DHBW Stuttgart

Chess-Engines in Rust

Von Albert Eisfeld und Lukas Piorek

T3INF4901 Wahlfach Programmieren mit Rust 2025

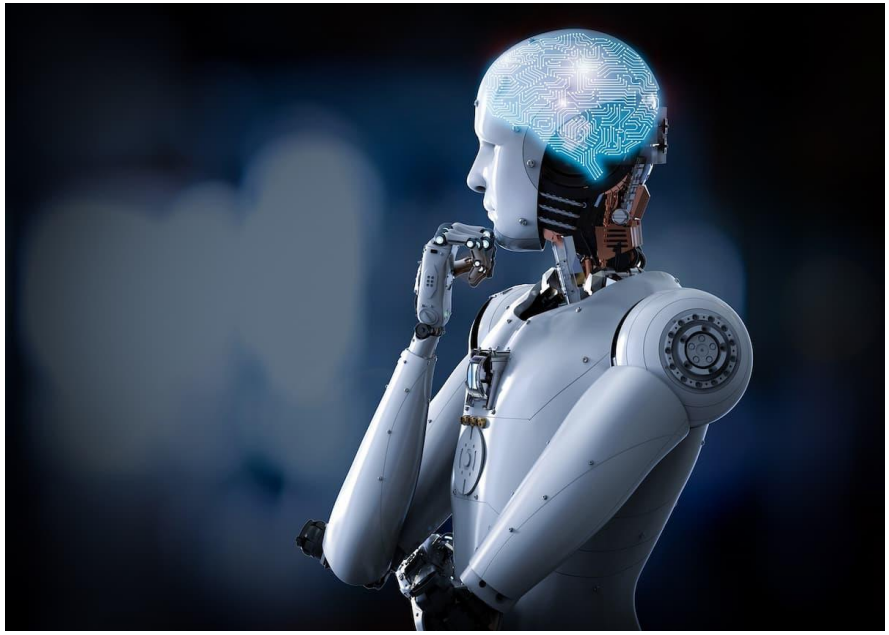
19.05.2025, Stuttgart

Gliederung

- Chess Engines Allgemein
- Adam (Engine von Albert)
- Thunfisch (Engine von Lukas)
- Ausblick
- Duell

Chess Engines

- Analysiert Positionen
- Generiert Züge
- Bewertet Züge
- Wählt Zug aus
- Keine GUI
- Kommunikation via UCI



Chess Engines vs Menschen

- Bester Mensch: Magnus Carlsen (2882 Elo)
- Bester Engine: Stockfish (3643 Elo)
- Mittlerweile besser als jeder Mensch

Kommunikation

>thunfisch(0): position startpos

>thunfisch(1): ucinewgame

>thunfisch(1): position startpos

>thunfisch(0): isready

<thunfisch(0): readyok

>thunfisch(0): go wtime 5000 btime 5000 movestogo 40

<thunfisch(0): info depth 1 seldepth 1 score cp 144 nodes 20 nps 392 time 50 tt 0 pv g1f3

<thunfisch(0): info depth 2 seldepth 2 score cp 0 nodes 428 nps 2114624 time 0 tt 0 pv g1f3 g8f6

<thunfisch(0): info depth 3 seldepth 5 score cp 126 nodes 3028 nps 12770982 time 0 tt 0 pv g1f3 g8f6 b1c3

<thunfisch(0): info depth 4 seldepth 6 score cp 0 nodes 58150 nps 32968590 time 1 tt 0 pv g1f3 g8f6 b1c3 b8c6

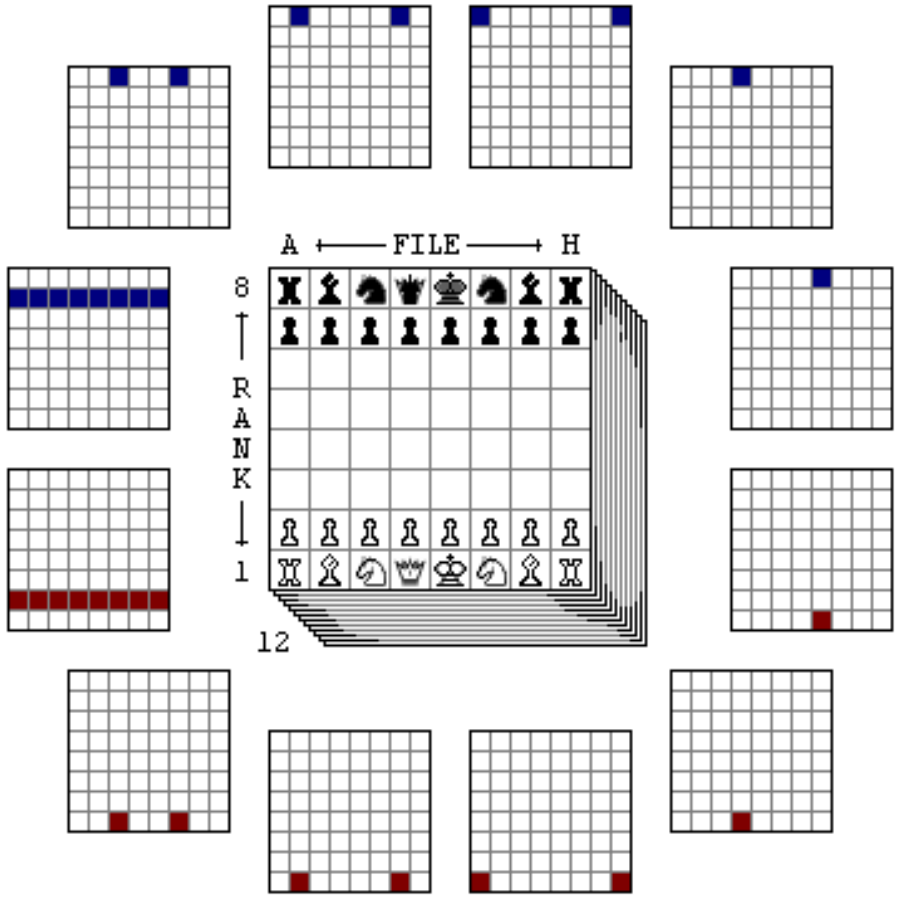
<thunfisch(0): info depth 5 seldepth 9 score cp 10 nodes 374696 nps 33233640 time 11 tt 0 pv g1f3 g8f6 b1c3 b8c6 a1b1

<thunfisch(0): info depth 6 seldepth 14 score cp 10 nodes 1287826 nps 40050816 time 32 tt 0 pv g1f3 g8f6 b1c3 b8c6 a1b1 a8b8

<thunfisch(0): bestmove g1f3

>thunfisch(1): position startpos moves g1f3

Repräsentation des Schachbretts: Bitboards



Adam

- Nutzt UCI Protokoll
- Evaluation über piece square tables
- Zugauswahl durch minimax Algorithmus
- Intern: Moves, Chessboards, etc. als unsigned integer
  > Learning: Struktur **vorher** überlegen spart Arbeit

Evaluieren

Piece-Square Tables

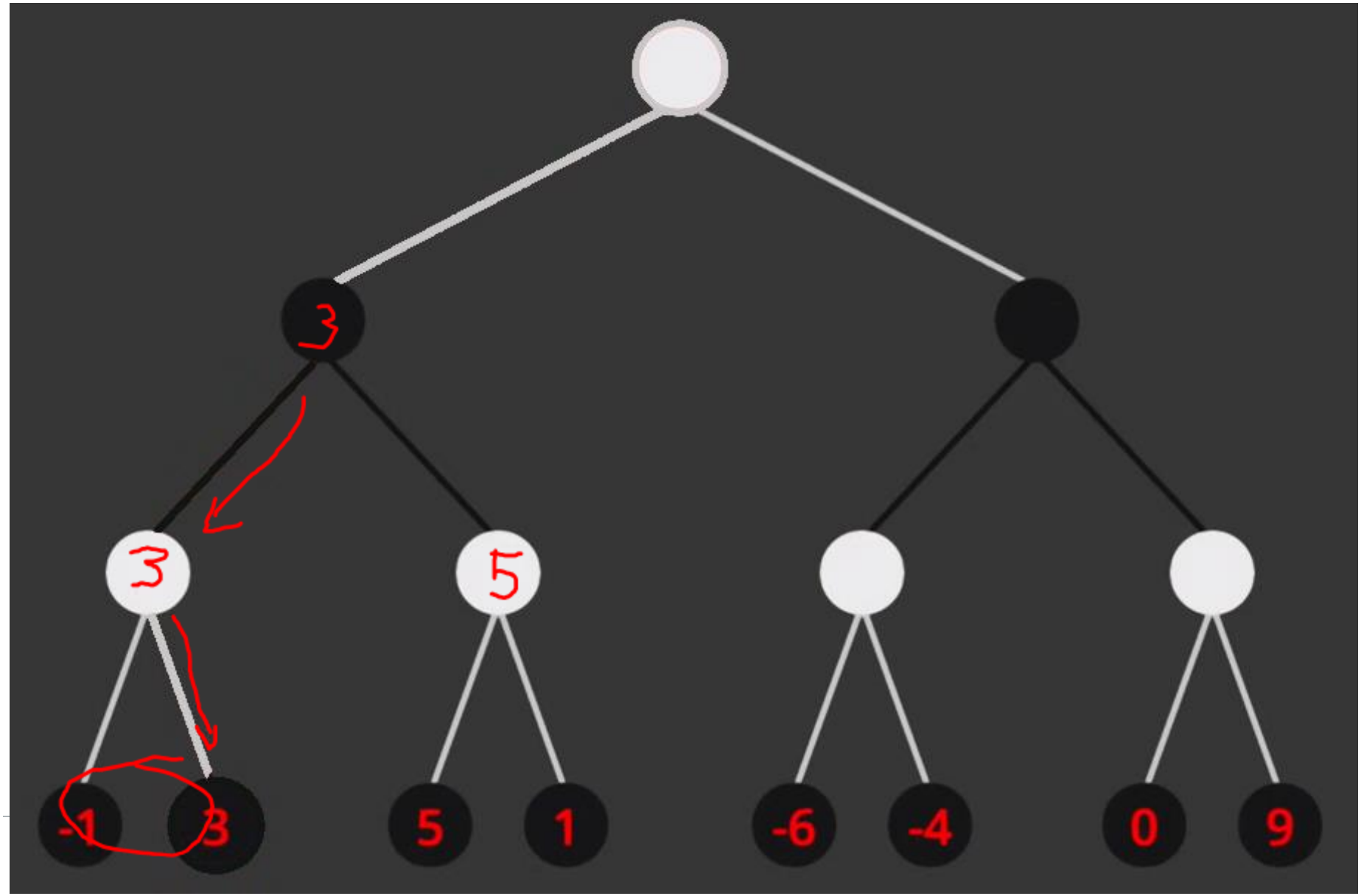
Pawn ♟						Bishop ♗						Knight ♘					
0	0	0	0	0	0	-4	-2	-2	-2	-2	-4	-10	-8	-6	-6	-8	-10
10	10	10	10	10	10	-2	0	0	0	0	-2	-8	-4	0	0	-4	-8
0	2	4	4	2	0	-2	1	2	2	1	-2	-6	0	4	4	0	-6
1	0	6	6	0	1	-2	2	2	2	2	-2	-6	1	4	4	1	-6
1	10	-4	-4	10	1	-2	1	0	0	1	-2	-8	-4	1	1	-4	-8
0	0	0	0	0	0	-4	-2	-2	-2	-2	-4	-10	-8	-6	-6	-8	-10
Rook ♖						**Queen ♕**						**King ♔**					
0	0	0	0	0	0	-4	-2	-1	-1	-2	-4	-6	-8	-10	-10	-8	-6
1	2	2	2	2	1	-2	0	0	0	0	-2	-6	-8	-10	-10	-8	-6
-1	0	0	0	0	-1	-1	0	1	1	0	-1	-4	-6	-8	-8	-6	-5
-1	0	0	0	0	-1	-1	1	1	1	0	-1	-2	-4	-4	-4	-4	-2
-1	0	0	0	0	-1	-2	0	1	0	0	-2	1	1	0	0	1	1
0	0	1	1	0	0	-4	-2	-1	-1	-2	-4	4	6	2	2	6	4

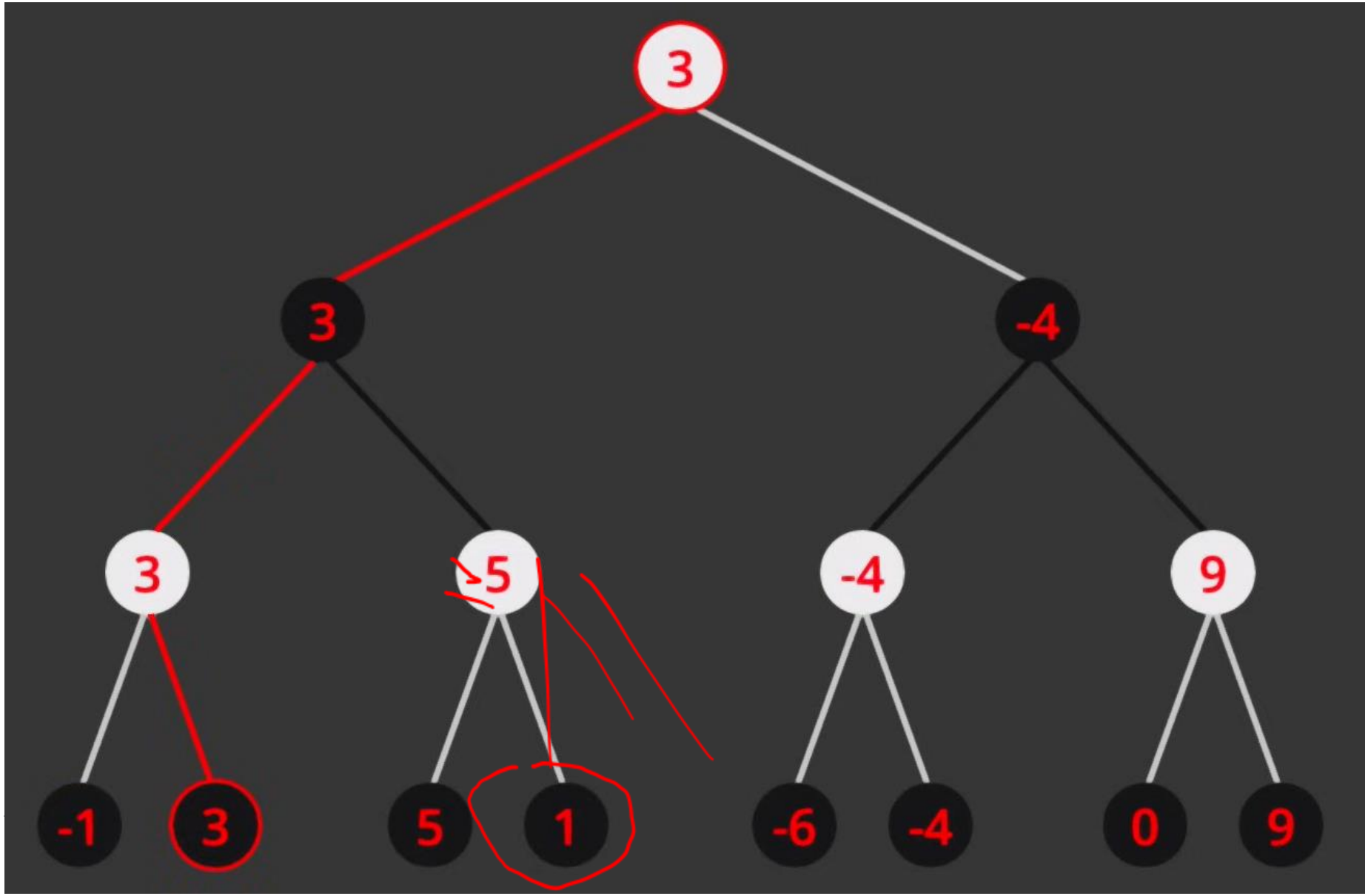
- Array pro Piece
- Wert pro Square + Materialwert
- Summe der Werte ergibt Wert der Stellung
- Weiß will hohe Zahl, Schwarz niedrige

Minimax - Vorüberlegung

- Annahme: wir sind weiß
- Unser Ziel: hohe Evaluation
- Gegner Ziel: niedrige Evaluation

Was ist der beste Zug?


3
3
5
-1
3
5
1
-6
-4
0
9


3
3
-4
3
5
-4
9
-1
3
5
1
-6
-4
0
9

Minimax

```
function minimax(position, depth, isWhite):
```

Minimax

```
function minimax(position, depth, isWhite):
    if depth == 0:
        return evaluate(position)
```

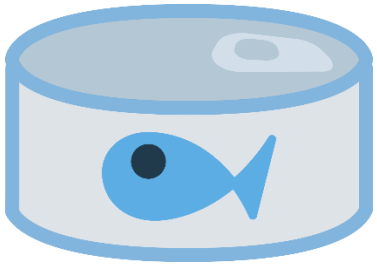
Minimax

```
function minimax(position, depth, isWhite):
    if depth == 0:
        return evaluate(position)

    if isWhite:
        maxEval = -infinity
        for move in generateLegalMoves(position):
            newPosition = makeMove(position, move)
            eval = minimax(newPosition, depth - 1, false)
            maxEval = max(maxEval, eval)
        return maxEval
```

Minimax

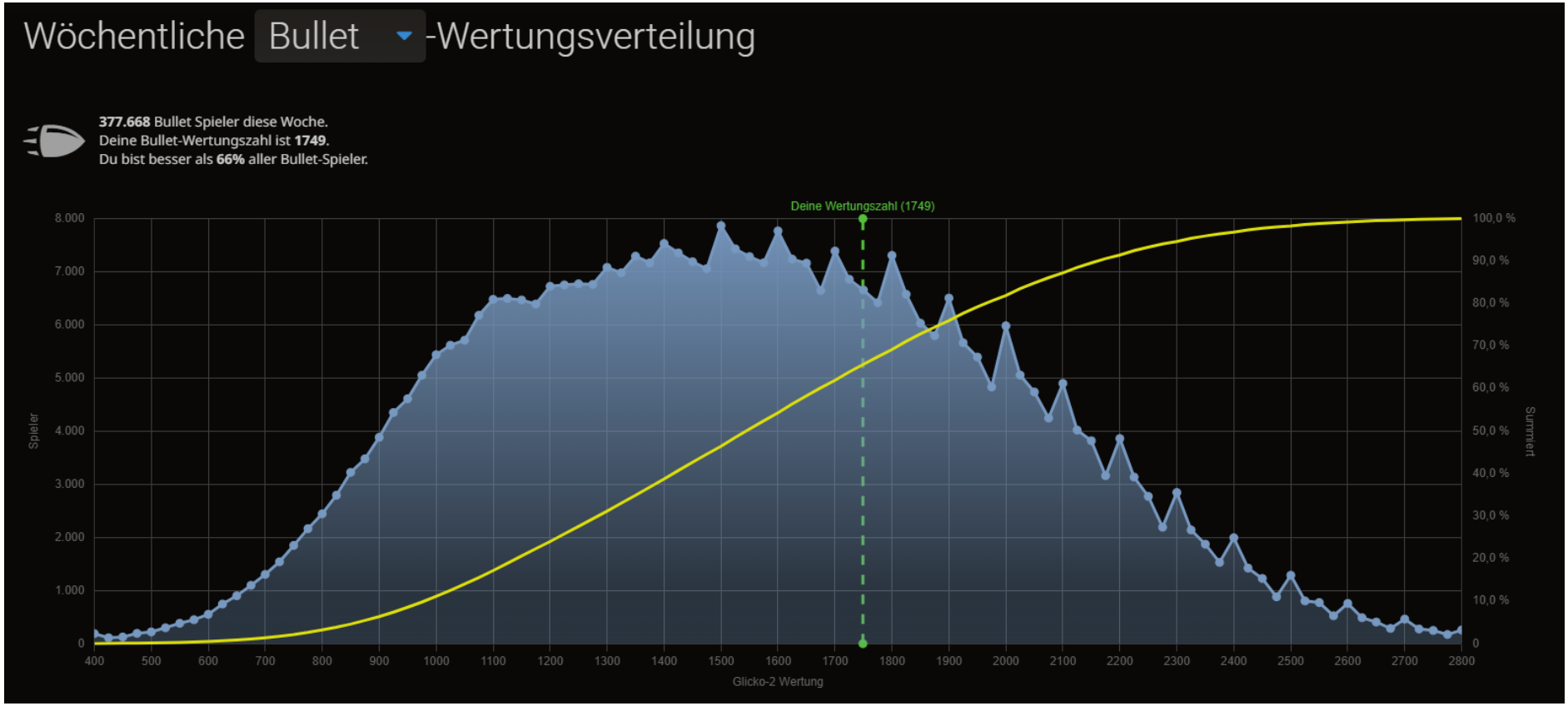
```
function minimax(position, depth, isWhite):
    if depth == 0:
        return evaluate(position)

    if isWhite:
        maxEval = -infinity
        for move in generateLegalMoves(position):
            newPosition = makeMove(position, move)
            eval = minimax(newPosition, depth - 1, false)
            maxEval = max(maxEval, eval)
        return maxEval
    else: # Flip for other side...
```

Thunfisch



- Benutzt UCI-Protokoll
- Registrierter Bot auf Lichess
- Ca 1750 Bullet Elo
- Ca 1600 Blitz Elo
- Multithreaded
- 100% Safe (kein unsafe, kein assembly)
- Cross Plattform (nicht auf x86 beschränkt)

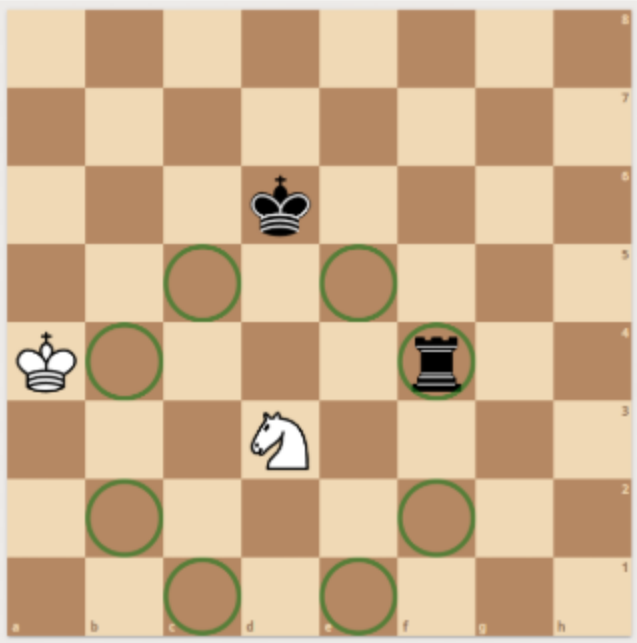
Vergleich auf Lichess



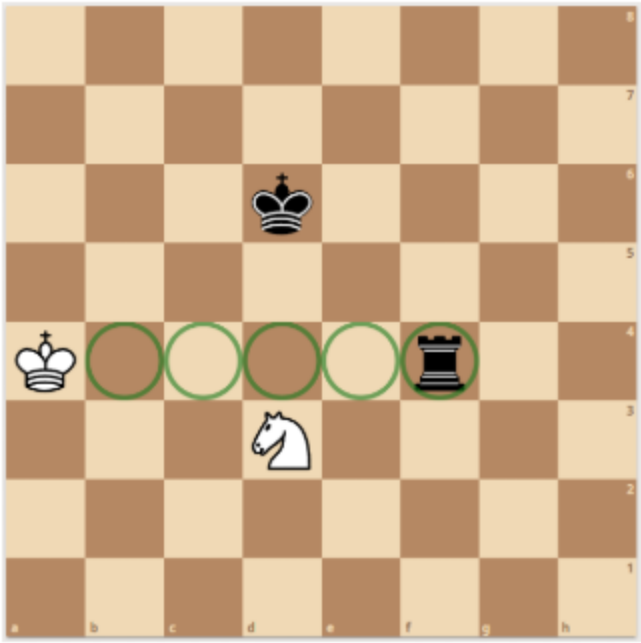
Zuggenerierung

- Bitboards
- Fast keine If-Statements alles Bitmasken
- Multithreaded
- Lookup-Tables für alle Figuren
- Angriffsmasken
- Pimasken
- Checkmasken
- ArrayVec

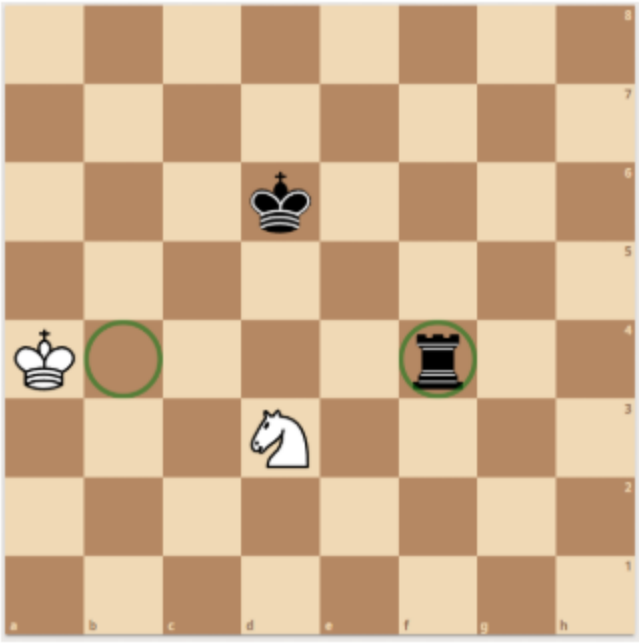
Zuggenerierung



Knight moves

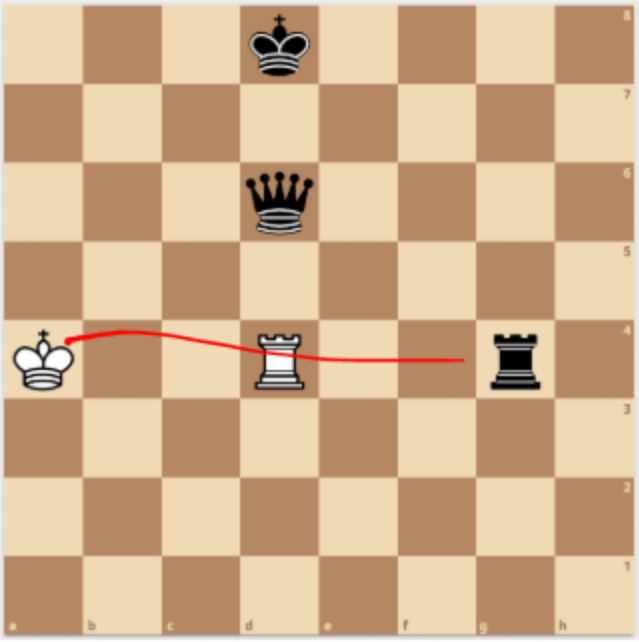


checkmask

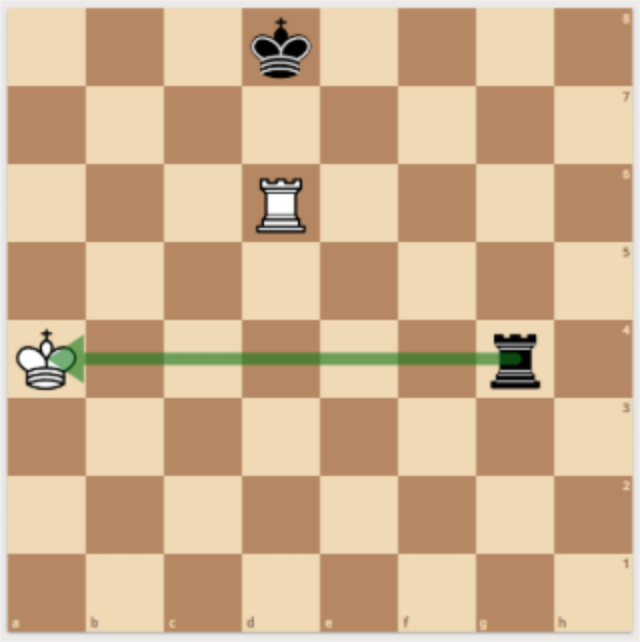


move & checkmask

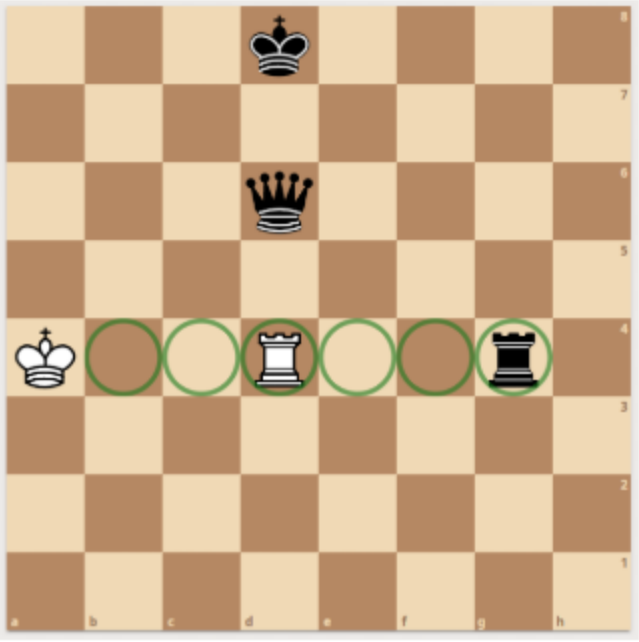
Zuggenerierung



Rook eyeing the king



Forbidden Rook move



Definition of pinmask

Suche

- Iterative Deepening
- Minimax Algorithmus
- Alpha Beta Pruning
- Quiescene Search
- Transposition Tables
- MVV-LVA Move Ordering
- Piece Square Table Evaluation
- Multithreaded

Zuggenerierung

- ca 2 Milliarden legale Züge pro Sekunde
- schneller als Stockfish* (9,5s vs 1,5s) (Hardware i5-13600kf 20 Threads)

```
go perft 7 --rayon
Perft: Depth=7 Nodes=3,195,901,860 Time=1.561s Nodes/sec=2,047,733,295
```

Ausblick

- Über 2000 Elo zu erreichen
- Besseres Branch Pruning
- Transposition Tables erweitern
- Move Ordering verbessern (Killer & History Heuristics)
- Eröffnungen hardcoden (Opening Books)

Duell