Title: Creating Different Types of Chess Players

Author: Kieran Finnegan

Abstract:

The primary objective of this project was twofold. Firstly, the aim was to implement the board game of chess in Lisp, effectively simulating the rules and mechanics of the traditional game. Secondly, and most importantly, the project focused on creating three different chess players with different approaches to gameplay.

The first player would adopt a strategy of making random moves, restricted only by the legality of each move given the current board state. This player essentially disregards any strategic consideration and relies solely on chance to determine its moves.

The second player, on the other hand, employs a scoring system based on Alan Turing's Turochamp to make informed decisions. By evaluating all possible moves and their respective scores, this player is able to make moves that maximize its potential advantage over its opponent. This approach offers a more strategic and calculated method of playing chess compared to the random move player.

The third player used a scoring system based on scores given to each square on a board for each piece. By evaluating all possible moves and their respective scores, this player is able to make moves that maximize the score of all the squares.

Sections:

Introduction:

Chess, a game with a history spanning since the 6th century, has long been considered a symbol of human intelligence and strategic prowess. The intricate interplay of pieces on the 8x8 board has captivated and challenged minds throughout the centuries, leading to the development of increasingly sophisticated chess strategies and theories (“History of Chess” - Andrew E. Soltis - 2021). As artificial intelligence has progressed, researchers have wanted to create machines that can rival or even surpass human performance. This paper presents an approach, development and the results of basic chess-players that were created throughout this project.

Initially, this project embarked on an ambitious journey, with the primary objective of developing a chess player utilizing the minimax algorithm and alpha-beta pruning to compete against human opponents. As the project progressed, however, it became evident that accomplishing these lofty goals within the given timeframe would be unattainable. Consequently, the project's focus shifted towards creating distinct chess players, each employing different decision-making strategies based on various aspects of the game.

Ultimately, a total of three distinct players were developed: a random player, a material player, and a location player. Each of these players demonstrated unique approaches to their decision-making processes, offering valuable insights into the effectiveness of different chess strategies and their potential for success in the game.

Background

Chess is a classic two-player strategy board game played on a square 8x8 grid, known as a chessboard, with 64 alternating light and dark squares. Each player begins the game with 16 pieces: one king, one queen, two rooks, two knights, two bishops, and eight pawns. The objective of the game is to capture the opponent's king.

Each of the pieces in chess have their own unique movement patterns that have been implemented, which are as follows:

1. King: The king can move one square in any direction: horizontally, vertically, or diagonally.

2. Queen: The queen is the most powerful piece on the board. It can move any number of squares horizontally, vertically, or diagonally, as long as its path is unobstructed.

3. Rook: The rook can move any number of squares horizontally or vertically in a straight line, as long as its path is unobstructed.

4. Bishop: The bishop moves any number of squares diagonally in a straight line, as long as its path is unobstructed. Since each player starts with two bishops, one on a light square and one on a dark square, the bishops can only ever move on their respective square colors.

5. Knight: The knight has a unique L-shaped movement pattern, moving two squares in a straight line, horizontally or vertically, followed by one square perpendicular to the initial direction, or vice versa. The knight is the only piece that can "jump" over other pieces on the board.

6. Pawn: The pawn moves forward one square vertically, but it captures diagonally, one square forward and to the left or right. On a pawn's first move, it has the option to move forward one or two squares vertically.

There are some special moves that certain pieces can make that have not been implemented, for example castling. (“How to Play Chess: 7 Rules to Get you Started” -CHESScom- 2022)

To enable the different players to make their decisions, I developed separate scoring systems for both the material player and the location player. These scoring systems helped evaluate and prioritize potential moves based on their respective strategies.

For the material player, I assigned a specific value to each unique chess piece, representing its relative "worth" in the game. I utilized the scoring system from Alan Turing's Turochamp, which assigns the following values (“Turochamp” -ChessProgrammingWiki- 2020):

Pawn = 1

Knight = 3

Bishop = 3.5

Rook = 5

Queen = 10

By each piece its own score, the material player can make informed decisions by assessing the potential material gain or loss resulting from a move.

On the other hand, the scoring system for the location player focuses on evaluating the positional strength of each square on the board for every piece type(“Piece-Square tables” -ChessProgrammingWiki-2019). Each unique chess piece has a set of assigned values for every square, reflecting the strategic advantage of occupying that particular square. For example the following is the table for white rook:

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

By employing these distinct scoring systems, the material and location players can effectively analyze the board's current state, weigh the benefits and drawbacks of potential moves, and select the optimal move according to their strategic approach.

Program Description

The program was developed using Common Lisp and the Common Lisp Object System for creating game objects and storing essential information.

One of the primary objectives of the program was to establish the chessboard, the individual pieces, and to integrate the game rules for each piece. Initially, I attempted to represent the board using a one-dimensional (1D) array consisting of 64 elements (“Board Representations in Computer Chess” - Likeawizard - 2022). Each element in the array would correspond to a square on the chessboard and indicate whether it was occupied and by which piece. However, as the project progressed, it became apparent that this approach was overly complicated and hindered the implementation of other components.

To overcome this challenge, I decided to modify the board representation from a 1D array to a two-dimensional (2D) array and created a square object in CLOS. This square object contained vital information such as rank, file, occupation status, and a list of the adjacent squares. The transition to a 2D array and the introduction of the square object streamlined the board representation and made it more intuitive to work with, ultimately facilitating the implementation of game mechanics and rules for each chess piece.

The next crucial component of the program involved implementing the movement rules for each chess piece. To achieve this, I created two separate functions for every piece: a move function and a legal-moves function (“Finding All Legal Chess Moves” - Christian Behle - 2021). These functions worked together to evaluate a requested move and determine its legality based on the piece's current square and the target square. Factors that could render a move illegal include violation of a piece's movement rules or the presence of an obstructing piece on the path to the target square.

After completing the move and legal-move functions for all pieces, I proceeded to incorporate the endgame rules. In traditional chess, a game concludes when a player achieves a checkmate against the opponent's king. However, after spending considerable time attempting to implement this rule, I opted for a simplified approach, ending the game when a player captures the opponent's king. To facilitate this, I developed a game-overp function that leverages bking-in-playp and wking-in-playp to verify if a king was removed from play during the previous turn.

With these foundational elements in place, I began designing an interface for two human players to compete against each other. The interface prompts players to input the current square of the piece they wish to move and the target square for the desired move. The program then employs the move function to relocate the piece to the target square, provided it constitutes a legal move.

After all this was completed, I proceeded to develop the different chess-playing strategies that constituted the primary focus of the project.

The first player implemented was the random player, which operates by generating a list of all the pieces the player has in play and randomly selecting one of them. The program then utilizes the get-all-possible-moves function to identify potential moves for the chosen piece. If the list of possible moves is non-empty, the player randomly selects a move from the list and executes it. Since there is no scoring function for this player, its decisions are uninfluenced by any strategic considerations.

The second player developed was the material player, which employs the scoring system outlined earlier to assess each potential move. The get-lowest-score function determines the optimal move for this player by examining all possible moves for each piece on the board, executing each move, and calculating the total score of the opponent's remaining pieces. The function returns the move that results in the lowest score. In cases where multiple moves yield the same lowest score, the player randomly selects a move from the list of equivalent options.

The final player implemented was the location player, which uses a distinct array for each piece type, comprising ten different arrays: bpawn-table, wpawn-table, wbishop-table, bbishop-table, wking-table, bking-table, knight-table, queen-table, brook-table, and wrook-table. Each array contains a score for every square on the chessboard, reflecting the strategic value of that position for the corresponding piece type.

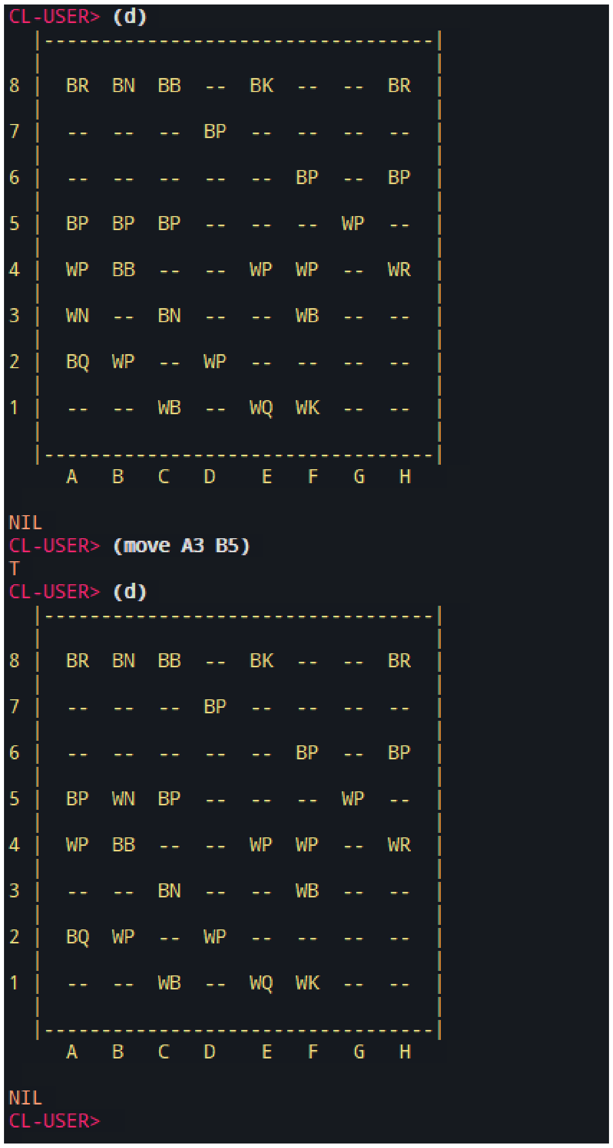
The location player's decision-making process is similar to that of the material player. It identifies all possible moves a player can make given the current board state, executes each move, and calculates the total score of all the squares occupied by the player's pieces. The program then returns the move or moves with the highest score, and the player randomly selects one from the list.

Upon completing the development of the different players, I proceeded to create methods that allowed a human to compete against the various players, as well as methods for the players to face off against one another. Additionally, I implemented methods to facilitate multiple game simulations and gather statistical data on the outcomes. These methods were used in testing the performance of the individual players and obtaining the results required for evaluating the success of the project.

Demos

This section is going to show the demo some of the functions that demonstrate the project best.

Move Demo



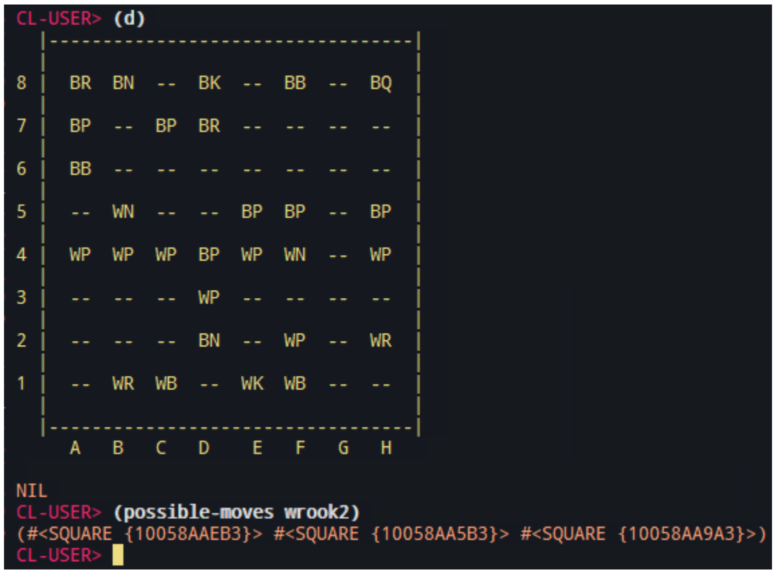
```
CL-USER> (d)
   |---------------------------------|
   |                                 |
8  |  BR  BN  BB  --  BK  --  --  BR |
   |                                 |
7  |  --  --  --  BP  --  --  --  -- |
   |                                 |
6  |  --  --  --  --  --  BP  --  BP |
   |      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

NIL
CL-USER> (move A3 B5)
T
CL-USER> (d)
   |---------------------------------|
   |                                 |
8  |  BR  BN  BB  --  BK  --  --  BR |
   |                                 |
7  |  --  --  --  BP  --  --  --  -- |
   |                                 |
6  |  --  --  --  --  --  BP  --  BP |
   |                                 |
5  |  BP  WN  BP  --  --  --  WP  -- |
   |                                 |
4  |  WP  BB  --  --  WP  WP  --  WR |
   |                                 |
3  |  --  --  BN  --  --  WB  --  -- |
   |                                 |
2  |  BQ  WP  --  WP  --  --  --  -- |
   |                                 |
1  |  --  --  WB  --  WQ  WK  --  -- |
   |                                 |
   |---------------------------------|
      A   B   C   D    E   F   G   H

NIL
CL-USER>
```

This Demo is showing the move function in action
It moves the white knight on A3 to capture the black pawn on B5

Possible Moves Demo:

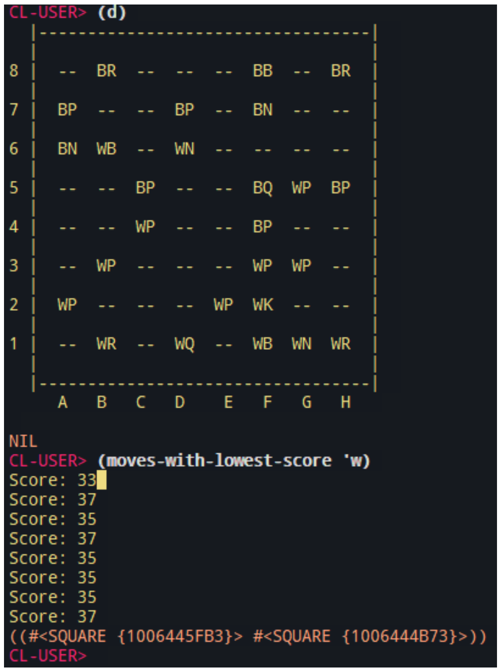


```
CL-USER> (d)
   |--------------------------------|
   |                                |
8  |  BR  BN  --  BK  --  BB  --  BQ  |
   |                                |
7  |  BP  --  BP  BR  --  --  --  --  |
   |                                |
6  |  BB  --  --  --  --  --  --  --  |
   |                                |
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>
```

This demo shows the possible moves method. The white rook On h2 it has 3 possible squares that it can travel to.

Material Player Scoring Demo:

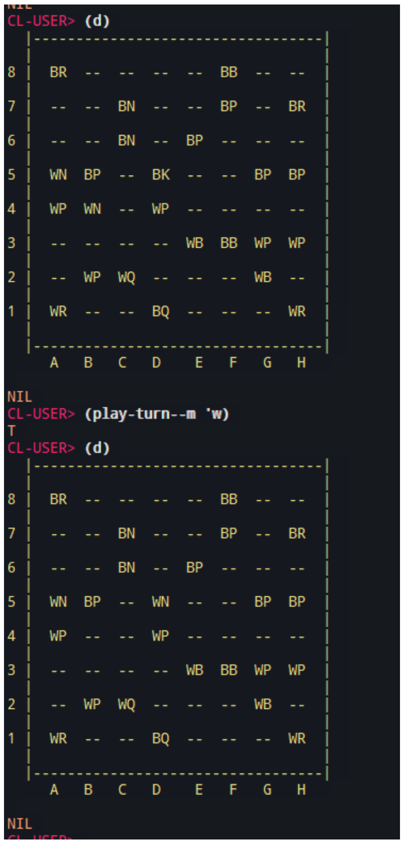


```
CL-USER> (d)
   |--------------------------------|
   |                                |
8  |  --  BR  --  --  --  BB  --  BR  |
   |                                |
7  |  BP  --  --  BP  --  BN  --  --  |
   |                                |
6  |  BN  WB  --  WN  --  --  --  --  |
   |                                |
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>
```

This demo show shows the scoring function for the material player. It returns the move with the highest score for the white player.

Material Player Move Demo;



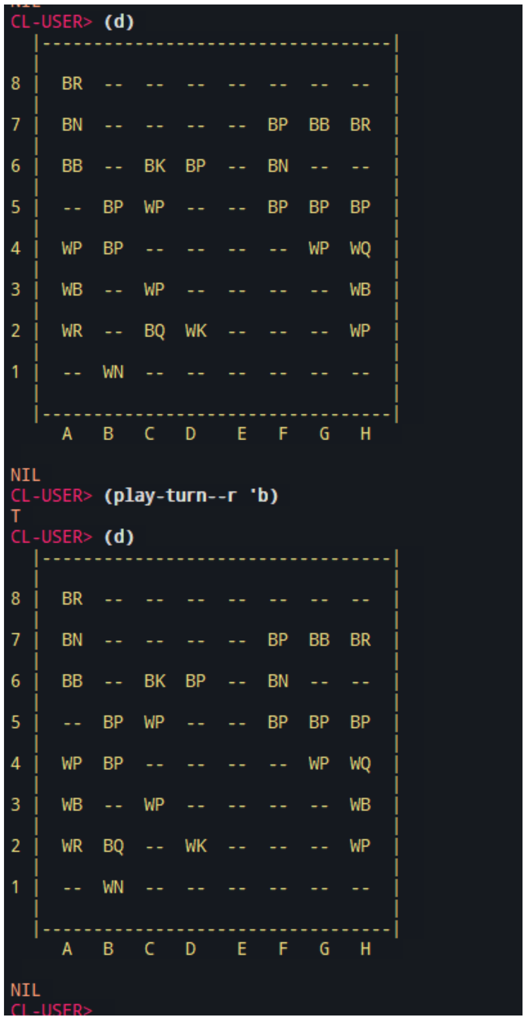
```
CL-USER> (d)
  |---------------------------------|
  |                                 |
8 |  BR  --  --  --  --  BB  --  -- |
  |                                 |
7 |  --  --  BN  --  --  BP  --  BR |
  |                                 |
6 |  --  --  BN  --  BP  --  --  -- |
  |                                 |
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
```

The white knight on B4 take the black king on D5

Random Player Moving Demo:



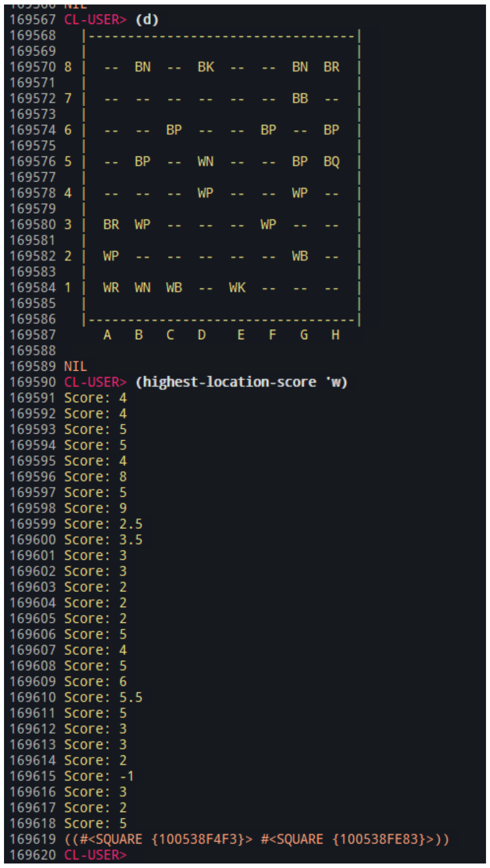
```
CL-USER> (d)
  |---------------------------------|
  |                                 |
8 |  BR  --  --  --  --  --  --  --  |
  |                                 |
7 |  BN  --  --  --  --  BP  BB  BR  |
  |                                 |
6 |  BB  --  BK  BP  --  BN  --  --  |
  |                                 |
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 black queen on C2 moves to B3
Instead of capturing the white king
on D2

Location Player Scoring Demo:



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

NIL
CL-USER> (highest-location-score 'w)
Score: 4
Score: 4
Score: 5
Score: 5
Score: 4
Score: 8
Score: 5
Score: 9
Score: 2.5
Score: 3.5
Score: 3
Score: 3
Score: 2
Score: 2
Score: 2
Score: 5
Score: 4
Score: 5
Score: 6
Score: 5.5
Score: 5
Score: 3
Score: 3
Score: 2
Score: -1
Score: 3
Score: 2
Score: 5
((#<SQUARE {100538F4F3}> #<SQUARE {100538FE83}>))
CL-USER>
```

The function returns a the move with the highest score possible based on the location of the players pieces

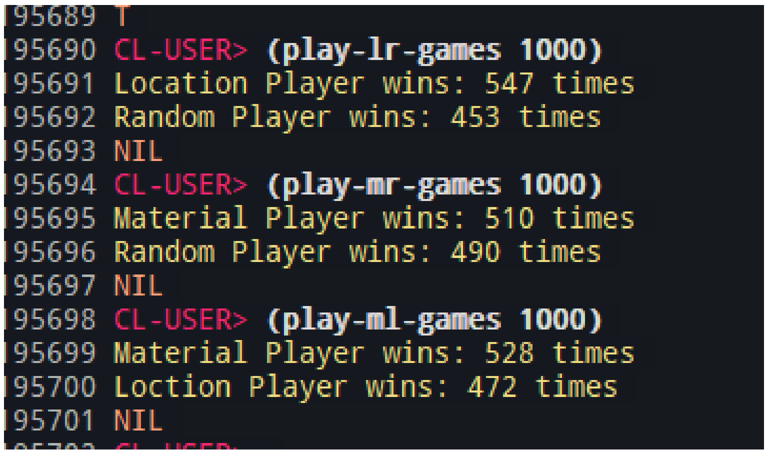
Results:

Upon the completion of the project, I conducted an extensive evaluation involving the three distinct players, pitting them against each other in a series of 1000 simulated games. Contrary to my initial expectations, the outcomes revealed a much closer competition than anticipated. I had originally hypothesized that the material player would significantly outperform the random and location players.

While the material player emerged victorious in a majority of the games against its counterparts, the winning margins were notably slim. Against the random player, the material player secured 510 wins, accounting for 51% of the games. In matchups with the location player, the material player won 528 games or 52.8% of the time. Interestingly, the location player outperformed the random player, winning 547 games or 54.7% of the time. The narrow margins between the material player and its opponents were indeed surprising, as I expected a more dominant performance.

The most intriguing aspect of these results, however, was the location player's superior win rate of 54.7% against the random player, surpassing the material player's success in the same matchup. This finding suggests that the location player could be a more favorable choice when competing against the random player.

In conclusion, the project's outcomes indicate that, overall, the material player has a higher likelihood of defeating both location and random players. However, when facing the random player, the location player emerges as a more advantageous choice.



```
T
CL-USER> (play-lr-games 1000)
Location Player wins: 547 times
Random Player wins: 453 times
NIL
CL-USER> (play-mr-games 1000)
Material Player wins: 510 times
Random Player wins: 490 times
NIL
CL-USER> (play-ml-games 1000)
Material Player wins: 528 times
Loction Player wins: 472 times
NIL
```

The statistics of 1000 games

Reflections and Conclusions:

This project, although not aligned with the initial vision, proved to be a valuable learning experience and a successful endeavor. Throughout the course of the project, I gained a deeper understanding of chess and the intricacies of planning and executing a large-scale project. The initial plan was ambitious, and the limited timeframe and experience presented challenges that prompted a reassessment of the project's scope.

Upon realizing the need for adjustments, I revised the plan to align more closely with the available resources while still adhering to the core objectives. As a result, the final project laid a solid foundation that can be expanded and refined in the future, potentially evolving into a more comprehensive and sophisticated program.

In conclusion, the project taught valuable lessons about project management, adaptation, and the importance of setting realistic goals. The experience gained from this project can be applied to future endeavors, paving the way for continued growth and improvement.

Bibliography:

1. “Board Representations in Computer Chess” - Likeawizard - 2022
   https://lichess.org/@/likeawizard/blog/review-of-different-board-representations-in-computer-chess/S9eQCAWa

2. “How to Play Chess: 7 Rules to Get you Started” -CHESScom- (2022)
   https://www.chess.com/learn-how-to-play-chess

3. “History of Chess” - Andrew E. Soltis - (2021)
   https://www.britannica.com/topic/chess/History

4. “Finding All Legal Chess Moves” - Christian Behle - (2021)
   https://levelup.gitconnected.com/finding-all-legal-chess-moves-2cb872d05bc6

5. “The Original CHess Engine: Alan Turing’s Turochamp - the_real_greco- (2021)
   https://www.chess.com/blog/the_real_greco/the-original-chess-engine-alan-turings-turochamp

6. “Turochamp” -ChessProgrammingWiki- 2020
   https://www.chessprogramming.org/Turochamp#Evaluation_Features

7. “Piece-Square tables” - ChessProgrammingWiki-2019
   https://www.chessprogramming.org/Piece-Square_Tables