Checkers Game Data Model Concept

Yu Hou 14256376

General rules

Checkers is played by two opponents, on opposite sides of the game board. One player has the dark pieces; the other has the light pieces. Players alternate turns. A player may not move an opponent's piece. A move consists of moving a piece diagonally to an adjacent unoccupied square. If the adjacent square contains an opponent's piece, and the square immediately beyond it is vacant, the piece may be captured (and removed from the game) by jumping over it. Only the dark squares of the checkered board are used. A piece may move only diagonally into an unoccupied square. Capturing is mandatory in most official rules, although some rule variations make capturing optional when presented. In almost all variants, the player without pieces remaining, or who cannot move due to being blocked, loses the game.

Checker Board

The checkerboard is designed as an 8*8, 64 alternate dark and light squares board.

For UI representation, use 2D array to represent a Checker board. Each array element holds two values, one is the position, the other is the state (empty or has a piece).

Player

Use String name to get the name of the player. Who will go first is determined randomly by color. Once a movement is detected, it will first check whether it is legal and then move on the board, otherwise it will notify the user and request to move again.

AI Algorithm

If AI moves first, choose one piece of movement randomly, otherwise, choose the nearest to the user’s one to move. The next step is to refine the pieces closest to the user’s piece until the next step exceeds the user's piece. Then we will find the shortest way to the bottom of the board, then will move one step, and wait for the next action of the user. If the user's behavior is on the path of the shortest path, move one more step, else, recalculate the movement to fine next user’s piece that is closest to the bottom. Once the AI reaches the bottom of the checkerboard, the calculation path is changed to the closest user’s piece. Then start to move to the user.

```
Class bool check (int player, int piece, int from[], int to[])
{
        if(player == 0)//dark player
        {
                while(1){

                        if(to[0] == from[0] + 1)
                        {
                                if(to[1] == from[1] -1 or from[1] + 1)
                                        if no piece, return true;
                                        else if has a piece,
                                                row = to[0] + 1, col = from[1] + 1
if row + 1 col + 1 or col - 1 no piece, return true;
                }
        }
}
```

Game State Determine

A player wins by capturing all of the opposing player's pieces or by leaving the opposing player with no legal moves. A Player loses by he doesn't have any piece left, or cannot move because being blocked. The game ends in a draw, if neither side can force a win.

```
Class int Determine()
{
        if rand() % 1 = 0
                return 0 //not get dark
        else
                return 1;
}

Class void winner(int user1, int user 2)
{
        if(user1 == 0)
                print user 1 lost;
        else if (user 2 == 0)
                print user 2 lost;
        else
                print draw
}
```