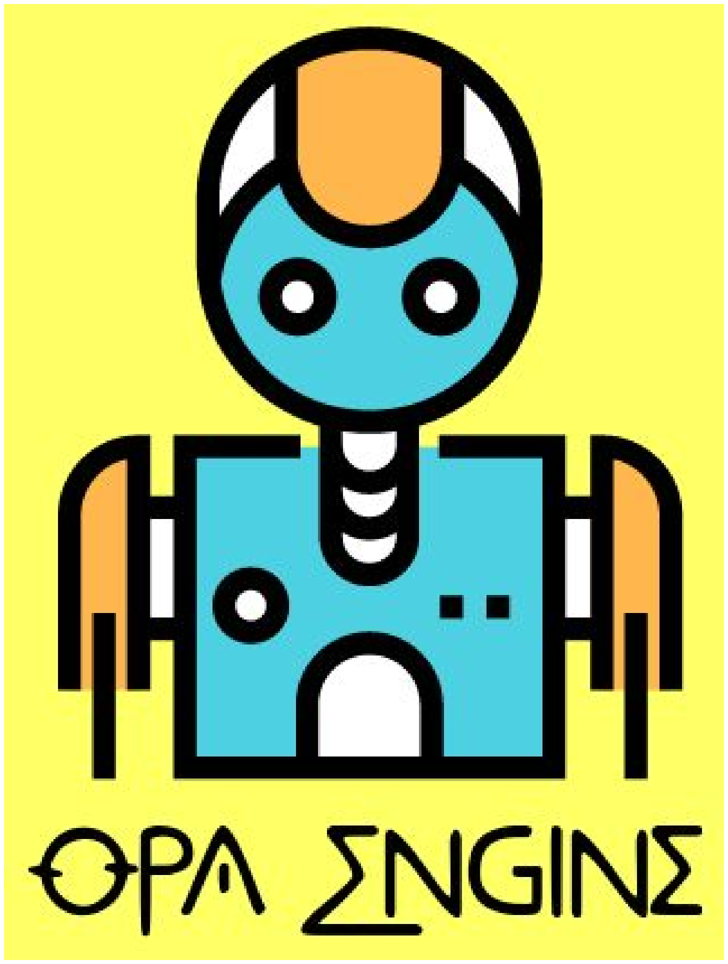
CSE386: Artificial Intelligence

Intelligent Chess Game (OPA Engine v1.0)



Ali Mohamed Anwar
Omar Mohamed Adel
Peter Magdy Safwat

Introduction

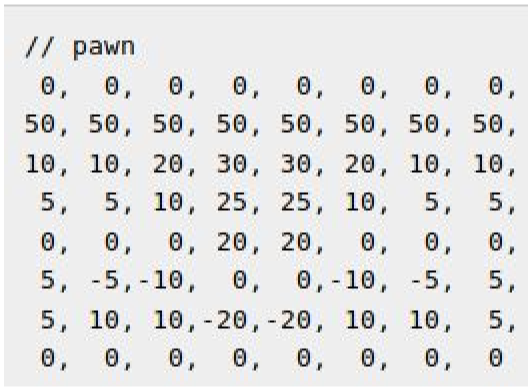
The goal of this project was to implement a Chess game using the Minimax algorithm with alpha-beta pruning. This required designing an appropriate chess board evaluation function to be used as the algorithm's utility function. The game allows a human player to play against the algorithm. The algorithm uses a depth-first strategy when exploring the game tree to ensure efficient memory usage.

The game supports both console interface and GUI when linked to Arena.

How it works

1. Classes:
    a. `Board:` Deals with the Chess Board (initializing, updating, printing, applying moves, ..)
    b. `Piece:` Handles each piece type, color, and checks for eligible moves/eating.
    c. `CommonMethods:` Global common functions/variables used to ease the coding process.
    d. `AI:` Evaluation and alphabeta algorithm implementation.
    e. `UCI:` Protocol communication methods according to the UCI documentation.

2. Main Functions:
    a. `alphabeta():` Takes game arguments (player side & board) alpha, beta, and desired depth. Function then checks for next eligible move, apply it and calls itself recursively with decrementing the depth till it reaches 0 applying the min-max alphabeta algorithm.
    b. `evaluate():` Evaluation occurs according to piece weight and location weight.



```
// pawn
 0,  0,  0,  0,  0,  0,  0,  0,
50, 50, 50, 50, 50, 50, 50, 50,
10, 10, 20, 30, 30, 20, 10, 10,
 5,  5, 10, 25, 25, 10,  5,  5,
 0,  0,  0, 20, 20,  0,  0,  0,
 5, -5,-10,  0,  0,-10, -5,  5,
 5, 10, 10,-20,-20, 10, 10,  5,
 0,  0,  0,  0,  0,  0,  0,  0
```

Example: Pawn Evaluation Points

c. `apply_move():` Takes move as argument, checks if valid or not and accordingly makes the decision to apply it or not on the Chess Board.

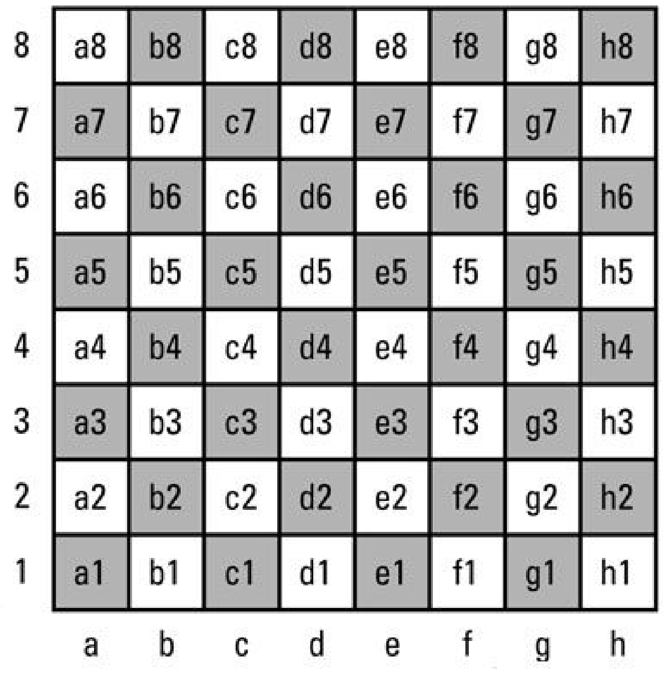
d. `update_board():` Takes location of each piece and updates the Chess Board (2D Array).

e. `print_board():` Prints the board after each applied move with board coordinates ([1-8][a-h]).

f. `blocked():` Checks if piece valid path is blocked with any other piece or not.

User Guide

Board Layout:

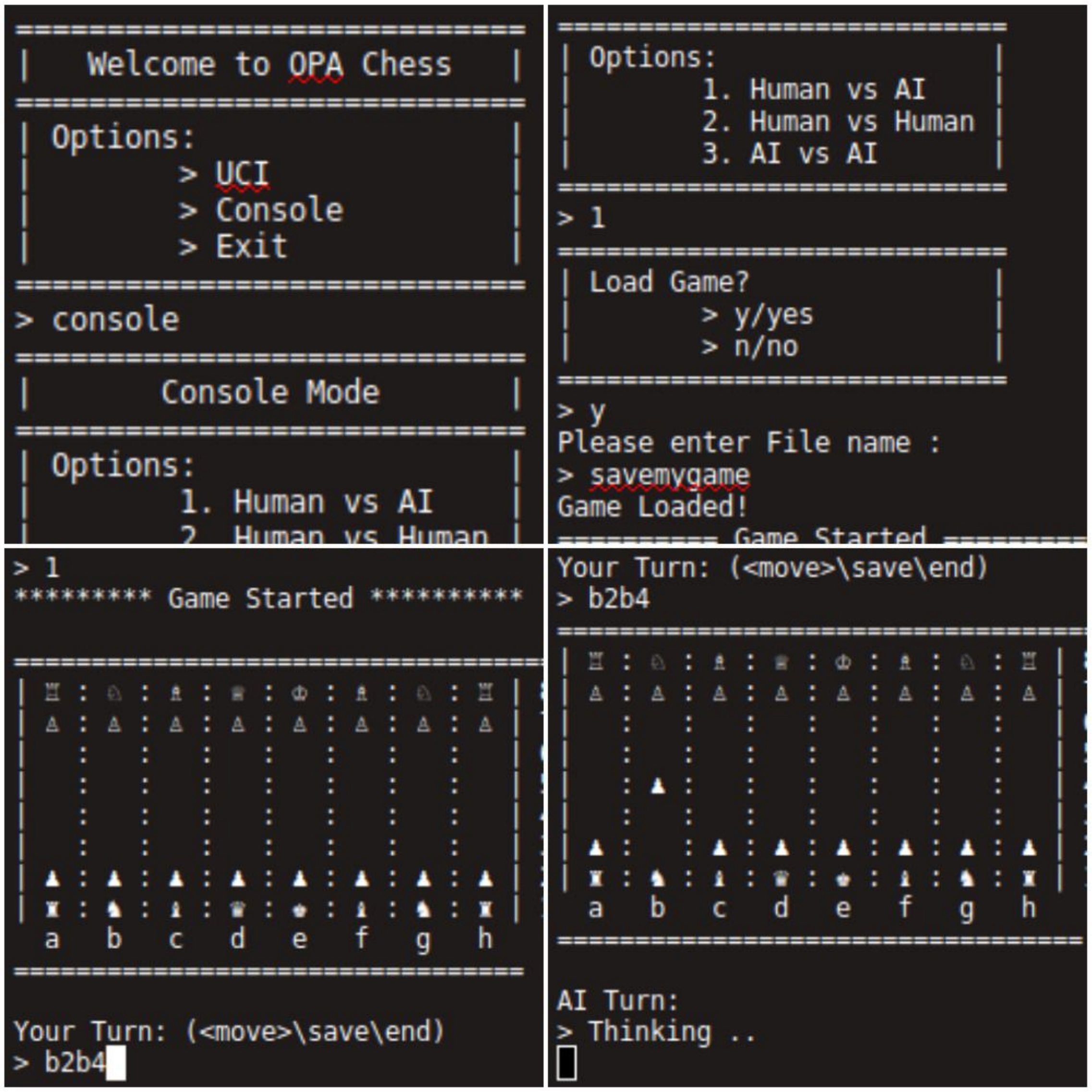

8 a8 b8 c8 d8 e8 f8 g8 h8
7 a7 b7 c7 d7 e7 f7 g7 h7
6 a6 b6 c6 d6 e6 f6 g6 h6
5 a5 b5 c5 d5 e5 f5 g5 h5
4 a4 b4 c4 d4 e4 f4 g4 h4
3 a3 b3 c3 d3 e3 f3 g3 h3
2 a2 b2 c2 d2 e2 f2 g2 h2
1 a1 b1 c1 d1 e1 f1 g1 h1
a b c d e f g h

Console Based Mode:

1. Player types “`console`” to enter game in console-based mode.
2. Player selects game mode (`Human vs AI, Human vs Human, AI vs AI`).
3. (If Human plays) Player is given the option to load from previously saved game or start new game.
4. Player chooses his/her preference of color.
5. Chessboard is displayed to Player.
6. *Player makes the next move in 4 chars (e.g “`b2b3`” will move pawn one step forward).
7. AI thinks of (evaluates) the next eligible best move.
8. Loops through steps (2-7) till game is over.

* Player has the option to save/load game at any instant while playing by entering “`save`” or “`load`” then the saved file name.

Screenshots:



```
==============================
|   Welcome to OPA Chess     |
==============================
| Options:                   |
|          > UCI             |
|          > Console         |
|          > Exit            |
==============================
> console
==============================
|        Console Mode        |
==============================
| Options:                   |
|          1. Human vs AI    |
|          2. Human vs Human |
```

```
=============================
| Options:                  |
|          1. Human vs AI   |
|          2. Human vs Human|
|          3. AI vs AI      |
=============================
> 1
=============================
| Load Game?                |
|          > y/yes          |
|          > n/no           |
=============================
> y
Please enter File name :
> savemygame
Game Loaded!
========== Game Started ==========
```

```
> 1
********* Game Started **********

===================================
| ♖ : ♘ : ♗ : ♕ : ♔ : ♗ : ♘ : ♖ |
| ♙ : ♙ : ♙ : ♙ : ♙ : ♙ : ♙ : ♙ |
|   :   :   :   :   :   :   :   |
|   :   :   :   :   :   :   :   |
|   :   :   :   :   :   :   :   |
|   :   :   :   :   :   :   :   |
| ♟ : ♟ : ♟ : ♟ : ♟ : ♟ : ♟ : ♟ |
| ♜ : ♞ : ♝ : ♛ : ♚ : ♝ : ♞ : ♜ |
  a   b   c   d   e   f   g   h
===================================

Your Turn: (<move>\save\end)
> b2b4
```

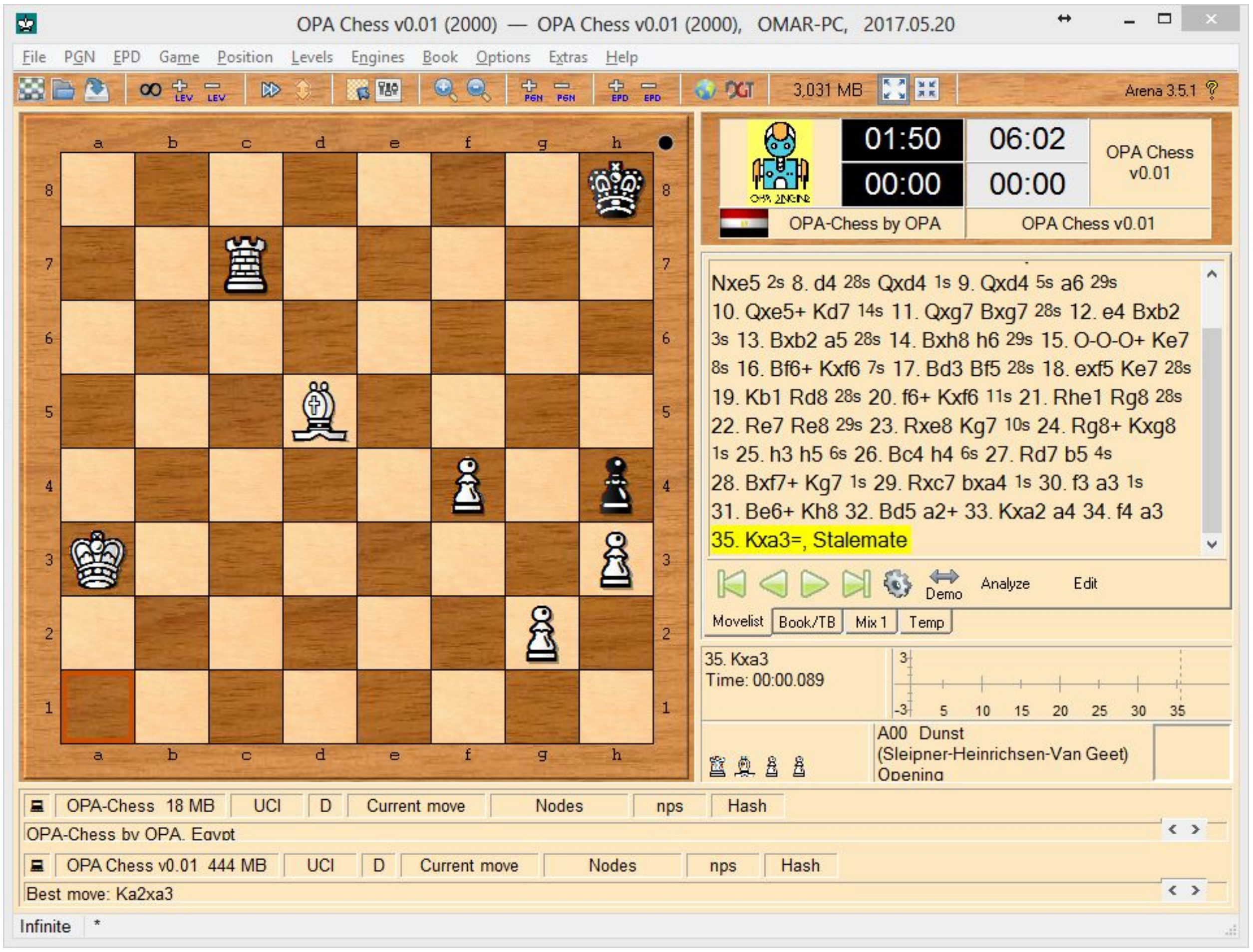
```
Your Turn: (<move>\save\end)
> b2b4
===================================
| ♖ : ♘ : ♗ : ♕ : ♔ : ♗ : ♘ : ♖ |
| ♙ : ♙ : ♙ : ♙ : ♙ : ♙ : ♙ : ♙ |
|   :   :   :   :   :   :   :   |
|   :   :   :   :   :   :   :   |
|   : ♟ :   :   :   :   :   :   |
|   :   :   :   :   :   :   :   |
| ♟ :   : ♟ : ♟ : ♟ : ♟ : ♟ : ♟ |
| ♜ : ♞ : ♝ : ♛ : ♚ : ♝ : ♞ : ♜ |
  a   b   c   d   e   f   g   h
===================================

AI Turn:
> Thinking ..
```

Console Game Example

GUI Based (Arena):

1. Import Game Engine (`.jar` file) into Arena Software, then load the Engine as game "engine 1".
2. Load any other world-wide Chess Game Engine.
3. Press on "`Demo`" button to make the engine play against itself.
4. Press on "`Gear`" button to let the engine calculate and make a right move.
5. Press on "X" button to make the engine stop thinking and play the latest best move calculated.

Screenshots:



OPA Chess v0.01 (2000) — OPA Chess v0.01 (2000), OMAR-PC, 2017.05.20
File PGN EPD Game Position Levels Engines Book Options Extras Help
3,031 MB
Arena 3.5.1
01:50 06:02
00:00 00:00
OPA Chess v0.01
OPA-Chess by OPA
OPA Chess v0.01
Analyze Edit Demo
Movelist Book/TB Mix 1 Temp
OPA-Chess 18 MB UCI D Current move Nodes nps Hash
OPA Chess v0.01 444 MB UCI D Current move Nodes nps Hash
Infinite *

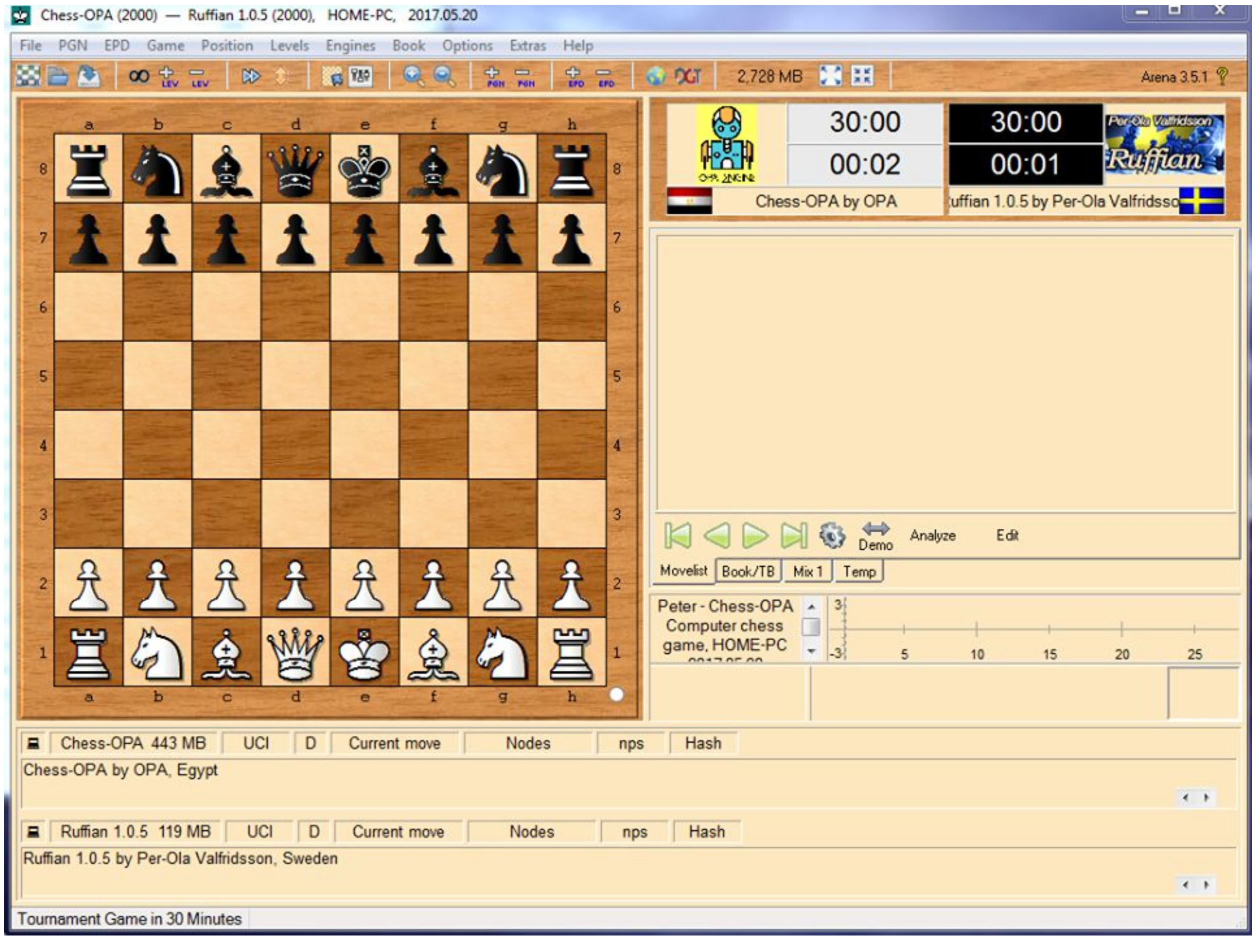
Nxe5 2s 8. d4 28s Qxd4 1s 9. Qxd4 5s a6 29s 10. Qxe5+ Kd7 14s 11. Qxg7 Bxg7 28s 12. e4 Bxb2 3s 13. Bxb2 a5 28s 14. Bxh8 h6 29s 15. O-O-O+ Ke7 8s 16. Bf6+ Kxf6 7s 17. Bd3 Bf5 28s 18. exf5 Ke7 28s 19. Kb1 Rd8 28s 20. f6+ Kxf6 11s 21. Rhe1 Rg8 28s 22. Re7 Re8 29s 23. Rxe8 Kg7 10s 24. Rg8+ Kxg8 1s 25. h3 h5 6s 26. Bc4 h4 6s 27. Rd7 b5 4s 28. Bxf7+ Kg7 1s 29. Rxc7 bxa4 1s 30. f3 a3 1s 31. Be6+ Kh8 32. Bd5 a2+ 33. Kxa2 a4 34. f4 a3 35. Kxa3=, Stalemate

35. Kxa3
Time: 00:00.089

A00 Dunst
(Sleipner-Heinrichsen-Van Geet)
Opening

OPA-Chess by OPA, Egypt

Best move: Ka2xa3


Chess-OPA (2000) — Ruffian 1.0.5 (2000), HOME-PC, 2017.05.20

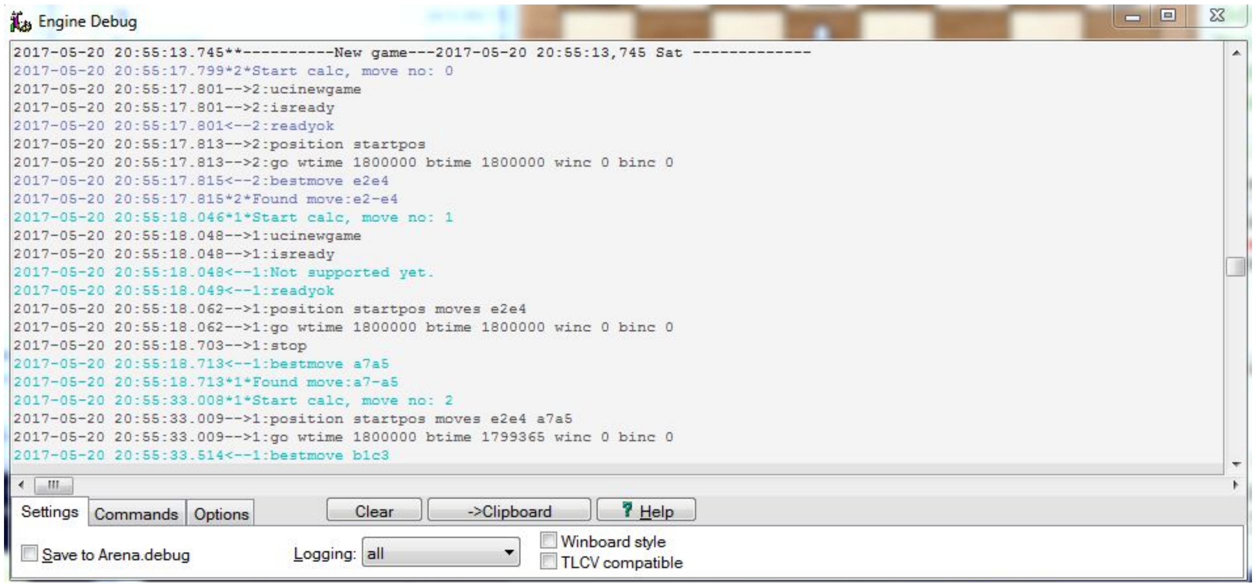
30:00	30:00
00:02	00:01
Chess-OPA by OPA	Ruffian 1.0.5 by Per-Ola Valfridsson

Peter - Chess-OPA Computer chess game, HOME-PC

	Chess-OPA 443 MB	UCI	D	Current move	Nodes	nps	Hash
Chess-OPA by OPA, Egypt							

	Ruffian 1.0.5 119 MB	UCI	D	Current move	Nodes	nps	Hash
Ruffian 1.0.5 by Per-Ola Valfridsson, Sweden							

Tournament Game in 30 Minutes


Engine Debug

```
2017-05-20 20:55:13.745**----------New game---2017-05-20 20:55:13,745 Sat -------------
2017-05-20 20:55:17.799*2*Start calc, move no: 0
2017-05-20 20:55:17.801-->2:ucinewgame
2017-05-20 20:55:17.801-->2:isready
2017-05-20 20:55:17.801<--2:readyok
2017-05-20 20:55:17.813-->2:position startpos
2017-05-20 20:55:17.813-->2:go wtime 1800000 btime 1800000 winc 0 binc 0
2017-05-20 20:55:17.815<--2:bestmove e2e4
2017-05-20 20:55:17.815*2*Found move:e2-e4
2017-05-20 20:55:18.046*1*Start calc, move no: 1
2017-05-20 20:55:18.048-->1:ucinewgame
2017-05-20 20:55:18.048-->1:isready
2017-05-20 20:55:18.048<--1:Not supported yet.
2017-05-20 20:55:18.049<--1:readyok
2017-05-20 20:55:18.062-->1:position startpos moves e2e4
2017-05-20 20:55:18.062-->1:go wtime 1800000 btime 1800000 winc 0 binc 0
2017-05-20 20:55:18.703-->1:stop
2017-05-20 20:55:18.713<--1:bestmove a7a5
2017-05-20 20:55:18.713*1*Found move:a7-a5
2017-05-20 20:55:33.008*1*Start calc, move no: 2
2017-05-20 20:55:33.009-->1:position startpos moves e2e4 a7a5
2017-05-20 20:55:33.009-->1:go wtime 1800000 btime 1799365 winc 0 binc 0
2017-05-20 20:55:33.514<--1:bestmove b1c3
```

Save to Arena.debug [ ]   Logging: all   Winboard style [ ]   TLCV compatible [ ]

Bonus Features Implemented

1. Implement an iterative-deepening variation of the alpha-beta algorithm to support both: a “play now” button and a time limit setting. If a “play now” button is supported, clicking it would force the game to respond immediately with the best move found so far. Also if a time limit setting is supported, the game will be required to respond with the best move found when the time limit has expired (in each turn).
2. Support game saving and loading in addition to starting the game from any board state (not necessary the standard chess state) .
3. Support a networking mode allowing two instances of the game (running on the same machine or on different machines) to play against each other via without any human intervention. This function will be particularly useful if multiple teams agree on a communication protocol allowing their games to play against other.

Challenges we ran into

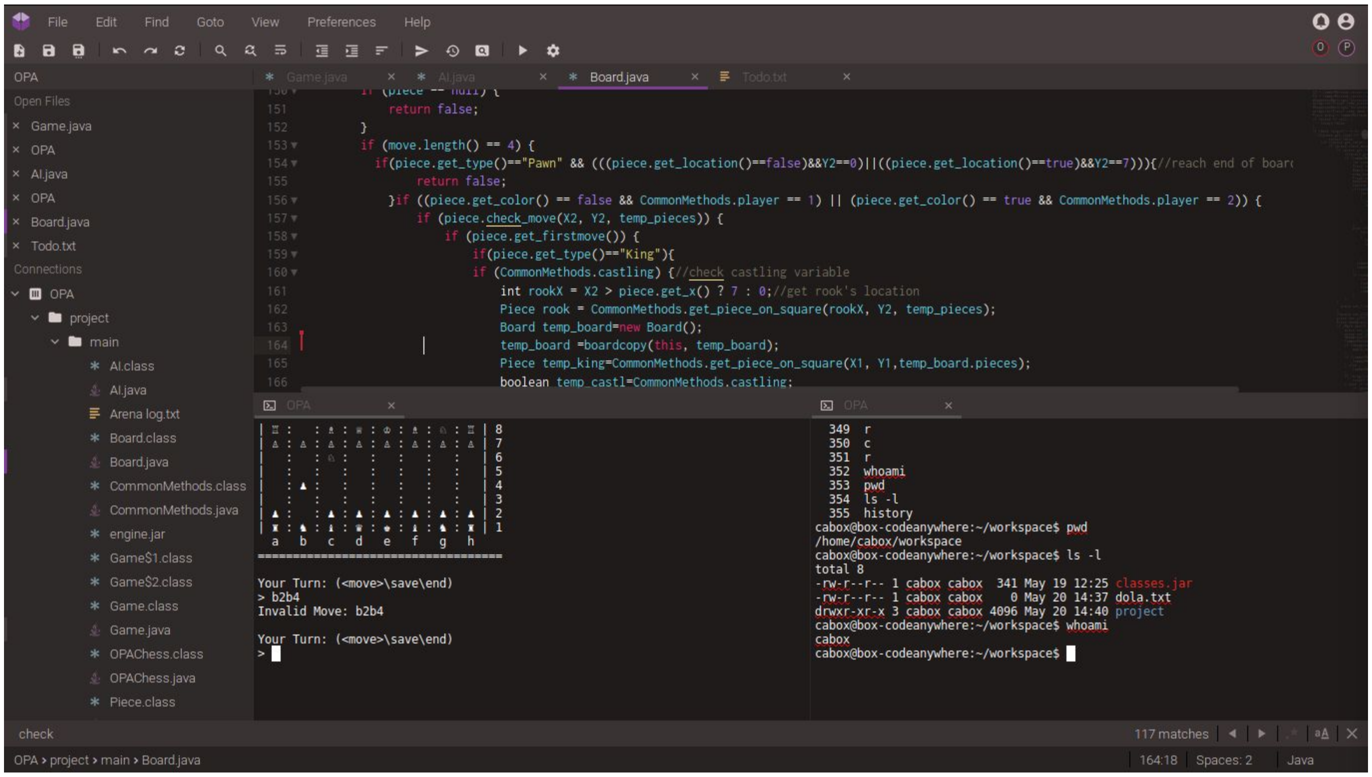
- Zero knowledge of UCI protocol commands/linking methods prior to this project.
- Implementing Chess special moves (Promotions, En Passant, Castling, ..).
- Stopping the AI engine and returning with the best move it reached till then.
- Generating all the next possible moves for each piece.
- Verifying valid moves in case of a checked King.

Accomplishments that we're proud of

- Learned a lot about UCI and the Arena Interface.
- Using the Extreme Pair Programming methodologies to develop a project.
- Learned more about how to evaluate the board (min-max) and generate the best move.
- Learned more about the Chess Game rules.
- Managed to compete against Chess Game Engines from all the over the world.

Who did what

With some bash scripts we managed to configure a container on Codeanywhere with Java Development stack (OpenJDK 7, OpenJDE 7 and Tomcat 7) preinstalled. We used this container to multi-code, compile, debug, and test the project in real time; so it was mainly XP programming (no tasks splitting).



```java
      if (piece == null) {
        return false;
      }
      if (move.length() == 4) {
        if(piece.get_type()=="Pawn" && (((piece.get_location()==false)&&Y2==0)||((piece.get_location()==true)&&Y2==7))){//reach end of boar
            return false;
          }if ((piece.get_color() == false && CommonMethods.player == 1) || (piece.get_color() == true && CommonMethods.player == 2)) {
            if (piece.check_move(X2, Y2, temp_pieces)) {
              if (piece.get_firstmove()) {
                if(piece.get_type()=="King"){
                if (CommonMethods.castling) {//check castling variable
                  int rookX = X2 > piece.get_x() ? 7 : 0;//get rook's location
                  Piece rook = CommonMethods.get_piece_on_square(rookX, Y2, temp_pieces);
                  Board temp_board=new Board();
                  temp_board =boardcopy(this, temp_board);
                  Piece temp_king=CommonMethods.get_piece_on_square(X1, Y1,temp_board.pieces);
                  boolean temp_castl=CommonMethods.castling;
```

```
| ♖ :   : ♗ : ♕ : ♔ : ♗ : ♘ : ♖ | 8
| ♙ : ♙ : ♙ : ♙ : ♙ : ♙ : ♙ : ♙ | 7
|   :   : ♘ :   :   :   :   :   | 6
|   :   :   :   :   :   :   :   | 5
|   : ♟ :   :   :   :   :   :   | 4
|   :   :   :   :   :   :   :   | 3
| ♟ :   : ♟ : ♟ : ♟ : ♟ : ♟ : ♟ | 2
| ♜ : ♞ : ♝ : ♛ : ♚ : ♝ : ♞ : ♜ | 1
  a   b   c   d   e   f   g   h
===================================

Your Turn: (<move>\save\end)
> b2b4
Invalid Move: b2b4

Your Turn: (<move>\save\end)
> 
```

```
  349  r
  350  c
  351  r
  352  whoami
  353  pwd
  354  ls -l
  355  history
cabox@box-codeanywhere:~/workspace$ pwd
/home/cabox/workspace
cabox@box-codeanywhere:~/workspace$ ls -l
total 8
-rw-r--r-- 1 cabox cabox  341 May 19 12:25 classes.jar
-rw-r--r-- 1 cabox cabox    0 May 20 14:37 dola.txt
drwxr-xr-x 3 cabox cabox 4096 May 20 14:40 project
cabox@box-codeanywhere:~/workspace$ whoami
cabox
cabox@box-codeanywhere:~/workspace$ 
```

Try it out

Github Repo

Devpost