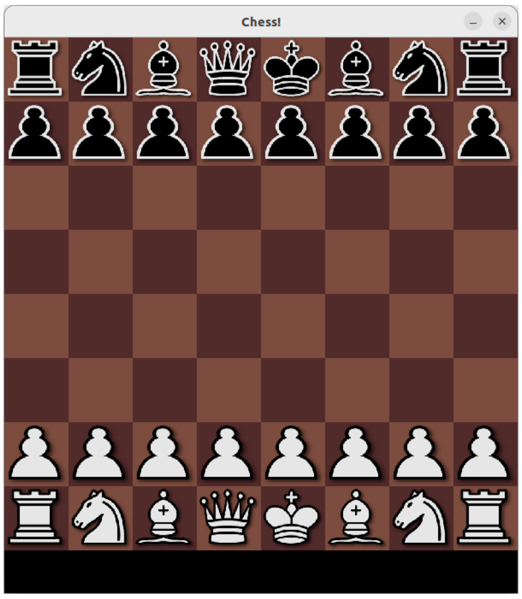
Fundamentals of Artificial Intelligence
Lab 3: AI Chess Player – the Minimax Algorithm

Perfect and Absolute Order

Once again, a new message from "LemML" strikes, marked as highly important. Of course it's an urgency, you think, it's their 24'th message in the last couple of days! However this one looks different, coming from the "VUAO" project which you've never heard of.

Reading through the lines, written in a charming and lucid manner, you learn of the task ahead of you. The letter asks you to provide the sender with a way of elegantly moving disk-like structures over a square grid. A previous company proposed them a game called "cangoorou", yet the client, while initially content with the jumpiness, deemed it not aesthetic enough.

You smile at the arrival of an obvious solution in your head – a chess engine! Exited you decide to rush the development, remembering all the tricks you've studied when you were still a freshman at university. And while the people at "VUAO" will think about the aesthetic value of a chess set and whether they could accept such changes to their perfect and absolute order, you'll send them a finished solution. Those *phools* won't see it coming!



General guidelines

- You can use the provided code with Python, or you can implement your own code with a user interface using a coding language of your choice. If you are using a web-based UI implementation, you should still **implement your code logic in the backend**. If you are deploying on *lichess.org*, you don't need to implement a custom user interface.
- **Plagiarism will not be tolerated.**

Grading policy

- **Task 1**: Implement the MiniMax algorithm with the following scoring function:

  $$Score = MaterialValue + PositionalValue$$

  For computing the *MaterialValue*, each piece is assigned a value (e.g., Pawn = 1, Knight = 3, Bishop = 3, Rook = 5, Queen = 9). Then you sum these values for your pieces and substract the value of the pieces of the oponent.

  For computing the *PositionalValue*, you should take into account the position of each pieces on the board (e.g the more squares a pawn has travelled, the higher their PositionalValue etc.). You should then substract the opponent's PositionalValue from your pieces' PositionalValue. ***(3p)***

- **Task 2**: Implement Alpha-Beta Prunning. ***(2p)***

- **Task 3**: Implement an improved scoring (evaluation) method for MiniMax. For example, you could add values like *KingSafetyValue*, *MobilityValue* (nr of legal moves to each side), *PawnStructureValue* (can include penalties for isolated pawns, doubled pawns, and bonuses for passed pawns or a strong pawn chain), etc. You can also use Heuristic Evaluation Functions. Be creative! ***(1p)***

- **Task 4**: Add at least one improvement to the MiniMax algorithm from the following list: Progressive Deepening, Transposition Tables, Opening Books, Move Ordering, Aspiration Window etc ***(1p)***

  Note: you can get bonuses for implementing more improvements. For the second improvement you get 0.5p, for 3rd you get 0.25p, for 4th - 0.125p and so on...

- **Task 4**: Deploy your solution on `licess.org`. You can follow this tutorial: `https://github.com/lichess-bot-devs/lichess-bot` ***(1p)***

- **Report & Presentation of the solution**:

  Explanations, report formatting, code quality, comments in the code, etc. ***(2p)***

Useful links

- `https://github.com/lichess-bot-devs/lichess-bot`
- `https://www.youtube.com/watch?v=_vqlIPDR2TU`
- `https://saturncloud.io/blog/improving-minimax-algorithm-a-guide-for-data-scientist`
- `https://www.chessprogramming.org/Aspiration_Windows`

Good Luck!