Monkey Chess V1.0
Developer Manual

Monkey Coding LLC: Zach Nicholson, Theodore Tang, Anthony Do, Justin Hui, Tin Vo

Table of Contents

Glossary

Array: data structure that stores multiple values in fixed size organized by matrices.

Backrank: Rows 1 and 8 of a chess board.

Bishop: A piece that can only move diagonally.

Boolean: logical true and false represented by 1 and 0.

Castling: A special move that allows the player to move two pieces at once (rook and king)

Check: When a king is attacked by another piece.

Checkmate: a check from which a king has no legal moves to avoid the attack; one of the ways that results in ending the game.

Chess Board: An 8 by 8 board with alternating colors.

En Passant: A special move that allows a pawn to capture an opponent's pawn while moving.

Enum: User defined data type in C to assign names to integral constants.

GUI (Graphical User Interface): The output of the program that the user sees.

Header File: File and function declaration and macro definitions for C files.

King: A piece that can move in all directions, left, right,up, down, and diagonally.

Knight: A piece that moves in the share of an "L", two squares in one direction and then more more at a 90-degree angle.

Pawn: The least powerful chess piece which can be promoted to any other pieces (besides the king).

Piece: A figure used to play the game of chess.

Pointer: variable that stores memory address.

Promotion: When a pawn reaches the opposite side, or their respective backrank, it can change into any other piece, except a king.

Queen: A piece that moves like a rook and bishop.

Rook: A piece that can move horizontally or vertically as far as it wants (as long as the square is unoccupied)

Struct (Structure): User defined data type in C.

1 Software Architecture Overview

1.1 Main Data Types and Structures

The board will be an 8 by 8 array with a pointer for each square that will point to a struct piece or null if it is empty.

Pieces is struct of 5 types, an enum for piece color, an enum for the piece type, a boolean for moved, and an integer piece value

Color will be an enum of two entries. Black or White, to specify which team the piece belongs to

PieceType is an enum of 6 types, King, Queen, Rook, Bishop, Knight, and Pawn.

Status will be a boolean, we will be describing if a piece has moved and if special moves have been used. If a piece has moved, then the boolean will change to true.

Value is an integer used to calculate piece values for the AI to determine which pieces should be prioritized when capturing.

Additionally, there will be a linked list for all possible moves called MoveList, containing struct Move

The Move struct will contain the start and end coordinates as well as a pointer to the piece being moved.

Pastmoves is a movelist with entries of moves that have been executed first is the oldest move and last is the most recent move.

Gameasset struct is used to handle the game through GTK.

1.2 Major Software Components

The program will contain 4 header files:

Piece.h, Board.h, AI.h, and Gui.h.

5 C files:

Piece.c, Board.c, AI.c, Gui.c, and monkechess.c

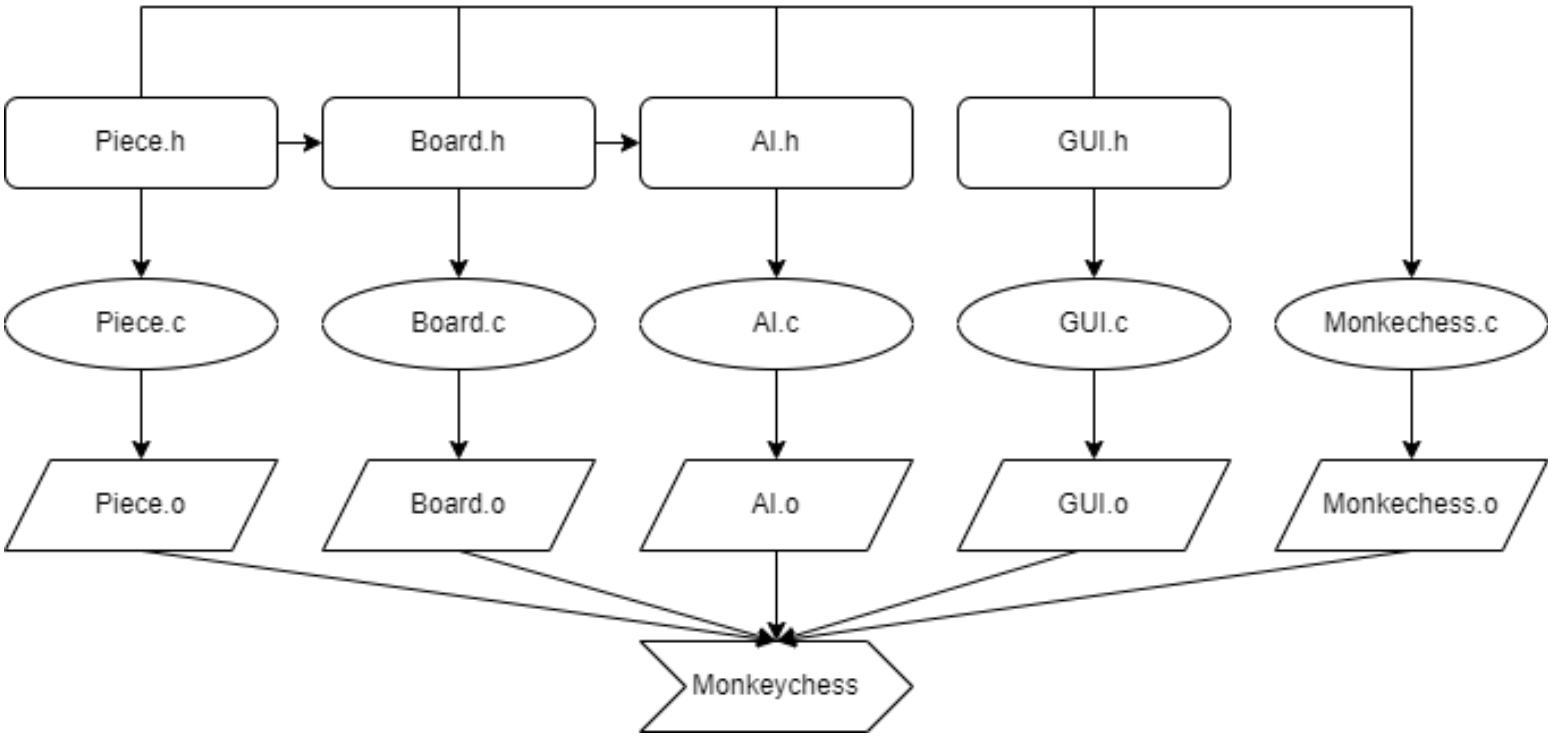
monkechess.c is the main function and will call on all other files. It will print the menu, and game logic, such as if no legal moves, you lose.

Piece.c is the lowest level and will only handle piece functions, such as promoting, creating, and deleting pieces.

Board.c will call on Piece.c, and handle all major board array changes, such as moving pieces, as well as creating and deleting the board and all pieces. Additionally, it will handle checking for legal moves. The chess log is also written inside this file.

AI.c will call Board.c, which will call it to know the general position of the pieces and call AIMove, which determines the possible moves, determine a move, and move the piece itself.

Gui.c will display the graphic interface for the entire chess program. It also takes in mouse and keyboard inputs for moves and as well as start menu options.



1.3 Module Interfaces

monkechess.c: main() calls board functions, AI functions, PrintMenu(), and PrintGame();
PrintMenu() & PrintGame() call GUI functions, Creates chessLog.txt

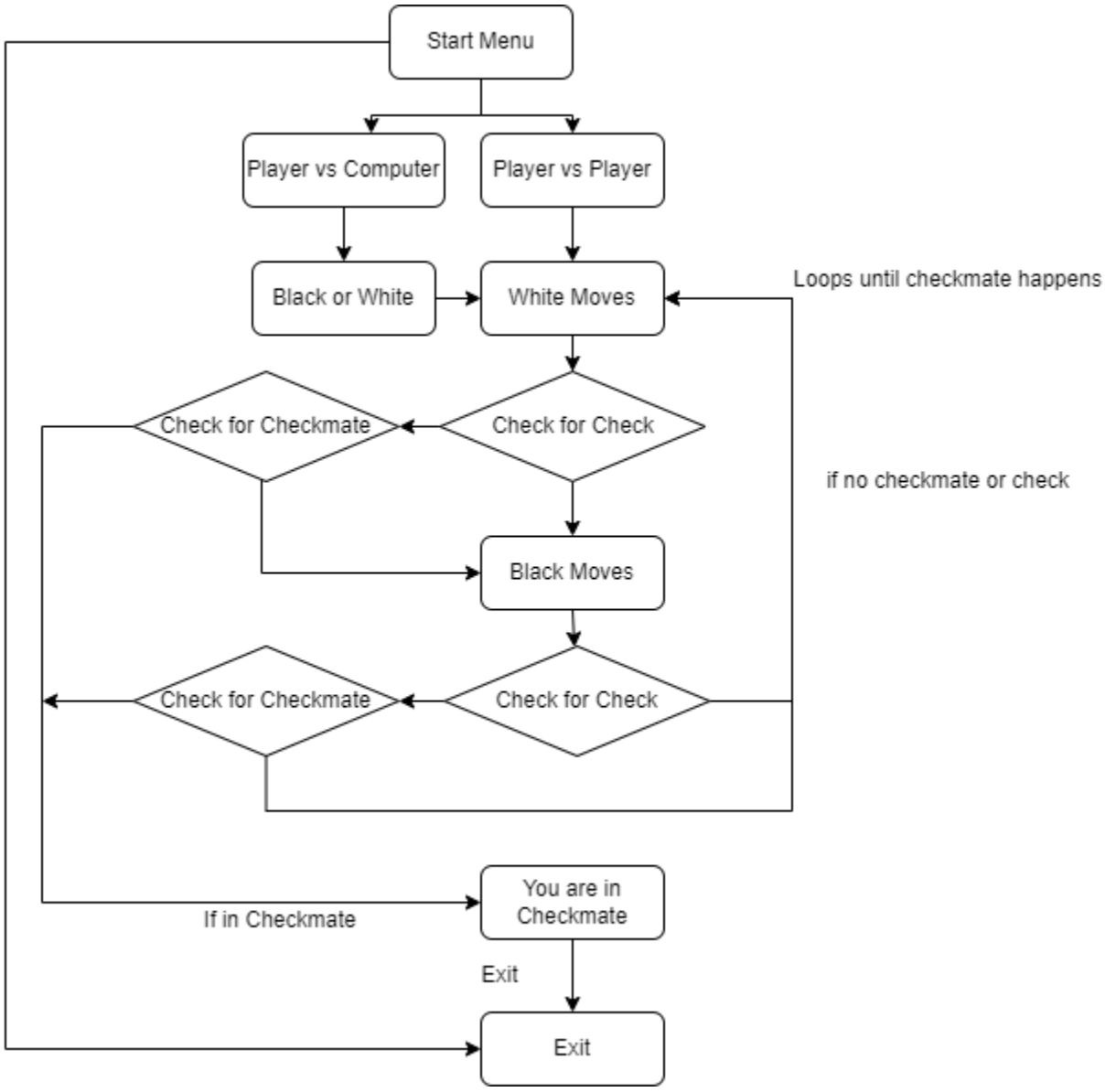
Piece.c: Does not call upon any other modules, and only takes in Piece structs to **alloc** or free memory

Board.c: Calls Piece functions, as well as handles user moves and calling AI functions. Also appends moves taken for a move log as well as checkmate.

AI.c: Takes a MoveList and chooses a move using the function RandomMove()

Gui.c: Takes the double pointer “board”, as well as game assets from monkechess.c and uses it to create GUI board and updates GUI after each move by processing user input.

1.4 Program Control Flow



2 Installation

2.1 System Requirements

- Linux operating system (Ubuntu, Debian, CentOs, etc.)
- SSH client (PuTTY, Command Prompt, Terminus, etc.)
- Terminal access
- Internet connection
- gtk-dev package to compile from source

2.2 Setup and configuration

The following instructions also can be found in the README.txt file packaged with the program.

Steps:

1. Download the .tar.gz file from a UCI EECS Server using the following cp command.

   ```
   cp /users/ugrad2/2017/winter/team13/monkeychess.tar.gz .
   ```

2. Navigate to the same directory as the tarball and use the following command to unpack the game files:

   ```
   tar -xf monkeychess.tar.gz
   ```

3. Start the game by executing the following command:

   ```
   make run
   ```

4. Enjoy The Game!!!

Uninstalling

The user needs to delete the “monkey chess” file from their linux server account using the following script in the /monkeychess/bin directory:

```
./uninstmkychess.sh
```

2.3 Building, Compilation, and Installation

The following instructions can be found in the INSTALL.txt file to compile Monkechess from the source code.

Steps:

1) Download the .tar.gz file from a UCI EECS Server using the following cp command.

```
cp /users/ugrad2/2017/winter/team13/monkeychess.tar.gz .
```

2) Navigate to the same directory as the tarball and use the following command to unpack the game files:

```
tar -xf monkeychess.tar.gz
```

3) Compile the game by executing the following command:

```
make
```

4) Start the executable using:

```
make run
```

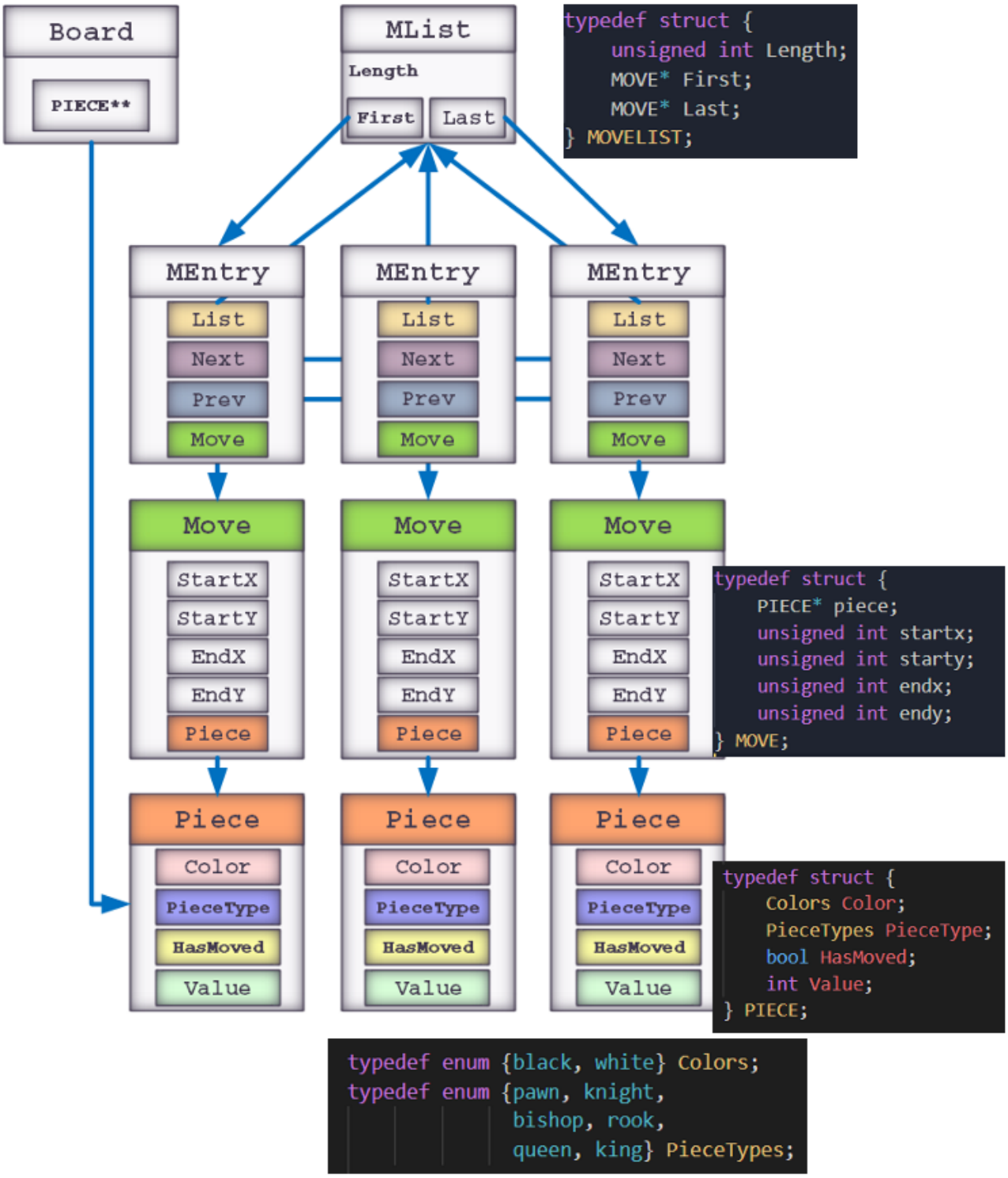
or

`./bin/monkechess` from the directory

5) Enjoy The Game!!!

3 Packages, Modules, Interfaces

3.1 Detailed description of data structures



```
typedef struct gameasset{
    char **dark_white  ;//
    char **dark_black  ;//
    char **light_white ;//
    char **light_black ;//
    char  *light       ;//
    char  *dark        ;//
    PIECE** board;
    Colors  team;
    MOVELIST *pastmoves;
}GAME_ASSET;
```

3.2 Detailed description of functions and parameters

AI.c

RandomMove function has a **pointer parameter** moveList that goes and first randomly picks a piece and then randomly picks a move that the piece can make.

```
void RandomMove(PIECE** board, Colors AIColor, MOVELIST* pastlist);
```

Board.c

CreateBoard is a function without any parameters. It creates the 8 by 8 array for the board and puts a pointer in each entry.

DeleteBoard removes all pieces and sets all the pointers in the array to NULL and frees the board.

CreateMove creates a move structure that will be appended to a move list with parameters piece startx, starty, endx, and endy.

DeleteMove frees the move structure in its parameter and the memory allocated to it

AppendMove appends the move parameter into the moveList parameter.

LegalMoveCheck takes a piece on the board and a move as parameters and checks if the move is legal.

MovePiece has parameters piece on the board, the move, and moveList. It will run LegalMoveCheck and as long as it is legal it will move the piece.

PossibleMoves creates a list of moves a piece can take using the parameters of board, piece itself, start and end positions, previous moves, and color of the turn.

MovePiece moves the piece structure to the new board address. It also checks for special moves as well as it it is in checkmate.
DeleteMoveList deletes and frees up memory after a MoveList is not needed.
PrintBoard prints out an ASCII chess board in the terminal.
AppendChessAction stores the move taken into a text document. It takes the start and end positions and the integer *taken* to show if the piece has been captured or not.
InCheck scans and detects if the player is in check.

```
PIECE** CreateBoard(void);

// Deletes the board pointer as well as the piece structs and
// any other structure used by the Board.
void DeleteBoard(PIECE** board);

// Creates a move structure which will be appended in another
// function.
MOVE* CreateMove(PIECE* piece, unsigned int startx, unsigned int starty, unsigned int endx, unsigned int endy);

// This function will delete the move structure by freeing
// all the memory allocated.
void DeleteMove(MOVE* move);

// This will append a move into the move list.
void AppendMove(MOVELIST* moveList, MOVE* move);

// This function will check if the out of the possible moves
// that a piece can make is legal or not. This will return an
// integer that determines this.
int LegalMoveCheck(PIECE** Board, MOVE* move, MOVELIST* pastlist, Colors turnColor);

// This will check all possible moves avaiable to a specific chess
// piece and return a list of the moves.
// creates the move list
MOVELIST* PossibleMoves(PIECE** Board, PIECE* piece, int startx, int starty, MOVELIST* pastlist);

// This function will move the piece structure to the new
// board address.
int MovePiece(PIECE** board, PIECE* piece, unsigned int startx, unsigned int starty, unsigned int endx, unsigned int endy, MOVELIST* pastlist, Colors turnColor);

//deletes movelist and delete entries combined
void DeleteMoveList(MOVELIST *moveList);

//void Movement(PIECE* piece);

void PrintBoard(PIECE** board);

// Adds the move made to a text file acting as a move log
void AppendChessAction(unsigned int startmovex, unsigned int startmovey, unsigned int endmovex, unsigned int endmovey,
                       unsigned int taken);

int InCheck(PIECE** board, Colors turnColor);
```

Piece.c

CreatePiece with parameters color and PieceType which are both enums. It creates a piece and gives it type (pawn, knight, bishop, rook, queen, king)
DeletePiece has parameters for piece and frees said piece.
PromotePiece which has parameters for the piece being promoted and what it's being promoted to (queen, rook, bishop, knight).

```c
//Creates and allocates memory for a new piece, returns a pointer to new piece
PIECE *CreatePiece(enum color, enum PieceType);

// Release the memory space for the Piece values
// Release the memory space for the Piece
void DeletePiece(PIECE *piece);

// Promote a piece
// Changes a pieces type and value
PIECE *PromotePiece(PIECE *piece, enum PieceType);
```

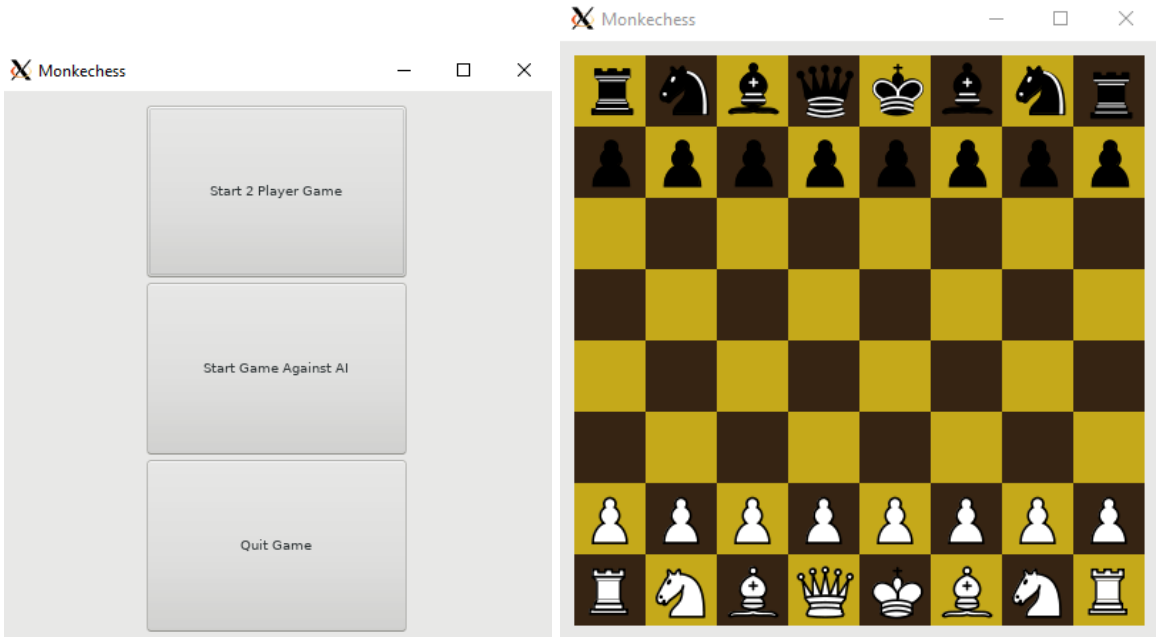
Gui.h

InitBoard background reference color of the chess board. Tells GUI color of background.
DrawBoard The function redraws the GUI board.Shows what type of space to show
CoordToGrid This function converts x and y coordinates into inputs used for GUI chess board.
Parameter c_x and c_y are the coordinates while *g_x and *g_y are the grid.
on_delete_event Deletes/quits GUI interface when mouse clicks on it.
area_click gets mouse click, calls CoordToGrid() to convert, and then calls MovePiece() to move chess pieces
Make_ai_move calls RandomMove() to have AI make a move using Asset.

```c
void InitBoard(GAME_ASSET *Asset); // Initialize a background reference for the GUI to check

void DrawBoard( GAME_ASSET *Asset); // Fills a GTK table with the correct image for each space on the board

void CoordToGrid(int c_x, int c_y, int *g_x, int *g_y); // Converts absolute click coordinates to gameboard coordinates

gboolean on_delete_event(GtkWidget *widget,
        GdkEvent  *event,
        gpointer   data);

gint area_click(GtkWidget *widget,            // GtkCallback function triggered on mouse click
                GdkEvent  *event,             // gets coords and calls CoordToGrid()
                gpointer  data);


gboolean make_ai_move(GAME_ASSET* Asset); // calls AI to make a move and display it
```

3.3 Detailed description of input and output formats

All the inputs from the user will be through their mouse using **GTK**. The program will read the mouse clicks, process what piece was clicked on and where it was clicked to, and update the graphics to output that information. The program will also log each movement made by both the player and the AI in **algebraic notation** in a text document. When promoting pieces, the program will ask the user to input a character in the terminal, then on the GUI the piece will be promoted to the corresponding piece type.



4 Development Plan and Timeline

4.1 Partitioning of tasks

Piece Movement

Board Manipulation

Move Operations

AI Operations

GUI Functionality

Piece Manipulation

Documentation

(More partitions decided later)

4.2 Team member responsibilities

Piece Movement: Justin and Anthony

Board Operations: Tin, Theo, Justin

Move Operations: Tin, Theo, Justin, Anthony

AI Operations: Everyone

GUI Functionality: Anthony and Zach

Piece Manipulation: Justin, Anthony, Zach

Documentation: Tin and Theo

References

Chess.com. “How to Play Chess: 7 Steps to Get You Started.” *Chess.com*, Chess.com, 18 Aug. 2021, https://www.chess.com/learn-how-to-play-chess.

“Diagrams.net - Free Flowchart Maker and Diagrams Online.” *Flowchart Maker & Online Diagram Software*, https://app.diagrams.net/.

“GTK+ 2 Reference Manual.” *GTK+ 2 Reference Manual: GTK+ 2 Reference Manual*, https://developer-old.gnome.org/gtk2/stable/index.html.

“Microsoft Visio Professional.” Microsoft Corp., 2001.

Copyright

Index