Use-case Specification

Version <0.2>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 2021/11/1 | 0.1 | Second draft. | ChenHan Group |
| 2021/11/17 | 1.0 |  | ChenHan Group |
|  |  |  |  |
|  |  |  |  |

# 1. Introduction

#### Purpose

This report describes the use-case model comprehensively, in terms of how the model is structured into packages and what use cases and actors there are in the model.

#### Definitions, Acronyms and Abbreviations

See Glossary.

#### References

Glossary.

Use-Case: < Init the Game >

**Brief Description**

Generating a chessboard situation

**Actor Brief Descriptions**

Player: An player is a user who want to use this game system

**Preconditions**

Player starts to use this game system.

**Stakeholders and Interests:**

1. Players can access quick responses and feedback from game system.

**Basic Flow of Events**

1.Player choose to start a new game

2.Player choose different difficulty.

3.Game system load game resources

4.Game system initialize the game checkerboard.

5.Game system generatres the game data.

# The use case ends.

**Alternative Flows**

1a. Player start a new game.

1. Game system load game resoureces
2. Game system load the game data.

**Post-conditions**

The game run successfully.

**Special Requirements**:

none

**Technology and Data Variations List:**

Mouse control

**Frequency of occurrence:**

It only happens when the player want to play this game

**Miscellaneous：**

none

Use-Case: < Player vs player >

**Brief Description**

Two players play Gobang

**Actor Brief Descriptions**

Player: An player is a user who want to use this game system

**Preconditions**

Player starts to use this game system.

**Stakeholders and Interests:**

1. Players can access quick responses and feedback from game system.

**Basic Flow of Events**

1.Player choose the player vs player game mode.

2.The first player choose whether to fall first or later.

3.Game system load game resources

4.Game system initialize the game checkerboard.

5.Game system generatres the game data.

6. Players fall one after another.

# The use case ends.

**Alternative Flows**

1a. Player instead of start a new game.

1. Game system load game resoureces
2. Game system load the game data.

**Post-conditions**

The game run successfully.

**Special Requirements**:

none

**Technology and Data Variations List:**

Mouse control

**Frequency of occurrence:**

It only happens when the player want to play this game with player vs player mode.

**Miscellaneous：**

none

Use-Case: < Player vs AI >

**Brief Description**

Player plays Gobang with the AlphaXiu AI.

**Actor Brief Descriptions**

Player: An player is a user who want to use this game system

**Preconditions**

Player starts to use this game system.

**Stakeholders and Interests:**

1. Players can access quick responses and feedback from game system.

**Basic Flow of Events**

1.Player choose the player vs AI game mode.

2.The player choose whether to fall first or later.

3.Game system load game resources

4.Game system initialize the game checkerboard.

5.Game system generatres the game data.

6. Player or AI fall one after another.

# The use case ends.

**Alternative Flows**

1a. Player start a new game.

1. Game system load game resoureces
2. Game system load the game data.

**Post-conditions**

The game run successfully.

**Special Requirements**:

none

**Technology and Data Variations List:**

Mouse control

**Frequency of occurrence:**

It only happens when the player want to play this game with player vs AI mode.

**Miscellaneous：**

none

Use-Case: < AI Judgement on putting up chess pieces >

**Brief Description**

How AI choose to put up chess pieces

**Actor Brief Descriptions**

AI: A player who is controlled by the system.

**Preconditions**

The game situation can be initialized. The AI worker works successfully.

**Stakeholders and Interests:**

1.Gamesystem: The AI player can gain a favorable situation for yourself.

**Basic Flow of Events**

1. The player puts up chess pieces.

2. It’s turn to AI player’s round.

3. Firstly AI player starts use the Minmax algorithm and AlphaBeta algorithm to search situations.

4. It firstly decides a depth to search.

5. It search a next situation by putting a step.

6. For a certain situation, AI will give a weight for this situation.

7. If searching is AI's turn and it's the leaf of the searching tree, it will come back to father node and update the α value and the β value of the father node. If α>β, stop putting a step and return.

8. It will stop searching until all the valid situations were extended.

9. Then it will choose a situation which has the biggest value.

# The use case ends.

**Alternative Flows**

\*a. In the searching situation, it find a min node which β value is smaller than it's father node's α value.

1. It will mask this node.

2. Then it won't search the child node of this node.

2a. In the searching situation, it find a max node which α value is bigger than it's father node's β value.

1. It will mask this node.

2. Then it won't search the child node of this node.

2b. Player choose the dismiss army button instead of conscripts.

1. The Game Windows pop up.
2. Player choose the number to dismiss
3. System determines whether the number of dismiss operation is legal.
4. This operation success and this round end.
5. Update the data on this area next round.
6. Show the data of the area next round.

**Post-conditions**

Update relative data. Continue the game.

**Special Requirements**:

none

**Technology and Data Variations List:**

Mouse control

**Frequency of occurrence:**

it only happens when the player implements it

**Miscellaneous：**

None