Chess Olympian v 0.0

Created by

Milk and Dim Sum

Table of Contents

Glossary

AI - computer player

API - application programming interface

Board - where the game pieces go.

Coords - coordinates on the board

Enum - enumerator data type used in C programming

Pieces - each piece of the game

AI - computer player

Makefile - the script that builds and compiles the program

Module - a component of the program

Node - structure used by AI

Structures - a data type that contains other date types

1. Software Architecture Overview

1.1 Main Data Types and Structures

Data Types

- int cmd

  Stands for "command," and will be used in main.c to determine whether or not to stay in the loop that processes every step of the game.

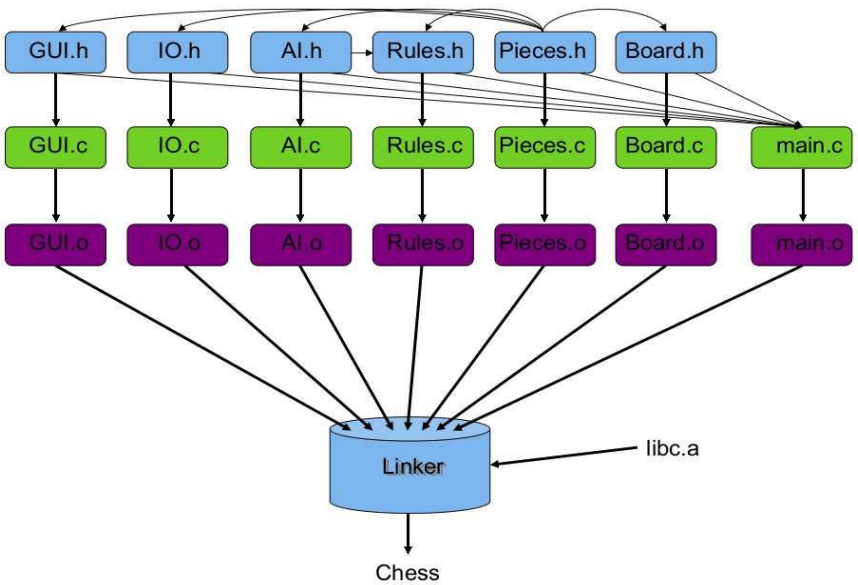
Structures

- typedef struct t_piece
- typedef struct t_coords
- typedef struct s_Node
- typedef enum { White, Black } t_player
- typedef enum { Pawn, Rook, Knight,
  Bishop, Queen, King } t_piecetype

1.2 Major software components

main.c Pieces.h Pieces.c IO.h IO.c Board.h Board.c

Rules.h Rules.c AI.h AI.c GUI.hGUI.c



1.3 Module interfaces

All of the board data will be stored in a 2D array of pointers of type struct Piece. The 2D array will be a 8x8 matrix and will be declared in main.c. In this fashion, any setup or manipulation to the game board can be done in main.c by using functions of included modules.

The AI (artificial intelligence) module will interact largely with Rules module for its dependency.

All modules will include the Pieces module so that all structure and data types passed between functions will be uniform.

Some functions for modules:

IO

```
int decodeMove (char* move, t_coords *from, t_coords*to);
void printMainMenu();
```

Rules

```
int checkLoc (t_piece *GameBoard[8][8], t_coords coords, int turn);
void genLegal(t_coords coords, t_piece *GameBoard[8][8], int legal[8][8]);
```

Board

```
void initBoard(t_piece *board[8][8]);
void copyBoard(t_piece *oldBoard[8][8], t_piece *newBoard[8][8]);
```

GUI

```
void displayScreen(s_board* current);
```

1.4 Overall program control flow

Once the program is executed, the main function will remain in a loop to detect events from the screen or from the console depending on the version of chess. In this loop, main.c will access board setup functions in Board.h such as the function initBoard(), which will allocate memory for all pieces on both sides as well as initialize the locations of all the pieces. When a player moves, main.c will access Rules.h, which in turn is dependent on Pieces.h. Functions that will be accessed in Rules will include checkLoc(), which will check the location a player has chosen for a pieces and check if it is legal. The function will then run through the possible moves of the pieces, which is in the genLegal() function, which

has a bit array of legal moves for any selected piece. Pieces.h be used by the Rules.h file, which will have to initialize pieces (initPiece()) and check for the piece type (getPiecesType()). When the computer moves, main.c will access AI.h which is heavily dependent on Rules.h and also dependent on Pieces.h. The game will continue to cycle through each opponent until the game ends. The user has the option of playing again or exiting the program.

2. Installation

2.1 System Requirements

- PC x86 with Linux OS
- Monitor, Keyboard, Mouse
- Data input methods: Floppy 3.5 / CD-ROM/ USB / Internet

2.2 Setup and Configuration

- Create all module files as described in section 1
- Put all sources files in src directory
- Create Makefile in root directory of project
- All object files should be compiled to obj directory
- Chess executable should be compiled to bin directory

2.3 Building Compilation and Installation

- Build:
    - `make all`
- Launch program with:
    - Go to installed directory
    - `bin/Chess`
        - For certain linux distributions, use
            - `./bin/Chess`

Software Specification: Section 3

3.1 – Data Structures

Piece

This struct contains three members, all of which are enumerated types. “Points” serves as both a piece type identification, and second as a value for use by the AI. “Team” and “Human” indicate what team the piece belongs to, and whether the piece is controlled by a human or an AI.

Source code:

```
typedef struct {
     t_player team;
     t_piecetype points;
} t_piece;
```

Coords

Coords contains coordinates for a piece. It will store an x and y location for easy access of coordinate locations.

Source code:

```
typedef struct {
  int locY;
  int locX;
}t_coords;
```

Node

Node is a structure used by the AI when creating a moves tree used to determine the next move. The four directional members determine the branch's position in the tree, and are Node pointers. There is a member that represents the value of the move, a member that represents the probability of the move, and a s_board member used to store the state of the board at that move.

Source code:

```
typedef struct {
        t_player turn;
        int pointTotal;
        int pointsOfMove;
        t_piece *board[8][8];
        s_Node *up, *down, *prev, *next;
} s_Node
```

3.2 Function Prototypes and Explanation

int decodeMove (char* move, t_coords *from, t_coords*to);

Takes in the input string from the user and parses it into a Coords structure for easy access later.

int checkLoc (t_piece *GameBoard[8][8], t_coords coords, int turn);

Used to check the validity of an inputed move by first checking the coordinates of the piece the user wants to move, then checking that the intended move is valid. This is done by checking the bitArray produced in the function below against the destination.

void genLegal(t_coords coords, t_piece *GameBoard[8][8], int legal[8][8]);

genLegal produces an array of equal size to the board. It is a logical array, in which a 0 represents an illegal move and 1 represents a legal move. It does this by checking the type of piece via the coordinates.

void printMainMenu();

Used to print the startup menu.

Void teamSwitch(s_board* current);

This is the last function passed in a turn. All it does is switch the CurrTeam type to the opposite team.

void movePiece(t_piece *GameBoard[8][8], t_coords coords1,t_coords coords2);

The movePiece function is primarily used to change the board array to reflect a move that a player makes. It also, however, is where special moves are considered and accounted for. Pieces are deleted if necessary from the board.

void initPiece(t_piece *p, t_player tm, t_piecetype pnt);

In its normal usage, initPieces will simply fill the board array with the pieces in their normal setup. However, it also has the capability to perform special setups per the user's request.

3.3

For the user to input the moves that he wants to make he will simply type the location of the piece that he wants to move followed by the location that he wants to move that piece to. The user will press enter when he is ready to submit his move. For example: b2 e5

The log file will contain a list of moves made by each player, one move on each line. The format will be as follows: br f3 f7. This is to be read as black rook at position f3 moved to f7. To signify that a piece was captured the log file would contain the following text: br f3 xwk f7. Meaning that a black rook at f3 captured a white knight at f7.

Team tasks and responsibilities

4.1 Partitioning of Tasks

GUI: Bruce,

Rules: Charlie, Jason

AI: Jesse, Carey, Kevin

Moves: Adam, Hunter

4.2 Team member responsibilities

Responsibilities have been divided into four subgroups, GUI, Rules, AI, and Moves. The moves group allows pieces to be moved by allowing the user to select a piece and a destination square, in which the move will be outputted to the screen. The board will be created by the GUI group, which will be responsible for displaying the pieces on the board and correctly displaying the outputs by the moves group. The rules group will work with the moves group in order to check if moves made by the user follow the basic rules of chess, and will also alert the user when moves are made into or out of check. AI is responsible for the chess strategy that will be executed by the program.

In order for the game to run, the basic framework of the game through the "moves" group must be designed in order for the other groups to have a starting point to build off of. The output of the "moves" will be used by the "Rules" group in order to check if moves are valid. AI will be built when moves and rules are both functional. GUI can be built throughout, as the user interface does not affect the functionality of the game.

Chess Olympian is distributed WITHOUT ANY WARRANTY, OR EVEN IMPLIED WARRANTY.

The software is licensed, not sold. This agreement only gives you some rights to use the software. Milk & Dim Sum reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the software only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the software that only allow you to use it in certain ways. You may not

- work around any technical limitations in the software;
- reverse engineer, decompile or disassemble the software, except and only to the extent that applicable law expressly permits, despite this limitation;
- make more copies of the software than specified in this agreement or allowed by applicable law, despite this limitation;
- publish the software for others to copy;
- rent, lease or lend the software; or
- use the software for commercial software hosting services.

You may make one backup copy of the media. You may use it only to reinstall the software.
Any person that has valid access to your computer or internal network may copy and use the documentation for your internal, reference purposes.
You may reassign the license to a different device any number of times, but not more than one time every 90 days. If you reassign, that other device becomes the "licensed device." If you retire the licensed device due to hardware failure, you may reassign the license sooner.
The first user of the software may make a one-time transfer of the software and this agreement directly to a third party. Before any permitted transfer, the other party must agree that this agreement applies to the transfer and use of the software. The first user must uninstall the software before transferring it separately from the device. The first user may not retain any copies.
The software is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the software. These laws include restrictions on destinations, end users and end use.

If you acquired the software in the United States, Washington state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.

If you acquired the software in any other country, the laws of that country apply.

This agreement describes certain legal rights. You may have other rights under the laws of your state or country. You may also have rights with respect to the party from whom you acquired the software. This agreement does not change your rights under the laws of your state or country if the laws of your state or country do not permit it to do so.

LIMITATION ON AND EXCLUSION OF DAMAGES. YOU CAN RECOVER FROM MILK & DIM SUM AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO THE

AMOUNT YOU PAID FOR THE SOFTWARE. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.

This limitation applies to

- anything related to the software, services, content (including code) on third party Internet sites, or third party programs; and
- claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

It also applies even if

- repair, replacement or a refund for the software does not fully compensate you for any losses; or
- Milk & Dim Sum knew or should have known about the possibility of the damages.

Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you. They also may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

References

- http://ximbiot.com/cvs/manual/cvs-1.11.23/cvs.html
- http://www.unknownroad.com/rtfm/gdbtut/gdbtoc.html
- http://gcc.gnu.org/
- http://wiki.libsdl.org/Introduction
- http://content.gpwiki.org/index.php/SDL:Tutorials:Displaying_a_Bitmap
- http://en.wiktionary.org/wiki/Appendix:Glossary_of_chess

Index