



(Vecteezy, n.d.)

Herm0ni Chess Bot

Design Document 2024-2025

**Student Name:** Seán Rourke

**Student Number:** C00251168

**Supervisor:** Joseph Kehoe

Contents

[1. Introduction 3](#_Toc182678266)

[1.1. Document Purpose 3](#_Toc182678267)

[2. System Architecture 3](#_Toc182678268)

[3. Sequence Diagrams 3](#_Toc182678269)

[4. Technologies 3](#_Toc182678270)

[4.1. C++ 3](#_Toc182678271)

[4.2. Lichess Bot API 3](#_Toc182678272)

[4.3. Hosting Platform 3](#_Toc182678273)

[5. Data Structures 3](#_Toc182678274)

[5.1. Bitboards 3](#_Toc182678275)

[6. Algorithms 3](#_Toc182678276)

[6.1. Heuristic Algorithm 3](#_Toc182678277)

[6.2. Minimax with Alpha-Beta Pruning 4](#_Toc182678278)

[7. References 4](#_Toc182678279)

# Introduction

Herm0n1 is an artificial intelligence (AI) chess bot aimed at being played by players for practice or fun, and other chess bots to determine comparative strength. The bot is designed to evaluate chess positions and determine the next move to make without the need for human input.

## Document Purpose

The purpose of this document is to detail the design and structure of the Herm0ni bot and its implementation on lichess.org. Interactions between different parts of the system will be illustrated using sequence diagrams. This document will contain details of the various technologies used to create the bot and the reasoning behind using those technologies. The data structures used by the bot will be explained and visualised. This document will also contain details of the various complex algorithms that the bot uses to play chess.

# System Architecture

# Sequence Diagrams

# Technologies

## C++

The code for the bot itself will be written in C++. The high speed of C++ is beneficial for efficient computation of complex algorithms. This is especially crucial in short chess games, such as blitz or bullet games. Being able to perform these algorithms quickly ensures that the skill level of the bot doesn’t take a hit when the time it it given to make each move is reduced.

## Lichess Bot API

The lichess bot API allows the bot to interact with the lichess.org website. The API allows a bot account on lichess.org to be controlled by the engine. It can send commands to the engine and process responses. The API is written in Python and messages sent to the engine are in JSON. Information about moves made by the player is presented in universal chess interface (UCI).

Having the bot implemented on lichess.org means that time and effort does not need to be spent developing a graphical user interface (GUI) for the games to be played. This means more time can be spent optimising the engine’s performance for improved chess ability.

## Hosting Platform

# Data Structures

## Bitboards

The position of a game is stores using bitboards. These are implemented using 64-bit integers, with each piece for each colour having their own bitboard. A 64-bit integer is used as each bit can represent a square of the chess board. A 1 bit in the integer represents the presence of a piece on that square. For example, at the beginning of the game, the white pawns take up eight squares along the second rank. For example, using a 64-bit integer, the white pawns are initialised as seen below.

Bitboard whitePawns = 0x000000000000FF00;

## Minimax Tree

# Algorithms

## Heuristic Algorithm

The heuristic algorithm is used to evaluate a position, determining who is in a more advantageous position, white or black, or whether the position is equal. An advantage for white will be represented with a positive number, an advantage for black will be represented with a negative number, an evaluation of 0 means that the position is equal. This algorithm is based on many factors. All evaluation functions will be run once, with values being added to the evaluation for the white pieces and subtracted from the evaluation for the black pieces. This is more efficient than running each function separately for the white and black pieces and then subtracting the black evaluation from the white.

### Heuristic Function:

function EvaluatePosition(board)

score = 0

for each piece in board

if white.owner == "white"

score += EvaluatePiece(board, piece)

else if piece.owner == "black"

score -= EvaluatePiece(board, piece)

end if

end for

return score

end function

function EvaluatePiece(board, piece)

score = 0

score += MaterialValue(piece)

score += PieceActivity(board, piece)

if piece.type == "king"

score += KingSafety(board, piece)

else if piece.type == "pawn"

score += PawnStructure(board, piece)

end if

score += ControlOfCenter(board, piece)

score += Development(board, piece)

score += SpaceContribution(board, piece)

score += Coordination(board, piece)

score += Mobility(board, piece)

return score

end function

The most basic evaluation factor material count. Each piece is given a value:

* Pawn: 1
* Knight: 3
* Bishop: 3
* Rook: 5
* Queen: 9
* King: 0 (not directly scored)

**Material Count Funtion**

function MaterialBalance(board)

score = 0

for each piece in board

if piece.type == "pawn"

value = 1

else if piece.type == "knight" or piece.type == "bishop"

value = 3

else if piece.type == "rook"

value = 5

else if piece.type == "queen"

value = 9

else

value = 0

if piece.owner == "white"

score += value

else if piece.owner == "black"

score -= value

end if

end for

return score

end function

As the game develops these values will be adjusted dynamically, for example, a rook on an open file or a bishop on an open diagonal will be more valuable than if they are trapped by other pieces or pinned to a more valuable piece. Similarly, most pieces are more valuable if they are in the middle of the board, especially if it is an open position or many of the pawns have been captured already.

## Minimax with Alpha-Beta Pruning

The minimax algorithm is used for the calculation of the next move. From a position, all possible moves are considered, then the possible responses to this move are considered. The position after these moves is evaluated to determine the best move to make.

# References

Vecteezy, n.d. *Vecteezy.* [Online]   
Available at: https://www.vecteezy.com/free-vector/chess-silhouette  
[Accessed 26 November 2024].