# Unoptimized Chess

## Version 1.0 By: Loading...™

**Contributors: Paul Khayet, Jose Diaz, Maher Tarek Homsi, Arhant Katare, Victor Dam, Cameron Bagheri**

**4/3/22**

# Table of Contents

# Glossary

**Attack -**    A piece is said to attack (or threaten) an opponent's piece if, on the next move, it could capture that piece

**Bishop -**    The bishop (B) is a piece in the game of chess. It moves and captures along diagonals without jumping over intervening pieces.

**Board -**    A board (or chessboard) is a game board used to play chess. It consists of 64 squares, 8 rows by 8 columns, on which the chess pieces are placed. It is square in shape and uses two colors of squares, one light and one dark, in a checkered pattern.

**Capture -**    To remove the opponent's piece from the board by taking it with one's own piece. Except in the case of an en passant capture, the capturing piece replaces the captured piece on its square.

**Castle -**    A move in chess involving an unmoved king moving 2 spaces left or right, and an unmoved rook moving adjacent to the king's new position, on the square that the king crossed over. Castling cannot be done if the king is in check, or if the king crosses through or over a space that would make it checked. No square in between the king and rook may be occupied by another piece.

**Check -**    A condition that occurs when a player's king is under threat of capture on the opponent's next turn. A king so threatened is said to be in check, and capturing the attacker, blocking the check, or moving the king to another position are the only potential legal moves if this condition occurs for the game to continue.

**Checkmate -** The result when a king has no squares to move in order to escape the state of check after it has been checked by a piece from the opposing side. In addition, in this condition, the check cannot be blocked, and the piece from the opposing side making the check cannot be captured. The game concludes at this point with the opposing side being the victor.

**Draw** -          A draw occurs in chess when neither player wins nor loses—the game ends in a tie.

**En Passant -** A form of advanced capture which occurs when the opponent's pawn makes their first two square move. When this happens your pawn is able to move diagonally, to behind the opponent's pawn, and in turn capture it without being on its square. This is the only move where a capture doesn't end with the capturing piece on the opponent's square.

**File** -          Rows that go from up to down across the chessboard, each containing 8 squares, and are referred to by letters, starting with a and ending with h. There are eight files on a board.

**King -**          The King (K) is a piece in the game of chess that has the option of moving one step in any 8 directions - forward, backward, left, right, and diagonally. The King can capture any piece. The king can be under check and if no legal moves are available to escape this condition of check, a result of checkmate occurs. If the king is not in check and no moves are available for the king as well as any other piece for one side (chess player), this results in a draw.

**Knight** -       The knight (N) is a piece in the game of chess, represented by a horse's head and neck. It may move two squares vertically and one square horizontally or two squares horizontally and one square vertically.

**Pawn** -          A pawn (P) moves by advancing a single square, but the first time a pawn moves, it has the additional option of advancing two squares. A pawn may not move backwards. A pawn captures diagonally forward one square to the left or right. If a pawn reaches its respective end of the chessboard, it can be promoted to a bishop, rook, queen, or knight.

**Piece** -         A chess piece, or chessman, is a game piece that is placed on a chessboard to play the game of chess. It can be either white or black, and it can be one of six types: king, queen, rook, bishop, knight, or pawn.

**Promotion -** Replacement of a pawn with another higher ranking piece such as a knight, bishop, rook, or queen which occurs when a pawn reaches its last rank (the space farthest from the player).

**Queen -** The queen (Q) is the most powerful piece in the game of chess, able to move any number of squares vertically, horizontally or diagonally, combining the power of the rook and bishop.

**Rank -** Columns that go from side to side across the chessboard, each containing eight squares, and are referred to by numbers, starting with 1 ending with 8. There are eight total on a board.

**Rook -** The rook (R) is a piece in the game of chess. It may move any number of squares horizontally or vertically without jumping, and it may capture an enemy piece on its path; additionally, it may participate in castling.

**Stalemate -** The result when one side (black or white) can make no legal move, and this side's king is not in a state of check. None of the pieces can move to another square so the game ends in a draw.

# 1
# Computer Chess

## 1.1 Usage Scenario:

The usage scenario for the first implementation of our program design includes a greeting message, followed by a prompt asking for the color the user chooses to play as. Then the game officially begins with an ASCII text display of a chessboard at the starting position according to the rules of chess. Depending on which side the user chooses (white or black), either the program prompts the user to enter the first move or the program chooses a move to make initially. Here is a sample proceeding of what our program displays at first.

```
Welcome to Unoptimized Chess, our newly released chess game from
Loading…™!
1. Start New Game (Human v. AI)
2. Game Settings
3. Exit Game
Choose option: 1
================================================================
================================================================
Please enter a side (color) to play as.
1. W
2. B
Choose option: 1
================================================================
================================================================
    +----+----+----+----+----+----+----+----+
 8  | bR | bN | bB | bQ | bK | bB | bN | bR |
    +----+----+----+----+----+----+----+----+
 7  | bP | bP | bP | bP | bP | bP | bP | bP |
    +----+----+----+----+----+----+----+----+
 6  |    |    |    |    |    |    |    |    |
    +----+----+----+----+----+----+----+----+
 5  |    |    |    |    |    |    |    |    |
    +----+----+----+----+----+----+----+----+
 4  |    |    |    |    |    |    |    |    |
    +----+----+----+----+----+----+----+----+
 3  |    |    |    |    |    |    |    |    |
    +----+----+----+----+----+----+----+----+
 2  | wP | wP | wP | wP | wP | wP | wP | wP |
    +----+----+----+----+----+----+----+----+
 1  | wR | wN | wB | wQ | wK | wB | wN | wR |
    +----+----+----+----+----+----+----+----+
      a    b    c    d    e    f    g    h
White to move. Please enter a move in the format of initial square,
piece, and destination square (e.g. e2 wP e4): e2 wP e4
```

```
    +----+----+----+----+----+----+----+----+
8 | bR | bN | bB | bQ | bK | bB | bN | bR |
    +----+----+----+----+----+----+----+----+
7 | bP | bP | bP | bP | bP | bP | bP | bP |
    +----+----+----+----+----+----+----+----+
6 |    |    |    |    |    |    |    |    |
    +----+----+----+----+----+----+----+----+
5 |    |    |    |    |    |    |    |    |
    +----+----+----+----+----+----+----+----+
4 |    |    |    |    | wP |    |    |    |
    +----+----+----+----+----+----+----+----+
3 |    |    |    |    |    |    |    |    |
    +----+----+----+----+----+----+----+----+
2 | wP | wP | wP | wP |    | wP | wP | wP |
    +----+----+----+----+----+----+----+----+
1 | wR | wN | wB | wQ | wK | wB | wN | wR |
    +----+----+----+----+----+----+----+----+
      a    b    c    d    e    f    g    h
```

The given illustrations display a possible interface for our initial design in its most elementary form. Some aspects of our program are still undefined, and we plan on having a developed graphical user interface with more features if time permits.

## 1.2 Goals

Our goal at Loading…™ is to provide a clear, easy-to-use interface that allows a human user to have an enjoyable experience playing against a computer, either for leisure or competitively. Goals for the program are that it permits only legal moves, does not result in a segmentation fault, and ends the game in appropriate fashion when checkmate or stalemate occurs. In addition, the program should not suddenly break, end the game abruptly or freeze during gameplay.

## 1.3 Features:

Features of our initial release include a welcome message, a menu to prompt specific modes of chess, and other basic options like exit, start game, and settings.

For the most basic implementation of our program design, we desire a game with the following characteristics.The program displays a chessboard based on ASCII text, supports a human-versus-AI play option, permits the user to choose the color (white or black), prompts the user to type a move, and updates the chessboard based on the user's move. The program shall also end the game appropriately if one side wins or there is a draw, and this release should proceed only if legal moves are made.

Additional bonus features of our program that may be updated in the future include a human-versus-human gameplay option, a graphical user interface as opposed to the initial interface containing ASCII text, and clock settings that enforce playing moves within a specific time period, as well as the ability for the player to pause the clock at any time. Once a player's time reaches 0 seconds, that player's side loses the game. We also have plans to possibly create a Bot vs Bot gamemode in order to pin the AI against itself. We may even implement algebraic notation as the official documentation of chess moves having completed other desired features. If we implement a clock later on, we may construct an option to pause the timer.

# 2
# Installation

## 2.1 System requirements

This program requires the following:
1. Linux System Version (kernel release information is version 2.6.32-754.33.1.el6.x86_64 or later)
2. GCC Compiler
    a. g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-23) version or later
3. Keyboard, Mouse, and a Screen able to display the GUI.
4. Incomplete: Some sort of software to display the GUI may be required later once we make more progress on this project.

## 2.2 Setup and Configuration

1. To unpack the tar file you must right click on the file and choose 'Unpack the archive'. If you are using the Linux terminal, use the command 'tar –xf UnoptimizedChess.tar.gz.' to extract the tar file in your current directory.
2. Once the files are extracted in the correct folder, open the console and make sure you are in the correct directory.
3. Use the command 'make' to compile the chess game without the GUI or use 'make chess' to make a version of the game with a GUI.
4. Once the make file has been created type the command './chess' in order to start the game!

## 2.3 Uninstalling

Once we have a finalized version of our program that is an authentic installation, uninstallation would proceed by typing 'sudo apt-get --purge remove UnoptimizedChess' in the Linux terminal.

For the time being, to uninstall Unoptimized Chess™ run the following commands in the Linux command shell:

1. If you want to uninstall the files for the game, you must first type the command 'make clean' in the current directory.
2. Then type the command 'ls' to see all of the files inside of the folder (we will change this in the future for our exact files) and then type 'rm filename' for all of the files.

# 3
# Chess Program Functions and Features

## 3.1 Description of function 1 (main)

• *User input, program output, screen shot, etc.*

The main function displays the first greeting and prompts the user to start the game, choosing which side to play as. In addition, the user can exit or see game settings if desired by typing a number from the choices given by the program. There will also be a loop within the main function to call other functions like "movechesspiece" (mentioned in next section) in order for the program to operate and allow the user to play the game of chess. Later on, our program may have the user simply drag a piece on the chessboard via the graphical user interface, and this change would be reflected in our main method. Because our function is still not yet defined fully, we have opted to not include a screenshot for any of the functions in this third section as of yet. For more information, on what they would accomplish, we invite the reader to check the usage scenario for sample playthrough.

## 3.2 Description of function 2 (movechesspiece)

• *User input, program output, screen shot, etc.*

The Function titled "movechesspiece" is going to be used in order to allow moving the piece that the human user has selected. For example, a player may choose the location a5, and the function will see that the player has chosen a square with a pawn on it. It will then prompt the user with the possible moves it can make and allow the player to choose the space it would like to move it to.

## 3.3 Description of function 3 (ResetBoard)

• *User input, program output, screen shot, etc.*

The ResetBoard function should be called after the user loses or wins a game against the computer. It should, however, only be used after a prompt is given to the user, indicating their win or loss. ResetBoard might also have functionality if we decide to implement a Reset option that the user can use at any time during the game.

# 4
# Back matter

## 4.1 Copyright

Copyright © 2022 Loading…™. All rights reserved. Developed in the United States of America. For usage in binary and source forms, modified or unmodified, the source code may be redistributed having satisfied the conditions below:

## 4.2 Error Messages

Error messages are displayed if the user makes an illegal move. In this case, the message below is displayed to the user.

```
ERROR. An illegal move has been made. Please enter an allowed move
according to the rules of chess. For more information on legal moves,
consult our user manual's glossary and/or index containing definitions
of specific pieces, their allowed range of squares to move to, and
other details regarding move limitations.
```

The human user may also incorrectly type a move as input. If the human user mistakenly types a square that does not exist or types a piece incorrectly without the correct short form as per our implementation, the following message is displayed to the user.

```
ERROR. Cannot parse move. Please enter a move in the following format,
currentSquare piece destinationSquare e.g. 'f1 wB c4'. For pieces,
include the player's color as well as piece type so that wB stands for
```

white bishop, bK stands for black knight, and so forth (see glossary for short forms of piece names).

In a future implementation with a clock, if our AI does not provide a move in less than one minute or such, our program may display the message down below.

**ERROR.** Program has run into problems calculating next move, please restart Unoptimized Chess. Would you like to submit a bug report? Type y/n: ____

## 4.3 Index

11

## 4.4 Additional Sources

EECS22L_Project_1_Handout

The images of the chessboard in the Usage Scenario section were provided by T.A. Vivek Govindasamy, Yutong Wang, and Prof. Rainer Domer in the Project 1 Handout.

https://www.winzip.com/en/learn/tips/extract-files/tar-gz-file/
Contains information that we used in our setup and configuration section

http://www.wachusettchess.org/ChessGlossary.pdf
Contains various definitions referenced in this User Manual.

https://www.knightschessclub.org/documents/basic-chess-rules.pdf
Contains descriptions regarding the rules and pieces of Chess for better understanding of the game.

https://go.dev/doc/install
Contains copyright notice of Go Installer that we modeled our copyright message after.

https://www.howtogeek.com/229699/how-to-uninstall-software-using-the-command-line-in-linux/
https://askubuntu.com/questions/939923/how-to-uninstall-software-that-was-installed-by-make-install
Contains information on uninstallation in Linux.

https://techterms.com/definition/system_requirements
Contains info about system requirements