Plan for chess

by
Tom Ebergen
David Knight

The plan for chess is as follows. Since working together on the same pieces of code may cause confusion between two partners, David and I have decided to split the project up into two main components. David will write the code involving setting up players, board, scoreboard, moving the pieces, and some special moves the pieces can take. I, Tom Ebergen, will work on writing the code for the computer levels 1 - 4.

This approach has been taken because the two pieces of code can be worked on independently. When completed, the computer class functions will return a string containing the next move that should be taken by the computer. This code will then be passed to a move function for a player/piece created by David. This way we can avoid having to update mutually used code based on revisions by our teammate. In addition, if partner A needs partner B to add a function, B just needs to know the parameters, return type, and where it needs to go. Then A can call the function and use it how he is supposed to.

Both members of their team plan to have their code done by Tuesday evening. This will give the two of us all of Wednesday, Thursday and Friday until 5:00pm to debug, test, and get our visual presentation working, i.e Xwindow. We will also look over each others code to make sure there are no mistakes.

Question 1: Chess programs usually come with a book of standard opening move sequences, which list accepted opening moves and responses to opponents' moves, for the first dozen or so moves of the game. Although you are not required to support this, discuss how you would implement a book of standard openings if required.

In order to implement a book of standard opening moves my partner and I would create an association list containing board states and subsequent moves that the computer would read from for the first couple of moves. Each element in the list/array would be a list with two elements. The first element would be a possible opening state of the board and the second element would be the next move that should take place. The computer would then read from this book of standard openings and be able to determine the best move from there. Hypothetically, this would only work for about the first 10 moves. The list could be longer but that may cause the computer to be noticeably slower.

Question 2: How would you implement a feature that would allow a player to undo his/her last move? What about an unlimited number of undos?

In order to allow a player to undo his or her last move or as may moves as they want to, we would implement another association list where each element is a list of three elements. The first element is the final position of the piece, the second element would be the initial position of the piece, and the third element would be the piece that was originally in the final position. The code would locate a piece at the final position move it to its initial position and replace the piece at the final position with the given piece in the third position. If the piece didn't take any other piece in its initial move, an empty piece would be placed there.

Question 3: Variations on chess abound. For example, four-handed chess is a variant that is played by four players (search for it!). Outline the changes that would be necessary to make your program into a four-handed chess game.

To implement four person chess we would create two more players, two more sets of pieces, and expand the grid. Then in setup we would give the players a choice to be on teams or play alone. In addition, we would include code that would categorize the nine coordinates at the corners to be out of bounds in order to follow the true grid of four person chess. The we would write some code to end the game once both kings are in check for team play, or all kings but one for single player. In addition, we would have to re-code a lot of the functions that involve making sure a king is or is not in check because of the added kings.