Chess Playing Machine

Kieran Finnegan

Original Project Goals

1. Creation of the board and pieces
2. Implement the rules of chess
3. Create a interface in order to play other players
4. Create a random player
5. Implement the minimax algorithm
6. Add alpha-beta pruning

What Was Actually Created

1. Creating the pieces and the chess board
2. Implemented most rules of chess
3. Created a interface in order to play
4. Created 3 different players
    a. Random
    b. Material player
    c. Location player

Representing The Board

- Originally used a 1D array of size 64
- This cause problems so I switched to a 2D array that was 8 x 8
- Created a board object to hold the array
- Created square object to storing in the array

```
10
11  ( defclass square ()
12    (
13      ( rank :initarg :rank :accessor rank )
14      ( file :initarg :file :accessor file )
15      ( occupied-by :accessor occupier :initform nil )
16      ( northwest-square :accessor nw :initform nil )
17      ( north-square :accessor n :initform nil )
18      ( northeast-square :accessor ne :initform nil )
19      ( east-square :accessor e :initform nil )
20      ( southwest-square :accessor sw :initform nil )
21      ( south-square :accessor s :initform nil )
22      ( southeast-square :accessor se :initform nil )
23      ( west-square :accessor w :initform nil )
24      ( number :initarg :number :accessor number :initform nil )
25    )
26  )
27
```

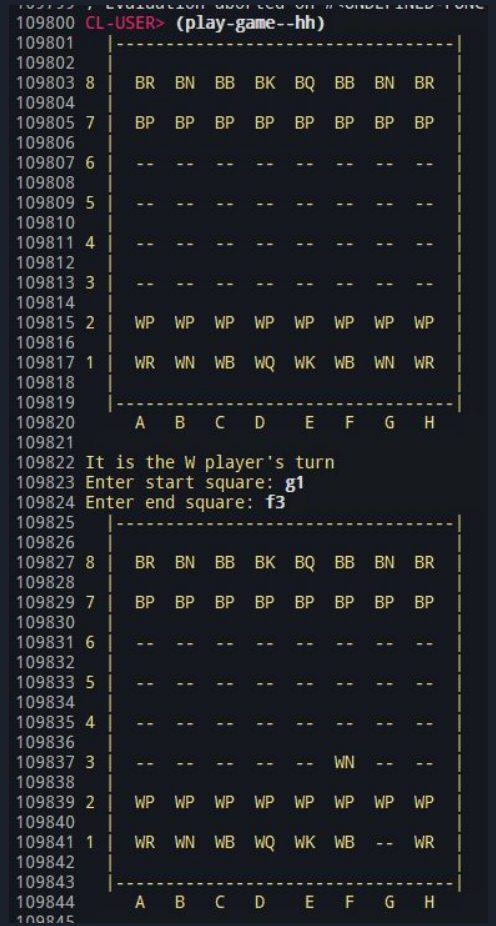
The Rules

- Take pieces of the other color
- Can’t move to or through square with your pieces on it
- The constraints that each piece type has when moving
- In order to get the legal moves of each piece
    - Legal-move method for each type
    - Possible-moves method for each type
        - Created a list of all the moves

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
( defmethod possible-moves ( ( piece piece ) )
  ( cond
    ( ( eq ( type piece ) 'pawn ) ( possible-moves-pawn piece ) )
    ( ( eq ( type piece ) 'queen ) ( possible-moves-queen piece ) )
    ( ( eq ( type piece ) 'king ) ( possible-moves-king piece ) )
    ( ( eq ( type piece ) 'rook ) ( possible-moves-rook piece ) )
    ( ( eq ( type piece ) 'knight ) ( possible-moves-knight piece ) )
    ( ( eq ( type piece ) 'bishop ) ( possible-moves-bishop piece ) )
  )
)
```

The Interface

- Very basic asks for the starting square and the ending square
- Each piece is represented with 2 letters
- Each Rank is numbered 1-8
- Each File is labeled A-H



The Random Player

- Makes move without considering anything about the board
- Creates a list of all the possible moves a color can make then randomly selects one from the list

```
(defmethod move ((curr-square square) (dest-square square) &aux color)
  (setf color (color (occupier curr-square)))
  (if (occupier dest-square)
      (remove-piece (occupier dest-square))
  )
  (if (legal-move (occupier curr-square) dest-square)
      (move-piece (occupier curr-square) dest-square)
    (format t "Invalid Move Chosen")
  )
)


(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 get-all-move-pair-list (pieces)
  (loop for piece in pieces
        append (get-move-pair-list piece))
)
```

Random Player Demo

```
CL-USER> (d)
  |---------------------------------|
  |                                 |
8 |  BR  BN  --  BK  --  BB  --  BQ |
  |                                 |
7 |  BP  --  BP  BR  --  --  --  -- |
  |                                 |
6 |  BB  --  --  --  --  --  --  -- |
  |                                 |
5 |  --  WN  --  --  BP  BP  --  BP |
  |                                 |
4 |  WP  WP  WP  BP  WP  WN  --  WP |
  |                                 |
3 |  --  --  --  WP  --  --  --  -- |
  |                                 |
2 |  --  --  --  BN  --  WP  --  WR |
  |                                 |
1 |  --  WR  WB  --  WK  WB  --  -- |
  |                                 |
  |---------------------------------|
    A   B   C   D   E   F   G   H

NIL
CL-USER> (possible-moves wrook2)
(#<SQUARE {10058AAEB3}> #<SQUARE {10058AA5B3}> #<SQUARE {10058AA9A3}>)
CL-USER>
```

The WR on H2 can only travel to 3 possible square

```
CL-USER> (d)
  |---------------------------------|
  |                                 |
8 |  BR  --  --  --  --  --  --  -- |
  |                                 |
7 |  BN  --  --  --  --  BP  BB  BR |
  |                                 |
6 |  BB  --  BK  BP  --  BN  --  -- |
  |                                 |
5 |  --  BP  WP  --  --  BP  BP  BP |
  |                                 |
4 |  WP  BP  --  --  --  --  WP  WQ |
  |                                 |
3 |  WB  --  WP  --  --  --  --  WB |
  |                                 |
2 |  WR  --  BQ  WK  --  --  --  WP |
  |                                 |
1 |  --  WN  --  --  --  --  --  -- |
  |                                 |
  |---------------------------------|
    A   B   C   D   E   F   G   H

NIL
CL-USER> (play-turn--r 'b)
T
CL-USER> (d)
  |---------------------------------|
  |                                 |
8 |  BR  --  --  --  --  --  --  -- |
  |                                 |
7 |  BN  --  --  --  --  BP  BB  BR |
  |                                 |
6 |  BB  --  BK  BP  --  BN  --  -- |
  |                                 |
5 |  --  BP  WP  --  --  BP  BP  BP |
  |                                 |
4 |  WP  BP  --  --  --  --  WP  WQ |
  |                                 |
3 |  WB  --  WP  --  --  --  --  WB |
  |                                 |
2 |  WR  BQ  --  WK  --  --  --  WP |
  |                                 |
1 |  --  WN  --  --  --  --  --  -- |
  |                                 |
  |---------------------------------|
    A   B   C   D   E   F   G   H

NIL
CL-USER>
```

The BQ one C2 moves to B3 instead of capturing the king

The Material Player

- Give each piece type a value
- Gives a score to each possible move based on the how the other color is affected
- Selects the move that will lower the score of the other player the most

Material Player Demo

```
CL-USER> (d)
   |---------------------------------|
   |                                 |
8  |   --  BR  --  --  --  BB  --  BR  |
   |                                 |
7  |  BP  --  --  BP  --  BN  --  --  |
   |                                 |
6  |  BN  WB  --  WN  --  --  --  --  |
   |                                 |
5  |   --  --  BP  --  --  BQ  WP  BP  |
   |                                 |
4  |   --  --  WP  --  --  BP  --  --  |
   |                                 |
3  |   --  WP  --  --  --  WP  WP  --  |
   |                                 |
2  |  WP  --  --  --  WP  WK  --  --  |
   |                                 |
1  |   --  WR  --  WQ  --  WB  WN  WR  |
   |                                 |
   |---------------------------------|
      A   B   C   D   E   F   G   H

NIL
CL-USER> (moves-with-lowest-score 'w)
Score: 33
Score: 37
Score: 35
Score: 37
Score: 35
Score: 35
Score: 35
Score: 37
((#<SQUARE {1006445FB3}> #<SQUARE {1006444B73}>))
CL-USER>
```

Only returns a single move, which is the best

```
CL-USER> (d)
   |---------------------------------|
   |                                 |
8  |  BR  --  --  --  --  BB  --  --  |
   |                                 |
7  |   --  --  BN  --  --  BP  --  BR  |
   |                                 |
6  |   --  --  BN  --  BP  --  --  --  |
   |                                 |
5  |  WN  BP  --  BK  --  --  BP  BP  |
   |                                 |
4  |  WP  WN  --  WP  --  --  --  --  |
   |                                 |
3  |   --  --  --  --  WB  BB  WP  WP  |
   |                                 |
2  |   --  WP  WQ  --  --  --  WB  --  |
   |                                 |
1  |  WR  --  --  BQ  --  --  --  WR  |
   |                                 |
   |---------------------------------|
      A   B   C   D   E   F   G   H

NIL
CL-USER> (play-turn--m 'w)
T
CL-USER> (d)
   |---------------------------------|
   |                                 |
8  |  BR  --  --  --  --  BB  --  --  |
   |                                 |
7  |   --  --  BN  --  --  BP  --  BR  |
   |                                 |
6  |   --  --  BN  --  BP  --  --  --  |
   |                                 |
5  |  WN  BP  --  WN  --  --  BP  BP  |
   |                                 |
4  |  WP  --  --  WP  --  --  --  --  |
   |                                 |
3  |   --  --  --  --  WB  BB  WP  WP  |
   |                                 |
2  |   --  WP  WQ  --  --  --  WB  --  |
   |                                 |
1  |  WR  --  --  BQ  --  --  --  WR  |
   |                                 |
   |---------------------------------|
      A   B   C   D   E   F   G   H

NIL
```

WN at B4 takes BK at D5

Material Player Code

```
( defun moves-with-lowest-score (color &aux temp-pieces  best-moves opposite-color-moves opposite-colo
 ( setf opposite-color ( oppo-color color ) )
 ( setf opposite-color-moves ( oppo-pieces-of-color color ) )
 ( setf all-move-pairs ( get-all-move-pair-list opposite-color-moves ) )
 ( setf min-score 100000 )
 ( setf best-moves all-move-pairs )
   ( dolist ( move-pair all-move-pairs )
     ( setf source ( car move-pair ) )
     ( setf destination ( car ( cdr move-pair ) ) )
     ( setf occupier-source ( occupier source ) )
     ( setf occupier-destination ( occupier destination ) )
     ( if ( not ( null occupier-destination ) )
       ( progn
         ( if ( eq ( type occupier-destination ) 'king )
           ( progn
             ( setf best-moves '() )
             ( push move-pair best-moves )
             ( return )
           )
         )
         ( setf temp-pieces ( remove occupier-destination ( pieces-of-color color )) )
         ( setf ( occupier destination ) occupier-source )
         ( setf ( cs occupier-source ) destination )
         ( setf ( occupier source ) nil )
         ( setf score ( compute-score color temp-pieces ) )
         ( if ( = score min-score )
           ( push move-pair best-moves )
         )
         ( if ( < score min-score )
           ( progn
             ( setf min-score score )
             ( setf best-moves '() )
             ( push move-pair best-moves )
           )
         )
       )
     )
       ( setf ( occupier source ) occupier-source )
       ( setf ( cs occupier-source ) source )
       ( setf ( occupier destination ) occupier-destination )

   )
   best-moves
)
```

```
( defun compute-wscore ( temp-pieces &aux score )
  ( setf score 0 )
  ( dolist ( wpiece temp-pieces )
    ( if ( not ( typep wpiece 'king ) )
      ( setf score ( + score ( val wpiece ) ) )
    )
  )
    score
)

( defun compute-bscore ( temp-pieces &aux score )
  ( setf score 0 )
  ( dolist ( bpiece temp-pieces )
    ( if ( not ( typep bpiece 'king ) )
      ( setf score ( + score ( val bpiece ) ) )
    )
  )
    score
)

( defun compute-score ( color temp-pieces )
  ( cond
    ( ( eq color 'b ) ( compute-bscore temp-pieces ) )
    ( ( eq color 'w ) ( compute-wscore temp-pieces ) )
  )
)
```

The Location Player

- Each piece type is given an array that represents value for all 64 squares on the board
- Adds the scores of all the squares that its pieces are on and then the square for each possible move
- Selects the move with the highest score

Location Player Demo

```
( setf *wbishop-table*
  ( make-array '(8 8) :initial-contents
    '( ( -2 -1 -1 -1 -1 -1 -1 -2 )
       ( -1  0  0  0  0  0  0 -1 )
       ( -1  0  0.5 1  1  0.5 0 -1 )
       ( -1  0.5 0.5 1  1  0.5 0.5 -1 )
       ( -1  0  1  1  1  1  0 -1 )
       ( -1  1  1  1  1  1  1 -1 )
       ( -1  0.5 0  0  0  0  0.5 -1 )
       ( -2 -1 -1 -1 -1 -1 -1 -2 )
    )
  )
)

( setf *bbishop-table*
  ( make-array '(8 8) :initial-contents
    '( ( -2 -1 -1 -1 -1 -1 -1 -2 )
       ( -1  0.5 0  0  0  0  0.5 -1 )
       ( -1  1  1  1  1  1  1 -1 )
       ( -1  0  1  1  1  1  0 -1 )
       ( -1  0.5 0.5 1  1  0.5 0.5 -1 )
       ( -1  0  0.5 1  1  0.5 0 -1 )
       ( -1  0  0  0  0  0  0 -1 )
       ( -2 -1 -1 -1 -1 -1 -1 -2 )
    )
  )
)
```

```
169567 CL-USER> (d)
169568   |---------------------------------|
169569   |                                 |
169570 8 |  --  BN  --  BK  --  --  BN  BR  |
169571   |                                 |
169572 7 |  --  --  --  --  --  --  BB  --  |
169573   |                                 |
169574 6 |  --  --  BP  --  --  BP  --  BP  |
169575   |                                 |
169576 5 |  --  BP  --  WN  --  --  BP  BQ  |
169577   |                                 |
169578 4 |  --  --  --  WP  --  --  WP  --  |
169579   |                                 |
169580 3 |  BR  WP  --  --  --  WP  --  --  |
169581   |                                 |
169582 2 |  WP  --  --  --  --  --  WB  --  |
169583   |                                 |
169584 1 |  WR  WN  WB  --  WK  --  --  --  |
169585   |                                 |
169586   |---------------------------------|
169587      A   B   C   D   E   F   G   H
169588
169589 NIL
169590 CL-USER> (highest-location-score 'w)
169591 Score: 4
169592 Score: 4
169593 Score: 5
169594 Score: 5
169595 Score: 4
169596 Score: 8
169597 Score: 5
169598 Score: 9
169599 Score: 2.5
169600 Score: 3.5
169601 Score: 3
169602 Score: 3
169603 Score: 2
169604 Score: 2
169605 Score: 2
169606 Score: 5
169607 Score: 4
169608 Score: 5
169609 Score: 6
169610 Score: 5.5
169611 Score: 5
169612 Score: 3
169613 Score: 3
169614 Score: 2
169615 Score: -1
169616 Score: 3
169617 Score: 2
169618 Score: 5
169619 ((#<SQUARE {100538F4F3}> #<SQUARE {100538FE83}>))
169620 CL-USER>
```

Returns the highest scoring move at 9

Results of Players Playing Each Other

Rankings of the 3 Players

1. Material Player
2. Location Player
3. Random Player

Each Player Played Each Other 1000 Times

Material Player vs Random:
- Material won: 510 time or 51%

Material Player vs Location:
- Material won: 528 times or 52.8%

Location Player vs Random:
- Location won: 547 times or 54.7%

```
|95689 T
|95690 CL-USER> (play-lr-games 1000)
|95691 Location Player wins: 547 times
|95692 Random Player wins: 453 times
|95693 NIL
|95694 CL-USER> (play-mr-games 1000)
|95695 Material Player wins: 510 times
|95696 Random Player wins: 490 times
|95697 NIL
|95698 CL-USER> (play-ml-games 1000)
|95699 Material Player wins: 528 times
|95700 Loction Player wins: 472 times
|95701 NIL
```

Continuing The Project

1. Combine the material and location players into one and see how it competes
2. Implement a better interface to use when actually playing the chess
3. Create a minimax player
4. Add Alpha-beta pruning