**Project Report: An Advanced Aeroplane Chess**

**Introduction**

Aeroplane chess is a modern Chinese chess game based on a kind of British chess. Like other chess game, aeroplane chess have been transplanted to computer platform, and we can find many versions of aeroplane chess game online. However, traditional aeroplane chess seems to lose its attraction among teenagers due to the restricted game mode. In this project, our group created an advanced aeroplane chess game. It can not only perform the function of traditional aeroplane chess, but also add new features to make the game totally different. Additionally, this game supports LAN battle, which means friends can play this game online. We hope by introducing this project, our generation can recall the happiness we had once upon a time.

Here is our group members’ name and contribution list:

(此处插入名单和分工表)

**Description of the game**

When you enter the beginning interface of this game, you can start a game, read the help, or exit the game. After you start a game, you can change the game mode (online/offline mode, traditional/advanced mode), numbers and type of players. （这里加图）

Game mode:

There are two game modes: traditional mode and advanced mode. Traditional mode is not different from the most common aeroplane chess game. As for the advanced mode, we add an ability card system to the game, which allow players to use certain ability cards to hinder enemies or buff their own chess to win the game. The mechanism of ability card system is as follows:

1. When players roll the dice to 1 or 6, the player get a chance to draw an ability card instead of moving a chess. One player can only possess at most 1 card at a time. If the player draws a card when having a card already, the original card will be replaced by the new card.
2. One ability card can only be used once, and one player can only use ability card once in one turn. When the player chooses to use ability card, then he or she cannot roll the dice or move chesses anymore.
3. There are four kinds of ability cards, and each of them can be drawn with equal possibility. They are:

* Attack: assign this ability to one OWN chess. Enemy chess less than 5 steps far from this chess will be crashed (send back to airport) unless they are protected.
* Defense: assign this ability to ANY chess. This chess will not be attacked (but still can be rammed) or be interfered in 5 rounds.
* Interfere: assign this ability to ANY chess. The chess cannot move/attack/defense in 4 rounds.
* Eliminate: assign this ability to ANY chess. The buff state (defensed/interfered) of the chess will be eliminated.

**Basic Framework**

There are three main components in this project, which are main interface, AI core and LAN system.

（这里需要图）

* **Main interface**
* **AI core**

The inspiration for this AI core comes from a design of Chinese chess AI. This aeroplane chess AI consists of three basic classes: the move generator (movegenerator.h / movegenerator.cpp), the evaluator (evaluator.h / evaluator.cpp) and the search engine (searchengine.h / searchengine.cpp). Move generator is a class which can generate and store all possible moves according to the current game state. Evaluator is a class which can evaluate the value of a specific move; the larger the value, the better the move. Search engine is the main class for this AI core; it combines initialized move generator and evaluator object in order to simulate a real player’s action. In addition, the AI core contains a define.h in order to store some common data structures and macros for the above three classes. Detailed implementation are as follows:

**define.h**

This head file defines some necessary data structures and macros for the AI. For example, the representation of chess ID, chess colors, chessboard coordinates and etc. They are configured as some macros for the convenience of programming.

There are three important data structures: CHESS, CHESSMOVE, and COORDINATE. CHESS is a structure containing the ID (representing the chess uniquely), the color, the coordinate, buff state and buff round left of a chess. CHESSMOVE is a structure containing the ID of the chess being manipulated, the roll point and the information about ability card using. COORDINATE is a structure containing a string representing the region and an integer representing the location. The reason for using this coordinate representation is that the chessboard of aeroplane chess is not full-covered. Some points on the two-dimensional plane is out of the chessboard, and chess will have different behavior in different region of the chessboard. Therefore, I did not choose two-dimensional coordinate in the convenience of computation.

**Move generator**

When creating a move generator, the constructor will set off mode (deciding what roll point can take off), set the game mode (traditional mode or advanced mode), and set random seed. Setting random seed when initializing will let the dice roll result different every time.

The core function for move generator is createPossibleMove(). This method generate currently possible move according to current chessboard, roll point, the color of the player and ability card the player have. Each possible move is constructed into a CHESSMOVE structure and is stored into the moveCount array.

**Evaluator**

Evaluator only have one function evaluate(). The main logic for evaluating is to compare the chessboard before and after a certain move. If the new chessboard results in a favorable situation, the value will be higher. If the new chessboard confronts a bad situation, the value will be lower.

Because of the mechanism of aeroplane chess, the evaluation process is basically a simulation of human player’s thinking mode. First, the evaluator counts benefits of a move. Benefits include bonus from using ability card efficiently, crashing enemy chess, moving own chess ahead, etc. Then the evaluator counts damages of a move. Damages include the threat from enemies and punishment from overlapping. The final value is counted by benefits subtracting damages.

**Search engine**

This is the main part of AI. When initializing, the search engine must point to a move generator object and an evaluator object in order to function. The play() method will let the AI change chessboard intelligently. This method simulates player behavior like this:

1. Copy the chessboard into cur\_Chessboard which is inside the search engine instance;

2. Roll the dice. Then create move with createPossibleMove().

3. If there is only one move, it must be "not possible to move" case. Then do not do any movement. If there is more than one possible move, search a good move and make move with three methods, searchAGoodMove(), makeMove(), useAbility().

4. If game is not over and roll point is six, continue until the third time roll. Else, player end its turn.

5. Copy the modified chessboard in place to the original one.

searchAGoodMove() will evaluate all possible move and choose the move with the maximum value. makeMove() and useAbility()will modify chessboard according to the move AI choose.

* **LAN system**