University of Nottingham Ningbo China
School of Computer Science

Academic Year 2015/16

AE2GRP Final Group Report

GRP-PD Advanced Xiangqi

Supervisor: Paul Dempster

Group Member:
Guanqiao HUI (6514888)
Han BAO (6514879)
Meng YUAN (6513307)
Qiwei SUN (6514901)
Yu QU (6513261)
Yuan FENG (6514885)

April 29, 2016

Contents

Chapter 1 Introduction

1.1 Introduction of Xiangqi

Xiangqi, named Chinese chess as well, is an board game that can be played by two players. Xiangqi is mostly played in China and countries borders on China such as Vietnam.
The game represents a battle between two opponents, with the goal of capturing the opponent's *General*. The board can be divided into two parts, the River and Palace, which restrict the movement of certain pieces. The pieces are placed on the intersections of the board lines (‘Xiangqi’, 2016).

1.2 Introduction to Advanced chess

Advanced Chess theory was first given by grandmaster Garry Kasparov during his career (1986~2005). A person and a computer work together to against another team with same compostion. Advanced Chess applied artificial intelligence technologies in computer science. The higher level the AI technique is, the more powerful the Advanced Chess is. Through the game play, the difficulty level could be adjusted and various demo competitors could be created as long as the play level have not achieved the calculating limitation of the program (‘Advanced Chess’, 2016).

1.3 Introduction of the project

The project is aimed to develop an Advanced Xiangqi program for Xiangqi players. Two different applications, desktop and Android version, were developed by the group. Addtionally, an engine was developed to provide intelligent suggestions for next possible moves and warnings of the pieces being captured.

The engine was developed in Java. There are four main aspects of engine, which are board display, move generation, search function and evaluation function. Firstly, the chess board was represent as an array including 90 elements. Secondly, before search, all possible moves would be generated. The warning messages and several checking rules were based on these move generation methods. Next, the search function used the Alpha-Beta pruning algorithm, where mutiple threads were used to implement search continuously. The move suggestions for next step were obtainded from this part. Finally, the evaluation function was based on the fixed value table, which would return a evaluated value of board.

The window of Desktop version UI includes menu, chess board, record board, and a button. The menu includes two functions which are NewChess and Exit. The player could click the chess piece to control the it move and record the location of pieces. Messages such as suggestions of movements and warnings of captured pieces in next turn could be displayed on the record board. By clicking the Undo button, the player could return to previous conditions.
Android version UI, was developed based on the combination of Java, XML and Android packages. It contains a main menu page, a new game page to start game and a help information page to introduce the purpose of this application. In the new game page, there is an Undo button which can perform undo actions. Similar to desktop version UI, Android UI could perform a complete Xiangqi game independently. Furthermore, the combination with engine enabled the functionalities of providing suggestions and warnings displayed on the Android screen. In order to achieve these, multiple threads were used. Different with Desktop version UI, message handler mechanism was used to display the searching results from engine.

Chapter 2 Software Implementation

2.1 Xiangqi engine

2.1.1 Introduction

The aim of this project was to implement Advanced Xiangqi program, which could give the moving suggestions for next step, and the warning of pieces that may be captured in the next step. To implement these functions, a search engine was necessary in this case.

The engine was independent from the UI part, where the engine had its own board representation and own move generation rule. The move generation rule differs from the Android version UI and Desktop version UI, which would be described in Section 2.1.3.2 in, detailed.

The engine implemented its search function based on the idea of calculating the value of pieces of current board, and aimed to find the highest value after current move, which had smallest losses in the subsequent steps. The detail of relevant strategies would be shown in Section 2.1.3, as well as the discussion on the performances of different strategies. Section 2.1.2 describes the background of current Xiangqi engines, and the reused of code in this engine. Section 2.1.4 introduces the instruction of how to use the engine.

2.1.2 Background of the Xiangqi engine

According to the Xiangqi White Paper research group (2014), Professor Shunqin Xu, known as the father of Xiangqi software, began the first study of artificial intelligence on Xiangqi software in 1985, the software of human vs. computer game were gradually developed. In 2004, xqbase (2004) described four protocols to formalize the Xiangqi software, which were Move Presentation, Board Record Rules (Forsyth-Edwards Notation), Chess Book Rules (Portable Game Notation) and Universal Chinese Chess Protocol (UCCI). Then, from 2006, several ‘computer vs. computer’ league matches were hold to compare and enhance the Xiangqi engine, and the champion of National Computer Games Tournament in 2015 was Xiang Qi Ming Shou (XQMS) (Technical Committee of Computer Games and Chinese Association for Artificial Intelligence, 2015).

Currently, most of the Xiangqi engines are written in C++, especially the competition engines and the commercial engines, such as Elephant Eye and XQMS. The engines

written in Java are usually the coursework project written by the students, or the simple project researchers, which are seldom taking part in the competitions and hard to evaluate the performance. Most importantly, there is no Advanced Xiangqi engine currently.

In this project, the Advanced Xiangqi engine was build based on a human vs. computer engine written by Zhijie Lee (2012) in Java, with the MIT license (The download address is available in reference list.). Only the 'AI' class in this Java engine was used to build the Advanced Xiangqi engine, where around 70% codes were reused from the original engine. The modifications were focus on achieving the requirements of Advanced Xiangqi. There were two reasons of selecting this Java engine: the first reason was the engine was easy to modify since the functions were well encapsulated. The second reason was the move generation in this Java engine was advanced, as explained in Section 2.1.3.2.

2.1.3 Implementation of the engine

This section is focused on what strategies were used in this engine. An engine should have a board to record pieces, a method to generate all possible moves in the next step, a search function to find one or several best moves, and an evaluation function to give the evidence of why the search results are better than others. Therefore, this engine could be explained in the following four aspects.

2.1.3.1 Board

The way of data storage in the memory influences the speed of searching, especially the speed of move generation while searching. In this engine, the data-structure used to store pieces was a 90-element array, which kept the original representation method in the original engine. Comparisons were made between the adopted representation and other methods in respect of Xiangqi and reasons of choosing this is demonstrated as follows.

According to Eppstein (1997), there are four methods to represent the international chessboard in general. First method is an 8*8 array of squares, since the international chess board in the real world is 8*8, and within the array, the numbers correspond to different types of pieces. The second one is a 10*10 extended array, which adds a boundary to optimize the algorithm of judging out of bounds in move generation. The third method is '0x88', which is a 136 element array. The difference between the 136 element array and 10*10 extended array is that the 136 element array can use bitwise operators '&' to judge out of bounds in move generation in accordance with the '8 * 8'characteristic of chess. The last method is storing the pieces in bit boards, as shown in Figure 2.1, which means storing each kind of pieces as an integer number. The 64 bit binary value of this integer number represents the 64 positions on the board.

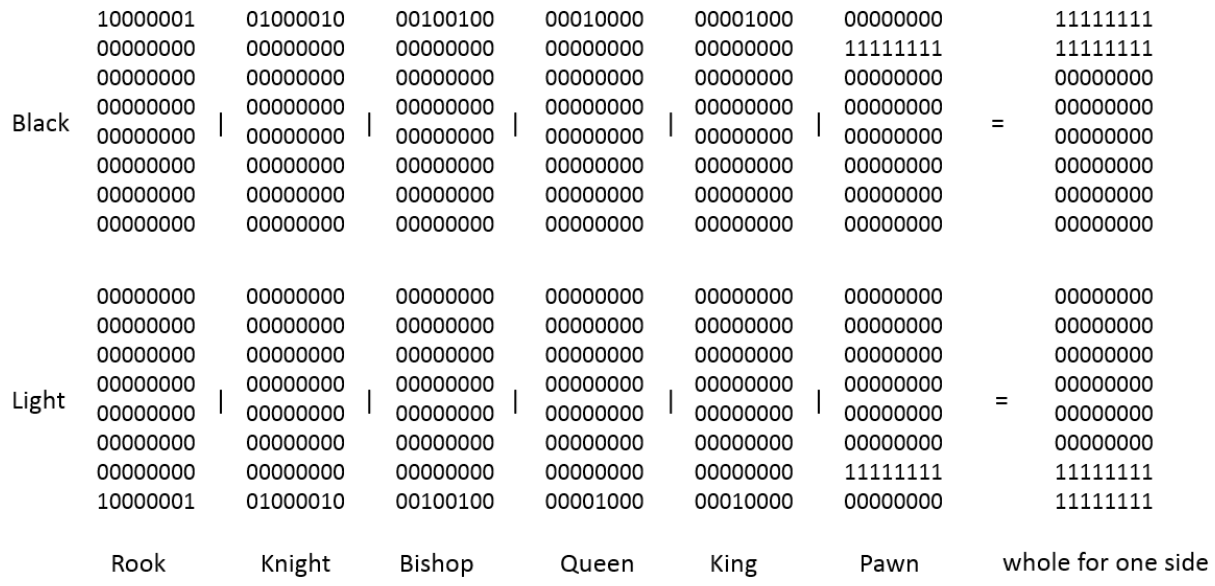


Figure 2.1 These pictures are the examples of bitboards in the international chess.

The left boards represents each kind of pieces that in the initial state, and the right boards are the boards to represent one side by using the bitwise operator '|'. One example is the light rooks are represent as lower left, which is '129' in decimal base.

For Chinese Xiangqi, as well as Advanced Xiangqi, the size of board is 9*10 in the real world, and the size of extended board is 13*14, which equals to 182. There was a version that changing the 90-element board to the extended 182-element board, which added two boundaries on each side. However, in practice, the performance of the extended 182-element board is worse then the performance of 90-element board. There are two reasons: firstly, the UI board, the fixed value tables (used for evaluation) and all the input and output strings are based on the 90-element board. Keeping these boards in a same format reduces instructions for calculations on transformation of coordinates. Secondly, the advantage of extended board is only beneficial to the move generation part. While in this part, using the index references between a 90-element array and a 182-element array is enough to check out of bounds, which means an alternative way using the same number of instructions exists. Therefore, in this case, the 90-element board is better than the 182-element board.

In addition, similar with the extended 182-element board, the advantage of 0x88 is barely beneficial to quickly restrict the boundary of different piece types on move generation part as well. It is more convenient to use the core idea of 0x88 in the move generation part directly, especially in the case that Xiangqi has more boundary limitations than chess. Details will be explained in Section 2.1.3.2.

The bit board was the most difficult method to implement during the development period. The advantages of bit boards are significant, which could create all the

possible moves for one kind of piece at same time. However, there is a crucial problem that increasing the complexity of creating a bit board in Chinese Xiangqi, which is, in the Chess game, the size of the board is 64, where the length is same as the number of bits in the 'Long' type in Java. While in Xiangqi game, the size of the board is 90, which need more than one integer to store it. This not only doubles the number of bit boards, but also makes the complexity of Xiangqi much higher than that of Chess. Large quantities of extra conditional statements will be needed, especially when one move is cross the two integers. Since there is no comparison version between a 90-size board and the bit board, which keeps other functions same, the specific difference on performances is not clear, and need more research in the future.

2.1.3.2 Move generation

During the searching process, move generation is an important aspect that determines the searching speed. The basic method of generating all possible moves is coding a rule for each kind of piece, similar to the rule for pieces in UI version. This method suits for two UI versions, because at each time there is only one piece moving. However, it is too slow to use it in the search engine, which needs to generate all the possible moves quickly at same time.

The original engine already had a skillful move generation function. Firstly, the generating rule was added an offset value on the previous position based on different kinds of pieces. Next, the destination was tested by a 13*14 extended array to determine whether the piece was out of the board boundary. Finally, since Xiangqi has several inner boundaries, such as the 3*3 boundary for generals, a 'legalposition' array and a 'maskpiece' array were used to check whether the move is legal, as shown in the Figure 2.2.

```
public static final int[] legalposition = {
    1, 1, 5, 3, 3, 3, 5, 1, 1,
    1, 1, 1, 3, 3, 3, 1, 1, 1,
    5, 1, 1, 3, 7, 3, 1, 1, 5,
    1, 1, 1, 1, 1, 1, 1, 1, 1,
    9, 1,13, 1, 9, 1,13, 1, 9,
    9, 9, 9, 9, 9, 9, 9, 9, 9,
    9, 9, 9, 9, 9, 9, 9, 9, 9,
    9, 9, 9, 9, 9, 9, 9, 9, 9,
    9, 9, 9, 9, 9, 9, 9, 9, 9,
    9, 9, 9, 9, 9, 9, 9, 9, 9
};

public static final int[] maskpiece = {8, 2, 4, 1, 1, 1, 2};

if (y==-1 || (legalposition[t] & maskpiece[p])==0) break;
```

Figure 2.2 The idea in 0x88 and move generation

In the '*maskpiece'* array, 8 means the soldiers, 2 means the generals and advisors, 4 means elephants and 1 means other kinds of pieces. The '*legalposition'* array represents the legal positions for all the pieces by using '&' operator. For example, the positions in 'legalposition' array which contain the value 9 can only be accessed by the pieces that have the value of 1 or 8 in the '*maskpiece'* array. This method largely decreased the lines of coding, as well as the executing time.

By using these 3 methods, all the possible moves could be created quickly. Although this move generator still cannot generate all moves at same time (where only the bit boards can do this), it optimized the move generating process as much as possible in the case of using a 90-element array.

In addition, the modifications based on the move generation function were obtaining the warning messages, which showed the pieces might be captured in the next step. The idea was if the move destination has pieces in opposite side; a data structure would be used to record this move. In this place, the data structure was simply a string, because the results of the warning message did not need more handling. Only the output string would be enough.

2.1.3.3 General Face and General Check

Compared with the original engine, the Advanced Xiangqi engine added two special rules regarding to the General, which should be in any Xiangqi engines. The first one was the two Generals cannot face to face, which meant if the two Generals are in the same column, in this column, at least one other piece would be needed to be placed between two Generals. To implement this, the first thing was to find one of the General, and kept checking that whether all the positions in this column were empty and whether the first access piece was the other General.

The second special rule was that if the General would be captured in the next step, player's move should avoid the General being captured. This rule was implemented in the part that recorded the warning messages. The idea was to update the player's move first, and then check if the General was still being captured in the next step. If the result was affirmative, then this move would be undone and error message would be displayed.

2.1.3.4 Searching and Thread

One purpose of this program was to provide the suggestions for the next move. Therefore, searching was the core of this project. Based on the research, generally, t two ways could be adopted to implement search. One is analyzing many Xiangqi game data, and quantitates the results as some fixed value tables based on the performance. Then, the search method focuses on finding the highest board value in the next step by a series of calculations. Another method is to search directly on some databases or data files, which contained from large quantity of previous Xiangqi game sources. After each game, the engine would record the results and

apply them to optimize the further searching strategy, such as search order. The first method was applied to implement the search function in current engine, similar as the original engine.

The primary search strategy on current engine was a depth-first searching called Alpha-Beta pruning algorithm, which was similar to the Min-Max search strategy, which finds the highest value compared with one level, but the smallest value recorded in the sub search tree. The difference between Min-Max search and Alpha-Beta pruning algorithm is that the second one can delete unnecessary sub trees in the search process. This method decreased the search time on one level to a large extent and ensured the smallest losses in the further several steps.

On this search function, details were modified to implement requirements. The requirements of search function were to give more than one best move for the next step, keep searching to deeper levels and update the suggestions until the user makes a move. The first requirement was met by giving five best moves, since five suggestions would be sufficient to read and easy to demonstrate. The implementing idea was to sort the moves by result values, and if the move was good enough to be recorded and was on the first level in the searching tree, which meant that it represented exactly the next step on the current Xiangqi board, then this move would be recorded into a data structure. After the search finished, the moves would be recorded from data structure to the string.

In order to meet the second requirement, multiple threads mechanism was needed. Otherwise the user could never make a move since the UI would be blocked when engine was searching for suggestions. This method used here was to make search class to implement *Runnable* interface, and rewrite the *run()* function in this class. In the *run()* function, a while loop was used to control the thread. Search function was put into this loop can stop searching easily and thread would be destroyed when the loop stopped.

2.1.3.5 Evaluation

Evaluation was used to assess the value of current board. Previously, each kind of piece had one fixed value no matter which position it located in the original engine. The modification on the original engine was to give each kind of piece different values in different positions. These values came from an open-source C++ Xiangqi engine, called Elephant Eye in xqbase website. The board value was added to all the same side values and then all the opposite values were reduced from it. The larger the value was, the better the current side situation would be and vice versa. From large to small, the order of average weights on the different pieces was Rook, Horse, Cannon, Elephant, Advisor, Soldier, and General respectively.

2.1.3.6 Limitations

During the testing period, there was a disadvantage of current engine: suggestions mainly focused on exchanging pieces to reach a higher value conditions, rather than to find a way to capture General from opponent side. This situation illustrated that the engine might perform well in the middle of the game, however, at the end of the game, the performance might not be optimal since some win conditions might be missed. One possible solution was to add a file containing various ending situations, and change the search method into a method that kept searching the win conditions in this file. There are some databases of ending situations, however, the open sources file with appropriate license was not found.

2.2 Desktop version user interface

2.2.1 Frame Structure

An overview of the Desktop version UI hierarchy is as follows: firstly, there is a window interface including a menu, a chessboard, a record board and a function button. Secondly, the chessboard controls pieces and records their positions. Thirdly, the record board not only notes each move action down but also shows tips and search results. Last, the engine is packed as a thread running before each step done. The UI design was based on the software requirement specification and considered about the feasibility and practicality. The user interface is shown in Figure 2.3 below.

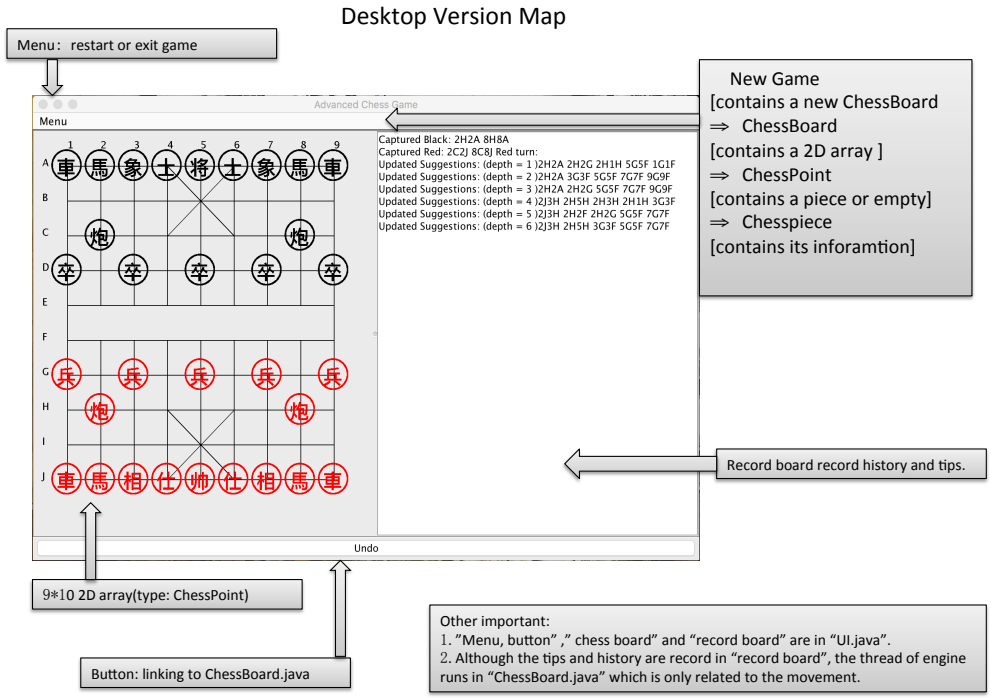


Figure 2.3 Desktop version UI frame structure

2.2.2 Overview of UI design

Desktop version UI used Eclipse as tool and Java as developing language. Swing and Awt packages in Java constructed the whole GUI for Desktop UI. It has four parts, menu, chessboard, button and record board, as showing in Figure 2.4 below.

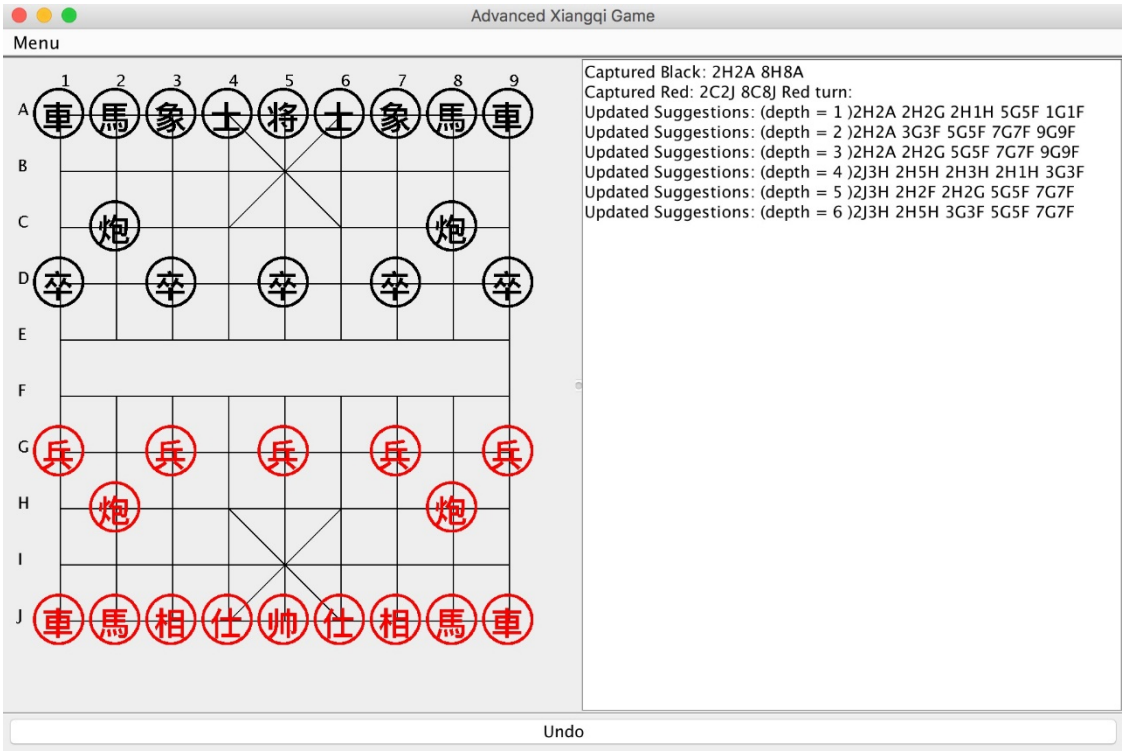


Figure 2.4 Desktop version UI

In the upper left corner, there is a menu bar, where user could exit or restart game in the menu.

The left part displays the graphical chessboard. Most chessboards use numbers or alphabets according to research (Bodlaender, H. and Duniho, F., 2014). Some prefer to lowercase letters and others prefer uppercase letter. In this version, ‘A’ to ‘J’ and ‘1’ to ‘9’ was adopted to give a certain and clear position in the board when the history points listed. In Advanced Xiangqi game, there are two sides, one is black and the other is red. Red goes first and user could choose any side following the rule. If user made an invalid movement, there would be an alert window and then a warning message displayed. When user clicks on a chess piece, the color would change to blue to indicate the piece was selected  .

The right part is the record board, containing the warnings and suggestions and which side is in turn at this time.

In the bottom, an Undo button could cancel the last move action in the history and move the last moved piece back to its previous states.

2.2.3 Implementation of UI

The Desktop version UI includes graphical interface, a changeable board. To construct the graphical chessboard, straight lines were draw. Different from most Xiangqi games, which adopt images as chessboards, lines need more calculation and testing using Swing learnt last semester. Similar to the style of chessboard, circles and Chinese words to represent different kinds of chess pieces in color red or black were created using Swing as well. Undo button in the bottom of Desktop version UI is the only button applied in the interface developed in Java, which could cancel the previous move action made by user and place the piece back.

2.2.4 Implementation of Interaction

2.2.4.1 Chessboard and Chess Pieces

There was a logical chessboard created and combined with the graphical chessboard. The logical chessboard was defined as a two dimensional array in which each coordinate is a position where the chess piece could be placed at. Since the chess pieces were displayed on the screen with Java rather than using images.The pieces were placed in a rectangular area. In this case, the response to the click action of user became more difficult.

2.2.4.2 Record Board Content

Once the game started, record board in the right half part of the UI would start to show the suggestions and warnings from searching engine. There would be one suggestion added to the record board as soon as the engine searched one depth deeper.

Since the deeper the engine searches, the better the suggestions would be, the user can choose to wait for more better suggestions. Particularly in Desktop version UI, it usually takes 10 seconds to do Depth-First-Search which depth equals to 5. Depth would increase to 6-7 at about 30 seconds to 1 minute. With the dramatically growth of search size, suggestions would appear much slower. Additionally, each move action by user would be recorded in the array list and displayed on the record board as well. As the amount of suggestions and warnings becomes bigger, the record board can be scrolled automatically. Furthermore, in order to be user friendly, undo function was designed while there is no chances to perform undo action in the traditional Advanced Xiangqi game.

In this application, when user clicked on Undo button, it deleted the last move recorded and restore to the original state. Error message would be output in the record board to avoid user from repeating undo actions.

2.2.5 Modifications

With in-depth understanding of Advanced Xiangqi and the requirements, the way of applying scientific methods and meet the requirements changed or became obsolete as well. This section mainly introduces the modifications and the reasons to change the original design to current design.

2.2.5.1 Interface graphics

Compared with the original design, the messages output becomes longer which could provide information to users in a clearer manner. Since the messages become longer, the width of record board is adjusted larger to suit the message as well.

Secondly, in order to be more user friendly, a chess piece would be highlighted when user clicks on it which provides an intuitive user experience.

Furthermore, in the original design version, there were four buttons Show, Check, Search and Undo while the current version only remained the Undo button. Reasons are follows: for Search and Check button, it served as displaying suggestions and warnings buttons. Currently, these messages would be displayed on the record board directly; therefore, these two buttons were cancelled. As for Show button, it supposed to highlight pieces, which are in danger of being captured. However, this lead to misunderstandings while the pieces could also be highlighted when it was clicked. Therefore, this button was canceled as well.

2.2.5.2 Thread running

Multiple threads mechanism was adopted to replace the single thread way of implementation because the user interaction would be blocked while engine was searching under single thread structure, which did not meet the requirement specification and was not practical as well.

2.2.6 Combination with engine

2.2.6.1 Implementation of Combination

Since the Desktop version UI can play game independently without providing suggestions and predictions, it was combined with the engine in order to achieve these functionality based on the moves that player takes.

To implement that, multiple threads were needed. Main search engine was set as a thread and is created when all components were initialized at the initialization stage.

Each time user made a move, capture or undo action, current running engine thread would be stopped and new thread would be made to start to run again. The status of current chessboard would be updated to search engine. Engine would search for the suggestions and warnings based on the updated information. As the result was found, engine would pass the information to Desktop version UI and displayed.

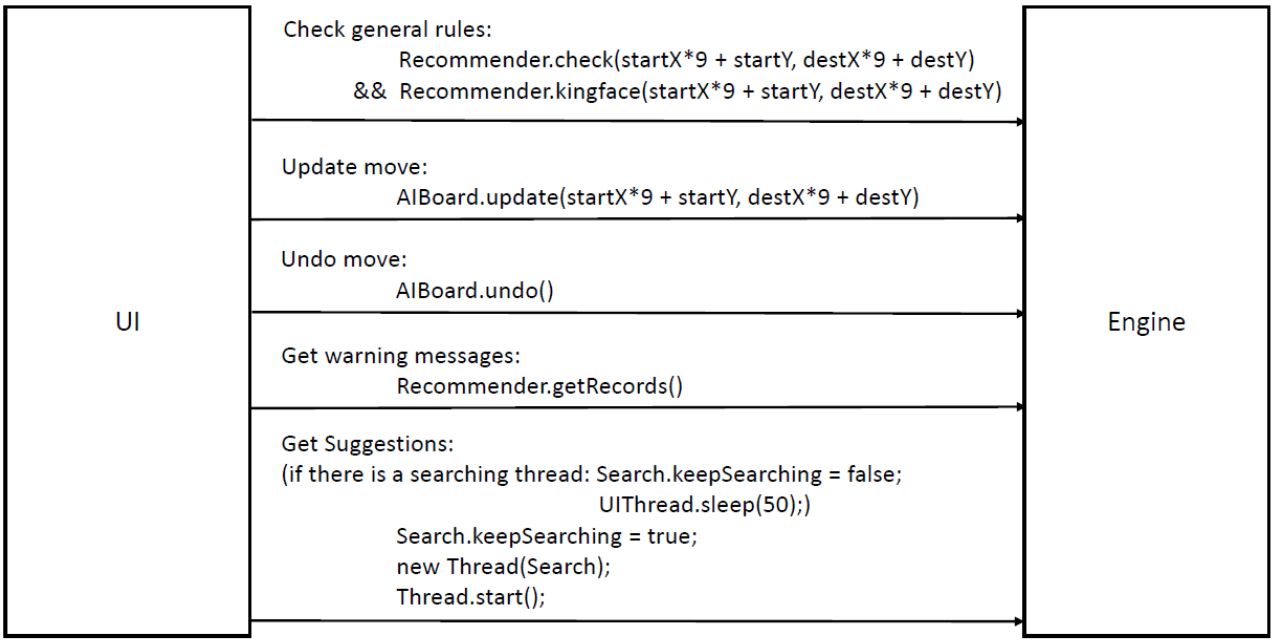


Figure 2.5 How can UI use Engine, where number 9 is the width of Xiangqi board.

As shown in Figure 2.5 above, the engine can be used by Desktop version UI in five conditions: update user move, undo move, get the warning messages of dangerous pieces (which may be captured in the next step), check two special rules related to general, and get move suggestions.

Particularly, before a new move was about to be updated, Recommender needed to be called. Check(startX*9 + startY, destX*9 + destY) and Recommender.kingface(startX*9 + startY, destX*9 + destY) functions, in which the input numbers were the coordinates of start position and destination, and 9 represented the width of the board. These two functions would return Boolean values; where true meant this move was invalid since the own-side General would be captured directly after this move. In this case, an error message would be outputted to notify user to choose another piece to move.

The move update, move undo and warning messages output could be achieved by calling AIBoard.update(startX*9 + startY, destX*9 + destY), AIBoard.undo() and Recommender.getRecords() functions respectively. After the engine board was updated, the previous searching thread would be destroyed by setting the Boolean value “keepSearching” to false, and a new searching thread would be run. Before the new

thread started, the Boolean value “keepSearching” needed to be set to true. Notice that if the two instructions: “keepSearching = flase”(to stop the previous thread) and “keepSearching = true”(to start a new thread) were too close, errors would happen since the previous thread was not totally stopping when the controlled Boolean value was set to be true. Therefore, “Thread.sleep(50)” was used to make sure the previous thread was stopped entirely before new thread run.

As shown in the right part of Figure 2.6, the raw output of warning messages and move suggestions was a string containing the related coordinates of pieces. For example, 1710 means (1, 0) would be captured by (1, 7) in the next step, or the suggestion of moving from (1, 7) to (1, 0). For the user-friendly purpose, the final output was changed in the implementation of two UI versions.

```
0  50 30 20 10 60 10 20 30 50

1  77 77 77 77 77 77 77 77 77

2  77 40 77 77 77 77 77 40 77

3  00 77 00 77 00 77 00 77 00

4  77 77 77 77 77 77 77 77 77

5  77 77 77 77 77 77 77 77 77

6  01 77 01 77 01 77 01 77 01

7  77 41 77 77 77 77 77 41 77

8  77 77 77 77 77 77 77 77 77

9  51 31 21 11 61 11 21 31 51
================================
   0  1  2  3  4  5  6  7  8
```

```
Captured Black: 1710 7770
Captured Red: 1219 7279
1710 1716 1707 4645 0605
1710 2625 4645 6665 8685
1710 1716 4645 6665 8685
1927 1747 1727 1707 2625
1927 1715 1716 4645 6665
1927 1747 2625 4645 6665
```

Figure 2.6 The left is a board for testing with the initial state.

The right is the raw output from the engine, recommendations.

2.2.7 Drawbacks

Actually, Desktop version UI spent the longest time and indeed had enough time to solve all problems. However, there still remain some aspects that are not as ideal as expected.

The first is that the UI design is not good enough while barely using Swing and the display style of record board could be improved as well.

Secondly, the codes still can be refactored to a more concise way and the structure of the Desktop version UI could be improved as well.

2.3 The Design and Implementation of Android Version User Interface

2.3.1 Sources of code

The Android version user interface was developed using Java and XML with combination of Android packages. The Android application was then combined with engine to achieve the functionalities of searching moving suggestions and warnings of pieces being captured. For Android version UI part, except the basic frame generated by Eclipse automatically and the rule of Xiangqi modified from Desktop version UI, the remaining code was created and combined by group members.

2.3.2 Overview of UI design

Figure 2.7 below shows the main menu displayed after the application starts. There are three buttons placed in line vertically, which represent three major functionalities of the Android Xiangqi application: start a new game, displaying the introduction of this application and exit the application immediately.

When user clicks the New game button, a new page with a game play scene would be jumped to. The Help button leads to a jump to a page containing the description of the purpose in designing this application. The Exit button could end the execution of the game and return to Android home screen. Figure 2.7 and Figure 2.8 describes the main menu and help page as below.

Figure 2.7 Main Menu

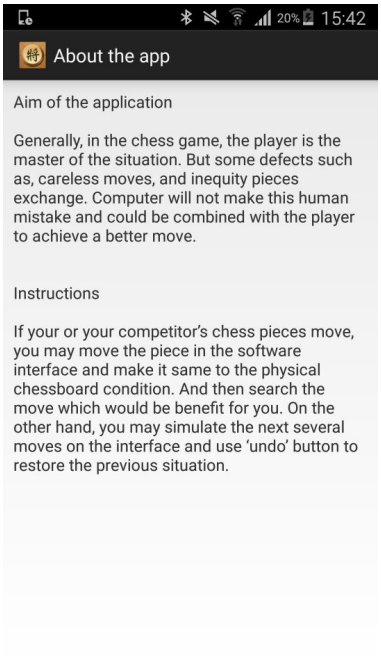


Figure 2.8 Help Page

Particularly, in the page that New game button jumps to, which plays the most important role in implementing the game, a chess board picture is set as the background and all chess pieces are displayed in their initial positions. There is an Undo button under the chessboard, which performs undo action. Simultaneously, the searching results from engine, which indicates the next suggested moves, would be displayed at the bottom of the screen. User can scroll this part to see more suggestions when size of suggestion field exceeds the boundary of the screen. Warnings indicates which pieces are in danger of being captured are displayed at the top of the screen. The UI is displayed as Figure 2.9, Figure 2.10 and Figure 2.11 below.

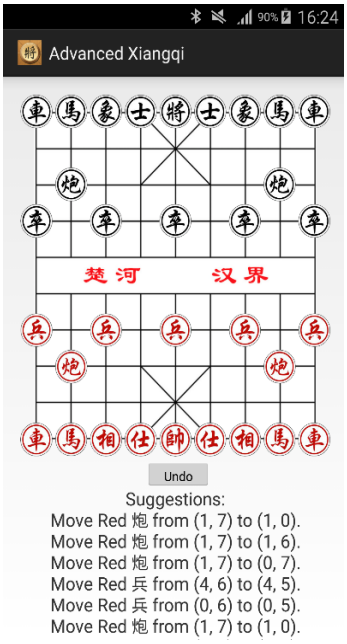


Figure 2.9 New Game

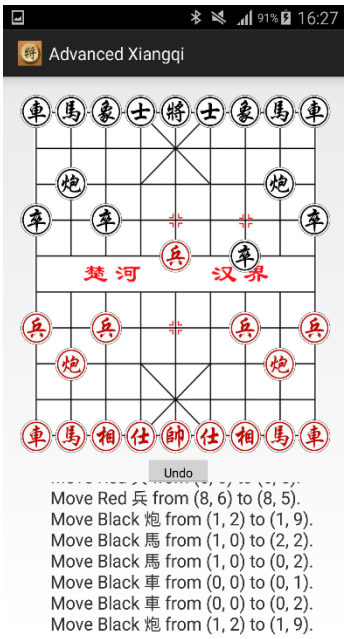


Figure2.10 Scroll Suggestions

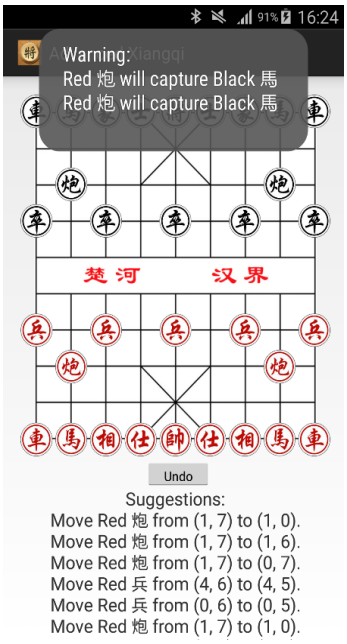


Figure 2.11 Warnings

When the piece is clicked, it will be highlighted as Figure 2.12 below.

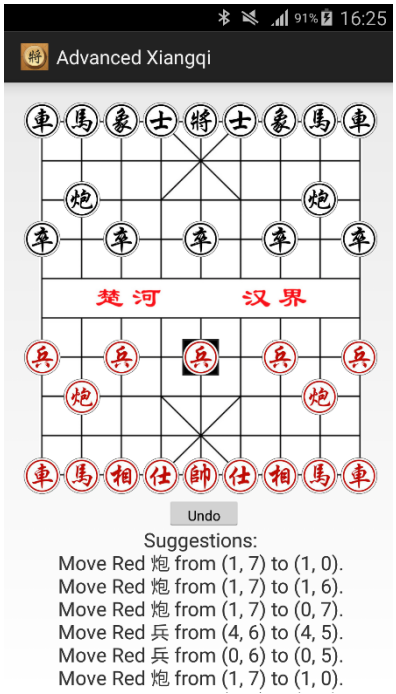


Figure 2.12 Click and Highlight

In this application, user can move a piece to a blank position or to capture an opponent piece by clicking the start position and destination in turn. Additionally, the previous move or capture action would be canceled and pieces would be moved back to their previous positions when Undo button is clicked. The UI before and after move and capture undo action is described in Figure 2.12, Figure 2.13, Figure 2.14 and Figure 2.15 below.

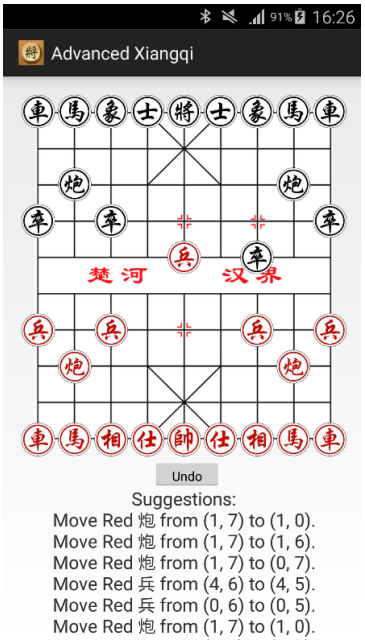


Figure 2.12 Before Move Undo

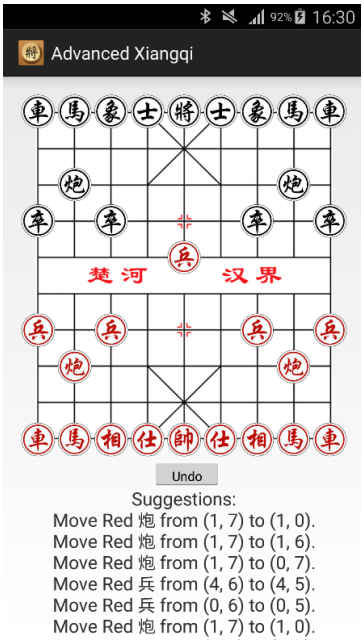


Figure 2.13 After Capture Undo

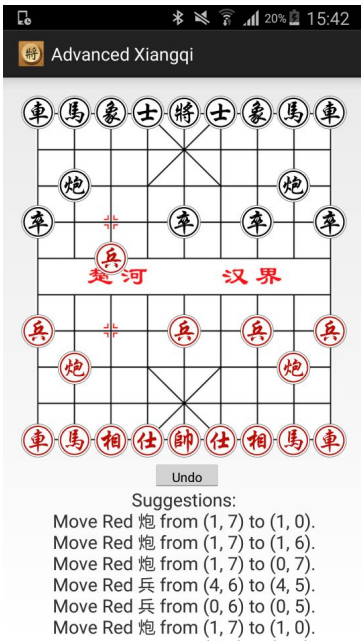


Figure 2.14 Before Capture Undo

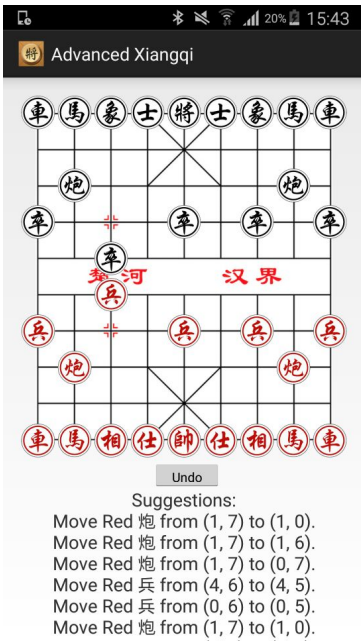


Figure 2.15 After Capture Undo

2.3.3 Implementation of UI

To implement user interface, Java and XML were applied with combination of Android packages.

For page from clicking “new game” button, in XML, physical chessboard and chess pieces were created and related properties such as sizes, ids and so on were specified. Java was used to implement the initialization and positioning of all pieces. If the Undo button was clicked, pieces would be moved back to their previous states and interface would be updated to achieve the display of the update of pieces positions. At the beginning, drag action was used to realize the movement of pieces. However, since the size of screen was limited and drag was not user friendly to some extent, a click action was replaced to perform the move of pieces with consideration of user experience.

There was a logical chessboard initialized in Java part as well. In order to achieve the display of initialize, move and capture pieces, this logical chessboard was initialized with combination of the physical chess pieces created in XML before, which was related to the implementation of detailed interaction with user and would be discussed in the section below.

For pages from clicking “help” button, the predefined output string in XML would display on the screen. After clicking Exit button, the page would be closed and jumped back to the Android home screen.

2.3.4 Implementation of Interaction

The whole Android application was event-driven. Each new window page was a new activity and each activity could have many objects, in this case, each chess piece was an object. We designed these pieces as buttons so that it could improve the efficiency since we could obtain which button was clicked directly rather than deciding by calculating their positions.

2.3.4.1 New game

In the page of start a new game, there was a physical chessboard handled by XML and Java files, combined with a logical chessboard handled by Java. They were both created at the beginning of the new game activity. Every time a chess piece, which was predefined in corresponding XML, was initialized, there would be a logical piece with physically created piece’s id assigned to the logical chessboard. In the initialization of logical chessboard, the positions of the physical chess pieces would be assigned together.

The handler of touch events implemented the main body of game play. It distinguished whether the user performed click on Undo button or on a chess piece. If the Undo button was clicked, it would undo the previous move step, if it existed. If user performed the click action on chessboard, a sequence of validation and move actions would be taken. To explain better, an example piece move and its execution steps would be described.

If user clicked on a button on the chessboard, program would first check whether it was a valid chess piece on user's side. If not, error message would be displayed and user allowed choosing again. If a valid piece was selected, it would be highlighted and then user was allowed to choose the second position, where user would like to place the piece at. After the second position was clicked, the rule would be checked to decide whether it is valid or not. If valid, corresponding move or capture actions would be taken. If General was captured, game ended and end message would be displayed on the screen.

After any update related actions taken, the whole view would be repaint and the change of positions would be displayed immediate after the destination selected.

2.3.5 Combination with engine

2.3.5.1 Implementation of combination

Most of the combination methods in Android version UI were implemented in the same way as those of Desktop version UI. There are some differences exits as follows.

When the Android version application started, UI thread would be executed which in charge of UI and monitoring the click events from user and response. Since the threads were insecure in Android, UI could only be updated in UI thread which prevented the suggestions from displaying on the screen.

In this case, message handler mechanism was implemented to achieve the communication between engine thread and UI thread. Message from engine thread was posted to the UI thread and therefore it could be displayed on the screen.

2.3.6 Comparison and drawbacks

Compared to requirement specification specified at the beginning of Android version UI, our application can meet the requirement basically such as performing complete Xiangqi game, giving suggestions and predictions while playing and so on.

However, due to the limitation of time and knowledge of Android, there still remain many aspects that have the potential to be improved. For example, appearance of the interface can be designed and organized in a more good-looking way. Sounds can be

added to the application while game playing: background sound keeps playing and special sounds may appear when user takes specific actions such as move, capture a normal piece or capture the Generals. Additionally, timer can also be added to show the time count down for one side turn or simply show the time counting for the whole game.

Chapter 3 Software Engineering Development

3.1 Software Development Strategy

The software strategy used in the project was V model as the Figure 4.1 shows below:

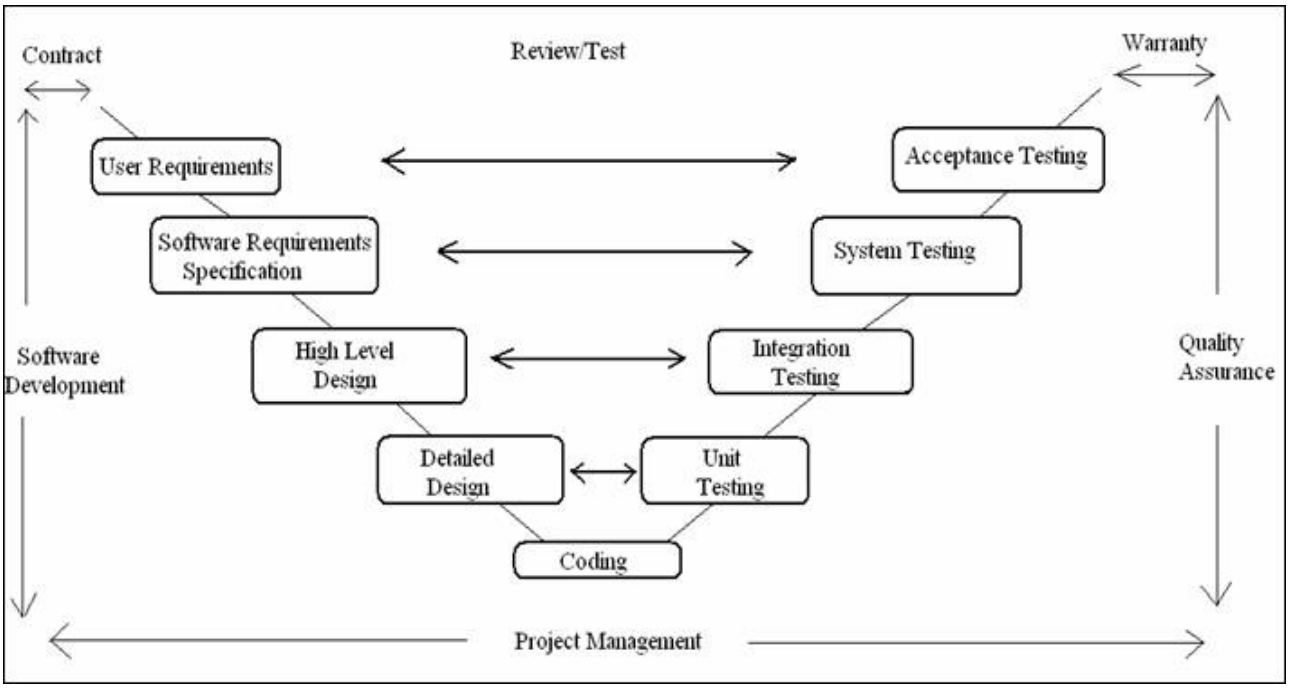


Figure 4.1 V model (Sqtcadmin, 2012)

V model demonstrates the relationship between the development and testing. On the left of model lists the procedure of software development and the procedure of software testing on the right. At the bottom of model is the core step, coding.

All things considered, the V model is symmetric logically, which seems to us that is convenient to perform the testing stage as soon and early as possible once the development activity begins.

3.2 Software Requirement Specification

3.2.1 Introduction

As mentioned in Chapter 1, this Advanced Xiangqi software is used to assist the user to analyze the Xiangqi game strategy through providing recommendations and

suggestions. The design of initial target was the version that the software runs on a desktop version, however, it would be much more considerable and convenient for the users if the software could be used on a smart phone. Therefore, we decided to also attempt to develop Android version of the Advanced Xiangqi application. Even though Android development is a unfamiliar field that we never learned before, we determined to work out a workable and user friendly android application with the motivation of provide convenient software to users.

The software requirement specification is based on the discussion among the project group members and the supervisor. It is divided into two parts: the functional requirements and the non-functional requirements and the requirements are distinguished by the indispensable and optional ones. There are also explanations about the why those requirements are important and analyses about the risks and challenges to achieve them.

3.2.2 Scope

The requirement specification contains the requirements for the Advanced Xiangqi program in desktop version and android application version in the software engineering group project.

3.2.3 Functional Requirements

The most basic function is that the software must allow user to update moves on the chess board interface for both the user side and the opponent side. In addition, the user must be able to undo a move if the piece is moved by mistake. These two requirements are critical for the software because, firstly, the Xiangqi engine needs the updates of both sides to search the recommended moves and analyze the pieces in danger of capture or which opponent's pieces can be captured by user. Secondly, sometimes the user wants to try several different moves to compare and select the most useful one. Thirdly, it is very likely that user will move a piece by mistake and as for the move disobey the Xiangqi rules the software must undo the move automatically and show a warning text.

Another indispensable functional requirement is that the software must recommend moves for the user side and predict moves of the opponent side. This function is aiming at broadening user's view of thinking about the game he is playing through providing user with possible moves from user side and opponent side. Besides, in order to better help the user to analyze the game, the software should also figure out the potential captured pieces or the ones of opponent side that can be captured and the interface should contain warnings for the pieces potentially be captured or the potential danger of losing the game. This function is used to help the user realize the potential danger and

the possibility of beating the opponent which he may overlook when analyzing the game by themselves.

There are some challenges to achieve those function. The running speed of the software tend to be slow as the depths of search increasing and the speed is various when the software runs on different devices. So it may contribute to the inconvenience that two users using the software on different devices and obtain the recommendations in different speeds. Other challenge is that ideally, the number of useful recommendations of moves should be as much and useful as possible, however, as the Xiangqi game going on, the pieces left will be less and less so the number of recommendations will also reduce.

Besides, in order to satisfy the requirement that some users want to review the game they played before, the software should record and save every move in the game and every round of game played on the software. However, this function is only an optional one because it is unnecessary for helping user play the current Xiangqi game.

3.2.4 Non-functional Requirements

Apart from the functional requirements mentioned above, there are also some non-functional requirements which are indispensable for enhancing the user-friendly nature of the software. Firstly, the software must keep searching and providing for the possible moves until the user makes a move to interrupt the search. This requirement allows the software to provide recommendations continuously during the time period when the user is thinking. The challenge is that we need to use multi-threading to achieve this requirement. The user interface and search procedure should be in separate threads which increases to the complexity of the project. Secondly, the chess piece must be highlighted in the interface if the user click it successfully and be located into correct position automatically if the user not click on the exact position. The reason for this compulsory requirement is that it will inform the user whether the click is valid and avoid the mistake of improper clicks. Thirdly, the device should keep un-lock during the game playing time. The reason for this requirement is that it is inconvenient for user to un-lock the computer or smart phone and this also disturbs the user when they are thinking of the game. Finally, the interface should be compatible on different android devices with various sizes and ratio of screens. High compatibility increases the scale of the software use but generate challenges of finding out a proper method for interface auto-adaptation.

By the end of the project, the group achieved the compulsory functional requirements: moving piece, undoing move and providing recommendations, predictions as well as warnings on both desktop version and Android version and the non-functional requirements: keeping searching, implementing multi-threads and highlighting chess piece.

3.3 Software Quality assurance

After achieving every certain requirement, specific group members were in charge of the quality checking of the software. The checking contents included whether the function met the user requirements and whether the software could deal with some improper usage.

3.4 Software Testing

Unit testing:

After the process of coding, we already obtain a primary product that can implement the chess pieces activity as an example of move and capture. The first testing stage is unit testing. Unit testing aims at examining the most fundamental methods in the class. There are several methods requiring to be tested as follows:

isWin

It is a method to judge whether any player win the game according to if the Generals (using G to represent later) is captured by opponent. If in the second position that user clicks sets the chess piece G, it will return true. As a result, one string “XXX lose” will present on the screen. In unit testing, the result is reverse for the first time. Fortunately, the reason for that is the judgment of whose turn went wrong which is not difficult to correct.

isOpponent

It is a method to judge whether the chess piece user chooses is opponent. First of all, the method examines the target chess piece, in other words, the piece located in the position that user chooses to move to is red or black. After that, calculate whose turn currently using the clicktime. It performs well in the unit testing.

isChessExist

It is a method to judge whether there is a chess piece in the target position by two traversals of array which contains the red and black pieces. Everything is normal as expected.

isValidMove

It is a significant one among overall methods on account of that it determines the correctness and regularity of the game. Every pieces, including G, Rooks(using R to represent later), Horses(using H to represent later), Elephants(using E to represent later), Advisors(using A), Cannons(using C to represent later), Soldiers(using S to represent later), should obey the corresponding rule, both movement and capture.

Because of the variety of rules, it must be tested respectively. For example, R can move in a straight line. That is to say, either the horizontal coordinate or vertical coordinate should keep consistent. Therefore, supposing that both coordinate changes after moving, it will return false. There are plenty of rules should be covered. As we perform testing carefully and comprehensively, we make a complement.

move

It is a method to implement the movement of chess pieces. We performed testing by detecting the existence of pieces in the position that user choose before and after the movement. It is successful to pass the testing.

beingCaptured

It is a method to implement the capture of chess pieces. In essence, it consists of two parts, moving own piece and disabling target piece of opponent.

Integration testing:

The second testing stage is integration testing, whose purpose is to examine the accuracy of subsystem and combination of basic methods. As an instance, we intend to test whether the second step that user choose is valid. The testing method should consist of three methods before, isOpponent, isChessExist and isValidMove. As a consequence, there is not severe problem and testing performs well.

System testing:

The third testing stage is system testing, aiming at detecting whether the functions and requirements satisfies.

The requirements contains functional requirements and non-functional requirements. The main functional functions we implement contains undo, searching and error text. After the process of testing, there are several issues existing in Undo, including the issue of code, for example, that the counter of click times may go wrong and the issue of usability, that we can merely encourage users to undo one step. We correct the counter to make sure that Undo can work normally. Nevertheless we can’t implement more-step Undo for the moment.

The non-functional requirement consists of two aspects, keep searching, supplying warning of dangerous pieces. Some haven’t been implemented yet. As a result, it performed well, showing the correct warning on the top.

3.5 Software Evaluation

After implementing all the requirements, we set about asking several end users to use and evaluate the application. The end users we choose must have a deep knowledge of

the rule of Xiangqi. There are two versions required to be evaluated. Some problems appear during the process of evaluation.

One version is the desktop version. The main and common problem that most end users put forward is that the suggestions and prompts exhibited on the right column can hardly be distinguished immediately because the program listed the coordinate of chess pieces. They had to read the coordinates and find the position it indicate. It is impossible for them to keep on doing it during the game. Instead, they preferred to read the prompts when they get stuck in a dangerous circumstance. However, it can usually hardly prevent losing the game in that it is too late. There are other valuable feedbacks. One end user pointed that popups showing the wrong messages should be replaced by the transitory labels appearing for several seconds. Because he thinks it boring and meaningless to click “OK” to disappear popups every time. Another end user advices that it should show all the possible position of that the chess piece that user choose can move to. Another end user presented that sometimes it didn’t work when he clicked the chess pieces. The interaction between user and application can be improved.

Android version is the other version. The evaluation from end users is almost similar with the desktop version. That it takes too much time to identify the prompt is the primary problem.

In general, our application is satisfying among all the end users except the inconvenience of checking the prompts right. They can get used to the application soon and benefit from the application.

Chapter 4 Project Reflection

Overall, the research and implementation process were closely related to the software requirements specification We tried to enhance the user friendly in both the general software design and the engine and user interface details. However, there still existed some problems and short comings during the project development process. The following of this chapter contains the reflection of the project in terms of software technique and team work and management aspects.

4.1 Technique reflection

Based on the research about the Xiangqi engine, As mentioned in Chapter 3, the algorithms in a workable engine are highly complex and require high level knowledge that we cannot learn and apply it within the time period planned to finished the engine. Therefore, we decided to use the engine code already written by others. However, the research of selecting the suitable Xiangqi engine we could use in the project was not sufficient. The first engine we chose to use could provide some recommendations but in a slow speed as the search depth increase. Moreover, in order to achieve other requirements such as providing the warnings of piece in danger, we need to modify the code of the engine. However, the license of the engine did not allow any modification of it. Therefore, we had to find another engine with more license permission and it took about three weeks for us to find a suitable engine and study how it works for searching and analyze moves. Another experience was that we also spent about one week to study how the bit-board approach worked to search because the research showed the speed of bit-board search method was very fast. However, we finally found out that the bit-board represent was only suitable for international chess so we abandoned the plan of using bit-board representation. Fortunately, we found a suitable engine with license permission and fast search speed and did not cause serious delay of the whole process. From these experiences, we learned the importance of research before implementation and for the future projects, we could prevent the waste of time through carrying out more efficient and effective research.

4.2 Team work and management reflection

Our group consists of six members including group leader Han BAO, secretary Yuan FENG, and other four members: Guanqiao HUI, Meng YUAN, Qiwei SUN and Yu QU. The team leader is responsible for planning the project schedule, allocating tasks and monitoring the work quality of members. The secretary is in charge of writing meeting agendas before every group meeting and meeting minutes after every group meeting. Besides, the secretary also needs to collect and edit the group reports and external

email communications. The task allocation among group members are as follows: Han BAO was responsible for desktop version user interface, Xiangqi engine research and implementation, desktop version software testing. Guanqiao HUI was responsible for Xiangqi rules implementation, desktop version software testing. Meng YUAN was responsible for Xiangqi engine research and implementation, multi-thread implementation. Qiwei SUN was responsible for Xiangqi rules implementation, software evaluation, Android version software testing. Yu QU was responsible for project website build, Android user interface research and implementation, software evaluation. Yuan FENG was responsible for software requirement specification, Xiangqi engine research and implementation, Android user interface research and implementation, software evaluation. Besides, all of the group members contributed to the interim and final group reports writing.

Firstly, as mentioned above, sometimes the task allocation among group members was not very clear and this led to slightly unfair workload among us. It would be improved a lot if we could divide all the unfinished tasks into small sub-tasks and ensure every member is clear about what to do and how to do. Secondly, the project had both group meeting time and individual working time. During the time period of group meeting, we could solve problems together through interaction in time, to contrast, it was difficult to track every member's individual working time in terms of working quality, quantity and methods. For the future projects, this would be improved to great extent if we could checking the daily working process through report the results on the group communication platform where allowed all group members to share resource, documents and code and figure out problems or provide suggestion to other members in time. Thirdly, the time management was also an aspect that we need to improve. For example, the time spending on design and attain permission of the evaluation plan was underestimated and this caused a crush and anxious in the evaluation process. Therefore, the group should predict possible accidents that could happen to delay the process and leave sufficient time when design the time plan for the next time.

References:

‘Advanced chess’ (2016) Available at: https://en.wikipedia.org/wiki/Advanced_Chess (Accessed: 14th April 2016)

Bodlaender, H. and Duniho, F. (2014), “Xiangqi: Chinese Chess” available at: http://www.chessvariants.com/xiangqi.html (accessed: December 12th, 2015)

David Eppstein (1997) *Strategy and board game programming.* Available at: http://www.ics.uci.edu/~eppstein/180a/970408.html (Accessed: 17th April, 2016).

Sqtcadmin (2012) *“V” Model Testing Concept* Available at: https://sqtc.wordpress.com/2012/01/03/v-model-testing-concept/ (Accessed: 28th April, 2016).

Technical Committee of Computer Games and Chinese Association for Artificial Intelligence (2015) *2015 全国计算机博弈大赛冠亚季军获奖明细与统计 (The statistics of computer AI competition in 2015)*. Available at: http://www.caaigames.net/info/news150928.asp (Accessed: 13th April, 2016)

‘Xiangqi’ (2016) Available at: https://en.wikipedia.org/wiki/Xiangqi (Accessed 14th April 2016).

Xiangqi White Paper research group (2014) *《象棋白皮书》关于象棋现状的调查报 1.4.2 象棋软件 (Xiangqi White Paper: the report of Xiangqi status. Chapter 1.4.2 Xiangqi Software)*. Available at: http://www.gdchess.com/bbs/UpLoadFile/2014-7/2014073001315249746.pdf (Accessed: 20th April, 2016).

Xqbase (2004) *Rules and protocols.* Available at: http://www.xqbase.com/protocol.htm (Accessed: 23rd April, 2016).

Zhijie Lee (2012) *The original human vs. computer engine.* Available at: https://github.com/zhijie/ChineseChess4Android (Accessed: 25th April, 2016).

Appendix A: Meeting minute

Record of 1st meeting

Date & time	10:00 a.m. Sep 24th, 2015
Venue	SEB 324
Participants	Han BAO, Guanqiao HUI, Meng YUAN, Qiwei SUN, Yuan FENG, Paul Dempster
Absence	Yu QU(exchange out for the first semester)
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• The general understanding of the project requirement 1. The project is to develop an APP to help Xiangqi player playing Xiangqi games with people in real world. The APP is used to analyze the game and give recommendations. The player is on the main control. 2. The APP should be straight forward and easy to use and react fast. 3. The APP is used offline rather than online. 4. No need to have background music and instruction for new players. • Suggestions from supervisor 1. Start with do research: find out what is needed for the project. 2. Using existing Xiangqi engine will be helpful 3. C or Java could be consider to use for coding but Java will be more suitable 4. Developing a desktop program at first for designing, testing and debugging easily
Action points	**Decisions:** • Weekly formal meeting time: 10:00 a.m. every Thursday at SEB 324 • No formal meeting during the National holiday, the next one will be on Oct 8th, 2015. • Weekly informal meeting time: every Wednesday afternoon **Action points:** • Han BAO: make plan and timeline (by next informal group meeting) • All: do relevant research and discuss and settle down the final plan and timeline (by next informal group meeting) •

Record of 2nd meeting

Date & time	10:00 a.m. Oct 8th, 2015
Venue	SEB 324
Participants	Han BAO, Guanqiao HUI, Meng YUAN, Qiwei SUN, Yuan FENG, Paul Dempster
Absence	Yu QU(exchange out for the first semester)
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• Discuss the project plan - Part A (2015.10.8—2015.11.17): complete simple Chinese chess text-based game using java - Part B (2015.11.17—2015.12.27): finish desktop game with chess engine and user interface design - Part C (TBC): APP design • Comments about the plan - Divided each part into small subtasks and set up daily and weekly deadlines - Leave some extra time for Part B since it is difficult and complex - Do more research about using chess engine, interface and app design - Think about how the GUI, user, chess engine communicate in the game - Work hard on the design of app since it is quite different from desktop game • Comments about group work - Need to have agenda before each meeting - Make tables about what did every member did last week and what will do next week, always keep tracking and monitoring each other
Action points	• Meet at 10:00 a.m. on Saturday(10.10) and work on Part A and research

Record of 3rd meeting

Date & time	10:00 a.m. Oct 15th, 2015
Venue	SEB 324
Participants	Han BAO, Meng YUAN, Qiwei SUN, Yuan FENG, Paul Dempster
Absence	Yu QU, Guanqiao HUI(not on compus)
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• Comments about the requirement specification - add non-functional requirement part, such as allowing user to interrupt during the game - reorganize the interface requirements part • Suggestions about interface design - put some highlight for recommendations and allow use to click it and get further information and explanation - interface should be clear and easy to use - put warnings for the move that may end the game • Comments about the java game design - Need to ensure the chess engine can be used in the program as soon as possible - Since each piece has many possible move and there are many piece on both side, we should consider branch factors i.e. pick the most likely move to analyze and give recommendation. - Other factors to consider: different piece has different strength, the balance strategy for the whole game, at the beginning of the game: many piece, limited move space, at the end: less piece but more move space
Action points	• All: meet at 13:00 on this Saturday(10.17) • Working task for next week: - Han BAO: confirm every can use Git, GUI - Yuan FENG: fix the requirement specification, interim report outline, learn to use Git - Yu QU: design the website as required in the GRP lecture - Yuan MENG, Qiwei SUN, Guanqiao HUI: learn to use Git, GUI

Record of 4rd meeting

Date & time	10:06 a.m. Oct 29th, 2015
Venue	SEB 324
Participants	Han BAO, Meng YUAN, Qiwei SUN, Yuan FENG, Paul Dempster
Absence	Yu QU, Guanqiao HUI(not on compus)
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• Comments on requirement specification - Use words like: 'want to have, would, must, should' to distinguish the requirements. - Fix the format. • Comments on interim report outline - Put sections with similar content together. - Focus more on software engineering. - Use UML diagrams and prototypes to show the design. - Show how these design would be useful. • Reference issue - All the work we submit to school must be our own work. - We could put what is allowed to be refer or modified as the reference of our project. - For the Xiangqi or chess board pictures, search in WIKI firstly and find out which can be used or modified as reference. Or just take pictures by our own. • Problem need to discuss and make decision as a group - The process of developing: develop a desktop game with GUI and transfer it to Android or develop Android game directly.
Action points	• Have meeting at Saturday 1:00 p.m. • Every one need to talk about part of the meeting content

Record of 5rd meeting

Date & time	10:00 a.m. Nov 5th, 2015
Venue	SEB 324
Participants	Han BAO, Guanqiao HUI, Meng YUAN, Yuan FENG, Paul Dempster
Absence	Yu QU, Qiwei SUN(illness)
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• Engine: - License issue: figure out if it is allowed to be used or modified, contact with the people and ask for permission. - Functional issue: apart from recommendations one possible move, the engine should also provide the prediction for the move that the user want to try and know. The engine also need to predict the range of possible moves of opponent side. - Use java based engine • Android interface design: - Android not use swing - Rewrite user interface, different function and class • Test: - Use different kind of Android phones(screen size, storage) to test - Other users could download the application through link or OR code and use it. - Make sure the APP not ask for many permissions • Keep record - Record all the action points and decisions - Record the choice of each points and why we did this • Decisions: - Start interim report the beginning of December - Start transfer to Android from January - Finish the desktop game by the end of November
Action points	Engine research and modify: Meng YUAN, Yuan FENG GUI and rules and how UI communicate with programs: Han BAO, Guanqiao HUI, Qiwei SUN

Record of 6th meeting

Date & time	10:05 a.m. Nov 12th, 2015
Venue	SEB 324
Participants	Han BAO, Yuan FENG, Qiwei SUN, Paul Dempster
Absence	Yu QU, Guanqiao HUI, Meng YUAN
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• Engine: - Use the '.exe.' engine program is not the best choice because it cannot run in all operation systems and computer. - Find java source code for engine which could be used in the program. - Write a engine by ourselves if no suitable engine is found. • Interface: - The click operation should judge whether it need to click a new piece or to move the piece already picked.
Action points	Engine research and modify: Meng YUAN, Yuan FENG GUI and rules: Han BAO, Guanqiao HUI, Qiwei SUN

Record of 7th meeting

Date & time	10:10 a.m. Nov 19th, 2015
Venue	SEB 324
Participants	Han BAO, Yuan FENG, Guanqiao HUI, Meng YUAN Qiwei SUN, Paul Dempster
Absence	Yu QU
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• Android interface design - The size of Android phone screen is much smaller compared with computer's, the space should be used efficiently to show the most useful information • Search depth and breadth - Need to search both from depth first and breadth first aspects, and show the most useful predictions to the user - Provide top10 moves for both user and opponent - Combine different search methods properly • Running speed - Try to control search time within 1 minute - Try to abandon branches quickly • Game phases - Use different approaches in different game phases - Beginning: have popular pre-calculation, no need to search and save time - Middle: need to search - End: less piece, less possible move • Explain agenda - Every one should involved in and present themselves in the next meeting
Action points	Engine modify: Meng YUAN, Yuan FENG GUI and rules: Han BAO, Guanqiao HUI, Qiwei SUN Combine methods together: All

Record of 8th meeting

Date & time	10:00 a.m. Nov 26th, 2015
Venue	SEB 324
Participants	Han BAO, Yuan FENG, Guanqiao HUI, Qiwei SUN, Paul Dempster
Absence	Yu QU, Meng YUAN
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - UI design: Han BAO - improve the engine function: Meng YUAN, Yuan FENG • **The interface design** - Functions already achieved: click, move piece, eat piece, new game, check 'win' and 'lose'. - Functions need to add: highlight, warning, undo, history move, show text • **The engine issue** - Let the engine keep generate moves as many as possible within the required time. - Check if the emperor is in danger - Add the analysis of the consequence of the moves - Set function (maybe return a boolean) to check if there still required time remained. - Set the engine keeping working out the results and store the results in data structure(public variables), let interface to access it. • **Overall program** - The program should be multi-threading - Ideal running process: the interface tells the engine about the states of the chess board and the engine returns what user what.
Action points	Next week plan: Interim report outline and task distribution: Yuan FENG Engine modify: Meng YUAN, Yuan FENG GUI: Han BAO Combine methods together: Guanqiao HUI, Qiwei SUN Start interim report: All

Record of 9th meeting

Date & time	10:05 a.m. Dec 3rd, 2015
Venue	SEB 324
Participants	Han BAO, Yuan FENG, Qiwei SUN, Paul Dempster
Absence	Yu QU, Meng YUAN, Guanqiao HUI,
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - UI design: Han BAO - Interim report outline: Yuan FENG - Combine: Guanqiao HUI - Prototype: Qiwei SUN - Engine debug: Yuan MENG • **Interim report outline** - Combine the problems part with the part that describe what has been done together sorted by the topics. - Add more analysis of the prototype, how it will fit different kinds of the android devices. - Add the possible risks in the next stage and what the group will do. • **Prototype** - Think of the features that is really necessary and spend time on the main functions.
Action points	**Interim report DDL: 12.7,11p.m.** **Guanqiao HUI (700)** 1.1 Introduction about the project (200) 1.2 Information of Xiangqi and the existing Xiangqi game (300) 6.2 Management problems (200) **Meng YUAN (600)** 1.3.1 Research about xiangqi engine (200) 4.3.1. The structure of the engine code, how it achieve the functions? (100) 4.3.2 The description of chess engine, how the engine communicate with interface (200) 6.1.2 Xiangqi engine (100) **Han BAO (800)** 1.3.2 Android app development and GUI (200) 4.2.1 The structure of the user interface code, apply what what knowledge to achieve the functions (100) 4.2.2 The logical thinking of UI coding(200) 6.1.3 UI design (100) Part 8: Time plan(100-200) **Yuan FENG (1200)** Requirement specification(500) 2.1 Scope, project identification 2.2 Functional and non-functional requirements Key decisions and why (600) 5.1 Development steps 5.2 The programming languages and other tools 5.3 operating systems, computers, and any additional software and hardware to be used, 5.4 reasons for those decisions. 6.1.1 programming languages, and others(100) **Qiwei SUN (400)+prototype** Part 3: Initial design and user interface (300) 3.1 mid- fidelity prototype of the game interface of android 3.2 description of the design 6.1.4 Threads (100) **Yu QU (600)** 4.1 Website function description, content description (300) Part 7: possible risks in the next stages (300) 7.1 Debugging and testing2 Transform from desktop to android app: redesign UI, coding

Record of 10th meeting

Date & time	10:20 a.m. Dec 10th, 2015
Venue	SEB 324
Participants	Han BAO, Yuan FENG, Meng YUAN, Paul Dempster
Absence	Yu QU, Guanqiao HUI, Qiwei SUN,
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - Finished the draft interim report. • **Need to improve in the improve in the report** - The format: space, font, picture size, capital letter, consistency - Change the title name of each part, have a number for picture and chart - Content: write higher level content, i.e. how and why choose the development methods - Consistency: the terms should be consistent in the report, e.g. Advanced Xiangqi, application, etc - Choose proper words to express the meaning • **Plan for next week** - Finish the interim report
Action points	**Fixed the report during the weekend 12.12—12.13** **Have a meeting next Monday or Tuesday**

Record of 11th meeting

Date & time	14:00 22nd Feb 2016
Venue	SEB 438
Participants	Han BAO, Yuan FENG, Yu QU, Guanqiao HUI, Qiwei SUN, Paul Dempster
Absence	Meng YUAN
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - Studied how to develop Android UI - Studied the engine • **The Android version** - Better to have the Android 4 or 5 edition in order to suitable for more phones • **The engine design** - Consider split a new chess board just analyze the small area that is most needed to be concerned • **Developing and testing methods** - Use real game play to test the efficiency and effect of engine • **Plan for next week** - Improve engine search approach - Work on the android UI design - Work out the program that can play a real game -
Action points	**Have formal meeting on every Monday 14:00 SEB 438** **Have group meeting every Thursday and Friday** **Work on Android UI: Yuan FENG, Yu QU, Qiwei SUN, Guanqiao HUI** **Work on Xiangqi engine: Han BAO, Meng YUAN**

Record of 12th meeting

Date & time	14:00 29nd Feb 2016
Venue	SEB 438
Participants	Han BAO, Yuan FENG, Yu QU, Qiwei SUN, Paul Dempster
Absence	Meng YUAN，Guanqiao HUI
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - Studied how to develop Android UI - Achieved button could be click and move to another activity - Changed Android version to 4.3.1 • **Content this week** • **Android development** - Learn the knowledge about the 'view' and 'activity' - Work out a chessboard that can click and move chess piece • **Task plan** - Focus on the tasks that are most important to the project - Rethink all the tasks need to do and rank them from high importance to low importance - Need to accelerate the pace - Allocate the workload in an efficient way • **Plan for next week** - Improve engine search approach - Work on the android UI that could show the chess board and click the piece - Work out the program that can play a real game
Action points	**Have formal meeting on every Monday 14:00 SEB 438** **Have group meeting every Monday and Thursday** **Work on Android UI: Yuan FENG, Yu QU, Qiwei SUN** **Work on Xiangqi engine: Han BAO, Meng YUAN, Guanqiao HUI**

Record of 13th meeting

Date & time	14:00 7th Mar 2016
Venue	SEB 438
Participants	Han BAO, Yuan FENG, Yu QU, Qiwei SUN, Meng YUAN，Paul Dempster
Absence	Guanqiao HUI
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - Chess engine: finish the initialization of the board - Android UI: finish the layout of chess board, move piece • **Content this week** • **Chess engine redesign** - Need to accelerate the pace of design the algorithms - Use Bit board to represent the chess board - Each kind of piece has a individual chess board - The engine is for both desktop version and Android version • **Android UI** - Improve the UI: let the piece to be put on proper position automatically - Need to have a method to tracking the piece move • **Plan for next week** - Chess engine:Finish move piece part(If it finish earlier, start working on the search algorithm) - Android UI: Finish move piece rule part, show the text for tips, add some functional buttons
Action points	**Have formal meeting on every Monday 14:00 SEB 438** **Have group meeting every Monday and Thursday** **Work on Android UI: Yuan FENG, Yu QU, Qiwei SUN** **Work on Xiangqi engine: Han BAO, Meng YUAN, Guanqiao HUI**

Record of 14th meeting

Date & time	14:00 14th Mar 2016
Venue	SEB 438
Participants	Han BAO, Yuan FENG, Yu QU, Qiwei SUN, Guanqiao HUI, Paul Dempster
Absence	Meng YUAN
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - Chess engine:Finish the move chess part and could search steps. - Android UI: Find a another method to set the location of piece, finish the rule part. • **Content this week** • **Chess engine** - The license is valid i.e the code is allowed to use and modify - Need to consider the search function in both breadth and depth search - Need to study the algorithms used in the engine(How and why they are used) • **Evaluation the engine** - Design an efficient and effective way to evaluate the chess engine for example, compare with other engines. - Testing and debugging the program(consider the user experience) • **Plan for next week** - Chess engine: improve the search function to meet the requirements - Android UI:Debugging the rule part • Try to combine the two parts together(desktop version(UI and engine and Android version)
Action points	**Have formal meeting on every Monday 14:00 SEB 438** **Have group meeting every Monday and Thursday** **Work on Android UI: Yuan FENG, Yu QU, Qiwei SUN** **Work on Xiangqi engine: Han BAO, Meng YUAN, Guanqiao HUI**

Record of 15th meeting

Date & time	14:06 21th Mar 2016
Venue	SEB 438
Participants	Han BAO, Yuan FENG, Yu QU, Qiwei SUN, Guanqiao HUI, Meng YUAN, Paul Dempster
Absence	N/A
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - Chess engine: Improve the breath search. - Android UI: Debug the rule part, restructure the program, improve button design(more user friendly). - Other: finish combine the desktop version UI with the chess engine. • **Content this week** • **Chess engine** - Need to analyze and show the warning that the piece is potentially being captured or the user can capture the opponent piece in the following steps. - Need to have a document to explain how the engine works. • **Android UI** - Need to find a method to reset the location of the piece move by mistake and the piece moved by user but not in a precise location. - Need to solve the problem some rules about moving pieces did not work. - Need to find a method to get the location of the piece. • **Plan for next week** • The engine: - Add functions to provide more reminders and suggestions. - Hand in a paper that explain how the engine works to supervisor. • Android UI: - Debug the rule part and enable to capture piece, Add other functional buttons and text. • Other: Try to combine the Android UI part and the chess engine part.
Action points	**Have formal meeting on every Monday 14:00 SEB 438** **Have group meeting every Monday and Thursday** **Work on Android UI: Yuan FENG, Yu QU, Qiwei SUN** **Work on Xiangqi engine: Han BAO, Meng YUAN, Guanqiao HUI**

Record of 16th meeting

Date & time	14:05 28th Mar 2016
Venue	SEB 438
Participants	Han BAO, Yuan FENG, Yu QU, Qiwei SUN, Meng YUAN, Paul Dempster
Absence	Guanqiao HUI(illness)
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - **The Android UI:** Finish debug the rule part (including all the move piece rule and judge win or lose) Can put the piece in right place precisely The piece can be put back when it move invalidly Refactor the code - **The chess engine:** Can work out five different suggestions Can show the warnings of piece is going to be captured - **Finish the outline of the final report** • **Content this week** • **Chess engine** - Need to improve search function: keep searching until the user make the next move. - Use two thread for engine and ui respectively. • **Android UI** - Need to find a way to suit the android mobile screens • **Others** - Adjust the outline of the final report - Reschedule meeting: at 4:00p.m on 5th, April • **Plan for next week** • The engine: Improve the search funstion • Android UI: Try to suit for mobile phone screens • Other: Adjust the report outline, change the program to two threads
Action points	**Have group meeting every Monday and Thursday** **Work on Android UI: Yuan FENG, Yu QU, Qiwei SUN** **Work on Xiangqi engine: Han BAO, Meng YUAN, Guanqiao HUI** **Final report: All**

Record of 17th meeting

Date & time	16:01 5th Apr 2016
Venue	SEB 438
Participants	Han BAO, Yuan FENG, Yu QU, Qiwei SUN, Meng YUAN, Guanqiao HUI Paul Dempster
Absence	N/A
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - **The Android UI:** It can show properly on the mobile devices - **The chess engine:** Change the depth into 6 Finish the double-thread part • **Content this week** • **Final report** - The timeline should be included in the appendix and compare the plan with what really happen in the project - The report should introduce what changes happen compared with plan and why the group decided to make the changes. - The deadline of the draft report will be 14th Apr - The individual report should include reflection and what learned in the project • **Chess engine:** need to check the situation that the game will end within one step • **Android UI:** Combine with the engine and make sure it work well • **Test and Evaluation:** write a document about what need to do in the test and evaluation process and get the permission from supervisor. • **Plan for next week** • The engine: improve the searching • Android UI: combine and test • Other: Start writing the final report
Action points	**Have group meeting every Monday and Thursday** **Work on Android UI: Yuan FENG, Yu QU** **Work on Xiangqi engine: Han BAO, Meng YUAN, Guanqiao HUI** **Final report: All** **Test and evaluation: Qiwei SUN**

Record of 18th meeting

Date & time	14:04 11th Apr 2016
Venue	SEB 438
Participants	Han BAO, Yuan FENG, Yu QU, Meng YUAN, Guanqiao HUI Paul Dempster
Absence	Qiwei SUN
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - **The Android UI:** Finish the undo, move, capture, judge win and lose part. - **Start writing the final report** - **Design the process of evaluation the engine** • **Content this week** • **Final report:** the content should be clear and helpful for readers to understand what has happened in the project(i.e. the goal, the design, the thinking, the problems, the tests, the evaluation) Not only focus on the limit of the word amount but try to make everything understandable and meaningful. It is really important to refer to the feedback from the interim report. • **Evaluation:** the users in the evaluation must be the ones that know how to play Xiangqi and the age, gender, occupation are not very important in this evaluation. The goal of it should be set(i.e. it is about evaluate the UI design or the engine or something else?). Try to avoid any ethic problems would happen during the experiment. Prepare for documents needed and get approval ASAP • **Android UI:** there are some problems with the thread and the interaction with the android UI and engine, try to use breakpoint approaches to debug and find solutions. • **Plan for next week** • Finish the draft final report. • Finish the relevant evaluation document and send to supervisor for approve. • Fix the problems in combining the android UI with engine.
Action points	**Have group meeting every Monday and Thursday** **Work on Android UI: Yuan FENG, Yu QU, Qiwei SUN** **Work on Xiangqi engine: Han BAO, Meng YUAN, Guanqiao HUI** **Final report: All** **Test and evaluation: Yuan FENG, Qiwei SUN**

Record of 19th meeting

Date & time	16:02 18th Apr 2016
Venue	SEB 438
Participants	Han BAO, Yuan FENG, Yu QU, Meng YUAN, Qiwei SUN, Guanqiao HUI Paul Dempster
Absence	N/A
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - Evaluation: finished the evaluation documents including the evaluation design, questionnaire, ethical issue state. - Android version: finished the combine with engine, still working on fixing some bugs and improvement. - Final report: finished the draft report • **Content this week** - **Final report** **Introduction chapter** Need to include more information about the project **Engine chapter** Conversion btw the UI and engine, use the different representation Academic and word use How much from engine and how much we modify, maybe calculate the percentage Class diagram Be clear about the useful details **Android UI chapter** Put screen shots of it in the report Need to mention the process of finding what is not good and how improve it **Test chapter** Acceptance testing and evaluation is not the same topic Acceptance testing is not need in this project, not the focus and aim of the group project **Reflection chapter** Explain why make decision about engine and write some reflections on good things **Timeline chapter** Compare original one with the real-happen one • **Plan for next week** • Finish the final report except for some test and evaluation part • Fix the problems in combining the android UI with engine. • Start evaluation when obtain the approval
Action points	**Have group meeting every Monday and Thursday** **Final report: All** **Test and Evaluation: Yuan FENG, Qiwei SUN, Yu QU** **Debug and Improve: All**

Record of 20th meeting

Date & time	2:03 25th Apr 2016
Venue	SEB 438
Participants	Han BAO, Yuan FENG, Yu QU, Meng YUAN, Qiwei SUN, Guanqiao HUI Paul Dempster
Absence	N/A
Chairperson	Han BAO
Secretary	Yuan FENG
Content of meeting	• **Review last week** - Evaluation: finished the evaluation of desktop version. - Android version: finished the combine with engine, the highlight and fixed bugs. - Desktop version: debug and improve. • **Content this week** - The open day should show how the software work and what we have done during the one year project and answer questions - The presentation should include the introduction of the project, what we learned, how the software work and the evaluation - Pay attention to the consistence of the final report • **Plan for next week** • Finish the final report • Prepare for the open day poster and presentation • Have a quick meeting next Monday 2:00pm
Action points	**Have group meeting every Monday and Thursday** **Final report: All** **Presentation prepare: Yu QU, Yuan FENG**

Appendix B: Test case

Rule:
The represent of chess board are as follows: the coordinate of X increases from 0 to 8 from left to right and the coordinate of Y increases from 0 to 9 from top to bottom.
Four methods to test:

isOpponent

isChessExist

isValidMove

isSecondValid

Test Case	Outcome
Rule.isOpponent(btn, 2)	False
Rule.isOpponent(btn, 4)	True
Rule.isOpponent(btn, 6)	False
Rule.isOpponent(btn, 8)	True
Rule.isChessExist(0, 1)	False
Rule.isChessExist(2, 2)	False
Rule.isChessExist(0, 0)	True
Rule.isChessExist(1, 2)	True
Rule.isValidMove(0, 0, 0, 1)	True
Rule.isValidMove(0, 0, 1, 1)	False
Rule.isValidMove(1, 0, 2, 2)	True
Rule.isValidMove(1, 0, 3, 2)	False
Rule.isValidMove(0, 9, 0, 8)	True
Rule.isValidMove(1, 9, 3, 8)	False
Rule.isValidMove(2, 9, 4, 7)	True
Rule.isValidMove(0, 6, 1, 6)	False
Rule.isSecondValid(0, 1, 0, 2, 8, red, black)	False

Rule.isSecondValid(1, 2, 1, 9, 8, red, black)	True
Rule.isSecondValid(7, 9, 5, 8, 4, red, black)	False
Rule.isSecondValid(7, 9, 6, 7, 4, red, black)	True

NewGame:

Three methods to test:

move

beingCaptured

undo

Test case	**Outcome**
newGame.move(0, -1, 0, 9), rule.isChessExist(0, 9) rule.isChessExist(0, 8)	 False True
newGame.move(-1, 2, 7, 0), rule.isChessExist(7, 0) rule.isChessExist(6, 2)	 False True
newGame.beingCaptured(0, 7, 1, 7, chessboard) rule.isChessExist(1, 7) rule.isChessExist(1, 0)	 False True
newGame.beingCaptured(0, -7, 7, 2, chessboard) rule.isChessExist(7, 2) rule.isChessExist(7, 9)	 False True
newGame.move(0, -1, 0, 9) newGame.undo(0, 9, 0, 8, false) rule.isChessExist(0, 8) rule.isChessExist(0, 9)	 False True
newGame.move(0, 1, 8, 0) newGame.undo(8, 0, 8, 1, false) rule.isChessExist(8, 1) rule.isChessExist(8, 0)	 False True
newGame.move(0, 7, 1, 7) newGame.undo(1, 7, 1, 0, true) rule.isChessExist(1, 7) rule.isChessExist(1, 0)	 True True
newGame.move(0, -7, 7, 2) newGame.undo(7, 2, 7, 9, true) rule.isChessExist(7, 2) rule.isChessExist(7, 9)	 True True

Appendix C: User Manual Desktop Version

1.1 Foreword
The user manual is made for helping Chinese chess players to during the Xiangqi game.

1.2 Term Definition

Undo: retract a false move in a chess game

2. Software Overview

2.1 Aim of the software

Generally, in the chess game, the player is the master of the situation. But some defects such as, careless moves, and inequity pieces exchange. Computer will not make this human mistake and could be combined with the player to achieve a better move.

2.2 Function

Use Alpha-Beta method and search through the whole chess board.

2.3 Performance

The searching engine would take different time in different search steps.

Four steps — 1 seconds
five steps — 15 seconds
six steps — 1mins
seven steps — 2mins

Two extra factors may affect the result:
The first is the performance of the computer
The second is that as more and more steps moved, the searching condition would be more complex and need more than standard time unit.

3. Use Instructions

3.1 In general
If your or your competitor’s chess pieces move, you may move the piece in the software interface and make it same to the physical chessboard condition. And then search the move which would be benefit for you.

On the other hand, you may simulate the next several moves on the interface and use ‘undo’ button to restore the previous situation.

3.2 Input

Click on the certain chess pieces and move the pieces to the object position.

3.3 Output

d. Updated suggestions: eg. 2J3H

(The chess pieces are located through x- and y- values. X- values are defined by 1~9 and y- values are defined by A~J)
So 2J3H means the piece which located in 2J moves to position 3H.
The more of depth value increase, the more reliable result would be gained.

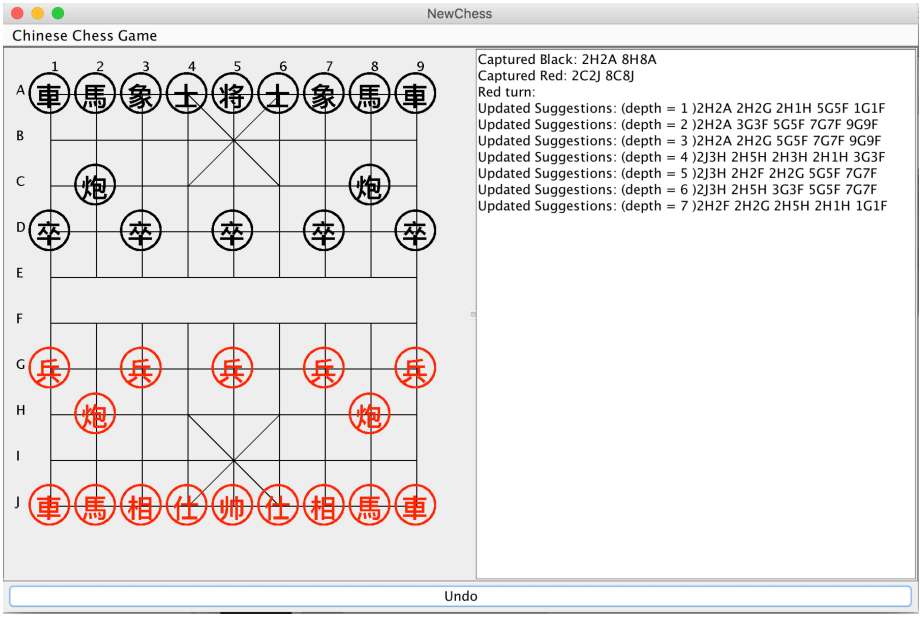
b. Capture red: eg. 2C2J

2C is the location of a black piece, 2J is of a red piece.
This means the red should prevent the 2J piece being captured by the 2C piece in black.

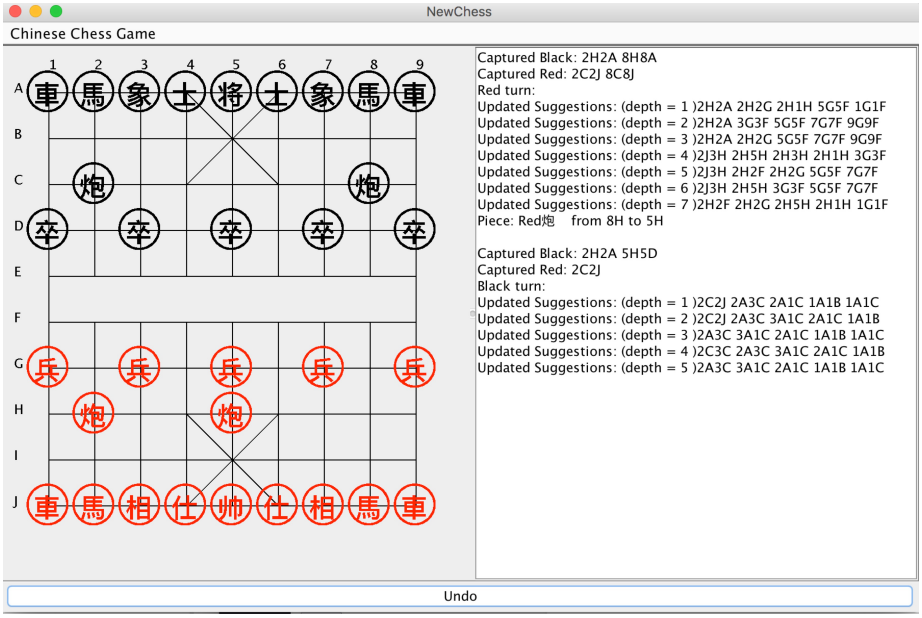
c. Capture black: eg. 2H2A

2H is the location of a red piece, 2A is of a black piece.
This means the black should prevent the 2A piece being captured by the 2H piece in red.

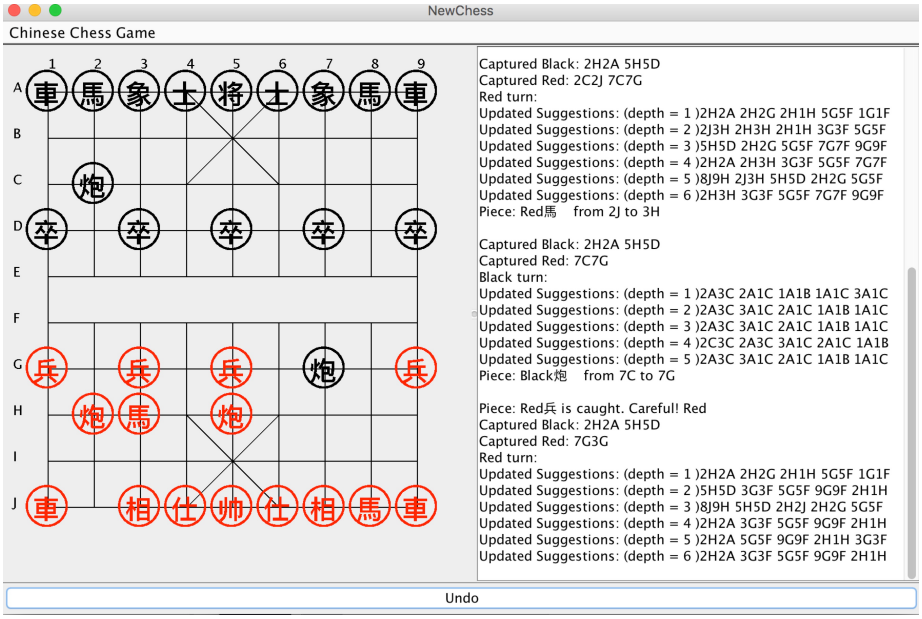
4. Instructions
a. Initial State



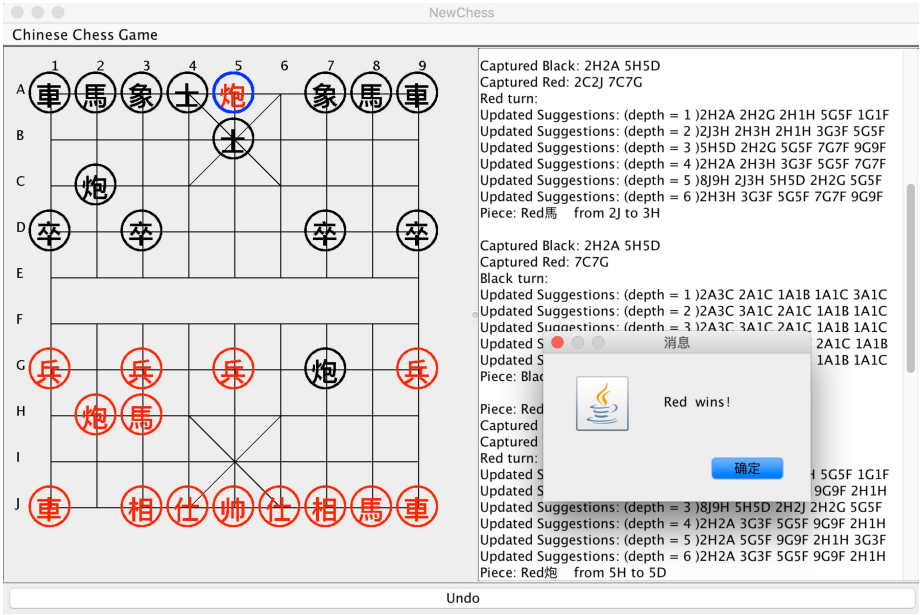
b. State after the first move



c. State of capture a piece



d. State of win



Appendix D: User Manual for Android Version

1. Foreword

1.1 Purpose

The user manual is made for helping Chinese chess players to during the Xiangqi game.

1.2 Term Definition

Undo: retract a false move in a chess game

2. Software Overview

2.1 Aim of the software

Generally, in the chess game, the player is the master of the situation. But some defects such as, careless moves, and inequity pieces exchange. Computer will not make this human mistake and could be combined with the player to achieve a better move.

2.2 Function

Use Alpha-Beta method and search through the whole chess board.

3. Use Instructions

3.1 In general
If your or your competitor's chess pieces move, you may move the piece in the software interface and make it same to the physical chessboard condition. And then search the move which would be benefit for you.
On the other hand, you may simulate the next several moves on the interface and use 'undo' button to restore the previous situation.

3.2 Input

Click on the certain chess pieces and move the pieces to the object position.

3.3 Output

a. Suggestions: eg. Move Red 炮 from (1, 7) to (1, 0)

(The chess pieces are located through x- and y- values. x- values are defined by 0~7 from the left to the right side, and y- values are defined by 0~8 for the top to the bottom) So ‘Move Red 炮 from (1, 7) to (1, 0)’ means the piece which located in (1, 7) moves to position (1, 0).
The more of depth value increase, the more reliable result would be gained.

4. Instructions and Tips

Figure 4.1 Welcome interface

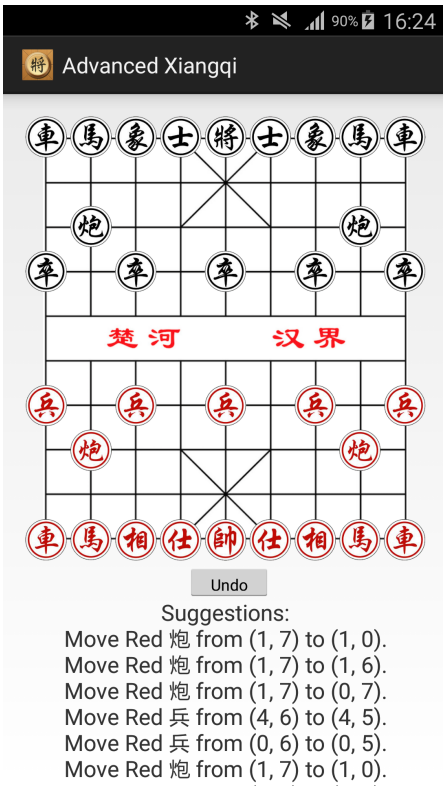
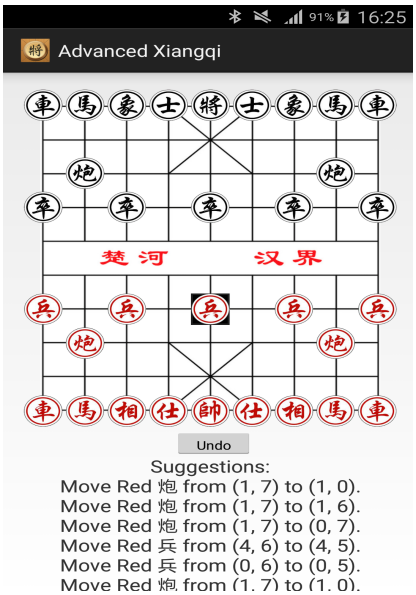


Figure 4.2 New game



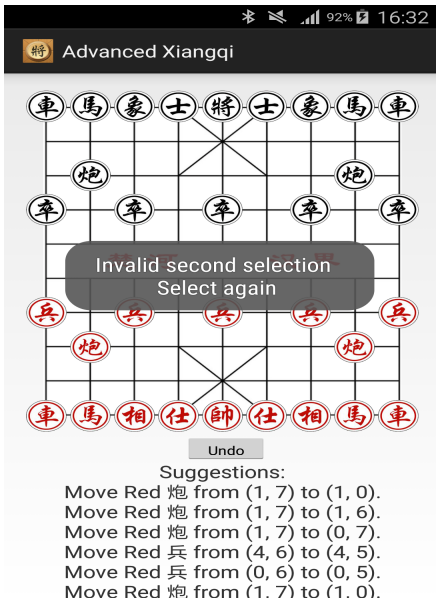


Figure 4.3 Click on the piece

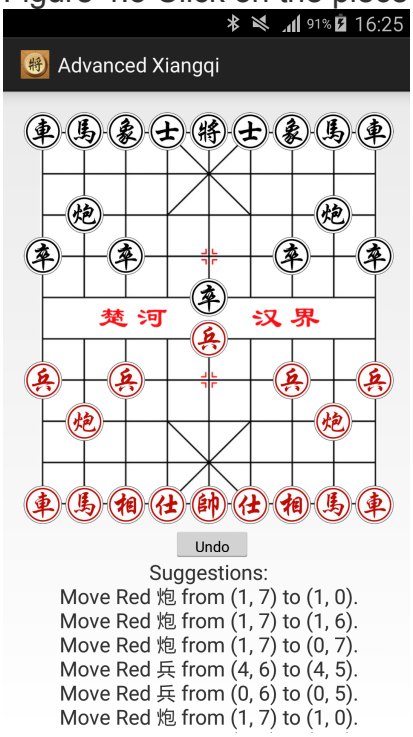


Figure 4.4 Invalid second selection

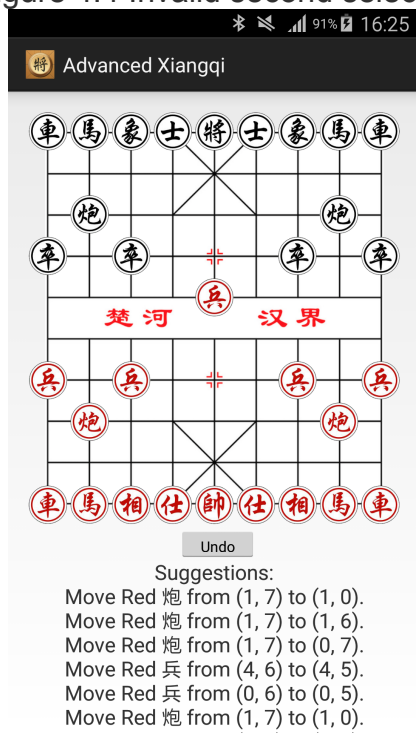


Figure 4.5 Status before capturing a piece piece

Figure 4.6 Status after capturing a

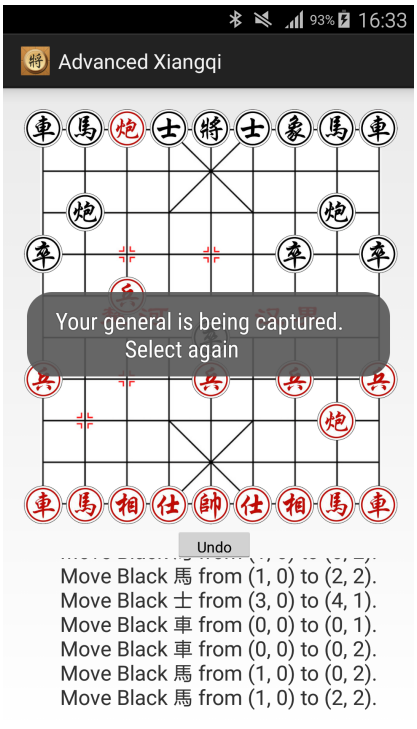


Figure 4.7 Captured general

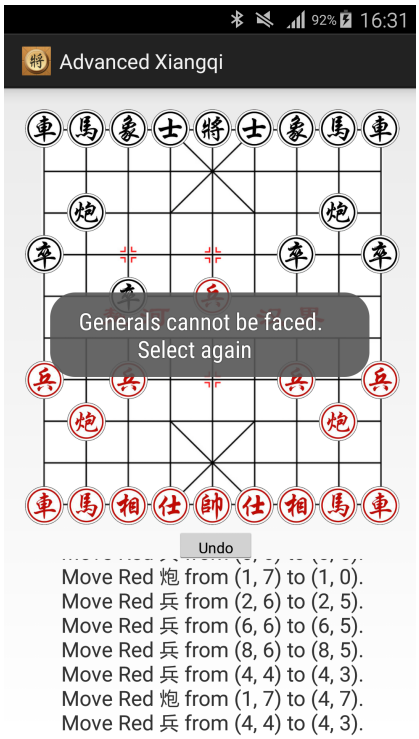


Figure 4.8 General face to face

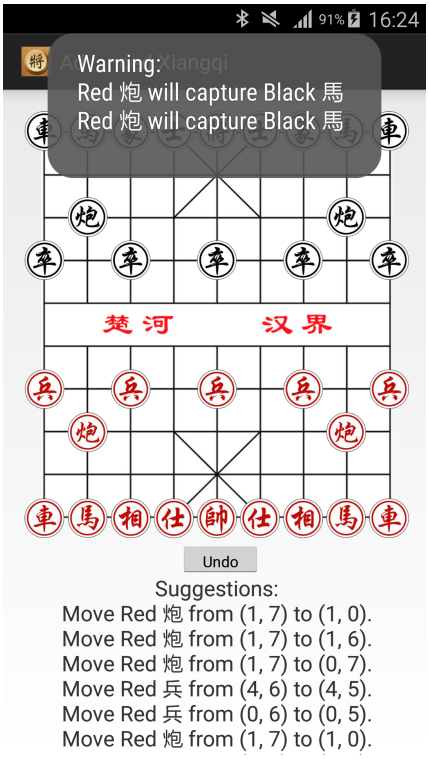


Figure 4.9 Warning of captures

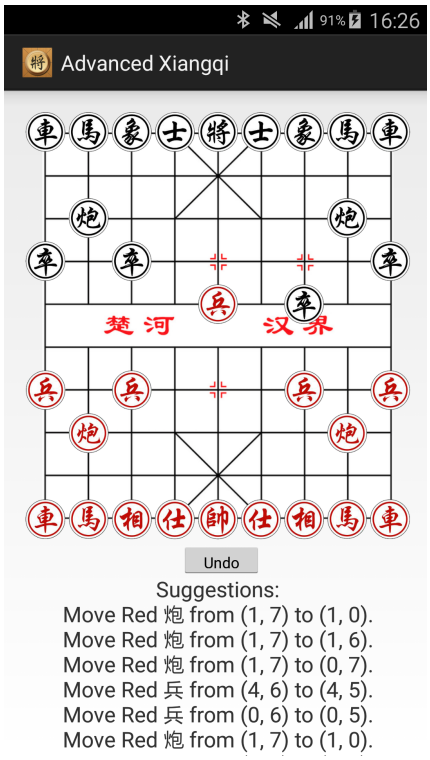


Figure 4.10 Before undo

4.11 After undo

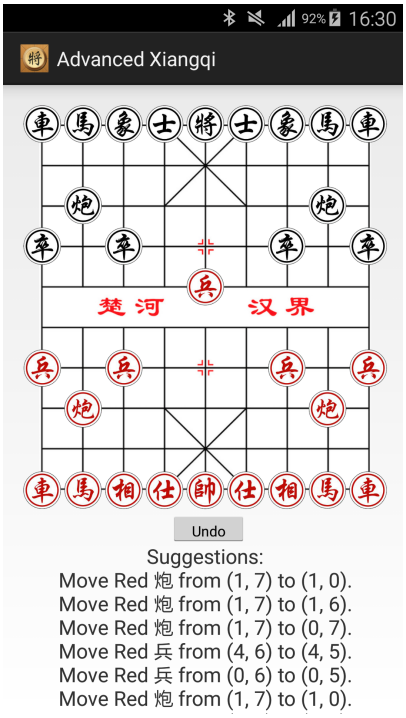


Figure 4.11 undo

Appendix E: Timeline

2015.9.18	Timeline first made	Finished
2016.9.25	After first meeting with supervisor, timeline was changed	Finished
2015.9.28	Check the timeline	Finished
2015.10.5	Check the needed equipments	Finished
2015.10.9	Start work as planning -- research	Finished
2015.10.30	Start Desktop version	Finished
2015.11.30	Finish Desktop edition UI and engine	Finished
2015.12.1	Start Interim report	Finished
2015.12.7	Finish Interim report draft	Finished
2015.12.10	Meet with supervisor about the interim report	Finished
2015.12.13	Improve interim report	Finished
2015.12.14	Finalization	Finished
2015.12.18	Submit interim report	Finished
2016.1.20	Start android edition (learn android language by ourselves)	Finished
2016.2.27	Finish android edition UI part	Delay（Finished on March 30）
2016.3.3	Finish android edition movement	Delay（Finished on March 30）
2016.3.10	Finish android edition engine	Finished
2016.3.17	Finish android edition part	Delay（Finished on April 15）
2016.3.28	Start testing	Delay
2016.4.20	Work for presentation and improve GRP final report	Finished
2016.4.29	Final GRP work and report deadline	
2016.5.1	Final presentation practice	

2016.5.4	Open day and presentation	