Chess Engine ‘Gambit’ Development

Aims:

To understand standard AI techniques and algorithms and how they can be applied or adapted to play chess. Experiment with and contribute unexplored ideas for chess engine development.

Project Description:

Develop a chess engine that takes advantage of standard chess programming principles and practices, whilst developing a new move decision algorithm and catering for a better engine vs human experience by making more exciting moves. The new move decision algorithm should account for its current position (whether it is winning or losing) and whether the opponent is strong or weak. Do this using a low level language such as C or C++.

Background:

Often a strategy adopted by humans in games when losing or against a stronger opponent would be taking more risks and effectively gambling. After all, if you’re losing, you can’t keep doing the same things. You must take a risk and hope for an oversight or blunder which puts you in a winning position. An example of this in chess are gambits. These often are not standard theory, and if manoeuvred well by your opponent can often result in a worse position for the player who played the gambit. However, when the gambit succeeds, the player gets a very strong position.

Early Deliverables

1. Be able to represent a board state given any FEN
2. Be able to print a board to the terminal
3. Be able to generate all possible moves for any given position with efficient legality checks
4. Be able to test these move generations using perft
5. Produce a report detailing methods used and include the key concepts used to develop the engine to its current state, such as explaining perft, the concept of bitboards and bit shifting, and move generation algorithms and optimisations such as lookup and transposition tables.

Late Deliverables

1. Be able to search through generated moves and decide which one to play
2. Be able to update board representation after a move has been played
3. Be able to interact with a UI, whether a pre-existing one like cutechess-gui or a custom UI
4. Enable hosting on Lichess.com

5. Produce a report including SPRTs for each new feature and compare it to other engines to see if these ideas have promise. Can compare to other engines using cute chess. In the report, highlight example moves where the new approach has taken place. For example, the engine knows move x is best, but using the new algorithm, it decides on move y, and why it has decided to choose that move, for example it calculated that it is losing, and it has met the threshold to play a "bold" move.

Suggested Extensions:

- Most chess players have an Elo assigned by sites like Chess.com. However, should they not have that to hand, develop a means of evaluating their Elo
- Provide an opening book. Perhaps with well-known gambit lines and/or regular openings.
- Custom UI for interacting with Gambit
- Two modes, standard which uses the Gambit approach, and the other uses the standard alpha beta approach.

Reading and Resources:

Chess Engine Programming Standards

- Standard Chess Programming Principles and Practices: https://www.chessprogramming.org/Main_Page
- The UCI Protocol: https://backscattering.de/chess/uci/

Forums

- Talk Chess A forum for chess talk, including engine programming: https://talkchess.com/ https://discord.gg/mJuKQbrEmN
- Discord Server for Chess Engine Development: https://discord.gg/uTDs7U5Vh7

Engine Evaluation:

- Comparing Strength with Other Engines: https://www.chessprogramming.org/Cutechess-cli
- Engine used to determine strength: https://www.chessprogramming.org/Stash
- Engine Rating List: https://computerchess.org.uk/ccrl/4040
- Open Source Engine High Standard Engine: https://github.com/AndyGrant/Ethereal
- Open Bench for Evaluation: https://github.com/AndyGrant/OpenBench

Perft

- Move generation accuracy is verified using known numbers of legal moves from a given position.
- Information: https://www.chessprogramming.org/Perft
- Online Perft Interface: https://analog-hors.github.io/webperft/
- More Info on Test Results: http://www.rocechess.ch/perft.html

General Resources

- Pre-existing Open Source Engines: https://github.com/EngineProgramming/engine-list
- A Survey of Monte Carlo Tree Search Methods https://www.researchgate.net/publication/235985858_A_Survey_of_Monte_Carlo_Tree_Search_Methods
- Opening Books: https://github.com/official-stockfish/books