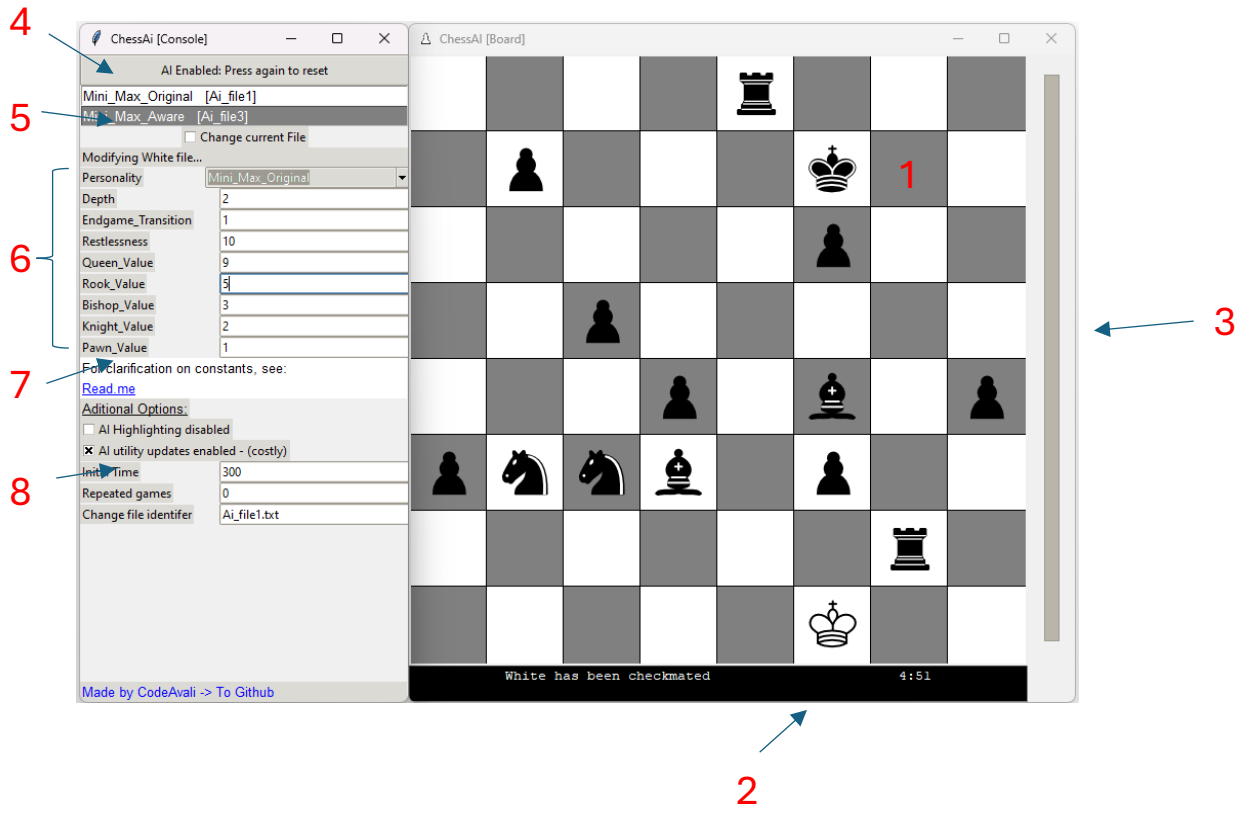
ChessAI Beta

This is an A Level Computer Science NEA focused on building a chess engine and using it as a medium to allow users to interact with/against an implementation of a chess bot, with the specific 'workings' of the AI player/algorithm able to be changed based on choose-able weights.

GUI Overview



Components

1. Board
2. Display for Time & Active Turn
3. Utility Bar
4. AI Enabled/Pause/Reset button
5. Personality Tag & Changing Active File
6. AI Weights
7. Read.me reference
8. Additional Options

AI Weightings

These are the values that can be changed in the ChessAI engine to change the playstyle/preference logic of the AI players.

'Personality' – Selectable Algorithms

To select a Player as being controlled by a Human Player – simply select the 'Human' Option.

Otherwise, the following algorithms are available:

- **Random_Pick** – Selects a Random legal move to perform.
- **Order_Pick** – Selects 'best' move according to the MVV-LVA heuristic.
- **Mini_Max_Original** – The 'classic' Mini_Max algorithm, using piece_square values.
- **Mini_Max_Optimised** – Improved 'loseless' MiniMax algorithm, using AB pruning and Move Ordering.
- **Mini_Max_Iterative** – Uses Iterative Deepening, to provide 'lossy' performance improvements.

Simply select one of these options to assign that player to be that respective algorithm.

Additional Weightings

The following additional weightings are available:

- **Depth** – The complexity that the algorithm is allowed to reach. For the Mini_Max algorithms; this is the maximum amount of moves from the current state the Algorithm is allowed to explore. Please note that at increasing Depth, AI algorithms begin to recognise beneficial behaviour (such as not hanging pieces at depth=2) – but this has a significant computational cost and WILL slow down execution.
  Recommended values, 1-4. Higher Depth WILL lead to slow/no execution.
- **Endgame_Transition** – PeSTO's evaluation function includes piece square values for the midgame and the endgame, the value of this weight changes how much impact each table has on the overall evaluation. For example, at 1; the evaluation will move 1% towards the end game evaluation tables per executed turn.
- **Restlessness** – The amount of 'non-progressing' turns before the evaluation function has a random factor of 1 point of utility for each possible (root) move. Note that any 'progressing' move is any capture, or pawn movement. Significantly, this random factor is modelled exponentially, to eventually pick a non-ideal move.
- **Queen_Value** – Utility value associated with a Queen.
- **Rook_Value** – Utility value associated with a Rook.
- **Bishop_Value** – Utility value associated with a Bishop.
- **Knight_Value** – Utility value associated with a Knight.
- **Pawn_Value** – Utility value associated with a Pawn.

The initial values for pieces are **9, 5, 3, 3, 1** to match the conventional relative values.

Additional Options

The following Additional Options are Available:

- **AI highlighting (Dynamic thoughts)** – By enabling this option, the current 'best processed' move will be visually displayed on the board, akin to how a human will make a move using the GUI.
- **AI utility updates (Dynamic utility)** – After each processed 'possible move', whilst the option is enabled, the utility bar is updated with the average expected utility for the AI algorithm.
- **Initial Time** – When starting a game, this value will be used as the initial time for both players.
- **Repeated Games** – If you wish to automatically repeat a number of games, input the number of times into the entry-box. Once a game finishes, it will repeat and decrement the number till it reaches 0.
- **Change File Identifier** – You can change the identifier to another text-file containing the respective weights. If the file is valid, then it will automatically be loaded in and displayed in the personality tag/entry boxes.

Creating your own AI Algorithm / Personality

Write your algorithm within algorithms/Ai_Created_1 (or Ai_Created_2).

Every required variable will be given as part of the parameters, although you may choose to disregard any.

Please note the following pre-made methods:

- **main.ai_perform(W_Moves, B_Moves, move_from, move_to, temp)**
- **main.peice_square_optimised(board, W_Move, B_Move)**
- **main.king_attacked(Opposing_moves, board)**
- **main.lazy_pin(board, White_Moves, Black_Moves, for_White)**
- **main.OrderMoves(Moves, board, for_White)**

You should return your selected move for your initial call, which will end the algorithm's execution.

From there, select Ai_Created_1 or Ai_Created_2 to run your algorithm once programmed.

Credits

Created by CodeAvali for the OCR A Level Computer Science NEA Project, developed between 2023-2024.

If there are any issues/questions, please go see GitHub Issues.