

chess game

By Amr Mohamed & Michael Raafat



December 24, 2015

csed 19

Faculty of Engineering

Welcome to our project report. we are proud to present this game to you. We hope you will enjoy playing it ☺ .

**Description of our application and the features it provides:**

* Human vs human game mode. As our application allows multiple game modes, one of them allows you to play against your friends in a tensed chess game.
* Human vs computer game mode. The other mode allowed in our game which allows you to train against an easy programmed computer, allowing you to test your strategies and plans against a reactionary based player.
* We are proud to announce our game can support Castling and threefold repetition as addition for the normal checkmate, stalemate & promotion. Allowing a more complex game experience than you have ever played.
* You can undo and redo at any time you or your friend want! Because we know just like in real life games, sometimes you can both agree about redoing a move that was a mistake. This feature is available at both game modes.
* You can save the game in both game modes at any time you want with exiting the game. In case you want to overwrite a saved game, now you can! Naming a file with an existing saved file name will prompt you if you want to overwrite. Also you have the ability to save as many files as you want.
* NOW, you can manage your saved games. Either by loading them or deleting them, it is all possible due to our new user interface. Allowing you to load the games you want at both game modes or even delete them! And when loading, you will be able to undo or redo as you want.
* An impressive user interface that has many options like tutorial about using our game to loading and deleting saved games to even exiting the game.

**Overview of our design:**

At the start of this project, our top priority was to find a way to transform the chess game to its main logic components which together will compose our game. So, we began decomposing the concept of chess into the simplest concepts. Starting with functions that will be responsible of the board we will play on to the functions which decide which square is black or white.

Then getting to the functions that will check on basic things like if this square is empty or does it have a black piece on it. Is the column the user inserted even there?! Can this pawn be promoted or not? These functions were like the base of the building or in this case the game for us.

As we went to more complex functions, the functions that’s responsible of calculating every possible for every piece. Then, it was reality meets predictions so we made functions that transform user inputs to our board coordinates and suitable format and other functions that compare what the user inputs with our predicted moves and valid places. Adding all of these functions gave us the main turn functions, so know we had functions that are like the turns the players play in real life.

When does the game end? That was now what we thought which led to making functions for the win/draw conditions. Noticing that stalemate and checkmate have a lot in common, we made one function with the two which added to other functions gave us our end game conditions.

We had our game engine complete and ready to run, now to make the interface for the player was quite easy as we had everything. We began working on the save/load & undo/redo logic. We dealt with that by using arrays like book logs that tell us the states of things at any given turn and we implemented those log books at the code of the game, the save/load was then easier because we identified everything we needed to know to play. Allowing us to easily save them. And we made a simple interface as our main algorithm that from it, we choose which functions to call, and these functions while working will call sub functions like we already explained.

We then began on implementing more features while debugging current ones, allowing us to add castling, threefold repetition, primitive computer play game mode & improving undo/redo and save/load. Then adding deleting and improving our menu system and game interfaces and fixing more bugs!

**Description of some functions:**

* board functions :
* printall function receives the board . it check if there is dead pieces it prints them and then it prints the board
* boardhistorywrite function receives the board and copies it into another array that is like a book .
* blockboardreset function receives the board and the square. It reset the square either it’s black or white.
* Check functions :
* checkblack2 function which takes the board and a square coordinates and by comparing what is in the square to an array containing all the black pieces , and then it tells us if it does have a black piece or not.
* promoteblack function takes the board and a square coordinates and tells us if it can be promoted in the next move.
* checkrow function checks if the input user for the row is valid or not.
* whiteverify function checks if the input for the promotion is right or wrong.
* Input transforming functions:
* transformCoord function transforms the user input which is char to suitable format which is integer values .
* Movements functions:
* whitequeen function receives current place and board and an empty array and then fills the array with all the possible moves for the white queen
* allpossiblewm function takes the board and then return an array with all the possible white moves.
* Win and Draw functions:
* checkingwhiteking function receives board and checks if the white king is in check.
* matewhite function receives board and checks if there any possible moves to do without putting the white king in check.
* Checking input with predicted functions:
* checkmovewhite function receives the board and the current place and the future place and checks if this move will put the white king in check.
* validplace function receives the board,current place,future place and possible move list .If the future move is on the list returns 1.
* Collective turn functions:
* turnblack function it’s responsible for receiving the orders from the user and doing it.
* File handling functions:
* savegame function is responsible for saving all the variables of a certain game in a file with a name taken from user input.
* loadgame function is responsible for loading all the variables of a certain game in a file with a name taken from user input.
* Game functions:
* gameengine is responsible of the main game flow.
* newgame is responsible for initializing a new game.
* resumegame is responsible for resuming a saved game.
* Main function :
* Will be responsible of the main user interface.

Note: Any black function has a white function and vice versa.

**User Guide**

* Main Interface:
* H for options and commands
* L to load your latest saved game
* N to start a new game
* D to delete a saved game!
* B to try our new mode(human vs Machine)
* E to close the game
* In Game options:
* The game commands are simple to move a piece you enter it's current place with it's future one like (A2A4).
* In case of promotion , you add a letter at the end to define the promotion and it's based on the player! Black player with capital and white player with small letters like (A7A8q).
* To undo , simply write un.
* To redo, simply write re.
* To save at any time ,simply write sv.
* To exit the game , write ex.