How to Build a Chess Engine

A Project presented to

Mr. Dwight Strong

Computer Information Systems

Pittsburg State University

Pittsburg, Kansas

---

by

Lucas de Sa Lopes Palma

April 26th, 2024

Table of Contents

Introduction to Chess Engines

The first chapter of this research will provide an understanding about the history of chess, current chess engines, and goals of this project. This project is a personal achievement for me since I am a big fan of the game and play regularly. I intend to describe the necessary process in order to achieve a functional chess engine.

Chess is a millennium board game which has improved throughout centuries. Its origins can be tracked to ancient India in the 6^{th} century. Moreover, its rules were modified until the current ones we have nowadays. Chess is a fascinating game which involves strategies and tactical moves which has been updated from generation to generation.

The rise of chess engines started in 1997. Garry Kasparov, the world chess champion, was defeated by IBM's Deep Blue engine. It was a mark for AI development since it made possible for computers to defeat the experience and intuition of a legendary player who is recognized until today. From this point forward, chess engines were being improved and defined unbeatable to humans. There are several popular ones currently, such as, Stockfish and Alpha Zero. Those powerful chess engines now compete in chess engine competitions to define the best algorithm between all of them.

The goal of this project is to provide a guide on how to build a chess engine. The idea of developing this algorithm seems complicated at first, but another aspect of this research is to facilitate the understanding of how those algorithms are going to work and their purpose. This project aims to explain from how can you play chess to how is a chess engine built.

Basics of Chess

Initially, we need to understand how a chess game works. It is crucial to learn the chess rules so we can build an accurate chess engine. The game of chess has very simple rules that haven't changed much over the past decades. The fundamental guidelines are listed below:

The Chessboard: The game is played on a square board with 64 squares that are arranged in an 8 by 8 grid and alternate between the colors black and white. Those squares also have coordinates. By looking from white's perspective, columns are identified from "a" to "h" (left to right), and rows are identified from 1 to 8 (from bottom to top).

The Pieces: Two rooks, two knights, two bishops, one king, one queen, and eight pawns are the 16 pieces that each player begins with.

Piece Movements: Every kind of piece has an own style of moving:

1. The king can travel in any direction for one square.

2. Any number of squares in any direction—horizontally, vertically, or diagonally—can be moved by the queen.

3. Any number of squares can be moved vertically or horizontally by rooks.

4. With two squares moving in one direction and one square moving at a 90-degree angle, knights move in an L-shape.

5. Bishops can make diagonal moves of any number of squares.

6. Pawns advance one square, although they are captured diagonally. They can choose to advance two squares forward on their first turn.

Capturing: An opponent's piece is taken off the board and captured when a piece advances to a square that it now occupies.

Check and Checkmate: A player must take a move to neutralize the threat if their king is being attacked (in "check"). "Checkmate" happens when the opponent's king cannot escape from a "check", and the player wins the game.

Special Moves: There are a few special moves in chess:

Castling: The king and one rook can make a special move called castling, where the king moves two squares towards the rook, and the rook moves to the square next to the king. Castling is subject to certain conditions and is used to improve the king's safety and activate the rook.

En passant: If a pawn moves two squares forward from its starting position and lands beside an opponent's pawn, the opponent has the option to capture the first pawn "en passant," as

if it had only moved one square forward. The pawn is now located one square behind the pawn that moved two squares.

Promotion: If a pawn successfully advances to the opponent's back rank (last row), it can be promoted to any other piece (except a king).

Data Structures and Board Representation

A chess engine needs a board representation to keep track of where the pieces are positioned. It is also useful to understand what are the possible moves in a certain state of the game. It is important to realize that the choice of data structure influences the accuracy and algorithm of the chess engine. Therefore, it is fundamental to analyze and make an assertive decision. In this paper, we are covering two types of board representation. These are piece centric, and square centric.

Piece Centric: Lists, arrays, or sets of every piece remaining on the board is a piece-centric representation, together with the corresponding information about which square each piece occupies. A strategy that is widely used that is piece-centric is the set-wise bitboard method. Each sort of component has a single 64-bit word whose one-bits correspond to its occupancy.

Square Centric: The inverse association is implemented via the square centric representation. The most used square-centric representations, mailboxes, or its 0x88-enhancements, are essentially arrays of direct piece-codes, which include out-of-board codes and the empty square. Piece-list entries may also be referred to in hybrid solutions.

Move Generation

This chapter focuses on inserting appropriate algorithms in the code for move generation. It is important to implement this section carefully because it will be used to calculate all possible moves in the board. Finding the best moves will be a challenge which highly depends on the accurate move generation in the chess board. The project will list below the constraints which have to be respected to accurately move the pieces.

Piece-specific Move Generation: Develop algorithms for each sort of chess piece. Those rules were clarified in chapter 3.

Move Validation: After generating probable moves, validate each one to ensure it does not put the player's own king in check. Validate moves to avoid illegal moves caused by pinned pieces, as well as moves that result in capturing the opponent's king.

Move Storage and Return: Save the created legal moves in a data structure, such as the board representations mentioned in chapter 3, so that the chess engine can evaluate and make decisions. Return the list of moves to the engine's main algorithm for evaluation.

Evaluation Function

The evaluation function is an essential part of a chess engine. It evaluates the current chess position and assigns a number value indicating the engine's assessment of the position's desirability. When creating an evaluation function, consider the following factors:

Piece Values: Determine the worth of each chess piece depending on its strength. Traditionally, pawns are worth one point, knights and bishops three points, rooks five points, and queens nine points. These parameters can be changed depending on the engine's evaluation technique.

Material Balance: Determine the material balance by comparing the overall value of the engine's and opponent's pieces. A positive balance indicates an advantage for the engine, whereas a negative balance shows a disadvantage.

Pawn structures: Consider issues like pawn islands (groups of pawns separated by empty files), doubled pawns, solitary pawns, and pawn majorities. In general, a strong pawn structure is better for a chess position, whereas flaws might be exploited by the opponent.

King Safety: Evaluate the engine's king for safety. Consider pawn cover in front of the king, open files near the king, and the existence or absence of the opposing assaulting pieces. A safer king position is often preferred.

Mobility: Evaluate the mobility of the engine's parts. Pieces with more free squares to move have greater mobility and can exert more effect on the board. Consider concerns such as centralization, open files, and key square management.

Development: Evaluate the development of the engine's components. Pieces that are actively developed and coordinated are more valuable. Consider aspects such as piece activity, coordination, and effective piece placement.

King dangers: Evaluate possible dangers to the opponent's king. Consider elements such as king attacks, open lines of attack, and weak squares surrounding the opponent's king. Exploiting these weaknesses can put you in a better position.

In summary, balancing the evaluation phrases is critical for providing an appropriate appraisal of the position. Assign weights to each evaluation factor based on its relative importance. Adjust these weights using testing and analysis to obtain the correct balance. Fine-tuning the evaluation function is a continual process that entails studying games, collecting feedback, and making incremental changes.

Search Algorithms

Chess engines rely heavily on search algorithms, which examine potential moves and judge current positions to help the user make judgments. The minimax algorithm with alpha-beta pruning is a common search strategy frequently utilized in chess engines. Here's a summary of these ideas:

Minimax method: This recursive method looks at every action that can be made and all of its consequences as it explores the game tree. It is assumed that both players perform as best as they can. To decide the optimum course of action for the engine at each level of the game tree, it alternates between maximizing and decreasing the points. Understanding that positive results are better for white and negative values are better for black.

Evaluation Function: To allocate scores to places, the minimax algorithm uses an evaluation function (explained in the previous chapter). An estimate of the engine's wish at a specific position is given by the evaluation function. Throughout the search process, the algorithm uses the scores to make decisions.

Alpha-Beta Pruning: As the search depth increases, the time complexity of the minimax method grows. The use of alpha-beta pruning increases efficiency. The game tree's branches that are defined unnecessary for the outcome are removed.

Two variables are maintained during alpha-beta pruning: alpha and beta. The lowest score that the maximizing player (engine) is guaranteed is represented by alpha, while the highest score that the minimizing player (opponent) is guaranteed is represented by beta. Therefore, positions that are certain to be worse than the current best move are not evaluated by the algorithm. Alpha-beta pruning dramatically lowers the number of places that must be investigated by removing unnecessary branches, making the search process quicker and more effective.

Conclusion

In summary, the paper demonstrated how to work with chess engines and develop your own. Chess engines require expertise by the programmer to become more powerful and effective. For that reason, having a base of how it works will kick off the beginning of a journey for developing an efficient chess machine capable of surpassing others.

The main purpose of this research is to serve as a base for developing chess algorithms. Understanding the dynamics of the game and algorithm are essential keys for creating a chess engine. The project believes that possessing the knowledge regarding board representation, move generation, evaluation functions, and search algorithms are key ingredients for being a successful programmer in this field.

For the future of this project, coding exemplifications and board visualization are substantial ingredients. I imagine the future of chess engines relying on powerful processors and complex algorithms. The Top Chess Engine Championship (TCEC) is a chess engine tournament which involves the most powerful chess engines of modern days. This kind of event serves as a perfect stage for these algorithms to keep improving and becoming more powerful.

References

Main page. Chessprogramming wiki. (n.d.). *https://www.chessprogramming.org/Main_Page*

The Fascinating Programming of a Chess Engine. (n.d.). Www.youtube.com. Retrieved April 27, 2024, from https://youtu.be/w4FFX_otR-4?si=fHe3DyQWm_QsxV3e

Coding Adventure: Chess. (n.d.). Www.youtube.com. Retrieved April 27, 2024, from https://youtu.be/U4ogK0MIzqk?si=1_jZDubECIrOMwOc