PROJECT PROPOSAL

Names of students
1.Navin Chandak (Roll 110050047)
2.Ayush Kanodia (Roll 110050049)

Title of the project
CHESS MASTER (tentatively)

Brief description of the problem
To simulate the game of chess between two players, and a player and a computer. For the detailed rules of chess, please refer to the following link :
http://www.fide.com/component/handbook/?id=124&view=article

Idea of the solution

Technical part
Board Representation
We shall represent the board using standard data structures in scheme. This standard representation may use predefined data strucutres such as vectors or lists. We intend to use object oriented features to simplify coding and to enhance the readability of code.

Basic Game Set Up, Definition of legal moves and possible moves
The focus now is to set up a framework for legal moves. This will be achieved using unique representations for the various chess pieces and separate functions which will return all possible moves of a particular piece given the state of the board. There will be a basic framework which will allow searching of moves. And, considering each one of these moves, the discovery of all possible moves can be made possible.

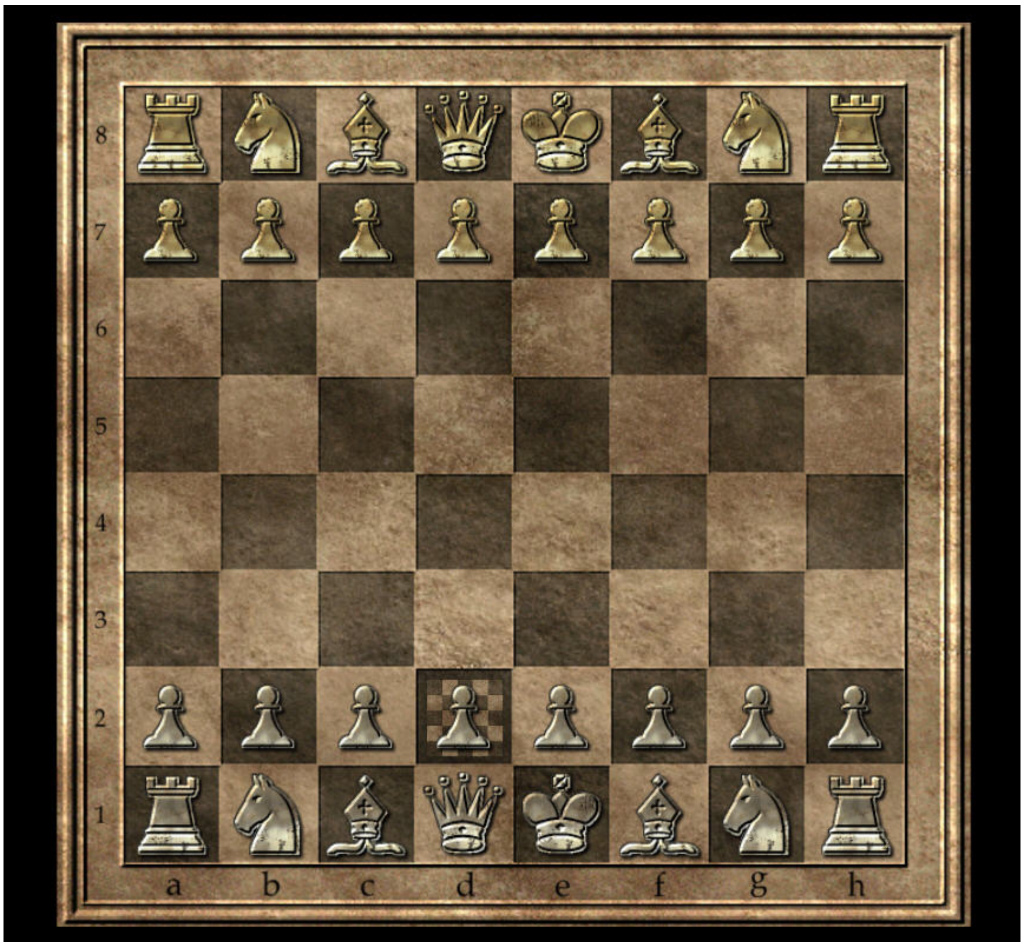
Algorithmic part
We are planning to search for solutions to our problem by using the standard minimax algorithm for maximizing the minimum gains out of any possible situation in an n-player (in our case two player) game like chess. For details, please refer to, http://en.wikipedia.org/wiki/Minimax We shall basically make a tree of possible moves for any given situation in the game. Each level of increase in the depth of the tree would represent consideration of an extra chance. Each of the final nodes of the tree (which essentially will be a state of the chessboard) will be examined for a measure of its desirability. This measure will be subjective to some extent, and will need to be optimised for better performance, and an accurate assessment of this value will hold the key to the computer player's performance.

Apart from this, we will need to make a lot of heuristic approximations in the solution to any given situation, since otherwise, the number of possible cases may become computationally too large, the solution too time consuming, and the game too slow to be played. Also, this will allow us to consider more moves in the first place, which should improve the performance of the computer player considerably. Also, given that the game is completely solvable, and the limitation simply is computation power, the game would probably end up being only as good as to the number of moves it is able to consider before making a move

Sample input output

Input This would typically be mouse clicks to select the pieces to be moved from one spot to another, and to decide the piece to convert to in pawn conversion, or to decide the level at which the user wishes to play the game.

Output Typical output would be the representation of a particular state on screen, some thing of this sort (The following picture is just indicative). It will include highlighting of the piece the user wishes to move, displaying valid moves, and also displaying appropriate messages for the user during gameplay.



Discussion

We are planning to use the graphics library of drracket (The Racket Graphical Interface Toolkit (racket/ gui/base)) for graphics part of our project. The problem of graphics doesn't seem to be too challenging, if at all tiring; the real challenge will lie implementing a good algorithm and reasonable heuristics, which will we shall attempt at our best level.