Queen Chess Software Specification

Final Software Release:Version 1.0

Team 18 - The Queen
Bryan Nguyen
Luke Bugbee
Nathan Twigg
Yedarm Park
Josh Cruz
UC Irvine - EECS 22L - Spring 2020

Table of Contents

Glossary:

Application Procedural Interface (API)
An procedural interface that provides access to the simulation data and conveys all physical and nonphysical information used by the application

Array
A data structure, which can store a fixed-size collection of elements of the same data type

Control Flow
The order in which functions are called or statements are executed throughout the program based on sequence or conditions

Data Structure
The organization, management, and storage format that enables efficient access and modification

Data Types
Attribute of data which tells the compiler how the programmer intends to use the data. The most common data types being: int, floating points, char, bool

Directory
An organizational unit, or container, used to organize folders and files

Enum
A data type consisting of a set of named values called elements, members, enumeral, or enumerators of the type

Function
A block of organized, reusable code that is used to perform a single, related action

Graphical User Interface (GUI)
A form of interface that allows the user to communicate with the computer through graphical images and icons that build on the typical text-based interfaces

Hardware
Physical device used in or with a machine

Interface
Program enabling a user to communicate with the computer

Module
A software component or part of a program that contains one or more routines. One or more independently developed modules make up a program

Software
Collection of code, programs, and other operating information used by a computer

Struct
A composite data type declaration that defines a physically grouped list of variables under one name in a block of memory, allowing the different variables to be accessed via a single pointer or by the struct declared name which returns the same address

Parameters
A special type of variable that serves as an argument used to pass information between functions or procedures

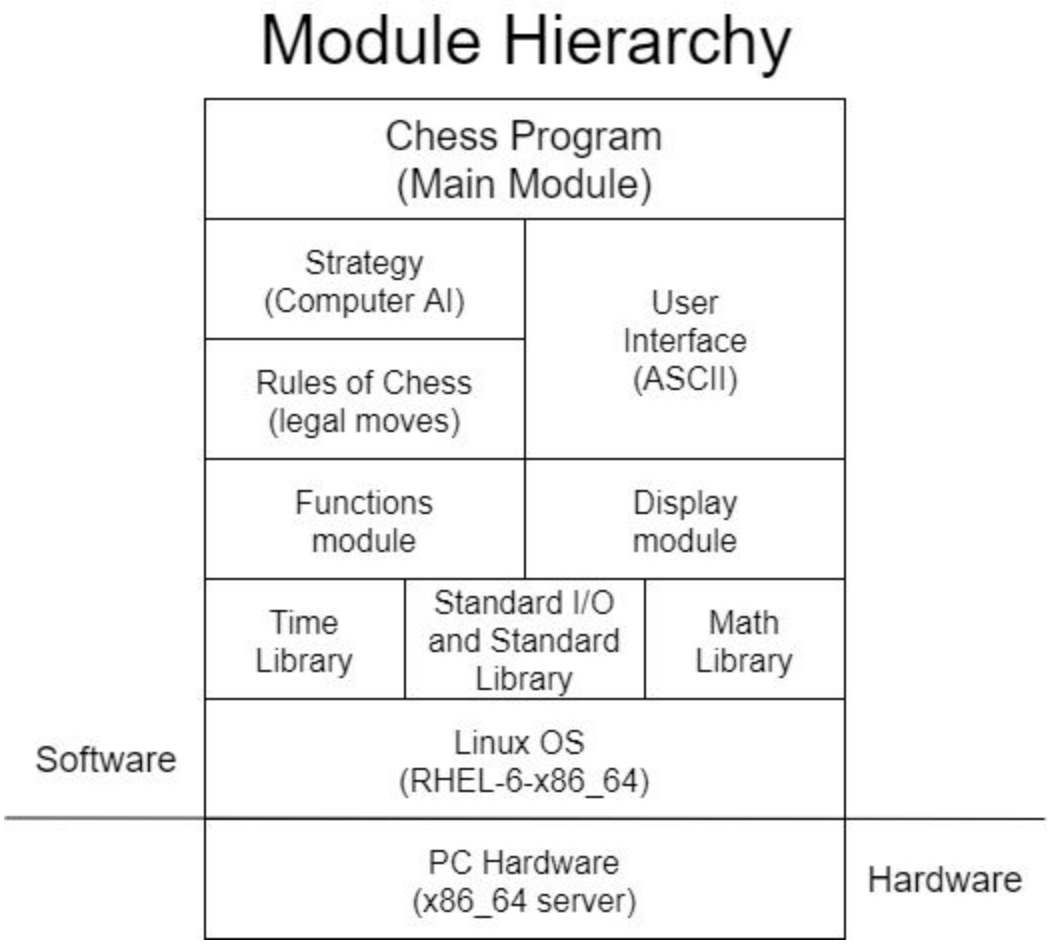
Chapter 1: Software Architecture Overview

1.1 Main data types and structures

We will represent the main game board through a 2D integer array in which its values correspond to an empty space or a specific piece. Furthermore, negative and positive values will be used to represent each piece to determine if it is a white piece (negative) or a black piece (positive).

1.2 Major software components

A basic diagram of the overall module hierarchy is outlined below. Modules and libraries may be added or removed as the development process continues. An example of such change could be the implementation of a GUI (graphical user interface) which would replace the ASCII user interface. This would also invoke either the SDL or GTK library.



1.3 Module interfaces

Application Procedural Interface (API) of modules (source and header files)

Main.c // where the main game control flow will take place, including user input

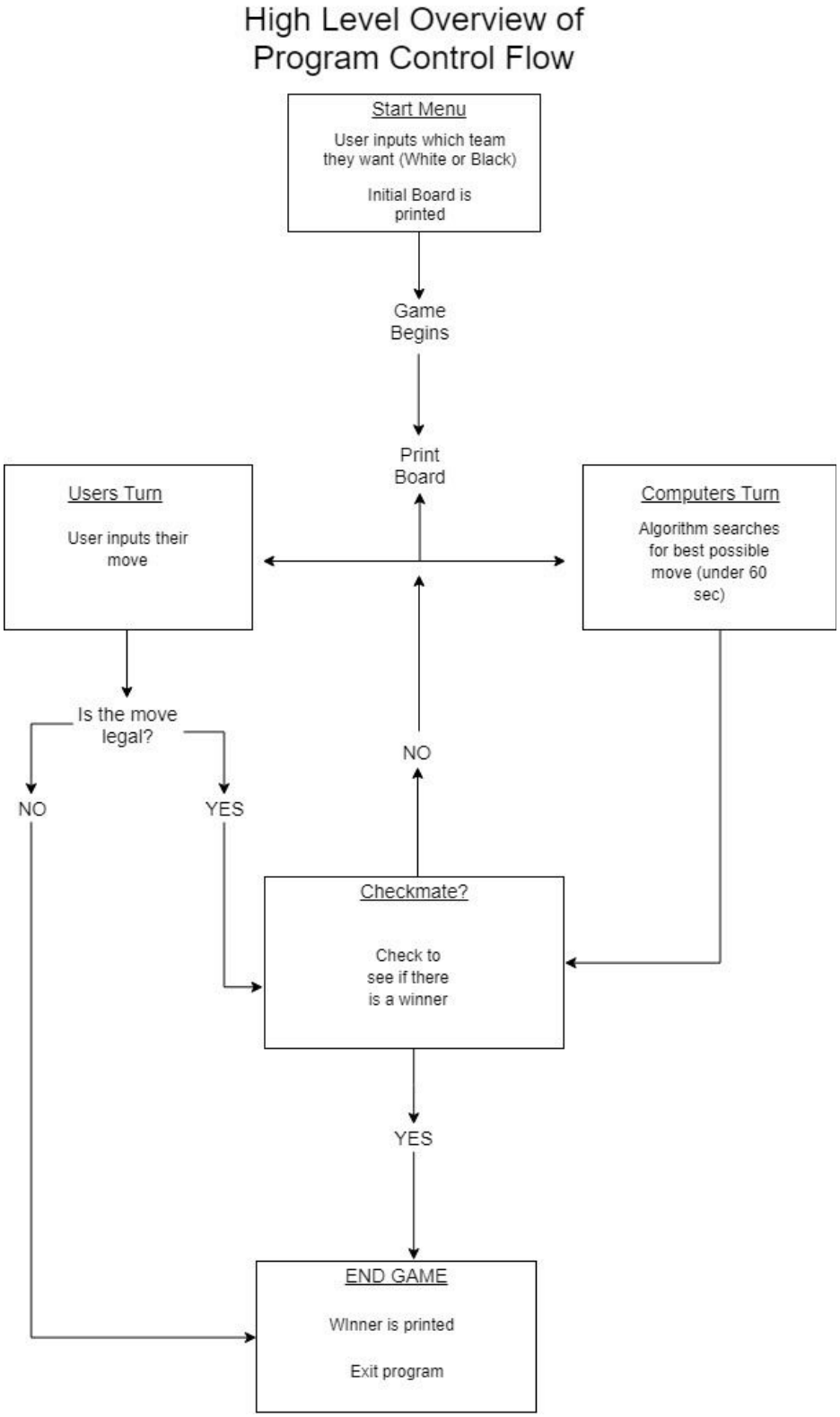
Functions.h / Functions.c // where the chess functions are located

Display.h / Display.c // where the user interface is located

Definitions.h // where the data structures definitions are located

AI.h / **AI.c**// where AI will be located

1.4 Overall program control flow



Chapter 2: Installation

2.1 System requirements, compatibility

Operating System	Linux or Windows with access to a Linux environment
CPU	Intel Core 2 (Intel Core 5 recommended)
Memory	50 MB RAM (100MB recommended)
Hard Disk Space	10 MB Available
Media	CD-ROM, 2x or higher

2.2 Setup and configuration

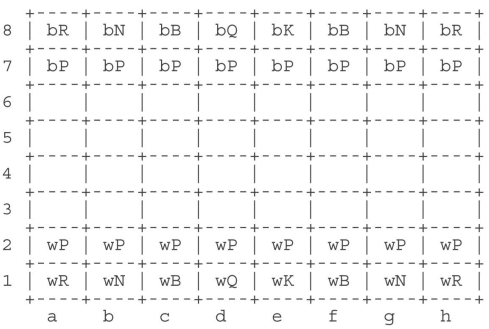
1. Type “git clone https://github.uci.edu/20SEECS22L/Team18.git” into the Linux command line.

2.3 Building, compilation, installation

1. In the Linux Command line, type “tar -xvzf Chess_V1.0_src.tar.gz”
2. Then type “make” to compile the program.
3. Then type “make test” and enjoy the game!

Chapter 3: Documentation of packages, modules, interface

3.1 Detailed description of data structures



We are representing our board in a 2D array: int board[8][8]
The piece type will be represented as an integer in the 2D array:
Empty = 0
Pawn = 1, 10 *(only if it moves two spaces initially)
Rook = 2, 7 *(left side of the board initially), 17 *(right side of the board initially)
Knight = 3
Bishop = 4
Queen = 5
King = 6, 8 *(initial value)
White = negative values | Black = positive values

3.2 Detailed Description of Functions

Functions.h:

void setBoard(int board[8][8]);

- Function fills the array with the “pieces” on a chess board with its corresponding number. It is called at the beginning of every game.

int getCheckMate(int board[8][8], char team);

- The function first checks what team the player is on then it loops through each spot on the board to find the position of the king and then checks the surrounding spaces to see if the king is in Checkmate.

int checkPos(int board[8][8], char team, int x, int y);

- This function assists the “getCheckMate” function by seeing what pieces can attack the spaces surrounding the king.

void Move(int board[8][8], int x, int y);

- Function takes in the board and user input. It converts the user inputs to a specific location and piece on the board and determines which type of functions should be used.

void PawnMove(int board[8][8], int piece, int a, int b, int c, int d);

- If the piece chosen in Move(...) is (+/- 1 or +/-10) it goes to PawnMove which checks if the final position inputted by the user is a valid type of pawn move. If the move is valid, the function updates the board with the piece in the final location. The function includes special ability of the initial move but Promotion and En Passant is a separate function called by PawnMove if certain conditions are satisfied.

void RookMove(int board[8][8], int piece, int a, int b, int c, int d);

- If the piece chosen in Move(...) is (+/- 2 or +/-7 or +/- 17) it goes to RookMove which checks if the final position inputted by the user is a valid type of rook move. If the move is valid, the function updates the board with the piece in the final location. The function includes the special ability of castling by calling Castling(...) if certain conditions are satisfied.

void KingMove(int board[8][8], int piece, int a, int b, int c, int d);

- If the piece chosen in Move(...) is (+/-6 or +/-8) it goes to KingMove which checks if the final position inputted by the user is a valid type of king move. If the move is valid, the function updates the board with the piece in the final location. The function includes the special ability of castling by calling Castling(...) if certain conditions are satisfied.

void BishopMove(int board[8][8], int piece, int a, int b, int c, int d);

- If the piece chosen in Move(...) is (+/-4 or +/-7) it goes to BishopMove which checks if the final position inputted by the user is a valid type of Bishop movement. If the move is valid, the function updates the board with the pieve in the final location.

void KnightMove(int board[8][8], int piece, int a, int b, int c, int d);

- If the piece chosen in Move(...) is (+/-6 or +/-8) it goes to KnightMove which checks if the final position inputted by the user is a valid type of knight move. If the move is valid, the function updates the board with the piece in the final location.

void Castling(int board[8][8], int piece, int a, int b, int c, int d);

- When neither the king or the rook have moved, and the spaces between those pieces are zero, the player is allowed to “castle” which changes the position of those two pieces.

void Promotion(int board[8][8], int piece, int a, int b, int c, int d);

- When a player reaches the opposing players side with their pawn, the user is allowed to change that pawn into a rook, queen, knight, or bishop by user input

void EnPassant(int board[8][8], int piece, int a, int b, int c, int d);

- Function that allows the “En Passant” rule in chess.

int BlackCheck(int board[8][8]);

- Function that checks if the Black team is in check by looping through each spot on the board and seeing if the piece on that spot is putting the opposing king in check.

int WhiteCheck(int board[8][8]);

- Function that checks if the White team is in check by looping through each spot on the board and seeing if the piece on that spot is putting the opposing king in check.

Display.h

int pos(char p[3]);

- Function that converts the users move input into an integer to easily access the board

void printBoard(int board[8][8]);

- Function that takes the current state of the board and prints it to the command line

void LogFile(const char filename[], int board[8][8], char init[3], char fin[3], int flag);

- Function that records all moves made during the game and writes it to an output text file

3.3 Detailed Description of input and output formats

Format of a move input by the use:

- The user will first type in the location of the piece they want to move
- The user then types in what type of piece they are moving
- The user then types in the location they want to move to

Format of a move recorded in the log file:

- The user will first name the text file before the game starts
- The program will store the inputs of locations and pieces

- The program prints out the initial location, the type of piece moved, and the location the piece moved on the screen for the user to see as well as write to an ongoing text file
- At the end of the game, the program saves and outputs a text file of all the moves made during the game

Chapter 4: Development plan and timeline

4.1 Partitioning of tasks

Nathan- Functions
Bryan- Functions
Luke- AI Algorithm
Yedarm- Creating structs or board and pieces
Josh- User Interface, Log file of moves

4.2 Team member responsibilities

Bryan- Manager
Nathan- Reflector
Luke- Reflector
Yedarm- Presenter
Josh- Recorder

Terms of use

References:

Christensson, Per. "System Requirements Definition." TechTerms. Sharpened Productions, 30 January 2020. Web. 08 April 2020. <https://techterms.com/definition/system_requirements>.

EECS22L, Spring 2020, University of California Irvine, Project 1 - Software Architecture Specification Slides 4.1 “A Simple User Interface”.

Bitboards. (n.d.). Retrieved from https://www.chessprogramming.org/Bitboards

Index: