EECS 22L: Project 1

Prepared by: Weiwei Chen, Che-Wei Chang, and Prof. Rainer Dömer

January 8, 2013

1 The Game of Chess

This is the first team programming project of this course.

The goal of this programming exercise is to develop a chess program in which a user can play interactive chess against the computer. Of course, this is a big task, but it is not as difficult as it may sound.

This project is designed to be an interesting exercise where you can practice all elements of software programming and engineering and work in teams. In particular, you will practice specifying and documenting the software program, designing data structures and algorithms, designing software modules, writing original source code, testing and debugging the software program, and collaborating and communicating effectively in a team.

Now, let's go!

1.1 The rules of chess

As a first step, you should make yourself familiar with the rules of the chess game. The official rules of chess are found online on the web pages of the US Chess Federation in the Beginners Area at:
`http://archive.uschess.org/beginners/letsplay.pdf`

In case the online material is not accessible, a copy of this file is available on the **TA Infos** tab of the web page for this course.

Using the information from this online file, learn the rules of chess. Make yourself familiar with the goal of the game, the game play and the chess board. Specifically, learn about the six different pieces and how they can move on the board.

- The queen can move horizontally, vertically, and diagonally across the board.
- A rook can move horizontally and vertically across the board.
- A bishop can move diagonally across the board.
- A knight can jump to eight different squares which are two steps forward plus one step sideways from its current position.
- The 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.
- A pawn can 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.

Special Moves:

- ”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, 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.

- ”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. En passant capture is a common theme in chess compositions.

The chess game ends as soon as one king is trapped in ”checkmate”. That is, there is no move for the player possible which would get his king out of check. Then the player loses.

1.2 Program Features

In this course, you will practice the major tasks in software engineering to build your own software product.

We will not provide detailed instructions on how to design the program as we did for the assignments in EECS22. Instead, your team needs to come up with your own choices and practice designing the software architecture of a program and document it.

There are several functions that we expect the Chess program to have. These are the customer requirements for this project. As the designer, you will try to meet these requirements and provide more advanced options to satisfy and secure your customers.

Here are the basic functions that are required for this Chess Game program:

1. The game follows the official rules of Chess (see section 1.1).
2. The program shows a game interface where the player can see the game board and make moves. A minimal example of a game board interface is shown in section 1.2.2.
3. The program supports an interactive player and an auto player so that the user can play against the computer, one user can play against the second user, and the computer can play against itself.
4. The program keeps a log of all the moves so that the user can replay a certain game.
5. The program can take a specific board setup (usually not the initial setup) and continue to play the game.
6. The program supports different levels of players, such as beginner, intermediate, expert.
7. The auto player makes the move efficiently, i.e. the program compute the next move in a reasonable time.

There are some advanced options that are also desirable to have to satisfy the customers:

1. A satisfied customer will expect some graphics (we will discuss some GUI libraries in one of the lectures).
2. A satisfied customer will expect some hints on the next smart moves. Combined with the support of specific board setup, the program can now meet the specific customer request of solving the ”Position No. 6241” problem in LA Times on 01/06/13 (See Appendix). What is the smart next move for such board setup?
3. A satisfied customer will expect to have the chance to withdraw some of the moves.

4. The program supports the time control for both players.

You have the freedom to choose or come up with further advanced features so as to make the game more realistic, smarter, and easier to play. In the last week of this project, we will have a tournament between different teams as a demonstration of your work. If you wish to win the game, try advanced features to make your program better!

1.2.1 Design the program

We list several steps here to build a medium-sized programming project.

- *Design the software application specification*: work as a team to decide the functionalities of the program, the *input* and *output* of the program, and other things that describe the *features* of the program for the users.
- *Design the software architecture specification*: work as a team to design the data structure, program modules, application program interface (API) functions between modules, and basic algorithms that will be used to solve the problem.
- *Build the software package*: write the source code and implement the program. Each team member may be in charge of their own modules and work in parallel to implement the program. Use Makefile for rule-based compilation to integrate the modules from different owners.
- *Test and debug the software*: work as a team to decide the testing strategies, write the automatic test program or script, and debug the program if some of the test cases fail.
- *Version control and collaboration*: use version control applications to maintain the team project documentation and source code files. Team members can synchronize their own work with the others through the team project repository.
- *Software release*: release the software package with the executable program and documentations, e.g. the README file, user tutorials, etc.

1.2.2 An ASCII text-based chess board

In this section, we will show an example for a simple ASCII text-based chess board. You may use it as the initial interface of the chess game.

The following text shows the chess board with grids, pieces, and letter/numerical notions. It can be drawn by displaying the symbols such as *'+'*, *'-'*, *'|'* as well as numbers and letters on the screen. Different chess pieces can be represented in letters which indicate their colors ('b' for black, 'w' for white), and names (*'R'* for Rook, *'N'* for Knight, *'B'* for Bishop, *'Q'* for Queue, *'K'* for King, and *'P'* for Pawn).

This is the initial setup of the pieces on the board.

```
  +----+----+----+----+----+----+----+----+
8 | bR | bN | bB | bQ | bK | bB | bN | bR |
  +----+----+----+----+----+----+----+----+
7 | bP | bP | bP | bP | bP | bP | bP | bP |
  +----+----+----+----+----+----+----+----+
6 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
5 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
4 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
3 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
2 | wP | wP | wP | wP | wP | wP | wP | wP |
  +----+----+----+----+----+----+----+----+
1 | wR | wN | wB | wQ | wK | wB | wN | wR |
  +----+----+----+----+----+----+----+----+
    a    b    c    d    e    f    g    h
```

The following text shows the chess board with one white pawn being moved from grid *e2* to *e4*.

```
  +----+----+----+----+----+----+----+----+
8 | bR | bN | bB | bQ | bK | bB | bN | bR |
  +----+----+----+----+----+----+----+----+
7 | bP | bP | bP | bP | bP | bP | bP | bP |
  +----+----+----+----+----+----+----+----+
6 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
5 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
4 |    |    |    |    | wP |    |    |    |
  +----+----+----+----+----+----+----+----+
3 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
2 | wP | wP | wP | wP |    | wP | wP | wP |
  +----+----+----+----+----+----+----+----+
1 | wR | wN | wB | wQ | wK | wB | wN | wR |
  +----+----+----+----+----+----+----+----+
    a    b    c    d    e    f    g    h
```

1.2.3 Team work

The programming projects in this course will be performed by student teams. Teams of 4-5 students will be formed at the beginning of this project. Team work is an essential aspect of this class and every student needs to contribute to the team effort. While tasks may be assigned in a team to individual members, all members eventually share the responsibility for project deliverables.

A *team account* will be provided on the `ladera` server for each team to share data among the team members. Since teams will compete in the projects, sharing of data across teams is not permitted.

Every student is expected to show up in both the discussion and lab sessions for team meetings. Attendance at the lecture, discussion and lab sessions is mandatory for the sake of team work.

2 Extra Credits

Bonus points can be earned for this project. To name a few cases for extra credits:

- The winners for the tournament games.
- The team delivers all the functionalities that are specified in the specifications.
- The program supports some exciting advanced features.

3 What to turn in

To submit your team's work, you have to be logged in the server `ladera` by using your **team's account**. Also, you need to create a directory named `chess` in your team account, and put all the deliverables in that directory.
Note that one team only needs to submit one set of deliverables.

Here is a checklist of the files you should have for each week:
We do require these *exact* file names. If you use different file names, we will not see your files for grading.

Now, you should change the current directory to the directory containing the `chess` directory. Then type the command:

```
%  /ecelib/bin/turnin22
```

Table 1: The Chess Grame project deliverables

Week	File Name	File Description	Due Date
1	`Chess_User_Spec.pdf`	The application specification	01/14/13 at 12:00pm
2	`Chess_SW_Spec.pdf`	The software architecture specification	01/21/13 at 12:00pm
3	`Chess_Alpha.tar.gz`	The alpha version of the chess program, including the program source code and documentation	01/28/13 at 12:00pm
4	`Chess_Beta.tar.gz`	The beta version of the chess program, including the program source code and documentation especially the software testing specification and status	02/04/13 at 12:00pm
5	`Chess.tar.gz` `Chess_Source.tar.gz`	The released software package for the chess game and the program source code and documentation	02/11/13 at 12:00pm

which will guide you through the submission process.
You will be asked if you want to submit the script file. Type yes or no. If you type "n" or "y" or just plain return, they will be ignored and be taken as a no. You can use the same command to update your submitted files until the submission deadline.
Below is an example of how you would submit your homework:

```
% ls # This step is just to make sure that you are in the correct directory that contains chess/
chess/
ladera% /ecelib/bin/turnin22
============================================================
EECSL 22L Spring 2013:
Project "chess" submission for weiweic
Due date:  Mon Jan 14 12:00:00 2013
* Looking for files:
* Chess_User_Spec.pdf
============================================================
Please confirm the following:  *
"I have read the Section on Academic Honesty in the *
UCI Catalogue of Classes (available online at *
http://www.editor.uci.edu/catalogue/appx/appx.2.htm#gen0) *
and submit myoriginal work accordingly." *
Please type YES to confirm.  y
============================================================
Submit Chess_User_Spec.pdf [yes, no]?  y
File Chess_User_Spec.pdf has been submitted
============================================================
Summary:
============================================================
Submitted on Sun Jan 6 00:55:31 2013
You just submitted file(s):
Chess_User_Spec.pdf
% _
```

3.1 Verify your submission

This step is optional, but recommended. If you want to confirm which files you have submitted, call the following command:

```
% /users/grad2/doemer/eecs22/bin/listfiles.py
```

This command lists your submitted files. Don't worry if you submitted too many files. We will only look at the files with defined names (here: `Chess_User_Spec.pdf`, `Chess_SW_Spec.pdf`, `Chess_Alpha.tar.gz`, `Chess_Beta.tar.gz`, `Chess.tar.gz` and `Chess_Source.tar.gz`) and ignore other files.

4 Appendix

The article with a specific customer request for the Chess game from LA Time on 01/06/13:
`https://eee.uci.edu/13w/18020/tas/Chess_LATimes_20130106.pdf`