Ai Chess Game

Design Approach:

The Ai Chess game consists of four classes: Algorithm, Evaluation, Piece Position and Gui.

The main user interface of the chess game is defined in the Gui class.

The interface consists of chess board, all the input required by the users is suppose to be selected from the pop up menu's, only at the beginning of the game.

I have implemented the pop up menus as it is easy to take values from the user and set it to an assigned task.

The first pop up ask for the search depth, i.e. how deep the ai should go in order to evaluate the optimum move. The second pop up asks the user to enter the branching factor i.e. the number of children at each legal node, the game should consider while evaluating a move. The next pop up asks for an Game playing mode. I've implemented two modes: Normal and Aggressive. The pop up after that asks for the user whether they want to block AI's piece permanently, i.e. Ai can't move that piece during the entire game. The final pop up asks the user to select a side: black or white. The interface lets the user know when the king is in check condition or the game is in a stalemate condition. It Also tells the user when he is in a checkmate position which results in Game Over.

Engine architecture:

The chess engine contains the alpha beta pruning algorithm and the different strategies adopted as specified by the user. The strategy is to create an alpha-beta tree diagram which returns the best outcome. Weights are assigned to each of the pieces and depending the the type of strategy followed, these pieces are pruned to give an optimum move

I've used array data structure to define the chess board as wells as for defining the weight of each piece on the board.

The search algorithm being used by the chess engine is Depth First search. This algorithm stops evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. Alpha beta algorithm prunes away branches that cannot possibly influence the final decision. I followed Jonathan Warkentin lessons in chessprogramming.wikispaces.com in order to develop and optimize my chess engine.

Observed Effects of Different Playing Strategies:

I have restricted the search depth to be maximum of 5 and the branching factor to 8 as the time taken to respond to a move when those conditions were selected by the user was slower than the lower search depths and branching factor. If the Branching factor was low, the response time was relatively fast, but the move wasn't optimum and resulted in loss of pieces by the Ai.

When the mode is set to aggressive, the Ai will always move a piece if it can capture an opponent's piece barring obviously the first few moves. This works by pruning all the branches in which there is no chance of taking opponent's move. This made the game being more exciting as Ai will always look to take pieces but this caused its own piece to be in situation where it was in jeopardy.

The next thing i implemented is that user can decide whether to block the movements of Ai's piece or not. I played the game by blocking a queen, I noticed that Ai will always try to protect the queen if it was possible. I implemented this blocking feature as it was the only time I could come close to or beat the Ai in chess considering that the search depth and branching factor was set to max.

In the End, I really enjoyed making this game although i lost nearly all the time but it was worth the effort.