Software Specification

Chess bot

Version 1.0 - April 2022

Authors: Daniel McClure, Angelina Negrete, Christiano Gravina

JTDA, Inc.

Table of Contents

Glossary

Pawn - a chess piece of the smallest size and value. A pawn moves one square forward along its file if unobstructed (or two on the first move), or one square diagonally forward when making a capture. Each player begins with eight pawns on the second rank, and can promote a pawn to become any other piece if it reaches the opponent's end of the board.

Knight - 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. Each player starts with two knights on the first row on the b and g files.

Bishop - A piece that moves and captures along diagonals without jumping over intervening pieces. Each player begins the game with two bishops. One starts between the king's knight and the king, the other between the queen's knight and the queen. The starting squares are c1 and f1 for White's bishops, and c8 and f8 for Black's bishops.

Rook - A piece that moves 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. Each player starts the game with two rooks, one in each corner on their own side of the board.

Queen - 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. Each player starts the game with one queen, placed in the middle of the first rank next to the king.

King - the most important piece in the game of chess. It may move to any adjoining square as well as perform a move known as castling. If a player's king is threatened with capture, it is said to be in check, and the player must remove the threat of capture on the next move. If this cannot be done, the king is said to be in checkmate, resulting in a loss for that player. A player cannot make any move that places their own king in check.

Castling - a move in the game of chess in which a player moves their king two squares toward a rook on the same rank and moves the rook to the square that the king has crossed. It is the only move in chess in which a player moves two pieces in the same move. Castling may be done only if neither the king nor the rook has previously moved, the squares between the king and the rook are unoccupied, the king is not in check, and the king does not cross over or end up on a square attacked by an opposing piece. Castling can be done with either rook.

Enpassant - a move to capture a pawn without moving the pawn onto the captured pawn's square. For this move to be legal one, the capturing pawn must have advanced exactly three ranks to perform this move. Two, the captured pawn must have moved two squares in one move, landing right next to the capturing pawn. Finally, the en passant capture must be performed on the turn immediately after the pawn being captured moves. If the player does not capture en passant on that turn, they no longer can do it later. The capturing pawn is moved behind the captured pawn.

Check - When the King is directly under attack by a piece from the opposing side. Check is a condition that occurs when a player's king is under threat of capture on the opponent's next turn.

Checkmate - any game position in which a player's king is in check and there is no possible escape. Checkmating the opponent wins the game.

Stalemate - When the player can make no legal move and their king is not in check, resulting in a draw/tie.

1 Software Architecture Overview

1.1 Main data types and structures

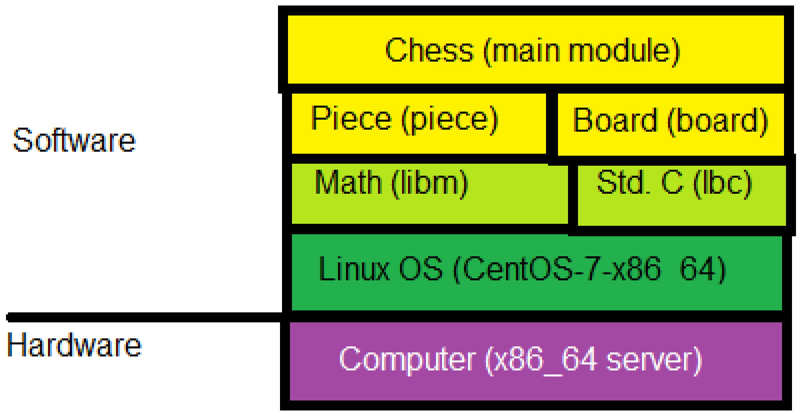
A 2D-array is used to represent the game board and its 64 spaces. It is an 8 by 8 matrix where the rows represent the rank of the pieces, and the columns represent the file of the pieces.

A data structure is used to represent a game piece. This contains an int value to represent what kind of piece it is, and a second int value to represent the color of the piece.

A data structure is used to represent the position of a move. This contains two int values that represent rank and file.

An array is used to hold the pointers to the possible moves of a piece.

1.2 Major software components



1.3 Module interfaces

Chess.c

Provides: main program

Requires: Piece.h and Board.h

Board.c

Provides: Setup of the chess board

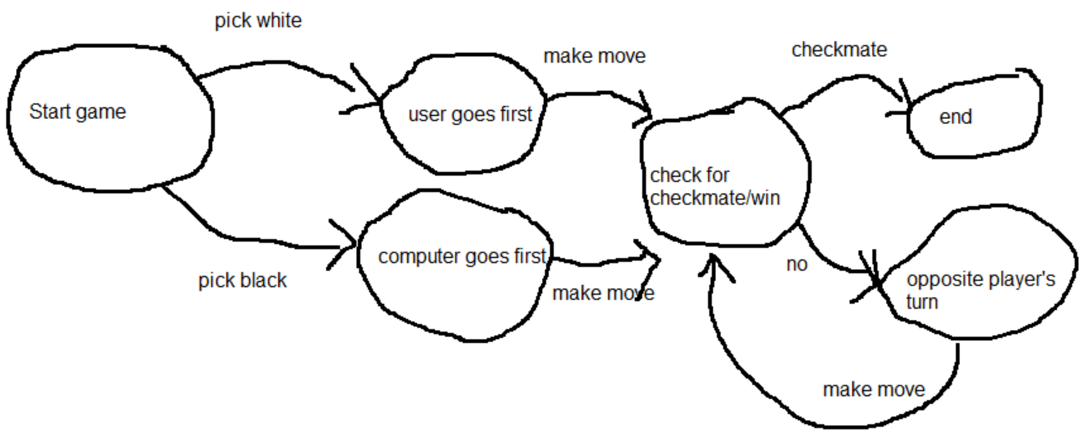
Requires: Board.h

Piece.c

Provides: Creation of each piece

Requires: Piece.h

1.4 Overall program control flow



2 Installation

2.1 System requirements, compatibility

Computer with PC hardware

Linux operating system

2.2 Setup and configuration

The program is already configured and just needs to be installed by the user. See below for installation instructions.

2.3 Building, compilation, installation

1. Program comes in a tar.gz package named JTDAChess.tar.gz
2. Install the package and extract it
3. Unpack the archive and type make to start the program
4. ./JTDAChess to start the game q

2.4 Uninstalling

To uninstall the program, type in the terminal : rm JTDAChess

3 Documentation of packages, modules, interfaces

3.1 Data Structures

This is the 2D array which represents the chessboard. The rows represent the rank, and the columns are the file.

PIECE *board[8][8];

This is the data structure which represents a piece on the chess board. Its type is an int which corresponds to a specific kind of piece (i.e. 0 would be a pawn, 1 would be a rook, 2 would be a knight, etc.). Its color is also an int, so 0 is white and 1 is black.

```
typedef struct Piece
{
	int type;
	int color;
} PIECE;
```

This is the data structure which represents the location of a move. The rank of the move is the first index of the 2D array which represents the board, and the file is the second index of that array.

```
typedef struct Move
{
	int rank;
	int file;
} MOVE;
```

This is the array that holds a list of possible moves for a given piece.

```
MOVE *moves[];
```

3.2 Functions and Parameters

The PrintBoard() function displays the chessboard in an ASCII representation.

```
void PrintBoard();
```

The NewPiece() function creates a new piece in memory given an int value for its type and an int value for its color.

```
PIECE *NewPiece(int value, int color);
```

The DeletePiece() function frees a given piece from memory and removes it from the game.

```
void DeletePiece(PIECE *p);
```

The MovePiece() function takes in two MOVE pointers in order to move the piece in the position given by initial to the position given by final.

void MovePiece(MOVE *initial, MOVE *final);

The CreateMove() function creates a possible move in memory given two int values for its rank and file.

MOVE *CreateMove(int rank, int file);

The DeleteMove() function frees a given move from memory.

void DeleteMove(MOVE *m);

The CheckForCheck() function checks if the given king piece is in check. It returns 0 if not in check, and 1 if in check.

int CheckForCheck(PIECE *p);

The CheckForMate() function checks if the given king piece is in checkmate. It returns 0 if not in checkmate, and 1 if in checkmate.

int CheckForMate(PIECE *p);

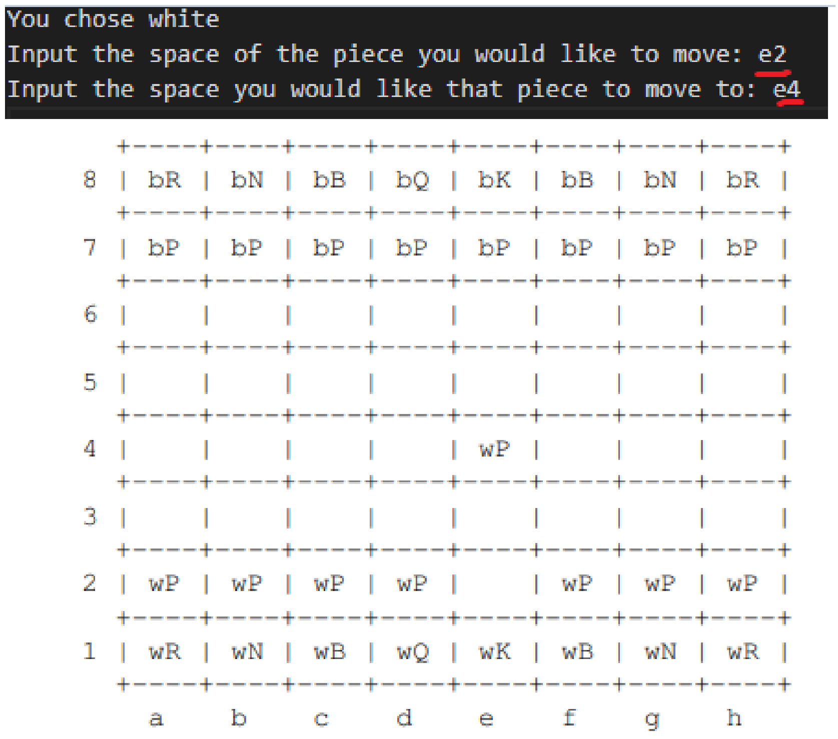
The PromotePiece() function takes a given pawn and promotes it to whatever type of piece is specified by the user.

PIECE *PromotePiece(PIECE *p, int type);

3.3 Input and Output formats

The format of a move input by the user

In order to make a move the user will input the start and end positions of a piece. The program will display the prompt: “Input the space of the piece you would like to move:” The user will input a character (a-h) followed by a number (1-8) to indicate a piece's position. The program will then prompt the user to input the space they want to move the selected piece. If the move can not be made the user will receive an error message stating that the move is invalid and to choose another position.

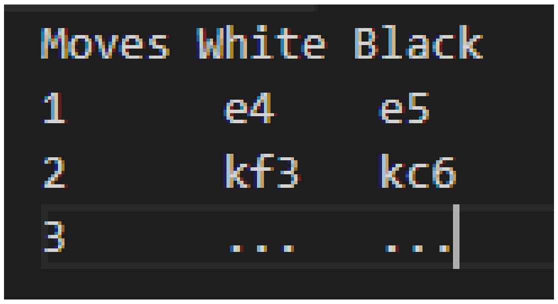


```
You chose white
Input the space of the piece you would like to move: e2
Input the space you would like that piece to move to: e4
```

```
   +----+----+----+----+----+----+----+----+
 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 format of a move recorded in the log file

After the game is complete a log file will be given to the user that contains all the moves that were made during the game. The text document will display the move each piece made and which player made it.



```
Moves White Black
1      e4    e5
2      kf3   kc6
3      ...   ...
```

4 Development plan and timeline

4.1 Partitioning of tasks

1. Create a piece structure
    a. Each piece has a type, a color and a location
    b. Type of piece will define how it moves
    c. Kings and rooks need a special variable to see if they moved. If they did not, castling is an option
    d. Pawns need to check the opponent's last movement to see if an en passant is possible.
2. Make a board where pieces will stay
    a. Maybe a 2d array of locations where the rows are 1 dimension and the columns are the other dimension
    b. Location structure (quick thought)
        i. Is it occupied?
        ii. color
3. Make pieces move
    a. Check for movement validity
        i. Check if move will make the king come in check
        ii. Kings and rooks need to see a special variable to see if they moved. If they did not, castling is an option
        iii. Pawns need to check the opponent's last movement if an en passant is possible.
    b. Check if there is anything on the landing location and if it can be killed
4. Check for check - if the king is threatened

        a. Check which moves will get the king out of check
            i. If there are no moves, checkmate, the game is over
5. Output list of moves made during the game
    a. Download a text document with the moves

4.2 Team member responsibilities

- Main program - Daniel
- Menu - Daniel
- Board - Daniel
- Chess objects - Christiano
- Lists of moves - Christiano
- Chess rules - Angelina
- Log file module - Angelina
- Strategy (AI) - GROUP
- Documentation - GROUP
- Testing - GROUP

Copyright

References

EECS 22L: Project 1 Handout; Prepared by Vivek G. and Yutong Wang; Prof. Rainer Dömer; 29 March 2022.

EECS 22L: Chess Software Specification Grading Criteria; Prepared by Vivek G. and Yutong Wang; Prof. Rainer Dömer; 4 April 2022.

Index