chessPieceLogic

Created by: peterdownie.com

Software Version: 2.41e

Greater Code Name: Atomic

Lesser Code Name: Proton

Service Version: 1.04

Purpose

* Chess Piece logic calculates legal moves or just makes them from a fen or arrays

lowerText

technicalSpecifications

To use this webservice you will need to call one of two functions.startingPieces Or setPieceArrayBasically they create a list of pieces with positions. When you want to save this datato your database(not provided in this webservice) you call getPieceArray. return[1] is white and return[2] is black in feed.You can check to make sure that a move is legal before you commit it with checkForLegalMove and if you want to seeall legal possible moves you can view netmoves. To change data 1 at a time you can use makeAMove and if you wantto change multiple use multiMove.

Main Methods

startNewGame
generate_fen
set_fen
__construct
getPieceArray
setPieceArray
makeAMove
multiMove
netMoves

descriptionOf_pawn
descriptionOf_knight
descriptionOf_bishop
descriptionOf_rook
descriptionOf_queen
descriptionOf_king
descriptionOf_multiPiece
descriptionOf_all_moves
checkForLegalMove
descriptionOf_calculateCastleMoves
descriptionOf_getLastEnemyMove
descriptionOf_getLastMove
descriptionOf_noMoves_STALEMATE_OR_CHECKMATE_Detect

Method Details

startNewGame

* This is the fen when the game starts
 * This replaces the old version

generate_fen

Gets the current fen

set_fen

Parameter #0 [<required> $fen]

Set a fen!

__construct

Check to see if you can connect to webservice. Also returns the class name for smarter connection. You do not really need to call this, makes connection easier.

getPieceArray

get the array white and then the array black in 1 and 2 in feed
This is just here for legacy reasons, you should really just return the FEN

setPieceArray

Parameter #0 [<required> array $white]

Parameter #1 [<required> array $black]

Parameter #2 [<optional> $turn = 1]

Parameter #3 [<optional> $castlingAvialabilty = '']

Parameter #4 [<optional> $enpassant_move = '']

Parameter #5 [<optional> $halfMoveClock = '']

Parameter #6 [<optional> $fullMoveClock = '']

sets the pieces of white and black. You could alos use startingPieces instead if the game has not started.
You may now pass null and it will leave the pieces as they are for that color. If you leave both null you will get a warning.
Turn is either 1 or 2
Castling availablity must be KQkq(any combination in order) or - It now be blank for neither

makeAMove

Parameter #0 [<required> $color]

Parameter #1 [<required> $move]

Parameter #2 [<optional> $real = true]

MakeAMove takes an input and makes a move regardless if the move is legal or not. Used to return piece value. Now returns array of data."
. ". See tech.
 * Update! Returned to using 1 or 2
 * Setting real to false disables this function. This is used to find moves that are illegal, which would otherwise write the moves as made!

multiMove

Parameter #0 [<required> $color]

Parameter #1 [<required> $moves_csv]

MultiMove takes a move csv and calls makeAMove with the values. More Info in Tech. Returns an output see tech specs.

netMoves

Parameter #0 [<required> $color]

Parameter #1 [<required> $csvMoves]

netMoves calculates all legal moves and returns an array of moves. Returns in feed a2-a3,a2-a4,a7-a8=Q,e1-f1. Gets rid of illegalmoves
when the king is in check and cannot castle as well as enabling enpassante.

The csv Moves is for enpassante and such, it does not calculate from there as this uses addition computer resources.

descriptionOf_pawn

descriptionOf_knight

descriptionOf_bishop

descriptionOf_rook

descriptionOf_queen

descriptionOf_king

descriptionOf_multiPiece

descriptionOf_all_moves

checkForLegalMove

Parameter #0 [<required> $color]

Parameter #1 [<required> $move]

Parameter #2 [<required> $csvMoves]

Takes a move and makes sure that it is legal, does not check all of the other moves to save processing power
checkForLegalMove takes a move in the format square1-square2, ie a2-a3... The net moves takes a value from netMoves in an
array. If the value is in the array than the move is indeed legal. This function returns 1 of 3 values, the first is LEGAL_MOVE
and the second is ILLEGAL_MOVE. IF there are no moves than CHECKMATE or STALEMATE is given and the game is over

descriptionOf_calculateCastleMoves

descriptionOf_getLastEnemyMove

descriptionOf_getLastMove

descriptionOf_noMoves_STALEMATE_OR_CHECKMATE_Detect

Method Technical Specifications

startNewGame

This replaces the old starting pieces, has less complexity and returns the value of set FEN.

generate_fen

No technical specifications found

set_fen

No technical specifications found

__construct

No technical specifications found

getPieceArray

Possible output values
CANNOT_RETURN_EMPTY_BLACK_AND_WHITE (MINOR): Black and white pieces were not set. You should call startingPieces or setPieceArray.
OUTPUTTING_FEED(NOMINAL): feed will return [1] for white and [2] for black.

setPieceArray

Possible outputsPOINTLESS_CALL(MINOR): This will change not a thing.
PIECES_SET_SUCCESSFULLYNOMINAL: pieces modified successfully.

makeAMove

Color Possible Input Values
1: The color is white
2: The color is black

white: Not case sensitive, the color is white!
black: Not case sensitive. The color is black!
Move Possible inputs
should be of the format a2-a3, rank(small case) file then dash than the rank and file to without spaces. If you reach a promotion square than it must have the format equals and than the upper case piece. Not square is variable and not static like the examples. Castling will be explainedbelow promotion
Possible Promote Pieces
e7-e8=N: Promote to kNight. Notice this is not K which is King!
a2-a1=B: Promote to Bishop
h2-h1=R: Promote to Rook
h7-h8=Q: Promote to Queen
Castling
Castling is obtained when the king moves two squares instead of one to a preset square. This means if the program has been modified than it will not work likely. If the king is e and moves to g or if the king is e and moves c than it will castle. You need to make sure this is legal before doing it as strange results may occour if it is not possible.
Piece Arrays
contain all the pieces in an array'Pa2,'Pa3...'

multiMove

Moves must conform with the specifications detailed at makeAMove tech. No longer returns pieces
Possible outputs!
MULTIPLE_MOVES_MADE_SUCCESSFULLY(NOMINAL): Moves made.
MULTI_MOVE_FAILED:(VARIABLE): Errors, see error_message array element.

netMoves

Explanation of parameters required.
Color
1 or white: white, the string value is case insensitive.
2 or black: black, the string is case insensitive.

CsvMoves: Must be in the form of e2-e4,e7-e5,g1-f3,b8-c6
White/Black pieces array takes an array of black pieces in the form Pa2,Pb2,...Nb1..

descriptionOf_pawn

No technical specifications found

descriptionOf_knight

No technical specifications found

descriptionOf_bishop

No technical specifications found

descriptionOf_rook

No technical specifications found

descriptionOf_queen

No technical specifications found

descriptionOf_king

No technical specifications found

descriptionOf_multiPiece

No technical specifications found

descriptionOf_all_moves

No technical specifications found

checkForLegalMove

Explanation of Inputs first and then returns.
color1 or white: white, the string is case insensitive
2 or black: black, the string is case insensitve.

Move: The move you intend to make, must be in the format of a2-a3, a7-a8=Q

csvMoves: moves in csv of the same format as move.

White/Black Pieces array is an array of pieces Pa2,Pb2...Possible outputs!

LEGAL_MOVE(NOMINAL): The move is legal!

ILLEGAL_MOVE(NOMINAL): The move is not legal!

CHECK_FOR_LEGAL_MOVE_FAILURE(VARIABLE): Function hfailed, see error_message array element.

descriptionOf_calculateCastleMoves

No technical specifications found

descriptionOf_getLastEnemyMove

No technical specifications found

descriptionOf_getLastMove

No technical specifications found

descriptionOf_noMoves_STALEMATE_OR_CHECKMATE_Detect

No technical specifications found