# C Project Report: *CHINESE CHESS*

**Abstract**

We want to make a game about Chinese chess. In the game, there is one or two player,and there are two mode to choose-people to people and people to meachine.In the first mode, either of player control one side,and move a piece from one place to anothor place in turn,until one side meet the condition of win-kill another side’s king.In another mode,one player will control red side to battle with black side controlled by computer who will try its best to win.And there also has a starting interface including two mode’s start and setting menu where you can make some adjustmest in this game as you will.

**Introduction/Problem Statement**

Chinese chess, as one of the classic Chinese chess games since ancient times, is convenient, fast and easy to operate. Loved by people. Therefore, our group designed a chess game with good performance, which can meet the needs of most students who love chess.Our project is a excellent Chinese chess game.

Our project has many advantages. Firstly our game has two modes, which can give players more and better game experience. What is more, the game is simple and easy to play. We use a visual and beautiful interface to make it easy to play and ues a convenient and powerful control way- mouse control as our control but also we can set our control way to another way keyboard control.

Another advantage is that we have several beautiful and powerful interface ,for example we have a powerful setting menu, we can make many changes using it, except that ,we have many another small parts, like groundsound,soundspecial,volume and all so.

Group Division

|  |  |  |
| --- | --- | --- |
| Name | Major | College |
| Fan yunying | Math | Jilin University9 |
| Peng yibo | Math | Jilin University |
| Zhang fangning | Biology | Jilin University |

1. **person one**

Design the algorithm related parts, use aphla-beta algorithm, and construct a evaluate function to grade every single situation,and design and accomplish the main and setting interface by QT and fulfill those function by signal slot mechanism,and design the whole thought of this game design.

1. **person two**

the rules of piece and mobile control

1. **person three**

game interface design

**Analysis**

First of all, for the working of interface,we firstly make a small chiness chess game without interface and mouse control which achieves move by scanf the the coordinates entered,so that we can create interface at the same time.We divide the main game into two first-people to people and people to mechine. In the people to people part,we need to do more basic work: first we need to define some basic ingredients like chessboard,piece and flag and so on. Firstly, we want to define a Chessboard using a two demision array and put different letters to present different piece but letter we find that use numbers to replace letters will simplify latter followed work greatly, besides for one single piece, we still need more parameter to describe it, first we to judge its flag, we can make a function but its too simple to be not suitable to make a function ,so we choose to use an array to judge a piece’s flag by compare its position in this array(cause a piece is present by a number,we can use this number to serve as the serial number in this array),next what we face is that piece’s move must obey some rules,how can we make the piece move as the rules, so for a piece,we want to genetate all the possible move and store them in a list, then we compare input move with every move in the list,so we can move as rules ,but how can we generate all points which meets the rules? We want to make a clear rules and try on every single piece,for one single piece we go through all the types of pieces, a piece of beauty according to the direction of the classification of statistics.After that we have a problem that we have to give a input move,there two way can be used,one is thekeyboard and another is the mouse,at first,we

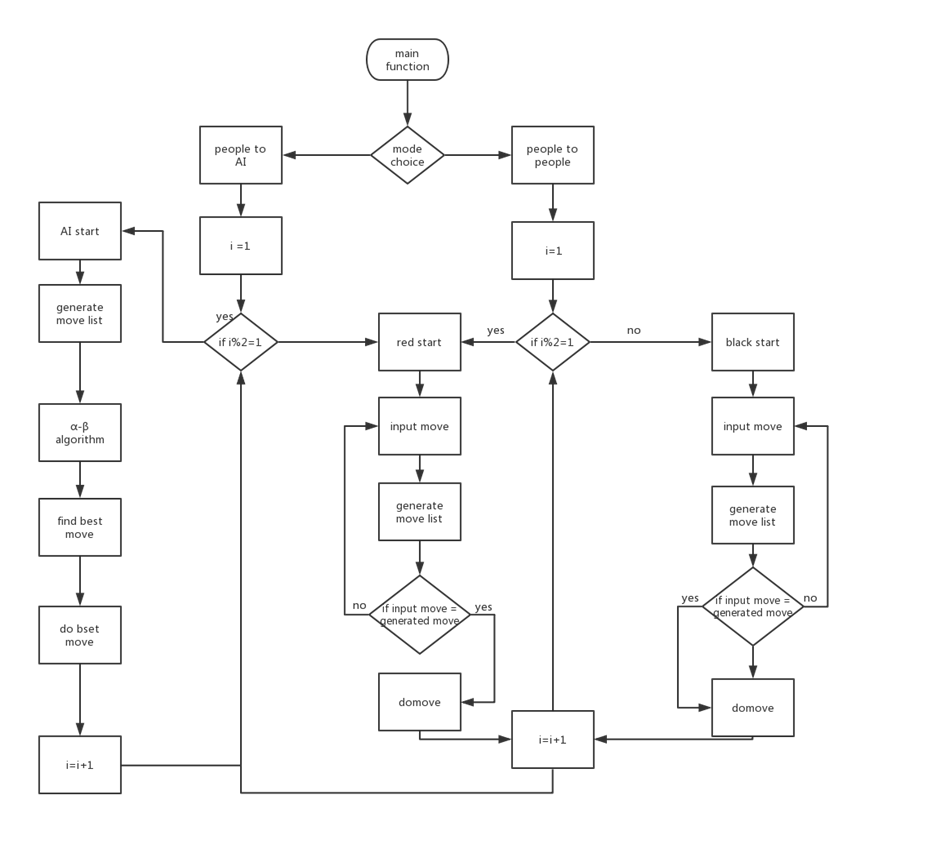
Choose the keyboard-we scanf the position of start and end point ,because it’s easy so we cando test about this program and in a final edition,we will choose mouse,because this control is more friendly to users.

Another mode is AI mode,the most important is make computer make a best move,so we need to find this move,to achieve this goal,we use a classic algorithm named Alpha-beta algorithm to do that,but among this process,what we need to do is a function to evaluate and grade every single situation,and the standard of judgement can be main four:piece’s kind,position,flexibility,relation.

**Design**

First of all, we will in general introduce our three main design and after that display more specific detail.The main three design is the design of game main flow ,the design of the algorithm,and thedesign of interface.As the analysis said,for the sake of make full use of time,the program have two edition,game without interface edition and final edition,out of convenienc,we will display either of two editions.

**1.The design of main game flow**



As we can say,the main flow have two main lines,one is people to people,another is people to mechine, in every single mode we need to make a variable to record the round so that both of two side can move in turn,then,what we do is to finish flow of one round for red,black,human,mechine side,but if we set huamn is red side,we just slove three situation,in fact we did so.

The situation of Black and Red is similar. The only difference is the number of rounds we set. For each successful operation (corresponding to the movement of the rule), the number of rounds will be increased by one, so we set the number of rounds to an odd number. Execute the red side operation and vice versa. Therefore, we only explain in detail the red side and computer operation. First start, pass scanf a number to mode selection, if the number is 0, then the mode is 1 for the man-machine mode after the mode selection, and then through the traversal method printf out the board and each of the pieces above it through a for The loop loops through the rounds and uses an if condition to determine which round operation should be performed. For the red round operation, we first move through the external input (this process will be explained in detail later), then we need to determine whether the move meets the rules. According to the analysis part, we know that we need a function. Generate all feasible moves (this function will be described in detail later), then store all possible moves in an array called the sequence of the law, and then determine whether the input movement is within the sequence of the law, If not, re-enter the input. If yes, perform a move, that is, assign the name of the chess piece of the starting point coordinate to the name of the ending piece to realize the movement. After the movement is realized, we will display the new board and set it in a similar way. A pointer variable step indicates the number of rounds. The value of the address of the pointer variable is incremented by one for each successful execution. The reason for setting the amount of change to the pointer is to ensure that one can be added in different round operations. The man-machine mode is to find the best move directly and execute it and then add one to the number of rounds. As for how to find the best move, it is the content of the following algorithm part, which will not be discussed here.

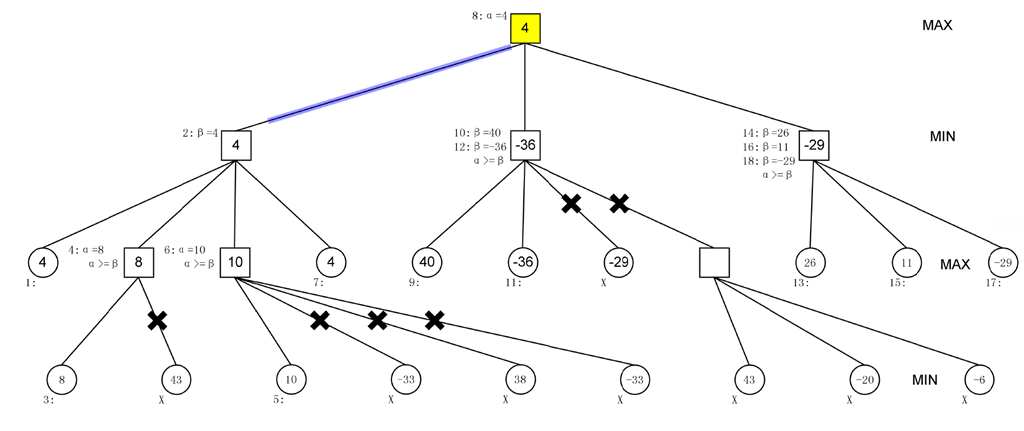
In summary, the main flow of the game is described above, in addition to generating the procedural function, the external input movement and the 2 algorithm part, the problems of everyone and man-machine parts have been solved. Below we will talk about the second major design-algorithm design.

**2.algorithm**

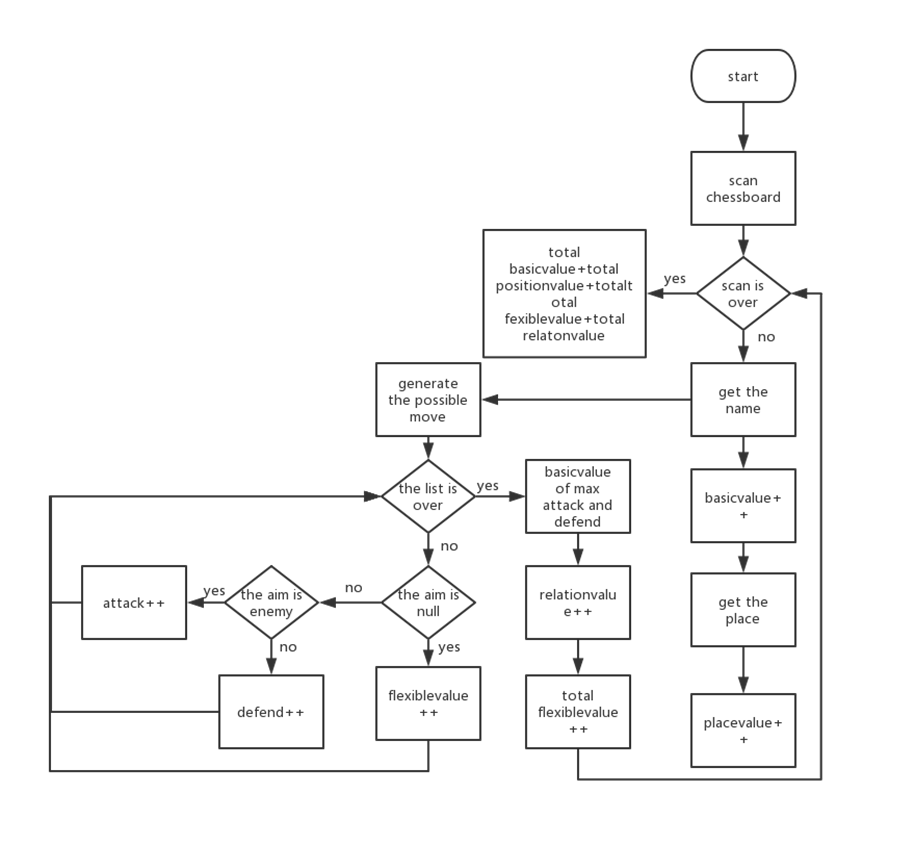
In chess, both players know where each piece is, they take turns and can take any reasonable move. The purpose of playing chess is to kill the other party, or to avoid being killed. The chess program uses the "search" function to find the law. The search function gets the game information and then looks for the best move for the program side. A shallow search function is implemented using "tree search". A chess game can usually be regarded as a large n-tree ("n-fork tree" means that every node of the tree has any number of branches leading to other nodes), and the current situation on the board is the "root situation". "or "root node." Taking a step from the root situation, the situation reaches the "branch" of the root situation. These situations are called "subsequent situations" or "subsequent nodes." There is a series of branches behind each subsequent situation, and each branch is a reasonable move for this situation. The tree of chess is very large and very deep. Each game is a huge n-fork tree. If you can find the best way to play for both sides through the tree search. This shallow algorithm is referred to herein as "minimum-maximum search." It is also possible to use mathematics to deal with chess in this way, but it is not feasible to use a computer to implement it in the near future. Even so, we can still play chess with a program based on minimum-maximum search. Searching for the first few steps in a given situation is possible compared to the smallest-maximum search for the entire tree. Since the situation of the leaf nodes failed to search for killing or chess, a heuristic function called "evaluation" was used to assign values ​​to these situations.

Usually a chess situation has about 35 reasonable moves, so with a minimum-maximum search to search for a layer of depth, there are 35 situations to check. If you use this function to search two layers, there are 352 situations to search. . This is already thousands, and it doesn't look like it, but the numbers are growing very fast. For example, the search for six layers is close to 2 billion, and the search for ten floors is more than two trillion. To check the first few layers of the search tree and use heuristic evaluations on the leaf nodes, it is important to do the deepest possible search. The min-max search does not allow for deep searches because the effective branching factor is too large. This is why we chose Alpha Beta search.

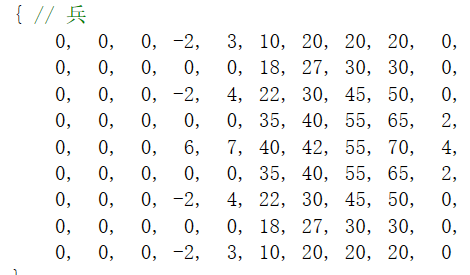
The working principle is shown in the figure. Since this algorithm is more classic, it is not described here.



In fact, what we need to do in this algorithm is to construct a function to score and evaluate each individual situation, so our main work in the algorithm part is to build an evaluation function. In the initial stage of constructing the evaluation function, we reviewed some of the relevant data, and learned that the accuracy of an evaluation function to assess the situation largely determines the chess power of AI. We set a comparative basis and simple evaluation function here. Considering four parameters to evaluate a situation, we first know that the value of each piece is different, so each piece will have a basic value that depends on the type of its piece, and the position of each piece is different. Different, in which the value of the soldier is most obvious with the change of position, and the value of the soldier is not obvious with the position, so each piece has a value depending on its type and position. Another consideration is the mutual relationship between them. Relationship, for a pawn, if its landing point is not occupied, then we call this pawn a free state at this point. If the landing point is occupied by the enemy, then the enemy can be attacked by the pawn. It is the attack state, and if it falls to the friendly army, if the enemy attacks this friendly army, it will be changed. Strike, that is, the piece can play a role in protecting friendly forces, called for the protection of state. The value of the relative relationship of a pawn is reflected in the value of the pawn he can attack and protect, and the number of free states in which he moves. In summary, the preliminary concept of the four basic parameters, after the review of the data, we have obtained a lot of experimental verification for the comparison of scientific data set as the basis for us to set up relevant data, specific information about these data, in the next A detailed description will be given in the module. Below we will use the basic data to score and evaluate each situation. The specific process is as follows:



It can be seen that we first traverse the board. For each board with a non-zero value, it has a child. For each board with a child, we can judge the type of the board by its number and pass the checkerboard on the two-dimensional board. The sequence number in the array calculates its coordinates. After that, we can calculate the basic value and position value of the piece by the corresponding formula through the known data table.



Take bing’s example.

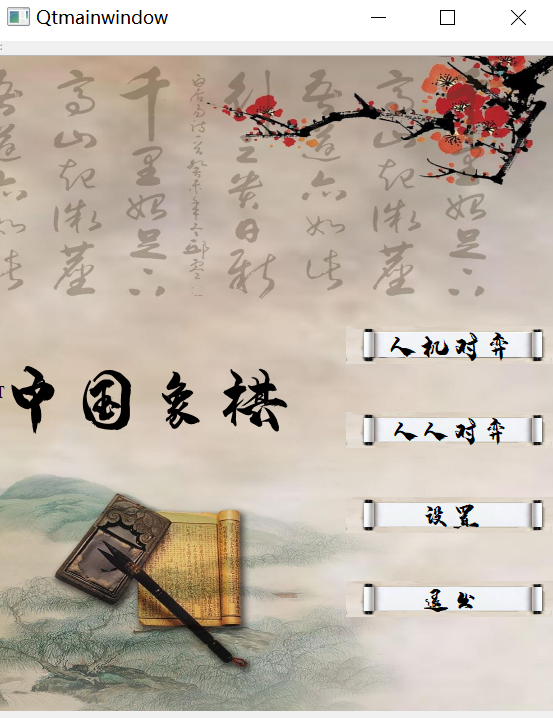
At the same time, we again produce all possible points for this piece (this function will be explained in detail later, in fact almost the same as the generated function in the main program but also different)

Then deal with the three cases of the falling point. If the falling point is empty, increase its free value. If the falling point is an enemy or friendly, deposit it in the relational table (a record has an attack or protection relationship with the studied piece). After the analysis of all the possible falling points corresponding to the piece, the free values ​​of all the points are added as the free value of the piece, and then the one with the largest basic force is selected in the relational array. The best value that the chess piece can achieve in terms of offense and defense is recorded as the relationship value. After calculating the four values ​​for a piece, continue to calculate the value of the next piece, and finally add the four values ​​of all the pieces, respectively, to obtain the total valuation of the red and black, and use the operation round The valuation is subtracted from the valuation of the other party, which is the valuation of the situation. So far we have basically achieved an assessment of a given situation.

**3.third major design - interface design**

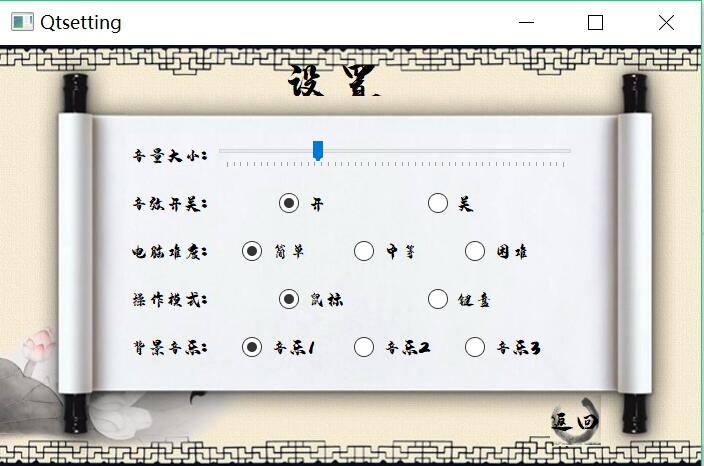
There are three main interfaces in the program, the setting interface and the game interface.

**3.1 main interface**



The main interface mainly plays a function of cutting pages. There are four options on the main interface: everyone's game, man-machine game, setting and exiting the game. The QT signal slot mechanism is used to realize these four functions, namely in the slot function. Construct a new form in it, then close the main form, and the exit key is to close the main form directly.

**3.2 setting interface**



There are some adjustments and settings for the basic settings of the game in the settings interface, mainly by the following options: music volume, sound effects, background music selection, game difficulty selection and return button. Mainly through QTdesigner to operate the corresponding control and its style, and then realize the function of its slot function

Most of them want to pass a parameter to the game subject. This parameter is used to control the switch and selection of the music sound in the game. (The functions of playing the music sound are all designed in the game and will be discussed later.)

**4.other designs**

**4.1 External input movement function**

4.1.1 Keyboard manipulation

The implementation of the keyboard operation is relatively simple, and the implementation method in the analysis is almost the same, it will not be repeated here.

4.1.2 mouse manipulation

supplement

**4.2 Generate the normal function and generate the possible drop function**

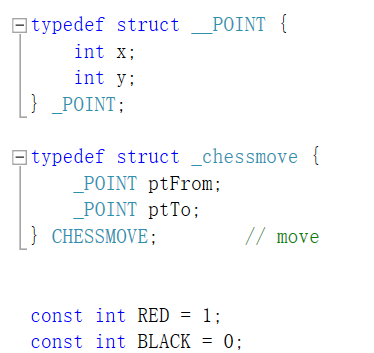
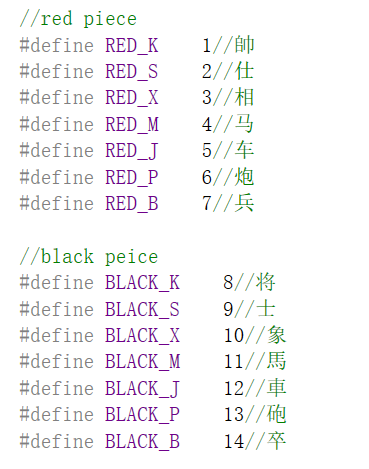
Since the two generation functions are too large, but the design idea is relatively simple, there is no detailed description here. Only some differences between the two generation functions are compared,When a feasible drop point is generated in the algorithm, the feasible fall point generated is not necessarily a feasible move. It has three possible free positions, attack positions and protection positions, and the generated drop function used in the main program generates Feasible movement, that is, its free and free position and attack position. If there is a friendly position, the movement cannot be realized. This is the difference between the two generation drop functions. Considering that the functions of these two functions are relatively large and the content is highly identifiable, we have conceived to unify or merge these two functions, which can reduce a lot of space, but for reasons of time, this idea has not been realized. But in order to reduce the generation of the falling point function in the next module display algorithm only after the length

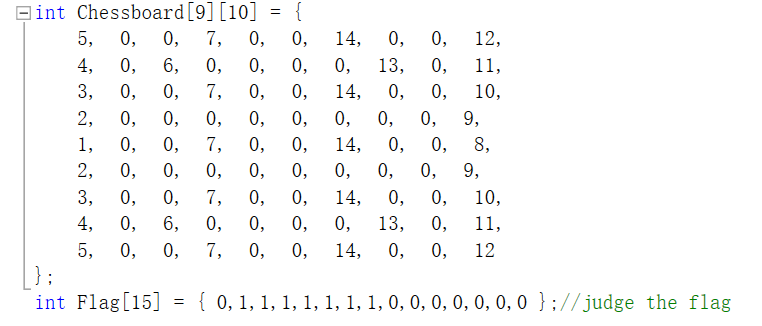
**4.3 Several small functions**

These small functions about music sounds, etc. are shown directly in the implement section and are not shown here.

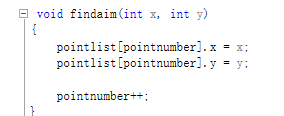
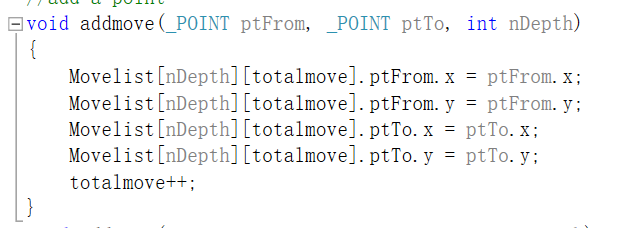
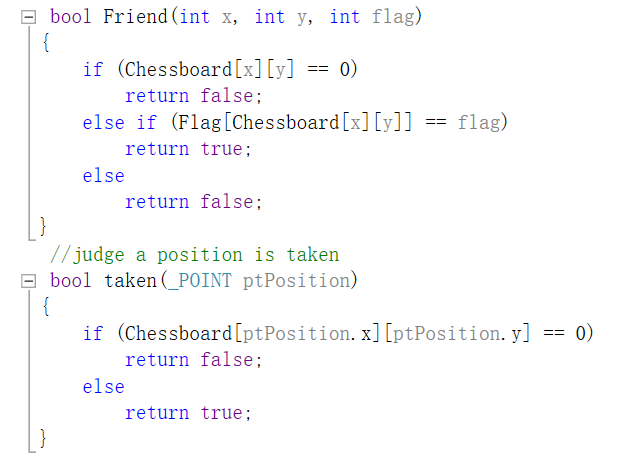
**Implementation**

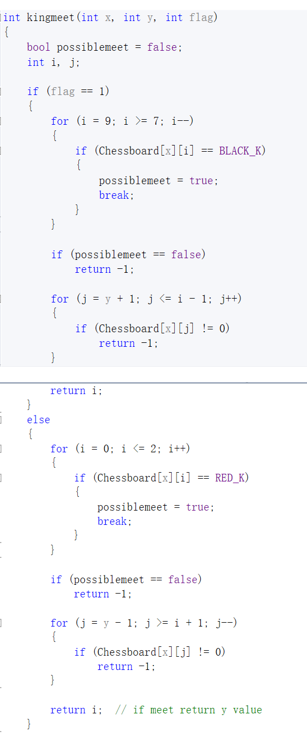
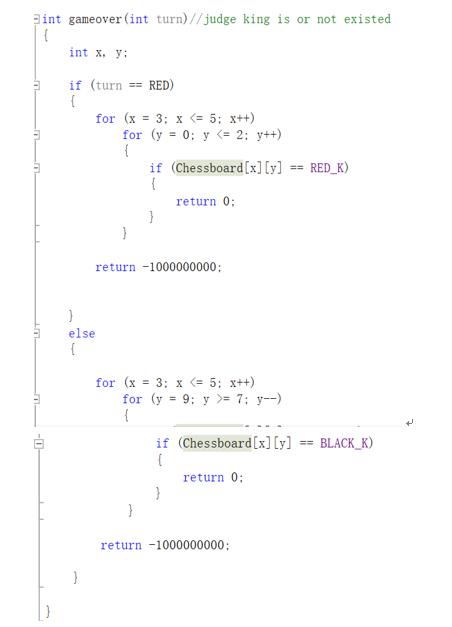
**1.Several settings in everyone's battle**

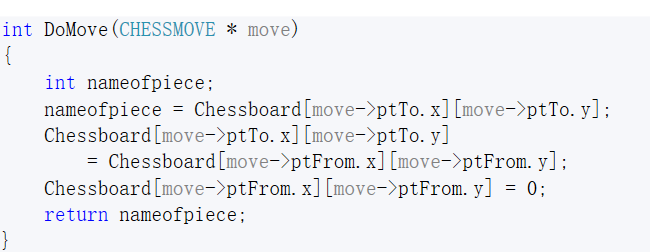




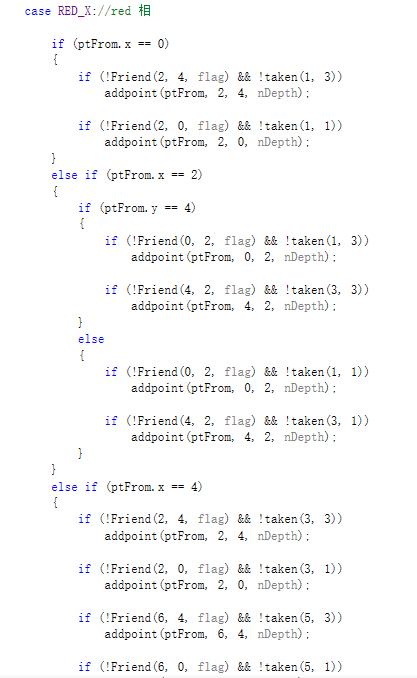
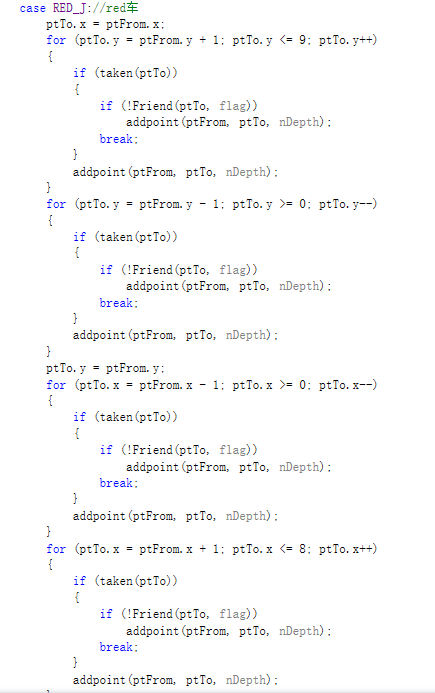
**2.Several functions in everyone's battle**

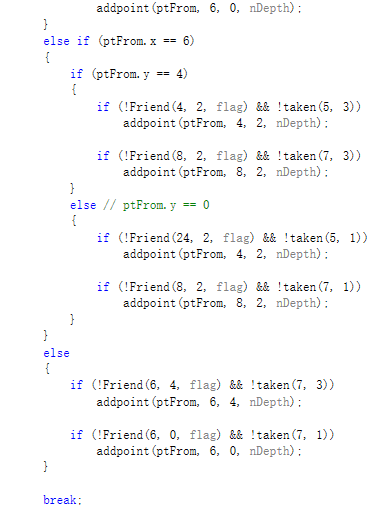


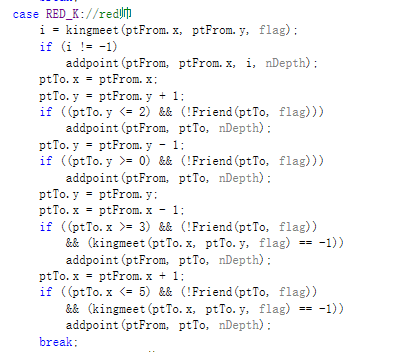
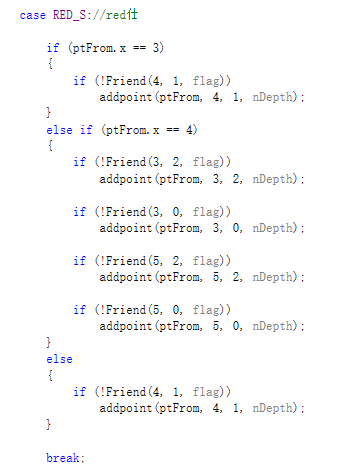


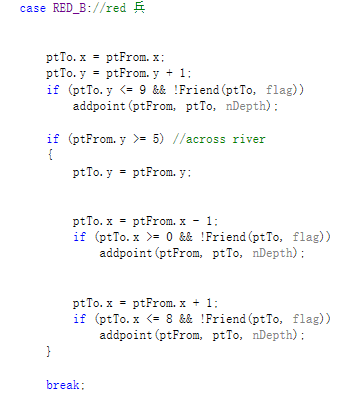


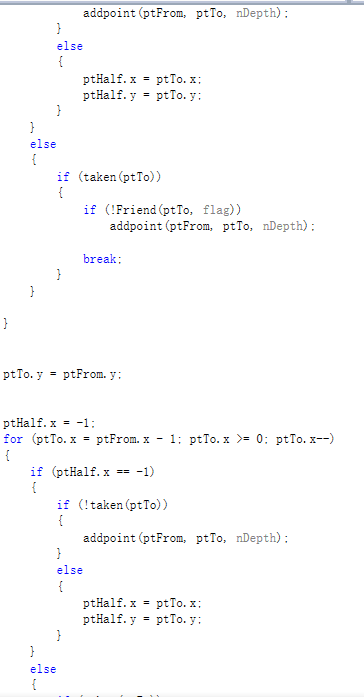
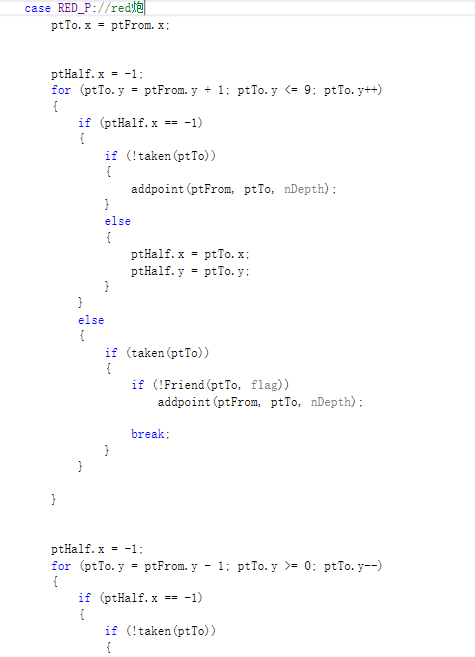
**3.Implementation of generating drop function in algorithm**

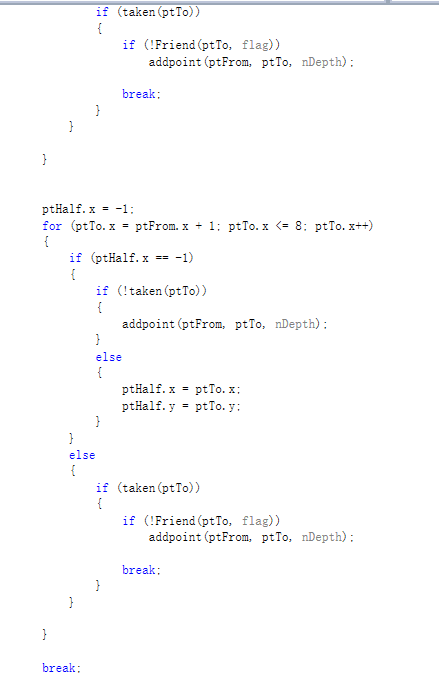


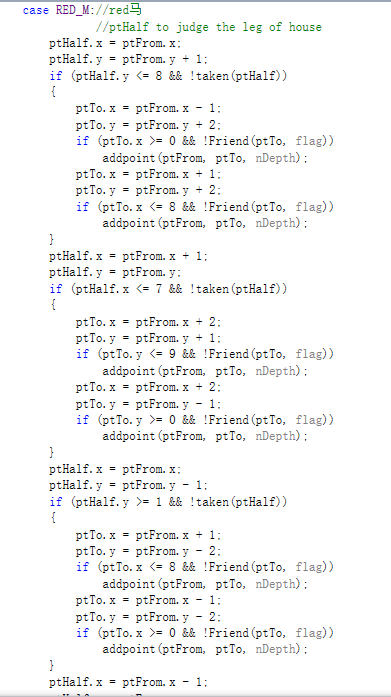
C:\Users\hp\Documents\Tencent Files\1016598068\Image\C2C\H[427O`QBCN{_608T9DR$6W.png 

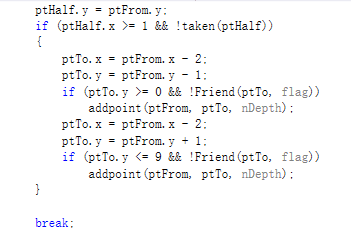




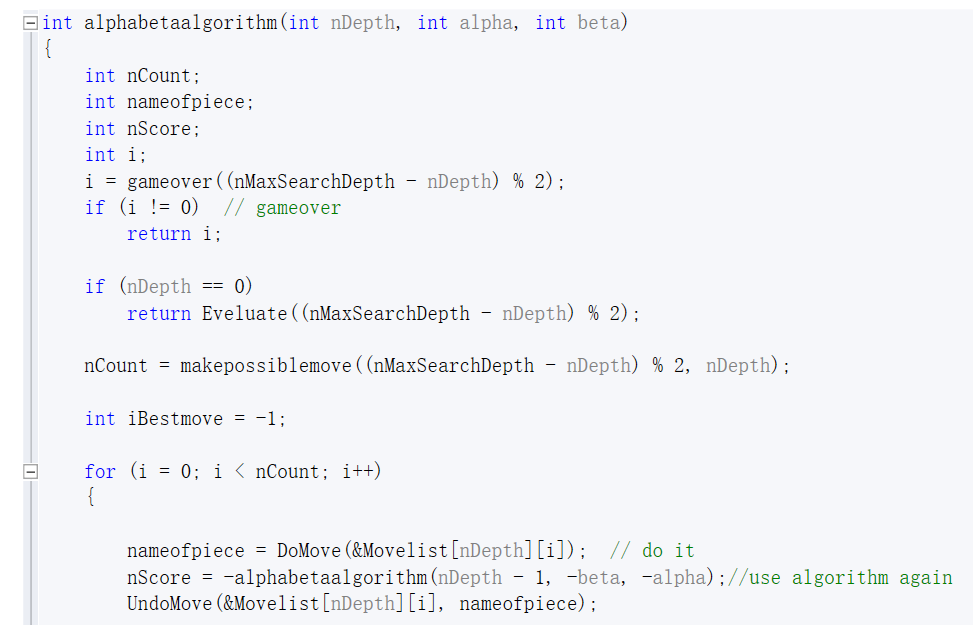


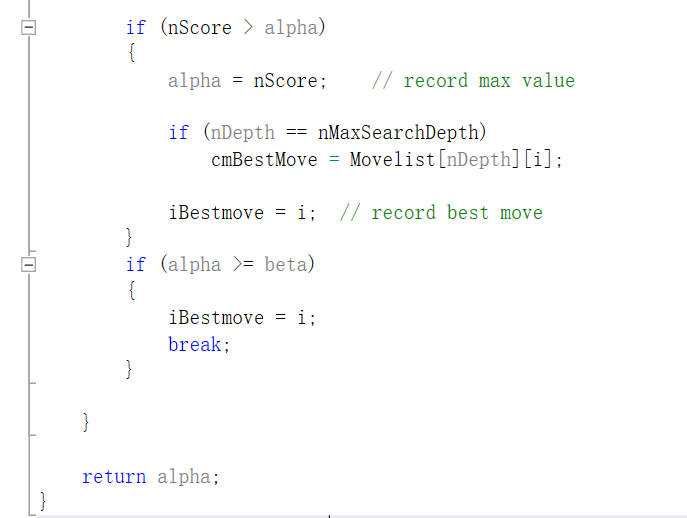


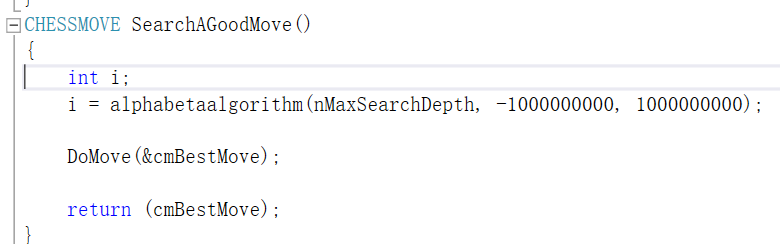




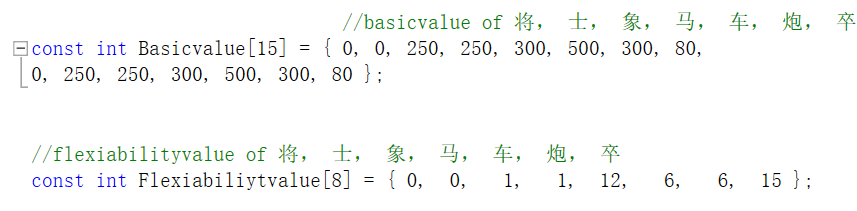
**4.for the best step implementation**

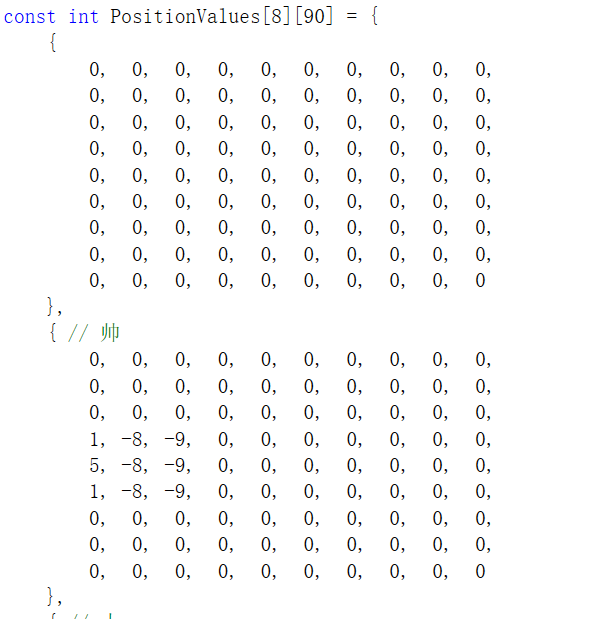


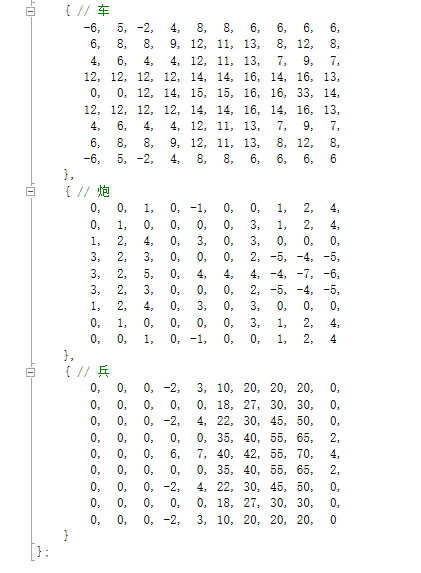
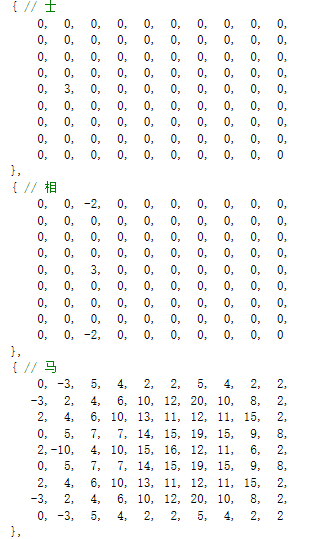




**5.The representation of the parameters in the evaluation function and the implementation of the calculated evaluation values**







**Testing and Debugging**

**Result&Conclusion**

The aim of our project which is running a game has already achieved .In this game, both can we battle with another person and computer,and our beautiful interface and delicate music will polish this game,and friendly setting can make more people to play it more easierly

After the programing, we have learnt a lot. We not only learn the knowledge from class, but also learn more knowledge from our classmates and from the internet. When we met problems, we learn to find the answers or the solutions by ourselves. I think this is one of what we have learnt in the C programing class. The next thing we have learnt is that we learn to collaborate better with our classmates. In the programing process, we write different codes. And we learn how to divide the works and how to link them. The final thing we learned is how to express us properly, how to give a presentation and to grab the audience’s attention. When we give the presentation, we should meet what professor demand. Now the programing is done. But I think that we will often use the C programing and we will never stop learning the C programing in the future. Finally thanks to professor Hao, and Professor AT.