AI Final Report

Mentor : Chirumamilla Bhargav
Animesh Jain 220120005
Gurjaipal Singh 220120008
Tanishque Soni 220120025
Upakramanika Bishnu 225100039
Mahamkali Sri Phaneeswar 225100023
Shinde Manav Rajendra 225100032

April 2024

Contents

1 Introduction

Chess is an abstract strategy game that involves no hidden information the entire 64 squares are observable by both player at all time and no elements of chance as the move chosen by the player get reflected on the board as it is.

In the groundbreaking paper on computer chess, "Programming a Computer for Playing Chess", was published in 1950 by Claude Shannon. He wrote:

The chess machine is an ideal one to start with, since:

1. the problem is sharply defined both in allowed operations (the moves) and in the ultimate goal (checkmate)
2. it is neither so simple as to be trivial nor too difficult for satisfactory solution
3. chess is generally considered to require "thinking" for skillful play; a solution of this problem will force us either to admit the possibility of a mechanized thinking or to further restrict our concept of "thinking"
4. the discrete structure of chess fits well into the digital nature of modern computers

This report provides an overview and analysis of the Chess Game GUI developed by the group 2, covering the components and functionality implemented in the project as well as the AI Algorithms implemented in the projects refer to appendix in order to know how to run the program [5.1]

2 Project Structure

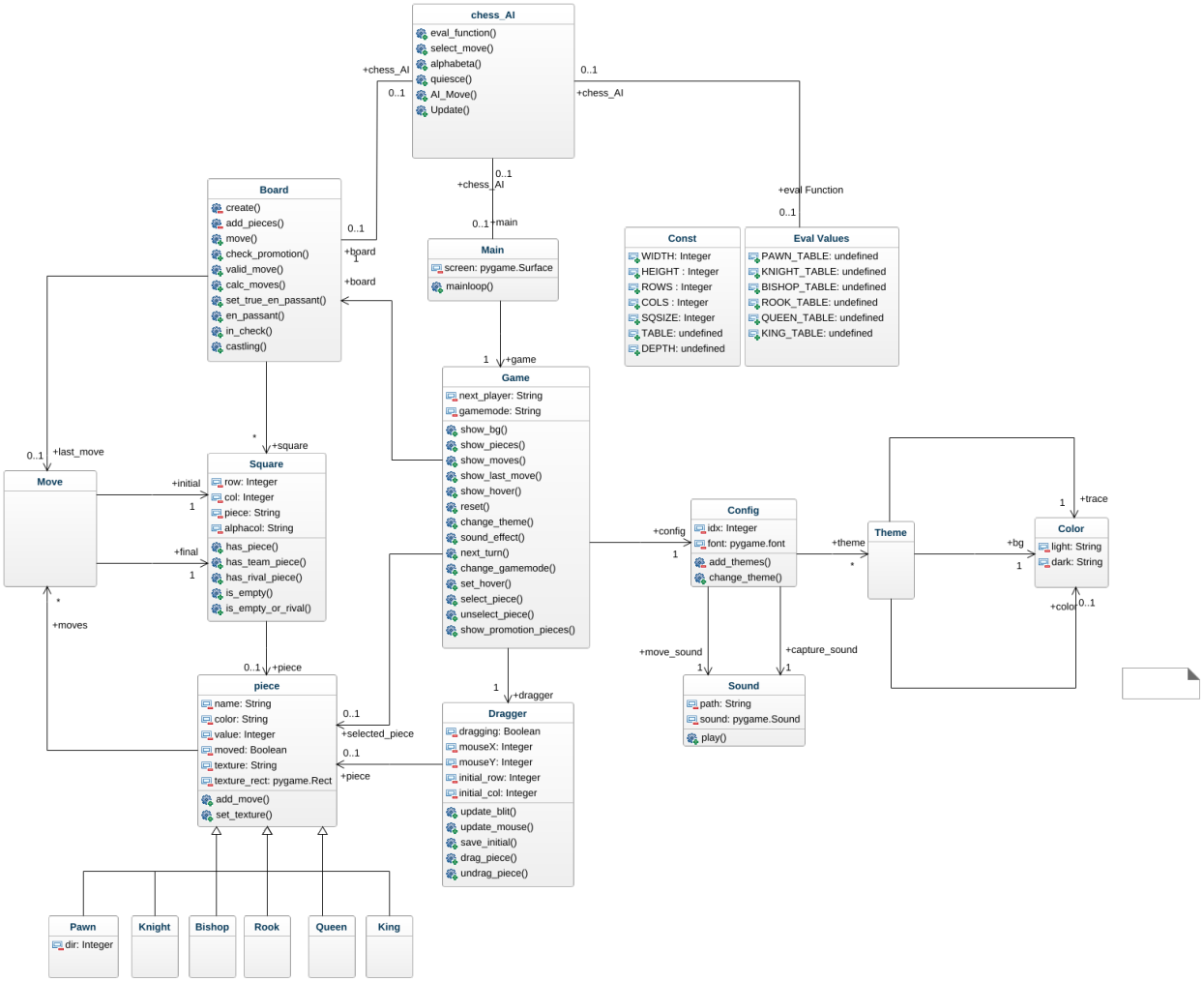
The project consists of several Python files organized into different modules:

1. **board.py**: Contains the logic and rendering for the chessboard.
2. **chess AI.py**: Contains evaluation function, minimax algorithm and helper function for the AI agent to interact with GUI and decide and make moves
3. **dragger.py**: Implements the draggable functionality for moving chess pieces.
4. **Eval Values.py**: Contains all the eval table for AI. It is explained din detail in section [3.1]
5. **game.py**: Manages the game flow and interaction between components.
6. **main.py**: Entry point for running the chess game.
7. **move.py**: Manages and executes chess piece movements within the game.
8. **piece.py**: Defines the chess piece classes.

9. **square.py**: Defines the chessboard squares and their properties.

10. other helper files includes **color.py config.py const.py sound.py** and **theme.py**

11. **depth_log_parser.py**: It parses the depth log created by the AI file. Prints out the maximum depth reached for each call of alphabeta pruning.



More details and GUI implementations are giving in appendix [5.2]

3 AI Implementation

3.1 Evaluation of states

For any AI algorithm it is important to select a good Evaluation function.

So for our evaluation function we have 3 parameter deciding the valuation on the current state of the board

1. Base Value of the Piece. It act as the default value for a piece which defines as keeping everything else constant how powerful a piece is. For Ex : A pawn is values as 1 unit while a Queen as 9
2. Temporal Values depending on what stage (Opening, Mid or End) game is in. As the game progress the importance of different piece change. For Ex Rooks becomes more important as game progress as more boards gets free for them to move around
3. Spacial Values based on where the piece is located on the board. As the location of piece is important to its mobility and usefulness. For Ex: A horse that is stuck in the corner is much less powerful than one in the central squares

All the values used in our implementation in provided in the appendix 5.6 below. All the values are stored in tables for each piece. Also the Temporal Variation of the Base values are combined into one table to simplify implementation

We can use different table to change the style of play of the bot as we have to different set of tables (Simple and PeSTO)

Simple table allows for a more balanced play while PeSTO leads to a more aggressive play by the bot

In conjunction to this there is logic implemented inside the AI_Move function to determine in which stage (Opening, Mid or End) the game is in. It is done by checking some criterion of the game such how many moves have been played, how may and which all pieces has been captured. As can be seen in Code Snippets provided 1

3.2 Negamax

We have used Negamax implementation of the Minimax algorithm.

This takes advantage of the fact it is a 2 player zero-sum game. Instead of minimax's 2 functions, one for minimising player and one for maximising, negamax utilises only one.

The negative of the maximum is passed up for each player. The code for negamax with alphabeta pruning can be found below. 3.5

3.3 Alphabeta Pruning

AlphaBeta Pruing is used to optimize the search algorithm. It is a very basic implementation of the algorithm similar to the one done is assignment

3.4 Quiescence

There are states that have a good value if we evaluate just the state itself, and seem like a good state, but it might be followed immediately by a capture of a piece by the opponent.

This is the result of depth limitation, and is called **Horizon Effect**. This is overcome by Quiescence search, where we see if a state is "quiet." If it's not, we go down the tree till we reach a quiet state with no captures and send back the evaluation for that state.

So now, we are doing a variable depth search. The code for quiescence search can be found below. 3.5

Illustration of the Horizon Effect

3.5 Code Snippets

```
def quiesce(self, alpha, beta):

    stand_pat = self.eval_function()

    if (stand_pat >= beta):
        return beta
    if (alpha < stand_pat):
        alpha = stand_pat

    for move in self.AI_Board_ref.legal_moves:
        if self.AI_Board_ref.is_capture(move):
            self.AI_Board_ref.push(move)
            score = -self.quiesce(-beta, -alpha)
            self.AI_Board_ref.pop()
            if (score >= beta):
                return beta
            if (score > alpha):
                alpha = score

    return alpha
```

Quiescence search

```python
def alphabeta(self, alpha, beta, depthleft):
    bestscore = -9999
    if (depthleft == 0):
        return self.quiesce(alpha, beta)
    for move in self.AI_Board_ref.legal_moves:
        self.AI_Board_ref.push(move)
        score = -self.alphabeta(-beta, -alpha,
                                depthleft - 1)
        self.AI_Board_ref.pop()
        if (score >= beta):
            return score
        if (score > bestscore):
            bestscore = score
        if (score > alpha):
            alpha = score
    return bestscore
```

alphabeta

```python
def select_move(self, depth):
    bestMove = chess.Move.null()
    bestValue = -99999
    alpha = -100000
    beta = 100000
    for move in self.AI_Board_ref.legal_moves:
        self.AI_Board_ref.push(move)
        boardValue = -self.alphabeta(-beta, -alpha,
                                     depth - 1)
        if boardValue > bestValue:
            bestValue = boardValue
            bestMove = move
        if (boardValue > alpha):
            alpha = boardValue
        self.AI_Board_ref.pop()
    return bestMove
```

select move

Figure 1: AI Move

```python
def AI_Move(self, display_board, depth):
    mov = self.select_move(depth)

    if self.AI_Board_ref.is_capture(mov):
        self.captured_piece += 1

    is_mid = 0

    is_mid += (self.captured_piece >= 4) + (int(self.AI_Board_ref.fen()[-1]) >= 8) + (self.AI_Board_ref.fen().split()[-4] == "-")

    if is_mid >= 2:
        self.game_stage = "M"


    wn = len(self.AI_Board_ref.pieces(chess.KNIGHT, chess.WHITE))
    bn = len(self.AI_Board_ref.pieces(chess.KNIGHT, chess.BLACK))
    wb = len(self.AI_Board_ref.pieces(chess.BISHOP, chess.WHITE))
    bb = len(self.AI_Board_ref.pieces(chess.BISHOP, chess.BLACK))
    wr = len(self.AI_Board_ref.pieces(chess.ROOK, chess.WHITE))
    br = len(self.AI_Board_ref.pieces(chess.ROOK, chess.BLACK))
    wq = len(self.AI_Board_ref.pieces(chess.QUEEN, chess.WHITE))
    bq = len(self.AI_Board_ref.pieces(chess.QUEEN, chess.BLACK))

    is_end = 0

    is_end += ((wn + bn + wb + bb) < 4) + ((wr + br) < 2) + ((wq + bq) == 0) + ((self.captured_piece > 10))

    if (is_end >=3 ):
        self.game_stage = "E"
```

4 Contributions

- Animesh : Wrote Evaluation Function and Did Integration of GUI and AI
- Upakramanika : Wrote the AI Engine with Minimax(Negamax) and Quiecence Search also did testing
- Manav : Write the code to look for checks and en passant, testing
- Gurjaipal : Did GUI,in particular Wrote Function that calculates all possible(valid)moves a specific piece can move on the board,including making the pieces move on the board, testing
- Tanishque : Did Base GUI Board.py, dragger.py , made ReadMe.md file
- Phaneeswar : Did GUI, mainly sound and themes, testing

5 Appendix

5.1 Instruction to run the file

1. Download the zip file in to a empty folder named, say, Group2
2. Extract both the src and assets folder file into the same folder
3. From the command line run: "python3 src\main.py"
4. This should run the code in its default setting that is Human vs AI with AI having the black pieces and Spatial table to be simple
5. In order to run another configuration run the following command: "python3 src\main.py" with 3 argument ai_vs_ai, ai_player and table each taking only following values and are case sensitive [true or false] for ai_vs_ai, [black, white] for ai_player and [simple, PeSTO] for table
6. For example : "python3 src\main.py true black PeSTO" for a ai_vs_ai game with PeSTO table in this case 2nd argument doesn't matter but for example : "python3 src\main.py false white PeSTO" here it mean AI plays as white against human
7. After running the commands provided above, it creates a depth_log.txt file in the folder Group2. Then run the following command in the Group2 folder: "python3 depth_log_parser.py"

5.2 Project structure

5.3 Functionality Overview

5.3.1 Chessboard Rendering (`board.py`)

The `Board` class in `board.py` handles the rendering of the chessboard. It creates an 8x8 grid of squares using the `Square` class from `square.py`. The board is displayed using a graphical user interface (GUI) for easy interaction.

5.3.2 Chess Piece Movements (`move.py`)

The `move.py` module manages and executes chess piece movements within the game. It handles move validation, piece movement execution, and special rules such as en passant and castling.

5.3.3 Chess Pieces (`piece.py`)

The `Piece` class defines the properties and behavior of chess pieces. Subclasses like `Pawn`, `Rook`, `Knight`, etc., are implemented to represent different types of pieces. Each piece knows its current position on the board.

5.3.4 Draggable Pieces (`dragger.py`)

The `Dragger` class enables the drag-and-drop functionality for moving chess pieces. It handles mouse events to track when a piece is being dragged and dropped. This feature enhances the user experience by providing a more intuitive way to move pieces.

5.3.5 Game Management (`game.py`)

The `Game` class manages the overall game flow. It initializes the board, sets up pieces, and handles player turns. Moves are validated based on chess rules to ensure legal moves.

5.3.6 Main Program (`main.py`)

The `main.py` file serves as the entry point for running the chess game. It creates an instance of the `Game` class and starts the game loop.

5.4 Implementation of GUI - Themes, Colours, Sound Effects

To enhance the user experience in our chess game, we added various visual effects and sound effects. At the beginning, I majorly concentrated on integrating sound effects to the game enhancing the auditory feedback and immersion. Sound effects for movement and capturing are introduced as part of this.

Additionally I also focused on implementing themes allowing players to personalize their appearance of chess board, In this I have incorporated 4 colors, which they can be cycled through by pressing the key 'T' while playing the game.

For determining the color of each square on the board, I have used the sum of row and column indices, if the sum is even, it uses the light color from the theme otherwise dark.

And finally while developing valid movement colors for dragged piece, I have looped through the valid moves, it highlight specific squares in the board indicating the total number of valid moves. Also then it stores the move and then highlights the current position of the piece and previous position of the piece.

5.5 Valid Move

The calcMoves method calculates all possible valid moves for a given piece at a specified position on the board. It considers factors such as piece type, board state, and game rules to determine legal moves, including capturing enemy pieces and executing special pawn moves like en passant.

5.5.1 Place Movements

The move method handles the movement of chess pieces on the board. It validates player moves, enforces game rules (such as piece captures and special moves), and updates the board state accordingly. The Move class facilitates the management of chess moves within the chess program, allowing for the creation of move objects, displaying moves in a human-readable format, and comparing moves for equality.

5.5.2 In Check Condition

The *in check* condition occurs when a player's king is under direct threat of capture by an opponent's piece. In such a situation, the player must make a move to remove their king from danger. Failure to do so may result in losing the game.

5.5.3 En Passant Rule

The *en passant* rule is a special pawn capture move in chess. It allows a pawn that has just moved two squares forward from its starting position to be captured by an opponent's pawn as if it had only moved one square forward. This rule prevents a pawn from bypassing an opponent's pawn capture threat.

5.6 Evaluation Values

5.6.1 Simple Evaluation Policy

```
"O" : [0,  0,   0,   0,   0,   0,  0,  0,
       5, 10,  10, -20, -20,  10, 10,  5,
       5, -5, -10,   0,   0, -10, -5,  5,
       0,  0,   0,  20,  20,   0,  0,  0,
       5,  5,  10,  25,  25,  10,  5,  5,
      10, 10,  20,  30,  30,  20, 10, 10,
      50, 50,  50,  50,  50,  50, 50, 50,
       0,  0,   0,   0,   0,   0,  0,  0],
"M" : [0,  0,   0,   0,   0,   0,  0,  0,
       5, 10,  10, -20, -20,  10, 10,  5,
       5, -5, -10,   0,   0, -10, -5,  5,
       0,  0,   0,  20,  20,   0,  0,  0,
       5,  5,  10,  25,  25,  10,  5,  5,
      10, 10,  20,  30,  30,  20, 10, 10,
      50, 50,  50,  50,  50,  50, 50, 50,
       0,  0,   0,   0,   0,   0,  0,  0],
"E" : [0,  0,   0,   0,   0,   0,  0,  0,
       5, 10,  10, -20, -20,  10, 10,  5,
       5, -5, -10,   0,   0, -10, -5,  5,
       0,  0,   0,  20,  20,   0,  0,  0,
       5,  5,  10,  25,  25,  10,  5,  5,
      10, 10,  20,  30,  30,  20, 10, 10,
      50, 50,  50,  50,  50,  50, 50, 50,
       0,  0,   0,   0,   0,   0,  0,  0]}
```

Pawn

```
"O" : [-50, -40, -30, -30, -30, -30, -40, -50,
       -40, -20,   0,   5,   5,   0, -20, -40,
       -30,   5,  10,  15,  15,  10,   5, -30,
       -30,   0,  15,  20,  20,  15,   0, -30,
       -30,   5,  15,  20,  20,  15,   5, -30,
       -30,   0,  10,  15,  15,  10,   0, -30,
       -40, -20,   0,   0,   0,   0, -20, -40,
       -50, -40, -30, -30, -30, -30, -40, -50],

"M" : [-50, -40, -30, -30, -30, -30, -40, -50,
       -40, -20,   0,   5,   5,   0, -20, -40,
       -30,   5,  10,  15,  15,  10,   5, -30,
       -30,   0,  15,  20,  20,  15,   0, -30,
       -30,   5,  15,  20,  20,  15,   5, -30,
       -30,   0,  10,  15,  15,  10,   0, -30,
       -40, -20,   0,   0,   0,   0, -20, -40,
       -50, -40, -30, -30, -30, -30, -40, -50],

"E" : [-50, -40, -30, -30, -30, -30, -40, -50,
       -40, -20,   0,   5,   5,   0, -20, -40,
       -30,   5,  10,  15,  15,  10,   5, -30,
       -30,   0,  15,  20,  20,  15,   0, -30,
       -30,   5,  15,  20,  20,  15,   5, -30,
       -30,   0,  10,  15,  15,  10,   0, -30,
       -40, -20,   0,   0,   0,   0, -20, -40,
       -50, -40, -30, -30, -30, -30, -40, -50]}
```

Knight

```
"O" : [ -20, -10, -10, -10, -10, -10, -10, -20,
        -10,   5,   0,   0,   0,   0,   5, -10,
        -10,  10,  10,  10,  10,  10,  10, -10,
        -10,   0,  10,  10,  10,  10,   0, -10,
        -10,   5,   5,  10,  10,   5,   5, -10,
        -10,   0,   5,  10,  10,   5,   0, -10,
        -10,   0,   0,   0,   0,   0,   0, -10,
        -20, -10, -10, -10, -10, -10, -10, -20],

"M" : [ -20, -10, -10, -10, -10, -10, -10, -20,
        -10,   5,   0,   0,   0,   0,   5, -10,
        -10,  10,  10,  10,  10,  10,  10, -10,
        -10,   0,  10,  10,  10,  10,   0, -10,
        -10,   5,   5,  10,  10,   5,   5, -10,
        -10,   0,   5,  10,  10,   5,   0, -10,
        -10,   0,   0,   0,   0,   0,   0, -10,
        -20, -10, -10, -10, -10, -10, -10, -20],

"E" : [ -20, -10, -10, -10, -10, -10, -10, -20,
        -10,   5,   0,   0,   0,   0,   5, -10,
        -10,  10,  10,  10,  10,  10,  10, -10,
        -10,   0,  10,  10,  10,  10,   0, -10,
        -10,   5,   5,  10,  10,   5,   5, -10,
        -10,   0,   5,  10,  10,   5,   0, -10,
        -10,   0,   0,   0,   0,   0,   0, -10,
        -20, -10, -10, -10, -10, -10, -10, -20]}
```

Bishop

```
"O" : [  0,  0,  0,  5,  5,  0,  0,  0,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
         5, 10, 10, 10, 10, 10, 10,  5,
         0,  0,  0,  0,  0,  0,  0,  0],

"M" : [  0,  0,  0,  5,  5,  0,  0,  0,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
         5, 10, 10, 10, 10, 10, 10,  5,
         0,  0,  0,  0,  0,  0,  0,  0],

"E" : [  0,  0,  0,  5,  5,  0,  0,  0,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
        -5,  0,  0,  0,  0,  0,  0, -5,
         5, 10, 10, 10, 10, 10, 10,  5,
         0,  0,  0,  0,  0,  0,  0,  0]}
```

Rook

```
"O" : [ -20, -10, -10, -5, -5, -10, -10, -20,
        -10,   0,   0,  0,  0,   0,   0, -10,
        -10,   5,   5,  5,  5,   5,   0, -10,
          0,   0,   5,  5,  5,   5,   0,  -5,
         -5,   0,   5,  5,  5,   5,   0,  -5,
        -10,   0,   5,  5,  5,   5,   0, -10,
        -10,   0,   0,  0,  0,   0,   0, -10,
        -20, -10, -10, -5, -5, -10, -10, -20],

"M" : [ -20, -10, -10, -5, -5, -10, -10, -20,
        -10,   0,   0,  0,  0,   0,   0, -10,
        -10,   5,   5,  5,  5,   5,   0, -10,
          0,   0,   5,  5,  5,   5,   0,  -5,
         -5,   0,   5,  5,  5,   5,   0,  -5,
        -10,   0,   5,  5,  5,   5,   0, -10,
        -10,   0,   0,  0,  0,   0,   0, -10,
        -20, -10, -10, -5, -5, -10, -10, -20],

"E" : [ -20, -10, -10, -5, -5, -10, -10, -20,
        -10,   0,   0,  0,  0,   0,   0, -10,
        -10,   5,   5,  5,  5,   5,   0, -10,
          0,   0,   5,  5,  5,   5,   0,  -5,
         -5,   0,   5,  5,  5,   5,   0,  -5,
        -10,   0,   5,  5,  5,   5,   0, -10,
        -10,   0,   0,  0,  0,   0,   0, -10,
        -20, -10, -10, -5, -5, -10, -10, -20]}
```

Queen

```
"O" : [  20,  30,  10,   0,   0,  10,  30,  20,
         20,  20,   0,   0,   0,   0,  20,  20,
        -10, -20, -20, -20, -20, -20, -20, -10,
        -20, -30, -30, -40, -40, -30, -30, -20,
        -30, -40, -40, -50, -50, -40, -40, -30,
        -30, -40, -40, -50, -50, -40, -40, -30,
        -30, -40, -40, -50, -50, -40, -40, -30,
        -30, -40, -40, -50, -50, -40, -40, -30],

"M" : [ -30, -40, -40, -50, -50, -40, -40, -30,
        -30, -40, -40, -50, -50, -40, -40, -30,
        -30, -40, -40, -50, -50, -40, -40, -30,
        -30, -40, -40, -50, -50, -40, -40, -30,
        -20, -30, -30, -40, -40, -30, -30, -20,
        -10, -20, -20, -20, -20, -20, -20, -10,
         20,  20,   0,   0,   0,   0,  20,  20,
         20,  30,  10,   0,   0,  10,  30,  20],

"E" : [ -50, -40, -30, -20, -20, -30, -40, -50,
        -30, -20, -10,   0,   0, -10, -20, -30,
        -30, -10,  20,  30,  30,  20, -10, -30,
        -30, -10,  30,  40,  40,  30, -10, -30,
        -30, -10,  30,  40,  40,  30, -10, -30,
        -30, -10,  20,  30,  30,  20, -10, -30,
        -30, -30,   0,   0,   0,   0, -30, -30,
        -50, -30, -30, -30, -30, -30, -30, -50]}
```

King

5.6.2 Pesto's Evaluation Function

```
mg_pawn_table = [
      0,   0,   0,   0,   0,   0,  0,   0,
     98, 134,  61,  95,  68, 126, 34, -11,
     -6,   7,  26,  31,  65,  56, 25, -20,
    -14,  13,   6,  21,  23,  12, 17, -23,
    -27,  -2,  -5,  12,  17,   6, 10, -25,
    -26,  -4,  -4, -10,   3,   3, 33, -12,
    -35,  -1, -20, -23, -15,  24, 38, -22,
      0,   0,   0,   0,   0,   0,  0,   0,
]

eg_pawn_table = [
      0,   0,   0,   0,   0,   0,   0,   0,
    178, 173, 158, 134, 147, 132, 165, 187,
     94, 100,  85,  67,  56,  53,  82,  84,
     32,  24,  13,   5,  -2,   4,  17,  17,
     13,   9,  -3,  -7,  -7,  -8,   3,  -1,
      4,   7,  -6,   1,   0,  -5,  -1,  -8,
     13,   8,   8,  10,  13,   0,   2,  -7,
      0,   0,   0,   0,   0,   0,   0,   0,
]
```

Pawn

```
mg_knight_table = [
    -167, -89, -34, -49,  61, -97, -15, -107,
     -73, -41,  72,  36,  23,  62,   7,  -17,
     -47,  60,  37,  65,  84, 129,  73,   44,
      -9,  17,  19,  53,  37,  69,  18,   22,
     -13,   4,  16,  13,  28,  19,  21,   -8,
     -23,  -9,  12,  10,  19,  17,  25,  -16,
     -29, -53, -12,  -3,  -1,  18, -14,  -19,
    -105, -21, -58, -33, -17, -28, -19,  -23,
]

eg_knight_table = [
    -58, -38, -13, -28, -31, -27, -63, -99,
    -25,  -8, -25,  -2,  -9, -25, -24, -52,
    -24, -20,  10,   9,  -1,  -9, -19, -41,
    -17,   3,  22,  22,  22,  11,   8, -18,
    -18,  -6,  16,  25,  16,  17,   4, -18,
    -23,  -3,  -1,  15,  10,  -3, -20, -22,
    -42, -20, -10,  -5,  -2, -20, -23, -44,
    -29, -51, -23, -15, -22, -18, -50, -64,
]
```

Knight

```
mg_bishop_table = [
    -29,   4, -82, -37, -25, -42,   7,  -8,
    -26,  16, -18, -13,  30,  59,  18, -47,
    -16,  37,  43,  40,  35,  50,  37,  -2,
     -4,   5,  19,  50,  37,  37,   7,  -2,
     -6,  13,  13,  26,  34,  12,  10,   4,
      0,  15,  15,  15,  14,  27,  18,  10,
      4,  15,  16,   0,   7,  21,  33,   1,
    -33,  -3, -14, -21, -13, -12, -39, -21,
]

eg_bishop_table = [
    -14, -21, -11,  -8, -7,  -9, -17, -24,
     -8,  -4,   7, -12, -3, -13,  -4, -14,
      2,  -8,   0,  -1, -2,   6,   0,   4,
     -3,   9,  12,   9, 14,  10,   3,   2,
     -6,   3,  13,  19,  7,  10,  -3,  -9,
    -12,  -3,   8,  10, 13,   3,  -7, -15,
    -14, -18,  -7,  -1,  4,  -9, -15, -27,
    -23,  -9, -23,  -5, -9, -16,  -5, -17,
]
```

Bishop

```
mg_rook_table = [
     32,  42,  32,  51, 63,  9,  31,  43,
     27,  32,  58,  62, 80, 67,  26,  44,
     -5,  19,  26,  36, 17, 45,  61,  16,
    -24, -11,   7,  26, 24, 35,  -8, -20,
    -36, -26, -12,  -1,  9, -7,   6, -23,
    -45, -25, -16, -17,  3,  0,  -5, -33,
    -44, -16, -20,  -9, -1, 11,  -6, -71,
    -19, -13,   1,  17, 16,  7, -37, -26,
]

eg_rook_table = [
    13, 10, 18, 15, 12,  12,   8,   5,
    11, 13, 13, 11, -3,   3,   8,   3,
     7,  7,  7,  5,  4,  -3,  -5,  -3,
     4,  3, 13,  1,  2,   1,  -1,   2,
     3,  5,  8,  4, -5,  -6,  -8, -11,
    -4,  0, -5, -1, -7, -12,  -8, -16,
    -6, -6,  0,  2, -9,  -9, -11,  -3,
    -9,  2,  3, -1, -5, -13,   4, -20,
]
```

Rook

```
mg_queen_table = [
    -28,   0,  29,  12,  59,  44,  43,  45,
    -24, -39,  -5,   1, -16,  57,  28,  54,
    -13, -17,   7,   8,  29,  56,  47,  57,
    -27, -27, -16, -16,  -1,  17,  -2,   1,
     -9, -26,  -9, -10,  -2,  -4,   3,  -3,
    -14,   2, -11,  -2,  -5,   2,  14,   5,
    -35,  -8,  11,   2,   8,  15,  -3,   1,
     -1, -18,  -9,  10, -15, -25, -31, -50,
]

eg_queen_table = [
     -9,  22,  22,  27,  27,  19,  10,  20,
    -17,  20,  32,  41,  58,  25,  30,   0,
    -20,   6,   9,  49,  47,  35,  19,   9,
      3,  22,  24,  45,  57,  40,  57,  36,
    -18,  28,  19,  47,  31,  34,  39,  23,
    -16, -27,  15,   6,   9,  17,  10,   5,
    -22, -23, -30, -16, -16, -23, -36, -32,
    -33, -28, -22, -43,  -5, -32, -20, -41,
]
```

Queen

```
mg_king_table = [
    -65,  23,  16, -15, -56, -34,   2,  13,
     29,  -1, -20,  -7,  -8,  -4, -38, -29,
     -9,  24,   2, -16, -20,   6,  22, -22,
    -17, -20, -12, -27, -30, -25, -14, -36,
    -49,  -1, -27, -39, -46, -44, -33, -51,
    -14, -14, -22, -46, -44, -30, -15, -27,
      1,   7,  -8, -64, -43, -16,   9,   8,
    -15,  36,  12, -54,   8, -28,  24,  14,
]

eg_king_table = [
    -74, -35, -18, -18, -11,  15,   4, -17,
    -12,  17,  14,  17,  17,  38,  23,  11,
     10,  17,  23,  15,  20,  45,  44,  13,
     -8,  22,  24,  27,  26,  33,  26,   3,
    -18,  -4,  21,  24,  27,  23,   9, -11,
    -19,  -3,  11,  21,  23,  16,   7,  -9,
    -27, -11,   4,  13,  14,   4,  -5, -17,
    -53, -34, -21, -11, -28, -14, -24, -43
]
```

King

5.7 Results

Here are the result of the game played against Chess.com AI at various levels:

5.7.1 Game 1

opponent (white) rating: 700

[White : Chess.com AI]
[Black : Our AI]
[Result : "*"]

1. e4 Nf6 2. e5 Nd5 3ḋ4 Nc6 4. Nf3 d6 5. h4 dxe5 6. Nxe5 Nxe5 7. dxe5 Be6 8.f3 f6 9. Bb5+ c6 10. a4 cxb5 11.axb5 fxe5 12. Na3 Qa5+ 13. Kf2 Qb6+ 14. Kg3 Qd4 15. Qxd4 exd4 16. c4 dxc3 17. bxc3 Nxc3 18. Rf1 Bd7 19. Bf4 Ne2+ 20. Kh2 *

r3kb1r/pp1bp1pp/8/1P6/5B1P/N4P2/4n1PK/R4R2 b kq - 5 20

5.7.2 Game 2

opponent (white) rating: 900

[White Chess.com AI]

[Black Our AI]

[Result "1-0"]

1. Nf3 Nf6 2. c4 d5 3. Ng5 dxc4 4. Nxf7 Kxf7 5. a4 Nc6 6. Qc2 Be6 7. Ra2 Nb4 8.Qc3 Nxa2 9. Qc2 Nxc1 10. Qxc1 Kg8 11. b4 cxb3 12. Rg1 Qd6 13. e3 Qxh2 14. Qc4 Bxc4 15. Bxc4+ e6 16. Bxe6# 1-0

r4bkr/ppp3pp/4Bn2/8/P7/1p2P3/3P1PPq/1N2K1R1 b - - 0 16

5.7.3 Game 3

opponent (white) rating: 1200

[White Chess.com AI]

[Black Our AI]

[Result "*"]

1. c4 d5 2. e3 Nf6 3. Nc3 Be6 4. b3 Nc6 5. f4 d4 6. Nb5 a6 7. Na3 Qd6 8. Nc2 d3 9. Na3 O-O-O 10. c5 Qd5 11. Qf3 Qxc5 12. e4 Qd4 13. Rb1 Qxe4+ 14. Qxe4 Nxe4 15. h4 Nb4 16. Nc4 Bxc4 17. Rb2 Be6 18. Rh2 Nc2+ 19. Rxc2 dxc2 20. Ne2 Rd3 21. Ng1 Rd8 22. d3 Nc5 23. g3 Nxd3+ 24. Bxd3 Rxd3 25. Rg2 Rd1+ 26. Kf2 Rxc1 27. Ne2 Rd1 28. Ke3 c1=Q+ 29. Nxc1 *

2k2b1r/1pp1pppp/p3b3/8/5P1P/1P2K1P1/P5R1/2Nr4 b - - 0 29

5.7.4 Game 4

opponent (white) rating: 1400

[White Chess.com AI]

[Black Our AI]

[Result "*"]

1. e4 Nc6 2. Nf3 e5 3. Bb5 Nd4 4. Nxd4 exd4 5. O-O Nf6 6. e5 Ne4 7. d3 Nc5 8. e6 fxe6 9. b4 a6 10. Bc4 d5 11. Bb3 Nxb3 12. axb3 Bxb4 13. Ba3 Bxa3 14. g3 Bb2 15. Ra2 Bc3 16. Nxc3 dxc3 17. Ra1 O-O 18. Qe1 d4 19. Qe5 Rf3 20. b4 Qd7 21. Rae1 b6 22. Qe4 Qd5 23. Qxd5 exd5 24. Re2 Bh3 25. Rc1 Rd8 26. Ra1 Bg4 27. Re7 a5 28. bxa5 Rd7 29. Rxd7 Bxd7 30. a6 Bb5 31. Kg2 Rf8 32. a7 Bc6 33. g4 Re8 34. g5 h6 35. g6 Bb7 36. Ra4 c5 37. f4 Re2+ 38. Kg3 Rxc2 39. a8=Q+ *

Q5k1/1b4p1/1p4Pp/2pp4/R2p1P2/2pP2K1/2r4P/8 b - - 0 39

5.7.5 Game 5

opponent (white) rating: 1400

[White Chess.com AI]

[Black Our AI]

[Result "1-0"]

1. Nf3 Nc6 2. d4 Nf6 3. a4 Ne4 4. c3 d5 5. e3 Be6 6. Ne5 Nxe5 7. dxe5 Rc8 8. f3 Nc5 9. b4 Nd7 10. f4 f6 11. Bb5 a6 12. Be2 fxe5 13. O-O exf4 14. Qd3 Ne5 15. Qd4 Nc6 16. Bh5+ Bf7 17. Bxf7+ Kxf7 18. Qxf4+ Kg8 19. Qf7# 1-0

2rq1bkr/1pp1pQpp/p1n5/3p4/PP6/2P1P3/6PP/RNB2RK1 b - - 2 19

5.7.6 Game 6

opponent (white) rating: 850

[White Chess.com AI]

[Black Our AI]

[Result "0-1"]

1. e4 Nc6 2. Nf3 e5 3. Bc4 Nf6 4. d4 exd4 5. O-O Nxe4 6. Re1 d5 7. Bxd5 Qxd5 8. Nc3 Qc4 9. Rxe4+ Kd8 10. Re3 Bd6 11. Nxd4 Nxd4 12. Bd2 Be6 13. Rg3 Bxg3 14. hxg3 Re8 15. Bg5+ f6 16. Be3 c5 17. Bxd4 cxd4 18. Rb1 Bd7 19. Qf3 dxc3 20. Qxb7 Ke7 21. Rd1 Rad8 22. Qd5 Qxd5 23. Rxd5 cxb2 24. Rd1 Kf8 25. Kh2 Re2 26. Rb1 Rxc2 27. Rxb2 Rxb2 28. a4 Rd2 29. Kg1 Rd1+ 30. Kh2 Bxa4 31. Kh3 R8d2 32. f3 Rg1 33. Kg4 Rdxg2 34. Kh5 Bc6 35. Kh4 Bxf3 36. Kh3 Rh1# 0-1

5k2/p5pp/5p2/8/8/5bPK/6r1/7r w - - 2 37

5.7.7 Game 7

opponent (black) rating: 850

[Black Chess.com AI]

[White Our AI]

Result "*"

1. Nc3 c5 2. d4 d6 3. e4 d5 4. Bb5+ Nc6 5. exd5 a5 6. dxc6 bxc6 7. Bxc6+ Bd7 8. Bxa8 Qxa8 9. Kf1 Nf6 10. dxc5 h5 11. Nf3 Qb7 12. Ne5 Ng4 13. f4 h4 14. Nxg4 Qa8 15. Ne5 Bc6 16. Qe2 f5 17. Ng6 Be4 18. Nxh8 g5 19. Nxe4 Qxe4 20. Qxe4 fxe4 21. Ng6 Kd7 22. Nxf8+ Kc6 23. Ne6 gxf4 24. Bxf4 h3 25. gxh3 Kd5 26. Ng5 Kc6 27. Nxe4 Kd7 28. Rd1+ Kc8 29. c6 e6 30. Nc5 a4 31. c7 e5 32. Bxe5 a3 33. Bd6 axb2 *

2k5/2P5/3B4/2N5/8/7P/PpP4P/3R1K1R w - - 0 34

5.7.8 Game 8

opponent (white) rating:1200

[White Chess.com AI]

[Black Our AI]

[Result "1-0"]

1. d4 Nc6 2. Nf3 Nf6 3. d5 Nb4 4. c3 Nbxd5 5. e4 Nb6 6. e5 Ne4 7. Qc2 d5 8. Bd3 Bf5 9. Nh4 e6 10. Nxf5 exf5 11. g3 Nc4 12. f3 Nc5 13. Qe2 Nxd3+ 14. Qxd3 Qd7 15. Nd2 Nxe5 16. Qe3 Bd6 17. c4 Qe6 18. c5 Nd3+ 19. Ke2 Qxe3+ 20. Kxe3 Nxc5 21. b4 Be5 22. Rb1 d4+ 23. Ke2 d3+ 24. Kf2 Bd4+ 25. Kg2 Na4 26. Rb3 Nc3 27. Bb2 Nb5 28. a4 Bxb2 29. Re1+ Kf8 30. Rxb2 Nc3 31. Re5 Nxa4 32. Rb3 Rd8 33. Ra5 Nb6 34. Rxa7 Kg8 35. Rxb7 Rd7 36. Rb8+ Rd8 37. Rxd8# 1-0

3R2kr/2p2ppp/1n6/5p2/1P6/1R1p1PP1/3N2KP/8 b - - 0 37

5.7.9 Game 9

opponent (black) rating:1200

[Black Chess.com AI]

[White Our AI]

[Result "*"]

1. Nc3 c5 2. d4 cxd4 3. Qxd4 Qc7 4. Bf4 d6 5. O-O-O e5 6. Nd5 Qd7 7. Qe3 Nc6 8. Kb1 Nge7 9. Nxe7 exf4 10. Qxf4 g5 11. Qf6 Qxe7 12. Qxh8 Qe5 13. Qxh7 Bf5 14. Qg8 Nd4 15. Rc1 Qg7 16. Qxg7 Bxg7 17. e3 Nc6 18. Nf3 g4 19. Ng5 Bh6 20. h4 Bxg5 21. hxg5 Rc8 22. Rh8+ Ke7 23. Rxc8 Bxc8 24. Rd1 Ne5 25. e4 f5 26. gxf6+ Kf7 27. Rxd6 Be6 28. Rd8 a6 29. Rb8 b5 30. Ra8 g3 31. fxg3 Nd7 32. Bd3 Bg4 33. Ra7 a5 34. Bxb5 Kxf6 35. Bxd7 Bxd7 36. Rxd7 Ke5 37. Re7+ Kd6 38. Rf7 Ke5 39. Re7+ Kf6 40. Rf7+ Kxf7 41. e5 Ke7 *

8/4k3/8/p3P3/8/6P1/PPP3P1/1K6 w - - 1 42

5.7.10 Game 10

opponent (black) rating:1000

[Black Chess.com AI]

[White Our AI]

[Result "1-0"]

1. Nc3 d5 2. e4 dxe4 3. Nxe4 Nf6 4. Nc5 Nbd7 5. Bb5 b6 6. Bc6 Rb8 7. Nxd7 Bxd7 8. Qf3 g5 9. Kf1 h5 10. d4 Bxc6 11. Qxc6+ Qd7 12. Qxd7+ Kxd7 13. Bxg5 Rc8 14. Re1 c5 15. dxc5 Rxc5 16. Rd1+ Kc6 17. Bxf6 exf6 18. Rc1 Rh6 19. Nf3 a6 20. Kg1 Rg5 21. Nxg5 fxg5 22. h4 g4 23. f3 Bc5+ 24. Kf1 gxf3 25. gxf3 Bd6 26. Kg1 Bc5+ 27. Kf1 Be3 28. Re1 Bc5 29. Re8 Rd6 30. Re7 Rd5 31. Rxf7 Rd6 32. a3 Rd2 33. b4 Kb5 34. bxc5 bxc5 35. c3 Rd1+ 36. Kg2 Rd3 37. Rb1+ Ka5 38. Rc1 Ka4 39. Rc7 Kxa3 40. Rxc5 Rd2+ 41. Kg1 Kb2 42. Re1 Rc2 43. Re6 Rc1+ 44. Kg2 Rxc3 45. Rb6+ Rb3 46. Rxa6 Rxf3 47. Kxf3 Kb1 48. Kg2 Kb2 49. Rb6+ Ka1 50. Ra5# 1-0

8/8/1R6/R6p/7P/8/6K1/k7 b - - 6 50

5.7.11 Game 11

opponent (white) rating:1000

[White Chess.com AI]

[Black Our AI]

[Result "*"]

1. e4 Nc6 2. Nc3 e5 3. Bc4 Nf6 4. d3 Bb4 5. Bxf7+ Kxf7 6. Qd2 d5 7. exd5 Bxc3 8. bxc3 Nxd5 9. Rb1 Kg8 10. c4 Nb6 11. Ba3 Bg4 12. c3 e4 13. dxe4 Nxc4 14. Qc2 Nxa3 15. Qb3+ Nc4 16. Qxc4+ Kf8 17. f4 Na5 18. Qb4+ c5 19. Qxc5+ Kg8 20. Qd5+ Qxd5 21. exd5 Rd8 22. a4 Rxd5 23. Rxb7 Rd8 24. Rc7 Rd1+ 25. Kf2 Nb7 26. Rxb7 Rd7 27. Rb8+ Rd8 28. Rxd8+ *

3R2kr/p5pp/8/8/P4Pb1/2P5/5KPP/6NR b - - 0 28