Chess Game

Authors: Raheel Yanful (ray37), Nathaniel Watson (nww23), William Gao (wg253)

**2-3 page document containing the following information:

Main Project Handout

Plan for Meeting

Monday nights, Wednesday nights, Saturday afternoons

Proposal

Summarize the system you intend to build. Tell us what will be the most important functionality of your system. This summary should be about one page long. Provide:

A very short statement of the core vision or key idea for your system.

- We will build an implementation of a chess board with pieces that can be played by a user against another user. We will also design an AI that the User can play against.

A short (no more than six items) bulleted list of the key features of the system.

- Data structures (and JavaScript GUI?) representing a chess board, and each player's captured/remaining pieces
- Pieces with differing point values associated with spots on the board
- Ability to move and capture pieces on the board according to chess rules for certain pieces
- Single player and multiplayer capabilities
- A win condition
- An artificial intelligence that makes intelligent moves with ranging difficulties: Easy (random moves) Medium (Few moves of insight) Expert (Many moves of insight)

A narrative description of the system you intend to build. Go into enough detail that, if your charter were given to another team, and that team were never allowed to talk to you, they would still understand more or less the functionality that you have in mind.

Roadmap for Our System

There are going to be three "sprints" (implementation phases) after this charter: MS1 (alpha), MS2 (beta), MS3 (release) Each will last about two weeks. Make a list of what features you plan to implement in each sprint. Then, for the features you plan to implement in MS1, sort them into three categories— Satisfactory, Good, and Excellent—like how the programming assignments in this class have been. That gives you a plan for what to do if MS1 is going worse or better than you expected.

MS1: Basic 2 player chess game that allows user to

- A data structure representing a chess board, each players captured, and remaining pieces
- Pieces with differing point values associated with spots on the board
- Move different pieces on board according to rules
    - *piece* to *position on board*
- Winner vs loser
- Single player

MS2: An AI that makes sort of smart moves (maybe 1 move ahead)

Easy: random, medium: 2-3 move ahead, position evaluation

MS3: An AI that makes smart moves

Hard: Search tree with min-max and alpha-beta pruning, improved evaluation. Potentially different evaluation methods for early game, mid game, and end game

Preliminary Design Sketch

Spend some time thinking through what you are going to need to build. Of course, your plans will evolve throughout the project. But it's good to have talked as a team up front about what seems necessary. In about one page, answer the following questions:

What are the important modules that will be implemented? What is the purpose of each module?

- Game:
    - pieces, board, rules
- State:
    - captured/remaining for each player; validate moves
- Command:
    - legal moves
- Brain
    - Decision making based on binary search tree of future moves and position evaluation
- Interface

What data will your system maintain? What formats will be used for storage or communication? What data structures do you expect to use as part of your implementation?

The system will maintain information about where all the pieces are in a two-dimensional array, possible future moves in a binary search tree, other miscellaneous information will be stored in state.

What third-party libraries (if any) will you use?

Javascript might be used to visualize the board in a GUI

How will you test your system throughout development? What kinds of unit tests will you write? How will you, as a team, commit to following your testing plan and holding each other accountable for writing correct code?

Test-driven development for setting up the board and then play-testing afterwards

High level design -> Document/write specs -> test -> code

Files

1. Game:
    a. Module pieces: takes in functor to make piece?
    b. Module board,
        i. 2d array board
    c. Module rules

i. Check
ii. Win condition: checkmate
iii. Castling
iv. En passant
v. Functions to check validity

2. State:
   a. captured/remaining for each player; validate moves
   b. Alter game state
   c. State = 2d array with info on each box
3. Command:
   a. legal moves
4. Brain
5. Interface

Testing

1. Unit testings different features
   a. Ounit, #trace
2. Play testing
   a. Function loads game from input state