Chess AI Player Task 15: Create the Random Chess Player

Abstract : The goal of this task was to create the random player. This player creates a list of all of
the possible moves and randomly selects one of them. The random player is capable of playing
against a human or another random player.

Demo:

480185										- ļ
480186										!
480187	8	BR	BN	BB	BK	BQ	BB	BN	BR	!
480188										!
480189	7	BP	BP	BP	BP	BP	BP	BP	BP	ļ.
480190										1
480191	6									1
480192										1
480193	5									
480194										
480195	4									Ι_
480196										10
480197	3									
480198										1
480199	2	WP	WP	WP	WP	WP	WP	WP	WP	1
480200										1
480201	1	WR	WN	WB	WQ	WK	WB	WN	WR	1
480202										1
480203										-
480204		Α	В	C	D	Е	F	G	Н	
480205										
400000	-									
480206	Т									
480206		-USER>	(pl	ay-g	ame-	-rr)				
		-USER>	(pl	ay-g	ame-	-rr)				-
480207		-USER>	(p1	ay-g	ame-	-rr) 				-
480207 480208		-USER> BR	(pl	. ay-g 	ame-	-rr) 	BB	BN	BR	1
480207 480208 480209	CL-	 						BN	BR	
480207 480208 480209 480210	CL-	 						BN BP	BR BP	-
480207 480208 480209 480210 480211	CL ·	BR	BN	ВВ	BK	BQ	BB			
480207 480208 480209 480210 480211 480212	CL ·	BR	BN	ВВ	BK	BQ	BB			-
480207 480208 480209 480210 480211 480212 480213	8 7	BR	BN	ВВ	BK	BQ	BB			-
480207 480208 480209 480210 480211 480212 480213 480214	8 7	BR	BN	ВВ	BK	BQ	BB			-
480207 480208 480209 480210 480211 480212 480213 480214 480215 480216 480217	8 7 6	BR	BN	ВВ	BK	BQ	BB			-
480207 480208 480209 480210 480211 480212 480213 480214 480215 480216	8 7 6	BR	BN	ВВ	BK	BQ	BB			-
480207 480208 480209 480210 480211 480212 480213 480214 480215 480216 480217	8 7 6 5	BR	BN	ВВ	BK	BQ	BB			- - - - - - - -
480207 480208 480209 480210 480211 480212 480213 480214 480215 480216 480217 480218	8 7 6 5	BR	BN	ВВ	BK	BQ	BB			-
480207 480208 480209 480210 480211 480212 480213 480214 480215 480216 480217 480218 480219	8 7 6 5	BR	BN	ВВ	BK	BQ	BB BP 			-
480207 480208 480209 480210 480211 480212 480213 480214 480215 480216 480217 480218 480220	8 7 6 5 4 3	BR	BN	ВВ	BK	BQ	BB BP 			-
480207 480208 480209 480210 480211 480212 480213 480215 480215 480217 480218 480219 480220 480221	8 7 6 5 4 3	BR BP 	BN BP	BB BP	BK BP 	BQ BP	BB BP 	BP	BP	
480207 480208 480209 480210 480211 480212 480213 480214 480215 480216 480217 480218 480220 480221 480222	8 7 6 5 4 3	BR BP 	BN BP	BB BP	BK BP 	BQ BP	BB BP 	BP	BP	
480207 480208 480209 480210 480211 480212 480213 480214 480215 480216 480217 480218 480220 480221 480222 480223	8 7 6 5 4 3	BR BP WP	BN BP 	BB BP	BK BP WP	BQ BP 	BB BP WP	BP WP	BP	- - - - - - - - -
480207 480208 480209 480210 480211 480212 480214 480215 480216 480217 480218 480220 480221 480222 480223 480224	8 7 6 5 4 3	BR BP WP	BN BP 	BB BP	BK BP WP	BQ BP 	BB BP WP	BP WP	BP	- - - - - -
480207 480208 480209 480210 480211 480212 480213 480214 480215 480217 480218 480219 480220 480221 480222 480223 480224 480225	8 7 6 5 4 3	BR BP WP	BN BP 	BB BP	BK BP WP	BQ BP 	BB BP WP	BP WP	BP	

8	 BR	BN	ВВ	ВК	BQ	ВВ	BN	BR
7	l BP	ВР	ВР	ВР	ВР	ВР	ВР	
6								ВР
5								
4								
3						WP		
2	 WP	WP	WP	WP	WP		WP	WP
1	 WR	WN	WB	WQ	WK	WB	WN	WR
	Α	В	С	D	Ε	F	G	Н
8	BR 	BN	BB	ВК	BQ	BB	BN	BR
7	BP	BP	BP	BP	BP	BP	BP	
6								BP
5								
4								
3					WP	WP		
2	 WP	WP	WP	WP			WP	WP
1	 WR	WN	WB	WQ	WK	WB	WN	WR
	Α	В	С	D	E	F	G	Н
	l							

8						BK	BN	WN
7		BB					BB	
6	 WR				WN	ВР		
5	 	WP		WK				WP
4	 	WP		ВР		WQ	WP	
3	i WP	BN						
2			ВР		ВР		WB	
	i				DF		WD	
1	WR 		WB					
	 A	 В		D	 E	 F	 G	Н
						DI.		
8						BK	BN	WN
7	ļ						BB	
6	WR				WN	BP		
5		WP		ВВ				WP
4	 	WP		ВР		WQ	WP	
3	 WP	BN						
2	į		ВР		ВР		WB	
					DP		WD	
1	WR 		WB					
		 B		D.	 F	 F		н
					Ē	Ċ	J	
GAI	PLAYER ME OVE		ıs					
NII CL	L -USER>							

Code:

```
( defun random-move ( move-pairs &aux curr-square dest
square selected )
  ( setf selected ( nth ( random ( length move-pairs ) )
move-pairs ) )
  ( setf curr-square ( car selected ) )
  ( setf dest-square ( car ( cdr selected ) ) )
  ( move curr-square dest-square )
( defmethod get-move-pair-list ( ( piece piece ) &aux
curr-square poss-dests )
  ( setf curr-square ( cs piece ) )
  ( setf poss-dests ( possible-moves piece ) )
  ( mapcar ( lambda ( dest ) ( list curr-square dest ) )
poss-dests )
( defun play-turn--rr ()
  ( if ( game-overp )
    ( progn
       ( format t "GAME OVER" )
      nil
     ( progn
       ( random-white-move )
       ( if ( game-overp )
        ( progn
           ( format t "GAME OVER" )
           nil
```

```
( progn
          ( random-black-move )
          ( play-turn--rr )
      )
 )
( defun play-turn--hr ( color &aux curr-square csr csf
dest-square dsr dsf )
  ( if ( game-overp )
    ( progn
      ( format t "GAME OVER" )
      nil
    ( progn
      ( format t "It is the ~A player's turn~%" color )
      ( format t "Enter start square: " )
      ( setf curr-square ( parse-square ( string-trim " "
(read-line))))
      ( setf csr ( car curr-square ) )
      ( setf csf ( car ( cdr curr-square ) ) )
      ( format t "Enter end square: " )
      ( setf dest-square (parse-square (string-trim " "
(read-line))))
      ( setf dsr ( car dest-square ) )
      ( setf dsf ( car ( cdr dest-square ) ) )
      ( setf curr-square ( aref ( board *gameboard* ) csf
csr ) )
      ( setf dest-square ( aref ( board *gameboard* ) dsf
dsr ) )
      ( move curr-square dest-square )
```

```
( random-black-move )
      ( play-turn--hr 'w )
)
( defun play-game--hr ()
  ( play-turn--hr 'w )
( defun play-game--rr ()
  ( play-turn--rr )
( defun random-black-piece ()
  ( nth ( random ( length *black-pieces* ) ) *black-pieces*
( defun random-black-move ()
  ( random-move ( get-all-color-moves 'b ) )
)
( defun random-white-move ()
  ( random-move ( get-all-color-moves 'w ) )
)
( defun pieces-of-color ( color )
  ( cond
    ( ( eq color 'w ) *white-pieces* )
    ( ( eq color 'b ) *black-pieces* )
```

```
( defun oppo-pieces-of-color ( color )
  ( cond
    ( ( eq color 'b ) *white-pieces* )
    ( ( eq color 'w ) *black-pieces* )
)
(defun get-oppo-color-moves (color)
  (cond
    ( ( eq color 'w ) ( get-all-color-moves 'b ) )
    ( ( eq color 'b ) ( get-all-color-moves 'w ) )
( defun oppo-color ( color )
  ( cond
    ( ( eq color 'w ) 'b )
    ( ( eq color 'b ) 'w )
)
```