ChessCooker v3.0

By Brandon Dinh, Christopher Dao, Miguel Vidal, Ryan Lockie
Affiliation: The Chefs

Team7 - The Chefs

Table of Contents

Glossary

Array
This is a data type in C that can be used to store multiple variables of the same type with one reference, in a list style format.

Bishop
Piece that can move diagonally across the board until it reaches the end or a different piece. Captures in the same way it moves.

Black
One set of pieces is black, these always make the second move.

Capture
Removing an opponent's piece from the chessboard

Castling
A move in which the king is shifted two squares towards a rook on the same rank, placing the rook in the square crossed by the king. This is only possible when neither the king or rook have moved, the space between the two pieces is empty, and the king isn't passing through or finishing on a square that can be attacked by an enemy piece.

Check
If the king is under check, it means that if the king isn't moved in the next turn, the king will be captured. If the king is under check, the only legal moves left are ones that remove the king from check.

Checkmate
If the king is under check and the player has no options to prevent him from being captured the next turn. This is the win condition of chess.

Chessboard
The chessboard is an 8x8 grid, made up of ranks and files.

Doubly Linked List
A list where successive elements have pointers to the next element in the list and pointers to the previous element in the list.

Enum
User defined data type where members are given a constant integer value that also represents the member name.

En Passant

A move in which a pawn can capture another pawn that just moved two tiles out of its initial position, by moving in front of the enemy pawn, rather than on top of it. This must be done right after the pawn being attacked moved to spaces out of its initial position, or it is considered to be an invalid move.

File
A file is a vertical column on the board. There are 8 files, labeled a-h.

Header File
A file that initializes all functions that will be included in a specific c file, also determining the return types and Parameters of each function call.

King
Piece that can move diagonally, vertically, and horizontally, but only one space in the chosen direction. Captures in the same way it moves. Can come under check and if checkmated, the player loses.

Knight
Piece that can move eight different squares which are two steps forward plus on step sideway from its position. Capture the space it lands on.

Parameters
Values that are input into a function for various uses.

Pawn
Piece that can move vertically up the board one space or two spaces when moving from the starting rank. Capture one space diagonally in front of it, or if available alternatively perform the en passant move. Pawns are also the only pieces that can be promoted.

Promotion
If a pawn reaches the other side's starting rank, they can be turned into any other piece besides a king

Piece
It is either a king, queen, pawn, rook, bishop, or a knight.

Stalemate
Occurs when a King is not in check and on move, but has no legal moves that can be made. This results in the game being over and a draw.

Struct
User defined data type in C that allows a group of data items of different data types.

Queen
Piece that can move diagonally, vertically, and horizontally. Spaces that can be moved by this piece are only limited by the board and other pieces being in the way. Captures in the same way it moves.

Rank
A rank is a horizontal row on the board. There are 8 ranks, labeled 1-8.

Return Type
Data type of the value that a function returns.

Rook
Piece that can move vertically, and horizontally across the board until it reaches the end or a different piece. Capture in the same way it moves.

White
One set of pieces is white, these always move first.

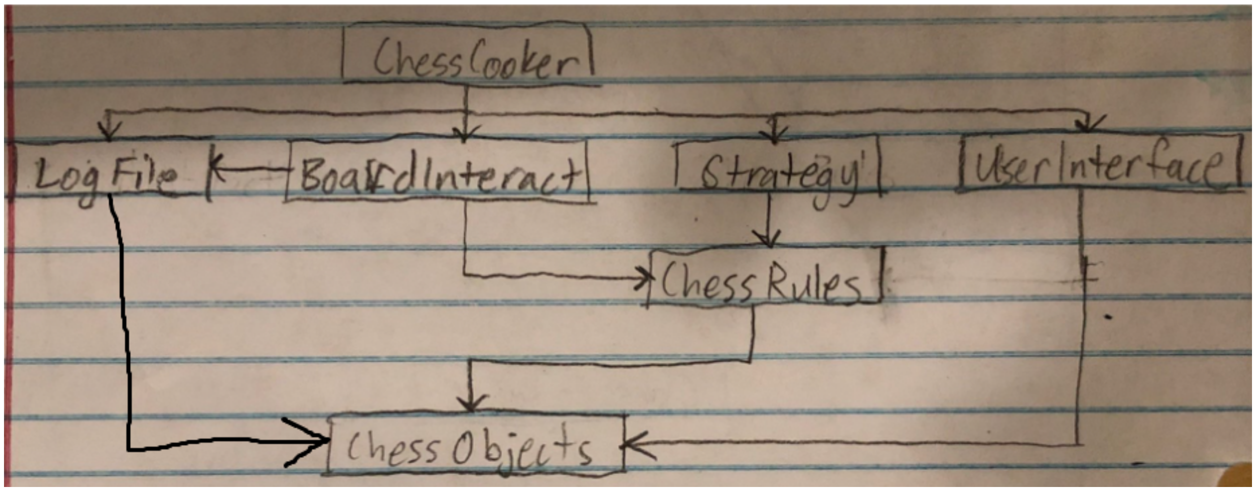
2-D Array
An array of multiple one-dimensional arrays.

1 Software Architecture Overview

1.1 Main Data Types and Structures

1. a player (who is either black or white)
2. a piece type (either king, queen, bishop, knight, rook, or pawn)
3. a piece (a combination of player/color and piece type)
4. a board (8x8 matrix of squares, each with or without a piece on them)
5. a position of a piece (known to the user as ”e2”, for example)
6. a move (a combination of a start and end position, e.g. ”e2 e4”)
7. a log (a list of moves)
8. a list of pieces (to keep track of captured pieces)

1.2 Major Software Components



1.3 Module Interfaces

ChessObjects.c

- CreatePiece()
  - Allocate memory for a t_Piece
- DeletePiece()
  - Deallocate memory for a t_Piece
- CreateMove()
  - Allocate memory for a t_Move
- DeleteMove()
  - Deallocate memory for a t_Move
- CreateMoveList()

    - Allocate memory for a t_MoveList
- DeleteMoveList()
    - Deallocate memory for a t_MoveList
- AppendMove()
    - Appends a t_Move to a t_MoveList
- OppositePlayer()
    - Switches the players (if white, become black, if black, become white)

BoardInteract.c

- LookupPiece()
    - on a board, lookup a piece at a given position (basically finding the piece)
- PlacePiece()
    - on a board, put a given piece onto a given position
- MovePiece()
    - on a board, move a piece from a position to another
- FileToInt()
    - Converts file of a position to format readable by t_Board
- RankToInt()
    - Converts rank of a position to format readable by t_Board
- IntToFile()
    - Converts int to File on a chessboard
- IntToRank()
    - Converts into to Rank on a chessboard
- DeleteBoard()
    - Deletes a t_Board
- CopyBoard()
    - Copies one t_Board to another t_Board
- IsLegalMove()
    - Returns whether or not a given t_Move is a legal move

UserInterface.c

- DrawBoard()
    - display board (graphic/textual)
- DrawPieces()
    - Draw pieces on the board
- DrawPromotionMenu()
    - Draws the menu prompting for the desired piece to be promoted to once a promotion occurs
- ValidPositionClickedPromotion()
    - Makes sure the promotion menu is being properly clicked on
- PromotionType()
    - Returns the type of piece that the user has chosen to promote to
- LoadPieceBMP()

    - Loads the picture of each piece
- GetPosition()
    - Returns the position on the board according to the user's click
- ValidPositionClicked()
    - Returns 1 if the position that was clicked was valid
- DrawPlayerMenu()
    - Displays the menu for choosing to be the black or white player
- PlayerChoice()
    - Returns White or Black according to the location of the player's click
- HandleChoosePlayer()
    - Handles the events that result in the user choosing a player
- DrawGamemodeMenu()
    - Displays the menu for the user to choose a gamemode
- HandleChooseGamemode()
    - Handles the events that result in the user choosing a gamemode
- RunHumanHumanMode()
    - Runs the human vs. human gamemode
- RunHumanComputerMode()
    - Runs the human vs. computer gamemode
- RunComputerComputerMode()
    - Runs the computer vs. computer gamemode

ChessRules.c

- ReachablePositions()
    - for a piece on the board, compute all reachable positions
- PieceLegalMoves()
    - for a piece on the board, compute all legal moves
- InCheck()
    - on a board, check whether or not a player's king is in check
- Checkmate()
    - on a board, check whether or not a player's king is in checkmate
- LegalMoves()
    - for a player and a given board, compute all legal moves
- CastlingMove()
    - It looks for any possible castling that is legal

Strategy.c

- ChooseBestMove()
    - from a list of moves, select the best one
- AnalyzeCapture()
    - It look for possible pieces to capture and decides which piece is best ta take if it is given more than one option to capture a piece
- AnalyzeFuture()

    - This will check possible future moves with a simulated virtual board

LogFile.c

- DeleteLog()
    - Delete log file
- RecordMove()
    - Records move and writes in log file

1.4 Overall Program Control Flow

1. Setup
    a. User is prompted to choose a gamemode: Human vs Human, Human vs Computer, or Computer vs Computer
    b. User is prompted to choose player: White or Black
2. Display the board
3. Repeat
    a. White player makes a move
    b. Display the board
    c. If black is in checkmate, white wins!
    d. Black player makes a move
    e. Display the board
    f. If white is in checkmate, black wins!
    g. If a stalemate occurs
        i. Game ends

2 Installation

2.1 System Requirements, Compatibility

- PC with x86_64 server
- Linux OS (CentOS-7-x86_64)

2.2 Setup and configuration

- Open a server that can run Linux
- Configure settings to support X11 forwarding
- Need to have an Xming server open

2.3 Building, Compilation, Installation

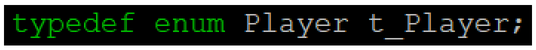
- Building
    - “make tar”
- Compilation
    - Can run “make clean” then “make”
- Installation
    - gtar xvzf Chess_V1.0_src.tar.gz
    - cd Chess_V1.0_src
    - cd bin
    - Run ChessCooker executable with “./ChessCooker”

3 Documentation of Packages, Modules, Interfaces

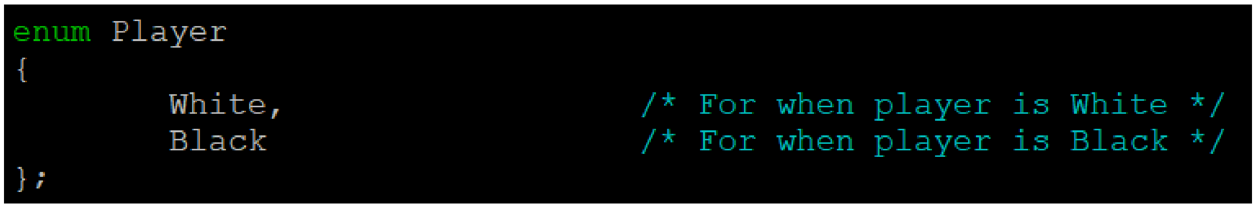
3.1 Detailed Description of Data Structures

t_Player

Description: A t_Player is an enum with two members: White and Black. A t_Player is meant to distinguish between whether a given piece belongs to White side or Black side.



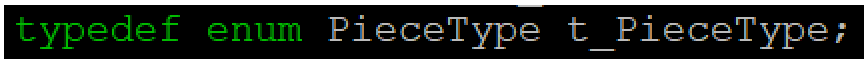
```
typedef enum Player t_Player;
```



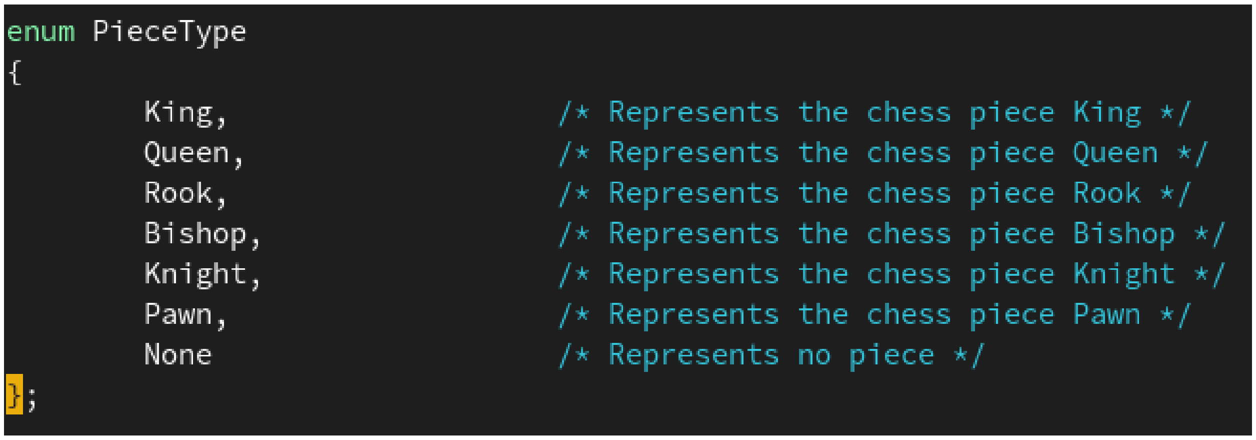
```
enum Player
{
        White,                  /* For when player is White */
        Black                   /* For when player is Black */
};
```

t_PieceType

Description: A t_PieceType is an enum with seven members: King, Queen, Rook, Bishop, Knight, Pawn, and None. This data structure is to distinguish the type of a piece on the chessboard, which can be one of the seven members of the enum which correspond to the same types in chess.



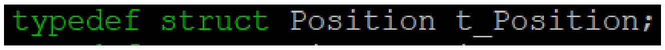
```
typedef enum PieceType t_PieceType;
```



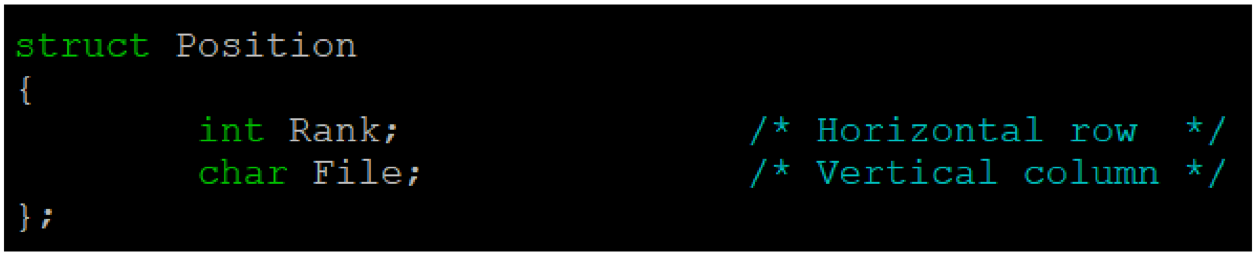
```
enum PieceType
{
        King,                   /* Represents the chess piece King */
        Queen,                  /* Represents the chess piece Queen */
        Rook,                   /* Represents the chess piece Rook */
        Bishop,                 /* Represents the chess piece Bishop */
        Knight,                 /* Represents the chess piece Knight */
        Pawn,                   /* Represents the chess piece Pawn */
        None                    /* Represents no piece */
};
```

t_Position

Description: A t_Position is a struct with two members: Rank and File. Rank is of type int and is an integer between 1-8 inclusive corresponding to the eight horizontal rows of a chessboard. File is of type char and is a character between A-H inclusive corresponding to the eight vertical columns of a chessboard. A t_Position is meant to represent a position on the chessboard.



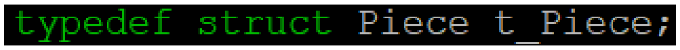
```
typedef struct Position t_Position;
```



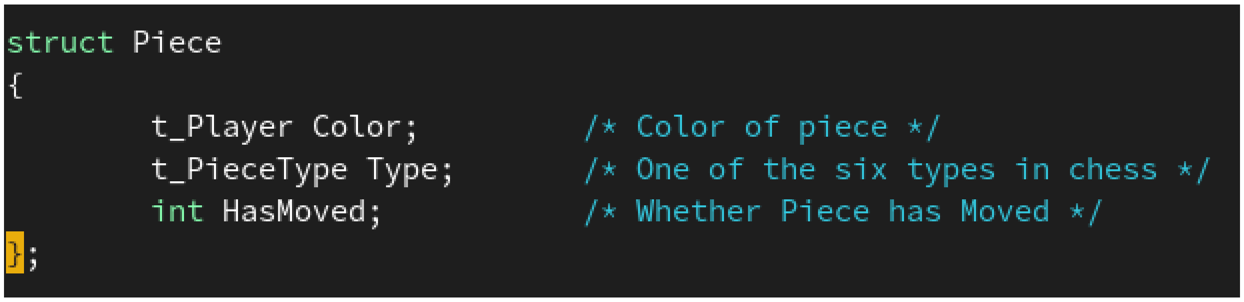
```
struct Position
{
        int Rank;                   /* Horizontal row  */
        char File;                  /* Vertical column */
};
```

t_Piece

Description: A t_Piece is a struct with three members: Color, Type, and HasMoved. Color is of type t_Player and identifies whether t_Piece is White or Black. Type is of type t_PieceType and identifies which of the six pieces in chess—King, Queen, Rook, Bishop, Knight, Pawn—that a t_Piece is. HasMoved is of type int and indicates the number of times that the piece has moved.



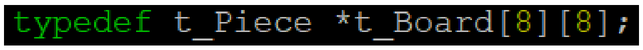
```
typedef struct Piece t_Piece;
```



```
struct Piece
{
        t_Player Color;         /* Color of piece */
        t_PieceType Type;       /* One of the six types in chess */
        int HasMoved;           /* Whether Piece has Moved */
};
```

t_Board

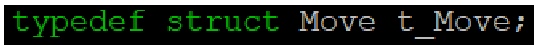
Description: A t_Board is a 2-D array of pointers to a t_Piece with size of 8 rows and 8 columns. A t_Board is meant to represent a chessboard where each element of the 2-D array is either a pointer to a t_Piece representing a chess piece or NULL.



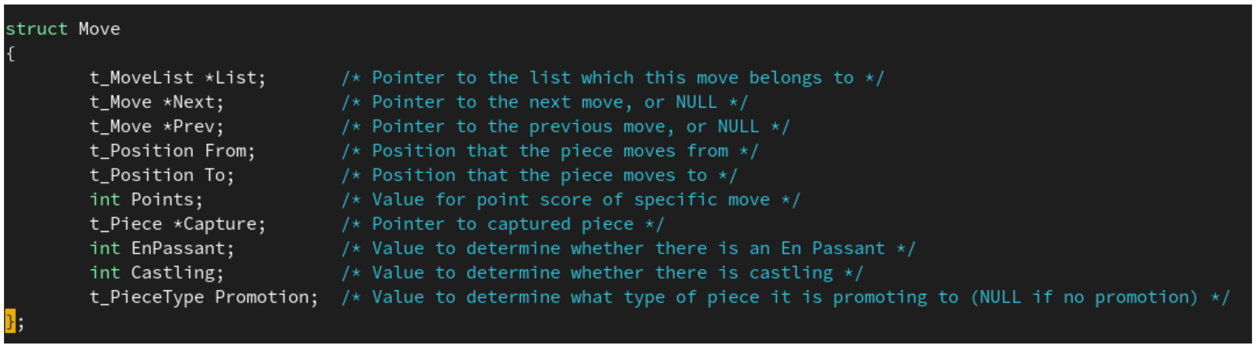
```
typedef t_Piece *t_Board[8][8];
```

t_Move

Description: A t_Move is a struct with ten members: List, Next, Prev, From, To, Points, Capture, EnPassant, Castling, Promotion. List is of type t_MoveList and is a pointer to the t_MoveList that a t_Move belongs to. Next is of type t_Move and is a pointer to the move that occurs after. Prev is of type t_Move and is a pointer to the move that occurs before. From is of type t_Position and represents the position that a piece is moving from. To is of type t_Position and represents the position that a piece is moving to. Points is of type int and indicates the value of a move to the computer player. Capture is of type t_Piece* and points to the piece that will be captured as a result of the move. EnPassant is of type int and indicates whether the move is an en passant move or not. Castling is of type int and indicates whether or not the move is a castling move or not. Promotion is of type t_PieceType and indicates the type of piece that the piece performing the move will be promoted to (if applicable). A t_Move is meant to represent a player's move in chess.



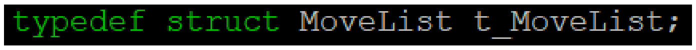
```
typedef struct Move t_Move;
```



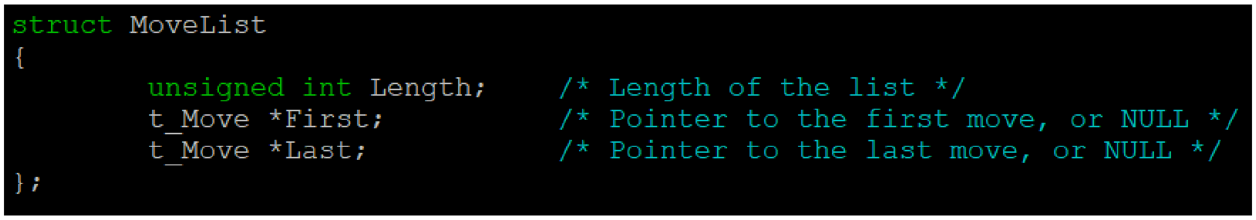
```
struct Move
{
        t_MoveList *List;       /* Pointer to the list which this move belongs to */
        t_Move *Next;           /* Pointer to the next move, or NULL */
        t_Move *Prev;           /* Pointer to the previous move, or NULL */
        t_Position From;        /* Position that the piece moves from */
        t_Position To;          /* Position that the piece moves to */
        int Points;             /* Value for point score of specific move */
        t_Piece *Capture;       /* Pointer to captured piece */
        int EnPassant;          /* Value to determine whether there is an En Passant */
        int Castling;           /* Value to determine whether there is castling */
        t_PieceType Promotion;  /* Value to determine what type of piece it is promoting to (NULL if no promotion) */
};
```

t_MoveList

Description: A t_MoveList is a struct with three members: Length, First, and Last. Length is an unsigned int that represents the length of the list. First is of type t_Move and is a pointer to the first move in the list. Last is of type t_Move and is a pointer to the last move in the list. A t_MoveList is meant to keep track of moves using a Doubly Linked List for the purpose of keeping a log of the moves made, or grouping them together in order to sort them based on some sort of criteria like whether that move is a legal move.



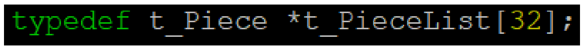
```
typedef struct MoveList t_MoveList;
```



```
struct MoveList
{
        unsigned int Length;    /* Length of the list */
        t_Move *First;          /* Pointer to the first move, or NULL */
        t_Move *Last;           /* Pointer to the last move, or NULL */
};
```

t_PieceList

Description: A t_PieceList is an array of pointers to a t_Piece with an array size of 32. A t_PieceList is meant to keep track of pieces that have been captured in a list.



```
typedef t_Piece *t_PieceList[32];
```

3.2 Detailed Description of Functions and Parameters

Function: int main(void)
Description: Main function in ChessCooker.c with no parameters. The return type is int. The purpose of the main function is to run and control the overall program flow of the Chess program.

Function: t_Piece *LookupPiece(t_Board *board, t_Position position)
Description: Function in BoardInteract.c with two parameters: board of type t_Board and position of type t_Position. The return type is a pointer to a t_Piece. The purpose of LookupPiece is to look up a piece on the given board at a given position (basically finding the piece).

Function: void PlacePiece(t_Board *board, t_Piece *piece, t_Position position)
Description: Function in BoardInteract.c with three parameters, board of type t_Board, piece of type t_Piece, and position of type t_Position. There is no return type. The purpose of PlacePiece is to put a given piece onto a given position on the given board.

Function: void MovePiece(t_Board *board, t_Move *move, int moveMode, t_Player color, t_PieceList pieces)
Description: Function in BoardInteract.c with five parameters: board of type t_Board, move of type t_Move, moveMode of type int, color of type t_Player, pieces of type t_PieceList. There is no return type. The purpose of MovePiece is to move a piece from one position to another on the given board.

Function: int FileToInt(t_Position position)
Description: Function in BoardInteract.c with one parameter: position of type t_Position. The return type is an int. The purpose of FileToInt is to take the value of File of the t_Position and convert that into an integer value between 0 and 7 so that other functions can work with the board more easily.

Function: int RankToInt(t_Position position)
Description: Function in BoardInteract.c with one parameter: position of type t_Position. The return type is an int. The purpose of FileToInt is to take the value of Rank of the t_Position and convert that into an integer value between 0 and 7 so that other functions can work with the board more easily.

Function: char IntToFile(int file)
Description: Function in BoardInteract.c with one parameter: file of type int. The return type is a char. The purpose of IntToFile is to go from an integer value between 0 and 7 and convert it to a char a - h representing the files on a chessboard.

Function: int IntToRank(int rank)
Description: Function in BoardInteract.c with one parameter: rank of type int. The return type is an int. The purpose of IntToRank is to go from an integer value between 0 and 7 and convert it to an integer value 1-8 representing the rank on a chessboard.

Function: void DrawBoard(t_Board *board, SDL_Surface *screen)

Description: Function in UserInterface.c with two parameters: board of type t_Board and screen which is a pointer to a SDL_Surface. There is no return type. The purpose of DrawBoard is to generate the graphical chessboard on the screen.

Function: void DrawPieces(t_Board *board, SDL_Surface *screen, SDL_Surface *pawn_w, SDL_Surface *king_w, SDL_Surface *queen_w, SDL_Surface *rook_w, SDL_Surface *bishop_w, SDL_Surface *knight_w, SDL_Surface *pawn_b, SDL_Surface *king_b, SDL_Surface *queen_b, SDL_Surface *rook_b, SDL_Surface *bishop_b, SDL_Surface *knight_b);
Description: Function in UserInterface.c with fourteen parameters: board of type t_Board, screen which is a pointer to a SDL_Surface, and twelve pointers to different SDL_Surface which represent one of 12 unique pieces on a chess board and are used to draw that piece on the board. The purpose of DrawPieces is to draw the pieces on the graphical chessboard considering their position on board.

Function: t_MoveList *ReachablePositions(t_Board *board, t_Position position)
Description: Function in ChessRules.c with two parameters: board of type t_Board and position of type t_Position. The purpose of ReachablePositions is to compute all reachable positions on the given board for the piece at the given position.

Function: int InCheck(t_Board *board, t_Player color)
Description: Function in ChessRules.c with two parameters: board of type t_Board and color of type t_Player. The return type is of type int. The purpose of InCheck is to check whether or not a player’s king is in check on the given board.

Function: int Checkmate(t_Board *board, t_Player color, t_PieceList pieces)
Description: Function in ChessRules.c with three parameters: board of type t_Board, color of type t_Player, and pieces of type t_PieceList. The return type is of type int. The purpose of Checkmate is to check whether or not a player’s king is in checkmate on the given board.

Function: t_MoveList *PieceLegalMoves(t_Board *board, t_Position position, t_Move *PrevMove)
Description: Function in ChessRules.c with three parameters: board of type t_Board, position of type t_Position, and PrevMove of type t_Move. The return type is a pointer to a t_MoveList. The purpose of PieceLegalMoves is to compute all legal moves on the given board for the piece at the given position.

Function: t_MoveList *LegalMoves(t_Board *board, t_Player color, t_Move *PrevMove, t_PieceList pieces)
Description: Function in ChessRules.c with three parameters: board of type t_Board, color of type t_Player, and PrevMove of type t_Move. The return type is a pointer to a t_MoveList. The purpose of the LegalMoves is to compute all legal moves for a player and a given board.

Function: t_Move *ChooseBestMove(t_MoveList *moves, t_Player color, t_Board board, int loops, t_PieceList pieces)
Description: Function in Strategy.c with five parameters: moves of type t_MoveList, color of type t_Player, board of type t_Board, pieces of type t_PieceList, and loops of type int. The return type is of type t_Move. The purpose of ChooseBestMove is to select the best chess move from a list of legal moves.

Function: t_Piece *CreatePiece(t_Player color, t_PieceType type)
Description: Function in ChessObjects.c with two parameters: color of type t_Player and type of type t_PieceType. The return type is a pointer to a t_Piece. The purpose of CreatePiece is to allocate the necessary memory space to create a t_Piece with the given parameters.

Function: void DeletePiece(t_Piece *piece)
Description: Function in ChessObjects.c with one parameter: piece which is a pointer to a t_Piece. There is no return type. The purpose of DeletePiece is to deallocate the memory space used by a t_Piece.

Function: t_Move *CreateMove(t_Position from, t_Position to)
Description: Function in ChessObjects.c with two parameters: from and to which are both of type t_Position. The return type is a pointer to a t_Move. The purpose of CreateMove is to allocate the necessary memory space to create a t_Move with the given parameters.

Function: void DeleteMove(t_Move *move)
Description: Function in ChessObjects.c with one parameter: move which is a pointer to a t_Move. There is no return type. The purpose of DeleteMove is to deallocate the memory space used by a t_Move.

Function: t_MoveList *CreateMoveList()
Description: Function in ChessObjects.c with no parameters. The return type is a pointer to a t_MoveList. The purpose of CreateMoveList is to allocate the necessary memory space to create an empty t_MoveList.

Function: void DeleteMoveList(t_MoveList *list)
Description: Function in ChessObjects.c with one parameter: list which is a pointer to a t_MoveList. There is no return type. The purpose of DeleteMove is to deallocate the memory space used by a t_MoveList.

Function: void AppendMove(t_MoveList *list, t_Move *move)
Description: Function in ChessObjects.c with two parameters: list, a pointer to a t_MoveList and move, pointer to a t_Move. There is no return type. The purpose of AppendMove is to append the t_Move to a t_MoveList.

Function: int RecordMove(t_Piece *piece, t_Position from, t_Position to)
Description: Function in LogFile.c with three parameters: piece of type t_Piece, from of type t_Position, and to of type t_Position. The return type is int. The purpose of RecordMove is to add a move to the log file by writing to a file. Zero is returned on success.

Function: int DeleteLog(const char *fileName)
Description: Function in LogFile.c with one parameter: fileName of type const char. The return type is int. The purpose of DeleteLog is to delete the log file. Zero is returned on success.

Function: void AnalyzeCapture(t_Move *move)

Description: Function in Strategy.c with one parameter: move of type t_Move. This function modifies the Points value of a given move depending on the piece it is going to capture.

Function: void AnalyzeFuture(t_Move *move, t_Board board, int loops, t_Player color, t_PieceList pieces)
Description: Function in Strategy.c with five parameters: move of type t_Move, board of type t_Board, loops of type int, color of type t_Player, and pieces of type t_PieceList. This function runs through potential future iterations of the board, assuming enemy moves will be based upon our algorithm.

Function: void DeleteBoard(t_Board board)
Description: Function in BoardInteract.c with one parameter: board of type t_Board. This function deletes pointers within a given board so it can be deleted when it is finished being used.

Function: t_Board CopyBoard(t_Board board)
Description: Function in BoardInteract.c with one parameter: board of type t_Board. This function makes a copy of a given board so that various functions can be used on it without disrupting the original board.

Function: t_Player OppositePlayer(t_Player color)
Description: Function in ChessObjects.c with one parameter: color of type t_Player. This function returns the opposite color type of whatever color was inputted.

Function: t_Position GetPosition(int x, int y, int space)
Description: Function in UserInterface.c with three parameters: x of type int, y of type int, and space of type int. This function returns the position on the board where the user clicked on the screen.

Function: int IsLegalMove(t_Move *move, t_Board board, t_Move *PrevMove, t_PieceList pieces)
Description: Function in BoardInteract.c with four parameters: move of type t_Move, board of type t_Board, PrevMove of type t_Move, and pieces of type t_PieceList. This function returns an int to represent whether or not a move is legal.

Function: t_MoveList *CastlingMove(t_Board board, t_Position position, t_PieceList pieces)
Description: Function in ChessRules.c with three parameters: board of type t_Board, position of type t_Position, and pieces of type t_PieceList. This function returns a list of moves that are valid and involve castling at any given board orientation.

Function: int Stalemate(t_Board board, t_Player color, t_PieceList pieces)
Description: Function in ChessRules.c with three parameters: board of type t_Board, color of type t_Player, and pieces of type t_PieceList. This function returns an int to signify whether or not the game is currently at a stalemate.

Function: void DrawPromotionMenu(SDL_Surface *screen, t_Player Color, SDL_Surface *queen_w, SDL_Surface *rook_w, SDL_Surface *bishop_w, SDL_Surface *knight_w, SDL_Surface *queen_b, SDL_Surface *rook_b, SDL_Surface *bishop_b, SDL_Surface *knight_b)

Description: Function in UserInterface.c with ten parameters: screen of type SDL_Surface, Color of type t_Player, queen_w of type SDL_Surface, rook_w of type SDL_Surface, bishop_w of type SDL_Surface, knight_w of type SDL_Surface, queen_b of type SDL_Surface, rook_b of type SDL_Surface, bishop_b of type SDL_Surface, and knight_b of type SDL_Surface. This function has no return value and is used to draw a menu to determine what the user desires to be promoted.

Function: int ValidPositionClickedPromotion(int x, int y, int space)
Description: Function in UserInterface.c with 3 parameters: x of type int, y of type int, and space of type int. This function returns an int to signify whether or not the correct part of the screen was clicked on.

Function: void DrawGamemodeMenu(SDL_Surface *screen)
Description: Function in UserInterface.c with 1 parameter: SDL_Surface *screen. This Function first draws out the three game modes we give to the user AI vs AI, Human vs AI, and Human vs Human.

Function: int HandleChooseGameMenu(SDL_Surface *screen)
Description:Function in UserInterface.c with 1 parameter: SDL_Surface *scree. This function gets the input from the user to choose game modes from AI vs AI, Human vs AI, and Human vs Human.

Function: int HandlePlayerMenu(SDL_Surface *screen)
Description:Function in UserInterface.c with 1 parameter: SDL_Surface *scree. This function gets the input from the user to choose what color do they want to play as white or black.

Function: int HandlePlayerMenu(SDL_Surface *screen)
Description:Function in UserInterface.c with 1 parameter: SDL_Surface *scree. This function gets the input from the user to choose what color do they want to play as white or black.

Function: t_PieceType PromotionType(int x, int y, int space)
Description: Function in UserInterface.c with 3 parameters: x of type int, y of type int, and space of type int. This function returns a Piece Type to signify what is going to be promoted.

Function: SDL_Surface *LoadPieceBMP(const char *name)
Description: Function in UserInterface.c with one parameter: name of type const char. This function returns a SDL image depending on the piece that is being used.

Function: void DrawPlayerMenu(SDL_Surface *screen)
Description: Function in UserInterface.c with parameter: screen of type SDL_Surface. This function has no return type but it draws the menu on the screen to determine which player will be selected.

Function: int PlayerChoice(SDL_Surface *screen, int x)
Description: Function in UserInterface.c with two parameters: screen of type SDL_Surface and x of type int. This function returns an int to signify what player choice the user made.

Function: int HandleChoosePlayer(SDL_Surface *screen)

Description: Function in UserInterface.c with parameter: screen of type SDL_Surface. This function returns in an int to signify what is occurring.

Function: int HandleChooseGamemode(SDL_Surface *screen)
Description: Function in UserInterface.c with parameter: screen of type SDL_Surface. This function returns an int to signify what game mode was chosen.

Function: int ValidPositionClicked(int x, int y, int space)
Description: Function in UserInterface.c with three parameters: x of type int, y of type int, and space of type int. This function returns an int to signify whether the position clicked was valid

Function: void RunComputerComputerMode(t_Board board, SDL_Surface *screen, t_PieceList pieces, SDL_Surface *pawn_w, SDL_Surface *king_w, SDL_Surface *queen_w, SDL_Surface *rook_w, SDL_Surface *bishop_w, SDL_Surface *knight_w, SDL_Surface *pawn_b, SDL_Surface *king_b, SDL_Surface *queen_b, SDL_Surface *rook_b, SDL_Surface *bishop_b, SDL_Surface *knight_b)
Description: Function in UserInterface.c with parameter 15 : t_Board board, SDL_Surface *screen, t_PieceList pieces, SDL_Surface *pawn_w, SDL_Surface *king_w, SDL_Surface *queen_w, SDL_Surface *rook_w, SDL_Surface *bishop_w, SDL_Surface *knight_w, SDL_Surface *pawn_b, SDL_Surface *king_b, SDL_Surface *queen_b, SDL_Surface *rook_b, SDL_Surface *bishop_b, and SDL_Surface *knight_b. This function runs the AI to go against itself so both players black and white will be AI playing chess against each other.

Function: void RunHumanComputerMode(t_Board board, SDL_Surface *screen, t_PieceList pieces, SDL_Surface *pawn_w, SDL_Surface *king_w, SDL_Surface *queen_w, SDL_Surface *rook_w, SDL_Surface *bishop_w, SDL_Surface *knight_w, SDL_Surface *pawn_b, SDL_Surface *king_b, SDL_Surface *queen_b, SDL_Surface *rook_b, SDL_Surface *bishop_b, SDL_Surface *knight_b)
Description: Function in UserInterface.c with parameter 15 : t_Board board, SDL_Surface *screen, t_PieceList pieces, SDL_Surface *pawn_w, SDL_Surface *king_w, SDL_Surface *queen_w, SDL_Surface *rook_w, SDL_Surface *bishop_w, SDL_Surface *knight_w, SDL_Surface *pawn_b, SDL_Surface *king_b, SDL_Surface *queen_b, SDL_Surface *rook_b, SDL_Surface *bishop_b, and SDL_Surface *knight_b. This function runs Human vs computer where the use playing the color they choose against a AI in chess.

Function: void RunHumanHumanMode(t_Board board, SDL_Surface *screen, t_PieceList pieces, SDL_Surface *pawn_w, SDL_Surface *king_w, SDL_Surface *queen_w, SDL_Surface *rook_w, SDL_Surface *bishop_w, SDL_Surface *knight_w, SDL_Surface *pawn_b, SDL_Surface *king_b, SDL_Surface *queen_b, SDL_Surface *rook_b, SDL_Surface *bishop_b, SDL_Surface *knight_b)
Description: Function in UserInterface.c with fifteen parameters: screen of type SDL_Surface, pieces of type t_PieceList, pawn_w of type SDL_Surface, king_w of type SDL_Surface, queen_w of type SDL_Surface, rook_w of type SDL_Surface, bishop_w of type SDL_Surface, knight_w of type SDL_Surface, pawn_b of type SDL_Surface, king_b of type SDL_Surface, queen_b of type SDL_Surface, rook_b of type SDL_Surface, bishop_b of type SDL_Surface, and knight_b of type SDL_Surface. This function has no return type but it is used to initiate the game in the human vs human setting.

3.3 Detailed Description of Input and Output Formats

- Syntax/format of a move input by the user
    - User inputs position of piece to be moved, then the new position of the piece to be moved
        - e.g. “e2e4”
- Syntax/format of a move recorded in the log file
    - Standard chess algebraic notation

4 Development Plan and Timeline

4.1 Partitioning of Tasks

- main program
- user interface (textual and/or graphics)
- chess objects (data structures for pieces, boards, moves)
- lists (or trees) of moves
- chess rules (possible moves, legal moves)
- log file module
- strategy (artificial intelligence, AI) module
- documentation
- Testing
- Makefile
- BoardInteract.c functions

4.2 Team Member Responsibilities

4/15/2023:

- ChessObjects.c (Christopher)

4/21/2023:

- ChessRules.c
  - for a piece on the board, compute all reachable positions (Miguel)
  - for a piece on the board, compute all legal moves (Miguel)
  - on a board, check whether or not a player's king is in check (Brandon)
  - on a board, check whether or not a player's king is in checkmate (Brandon)
  - for a player and a given board, compute all legal moves (Miguel)
- BoardInteract.c
  - on a board, lookup a piece at a given position (basically finding the piece) (Ryan)
  - on a board, put a given piece onto a given position (Ryan)
  - on a board, move a piece from a position to another (Ryan)
- LogFile.c (Christopher)
- UserInterface.c (Christopher)
  - display board (graphic/textual)
- Strategy.c (team)
- ChessCooker.c main function (Christopher)
- Makefile (Brandon)

4/22/2023:

- Testing (team)

4/30/2023:

  - Final Product

Ongoing:

- Documentation (team)

References

SDL Library: https://www.libsdl.org/release/SDL-1.2.15/docs/html/index.html

White king: https://creazilla.com/nodes/3161010-chess-clipart
White rook: https://creazilla.com/nodes/3161016-chess-clipart
White queen: https://creazilla.com/nodes/3161014-chess-clipart
White knight: https://creazilla.com/nodes/3174302-chess-clipart
White pawn: https://creazilla.com/nodes/3161013-chess-clipart
White bishop: https://creazilla.com/nodes/3174300-chess-clipart

Black pawn: https://creazilla.com/nodes/3161015-chess-clipart
Black knight: https://creazilla.com/nodes/3174301-chess-clipart
Black king: https://creazilla.com/nodes/3161005-chess-clipart
Black queen: https://creazilla.com/nodes/3161012-chess-clipart
Black rook: https://creazilla.com/nodes/3161011-chess-clipart
Black bishop: https://creazilla.com/nodes/3174299-chess-clipart

Index