Evan DePosit
June 10th, 2019
CS 564
Full-Stack Web Development
Journal Notes

Backend Notes on Problems Encountered and Remedies

- Problem: To begin the backend I had to create a mockup of the front end. One error I ran into was the board position was getting passed to the server as undefined. Even though it was initialized on the front end. This resulted in the incorrect turn order happening at the start of the game
    - The problem turned out to be that the variable in the object data had the same key name as the variable on the front end.
        - {'turn': turn}
    - When I named the objectdata key a different name from the local variable it was passed correctly to the backend
        - {'turn': localTurn}
    - But later I found that the front end had switched back to using {'turn': turn} and it worked. So the error was inconsistent.

- Problem: pieces that are connected share freedoms, so a peace can be completely surrounded by other pieces, but as long as it is connected to another piece of the same color it should remain uncaptured if the neighboring piece is adjacent to a blank space or if the pieces that it is connected to are connected to a piece that is adjacent to a blank space
    - I created an algorithm that checks to see if a piece if surrounded by the opponents pieces, if it is it is captured.
    - If the piece has blank space it is uncaptured and the algorithm moves on to the checking the next location on the board.
    - If the piece being checked is connected to other pieces of the same color, the algorithm recursively checks all the neighboring pieces of the same color to see if eventually it arrives a piece that is adjacent to a blank space. Once it finds the adjacent space, that freedom is returned up the call chain until it arrives at the first piece being checked in the group and marks it for not being captured.

- Problem: blank spaces count toward the score of a player if they are completely surrounded by that players pieces.
    - The algorithm simply adds a point to each players' score for each space that they occupy with a stone.
    - For groups of blank spaces the algorithm works similar to the capturing algorithm
        - When the algorithm encounters a blank space, it counts it and then recursively checks each blank space that is connected to it.

        - If encounters an adjacent space with a stone on it, it sets that color to true. When it reaches the last blank space it returns the total number of blank spaces it has encountered and which colors of stones it has encountered on the edge of the blank space.
        - If it has encountered both color of stones surrounding the blank space group, then neither player receives the points. If has only encountered one color surrounding the group of blank spaces, then the player of that color receives the points.

- Problem: I numbered the board left to right 1 to 19, top to bottom 1 to 19. But Tegan used the official numbering system of 1 to 19 left to right. 19 to 1 from top to bottom. Also the Y coordinate is listed before the X coordinate.
    - I had to go through the capturing algorithm and change all the conditions that preceded the checks for recursive call on the neighboring spot
        - Example: checking the left neighbor went from get_val(x-1, y) to get_val(x, y-1). So the y coord had to become the horizontal coordinate
        - Checking the position above went from get_val(x, y-1) to get_val(x+1, y) because now the board is numbered from bottom to top instead of from top to bottom.
    - I also had change the conditions under which surrounding stones were checked.
        - Previously I checked the stone above if only if the y coordinate was greater than 1.
        - But now the x is actually the y, and the vertical coordinate at the top is 19 so now I check the stone above if the x value is less than 19
    - After altering the condition statements and coordinates in functions calls, the capturing algorithm wasn't recursively calling itself correctly on stones below the stone being checked
    - Solution was to flip a greater sign that broke the if statement before the call on bottom neighbor.

- Problem: Scoring was not recursing far enough so it was counting the largest blank spot on the board as two disconnected blank spots. This was causing incorrect scoring.
    - I Console.log() the coordinates of each space that the function was being called on
    - The recursive call should go left, right, top, bottom, but it wasn't going left after the second row on the board like it should.
    - A condition in an if statement had a parentheses misplaced so it was never true, which caused the left blank space to never have the function called on it
    - I Moved the parentheses and the game was scored correctly

- Problem: I've been coding in python for my other project, so I kept forgetting to add the var in front of variables

- Node was surprisingly inconsistent in catching these mistakes, so it worked locally, but then the errors only showed up when we connected it to the front end.