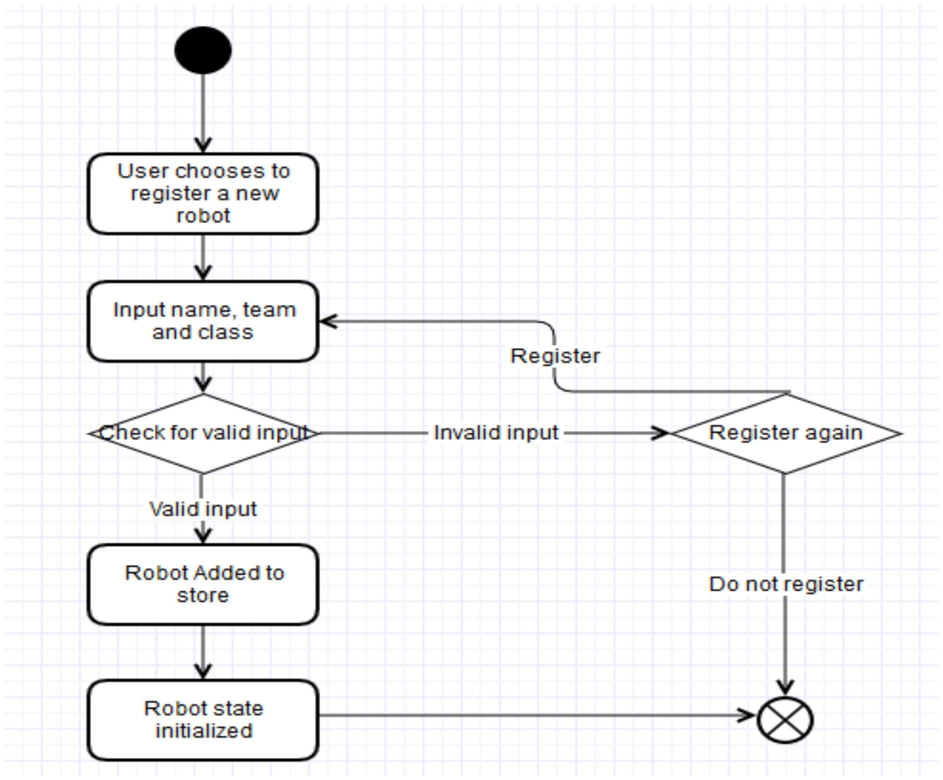


Figure 3.5.1: Activity Diagram for Register

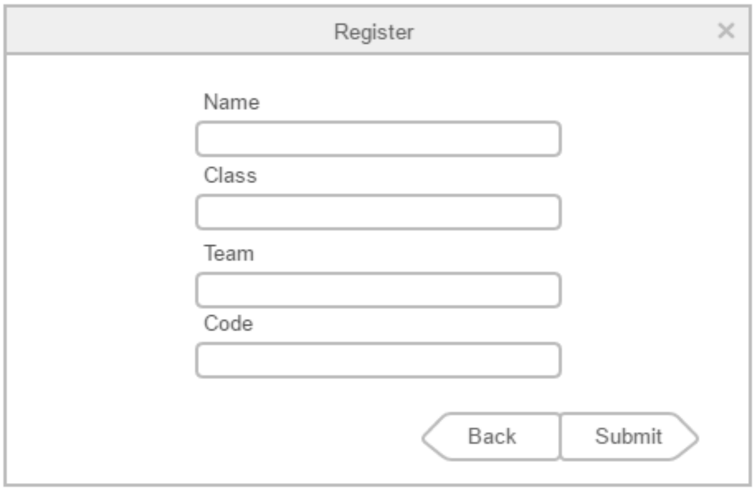


Figure 3.5.1: Interface for registering a robot

3.6 Revise Use Case

A user can change the code for a robot. Since the robot is different the old stats are no longer relevant and should be deleted.

Primary Actor: Robot Librarian

Precondition:

-The program is in the appropriate interface

Primary Scenario:

1. User chooses to revise a robot.
2. User chooses the robot to revise.
3. User inputs the new code for the robot to use.
4. The new code replaces the old code and the statistics for the robot is reset.

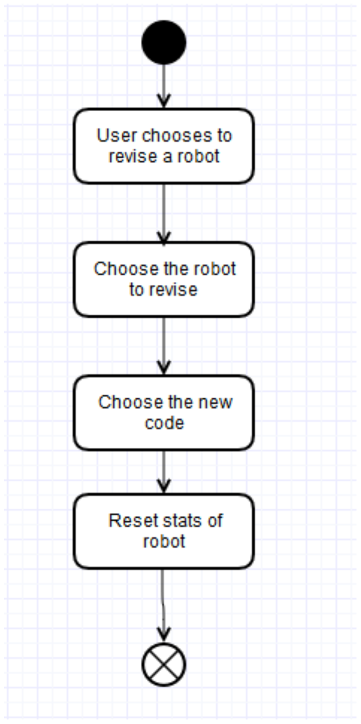
Post condition:

-The robot has the new code and all of its statistics have been reset

Error Conditions:

-User attempts to place robot on a filled team or attempts to use a name that is taken

-Display error message and prompt to retire old robot



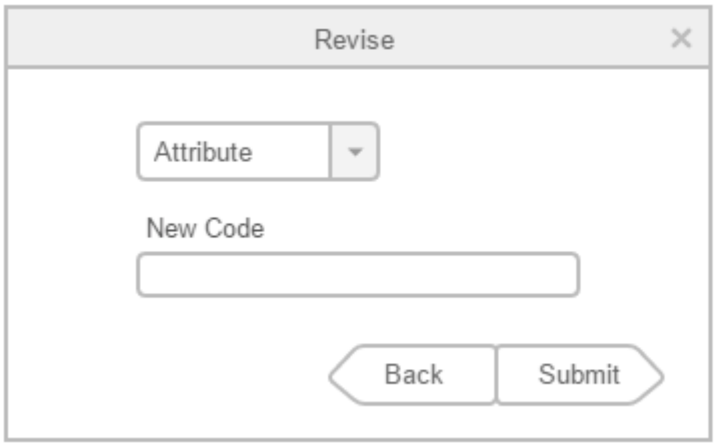


Figure3.6.1: Activity Diagram for Revise Figure3.6.2: Interface for Revise

3.7 Retire Use Case

A user can retire a robot, preserving its stats while allowing its name to be used by a new robot.

Primary Actor: Robot Librarian

Precondition

-The program is in the appropriate interface or has been prompted to retire a robot

Primary Scenario

1. User chooses to retire a robot.
2. User chooses the robot to retire.

Post Condition:

-The robot has been retired

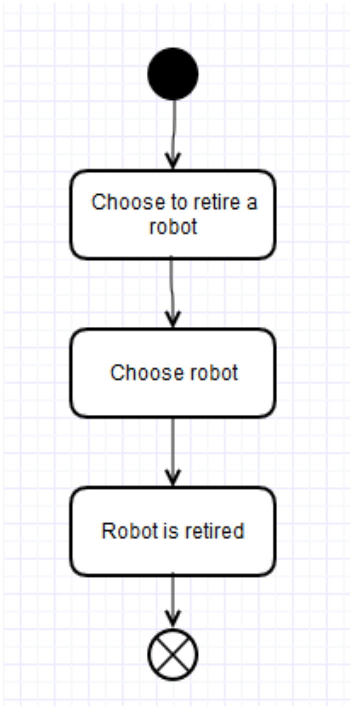


Figure3.7.1: Activity Diagram for retire

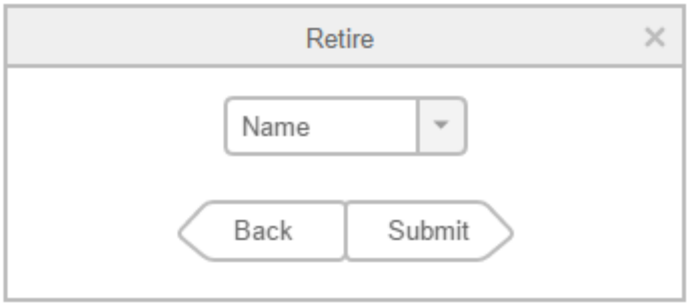


Figure:3.7.2 Retire interface

3.8 Move Use Case

Actor:
Players and GameMaster

Description:
If the player chooses to move, the player can move one connected tile a time until their move points run out. If the player chooses not to move, the player can also choose to attack or switch.

Preconditions:
1. The players have already selected their game modes and entered player information.
2. It’s the players’ round to make a move.
3. The player has at least 1 move point left.

Primary Scenarios:
1. The player chooses to move by clicking on option *move*.
2. The player moves 1 step a time until the move points run out.
3. The player chooses the destination, turns and moves.

Secondary Scenarios:
1. If the player clicks on option *move*, any time before running out of move points, including the situation of not moving at all, the player can choose to click on button *attack* then perform the action of Shoot.
2. If the player clicks on option *move*, any time before running out the move points, including the situation of not moving at all, the player can choose to click on button *switch* then perform the action of switching another robot to end it's round.

Error Conditions:
The player doesn’t have any move point left.
-The game board interface shows “You don’t have enough move points.”

Post conditions:
1. The player finishes its move action and updates the game board.
2. The players’ move points get updated.

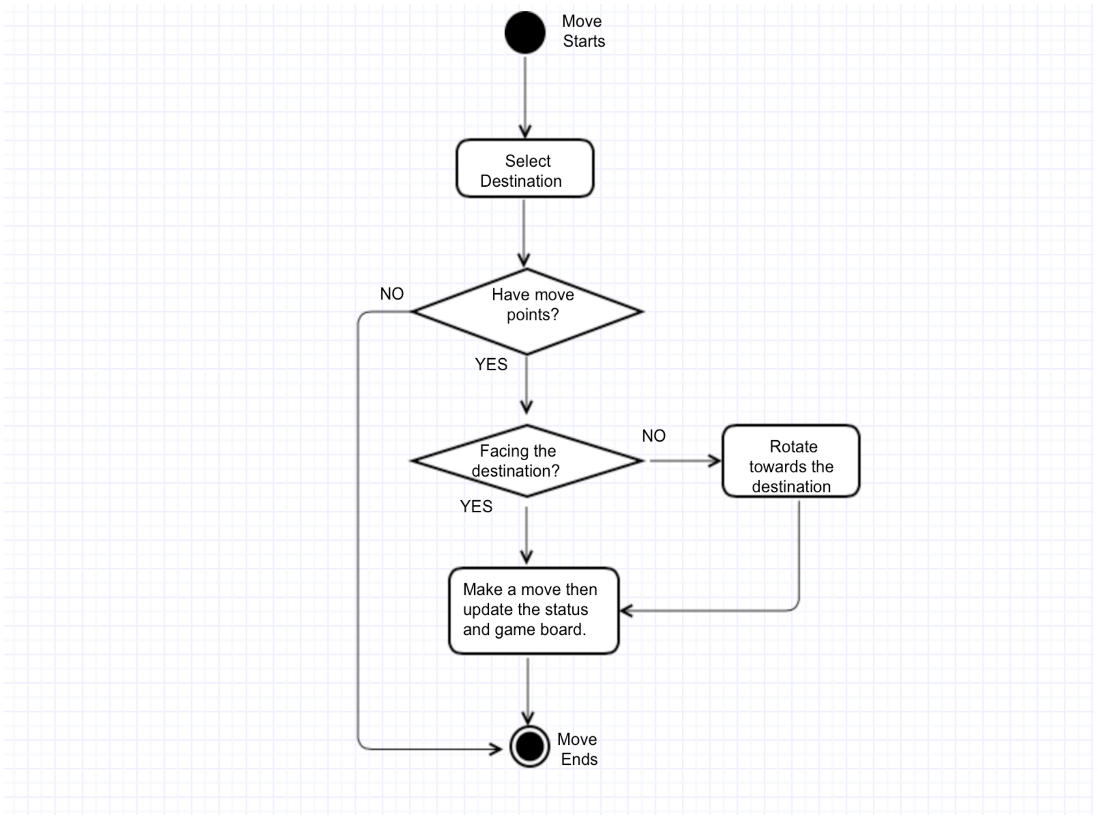


Figure 3.8.1 : Activity Diagram for Move

3.9 Attack Use Case:

The player may attack once per turn. The robot that attacks must must be in range for the attack to succeed. A robot may attack itself or its team members since an attack affects every robot in a target tile. The robot has an attack value that determines the strength of an attack.

Preconditions:

1. It is the player's turn.
2. The player has not already attacked this turn.
3. The game has determined which unit moves this turn.

Primary Actor: Player
Other Actors: Gamemaster

Primary Scenario:

1. Player selects target tile and attacks.
2. Game checks that the target tile is in range.
3. Every robot in target tile takes the appropriate amount of damage.
4. Update stats for shots fired, damage inflicted, damage absorbed etc for all robots involved
5. Attack is finished and Game Controller is notified of changes

Post Conditions:

1. The player's attack for the turn is used up
2. Damaged pieces hit points sent for update

Secondary Scenarios:

1. Target not in range or robot: Player must choose another target or not attack
2. If the player wants to choose another target the use case goes back to the first step of the primary scenario.
3. If the player does not want to attack, the user can click “switch” and the move is skipped and the board does not change.

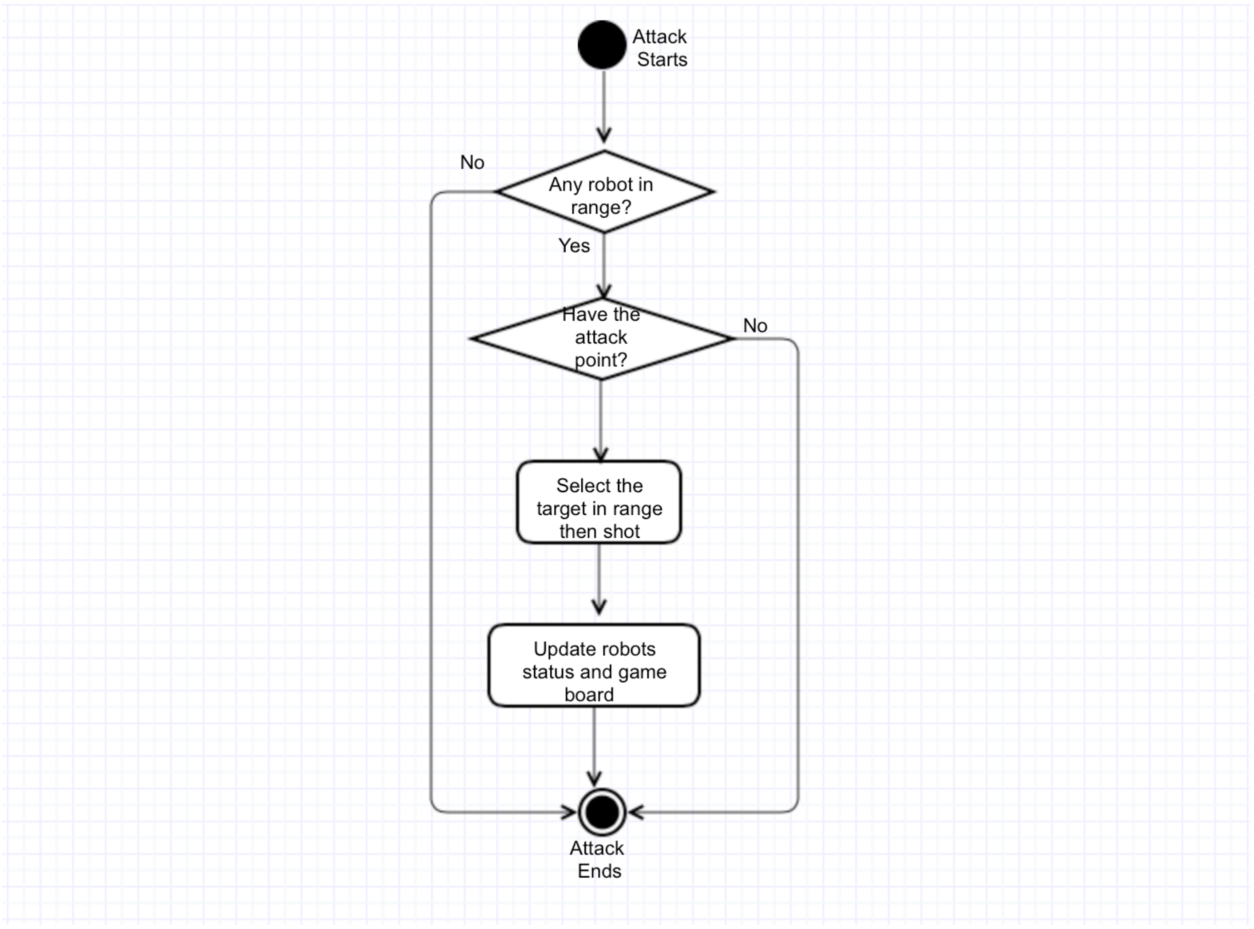


Figure 3.9.1: Attack Activity Diagram

3.10 Switch Use Case

Primary Actor: Player

The player click on "Switch" button, then ending the behavior of current robot and switching to the next team's survived robot which has the highest movement value and does not behave in this turn;

Precondition:
Player clicks on "Switch" Button

Primary Scenario:
1. Click on the "Switch"
2. Ending the behavior of current robot
3. Starting the behavior of next robot which has the highest movement value and does not behave in this turn

Post conditions:
Ending the behavior of current robot and switching to the next team's survived robot which has the highest movement value and does not behave in this turn

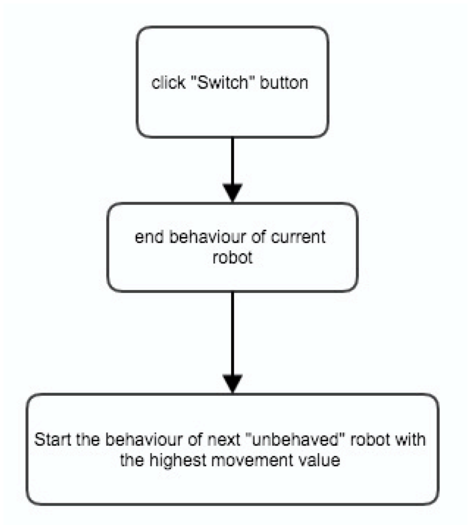

click "Switch" button
end behaviour of current robot
Start the behaviour of next "unbehaved" robot with the highest movement value

Figure 3.10.1: Switch Activity Diagram

3.11 Rotate Use Case

This happens every time before attack and move and it doesn’t cost a movement point.

Actor: Player

Preconditions:

1. Game starts and there is at least more than one side have robots alive.
2. Robot moves or attacks

Primary Scenario:

1. Figure out which direction to face
2. Rotate gun turret to desired direction

Post Condition:

1. The robot faces the right direction.

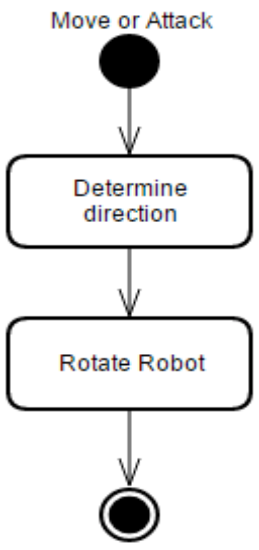


Figure 3.11.1: Activity Diagram for Rotate

3.12 UpdateStats Use Case

Primary Actor: GameMaster

Description:
The gameMaster would track and update info of the robot after it's behavior in the round, based on their behavior, including individual result, the damage, the status, the distance traveled and shots fired. Meanwhile, the gameMaster would always check whether the match has ended. If so, the team result would be updated at the end of match.

Preconditions:

1. After the behavior of each robot
2. After clicking on the “switch” button

Primary Scenario:

1. Initialization: initial statistics for all robots on the game boards.
2. during the behavior of robot A which is alive:
   - if A do nothing:
     - the gameMaster does nothing;
     - the A's status would be updated to "behaved" after click "switch".

   - if A is just moving:
     - update the distance travelled of A, adding by A's moving spaces;
     - if distance travelled of A in current round has been equal to Maximum Movement value of robot A, then robot A cannot move in current round;
     - the A's status would be updated to "behaved" after click "switch".

   - if A is moving and shooting at robot B:

     For robot A, we should
     - update damage inflicted, adding by A's attacking value;
     - update distance travelled of A, adding by A's moving spaces;
     - if distance travelled of A in current round has been equal to Maximum Movement value of robot A, then robot A cannot move in current round;
     - update shot fired of A, adding by 1;
     - the A's status would be updated to "behaved" after click "switch".

     For robot B, we should:
     - update the damage absorbed, adding by A's attacking value;
     - if the current damage absorbed is not less than health value of B, then change individual result of B to "destroyed";

3 For the new turn:
when gameMaster has perceived "new turn", the damage inflicted, distance travelled, Status and shots fired of all survived robots would be initialized again.

4 For the Match end:
When gameMaster has perceived that there existed only one team on the boards, the match has ended. Then gameMaster updated the team result for all robots in the match, "win" for all robots of last survived team and "lost" for others robots.

Extension Scenarios:

1. If A is shooting at one cell which has more than one robots:

for robot A:

- update damage inflicted, adding by (A's attacking value * the number of robots in target cell);
- if A moved, then update distance travelled of A, adding by A's moving spaces;
- update shot fired of A, adding by 1;
- the A's status would be updated to "behaved" after click "switch".

for other robots in the same cell:

- update the damage absorbed, adding by A's attacking value;
- if the current damage absorbed is not less than its health value, then change its individual result to "destroyed";

2. if robot A and other robots are in same cell, then A shoots at them:

for robot A:

- update damage inflicted, adding by (A's attacking value * the number of robots in this cell, including itself);
- if A moved, then update distance travelled of A, adding by A's moving spaces;
- update shot fired of A, adding by 1;
- update the damage absorbed, adding by A's attacking value;
- if the current damage absorbed is not less than its health value, then change its individual result to "destroyed";
- the A's status would be updated to "behaved" after click "switch".

for other robots in the same cell:

- update the damage absorbed, adding by A's attacking value;
- if the current damage absorbed is not less than its health value, then change its individual result to "destroyed";

3. if there is no robot left at the end of match, which means the robots of different team are destroyed at the same time:
    - Then gameMaster would update the team result "lost" for all robots in the match.

Post conditions:
The info for each robot has been updated after behavior of each robot in the round, including: individual result (survived or destroyed), team result (win or lost or unknown), the damage (inflicted and absorbed), status, distance travelled and shots fired;

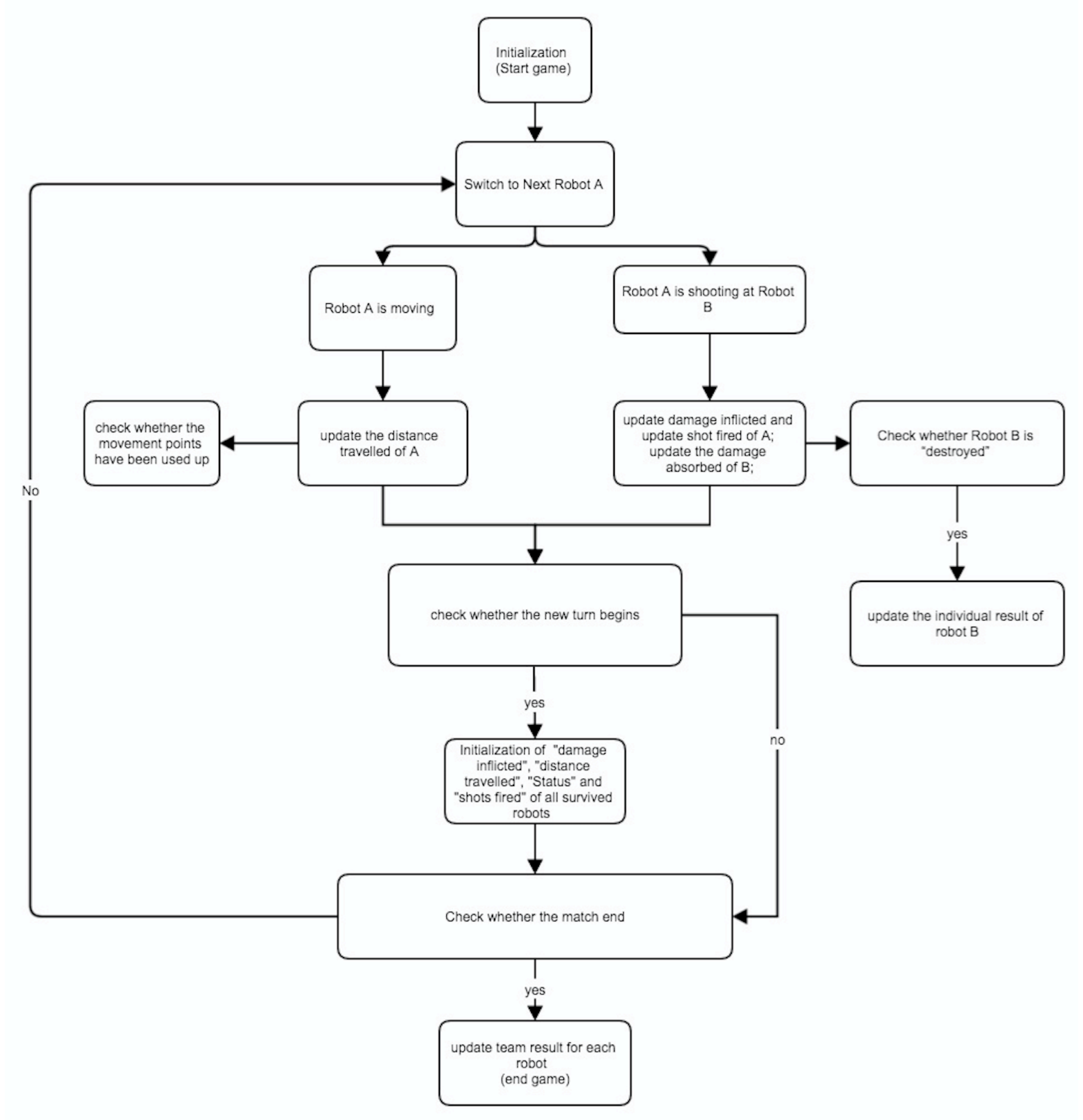

Initialization (Start game)
Switch to Next Robot A
Robot A is moving
Robot A is shooting at Robot B
check whether the movement points have been used up
update the distance travelled of A
update damage inflicted and update shot fired of A; update the damage absorbed of B;
Check whether Robot B is "destroyed"
yes
update the individual result of robot B
No
check whether the new turn begins
yes
no
Initialization of "damage inflicted", "distance travelled", "Status" and "shots fired" of all survived robots
Check whether the match end
yes
update team result for each robot (end game)

Figure 3.12.1: Activity Diagram for UpdateStats

3.13 UpdateGameBoard Use Case

Keep track of every attack, movement, turn, mouse click

Primary Actor: Game Master

Precondition:
Whenever there is attack action, movement action, turn action, or mouse click action happens, Game Master is told and it updates the game board.

Primary scenario:
1 Before the game starts

1. Display the the beginning interface, including the game mode selection, player selection, player color selection, all player's statics, start and finish button.
2. If the start button is clicked, it goes to populate the screen and load the game according to the number of players. If the finish button is clicked, terminate the game.

- React to movement action

(1) display the game board and player and the statics of the robot who is playing.
(2) when a movement action succeeds, the robot will spin so that the front of the robot towards the destination and move one space in the direction. And then update the statistics of the player and the the visible fields.
(3) When the movement action failed, it shows a dialogue prompting the player that the player the action failed and give the specific reason.

- React to attack action

(1) when an attack action succeeds, there will a bullet-similar object appear and starts from the the space of the attacker, move in the direction of the destination and ends in the destination space. When the attack failed, it will prompt that the attack is out of range.
(2) If the attack action causes any damage, it shows a prompts telling the player the action succeeds and update the statics of the attacker and the robot under attack.

- React to mouse click

(1) when a cell in the game board is clicked, if the cell is in the range of current robot, it will prompt two buttons. One shows move, another shows attack. Otherwise it doesn't respond.
(2) When the attack button is clicked, it will invoke attack action. When the move action is clicked, it will invoke the movement action. When out side of the the two buttons area is clicked, the two button will disappear and the player can choose again.

- React to switch button

Whenever the switch button is clicked, the game board will load next robot's statistics.

If the esc key is pressed during the game, there will be two buttons appear. One is finish button and another is continue button. When the finish button is clicked, the game terminates, when the continue button is clicked, it goes back the game mode.

2 After the game

Update the all robot's statistics. And display the winner and loser.
Postcondition:
The screen responds accordingly.

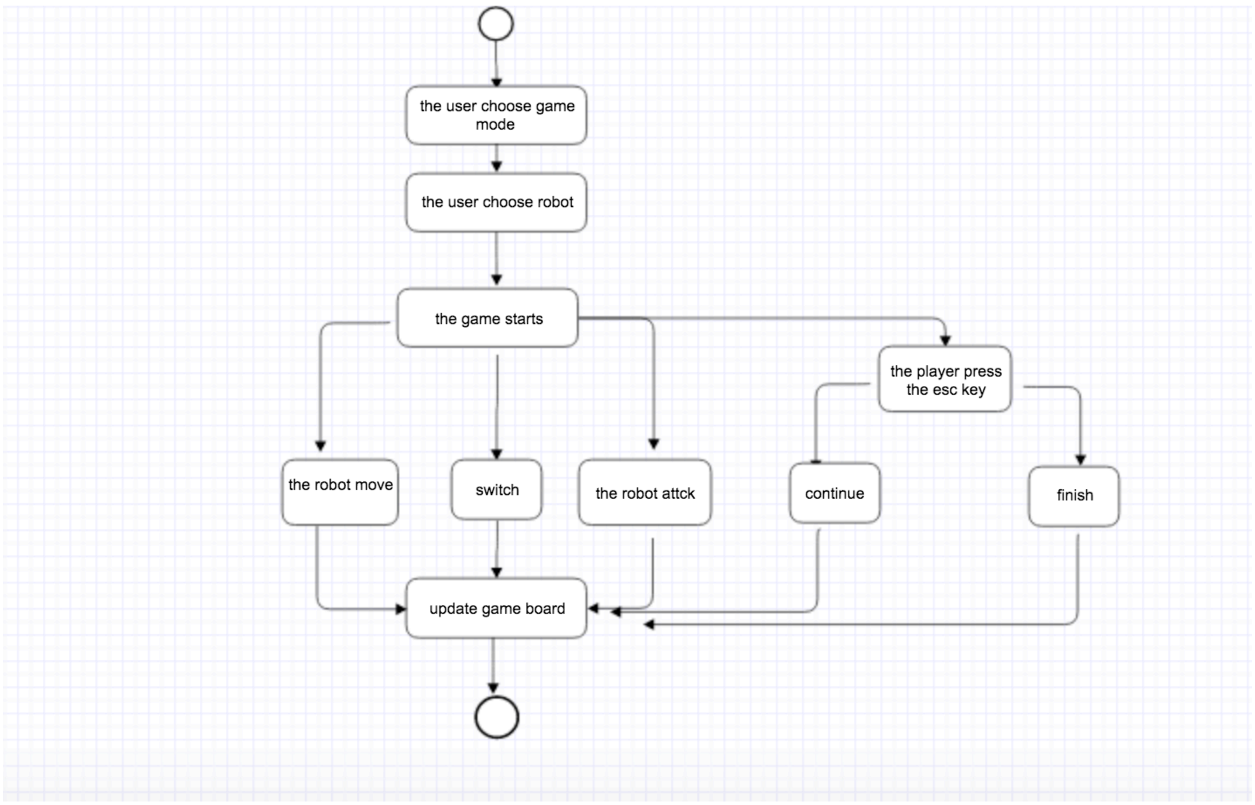


Figure3.13.1: Activity Diagram for UpdateGameBoard

3.14 Upload Use Case

The process of uploading updated stats to the JSON file following a completed game

Preconditions:

1. Game is completed and all relevant data is updated
2. Robot Librarian has address of the file to upload to

Primary Scenarios:

1. Robot Librarian is alerted that the game has ended
2. Robot Librarian reads data from stored values
3. Robot Librarian saved updated data to JSON file

Post Conditions:

JSON file and Robot Librarian have matching data

Error Conditions:

File does not exist

-make a new file

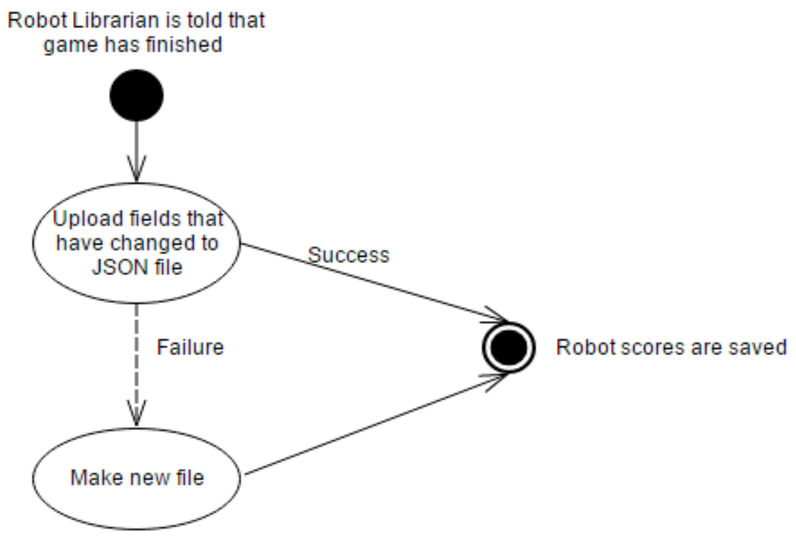

Robot Librarian is told that game has finished
Upload fields that have changed to JSON file
Success
Failure
Robot scores are saved
Make new file

Figure 3.14.1: Activity diagram for Upload

Executive Summary of Requirement Document

This requirement document contains **4** major parts which includes:

- Team identification & game summary.
- Use cases with identification of actors, actions and detailed scenarios.
- Activity diagram for primary scenarios. System diagram with clear relationship between each action.
- Basic game board interfaces.

To start, an introduction to our team was provided, followed by the general game description. Then, based on the logic sequence of the game, we developed **4** actors and **14** actions. For each action, we provided a few detailed scenarios that might associate with this specific action, as well as an activity diagram for primary scenarios. In addition, with all the actors and actions we had, we figured out the relationship between actions and constructed the system diagram. Finally, we designed **4** basic game board interfaces that matched with our initial setting of the game.

1. Team identification & game description.

- This is a team of five with 3 international students and 2 local students. We agreed on using democratic strategies on resolving conflicts. We meet at least three times a week.
- The game involves using robots with different attributes in different team to eliminate one another. The beginning of this document has provided a detailed game description.

2. Use cases.
As it's ultimately a computer game, the logic order of describing each use case is based on assuming how the player will play this game.

2.1 The game starts with the actor **HOST** with the action **select mode**, which associated with the situations of**:**

- Choosing from Human VS Human, Human VS AI or AI VS AI.
- Message with system to generate game board interface with appropriate side length.

2.2 Second step is **HOST select robot.** This action is closely connected to the action **Download, Enumerate, register, revise and retire,** and has situations including:

- Select robot(s) from robot library. This automatically activate **Download** and **Enumerate.** All the old robots will be displayed for choosing.
- Register new robot(s).
- Decides on whether to revise or retire robot(s) or not.

2.3 Parallel to the second step is the step that actor **Robot Liberian** performs the action **Download.**

- Downloading all the robots' information from robot library. The information includes the win&lose times and win lose wining rate of a specific robot.

2.3 Third step is the actor **player** (human or AI) manipulating the robots, which will associate actions including **Move, Rotate, Attack and Switch**.

- To move. Robot move one step a time if have move points. Robot will rotate first then move to the destination.
- To attack. Robot can attack only if it has enemies in range and have attack points. Robot(s) that being attacked lost health points.
- To Switch. This will end the current round for a specific robot.

2.4 Parallel to the third step is the step that actor **Gamemaster** performs the action **UpdateStats** and **update GameBoard.**

- Update the statistics of each robot in every round, to keep track of the move points, attack points and health point.
- Update the gameboard each time a robot moves or be destroyed.

2.5 The final step is the actor **Robot Librarian** performs the action **upload.**

- When the game ends, gamemater interact with robot librarian with the win&lose information of each robot.
- Robot librarian upload the win/lose information which is to be kept under the specific robot.

3. Activity diagram and system diagram.

- The document has **14** Activity diagrams, which is generated from the primary scenarios of each action.
- The system diagram has listed **4** actors and **14** actions, with arrows between actions to clearly demonstrate the relationship between actions.

4. Four basic interfaces.

- First, when entering the game, there is an welcome interface, which also can be used by host to select mode.
- Second, there is another interface displays all robots' information for host to select mode.
- Third, we will have the main interface for the game, with game board in the center and a small table at the down-right corner of the screen to display robot statistics.
- Lastly, there will be an interface designed specifically for the end of the game with the congratulations image for the winning team.

In conclusion, based on our understanding of the game, we have developed 4 actors and 14 actions. With the logical order of playing the game, we made clear the relationship between each action and came with the system diagram. Moreover, four different interfaces correspond to different steps of game were designed. The interfaces we have came up with now are subjected to be modified in the future.