JUNGKOOK CHESS 1.0
(Software Manual)

Henchmen:
Roy Guan
Abhishek Subandh
Kiran Vanuy
Yuxuan Lu

University of California, Irvine

Table of contents

Glossary

Structures:
Position - a structure that defines a space on the board and a way to keep track of which piece to display at that location
Piece - a structure to define a player’s location, type, and ability to perform certain special moves

Software Architecture Overview

1.1 Main data types and structures

Structures:

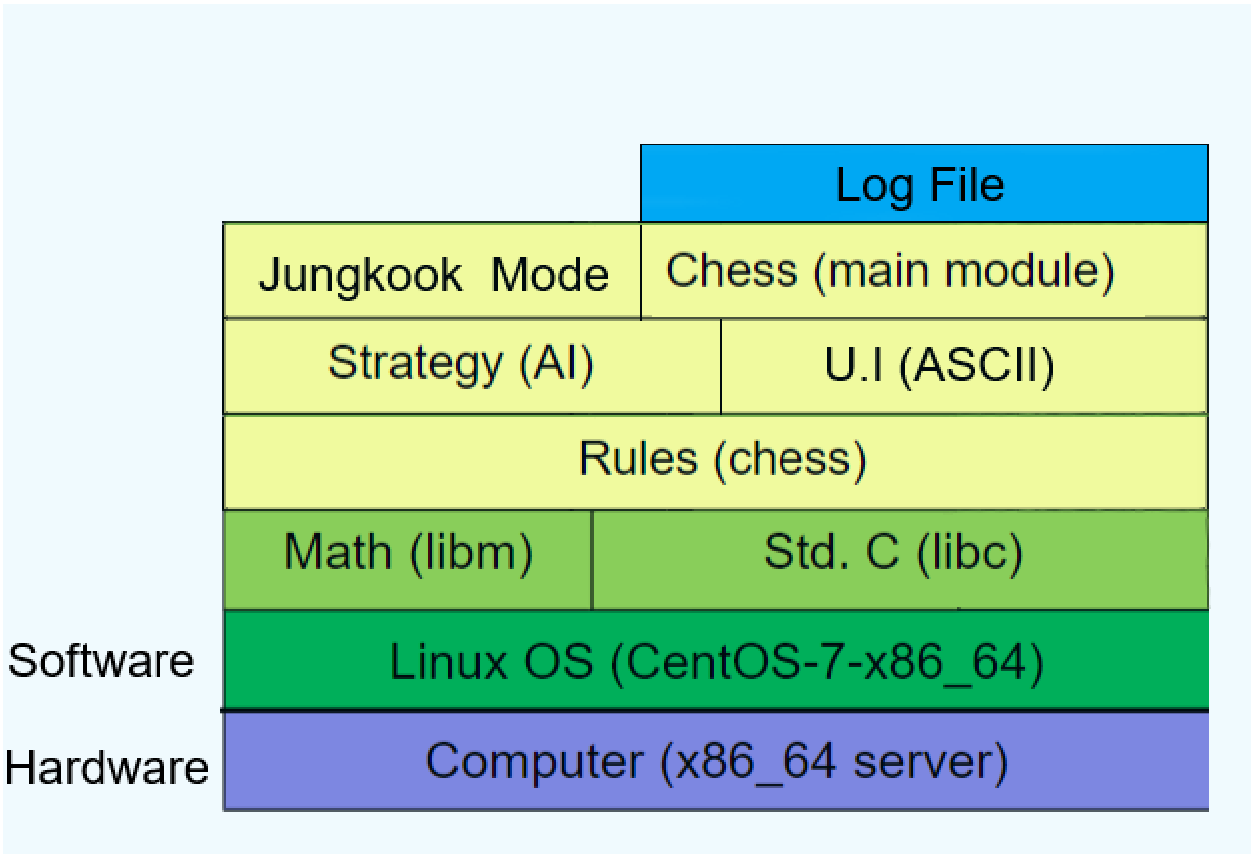
Our Chess program design is through the following main data types that are dependent on one another:

```
	Chess_moveList -
	Chess_move -
	Chess_position -
	Chess_board -
	Chess_player -
	Chess_piece -
```

Enum:

```
	Chess_piece_type
		EMPTY, PAWN, ROOK, KNIGHT, BISHOP, QUEEN, KNG
	Chess_player_color
		BLACK, WHITE
	Chess_player_type
		Player1, Player2, AI
```

1.2 Major software components



1.3 Module interfaces

Main: Initial Launch and Setup
Global Dependencies: modules that are used in the entire program

- Error Messages
- Definitions and constants

GUI: graphical user interface of the game board
Computer AI: computer generated moves

- API of major module functions

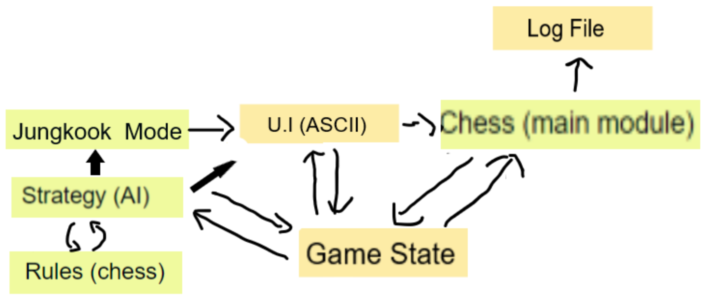
Jungkook mode

Provides: Suggestion of the best move to make for the player.

Requires: PlayerBestMove() from Strategy module. The Strategy Module needs the Rules module to make legal moves.

Exported Functions: JungkookMove()

Required Functions: PlayerBestMove()



Notes: Outputs a message into the console, but no other module depends on this output.

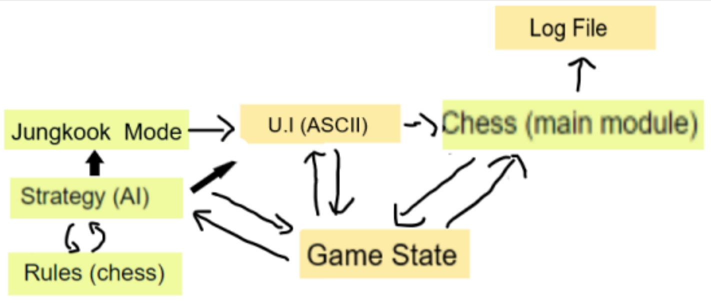
The Game State controls whether the message is output into the U.I

Chess (main module)

Provides: Outputs the board from the U.I, as well as from Jungkook Mode if its enabled.

Writes to log file

Requires: Status of the game from the Game State module.



Exported Functions:

LastMove() - to Log File, NextMove() to Game State

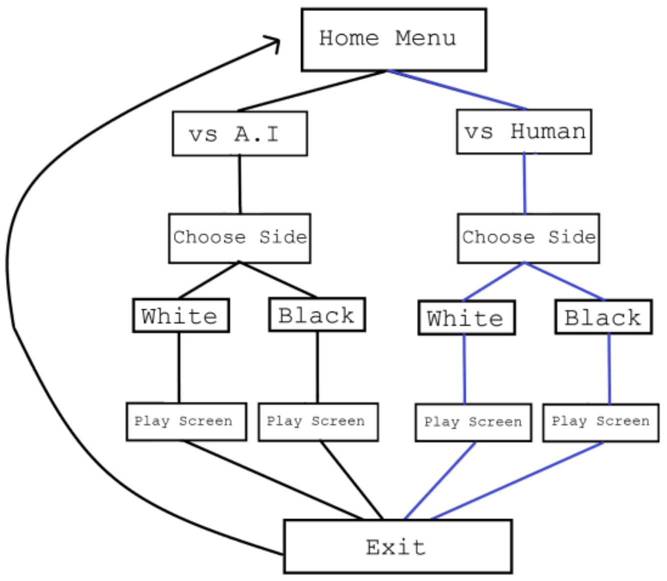
Required Functions:

PrintBoard(), GameState(), JungkookMove() (if enabled)

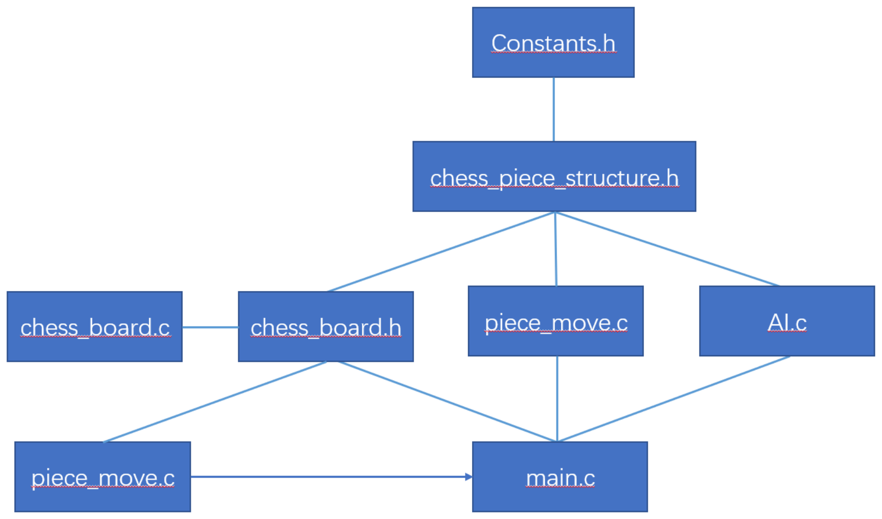
Notes: N/A

1.4 Overall program control flow

GAME STATES (Game State Module)



1.5 Decomposition



Installation

2.1 System requirements, compatibility

For Windows:

- PC Hardware (x86_64 server)
- Operating system: Linux OS
- Ensure that the computer is able to run a terminal with an ssh client (ex. Putty)
- Is able to access the UCI EECS servers (ex. crystalcove.eecs.uci.edu)

For Macs:

- Open the "Terminal" app that is already installed on the Mac
- Make a new remote connection to access the EECS servers

2.2 Setup and configuration

- Once connected to the EECS server, copy the files from the main server to your personal directory using the following command

  ```
  cp ~team8/JungkookChess.tar.gz .
  ```

- Unpack the archive file

2.3 Building, compilation, installation

- Use the command `make installation` to compile the necessary components of the executable game file
- To launch the game, use the command `bin/jungkookchess` and the game will launch
- Type `quit` to quit the game

- To uninstall, use the `cd JungkookChess` command to navigate to the game directory
- Type `make clean` to remove the files from your directory\
- Navigate to the directory that `JungkookChess` is contained in
- Type `rm JungkookChess` to delete the directory

3.1 Detailed description of data structures

Defined data structures:

Chess_piece_type; indicate the type of chess pieces.

Chess_player_color; black & white

Chess_player_type; we have either a player or an AI (uncomplete)

Chess_moveList; basically it’s similar to that student_list structure we learned in EECS 22, a doubly linked list to store move logs

Chess_move; it’s similar to the student structure we learned. Its linked with moveList and we need two position structures inside to store position and a piece structure to know what piece we are moving. There will also be a move index to generate the final movelist.

Chess_position; Indicate the position of chess pieces

Board; basically it's an 8x8 list of Chess_position structure.

Chess_player; player has a type (whether it's a player or an AI) and a color (whether it will move first).

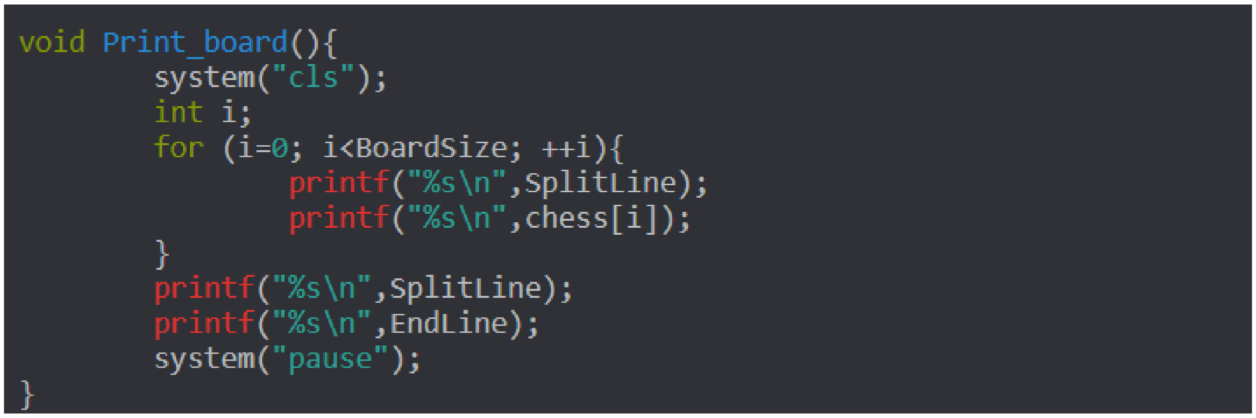
Chess_piece: Basically it stores everything, including name, type, and position of that specific chess piece.

```
struct Chess_piece{
        char                    Name[2];
        Chess_piece_type        TYPE;
        Chess_position          *Position;
        Chess_player            Player;
        int                     iPointX;
        int                     iPointY;
}Pieces[32];
```

3.2 Detailed description of functions and parameters

• Function prototypes and brief explanation
Within Chess Module: functions

1) Void Initialize_board()
    a) No parameters
    b) This function is responsible for the allocation of the 32 pieces and the 2d array board plus the movelist
2) Print board
    a) Displays the chess board on the console by iterating over the board 2d array
    b) Parameters: void
    c) prototype



```
void Print_board(){
        system("cls");
        int i;
        for (i=0; i<BoardSize; ++i){
                printf("%s\n",SplitLine);
                printf("%s\n",chess[i]);
        }
        printf("%s\n",SplitLine);
        printf("%s\n",EndLine);
        system("pause");
}
```

3) int legal_moves(Chess_move)
    a) Parameters: chess_move structure
    b) This function returns 0 if there are no legal moves and then 1 if there are legal moves
4) Int Game_State()
    a) This function indicates if it is true or false for the player A and B's turn
5) PIECE *Create_Piece(Board *board , int x, int y)
    a) This function will take in the array and the position values and create a piece and assign a pointer to the piece from the board at specified x and y position.
6) PIECE *Remove_Piece(Board *board, int x, int y)
    a) Parameters: 2d array board, two int
    b) This function will take in the array and the position values (x,y rank/file) and change the piece type into an empty if it is true that there is a piece asserted
7) Void PrintLog(Chess_MoveList)
    a) No parameters needed
8) Void JungkookMove()
    a) Generates comments that indicates whether the move was a good move or not based on the user inserted move

9) int Check(PIECE piece)
    a) Parameters: Piece structure containing type, color, etc.
    b) This function is responsible for notifying player if king is in danger of captured after a move at the end of each turn
10) Checkmate()
    a) Relies on check() and legal_moves() It notifies the user through a print statement that the player who is checked is in checkmate. Check and legal_moves need to be a 1 and 0 for the program to recognize that player ends.
11) Board *Clear_Board(Board *BOARD)
    a) Frees the pieces and the memory allocated
12) Void EndGame()
    a) Frees the board

AI Module Functions:

1) Chess_MoveList *AI_LegalMoves(Chess_Piece)
    a) This function would generate a list of possible legal moves that can be done by a given chess piece
2) Chess_move AI_BestMove(Chess_MoveList)
    a) Parameter: Chess_MoveList (list of moves)
    b) This function would rank each move and then return the highest ranked move

Note: Function names and list may vary from the final product

3.3 Detailed description of input and output formats

• Syntax/format of a move input by the user

Example: wR A4 to B4

```
printf("%s's turn: \n", player_type); // prompts player1 or player2
printf("Please make a move: ");

fgets(player_move_input, 60, stdin);

if (5 == sscanf(player_move_input], "%s %c%int to %c%int", &player_chosen_piece,
&chosen_piece_start_File, &chosen_piece_start_Rank, &chosen_piece_end_File,
&chosen_piece_end_Rank))
// example: wR A4 to B4  , 5 variables
{*convert the char Files into an int and convert it to a value 0-7
*subtract 1 from int Rank to get a value 0-7
*update board
}
```

• Syntax/format of a move recorded in the log file

Example of move recorded in log file:

Player 1 moved wR A4 to B4.

AI moved bP B3 to B4

```
printf("%s moved %s.\n", player_move_input);
```

4 Development plan and timeline

4.1 Partitioning of tasks

Schedule:
Week 1: Finished User Manual, Started on Chess board
Week 2: Finish Chess Board and Chess Structures, start Log File, start on input and output menu.
Week 3: Special Moves, Log File, Simple AI, Legal Rules
Week 4: Testing, optional more complex AI, GUI

1. **[Abhishek] - Decomposing the Program**:

2. **[ALL] - Create User Manual:** Self Explanatory. Click the link to access the doc.
   Divide Tasks Here:
   Jenny, Kiran - table of contents, glossary, introduction
   Abhishek, Roy - Installations, Functions, Back Matter
   Yuxan - Index

3. **[ALL] - Chess Structures:** Creates the data structures for the pieces.
   a. Pawn - [Kiran]
   b. Rook - [Roy]
   c. Knight - [Kiran]
   d. Bishop - [Yuxuan]
   e. Queen - [Yuxuan]
   f. King - [Roy]
   g. Chessboard Array - [Yuxuan]

4. **[Abhishek] - Special Moves:** Empessant and Castling
   Empessant: Abhishek
   Castling: Abhishek

5. **[Abhishek] - Log File** - Code that keeps track of all the moves and displays all the previous moves made by the human, and saves it into a file.

6. **[NAME] - AI:** Logic for the computer opponent of the game

7. **[NAME] - Legal Rules:** Logic that will prevent the pieces from making illegal moves
   Note: Coding the Knight could be tricky

8. **[NAME] - Testing:** autotest code for the program

9. **[Abhishek] - Documentation:** Keeps track of the progress of the program

4.2 Team member responsibilities

Abhishek - Keeping the team organized and on track, partitioning tasks, keeping a record of tasks.
Kiran - Looking over code and testing, helping make pieces.
Roy - Helping polish up deliverables, testing.
Yuxuan - Creating data structures of the pieces, tracking of the program versions, uploading and submitting files.

5 Back matter

5.1 Copyright

2022 BTS CHESS© All rights reserved.

5.2 Contact information

Women in STEM, inc.
1 STEM Way
Santa Clara, CA, 95054
womeninstem@gmail.com

5.3 Legal license

BTS Chess is not affiliated with Big Hit Entertainment or .

5.4 Disclaimer of warranty:

BTS chess does not take any responsibility for any loss of memory, files, or computer malfunction during the game operation. Please save a backup copy of all your desktop files before running this program.

• References

5.5 Index