Chess AI notes

Chess - a game of perfect information, where the positions of all the pieces are known to both players and each player can anticipate the opponent’s next move as a response to his/her own move

Minimax algorithm - Assumes that each player plays to the best of his/her ability. In the game of chess, or in most other two-player games, one of the players is assumed to be the “min” player while the other is assumed to be the “max” player, with the idea being that the min player tries to minimize the score of the board at each move of its own and the max player does the opposite i.e. tries to maximize the score of the board at his/her own move. This evaluation is useful in Chess, since an integer scoring system can be set up based on the current positions of the pieces in the game, where a more positive score favors the max player while a more negative score favors the min player.

Max algorithm:

The max algorithm in our chess game will search through every piece on the board, followed by making every possible move that piece can. For each of these moves, the mini algorithm is called, which does the same for the opposite player. Mini will call maxi once it has selected a move. These two algorithms bounce back and forth until the desired depth is found, once this is done, the board is evaluated and the score with the best outcome will be chosen. The AI will then pick the move which leads to that board in theory. This is a depth-first search.

Alpha Beta Pruning

The Alpha-Beta pruning algorithm is used to “prune” off useless sections of the trees, this can drastically increase speed of the minimax method. If a path of the tree is too bad of a move and cannot lead to a better move, we can stop searching that path immediately.

Board Evaluator

The board evaluator is dependent on two factors. The first is the score of the piece, the evaluator tallies the score of all the pieces, for example the pawn is worth 100 points for white and -100 for black. Pieces can also lose or gain score depending on their board position, for example the pawn gets +50 points for being in the centre 4 squares, helping the AI with its positioning. Each piece has an individual score table; for reference , here is the King’s

private static int[][] kingScoreTable = {

{ -100, -100, -100, -100, -100, -100, -100, -100 },

{ -100, -100, -100, -100, -100, -100, -100, -100 },

{ -100, -100, -100, -100, -100, -100, -100, -100 },

{ -50, -50, -50, -50, -50, -50, -50, -50 },

{ -10, 0, 10, 10, 10, 10, 0, -10 },

{ -10, 10, 10, 10, 10, 10, 10, -10 },

{ 30, 25, 0, 0, 0, 0, 20, 20 }, { 20, 10, 60, 10, 10, 0, 60, 20 } };

Since the King should be in your own end, it gains points for staying there, as well as losing points for venturing out. The King also gains 60 points for castling.

Sources:

Chess Bin. (2008). [Chess Board Evaluation. Retrieved from http://www.chessbin.com/post/chess-board-evaluation.aspx](http://www.chessbin.com/post/Chess-Board-Evaluation.aspx)

Chess Bin. (2008). Chess Piece Representation. Retrieved from http://www.chessbin.com/post/Chess-Piece-Representation.aspx

Chess Programming. (2009). Minimax. Retrived from https://chessprogramming.wikispaces.com/Minimax

Chess Programming. (2009).Alpha-Beta. Retrived from http://chessprogramming.wikispaces.com/Alpha-Beta

Ai Horizon. (2010). Chess Artificial Intelligence. Retrieved from http://www.aihorizon.com/essays/chessai/