War of Robotcraft

Requirement Document

Team: A3

Team Members:

Name	NSID	Student ID
Fu, Chen	chf354	11183491
He, Jiahuan	jih889	11183346
Wang, Shisong	shw940	11157916
Xie, Ruida	rux793	11194258
Yang, Chen	chy202	11183550

Date: Oct 2, 2016

1. Team Identity

We are team A3 with five team members. We take the democracy as the leadership style, and setup ground rules to keep an open teamwork atmosphere; therefore, all team members are well motivated to dedicate their selves into this project. The following is a concrete description for each member:

Fu, Chen

She is a third year undergraduate computer science student. She likes designing, so she hopes she contributes her designing talent to this project.

He, Jiahuan

In terms of credits he received, he is a second year student. But he has been speeding up his learning progress so that he can take third year class this term since he loves computer science and challenges.

Wang, Shisong

He is a third year undergraduate computer science student. He likes C# programming, and is willing to learn more about web programming in the future.

Xie, Ruida

He is a third year undergraduate computer science student. He is always curious about the unknown.

Yang, Chen

He is a third year undergraduate computer science student. He likes programming in Java with Swing to develop applications. Also, He likes learning new technologies.

2. Game Description

2.1. Summary

In this project, we wish to deliver a robot board game which can have three kinds of play modes which are human versus human, human versus AI and AI versus AI. And there is a remote storage mechanism which stores data and supports the local machines to download and

retrieve data. And in human versus human mode, players are able to connect to each other through a computer network. The game details are listed below.

2.2. Game name

War of Robotcraft

2.3. Number of Players

two, three or six

2.4. Map

The map of the game is a big hexagon consist of small hexagons. The size of the big hexagon depends on the number of players. If there are two players, the side length of the big hexagon is 5 small hexagons. If three players, the side length is 5 or 7 hexagons. If six players, the side length is 7 hexagons. Each corner of the hexagon holds a color which represents each player.

2.5. Roles (Robots)

each robot has 4 properties: Attack, health, movement, range.

- Scout: Highest movement ability, but weak at attack.
- Sniper: Has best vision.
- Tank: Has best attack ability and health points. But Tank is very poor at movement and vision,

2.6. Rules

- Start: each player has three robots with all kinds of roles.
- Move: movements start from the red player with its robot with highest movement ability.
- Round and turns: One round consists of many turns. Each turn means each player with at least one robot has played one robot unit. One round ends when all players have played all remaining robots once.
- Movement priority: each round always begins with the movement of the robot with the highest movement ability, and then, the next player moves its fastest movement robot and so on. Also, a robot can choose not to move.
- Movement points: one move cost one movement point, and one shot does not cost movement points. Changing directions does not cost extra movement points.

- Attack: A robot can only attack the enemies within the visual range. The robot gets attacked will be deducted the same health points as the attacker's damage value. Once a robot loses it's all health points, it dies and loses all abilities and vision. A robot can also attack himself.
- End: the game ends when any team of players loses all their robots. And the other team win the game.
- Quit: give up is allowed, and the player who gives up will lose the game.
- Positions: There are no limitations about how many robots can stand at the same spot. If a position gets attacked, all robots are damaged.
- Directions: There are 6 directions that a robot can face. Turning directions does not cost movement points.

2.7. Programming Tasks

I. The system must be developed, written and worked with the version control system.

II. The system is created using Java programming language, and it can be used on multiple platforms, including Microsoft Windows, Mac OS, and Linux.

III. The system should offer an entertaining and engaging graphic user interface. Menus will be interactive and easily accessible throughout the game. Game players can make choices of keyboard or mouse as input devices, and the game will response the input.

IV. The system should satisfy any combination of human and AI player, which is human vs human, human vs AI and AI vs AI.

V. The system should never crush or hang because of an error.

VI. The system will use MVC architecture to develop.

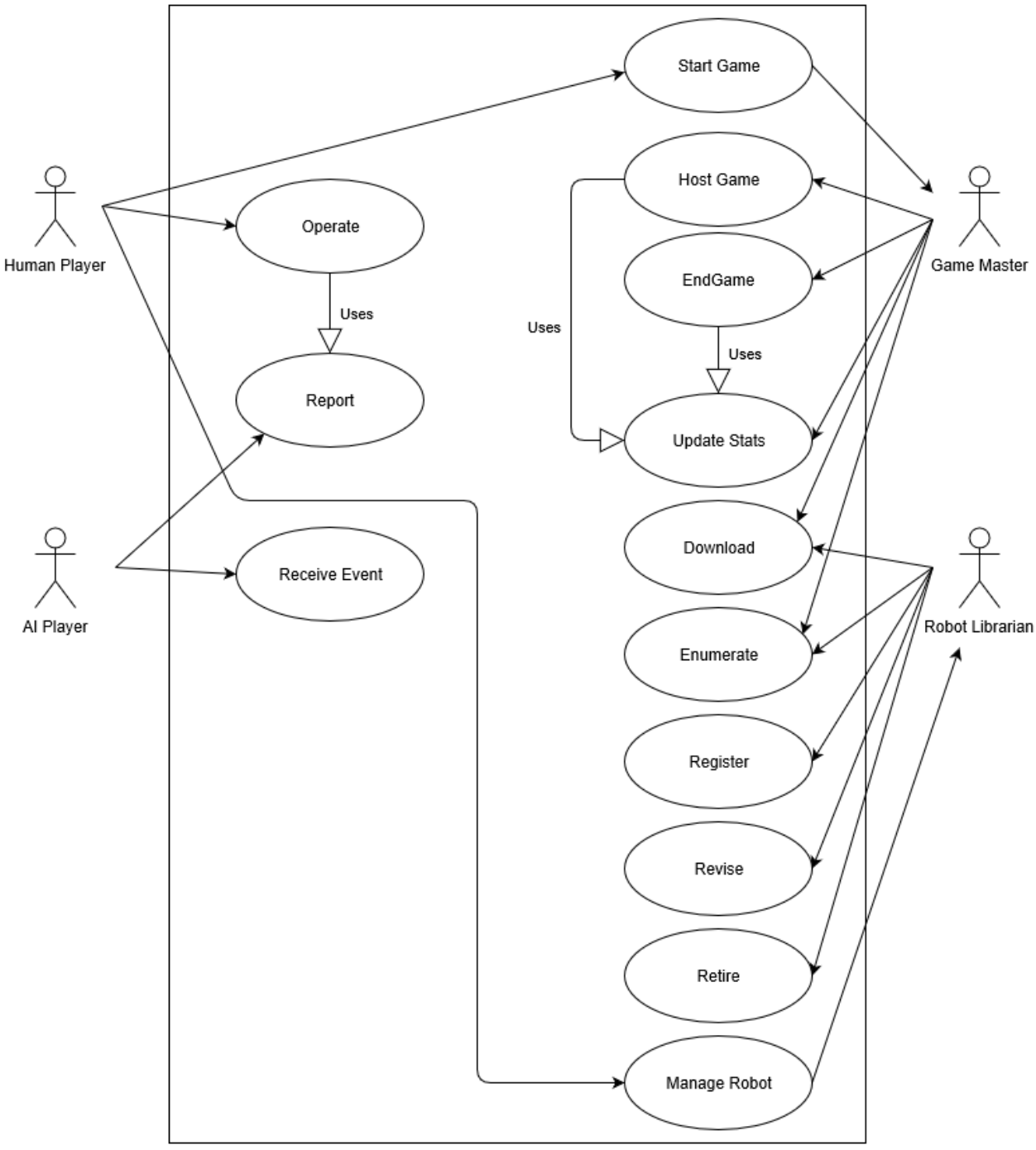
VII. The game data files should be platform independent.

VIII. The system code will follow the standard coding convention for the Java programming language, and will be easy to read and upgrade.

IX. The system should support minimum 2 players to play game at same time.

X. The system should give the user a brief guide for how to play when first invoked.

3. System Diagram



Human Player: Players controlled by humans.

Interface for Human Player
Shoot an enemy
Move a robot
Turn a robot to another direction
End current play
Surrender
Start a new game
Manage Robot
Exit Game

AI Player: Players controlled by artificial intelligence.

Interface for AI Player
Receive Event
Report current actions
Determine actions

Game Master: An administrator of the game.

Interface for Game Master
Display corresponding information
Download robots data for simulator
Update robots' stats
Enumerate robots' information
End game and show winnner
Save & load game

Robot Librarian: An administrator of robot data.

Interface for Robot Librarian
Register a new robot
Revise existing robot
Retire existing robot
Enumerate robots' information
Download robots data for simulator

4. Use Cases

4.1. Start game primary use case

- **Summary:** The game application is opened. When the human player selects the game mode (human vs human, human vs AI or AI vs AI) and the number of players, the game master initializes the game board and the first play starts up.
- **Actors:** Human Player and Game Master
- **Precondition:** The game application is opened.
- **Flow of events:**
    1. This use case begins when the game application is opened;
    2. The human player setup the game mode and number of players;
    3. The game master downloads robots' records;
    4. The game master creates a new game;
    5. The game master initializes the game board and starts the first play;
    6. This use case ends.
- **Post-condition:** New game created, and play one begins.
- **Error-condition:** If robots are not enough, the game master can not create a new game.
- **Data formats:** JSON
- **Necessity:** Must -have
- **Activity diagram:** Diagram 1

Start game secondary use case:

- **Precondition**: Robots are not enough.
- **Flow of events**:
    1. Go to the manage robots use case
- **Post-conditions**: Manage robots use case invoked.
- **Error-condition**: None
- **Necessity**: Must-have
- **Activity diagram:** Diagram 1

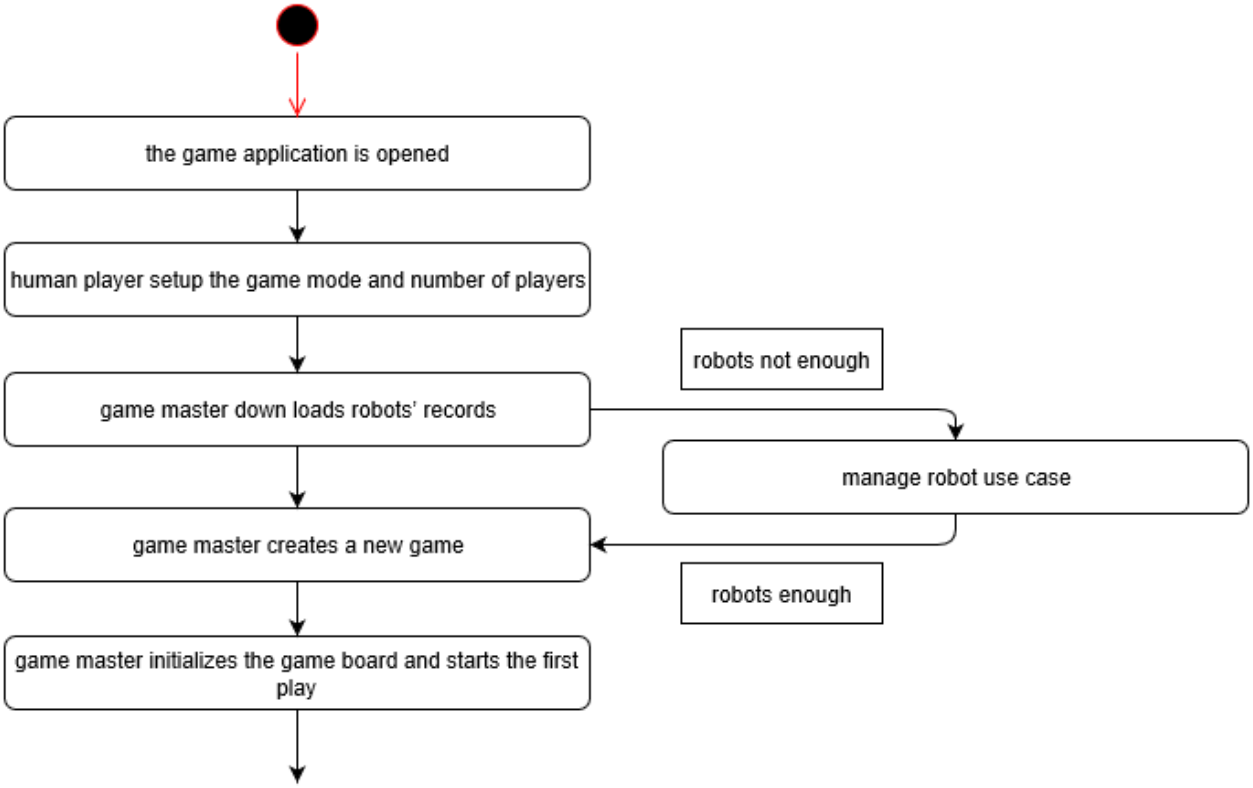


Diagram 1

4.2. Operate primary use case

- **Summary:** When a human player's play starts, the human player has the right to operate a robot on the game board. Then the human player operates the robot.
- **Actors:** Human Player
- **Preconditions:** A human player obtains the right to access current play.
- **Flow of events:**
    1. The use case begins when a human player's play starts.
    2. The human player makes decisions and operates the robot.
    3. Send a request for the report.
    4. The use case ends.
- **Post-conditions:** The human player's robots perform actions.
- **Error-conditions:** None
- **Necessity:** Must-have
- **Activity diagram:** Diagram 2

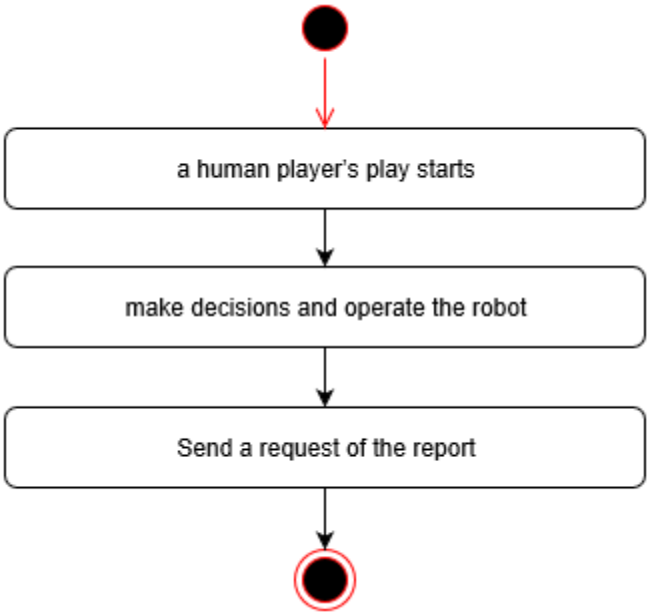


Diagram 2

4.3. Receive event primary use case

- **Summary:** When an AI player's play starts, the AI player receives the action permit from the game master.
- **Actors:** AI Player
- **Preconditions:** The game turns into the play of an AI player.
- **Flow of events:**
    1. The use case begins when the AI player receives the action permit;
    2. The AI player receives the message of the robots;
    3. The AI player makes decisions and operates the robot;
    4. The use case ends.
- **Post-conditions:** AI robots performed actions.
- **Error-conditions:** None
- **Necessity:** Must -have
- **Activity diagram:** Diagram 3

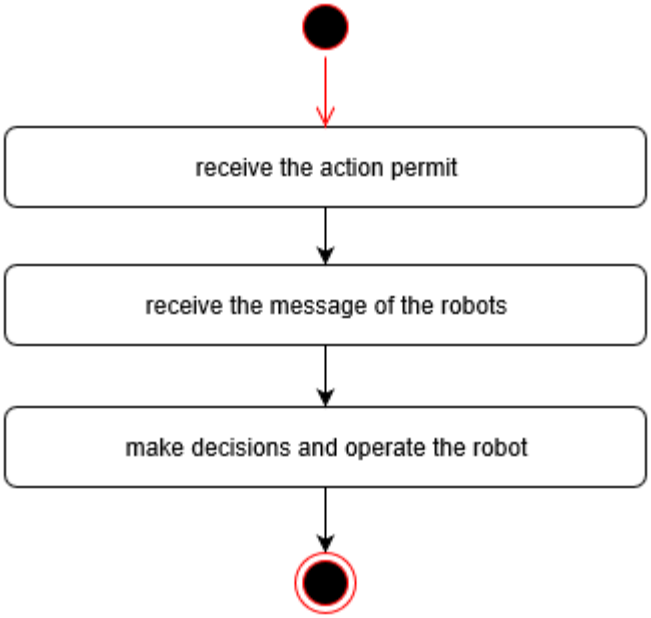


Diagram 3

4.4. Report primary use case

- **Summary:** When the human player and the AI player finish operating their robots, the operations are sent to the game master at the end of each play.
- **Actors:** Human Player and AI Player
- **Preconditions:** At the end of each play.
- **Flow of events:**
    1. This use case begins when at the end of each play.
    2. Send the report to the game master.
    3. This use case ends.
- **Post-conditions:** The game master receives the response and robots' actions shown up on the map.
- **Error-conditions:** None
- **Necessity:** Must-have
- **Activity diagram:** Diagram 4

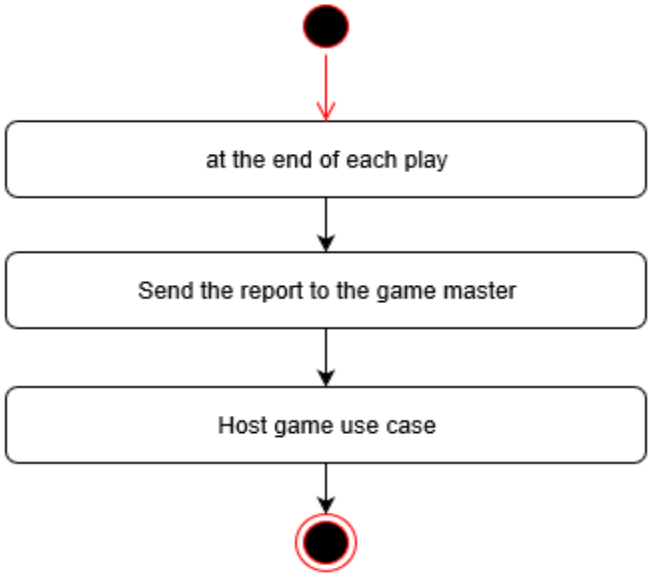


Diagram 4

4.5. Manage robots primary use case

- **Summary:** When the game application is opened, the human player has the right to manage their robots.
- **Actors:** Human Player and Robot Librarian
- **Preconditions:** The game application is opened.
- **Flow of events:**
    1. This use case begins when the game application is opened.
    2. The robots are modified.
    3. The robot librarian updates the robots.
    4. This use case ends.
- **Post-conditions:** Robots updated.
- **Error-conditions:** None
- **Necessity:** Must-have
- **Activity diagram:** Diagram 5

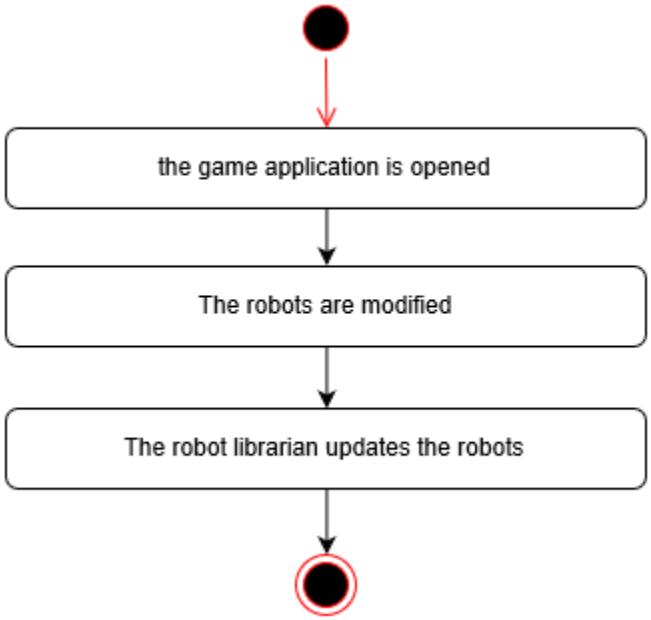


Diagram 5

4.7. End game primary use case

- **Summary:** The game master ends the game. When the game master receives a request to end a game, it determines which team survived and updates robot stats. At last, pop up a window show which team wins.
- **Actors:** Game Master
- **Preconditions:** The game master receives a request to end a game.
- **Flow of events:**
    1. The use case begins when the game master receives a request to end a game.
    2. The game master finds out which team survived.
    3. The game master updates robots' stats and marks that team win.
    4. The game master pop up a window show which team wins.
    5. The use case ends.
- **Post-conditions:** end game and robots' stats updated.
- **Alternate path:**
    1. One team chooses to surrender.
- **Error-conditions:** None
- **Necessity:** Must-have
- **Activity diagram:** Diagram 6

End game secondary use case

- **Preconditions:** Only two team survived and one of them chooses to surrender.
- **Flow of events:**
    1. The game master receives a surrender request from one player.

2. The game master updates robot stats by marking that team loses and all remain robots from that team as destroyed.
3. The game master updates robots' stats and marks the left team win.
4. The game master pop up a window show which team wins.
5. The use case ends.

- **Post-condition:** End game and robots' stats updated.
- **Necessity:** Must-have
- **Activity diagram:** Diagram 6

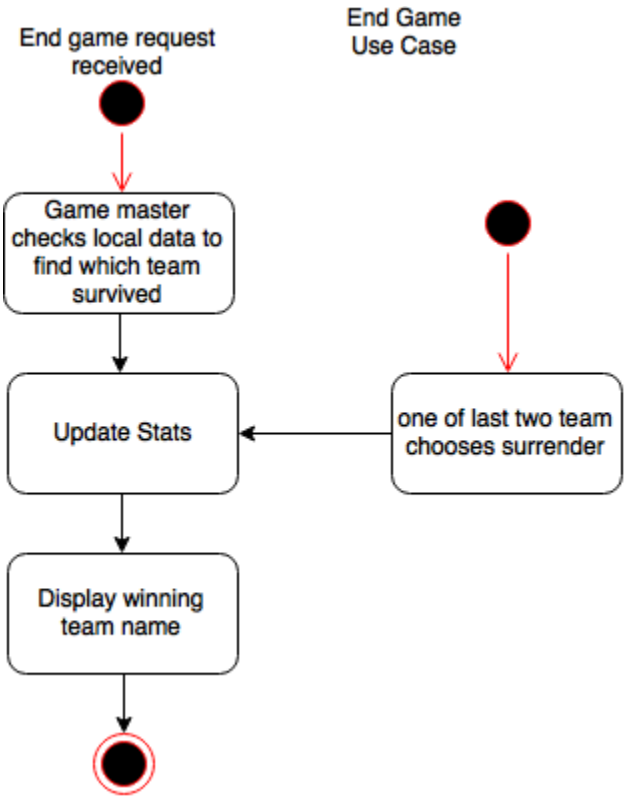


Diagram 6

4.8. Host game primary use case

- **Summary:** After the game started, the game master keeps the game up and running, including display changes on the map, start and end each play, track each robots action and update robots' stats.
- **Actors:** Game Master
- **Preconditions:** The game starts.
- **Flow of events:**
    1. The use case begins after the game start.
    2. The game master starts a play.
    3. The game master finds the robot for this play.
    4. The game master displays that robot's name on the screen.
    5. The game master tracks that robot's action.
    6. The game master updates robots' stats.

7. The game master removes the destroyed robot from the screen.
8. The game master ends a play.
9. Repeat step 2 to 8 until only one team survived, then send an end game request.
10. The use case ends.

- **Post-conditions:** Robots' stats updated, map display changed.
- **Alternate path:**
    1. One team chooses to surrender.
- **Error-conditions:** None
- **Necessity:** Must-have
- **Activity diagram:** Diagram 7

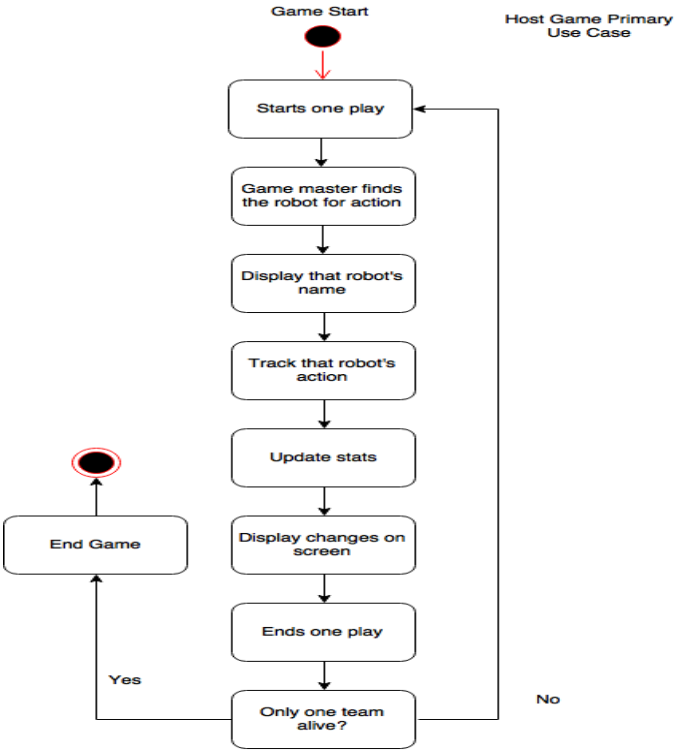


Diagram 7

Host game secondary use case

- **Preconditions:** More than one team survived and one team chooses to surrender.
- **Flow of event:**
    1. In step 5 of host game primary scenario, one team chooses to surrender.
    2. The game master updates robots' stats to mark all alive in game robots of that team as destroyed and lose.
    3. Continue step 7 of host game primary scenario.
    4. The use case ends.
- **Post-conditions:** Robots' stats updated and one team removed from the map.
- **Error-conditions:** None
- **Necessity:** Must-have
- **Activity diagram:** Diagram 8

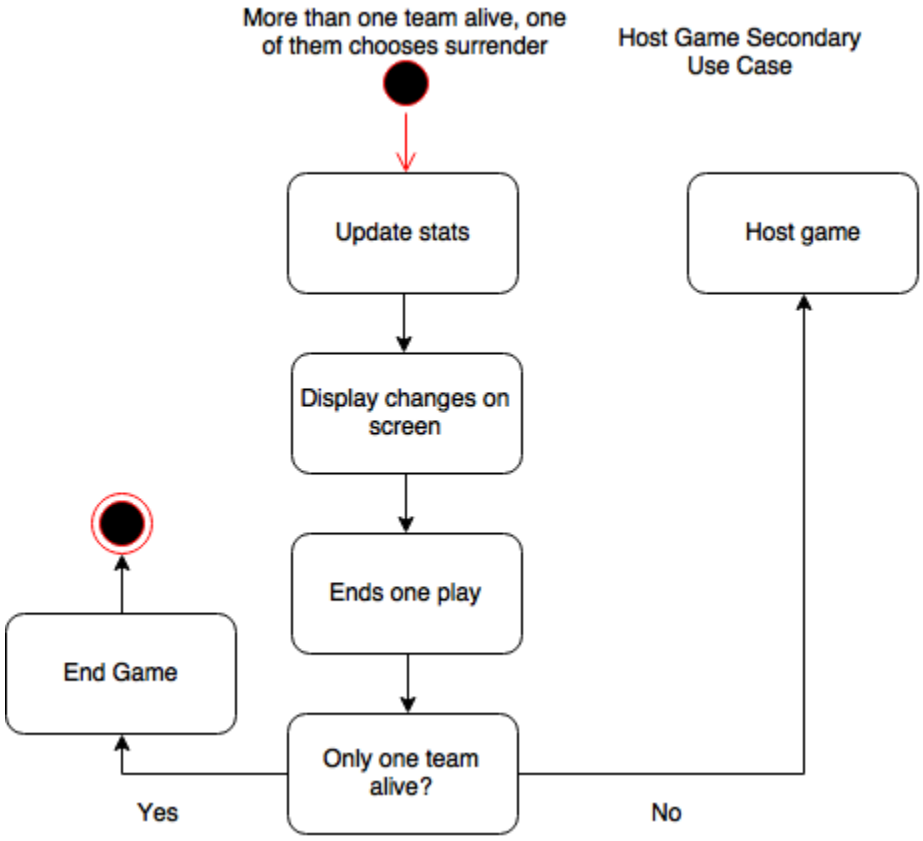


Diagram 8

4.9. Enumerate primary use case

- **Summary:** Display all robots stored and sort them. Show statistics and versions. The user can select sorting method.
- **Actors:** Game Master and Robot Librarian
- **Preconditions:** None
- **Flow of events:**
    1. The use case begins when the game master or the robot librarian receives an enumerate request.
    2. The game master displays all robots stored and sort them based on player's request.
    3. Show statistics and versions.
    4. The use case ends.
- **Post-conditions:** All robots stored are sorted and displayed
- **Error-conditions**: None
- **Necessity:** Must-have
- **Data format:** JSON
- **Activity diagram:** Diagram 9

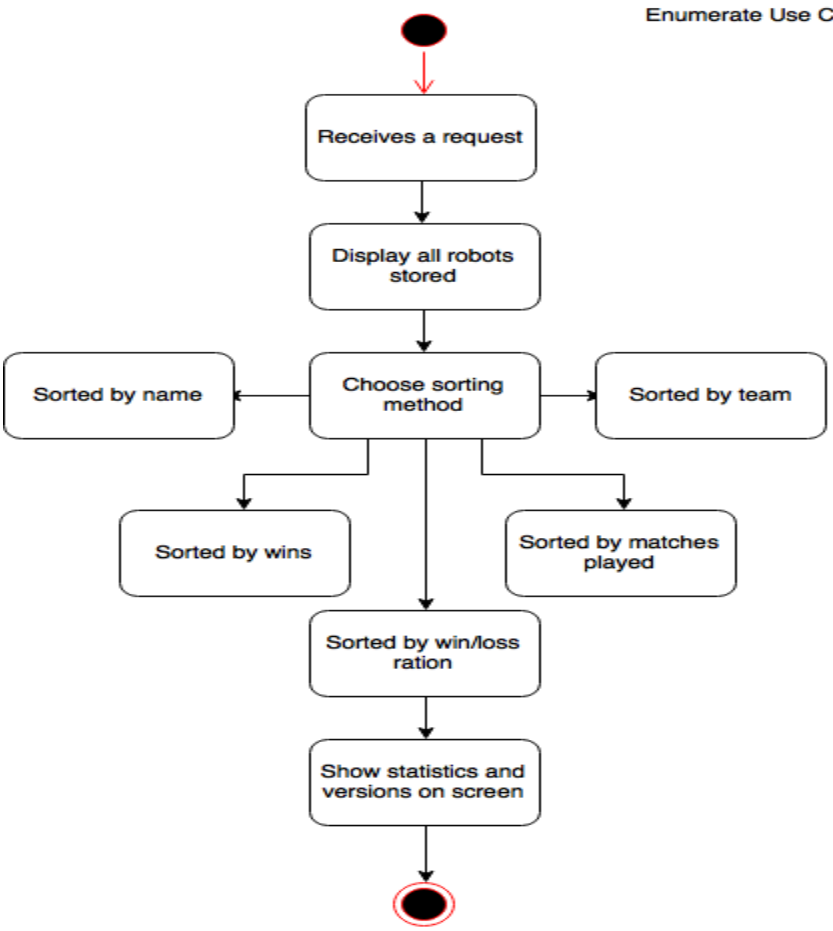


Diagram 9

4.10. Update stats primary use case

- **Summary:** The game master updates detail about stored robot records.
- **Actors:** Game Master
- **Preconditions:** Robots' stats changed.
- **Flow of events:**
    1. The use case begins when the game master tracks robots' stats changes.
    2. The game master updates robot stats.
    3. The use case ends.
- **Post-conditions:** Robot stats updated.
- **Error-conditions**: None
- **Priority:** Must-have
- **Data format:** JSON
- **Activity diagram:** Diagram 10

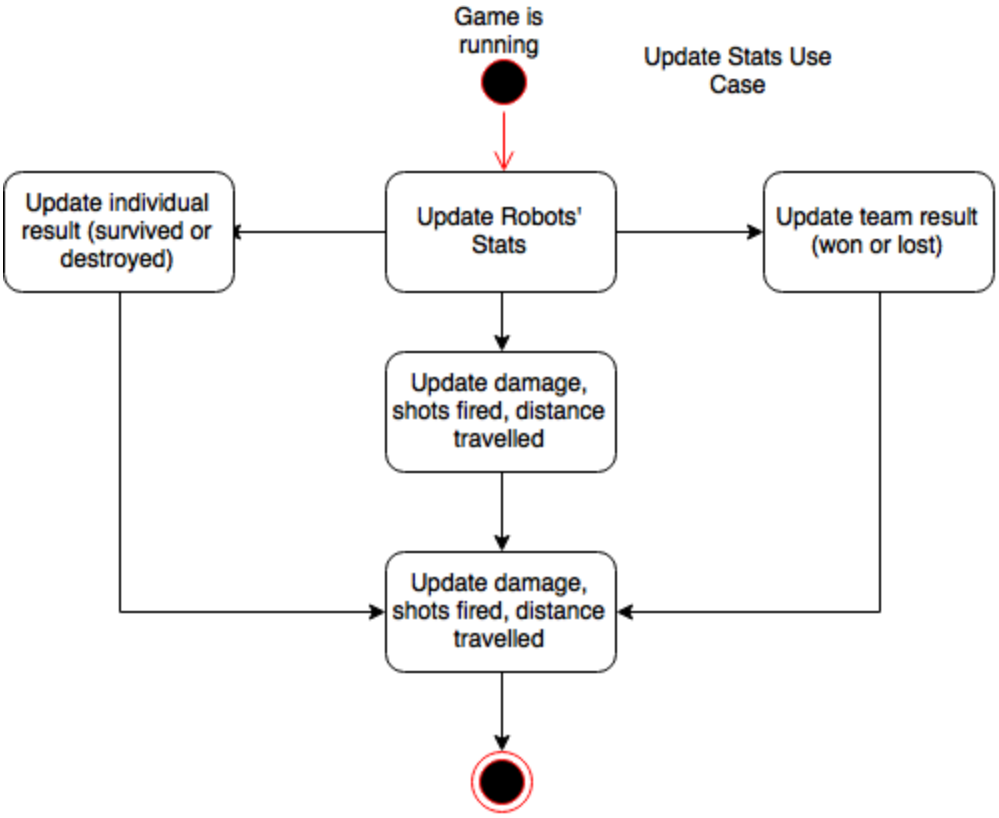


Diagram 10

4.11. Download primary use case

- **Summary:** The game master downloads robot information from robot librarian so simulator can run it.
- **Actors:** game master and robot librarian.
- **Preconditions**: Before the game start.
- **Flow of events:**
    1. The use case begins before the game start.
    2. The game master downloads all robots' records from the server.
    3. The use case ends.
- **Post-conditions:** robots' records downloaded
- **Error-conditions**: none
- **Priority:** Must-have
- **Data format:** JSON
- **Activity diagram:** Diagram 11

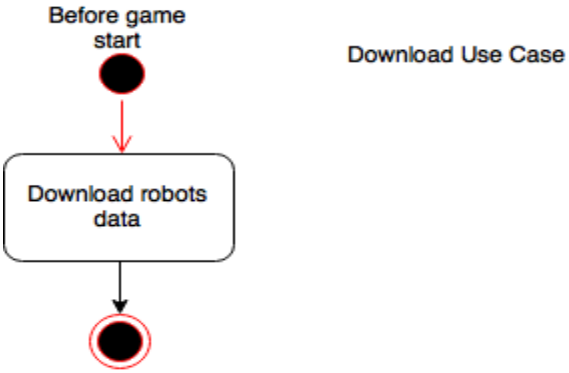


Diagram 11

4.12. Register primary use case

- **Summary:** Add a new robot to the store of robots.
- **Actors:** Robot Librarian
- **Preconditions:** The robot librarian receives the registration request.
- **Sequences:**
    1. The use case begins when the robot librarian receives a registration request.
    2. The robot librarian obtains a robot name from user input
    3. The robot librarian creates a new robot.
    4. The Robot Librarian adds the new-created robot into the store of robots.
    5. The use case ends.
- **Post-conditions:** A new robot is created and added to the store of robots.
- **Error-conditions**: The robot name already exists, go back to step 1.
- **Data format**: JSON
- **Necessity**: Must-have
- **Activity diagram:** Diagram 12

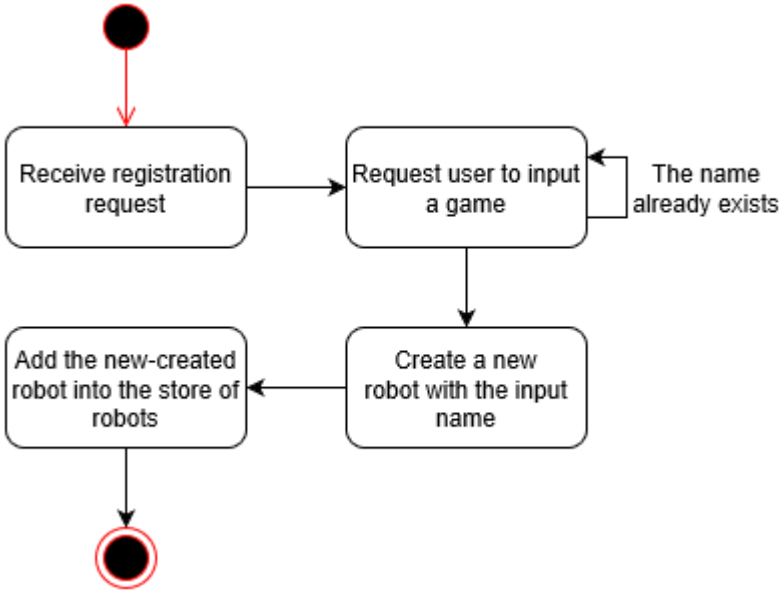


Diagram 12

4.13. Revise primary use case

- **Summary:** Replace an existing robot with the new code.
- **Actors:** Robot Librarian
- **Preconditions:** The robot librarian receives the revision request.
- **Flow of events:**
    1. The use case begins when the robot librarian receives a revision request.
    2. The robot librarian finds out the robot to be replaced.
    3. The robot librarian modifies the original robot.
    4. The use case ends.
- **Post-conditions:** An existing robot is replaced.
- **Error-conditions:** The robot does not exist, show a warning and exit.
- **Data format:** JSON
- **Necessity:** Must-have
- **Activity diagram:** Diagram 13

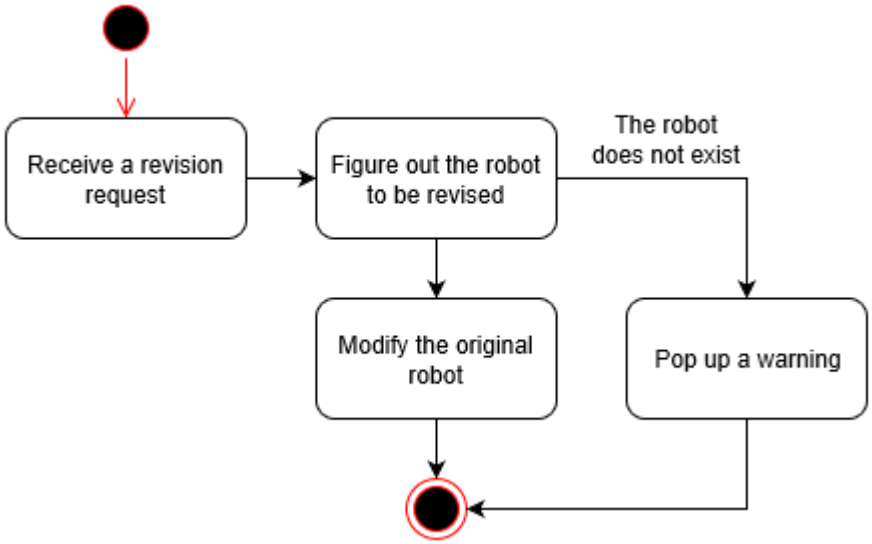


Diagram 13

4.14. Retire primary use case

- **Summary:** Freeze a robot so you can reuse name
- **Actors:** Robot Librarian
- **Preconditions:** receive the retirement request
- **Flow of events:**
    1. The use case begins when the robot librarian receives a retirement request.
    2. The robot librarian finds out the robot to be retired.
    3. The robot librarian marks the selected robot as frozen.
    4. The use case ends.
- **Post-conditions:** a selected robot is frozen
- **Error-conditions:** The robot does not exist, show a warning and exit.
- **Data-format:** JSON

- **Necessity:** Must-have
- **Activity diagram:** Diagram 14

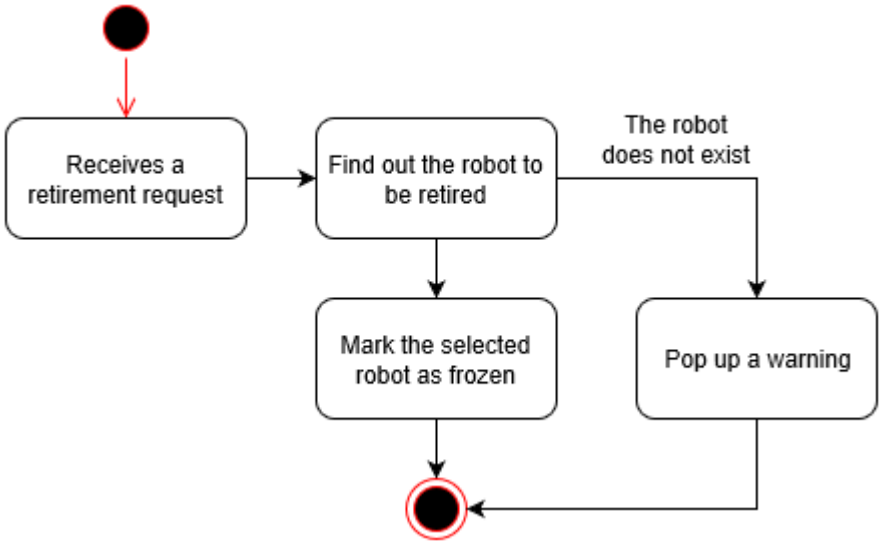


Diagram 14

4.15. Sound effects use case

- **Summary:** Robots action will have unique sound effects.
- **Necessity:** should-have
- **Activity diagram:** Diagram 15

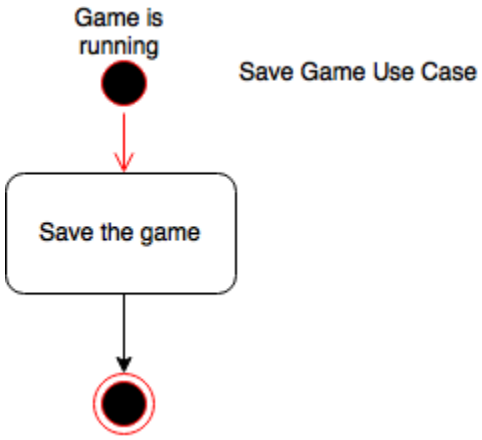


Diagram 15

4.16. Animation effects use case

- **Summary:** Robots action will have unique animation effects.
- **Necessity:** should-have
- **Activity diagram:** Diagram 16

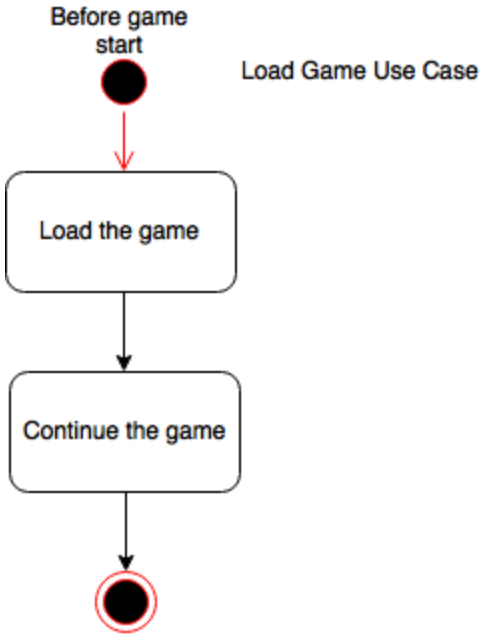


Diagram 16

4.17. Save game use case

- **Summary:** The game can be saved.
- **Necessity:** might-have

4.18. Load game use case

- **Summary:** The game can be load and continue the previous game.
- **Necessity:** might-have

4.19. Ladder use case

- **Summary:** The game might have a ladder to display the robot's ranking.
- **Necessity:** might-have

5. Platform

5.1. Hardware

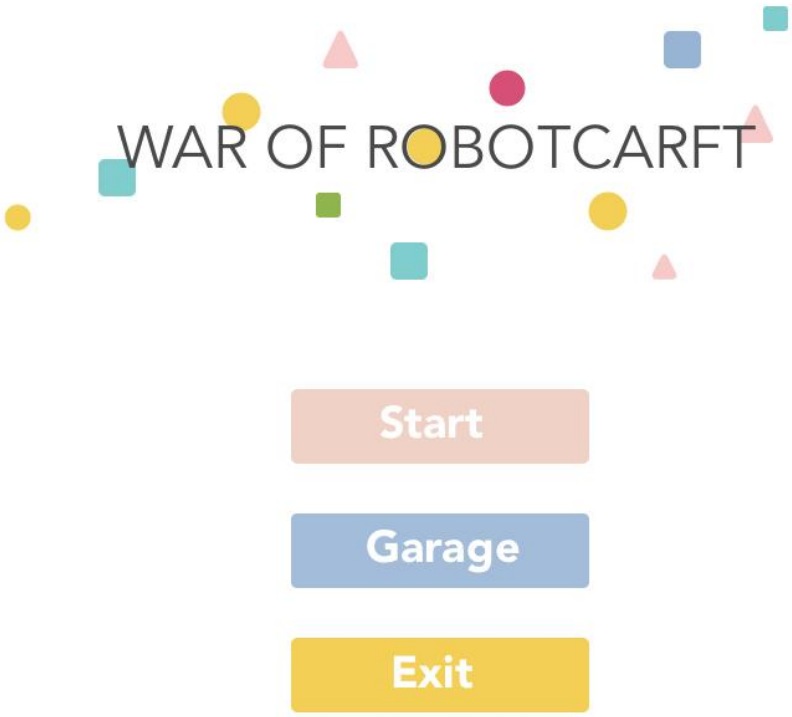
- Monitor
- Keyboard
- Mouse

5.2. Software

- Windows, Mac OS, Linux
- Java JVM
- GIT
- Eclipse

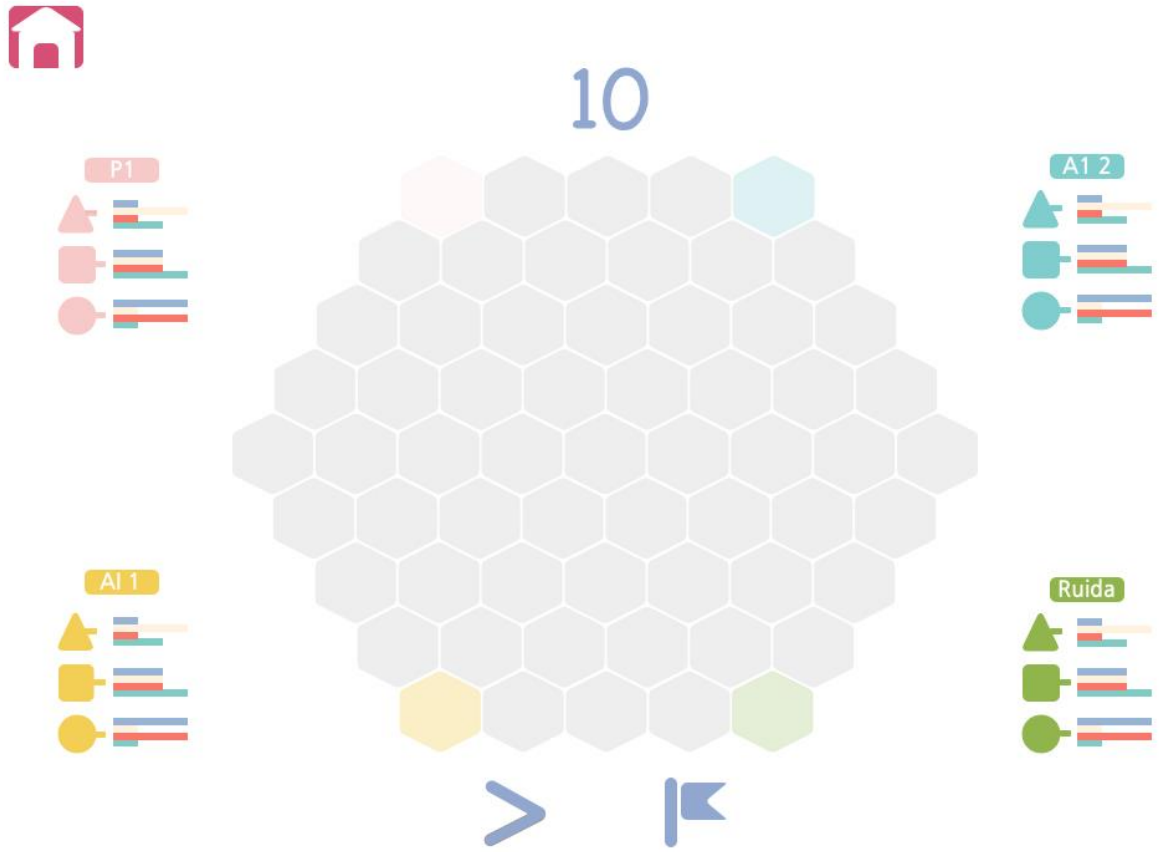
6. Storyboard

6.1. Menu Page



6.2. Start Page

6.3. Game Board



7. Summary

i. This is a **high-level requirement document** for the game project.

ii. The first section is a **game description**. It describes the rules and three modes of the robot game based on professor Dutchyn's notes, along with the map picture and other basics, which is aiming to give readers the concept of the game.

iii. In the second section, **programming tasks** are listed based on the functionality of the game.

iv. Then, a **system diagram** illustrates all how the actors, which are "human player, AI player, game master, and robot librarian", and actions of the game system interact with each other. Each actor comes from analysis of the entire system, and every action comes from demands of interactions of each actor, and sequences start from the action and from scenarios, all of which connect together and become

the entire system and construct the user interface. In this system, actors interact with other actors and the system through actions, and actions can use another action to accomplish a series of interactions.

v. According to professor Dutchyn's lecture: "actions become scenarios", **use cases** of all related actors are presented on the document in order to demonstrate sequences of performing actions and the results. In each use case, sequences diagrams are to illustrate the sequences triggered by a specific action, and the necessity indicates that if a use case is necessary to have or probably needed. Within each use cases, not only normal conditions are provided, but also some possible error conditions are mentioned in case exceptions raise, although solutions of which could be omitted in these rare cases. A few potential alternative paths are provided as well. For each use case, interface elements are provided.

vi. In addition, this document describes **execution environment** of the game including software and hardware requirements, which includes operating systems, developing tools such as IDE and version control system.