|  |
| --- |
| Hans, David, Zach, Francisco, Ben, Matthew  3 November, 2015 |

|  |
| --- |
| CS 3450 – Software Engineering |
| Software Design Description |
| Chinese Checkers |

Table of Contents

1. Introduction 2

1.1 Purpose 2

1.2 Scope 2

1.3 Overview 2

1.4 Reference Material 2

1.5 Definitions and Acronyms 2

2. System Overview 2

3. System Architecture 3

3.1 Architectural Design 3-4

3.2 Decomposition Description 5

3.3 Design Rationale 6

4. Data Design 7

4.1 Data Description 7

4.2 Data Dictionary 7

5. Component Design 8

6. Human Interface Design 13

6.1 Overview of User Interface 1 3

6.2 Screen Images 13

6.3 Screen Objects and Actions 14

7. Requirements Traceability Matrix 14

# **1. Introduction**

## **1.1 Purpose**

This document intends to describe the architecture and overall system design for a multiplayer chinese checkers game to any interested parties.

## **1.2 Scope**

The game will be multiplayer, with the option for single player; lan enabled, windows 7 compatible, and based on the traditional board game chinese checkers. The goal of which is to move all ten of your pieces to the side of the board opposite of your starting position by moving them into adjacent spots to them, or jumping over other pieces. Benefits of developing this game include; increased portability, ease of finding other players, the potential for additional content (i.e. new rules, varied options for game piece design, etc.), and the ability to play alone against an A.I. opponent. While not a complete redesign of chinese checkers, the game will be a substantial, practical improvement over the physical version.

## **1.3 Overview**

This document is organized using the decimal system. A brief synopsis of first level headings follows. The System Overview section gives a general description of functionality, context and design of the game. The System Architecture section gives a general description of the architecture used for the system (i.e. monolithic, server-client, etc.) The Data Design section covers how data is stored and handled in the system. Component Design provides algorithmic examples of code within the system. These examples may either be sudo code or diagrams. Human Interface will describe the ‘look’ and functionality of the GUI. The end of the document deals with the Traceability Matrix and possibly appendices.

**1.4 Reference Material**

No reference material was used in the writing of this document and no additional material are needed to understand this document.

## **1.5 Definitions and Acronyms**

The game: The software being developed as a virtualization of the board game ‘chinese checkers’.

Player: a human or AI game user.

# **3. System Architecture**

## **3.1 Architectural Design**

|  |
| --- |
| BlackBox.png |
| figure 3.3.1 |

**3.1.1 Overall Software:** We will be using a client-server architecture. The server will manage the game, inform individual clients whose turn it is and what moves have been made. Both the client and server will be built with Multi-tier architectures.

**3.1.2 User interface:** The user interface will be a subsystem of each client. It is responsible for displaying the game board and other data about the game and for receiving input from the user.

**3.1.2 Game Manager:** The game manager will be a subsystem of both the server and each client. It will be responsible for calculating legal moves and enforcing the rules. It contains a game board object. This means there will be local copies of the board on each client and a master copy within the server. It’s functionality will be mostly the same in the server and the client with a few small changes. Within each client the game manager relays information about the board and possible moves to the user interface. Within the server the game manager validates the moves and pushes the last move to each client. The server’s game manager is also responsible for keeping track of who's turn it is and making moves for computer players.

**3.1.3 Mail Box:** The Mailbox will be responsible for communicating back and forth between the server and the clients

## **3.2 Decomposition Description**

OO description is displayed through this Class Diagram

|  |
| --- |
| class.png |
| figure 3.2.1 |

**Activity Diagram**

|  |
| --- |
| activityDiagram.jpg |
| figure 3.2.2 |

## **3.3 Design Rationale**

A client-server architecture was chosen to allow for multiple players to play over a network. Both the client and the server will use a multi-tier architectures because It seems to break the data processing apart in a usable way. More complex architectures would add unneeded complexity, whereas a monolithic architecture would be difficult to implement, and maintain.

# 

# 

# **4. Data Design**

## **4.1 Data Description**

The primary set of data being stored will be the board state, the spaces occupied and by which player. It will be stored as a two-dimensional data structure capable of accessing individual elements as needed, as well as elements in the structure based off another elements, i.e. elements in the same row as another for purposes of moving pieces. Possible structures matching this description would be an array, dictionary, or hashtable, or similar structure.

The game state will also need be to saved, the game state being whose turn it currently is, no additional information is strictly required as everything else related directly to the gameplay, such as win condition, can be calculated as needed from the board state.

## **4.2 Data Dictionary**

board

space[17][8]

Note: only 121 of these spaces will be used, this is just the block that will be needed to fit every space in a manner that is easy to access for computing legal moves without making rule sets for every single space.

move(from, to)

listLegalMoves(space)

selectSpace(space)

highlightSpace(space)

revertHighlighting()

space

spaceState : player, empty, highlighting for legal moves/previous moves

associated GUI element

# 

# 

# 

# **5. Component Design**

**Pseudocode for all the classes**: Reference section 3.2 for the OO description

|  |
| --- |
| gameManager.PNG |
| figure 5.1 |

|  |
| --- |
| serverGameManager.PNG |
| figure 5.2 |

|  |
| --- |
| clientGameManager.PNG |
| figure 5.3 |

|  |
| --- |
| server.PNG |
| figure 5.4 |

|  |
| --- |
| client.PNG |
| figure 5.5 |

|  |
| --- |
| mailBox.PNG |
| figure 5.6 |

|  |
| --- |
| userInterface.PNG |
| figure 5.7 |

|  |
| --- |
| gameBoard.PNG |
| figure 5.8 |

|  |
| --- |
| spaces.PNG |
| figure 5.9 |

# **6. Human Interface Design**

## **6.1 Overview of User Interface**

A gui is provided to allow the user to interact with the software. There will be a start and exit menu item like traditional games have it. Upon start the user can choose to host or join a game. Upon starting a game the user will be able to see a Chinese Checkers board with pieces the color of which is chosen by the game. The user will be able to see other player's pieces and position as well. One piece will be moved at a time. The piece will be moved by left clicking on a it to pick it up and left clicking again on a legal position to move the piece to that spot. The turn will rotate and repeat around all players until a winner has been declared. Upon finishing the match, the game will end possibly bring the player back to the menu screen. The UI will have a ‘help’ menu and a tutorial.

## **6.2 Screen Images**

|  |
| --- |
|  |
| figure 6.2.1 The start menu |

|  |
| --- |
|  |
| figure 6.2.2 Game Board |

## **6.3 Screen Objects and Actions**

When clicking on the question mark the tutorial document will be open to assist the player.

# **7. Requirements Traceability Matrix**

Provide a cross-reference that **traces components and data structures to the requirements in your SRS document** **(We interpret this as only using the requirements that use the components and data structures of our design. The other requirements such as features and some functionality are not included).**

|  |  |
| --- | --- |
| [REQ-0005] This game will display a Chinese Checkers board which is shaped like a six pointed star. | GameBoard class. See section 3.2. |
| [REQ-0010] The board will have ‘holes’ within a hexagonal lattice such that each of the six point shall have ten holes and the hexagon in the middle will have 61 holes and sides of 5 holes long. | The 2d board array contains Space objects.  GameBoard class. See section 3.2. |
| [REQ-0030]The hosting player may be able to add computer players. | The ServerGameManager will have an NPC (Non Player Character) that a human can play against. See section 3.2. |
| [REQ-0035]The hosting player will be able to start the game. | The UserInterface will have the button for this functionality and will interact with the GameManager. See sections 6.1 and 3.2. |
| [REQ-0045] The holes in the starting locations for each player will be filled with ten pieces that are that player's color. | The GameBoard holds the data in the 2D board array. The UserInterface displays the data as colored pieces on the GUI. See sections 4.1 and 6.1. |
| [REQ-0050]Upon starting a game a random player will be chosen to move first. | The ServerGameManager has this functionality. See section 3.1.2. |
| [REQ-0055]Players will take turns depending on the starting locations in a clockwise direction. | The ServerGameManager has this functionality. See section 3.1.2. |
| [REQ-0065]Players shall only be able to move a piece of the same color. | The ClientGameManager handles this limitation. See section 3 and 4.1. |
| [REQ-0070]Players shall move one and only one piece on their turn. | The ClientGameManager will limit the player to legal moves. See section 3 and 4.1. |
| [REQ-0075]A player must make a move on their turn if able. | The ClientGameManager will limit the player to legal moves. See section 3. |
| [REQ-0080]A player shall be able to ‘walk’ a piece by moving it to an empty adjacent hole if there is one. | The ClientGameManager will limit the player to legal moves. See section 3. |
| [USA-005] A tutorial shall be provided in a documentation. See [SUP-0015] | As part of the GUI a tutorial will be provided. See section 6.1. |
| [USA-006] Possible moves shall be highlighted when selecting a piece on a player’s turn. See [LOO-0025]. | The image shows possible moves in a different color from the board. See figure 6.2.2. |
| [SUP-0015] The application shall include a help button in the drop down menu to explain the rules of chinese checkers. | A question mark can be clicked on to see rules about the game. See figure 6.2.2. |
| [LOO-0015] Each of the 6 triangles in the star of the board shall have a unique color from the rest of the starting locations. | See figure 6.2.2. |
| [LOO-0020] The color of the board shall be different from the starting colors. | See figure 6.2.2. |
| [NAV-0001] There shall be an exit button to quit the application on the top right corner of the application. This button shall be labeled with an “X”. | Section 6.1 exit button. |
| [Nav-0002] There shall be a help menu which shall contain the tutorial specified in requirement [USA-005]. | See section 6.3 |
| [IES-0001] The game must allow for multiplayer between 2 to 6 players over a network. | See figure 3.1.1 for networking |