CS5610-Web Development | Project 1: Othello

Project Report

Vipul Sharma

Manikantha Dronamraju

# Introduction and Game Description

The project is an implementation of the board game Othello. Othello is marketed/popular name of the game Reversi, invented in 1883 by either of two Englishmen (each claiming the other a fraud), Lewis Waterman or John W. Mollett. Othello is a strategy board game for two players, played on an 8x8 unchecked board. Each square in the grid can either be empty or contain a piece. There are sixty-four identical game pieces called disks (often spelled "discs"), which are light on one side and dark on the other. Players take turns placing disks on the board with their assigned color facing up. At the start of the game, four discs are placed in the center of the board, with two dark(black) disks diagonal across and two light discs diagonally across while all of them are adjacent. During a play, any disks of the opponent's color that are in a straight line and bounded by the disk just placed and another disk of the current player's color are turned over to the current player's color. A valid move in the game consists of a move where at least one of the piece is reversed.

The object of the game is to have most disks turned to display your color when the last playable empty square is filled. Some variants of the game exist where the starting position of the pieces differ from the standard order or the objective of the game is reversed, ie, the one having the least pieces at the end wins the game, are sometimes-but rarely played.

UI Design

The UI of the game has been mainly focused on three technologies: ReactJS, Bootstrap4 and CSS; however, we have also used some elements of react-strap too in it.

A dark background theme (black with a CSS gradient generated image) is used for the game. The board has been given a Forest Green color for the board, which I subtly florescent and soothing to eye at the same time.

The layout has been divided using Bootstrap’s containers, with appropriate padding where and when necessary. Reactstrap cards have been used instead of Bootstrap cards in the JSX for better compatibility, and to display the necessary information to the user in an eye-catching manner. react-shapes library has been used to draw disks, and use the auto-close with close b handle toastify element to display the data to the user during the game. We have also used Bootstrap Modal class for user input instead of pop-up windows to make the user experience better and comfortable.

The game is divided into two pages: the home page (or the lobby) where users can see all the games on the server and has the options to create/join/observe a game and the game page, where the game actually happens. Additionally, two more pages have also been provided to users, View all Games: gives a list of all the Games that are being played or have been completed with a full list of players involved and Observers if any. The page player list: Gives the list of players and the Game they are playing/have played. A navigation bar at the top facilitates for the easy movement of user to switch from one part to another.

The game page is divided into three parts, status boards, the Othello board and the Chat window. The status is displayed on the left corner, consisting of 4 cards which notifies, black & white players info and status, the score of each player at any moment and status of the game: is it in progress or it finished! This has been achieved by using various Reactstrap cards.

The center of the page is occupied by the main game board, the Othello game board, green in color. Here the user whose turn is up can click and chose the position of their disc which is reflected instantaneously if it is legal move. If the user clicks on any square where he/she cannot play, a toast message is displayed to notify the user. All the observers of the game can only view the game, their clicks will result into no change in the state of the game. The board is implemented as a series of Bootstrap buttons, which have been encapsulated into 8 rows and 8 columns.

The rightmost part of the game is provided with a chat window, where the users can chat among themselves. Observers, both registered and guest, can also chat in the chatroom. For each move a player makes, is also updated in the chatroom. The chat is visible to both the players and all the observers of the game. The chatroom is constructed by amalgamating simple html elements, text area, input box and buttons.

Implementation of Game Rules

We have tried to keep the rule of the game as close to the original rules of the game Reversi, with some minor twists to add some zest. The user who joins the Game is assigned as the white player. The user has to give a unique name for the game, if he/she fails to, they will automatically join any existing game as a spectator of Player 2. These actions are performed by the function add\_new\_player(). The observers are distinguished from the players by using the function who\_am\_I/2, which takes the game state and the user under question as arguments and check against the player of the game. If they are not one of them they are inducted as observers and appended to a map, used to display the list of observers in the game.

After a game is created, another player can look up for any games in which another player is waiting and join the game to play the game. The second joining player is assigned the black side of the game. The black assigned player moves first, in the meantime the other (white player) has to wait for the black player to finish his/her move.

The board is constructed by the function othello/0 which defines the grid position of the players, while initializing the cross position at the start of the game. The function client\_view/1 initializes the board first, which is used by the client to render it to the user.

A point is awarded for each successful flip of the user. The main function for handling this task is move/4, which takes the game instance, disc color and the position of click as input and process the input from the user. The function change\_player/1 keeps a track of all the moves, and increment/decrement their points and the state of the game, by calling the function update\_score/2 which take the fame and the cell number (location of the click as input). In the process the move is checked by the server with help of the functions is\_legal\_move?/6 & is\_legal\_move?/3 and same\_disc?/6 functions to make absolutely sure if the move was legal or not. If the move is not legal, an auto closing with feature to close toast message displays it to the user. React-toast library has been used for it. If everything is okay, the function change\_disc\_count/5 updates the disc counts for the players. No time limit has been imposed for a user to take a turn.

The game continues till all the squares have been covered by both the users or one of the user doesn’t have any valid move left. At the end of the game a winner is chosen, the player having the maximum score wins and the game ends. The status is conveyed to the user by a toast message and the change in status in the score board. The task is done by the function legal\_moves\_options/2.

The chat server is controlled by the start\_chat/3 and handle\_in/3 functions.