Michael Barns & Shiqi Wu

5/6/2017

Group 1: Fischer-Random Chess

Our project is a game consisting of two obscure variations of chess. Both variations, “Really Bad Chess” and “Chaos Chess”, are completely randomized every new game. All the rules are the same and the game is designed to be played by two players on the same computer. The constraints of our game are the same as regular chess. Some examples include each player can only move one piece at a time, and no pieces except the knight can jump over pieces.

Some constraints of our code are that each game mode has different rules for generating the game pieces, so two different snippets of code will be needed for each chess variation. The game is coded to run until either “stop game” is pressed in the GUI or a king is taken off the board. The GUI and looks of our program will be very similar to the simple Windows games like minesweeper and solitaire. The Gui will consist of a menu bar where players can check how to play, can start and stop the game, and can change game modes. Inside the main window there will be a blank chess board and two message boxes.

Both members equally contributed to the project. Tasks from idea making, to gui designing, to coding was equally delegated.

The most significant problem we encountered was with our array of class objects. We had a class that handled all the functions and variables for a single tile “SquareHandler” and in a separate class “squares” we created an 8x8 array of the first class “SquareHandler”. The problems started when we tried to manipulate the array from a different class then it was declared. There are three or four situations where this was needed. Our workaround was to change our array of class objects to static because of how many times it was needed to be manipulated across different cpps. Declaring it static gave use a whole set of new problems given that we never worked with an object as complex as a static 8x8 multidimensional array of class type objects. It was a learning experience and after finishing the project it’s great to know that we are capable of such complex programming tasks.

Our project consists of three headers and four sources. The headers are mainwindow.h squarehandler.h and squares.h. The sources are main.cpp mainwindow.cpp squarehandler.cpp and squares.cpp. The purpose of mainwindow is to create the game window on which everything is drawn and it also has seven different functions for when different menu bar options are pressed in the GUI. The main purpose of squares was to create all the squares of the chessboard. It has seven functions, three of them (formatting(QWidget),chessboard(QWidget),chessPieces(int)) spawn the graphics. Formatting and chessboard are both called upon creation of the square object in main.cpp At this point the gui is formatted and the chessboard is drawn. Chesspieces isn’t called until the player selects new game in the menu bar. The other four functions cover other miscellaneous tasks. changePixmap(int, int, char, char) changes the image on a single tile, this is needed when the player moves a chess piece. editText(QString, int) is basically a mutator function that edits the textboxes on either side of the chess board. checkForJump(int, int, int, int) checks to make sure the player isn’t trying to jump over a piece. clearBoard() is just a simple function that clears the board of all pieces. Finally, squarehandler handles all the tasks needed for each individual square. This includes setting the tiles setStyleSheets in squareGeneration(), setting the Qpixmaps of the chess pieces in pieceGeneration(). There is one function for when a square is clicked on, another to change the pixmap of a tile, and one that checks if the move the player is trying to make is a valid move. The last function in squarehandler is a simple function that takes a char and returns the full string. If I needed to display the name of a piece I would call this function with my char as an argument and it would return the full piece name.

To run our project, you will need to first select your game mode. The game mode can be selected by clicking ‘’game” then “game mode” in the menu bar at the top of the window. Once our game mode is selected click new game to begin playing. To move the pieces simply click on the piece you wish to move, then click on the tile you want to move to. If you accidentally click the wrong piece or change your mind you can deselect your choice by either placing your piece back where it originally was or by right clicking anywhere. The game will go back and forth between black and white, starting with white just like traditional chess. The game continues until either a king is removed or the game is ended with the end game function of the menu bar.