Design Document

Object Oriented Chess

Problem Definition

Two player Chess game. The players will enter their moves using Smith Notation. Smith Notation is a coordinate based system that allows the player’s game to be saved. Have the game be implemented using object oriented programming.

Design Overview

The game will be stored in an 8 X 8 array. When the array is read if the letter is capital then it will be a black piece, if it is lower case it will be white. There will need to be a bool that will switch back and forth for which turn it is.

A pawn once it has moved for the first time will become capitalized in the background. It can no longer move two spaces at once. The pawn will then only be able to move forward one space unless there is an object diagonal from it forwards which we will use the bishop code for only if there are targets available. If there is an enemy pawn next to them when the move up two spaces they will remain a lower case p until the enemy moves or they move up one. This way it will be known if en-passant will be valid or not. If it is black pawn and they are on row 5 and a white pawn is on row 5 than en-passant is valid and the black piece may capture it by going diagonally behind it

To make sure that the rook move is legal I will check to be sure only one dimension of the array is changed. For example to move the rook on a horizontal access only the array would only be able to change x board[x][]. For the bishop both dimensions of the array should change proportionally. For example both the x and y would be added by 5 to go diagonally up 5 spaces. The Queen will be the bishop and rook combined. If there is a space occupied in path of the desired location you won’t be able to move past it with the rook, queen, bishop, king and pawn. There will need to be a function to see how far a piece can go without hitting an object.

The knight will be required to have a change of 3 spaces total. It will have to move two spaces in one direction and one in the other. A good way of checking this is subtracting the new location x and y added from the old location x and y added and then checking if it is equal to 3. Then will have to be sure that either the change in x or Y is 2 and that the other is changed by 1.

|  |  |  |
| --- | --- | --- |
| *Start*  *Board* |  | Display of board  When the program is first started |
| *Make a*  *Move* |  | How the program will show the move that the user made in the game while in test mode. |
| *Display*  *Options* |  | If help is needed in remembering what commands are available the “?” will show the options. Has a list of all the options |
| *Quit*  *Option* |  | What will be displayed if user inputs quit a question if they would like to save the game or just quit |
| *Read*  *Option* |  | What will be displayed if user inputs read. It will ask the filename they wish to save their game to. |
| *Test*  *Option* |  | What will be displayed if the user wants to run through testbed. The game board will be displayed without the bells and whistles. |

Output

Input

|  |  |
| --- | --- |
| Input | Description |
| Coordinates ex b2b4 | By using Smith Notation will move a piece |
| ? | Will display option of valid user commands |
| Read | Allows user to bring up the option for a game file to be read. User must provide filename |
| Test | Allows the user to run through the game on a simplified board. |
| Quit | Option to exit the program |
| Help | Shows the possible moves for a selected piece |
| Rank | Shows the force time and space |

Errors

|  |
| --- |
|  |
|  |
|  |
|  |

File Format

What: The Smith Notation will be saved in a vector. One White Move and then One Black move per row.

How : The file will be read and would be inserted as moves are entered. When the file is read in a reset board function will set the board to its original state first. The file will be read in through a loop.

Guideline: The example file would look like this

|  |
| --- |
| A1a2 a2a4p  A3a4 a4a5  A5a6 |

All files will be stored as txt files.

User Errors

|  |  |  |
| --- | --- | --- |
| Name | Condition | Handle |
| Invalid coordinate | Not “letter # letter #” | reprompt |
| Moving empty piece | First coordinate is empty | reprompt |

File Errors

|  |  |  |
| --- | --- | --- |
| Name | Condition | Handle |
| Invalid Read File | File doesn’t exist, or file doesn’t follow Smith Notation | Reprompt |
| Invalid save file | The file they tried to save was full or already used | Reprompt |

Internal Errors

|  |  |  |
| --- | --- | --- |
| Name | Condition | Handle |
| Invalid move store | When storing the move just made somehow passed through is no longer smith notation | assert |
| Stuck pawn | If a pawn gets to the edge of the board but doesn’t promote. | assert |