Project – Chess GUI with Engine  
Evaluation  
Harley Reeves-Martin

Contents

[5.0.0 Comparison against initial objectives: 4](#_Toc8768842)

[5.1.0 User Feedback: 7](#_Toc8768843)

[5.2.0 Analysis of user feedback: 8](#_Toc8768844)

[5.3.0 Suggested Improvements: 9](#_Toc8768845)

# References

No references in this document.

* 1. Comparison against initial objectives:

|  |  |  |  |
| --- | --- | --- | --- |
| **No:** | **Objective** | **Done?** | **Comments** |
|  | be able to generate pseudo legal moves for a pawn. | Yes | The pseudo legal moves algorithm for a pawn works coherently with a board. |
|  | … a knight. | Yes | N/A |
|  | … a bishop. | Yes | N/A |
|  | … a rook. | Yes | N/A |
|  | … a queen. | Yes | N/A |
|  | … a king. | Yes | N/A |
|  | be able to store and use a move object with the board & pieces. | Yes | I created a move structure so that move objects could be instantiated, the boards make move method takes an “sMove” object as an argument. |
|  | be able to derive a subset of legal moves from a set of pseudo legal moves. | Yes | This was implemented in a method inside of the board class. |
|  | be able to setup the standard position of a chess board. | Yes | This was implemented in the constructor of a board. |
|  | be able to make any legal move on a board. | Yes | I implemented a method that determined the set of legal moves from a set of pseudo legal moves inside of the board class. |
|  | be able to determine the state of any given board (i.e. whether someone has won, if it is a draw, etc). | Yes | This was implemented in the board class by a method that is run whenever a move is made. The user is informed of a state change through the GUI. |
|  | be able to generate a map of which tiles are attacked by which alliance for any given board. | Yes | This was implemented through the “cAttackMap” class. |
|  | be able to store and then analyse any board to determine whether the previous move was legal (a ghost board). | Yes | This was implemented through the “cGhostBoard” class. |
|  | be able to setup a multiplayer game over any local area network, given that the user has all necessary permissions to open the connection. | No | Two general classes “controller” and “client” were implemented which would allow |
|  | be able to add time constraints to games. | Yes | This was achieved through implementing a class “cClock”, the user can choose how much time they want each player to have. |
|  | be able to represent a board graphically. | Yes | This was achieved using a board GUI class, the user can change the colour of the tiles through the settings. |
|  | be able to represent a piece graphically. | Yes | This was achieved with the held piece class. |
|  | allow the user to input moves using a graphical user interface. | Yes | This was achieved using event handlers in the GUI class, the users’ input is validated and if valid a move is made. |
|  | be able to graphically display a move list containing the moves played in the current game. | Yes | This was achieved using a custom move list GUI class. The user can interact with the move list. |
|  | be able to graphically display any previous positions whilst in a game via interaction with the move list. | Yes | Event handlers in the custom move list GUI class allows the user to select which previous position to show. The board graphics are then updated to represent the selected board. |
|  | be able to graphically display a clock if one is being used in the game. | Yes | This is achieved using a clock GUI class. |
|  | be able to graphically show all legal moves for a selected piece in a given position. | Yes | If the settings indicate that the user wants to show legal moves, the update graphics method in the board GUI class shows all legal moves for the selected piece. |
|  | allow the user to start a new game at any time. | Yes | This can be done by clicking either the new game button or play button on the main interface form. |
|  | allow the user to flip the board so that black can also be shown on the bottom of the board. | Yes | This can be done using the flip board button on the main form interface. |
|  | allow the user to undo a move (if they have enabled cheats in their preferences). | Yes | This can be done using the undo move button on the main form, only if cheats are enabled. If the user disables cheats, then the button is removed from the form. |
|  | allow the user to show an evaluation of the position in a message box (if they have enabled cheats). | Yes | This can be done using the show evaluation button on the main form if enabled. |
|  | allow the user to export an already played game in PGN. | Yes | The user can export any game played in my application to a PGN file, they can choose where the file is saved. |
|  | allow the user to export a position in FEN. | Yes | The user can export any game played in my application to a FEN file, they can choose where the file is saved. |
|  | allow the user to import and analyse an already played game from PGN. | Yes | The user can import and PGN file if it is valid. They can analyse the game by themselves or use the analysis functions provided in my application. In extremely complicated positions, my evaluation function may be inaccurate. |
|  | allow the user to import a position from FEN. | Yes | The user can import any position from a FEN string when creating a game. |
|  | allow the user to select and store settings for the application, storing preferences for things such as: tile colour, enabling cheats, etc. | Yes | The user can choose settings for the interface of my application using the settings form. |
|  | be able to traverse the game tree to any given depth. | Yes-No | The minmax algorithm I implemented can traverse the game tree to a maximum depth of 5. Due to the time complexity of the algorithm, it is impractical to allow the user to select a depth greater than 5. |
|  | be able to determine the best move in any given position. | Yes-No | The evaluation functions and minmax algorithm allows the engine to generate a reasonable move for most positions. However extremely complicated positions can prove too much for the engine to analyse, and the move will not be reasonable. |
|  | be able to determine which player has the advantage in any given position. | Yes | This is done by my evaluation function. |
|  | be able to play audio files (of .wav format) from within the program. | Yes | This was already achievable using a built-in function. |
|  | be able to setup an account. | Yes | Creating an account class and methods, the user is now able to setup an account and save it as a file. |
|  | be able to store a record and ELO rating associated with an account. | Yes | This is contained within the account class. |
|  | be able to store an account in its own file (“.acc” extension). | Yes | This is handled by the account class. |
|  | be able to store a table of accounts (“.dsv” extension). | Yes | The table of accounts is generated using the account classes shared methods. Whenever an account is created it is added to the accounts table. |
|  | be able to allow the user to login to their account. | Yes | This was achieved by being able to read account files back into account objects. |
|  | be able to secure the account using a username/password system. | Yes | This was done using the SHA-256 hashing algorithm, which created a username/password hash pair to be used. |
|  | be able to hash a password to ensure that the users’ data is secure. | Yes | Using the SHA-256 hashing algorithm. |
|  | be able to obfuscate an account file in order to preserve the integrity of the data. | Yes | This was achieved using the serializable attribute of the class, as well as using a binary file to store the account. |
|  | be able to obfuscate an account file in order to hide the password hash associated with their account. | Yes | This was achieved using the serializable attribute of the class, as well as using a binary file to store the account. |
|  | be able to show when a user is logged on, and appropriately update attributes as the user plays. | Yes | This was achieved by using the account window GUI class. Updates to the account record are made in the game class. |

* 1. User Feedback:

I sent my finished program to Mr Haworth, the head of the Chess club at my school, who used and tested my project with the chess club. I was provided the following feedback through email.

I also spoke with peers of mine, verbal feedback included statements such as:

1. Create a leader board to store which account is the best.
2. Constantly make backups of the board so that if the application is accidently closed it can be restored.
3. Add more game variants such as Crazyhouse.
4. Implement LAN play.



* 1. Analysis of user feedback:

The main client of my project sent feedback through email after given time to test my project with the chess club.

The overall feedback was mainly positive, the students of the chess club said that the program was well designed and said that they were impressed with the interactivity and responsiveness of my solution. This shows that my program is interesting, a goal I had when designing the project.

Furthermore, the feedback indicated that even weaker players of the chess club found the program easy to use, more so than a physical chess board. This indicates that the project is successful in offering a replacement to a physical chess set.

The account system also proved to be an important implementation, since it was stated that it provided the users with an easy to way to store their records, making the game more competitive and therefore more enjoyable.

I believe that the feedback indicates that my project was successful in drawing in newer chess players and helping them learn how to play the game, which was the main goal of this project. Furthermore, most of the players believed that the engine was strong enough (except for some users, which I will discuss in the next section).

5.3.0 Suggested Improvements:

Whilst my project met most of the originally proposed general and specific objectives, it did fall short in certain aspects.

Firstly, due to the time complexity of the algorithm used to search the game tree (Minimax), it was only realistic for the engine to search at a max depth of 5. This led to my engine being not as strong as I would have liked it to be, and this was reflected in provided user feedback, wherein some members of the chess club stated that the engine took too long to generate a move for higher depths, and some users believed that the engine was too easy to beat (after further discussion with the user, it was determined that those who made the second comment, could be considered advanced players. Therefore, I am not disappointed with this aspect). In order to improve the project, I could either implement a different tree search algorithm that would run faster, such as the Monte-Carlo tree search, or I could implement alpha-beta pruning, and extension of the Minmax algorithm that reduces the size of the game tree so that the user can search to a higher depth without increasing computation time.

The second shortcoming was the lack of an online feature allowing players to play across a local area network. Whilst I was able to implement the general classes that would allow me to enable the users to play across a LAN, due to time constraints, I was not able to implement it into the game class allowing the player to set the game up. Despite this, I plan to implement the feature before it is used by the chess club at my school.

An extension to my previous improvement would be furthering the online functionality of the program to support play over the world wide web rather than just on local area networks. This would undoubtably make the program more useful, since players would not have to always be on the same network. Furthermore, some networks (such as my schools network) do not allow the users to open a socket, and therefore the LAN functionality will be useless regardless, being able to connect to an external server would remedy this however.

Furthermore, it was suggested that I should add more variations, this means that my project would be suitable for a chess club, who may want to play more than just regular chess.

The final improvement I would have made is increasing the use of the accounts in the project. I would have done this by creating the leader-board as suggested by a user, as I believe this would have given users more of an incentive to create an account and play games on the application rather than on a board.