The Most Epicest Chess Game 3000

Software Specifications

Version 1.0.0

EECS 22L

Professor Rainer Doemer

AUTHORS:

Brandon Huynh

Diego Rodriguez-Orozco

Tram Le

Sarah Catania-Orozco

Pranav Balamurali

AFFILIATION:

University of California, Irvine

Table Of Contents

GLOSSARY

Pieces - The figures used to move around the board and play the game.

Pawn - The simplest piece in chess, can move forward one space only, unless it is the first time, in which it can move either one or two spaces forward.

Rook - Piece in chess that can move either horizontally or vertically as many spaces as it wants.

Bishop - A piece that can move in any diagonal direction as far as it wants.

Knight - A piece that moves in an ‘L’ shaped direction.

Queen - A piece that can move forward, backward, left, right, or diagonally as many spaces as it wants.

King - A piece that can move in any open space around it. If it is checkmated, the game is over.

En passant - A special move in chess in which a pawn captures a horizontally adjacent enemy pawn that has just advanced two squares in one move

Castling - A special move that can be performed once per player where if neither the King nor the Rook has previously moved and there are no pieces between them, the King is moved two squares towards the Rook while the Rook is moved to the square that the King crossed.

Capture - Also known as ‘Eating’, a piece is moved onto the same space as an opposing piece, resulting in that piece being removed from the game.

Check - A player’s King is in a position where it will be captured in the next turn and can be moved out of such a position of being captured.

Checkmate - A player’s King is in a position where it will be captured next turn, and it has no options to escape being captured.

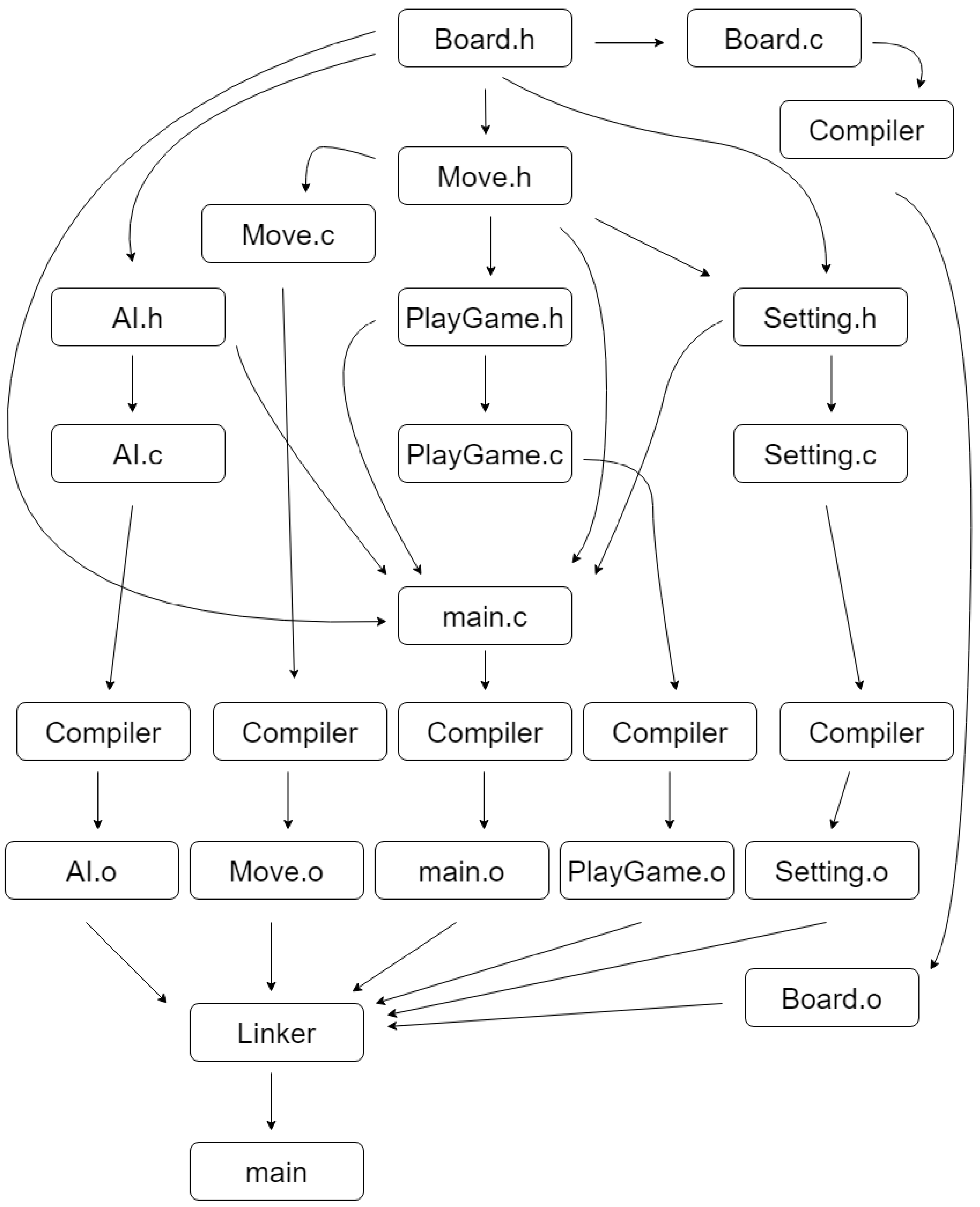
Promotion - A pawn reaches the opposite of the board and can change into any piece except a pawn or king.

1 | SOFTWARE ARCHITECTURE OVERVIEW

1.1 Main data types and structures

- char - Used to hold char arrays such as id variables to identify each piece.
- enum - Used to struct variables including color and type of piece.
- board[12][12] - 12x12 2D array of pieces

1.2 Major Software Components



1.3 Module Interfaces

API of major module functions

Setting.h

- PrintMenu
- playmenu
- Settings
- pickColor
- pickDifficulty

Board.h

- printBoard
- generateBoard
- generatePiece
- userColor

Move.h

- validMove
- isKingInCheck
- move
- fileToInt
- rankToInt

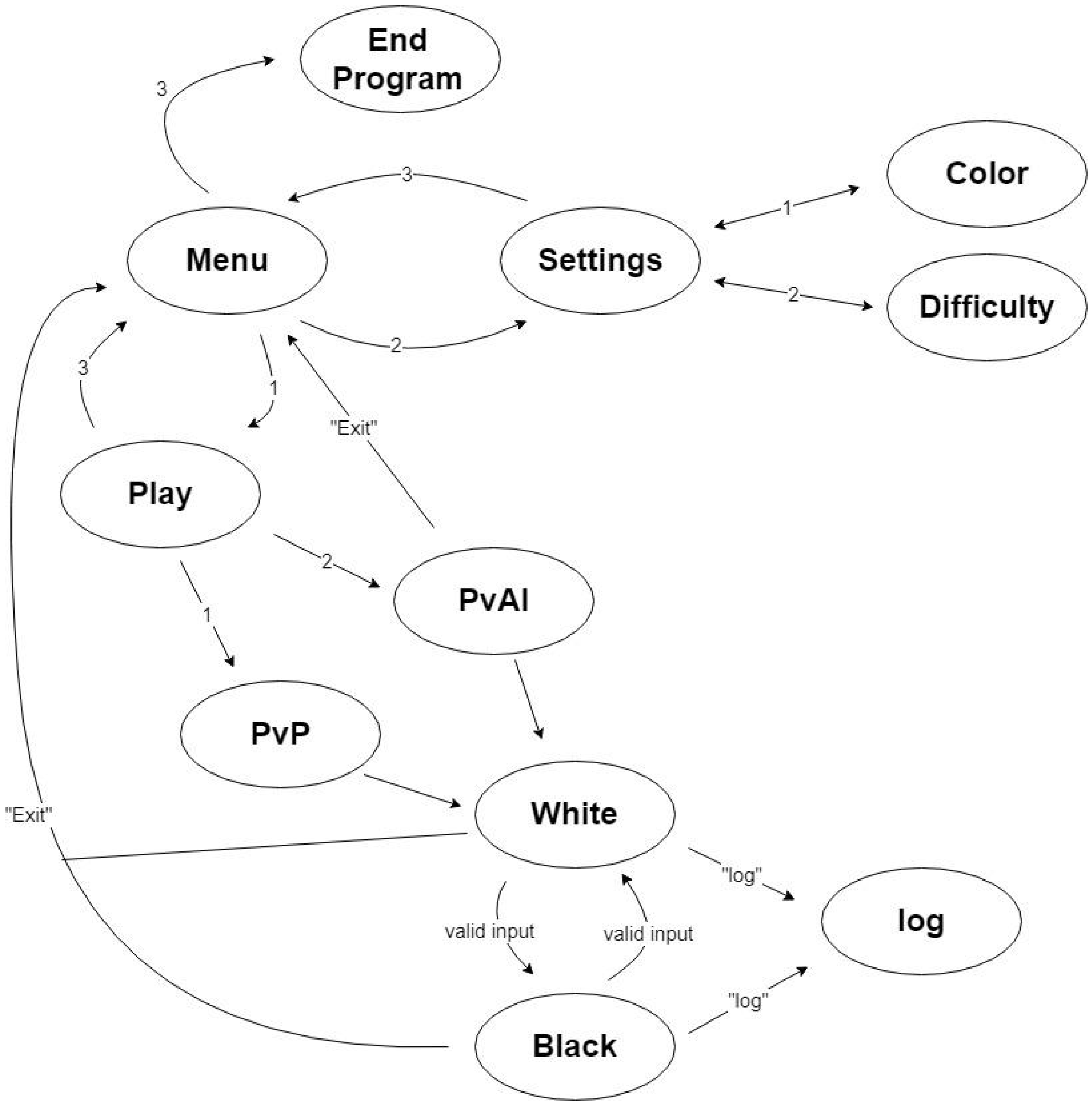
AI.h

- chooseRandPiece
- queenCapture
- generateAIMove
- AIinput

PlayGame.h

- PlayGame
- PlayGameAI
- colorCheck
- moveCheck
- promote

1.4 Overall Program Control Flow



2 | INSTALLATION

2.1 System requirements

x86-64 bit Linux System

2.2 Setup, configuration, building, compilation and installation

2.2.1 Installation from USB Drive

Unpacking the .tar.gz file to a location on your computer and run .exe file to launch the program.

2.2.2 Installation for a binary package

```
% tar -xvzf Chess_V1.0.tar.gz
% evince chess/doc/Chess_UserManual.pdf
% chess/bin/chess
```

2.2.3 Installation for a source code package

```
% tar -xvzf Chess_V1.0_src.tar.gz
% evince chess/doc/Chess_SoftwareSpec.pdf
% make
% make test
% make clean
```

2.2.4 Uninstallation

```
from
cd ..
rm -rf Chess_V1.0_src
rm -rf Chess_V1.0_src.tar.gz
```

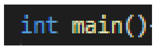
3 | DOCUMENTATION OF PACKAGES MODULES, INTERFACES

3.1 Detailed description of data structures

We decided to use a 2D array of pointers. Each of these pointers will be pointing to either NULL, an invalid piece, or one of 32 active game pieces. The pieces are defined by structs which contain color, type and identification information. After each move, the previous pointer pointing to the piece will be freed and set to NULL (identified as empty within the game). The reason we used pointers is for ease of use and efficient programming with each user input. Furthermore, char arrays will be used to track and log moves.

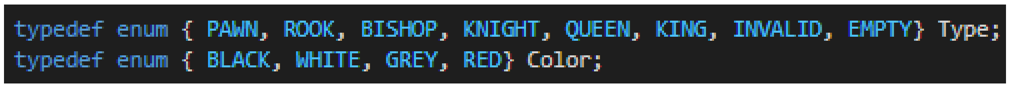
3.2 Detailed description of functions and parameters

3.2.1 Main.c
Brief explanation:The base function that will hold all other functions to piece the game together.

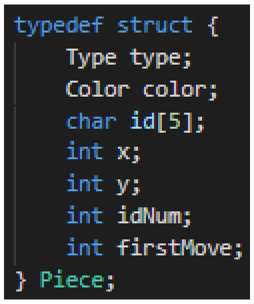


```c
int main()
```

3.2.2 Board.c/Board.h
3.2.2.1 Pieces function
Brief explanation: Gives each piece an ID and determines what it can and cannot do and which color team it is on.

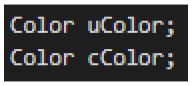


```c
typedef enum { PAWN, ROOK, BISHOP, KNIGHT, QUEEN, KING, INVALID, EMPTY} Type;
typedef enum { BLACK, WHITE, GREY, RED} Color;
```

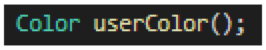


```c
typedef struct {
    Type type;
    Color color;
    char id[5];
    int x;
    int y;
    int idNum;
    int firstMove;
} Piece;
```

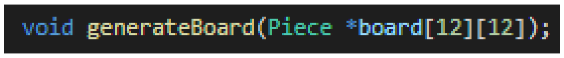
3.2.2.2 generateBoard function
Brief explanation: Generates a board with the initial positions of the pieces, where cColor and uColor are defined by userColor(). The board is a 2D array of pointers to structs of pieces.



```c
Color uColor;
Color cColor;
```

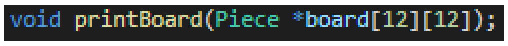


```c
Color userColor();
```



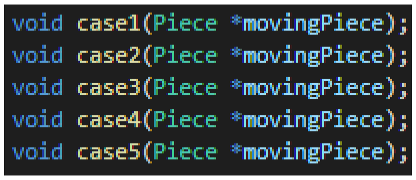
```c
void generateBoard(Piece *board[12][12]);
```

3.2.2.3 printBoard function
Brief explanation: Prints board given a 2D pointer array of structs as input parameter



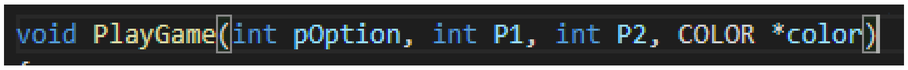
```c
void printBoard(Piece *board[12][12]);
```

3.2.2.4
Brief explanation:These functions is used to print the pieces

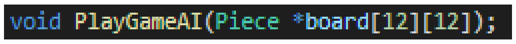


```c
void case1(Piece *movingPiece);
void case2(Piece *movingPiece);
void case3(Piece *movingPiece);
void case4(Piece *movingPiece);
void case5(Piece *movingPiece);
```

3.2.3 PlayGame.c/PlayGame.h
3.2.3.1 PlayGame and PlayGameAI function
Brief explanation: Determines what types of game the user chooses. This function will let the user choose a Human vs Human game or Human vs AI game.

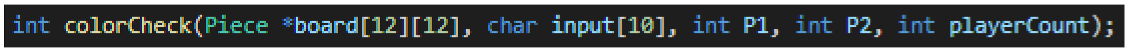


```c
void PlayGame(int pOption, int P1, int P2, COLOR *color)
```



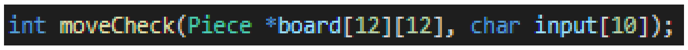
```c
void PlayGameAI(Piece *board[12][12]);
```

3.2.3.2 colorCheck function
Brief explanation: Takes the user's move input and converts it to a location on the board. This location on the board is then pointed to the color of the piece in that location and compares it to the current player and which color turn it should be. Returns 0 if false, 1 if true.



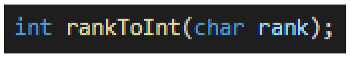
```c
int colorCheck(Piece *board[12][12], char input[10], int P1, int P2, int playerCount);
```

3.2.3.3 moveCheck function
Brief explanation: Takes the user's move input and converts it to a location on the board. This location on the board, then runs the validMove function and if the move is the first move, resets the firstMove to 1.

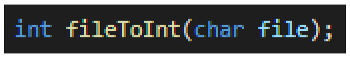


```c
int moveCheck(Piece *board[12][12], char input[10]);
```

3.2.4 Move.c/Move.h
3.2.4.1 rankToInt and fileToInt function
Brief explanation: Converts inputted file and rank into ints which can be used to determine the actual position within the board array internally

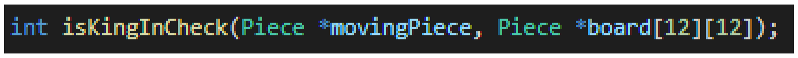


```c
int rankToInt(char rank);
```



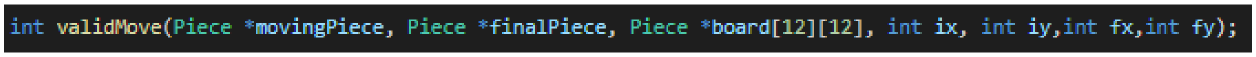
```c
int fileToInt(char file);
```

3.2.4.2 isKingInCheck function
Brief explanation: Checks if the player is put into check and disabled the color from moving any other piece unless the move is to move out of check.



```c
int isKingInCheck(Piece *movingPiece, Piece *board[12][12]);
```

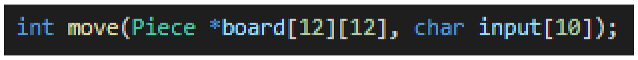
3.2.4.3 validMove function
Brief explanation: Checks if what the player/computer does is valid and prompts invalid to the player and AI if it is not.



```c
int validMove(Piece *movingPiece, Piece *finalPiece, Piece *board[12][12], int ix, int iy,int fx,int fy);
```

3.2.4.4 move function

Brief explanation: Converts the user's input into x, y coordinates and runs the validMove function to see if the move is possible. It then moves the piece from the initial location to the final location if the move is valid. move() also returns a boolean integer to use in the log function.

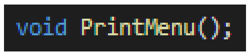


```c
int move(Piece *board[12][12], char input[10]);
```

3.2.5 Setting.c/Setting.h

3.2.5.1 PrintMenu function

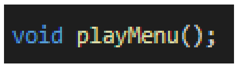
Brief explanation: Prints the initial start menu to greet the user.



```c
void PrintMenu();
```

3.2.5.2 playMenu function

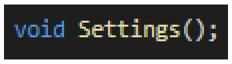
Brief explanation: this function prints the play menu and takes user input



```c
void playMenu();
```

3.2.5.3 Settings function

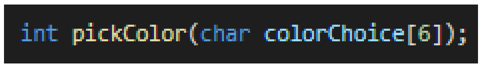
Brief explanation: Prints the settings that the user is able to change for the user and saves any changes to the settings.



```c
void Settings();
```

3.2.5.4 pickColor function

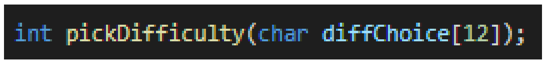
Brief explanation: The user is able to change which color they play as.



```c
int pickColor(char colorChoice[6]);
```

3.2.5.5 pickDifficulty function

Brief explanation: The user is able to change the difficulty at which the AI is played at.

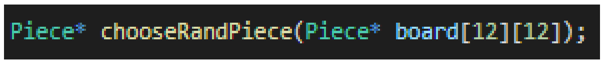


```c
int pickDifficulty(char diffChoice[12]);
```

3.2.6 AI.c/AI.h

3.2.6.1 chooseRandPiece function

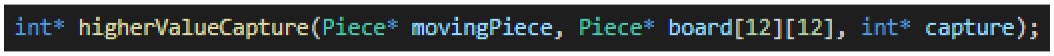
Brief explanation: This function will randomly select a piece with the random id and return it for the AI to make a move



```c
Piece* chooseRandPiece(Piece* board[12][12]);
```

3.2.6.2 higherValueCapture function

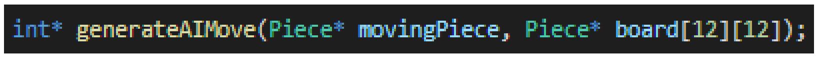
Brief explanation: This function will generate all valid captures and randomly select one if there is no capture of higher value present.



```c
int* higherValueCapture(Piece* movingPiece, Piece* board[12][12], int* capture);
```

3.2.6.3 generateAIMove function

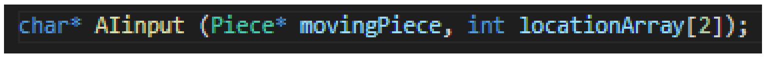
Brief explanation: This function will generate the random move of the AI, and random distance for queen, bishop, and rook.



```c
int* generateAIMove(Piece* movingPiece, Piece* board[12][12]);
```

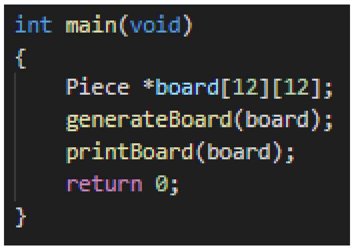
3.2.6.4 AIinput function

Brief explanation:This function will convert AI input from integer to char on external board


```c
char* AIinput (Piece* movingPiece, int locationArray[2]);
```

3.2.7 TestBoardDisplay.c

Brief explanation: This function is to test the Board display probably or not.


```c
int main(void)
{
    Piece *board[12][12];
    generateBoard(board);
    printBoard(board);
    return 0;
}
```

3.3 Detailed description of input and output formats

Syntax/Format of user input - We format the user input in the form of rank and file from previous position to next position.

Syntax/Format of user log - We plan to make the user log in the form of the universal chess move notation.

4 | DEVELOPMENT PLAN AND TIMELINE

4.1 Partitioning of Tasks

- Board Module
- Move Module
- Setting Module
- main Module
- AI Module
- PlayGame Module
- TestBoardDisplay Module

4.2 Team Member Responsibilities

Brandon - main.c, PlayGame.c, Setting.c

- Tasked to create the menu that the user will first see. Able to select: color (white or black), difficulty and timer
- Assembling program control flow, prompting user for inputs and looping invalid inputs, checking to make sure input is valid
- Creating and storing log file, allowing for access of log file in game
- Piece promotion

Diego - Board.c, Move.c

- Along with Sarah, tasked to implement the board, the pieces and how to input moves.
- Created “En Passant” inside validMove as a move for the Pawn pieces.
- Debugged and gave ideas to other team members

Tram - AI.c, Move.c, TestBoardDisplay.c, Makefile

- Along with Sarah, tasked to implement moves of pieces in the AI function, create a list of possible moves of each piece, generate random moves from the list, and implement King Castling.
- Create a Makefile to run the program. Create the doc files of README, INSTALL, COPYRIGHT.

Sarah - Board.c, Move.c, AI.c, PlayGame.c

- Along with Diego, tasked to implement the board, inputting moves, creating the pieces, initializing the board, and implementing all of the random and higher value capture functions in AI.c.

Pranav - Move.c, Board.c

- Tasked to make all the valid moves and special moves that each piece can do
- Print board look
- Debug

Back matter

Copyright

References

For any questions or concerns, please contact us:
Tram Le: tramtl1@uci.edu
Sarah Catania: catanias@uci.edu
Diego Rodriguez-Orozco: diegoar4@uci.edu
Brandon Huynh: brandlh1@uci.edu
Pranav Balamurali: pbalamur@uci.edu

Error messages

-Invalid Input
-Cannot attack own piece
-Segmentation Fault

Index

User	5, 6, 7