Using C++ to Realize Chinese Chess Double Player's Battle

Yuheng Jia

10300221

Department of Physics and Astronomy

The University of Manchester

PHYS 30672 OOP in C++ final project report

May 2021

Abstract

In this project, C++ programming language is used to create a program that allows two players to play Chinese Chess (Xiangqi). The code is divided into three header files and two source files. Each player has sixteen movable pieces, which are divided into seven basic pieces and each has its own moving rules. The chess pieces are organized by inheritance and polymorphism. The aim of the project is to allow two users to play Chinese Chess by moving one piece at a time, and decide whether to win or lose after a specific basic piece is destroyed.

1. Introduction

Board games have always been the first choice for people to spend their leisure time due to its diversity, playfulness, and clear and simple rules. Chinese Chess (pinyin: Xiangqi) is one of the most popular board games in China [1]. It describes a scenario of confrontation between two armies across the river. Each player leads an army, and each army has five soldiers, two cannons, two elephants, two horses, two chariots, two advisors and one general. The distribution of pieces of chess are symmetrical about the river in between. Each player is allowed to move one piece of chess in each round. A player wins if the general of another player is eliminated, and the game ends when a winner appears.

In this report, the rules behind the game will be discussed in section 2. In section 3, the code design will be discussed. The results will be shown in section 4, and in section 5 some further improvements are discussed.

2. Theory

2.1 The chessboard

The chessboard has size 10*9 as shown in Figure 1. Below the river is player 2 and above the river is player 1. The chess distribution above and below the river is the same. Take player 2 as an example, the nearest five pieces to the river are all Soldiers. Below the Solider line are two Cannons. At the bottom line, from the left to the right are Chariot, Horse, Elephant, Advisor, General, Advisor, Elephant, Horse, and Chariot. Therefore, there are two symmetrical lines, one is the river dividing two armies, another is the line across two generals of each army.

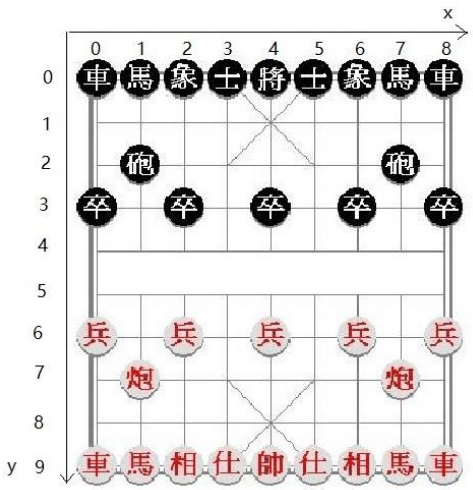

Figure 1: Schematic diagram of the chessboard for Chinese Chess, consisting of two armies, one above and one below the river. The vertical axis is labelled as y-axis, starts from 0 and increases downwards. The horizontal axis is labelled as x-axis, starts from 0 and increases towards right.

2.2 Chess rules

Each movement of piece of chess on the chess board is to destroy the enemy's defence and eliminate as more enemy's pieces of chess as possible. The ultimate goal is to remove the other player's General, in which case the player whose General is still alive wins. All chess rule are explained in terms of the player 2's pieces of chess for simplicity. The same rules are also applied to the player 1.

General. General is the only chess piece that each player has only one. The elimination of this piece means the end of the game. General is allowed to move along the line, either horizontally or vertically, in the region called "palace", which is 2*2 square region formed by $x \in [3,5]$ and $y \in [7,9]$. It can only move one step each time. If the destination the General moves to is occupied by a piece of chess, the General will eliminate that piece if it belongs to the enemy.

Advisor. There are two Advisors at each side of the General. The Advisors can only move within the palace, and it can only moves along the declining path. It can move $\sqrt{2}$ step each time. Usually they play the role of the General's bodyguards.

Elephant. In ancient wall, elephants were used to transport goods. Thus, it has greater movable distance. It is able to move from a point A to a point B $2\sqrt{2}$ steps away if there is no other piece of chess on its way. For example, if the Elephant starts from the position (2,9) as shown in Figure 1, then it is able to move to (0,7) or (4,7) if there is no piece of chess at either (1,8) or (3,8). However, Elephant is not able to move across the river to reach the other side of the board.

Horse. Horses are not as heavy as elephants, thus it has ability to move across the river. Each time the Horse is able to move $\sqrt{5}$ step if there is no obstacle on its path. For example, if the initial position of the Horse is at (4,6), then it is able to move to (3,4) or (5,4) if no piece of chess at (4,5), to (6,5) or (6,7) if no piece at (5,6), to (5,8) or (3,8) if no piece at (4,7), to (2,5) or (2,7) if no piece at (3,6).

Chariot. A Chariot can move either horizontally or vertically, and there is no limitation on the distance moved as long as there is no other pieces of chess along its path.

Cannon. The rule of movement for Cannon is different from other pieces of chess. A Cannon can move like a Chariot if there is no piece of chess at the destination position. However, the basic condition for a Cannon to attack another piece of chess is that there must be exactly one piece of chess between the Cannon's position and the destination.

Soldier. Soldier is the most common piece of chess on the board. Before crossing the river, it can only move directly towards enemy. When moving to the enemy's side, it can move in all directions except backwards, in a step of size one.

3. Code design

The codes are divided into five files in total. Three header files are `c_boardClass.h`, `ChessClass.h`, and `ChessboardClass.h` respectively. The headers, class definition, and function declaration of derived classes are stored in these header files. One of the source file, `ChineseChess.cpp`, stores the function definitions of derived classes' member functions, and another source file `MainChineseChess.cpp` stores the main code and an output function `battle_demo()`. There are two main difficulties, one is to coordinate the relationship of each chess piece on the chess board and display and update the chessboard, another is to realize the movement of each piece of chess by code. These two problems are resolved by using the properties of inheritance and polymorphism of C++.

The structure of the code is in a hierarchy of classes. There are three main classes have been used. The first one is `c_board` class, which is used to create the chessboard by inserting characters in a 2D array. Its class and member function definition are stored in `c_boardClass.h` header file. One of the main functions it contains is to check if a given position on the board is occupied by a derived chess or an `Empty` object, this is a Boolean function called `occupation()`. Another is to replace the object at a given position by an `Empty` object. This function is of type void, called `choices_clean()`. The main responsibility of this class is to create elements on a chessboard and store them in a 2D array of char type.

The second class is an abstract base class called `Chess`, which contains only pure virtual functions. The functions in `Chess` are inherited to two abstract derived classes, `Player1` and `Player2`, which divides the pieces of chess into two groups, and one derived class `Empty`, which represents the empty position on the chess board. The functions in classes `Player1` and `Player2` are then inherited to the derived classes of specific pieces of chess in each camp. There are two main functions in each derived chess class, one is `obstacle()`, which is of Boolean type and used to check if there are obstacles on the path of a piece of chess from its initial position to its destination. For chess pieces with more complicated moves, `obstacle()` is used together with `occupation()`, and for those chess pieces with simpler moves, such as Advisor and Soldier, the `obstacle()` always returns false, means no obstacle towards the destination. Another function is `move_decision()`, this Boolean function is used to determine whether or not the destination is occupiable. In each derived class, there are three parameters in parameterized constructor: the initial x value, the initial y value, and a pointer points to a `c_board` type object. The first two parameters are stored as public since they need to be changeable in order to update the chess piece position. The last parameter is a pointer and is stored as private, since the chess pieces are played on an unique chessboard. Another private data stored in each derived is `squared_distance`, this is an integer and it helps to locate the exact position the chess piece may travel. The class definition and member function declaration

are stored in `ChessClass.h` header file, while the member function definition is stored in `ChessClass.cpp` file.

The last class is called `ChessBoard`. This class is used to put pieces of chess on the chessboard. It includes `ChessClass.h` and has more complicated functionality. In this class, a pointer array, `Board`, is created and stored in private section. Each pointer points to the address of a chess piece on the chessboard. The class's constructor calls a function `Init()` which initialises each chess piece on the board, and their addresses are stored in the pointer array `Board`. Within the function `Init()`, another function `ChessCreated()` is called to replace the empty chess on the board by the correct chess piece. `printChessBoard()` is a function calls `c_board`'s member function `cbprint()` to display the chessboard. `MoveChess()` and `board_mdecision()` are other two important functions. The first one is used to process the movement assigned by one of the player and returns the name of the chess piece at the destination. In this function, `switch…case` statement is used to avoid using `if…else` statement when there are multiple options. The second one is an upgraded `occupation()` function, which decides if the chess piece, if there is one, at the destination is occupiable. The logic of this function is to compare if the chess pieces at the initial and final position are of the same team, where chess pieces belong to the same team all have capital or lowercase character `mark` stored in corresponding class's private section. Another important function in `ChessBoard` is `player_turn()` which is used to switch player in each round. In this function, `do…while` loop is used.

The first and the last classes combined solve the first problem mentioned in the first paragraph of this section, while the second problem is solved by the second abstract base class with all of its derived classes. The relationship of these classes are shown in Figure 2.

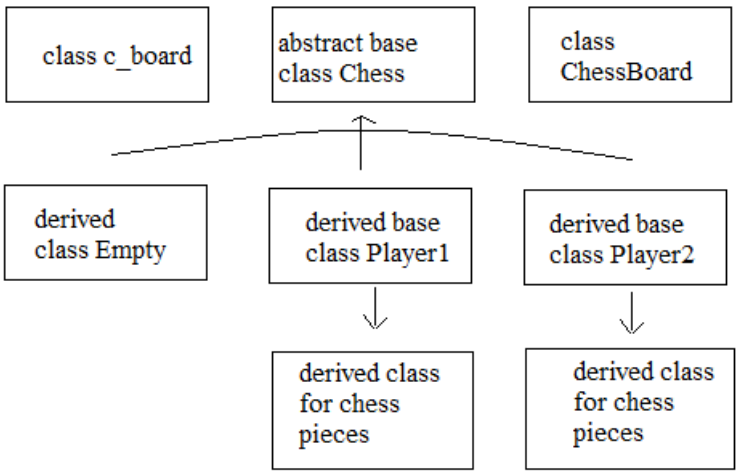

Figure 2: Schematic diagram of the class structure of the code. Chess is the only base class with all its member function are pure virtual, allowing its derived classes to inherit.

There is one function not in any classes called `battle_demo()`. This function is to initialize all classes and their members, and decide the winner when one team loses its General. This is the only function needed to start the game in `main()`.

4. Results

When run the code, a chessboard will be created and all chess members will be initialized on the correct position on the chessboard as shown in Figure 3. In each round, a player will be asked to input the position of a chess piece on the board to choose which chess piece to move, and then possible movements will be indicated on the chessboard in either numbers (1~9) or symbols, such as '!', '@', and ';' as shown in Figure 4. The user will then be asked to choose one of the position to move the chess piece. This action will either move the chess piece to an empty position or eliminate an enemy's chess piece. Once the General of one player is eliminated by another, the game is over and the player with the General alive wins, as shown in Figure 5.

Figure 3: Schematic diagram of the console window when initialise the program.

When choose which position to move the selected chess piece, the user may also type '<' to change the selected chess if accidently typed in wrong coordinates, as shown in Figure 4.

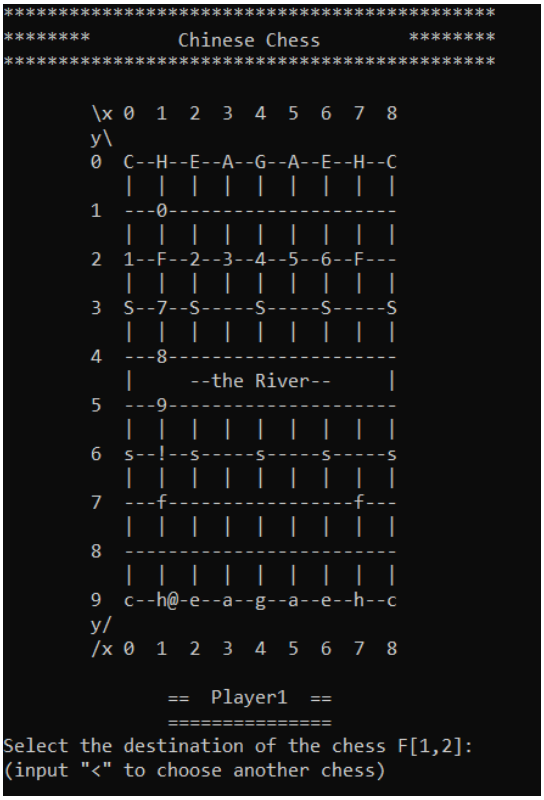

Figure 4: Schematic diagram of the console window when a chess piece is chosen. The numbers and symbols displayed on the chessboard are possible movements for that chess piece. It is free to choose by the user.

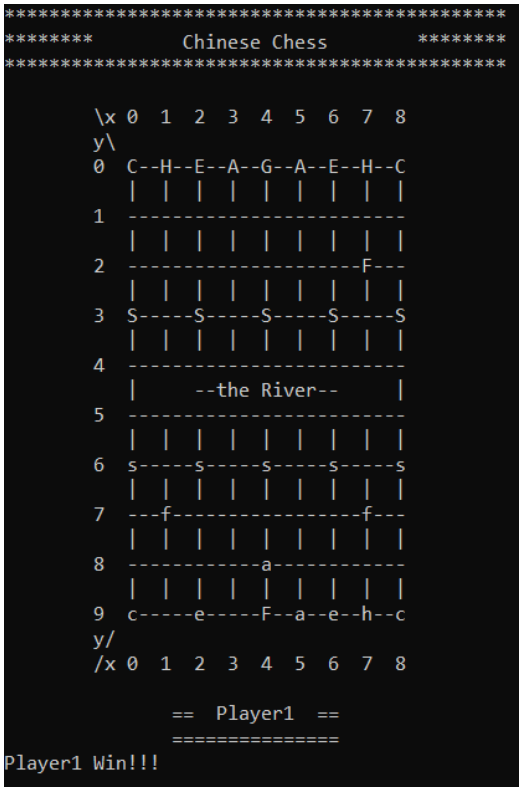

Figure 5: Schematic diagram of the console window when a player wins the game. The General of player 2 is eliminated the Cannon of player 1. Thus player 1 wins.

5. Discussions and conclusions

Chinese chess has more than 3500 years of history, it has some specified rules in different areas in China. For example, when there is no chess piece on the line between two Generals, one General may eliminate another directly and win the game. These kinds of special rules are not included in this code implementation. In addition, a more visually appealing manner of program display may be achieved by adding `<graphics.h>` library, which is used to add graphics to the program. There is also a point of inconvenience of the user interface. When the user inputs the position of a chess piece, some of the possible positions the selected chess piece may travel to are indicated in symbols, where symbol-like representation is not as easy as number to read on the chessboard. This is due to limited ASCII numbers [2].

In conclusion, as the results shown in section 4, the program is well compiled. No errors nor warnings shown and no memory leakage. The program successfully achieves all rules of Chinese chess and allows two players to play the chess. When one chess piece is chosen, all possible movements are shown and can be freely chosen by the player.

References

[1] http://en.chinaculture.org/library/2013-11/21/content_496587.htm, *Chinese Chess*, ChinaCulture.org, 11/05/2021.

[2] https://www.guru99.com/cpp-char.html, *C++ Char Data Type with Examples*, Guru99, 14/05/2021.

Number of words (excluding references): 2355.