KAMISADO BOARD GAME USE CASE AND INITIAL REQUIREMENTS Steven Barry & DANIEL kashani

13/02/2017

Table of Contents

[Steven Barry 2](#_Toc474763918)

[1.1 Functional Requirements 2](#_Toc474763919)

[1.2 Non-Functional Requirements 3](#_Toc474763920)

[2. Use Cases 4](#_Toc474763921)

[2.1 Actors 4](#_Toc474763922)

[2.2 Main Use Cases & Diagram 4](#_Toc474763923)

[2.3 Preconditions 6](#_Toc474763924)

[2.4 Main Flow of Events 6](#_Toc474763925)

[2.5 Alternative Flow of Events (Abnormal Expectancies) 7](#_Toc474763926)

[2.6 Post-Conditions 7](#_Toc474763927)

[2.7 Triggers 7](#_Toc474763928)

[Daniel Kashani 8](#_Toc474763929)

# Steven Barry

# Functional Requirements

* The system must implement the rules of the Kamisado board game.
* The game implementation must be able to take commands from the keyboard, whether the game is a GUI or command line development.
* The dragon tower pieces must be able to move either forward or diagonally.
* Players can decide whether to play as white or black, which dictates if they make the first move in the game.
* Multiple difficulty levels of AI to play against which will be either easy or hard mode.
* A Menu system should be implemented to give the main user the option of game type selection, including player vs player or player vs game.
* Speed mode will make both the player and the game make a move within the 5 second time limit, otherwise the game will be over and lost by whomever fails to make a move in the time limit.
* The optionality of “hints” being added into the game may be developed, this will only be available in easy mode vs the computer.
* The possibility of letting the user save their current gameplay status and resuming the game later is to be implemented.
* A score count and rematch feature will be developed letting the user play either best of 3, best of 5 etc.
* User will be able to undo their last move of the dragon tower, letting them reverse the gameplay slightly if a move has been made which is considered bad.
* A scoreboard may be implemented, displaying the fastest times user has beaten the AI in.
* There will be both speed and normal gameplay settings.

# Non-Functional Requirements

* The game itself should not take up too much space of the hard disk drive, doing so may also result in RAM overuse and unsatisfactory gameplay.
* Gameplay must be able and compatible with use in the desktop environment, any other implementation is added optionality’s such as mobile app development.
* Time constraints are placed at each milestone in the development process, at each stage specific deliverables are to be submitted.
* Legalities, e.g. keep in accordance with copyright law, the game must be genuine.
* As the program, will be coded in java it will be easily portable and compatible with and operating system.
* The AI must make a move within the given time limit when user is playing in speed mode.

# 2. Use Cases

# 2.1 Actors

The main and most obvious actors of the system are the player 1 user and the AI game both of which play a main role of keeping within the rules of the game and playing until the game is over. Secondary actors include the optionality of player 2 who will be available for player vs. player game and will inherit most of the behaviours of player 1.

Also, an additional actor may be the server, however as this is not a mandatory part of the system at this stage this will not have to be included as a use case actor.

# 2.2 Main Use Cases & Diagram

* Play game with the 2 “extends” use cases of 1 player game or 2 player game
* When selecting a 1 player game, speed mode and normal mode will be extended cases included in the diagram.
* View Scoreboard of game
* Extends feature of game – save current game
* View Main Menu – included as the user will have to choose from menu options to set up the game.
* Select difficulty will be an “include” of the main menu as the user will have to select level of gameplay.

Kamisado High level Use Case Diagram:



# 2.3 Preconditions

* Player has game installed on system
* Player Knows the rules of the Kamisado game
* Player can navigate using the keyboard

# 2.4 Main Flow of Events

1. Load Game and user is presented with main menu

1. Select difficulty level, of either easy or normal gameplay
2. Select whether game will be 1 player vs the AI or 2 player
3. If one player mode select either speed or normal mode

2. Play Kamisado

1. Select either black or white, indicating who will make the first move
2. Player 1’s turn
3. Make a move of one of the dragon tower pieces in either a straight or diagonal move up the board
4. Assuming the player has made a move according to the game rules update the game status and let either player 2 or AI make the next move
5. If the game is over then generate scorecard of current game results and previous games (Step 3), if not opponents turn i.e. player 2 or AI
6. Player 2 or AI to make a move:
7. Move dragon tower either straight or diagonally, and staying according to the game rules, i.e. the player has to move the colour the opponents piece is currently on
8. If move is valid set the current gameplay status and pieces in correct position
9. Gameplay switches and player 1 now has control of next dragon tower move
10. If game is finished, move to step 3, if gameplay is continuing move back to step 2B

3. Game Over

a) Generate scoreboard and display previous and current gameplay status

b) Update status of game and add tally to count of player 1 success of game

# 2.5 Alternative Flow of Events (Abnormal Expectancies)

* 1) User selects 2 player game, i.e. 1B, however user presses wrong input and meant to select 1 player game vs AI

2) Game has to be restarted

* User selects speed mode 1C)

1. Game is in speed mode, however one of the players whether it be player 1, player 2 or the AI, does not complete a move of the dragon tower within the time limit allocated
2. Due to time being over, the game is over and the opponent has won that specific game

* After both players have made a move of their pieces 2Bi), the game finds itself at a stalemate as both opponents have their pieces at a specific square in the game where neither of them can move their dragon tower
* As this happens the game is over and is called a “deadlock”, resulting in the last player to move their piece losing the game

# 2.6 Post-Conditions

The game is over and scoreboard is displayed to the user. After each game is finished the user will be given an update of their game progress and how many matches they have won and may also display the details of the game if it’s a best of 3 or best of 5 type game. After the user is completely done with the game their progress may be saved if they wish, then the application may be closed.

Also there is the possibility of the additional flow of events and post-conditions if the game is altered at a later date to include gameplay over a server, however at this time it is not a main concern and may be added during the implementation phase.

# 2.7 Triggers

Play Kamisado: load full game application using the executable file and then continue the process of playing the game.

Move dragon tower piece: Using keyboard use specific keys to move in a straight or diagonal direction.

Game Over: if one of the opponents places their piece in the opposite corner the game is over. Also if time runs out on speed mode this will trigger the game to be over and the current player will lose.

# Daniel Kashani