****

**Chessnut**© **Software Specification**

**Version 1.0**

By: Over Compens8 © (Team 8)

**Authors:**

Kevin Flaieh

Bibek Adhikari

Jose Fregoso

Nicholas (Grant) Carson

Andrew Dertli

**Affiliation**: University of California, Irvine

**Table of Contents**

**Glossary Page No.**

1. **Software Architecture Overview**  **………………………………………………..4-5**

1.1 Main data types and structures …………………………………………………..4

1.2 Major software components …………………………………………….. ……….4

• Diagram of module hierarchy ………………………………………………..5

1.3 Module interfaces …………………………………………………………………..5

• API of major module functions ……………………………………………....5

1.4 Overall program control flow……………………………………………………....5

**2**. **Installation ……………………………………………………………………………..6**

2.1 System requirements, compatibility ……………………………………………..6

2.2 Setup and configuration …………………………………………………………..6

2.3 Building, compilation, installation………………………………………………...6

**3. Documentation of packages, modules, interfaces……………………………7-8**

3.1 Detailed description of data structures …………………………………………..7

• Critical snippets of source code ……………………………………………..7

3.2 Detailed description of functions and parameters ……………………………7-8

• Function prototypes and brief explanation ………………………………....8

3.3 Detailed description of input and output formats ……………………………….8

• Syntax/format of a move input by the user ………………………………...8

• Syntax/format of a move recorded in the log file…………………………...8

**4. Development plan and timeline ……………………………………………………9**

4.1 Partitioning of tasks ……………………………………………………………….9

4.2 Team member responsibilities ………………………………………………….10

**5. Copyright……………………………………………………………………………...11**

**6. References……………………………………………………………………………12**

**7. Index…………………………………………………………………………………...13**

**Glossary**

**A**

**AI:** This refers to Artificial Intelligence, which is the ability of the computer to play a game by itself

**B**

**Bishop:** moves diagonally across the board.

**C**

**Castling:** is a special move in the game of chess involving the king and either of the original rooks of the same color. It is the only move in chess (except promotion) in which a player moves two pieces at the same time. Castling consists of moving the king two squares towards a rook on the player’s first rank (row), then moving the rook onto the square over which the king crossed. Castling can only be done if the king has never moved, the rook involved has never moved, the squares between the king and the rook involved are not occupied, the king is not in check, and the king does not cross over or end on a square in which it would be in check.

**E**

**En passant**: is a special pawn capture which can occur immediately after a player moves a pawn two squares forward from its starting position, and an enemy pawn could have captured it had the same pawn moved only one square forward. The opponent captures the just-moved pawn as if taking it "as it passes" through the first square. The resulting position is the same as if the pawn had moved only one square forward and the enemy pawn had captured normally. The en passant capture must be done on the very next turn, or the right to do so is lost. Such a move is the only occasion in chess in which a piece captures but does not move to the square of the captured piece. If an en passant capture is the only legal move available, it must be made.

**G**

**GUI:** This will create the user interface for the program so a game can be played.

**K**

**King:** can move in any direction, but only one step at a time. Also, the king must never move into check. There is also a special "castling" move for the king.

**Knight:** jump to eight different squares which are two steps forward plus one step sideways from its current position.

**M**

**Main file:** It calls all functions in order to start game

**P**

**Pawn:** move only forward towards the end of the board, but captures sideways. From its initial position, a pawn may make two steps, otherwise only a single step at a time. If the pawn reaches the end of the board, it is automatically promoted to another piece (usually a queen). There is also a special "en passant" move for the pawn.

**Q**

**Queen:** move diagonally across the board.

**R**

**Rules file:** Controls all movements of all pieces on game.

**Rook:** move horizontally and vertically across the board

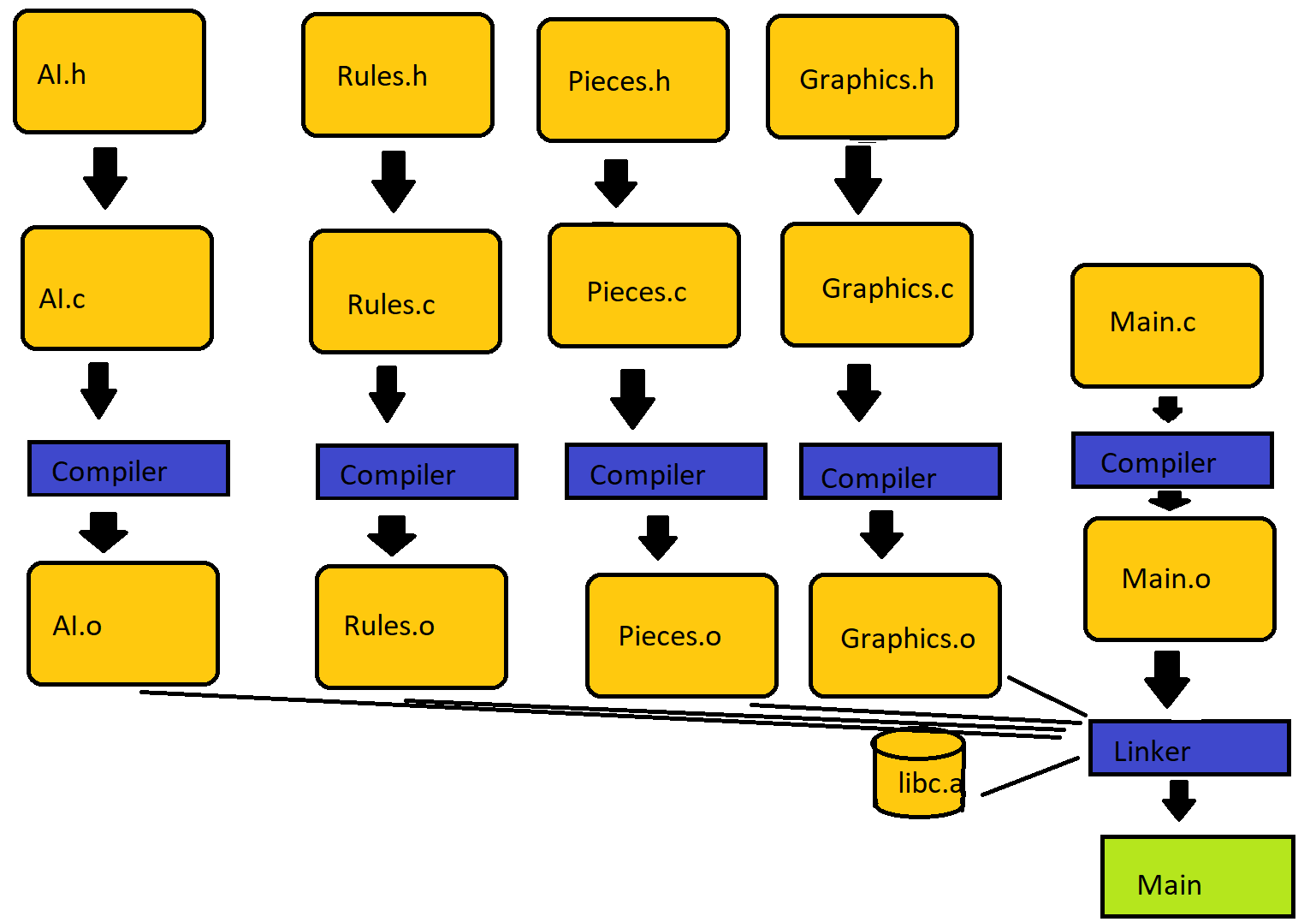
1. **Software Architecture Overview**

**1.1 Main data types and structures**

**Figure 1**

|  |  |
| --- | --- |
| **Element** | **Data Structure Type** |
| Pieces | Structures |
| Board | 2D Array |
| Movements Log | Double Linked List |
| Legal Movements | 1D Array |

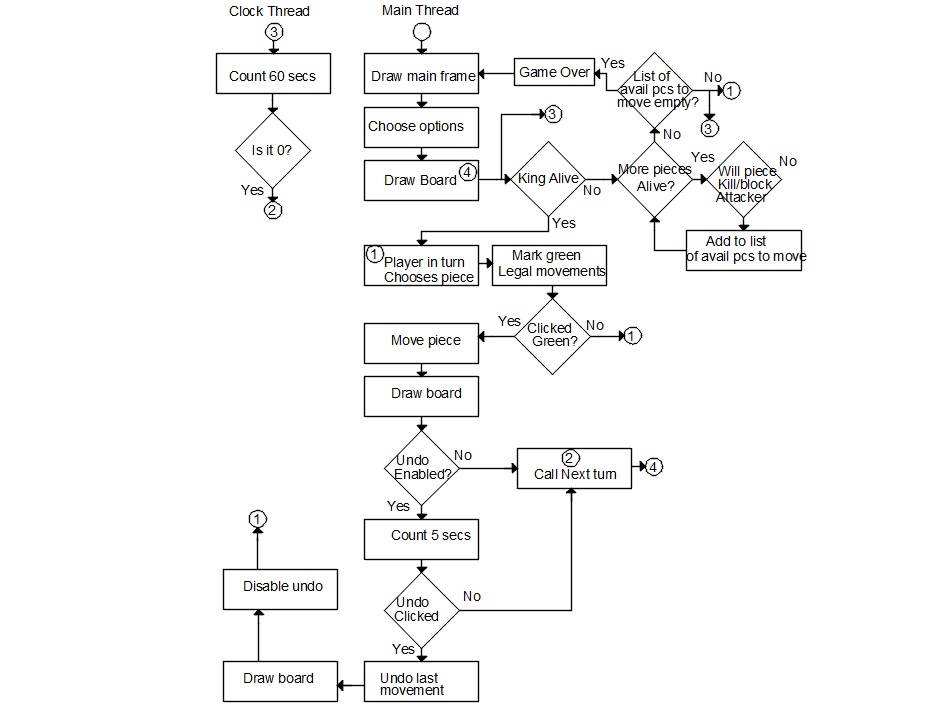
**1.2 Major software components**



**FIgure 2**

**1.3 Module interfaces**

* ***Rules****:* Contains the functions of each piece type. Tells the user and the computer player what are legal moves.
* ***Pieces***: Contains the structs of each piece type telling us position, color, etc.
* ***Graphics*:** Contains all of the graphics we are using for the GUI. Ex: what the board looks like, piece models, etc.
* ***Main*:** draws title screen then board and log. Builds initial 2D array for board. Builds linked list for Log. Moves pieces. Calls other files

**1.4 Overall program control flow**

**FIgure 3:**

**2. Installation**

**2.1** System requirements, compatibility

* **Operating System:** Linux
* **Disk Space:** 20 MB free.
* **RAM:** 512 MB or more highly recommended.
* **CPU:** Single core Intel processor or better (32-bits)

**2.2 Setup and configuration**

To install this software, copy the Chess\_V1.0.tar.gz and Chess\_V1.0\_src.tar.gz file from the host server to your personal Linux server using the *~cp* command.

**2.3 Building, compilation, installation**

To uninstall this software, type:

*rm Chess\_V1.0.tar.gz Chess\_V1.0\_src.tar.gz*

into the command line from the directory where those files are installed.

**3. Documentation of packages, modules, interfaces**

**3.1. Detailed description of data structures**

* Each piece will be a struct containing enums, ints, and chars.
* The board will be comprised of a 2D array with indexes of 1-8 and 1-8 (using and ascii converter). Within the array will be the pieces.
* The log will be a double linked list containing x and y locations of the previous positions the moved piece, its new moved location, and the piece type. We will use an ascii converter for the x values.
* Legal moves will be a single linked list and it will be the return type of the functions of piece types (located in rules).

**3.2. Detailed description of functions and parameters**

Int array[7] Straight(PIECE)

{

}

Int array[7] Diagonal(PIECE)

{

}

* Determines all of the legal moves in diagonal or straight directions from a piece. Used to simplify Queen, Rook, and Bishop functions.

* The function for every piece are going to return all of the legal moves a piece has. The most legal moves is a queen in the corner which is 21.

Int array[21] Pawn(enum color, char locx, int locy)

{

}

Int array[21] Bishop(enum color, char locx, int locy)

{

}

...etc

* Each function will check the surroundings of the piece and determine its legal moves.

Int array[4] Move(PIECE, MouseX, MouseY)

{

Int record[4];

if(PIECE->type == K)

{

PIECE->locx

PIECE->locy

}

Return record;

}

* Move will move a piece from where it is to a legal place the user clicks. Will kill pieces if that is one of the moves chosen. Move returns what will be recorded on the log

**3.3. Detailed description of input and output formats**

* The user inputs will be mouse click locations. They will click on a piece and they will be given legal move locations. If they click on a legal move the piece will be moved to that location.
* The log will be a double linked list containing x and y locations of the previous positions the moved piece, its new moved location, and the piece type.

**4. Development plan and timeline**

**4.1 Partitioning of tasks**

We decided to partitioning the project as follows, please look below Figure 5 for information regarding each task:

**FIgure 5**

|  |  |
| --- | --- |
| **Partition Name** | **N. of People** |
| Structure&Function development | 3 |
| GUI and main program file development | 2 |
| Artificial Intelligence development | All |

**Structure&Function development:**

Pieces.c & Pieces.h: This Pieces.c contains the function to create white pieces and black pieces.

Rules.c & Rules.h: This Rules.c contains the function for each pieces describing how each pieces move.

void Queen(PIECE \*click, int\* boardP);

void Bishop(PIECE \*click, int\* boardP);

void Rook(PIECE \*click, int\* boardP, PIECE \*whitePieces, PIECE \*blackPieces);

void Knight(PIECE \*click, int\* boardP);

void King(PIECE \*click, int\* boardP,PIECE \*whitePieces, PIECE \*blackPieces);

void Pawn(PIECE \*click, int\* boardP,PIECE\* whitePieces, PIECE\* blackPieces);

**GUI development:**

GUI.c and GUI.h: This GUI.c contains all the necessary function needed to build the GUI for chess game. For example:

void SettingsButtons(SDL\_Surface \*screen, SDL\_Surface\* image, Users\* users,SDL\_Event event);

void GetBoardCoordinates(SDL\_Event event,int\* dest);

**Main program file:**

Main.c contains the collection and execution of all functions developed by all programmers. It takes care of opening the graphic interface for the user and makes sure that rules functions are called for execution of the game.

**Artificial Intelligence development:**

AI.c & AI.h: This AI.c contains randomly generated number for pieces to move, and the levels that user can play (beginner,intermediate, and expert).

**4.2 Team member responsibilities**

**FIgure 6**

|  |  |
| --- | --- |
| **Member Name** | **Responsibility** |
| Kevin Flaieh | Structure&Function development:  In charge of development of the following files: Rules,AI,CVS,Log |
| Bibek Adhikari | In charge of development of the following files:  GUI, main program, makefile development |
| Jose Fregoso | In charge of development of the following files:  GUI, Pieces, main program, makefile & packaging of program |
| Nicholas (Grant) Carson | Structure&Function development: Rules of game, main program |
| Andrew Dertili | Structure&Function development : AI,Pieces Struct |

**5. Copyright**

* Overcompens8 © 2018
* This piece of Software is the sole property of Overcompens8 ©. You, the end user, have been given permission to use the software.

**6. References**

* **All GUI stuff (will know after Tuesday Lecture)**
* **Pieces Pictures**
* **Board Pictures**
* **AI Algorithm**

**7. Index**

Bishop,3

Board,3,4,5,7,11,

Castling,3

Check,3,8,

Documentation,3,7,

En Passant,3

Installation,6

King,3

Knight,3,

Pawn,3,8

Queen,3,7,8,

Rook,3,7,