Department of Computer Science and Engineering

Course Code: CSE-3636
Course Title: Artificial Intelligence Lab
Project Title: AI based Chess Game using Python

Project Documentation

Week 4: Generating all possible moves

Submitted by

Lamisa Mashiat (C201249)

Sohana Tasneem (C201266)

Sadia Haque Chowdhury (C201270)

Submitted to

Subrina Akter

Assistant Professor

Dept. of CSE

8th April, 2023

With the latest update to the ChessEngine.py script, all chess pieces can now be moved in the game. This means that players can enjoy a more realistic and challenging game of chess with the ability to move any of their pieces in a strategic manner.

ChessEngine.py

1. **getRookMoves(r, c, moves)**: This method generates all possible moves for a rook located at position **(r, c)** on the board. It considers all possible directions in which the rook can move (up, down, left, and right) and checks if the moves are valid. A move is valid if the end position is on the board, and either it is an empty square or an opponent's piece is occupying it.
2. **getKnightMoves(r, c, moves)**: This method generates all possible moves for a knight located at position **(r, c)** on the board. It considers all possible squares that a knight can move to, which are two squares vertically/horizontally followed by one square in the other direction. A move is valid if the end position is on the board, and either it is an empty square or an opponent's piece is occupying it.
3. **getBishopMoves(r, c, moves)**: This method generates all possible moves for a bishop located at position **(r, c)** on the board. It considers all possible diagonals in which the bishop can move and checks if the moves are valid. A move is valid if the end position is on the board, and either it is an empty square or an opponent's piece is occupying it.
4. **getQueenMoves(r, c, moves)**: This method generates all possible moves for a queen located at position **(r, c)** on the board. It considers all possible directions in which a queen can move (which are the same as for a rook or a bishop) and checks if the moves are valid. It calls **getRookMoves()** and **getBishopMoves()** methods to achieve this.
5. **getKingMoves(r, c, moves)**: This method generates all possible moves for a king located at position **(r, c)** on the board. It considers all possible squares that a king can move to (which are one square in any direction). A move is valid if the end position is on the board, and either it is an empty square or an opponent's piece is occupying it. It also checks whether the move would result in the king being under attack and eliminates such moves.