Huq2Chess- extension of a Java Chess Application

1. **Abstract**
   1. **Document’s Purpose**

Describes the requirements of Huq2Chess. It will extend the original chess application to include online support for gameplay, refurbished GUI and extended playing options, including Hexagonal Chess boards and custom pieces.

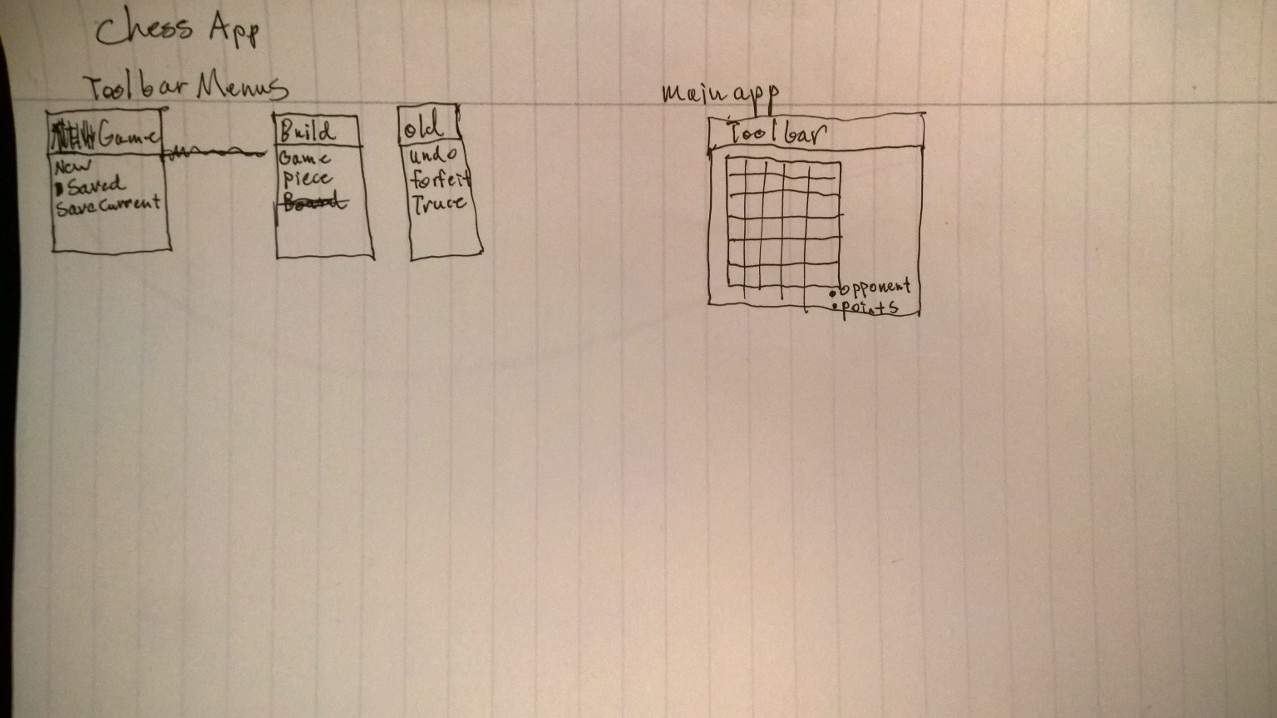
* 1. **Definitions**

Well-formed code: code with aesthetics. It goes beyond functional requirements to make it as reusable as possible. It’s well documented, Modular, shy, and self-explaining to name a few features.

Really Crappy AI: look, its either gonna try to look ahead 5 turns against another AI that looks ahead 3 or it’s gonna try some opening moves. It ain’t getting more than a week’s worth of work brah’.

* 1. **Background/Motivation**

Purely for enjoyment and validation. The CS242 course that the original code base was built for is designed to motivate students to write well-formed code as opposed to purely functional code. The application aims to prove that writing well-formed code is good by adding many complex and interesting features to the original code base. Also, who doesn’t think games are fun?

1. **Technical Specifications**
   1. **Platform**: Java Application
   2. **Languages**:
      1. Main application: Java
      2. Online support: may be done with any of PHP, HTML, Python+DJango depending on time and Language limitations
   3. **Coding-Standard**: <http://geosoft.no/development/javastyle.html>
   4. **SDK**:
      1. Main application: Java Development Kit 1.7
      2. Online support: Python
   5. **IDE**:
      1. Main application: Eclipse Kepler
      2. Online support: Eclipse+ PyDev+Aptana plug-in for Online Support
   6. **Interface**:
      1. Main application: Eclipse Kepler
      2. Online support: Eclipse+Pydev, Django DB for checking database, GoogleChrome for reviewing online odds and ends
   7. **Other Technical details**: my experience lies with Python and DJango, but I have yet to use a free running server (test server only). Because of this, cpanel might prove better to use when hosting, but cpanel lacks django support, so it still is not the most preferable option available.
2. **Functional Specifications**
   1. **Affordances**
      1. User can build, save, and delete a customized game of Chess
      2. User can save, load, and delete a game in progress
      3. User can play a custom or built in game of Chess
      4. User can play a game of chess remotely with another player
      5. Game can display animations of moves and chess pieces
      6. User can play against a really crappy AI
   2. **Features**
      1. Using Swing, menus will be opened up to allow users to build Chess Games, these will then be saved as a generated java file and indexed in a JSON file including games to be referenced later
      2. Similar to 3.b.i, but with Chess Pieces instead of games
      3. Using Swing, a menu will be available from the application’s toolbar to select a game for play
      4. User’s will register to a remote server and request a game from other users found in the remote server. When a match is found, both users will confirm and the server will send data to-and-from either user
      5. Found in the main application loop, a function will be used to run through each pieces’ animation, using a layering system to dictate which is drawn in front. It is possible that an “image box” will be used to detail where the animation takes place, so that not all of the board need be redrawn during each image drawing. Further, it is possible to parallelize the drawing function, using the image boxes as mutually exclusive spaces to draw in.
      6. The player can choose to play a game against a really crappy AI from the application’s toolbar
   3. Scope
      1. Users can build their games using a set of predefined boards and rules, and can add their own description using text.
      2. Custom boards can be changed with limitations: spaces can be “omitted” but the overall shape of the board, specifically the spaces, cannot be changed.
      3. Pieces follow similar limitations: moves can be built with limitations: movements are specified with vectors and a set of capture rules used by other pieces. They can add static images for the chess piece, but must use coloring conventions similar to other images for color swapping.
      4. User cannot play more than one game at a time
   4. MockUP
3. Timeline:

Week 1-Expand the Library

* Build Hexagonal Chess Functionality
* Build 1 or 2 Chess Games that use Hexagonal Chess
* Create GUI for Hexagonal board

Week 2-Make it Customizable

* Complete work on Hexagonal Chess, including menu for game selection and piece functionality
* Build Customizable ChessPiece and file interpreter
* Build Customizable ChessGames and file interpreter

Week 3- Make it Flashy

* Build menu and backend for loading games in progress
* Reverse display depending on player
* build application menus for Custom Pieces and Games

Week4- Share it with Friends

-cry at home alone

-build server to help connect players

-Build application menu for game requests

-Build Http request interpreter to interpret commands

-maybe a chat box?