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 5: Check, checkmate & stalemate

Submitted by

Lamisa Mashiat (C201249)

Sohana Tasneem (C201266)

Sadia Haque Chowdhury (C201270)

Submitted to

Subrina Akter

Assistant Professor

Dept. of CSE

15th April, 2023

The ChessEngine.py script has been updated to include functionality for detecting check, checkmate, and stalemate. The getValidMoves() method now considers checks before generating all possible moves, allowing it to remove any moves that would cause the current player's king to be attacked. The inCheck() method determines if the current player is in check by checking if the square containing the player's king is under attack by the opponent's pieces. The squareUnderAttack() method checks if a given square is being attacked by the opponent's pieces.

The ChessEngine script now has the following attributes:

- self.whiteKingLocation: tuple containing the current location of the white king
- self.blackKingLocation: tuple containing the current location of the black king
- self.checkMate: boolean indicating if the game is in checkmate
- self.staleMate: boolean indicating if the game is in stalemate

The following methods have been added to the ChessEngine script:

- getValidMoves(): Returns a list of all valid moves for the current player, taking into account checks, checkmates, and stalemates.
- inCheck(): Returns True if the current player is in check, and False otherwise.
- squareUnderAttack(r, c): Returns True if the square at row r and column c is under attack by the opponent's pieces, and False otherwise.

Method Details:

1. **getValidMoves():** This method returns a list of all valid moves for the current player, taking into account checks, checkmates, and stalemates. It performs the following steps:
   a) Generates all possible moves.
   b) For each move, it makes the move and generates all opponent's moves.
   c) For each of the opponent's moves, it checks if they attack the current player's king. If they do, the move is not valid and is removed from the list of all possible moves.
   d) If there are no valid moves left, it sets the checkMate or staleMate attribute to True depending on whether the player is in check or not.
   e) Returns the list of valid moves.

2. **inCheck():** This method determines if the current player is in check by checking if the square containing the player's king is under attack by the opponent's pieces. It performs the following steps:

   a) If the current player is white, it checks if the square containing the white king is under attack.

   b) If the current player is black, it checks if the square containing the black king is under attack.

   c) Returns True if the king is in check, and False otherwise.

3. **squareUnderAttack(r, c):** This method checks if a given square is being attacked by the opponent's pieces. It performs the following steps:

   a) Switches to the opponent's turn.

   b) Generates all possible moves for the opponent.

   c) Switches back to the current player's turn.

   d) Checks if any of the opponent's moves attack the given square.

   e) Returns True if the square is under attack, and False otherwise.