CS 154 Project

Chess

Team members-

110100024- Sagar Jha

110100092- Raghav Gupta

11D170020- Deepali Adlakha

Problem description-

In the project, we've aimed at implementing a single-player version of the board game of Chess, with the AI being simulated using the Minimax search algorithm (with alpha-beta pruning), with the board beingevalutaed using a slightly customized heuristic. The GUI is done with the Racket GUI toolkit.

Design of the program-

Heart of the chess engine-

We have a chess.ss file which has object definitions for each type of piece (pawn%, bishop%, rook%, knight%, queen%, king%), each of which is inherited from another generic class called chess-piece%.

More importantly, it has a class board% which, among other things, has functionality to check for the validity of the move made, capture of enemy pieces, checks, checkmates, stalemates, castling, and pawn promotion.

The board% class has a method which, given the current board position, finds out all the board positions that can be reached from this one in exactly one move. This makes use of methods to return all moves, which are present in each piece class (rook%, knight% etc).

The GUI module-

Please write something here

Artificial intelligence implementation-

When a move is required to be made by the computer, the function *get-best-move* (in minimax.ss) is called with the current board position. This generates the minimax tree, and at a fixed depth in the tree, evaluates the board positions (using the heuristic to evaluate board positions in evaluations.ss), and returns a move which gives us a reasonably good position for the computer.

Note here that we don't simply return the best move as returned by the minimax algorithm; this we avoid for the sake of the computer playing deterministically. Instead, what we do is while proceeding through the minimax search and evaluation, we maintain a list of reasonably good moves which can be candidates for the best move. Then, using random numbers, we select one of these candidate moves probabilistically (clearly, the probabilities are skewed enough so that the best move is returned most often, but there is some scope for other moves to be returned too).

Static evalutation of board position-

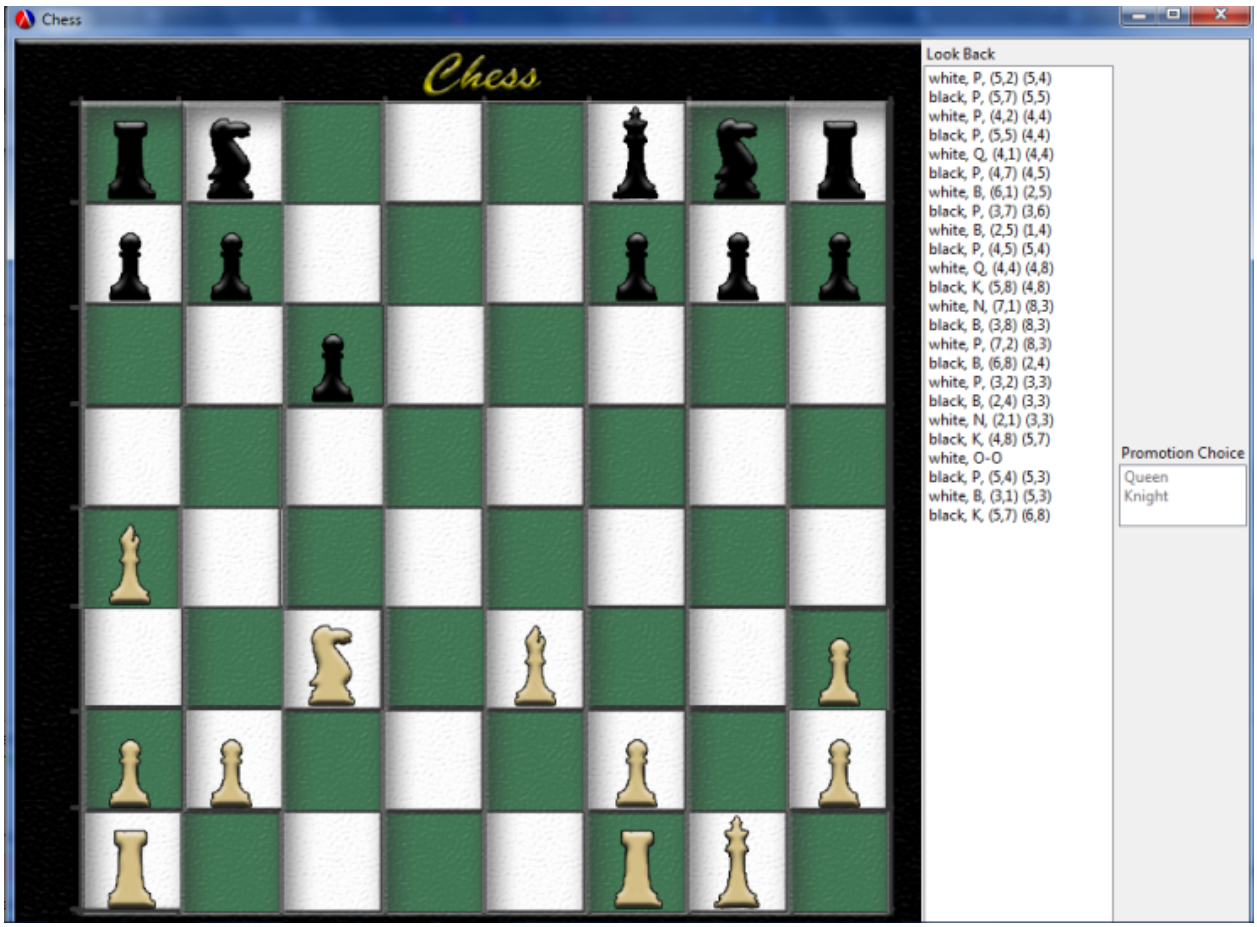
The board positions are evaluated with multiple heuristics which draw from the heuristics used for evaluation by the GNU Chess Engine, but with minor adjustments made for our implementation.

Minor points-

For self-use, we have a simple 2D-vector implementation and some custom functions for list manipulation.

Input/output samples-

In-game screenshot-



Limitations and bugs-

- The GUI does not work as expected on Ubuntu, but it runs fine on Windows.
- The evaluation heuristic used may not always return the best moves. This we attribute largely to our lack of in-depth knowledge and experience of chess playing.
- On Windows, the minimax search runs very slow, while it takes a doable amount of time when we run it on Linux (without the GUI, functioning, of course)
- A pawn can be, in the game, promoted to only a queen or a knight.
- Despite our best efforts to keep bugs out, some may inadvertently creep in during the game. This, we think, may be because the move-validity/castling/pawn promotion checks may not take into account some very rare cases.