COSC 3P71

TERM PROJECT- CHESS

INSTRUCTOR: DR. BEATRICE OMBUKI

CREATED BY:

IFTI AHMED - 6559348

ANWAR HASHMI - 6584957

A picture containing object, chessman, indoor, set

Description automatically generated

**CONTENTS:**

1) INTRODUCTION

2) USAGE GUIDELINES

3) INTERFACE

4) SYSTEM DESIGN DESCRIPTION

5) ALPHA-BETA (Minimax Algorithm with Alpha-Beta Pruning)

6) HEURISTIC FUNCTION

7) REFERENCES

**INTRODUCTION:**

The aim of this project is to provide the user a functional command line chess game supporting human vs human as well as human vs AI component. We have chosen to program this game in JAVA. We have not used any existing libraries and have implement the game completely from scratch. Our code comprises a main class which is the initial class connecting both AI and human vs human. We have separate classes for individual pieces defining their legal moves and applying constraints. In the player/AI class we connected all the individual classes are created a user-friendly board.

**USAGE GUIDELINES:**

The user has many options in a real chess game, and we have implemented the same. When the game begins, the user can choose to play a multiplayer game or with an AI.

Text

Description automatically generated

When it is players turn, they are first presented with a menu to choose which piece to move. We have additionally included a user guide here which they can reference as well for instructions. One the player selects a piece they are prompted to choose the steps and direction of movement.

Text

Description automatically generated

For certain pieces like king which have a fixed limit a higher input of steps will not be recognized. For pieces with multiple options, they can choose further which one to move.

We have also incorporated special move such as Castling and en-passant which are available to the user.

Text

Description automatically generatedText

Description automatically generated

**INTERFACE:**

The interface is represented as a board on the output terminal.

The board itself is represented a 2d array. While the user is given the option to choose the board dimension the optimal dimension for the board is 8.

A newly generated board will contain 8 pawns, 2 rooks, 2 knights, 2 bishops, 1 queen and 1 king for both black and white.

The white pieces are represented with capital letters

(Pawn-P, Rook-R, Knight-K, Bishop-B, Queen-Q, King-I) and black with

lower case (Pawn-p, Rook-r, Knight-k, Bishop-b, Queen-q, King-i ).

Background pattern

Description automatically generated with medium confidence

When a move is made the previous position of the piece is updated to empty (-).

To ensure the user doesn’t cheat each selected piece only display directions it can travel on.

**SYSTEM DESIGN DESCRIPTION:**

The board has built in constraints to prevent the player from choosing a direction/step which would take a piece off the board. For each piece multiple constraints have been applied. For example: a king can only move 1 step, a bishop can only move diagonally, a piece cannot go over another of same color. As with all other moves constraints are applied to special moves such as en-passant can only occur once for each pawn. Additionally, if the user makes an invalid move (entering 3 steps for king) the move is not valid and user is prompted to enter a new move.

There are 2 ways to end the game. Each time a player plays their turn the game automatically checks to see if a checkmate has occurred and the king is killed, which if true ends the game. A second option has been provided to both players on each turn to surrender resulting in the opponent winning. After every move the board is simultaneously update and printed. If a piece is killed it is eliminated from the board. If the king is killed the game ends.

**ALPHA-BETA (Minimax Algorithm with Alpha-Beta Pruning)**

First, we will take the value of alpha, beta, eri array, depth, MaximizingPlayer, value as parameters in the AlphaBeta method. So, value represents any random value between 0 and 6. These random values represents each of the pieces in the chess game.

So, after the parameters are passed, then our base condition of the method will be that if the value of depth is 0, then, we will check the heuristic function to see if the game has ended or not. If the heuristic function returns true, then the game has ended. Otherwise, this function will return a value and the game will continue.

Here, player is a maximizer and AI is a minimizer. So, initially, the value of MaximizingPlayer is passed as false to the method. So, it will go to the condition where, (MaximizingPlayer==false) in the method and then, the value will be passed to the condition where, (MaximizingPlayer==false) if the variable value is <=alpha. Then, again, the value will be passed to the condition where, (MaximizingPlayer==true) if the variable value is >=beta. This will continue, until the depth becomes 0(base condition is reached).

**HEURISTIC FUNCTION**

So, I divided my heuristic function into 2 functions (heuristicFunction () and Checkmate() ).

So, the heuristicFunction () will take the array (eri) as parameter and then scan through the whole array to check whether the character ‘I’ (which is the King piece of Human Player) is there or not. If the character ‘I’ is there, this means the King piece of the human player is alive and the game continues. But, if after searching through the whole array(eri), the variable ‘I’ is not found, this means the King of the human player is checkmated and out of game. This means, the AI has won the game.

So, the Checkmate () will take the array(eri) and the value of king coordinates (m1 and n1) as parameters. Then, this method will search through the whole array in order to check whether the character ‘i’ (which is the King piece of AI) is there or not. If the character ‘i’ is there, this means the King piece of the AI player is alive and the game continues. But, if after searching through the whole array(eri), the variable ‘i’ is not found, this means the King of the AI player is checkmated and out of game. This means, the Human Player has won the game.

**REFERENCES**

* [**https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/**](https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/)
* [**https://en.wikipedia.org/wiki/Chess**](https://en.wikipedia.org/wiki/Chess)
* [**https://www.chess.com**/](https://www.chess.com/)