Main Website: <https://chessprogramming.wikispaces.com/Chinese+Chess>

Summarizing Articles:

http://www.fam-petzke.de/cp\_aod\_en.shtml

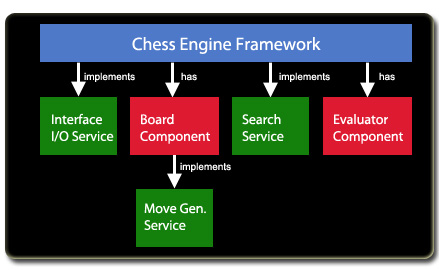
A chess engine consists basically out of 4 components

**Board** - This were the position of all pieces is stored, which side is on the move, castling rights, en passant square, but also a history of previous positions

**Move Generator** - This is the part of the engine responsible for generating all moves or special moves (captures only, moves that give check only) for a certain board position

**Evaluator** - The evaluator calculates a static score for a given board position. A score of 0 means equal chances for both sides. A high negative or positive score indicates a good position for either white or black. The score involves basic stuff like material but also positional factors like double pawns, king safety and passed pawns. The evaluator is responsible for the personality of the engine and influences its strategic or positional play.

**Search** - This is the algorithm that decides which moves look promising and are to be searched first and more thoroughly than other not promising looking ones. Search is responsible for the tactical strength of the engine. It is usually implemented as a recursive algorithm that calls itself to traverse the game tree up to a certain depth where then the evaluator is used to estimate a score for that position.



A series of articles: <http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/chess-programming-part-i-getting-started-r1014>

1. **Board Representation & Data Structure**
2. Bit Boards: represent the position of each piece on the board. 64 bits operation in chess – how about in Chinese chess?
3. Transposition Tables: a repository of past search results and well-known states that could be pre-programmed in; usually by hash table etc.
4. Generating Hash Keys for Transposition Tables
5. \* History Tables: a record of killer moves – increase their heuristic result
6. Pre-processing move generation: move generation is the most computation expensive step
7. **Movement Generation**
8. Selective generation: Examine the board, come up with a small number of "likely" moves and discard everything else. Not good is it’s not perfect!
9. Incremental generation: Generate a few moves, hoping that one will prove so good or so bad that search along the current line of play can be terminated before generating the others.
10. Complete generation: Generate all moves, hoping that the transposition table (discussed in Part II) will contain relevant information on one of them and that there will be no need to search anything at all. Need to order moves to speed up search.
11. **Basic Search**
12. Alpha-Beta Search Tree
13. Iterative Deepening Search
14. **Advanced Search**
15. **Evaluation**