Laboon Chess

CS 1530 - SPRINT 4 DELIVERABLE

10 Nov 2016

Craig Kodman (*)	[*c-kodman*]
David Bickford	[*drb56*]
Joe Meszar	[*outkst*]
David Tsui	[*luckyleap*]

* SCRUM MASTER

II. Accomplishments / Summary

Our focus for this sprint was on the two new features added by our customer: the addition of the feature for a user to flip the board and play from either the black or white side; the feature for a user to select a chess piece color and use the color in place of the black or white default chess pieces. In addition to the two features, we have also completed the exciting integration of the chess board with Stock Fish's binaries, allowing the user to play against the computer. As well, we are almost completed with our FEN string conversion. Once the FEN string conversion is complete, our program will be able to take into account special moves such as castling, en passant, and draws. Of course, we have also worked on maximizing our test coverage for the two features as well as building our tests for the features we are still working on.

During our sprint over the past two weeks, our team has been using the same methods of communication as the previous sprints. This communication involves mainly using slack messaging and phone communication for quick questions and meeting arrangements, personal meetings for scrum meeting, and working together physically for faster questions on specific code. Our scrum meeting has shifted to meeting immediately after class since that is when everyone is free. This meeting style, with slack, scrum meeting, and working together, has proven to be an effective method of communication as it is flexible and time efficient.

The difficulties we have encountered during this sprint were wide and varied as we have each worked mostly independently on separate features. For the two focused features, the primary difficulties came with using JavaFX libraries. In the first feature, one major problem that was encountered was converting the FEN string to the other side of the board. This was solved by using a StringBuilder to efficiently reverse the string and then getting the Chessboard array to take in the new string. In the second feature, there was a difficulty figuring out how to color the chess piece images. There were many options to take to complete this task; from editing the pictures and having separate, permanent images for each desired color, to programmatically changing the color of a single set of chess images. The latter option was taken, and a brushing function was found within the API of JavaFX—`Blend`. This function allows the program to take static images and color them to a wide array of artistic styles. In our case, we just wanted to paint the non-transparent parts of our image (the actual chess piece). Iterating over the chessboard to get to each chess piece was difficult as well due to the JavaFX `GridPane` object not having a built-in iterator. This was remedied by

creating an iterator that looped through all chess squares (JavaFX `Panes`) and looked for objects that held an image. Finally, in integrating the StockFish, there was an initial difficulty in getting the StockFish binary to take in the commands. However, it turned out that an empty string command had to be set for the StockFish engine had to be initialized.

III. User Stories Completed

As a chess player,
I want the ability to flip the board,
So that I can view the game from the perspective of the other player.

As a chess player,
I want the ability to change the colors of my pieces,
So that I can play the game with a pleasing color scheme.

As a chess player,
I want the ability select a game with a chess AI playing black side,
So that I can play against a computer opponent

As a chess player,
I want the ability to select a game against a chess AI playing white side,
So that I can play against a computer opponent

As a chess player,
I want the ability to select a game with no chess AI,
So that I can play against a human opponent

As a chess player,
I want the ability to select ‘Easy’, ‘Medium’, ‘Difficult’ for a chess AI,
So that I can play against a computer opponent at different difficulties

IV. GitHub Repository Link

https://github.com/drb56/CS1530

V. Defects

Defect 1 was solved. The cause of the defect was that moveStockFish() did not remove a chess piece from the board when it moves. This defect was solved by implementing a similar chess piece removal function, both in the data structure and in GUI every time the function is called.

Defect 2 is still not solved. The reason is that it is not a non halting issue for this sprint. Players, for now, can simply exit the program and rerun the program to play against a new opponent, or to start over. We will work to solve this defect next sprint.

Defect 3 was solved. The cause of the defect is that there was a boolean logic error in the condition that checks for whether a player can move based on his / her turn. This defect was solved by re-implementing the boolean logic.

Defect 4 was solved by adding “kqKQ” to the end of the string being passed into the test for testChessboardToFENGameStart(). The cause of this defect was that the functionality of the game was increased by allowing for the FEN string to account for castling once we added the method generateCastleFen(). This new method added “kqKQ” to the FEN string but our test didn’t account for this. Once we simply added “kqKQ” to the test string, the test passed.

Defect 5 is not solved as of right now. It will be solved in the next couple of days. The FEN string is updating castling options one move later than it should be. As it stands, maybe this isn’t really a defect because while white could possibly castle due to the annotation for castling at the end of the string, the string prevents this by only allowing black to move. Once it is white’s move, the correct castling annotation is in the FEN string. Not sure this has to be fixed then, but if it does need fixed it is probably an easy fix, but we are running out of time to take care of it during this sprint.

Defect 6 is not solved as of right now. Playing the game, the AI only had a king left and had only one legal move left, but the game stopped due to an error. This was found during manual testing and happened too close to the end of the sprint to address. This error was found more than once in manual testing, and always with the AI having only one legal move left.

Defect 1. Stock Fish did not replace a chess piece when it captures the chess piece

[Reproduction Step]:
1. Select ‘New Game: Computer Black’
2. Move pawn H2 to H4
3. Move pawn H4 to H5
4. Continue moving pawn forward until first capture by Black

[Expected Behavior]:
- As the white pawn is captured, the white pawn disappears from the board and the black capture piece now overlays the current space which the white pawn occupies.
- Stock Fish will continue to make further moves, moving chess pieces during its turn

[Observed Behavior]:
- As the black piece captures, the white pawn does not disappear
- Stock Fish stops making any further move

Defect 2. Selecting new game mid match does not reset pieces

[Reproduction Step]:
1. Select ‘New Game: Computer White’
2. Move pawn D7 to D5
3. Select ‘New Game: Computer Black’

[Expected Behavior]:
- The game is reset and all the pieces are back to the starting position with FEN: rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
- Computer is playing as Black

[Observed Behavior]:
- The game continues as is, no pieces are set back to position
- Computer continues playing as White

Defect 3. Selecting new match against Black AI opponent Makes Chess Pieces Unselectable

[Reproduction Step]:
1. Select ‘New Game: Computer: Black’
2. Select pawn ‘H2’

[Expected Behavior]:
- H2 pawn is shaded.

- Further selection of H3 of H4 moves pawn forward

[Observed Behavior]:

- H2 pawn is not shaded.
- No further actions on H2 pawn can be taken.

Defect 4. After adding the ability to determine via the FEN string if castling is still possible, the test testChessboardToFENGameStart() fails

[Reproduction Step]:

1. Type gradle test on the command line.

[Expected Behavior]:

- The test should pass by passing in the correct FEN string to represent the board at the start of the game.

[Observed Behavior]:

- The test failed because we added a method, generateCastleFen(), that added "kqKQ" to the FEN string at the beginning of the game, but the string we passed into the test did not include "kqKQ".

Defect 5. FEN string is updating whether castling is still legal one move after it should be.

[Reproduction Step]:

1. Select 'New Game: Multiplayer'
2. Move the white pawn from H2 to H4
3. Move the black pawn from H7 to H5
4. Move the white rook from H1 to H3

[Expected Behavior]:

- The manual test should pass by displaying the correct FEN string to represent the board after these three moves which should be "rnbqkbnr/pppppppp1/8/7p/7P/7R/PPPPPPP1/RNBQKBN1 b kqQ"

[Observed Behavior]:

- When looking at the FEN string at this point in the game the it was "rnbqkbnr/pppppppp1/8/7p/7P/7R/PPPPPPP1/RNBQKBN1 b kqKQ".

Defect 6. Playing against the AI, where the AI had a legal move left with their lone piece remaining (king), the game stopped and produced an error.

[Reproduction Step]:

1. Select ‘File’
2. Select ‘New Game: Computer Black’
3. Play the game until the AI only has a black king left and that king only has one move possible

[Expected Behavior]:

- The AI should make that only possible move, which is to C8.

[Observed Behavior]:

- In line 97 of LaboonChessDocumentController.java, String move = stockfish.getBestMove(fen, timeWait), the game breaks.