Project: Othello

CS 458

1 Overview

For this project you'll be creating an app that allows the user to play Othello[1] against another local, human player. Your app will need to facilitate actually playing the game – including updating the board after a move – as well as keeping score, indicating whose turn it is, and providing the opportunity to start a new game. This assignment also provides an opportunity for a bit of extra credit.

2 The Rules of Othello

Othello is a board game lightly adapted from the older game Reversi. The primary difference is that Othello defines the start state of the board, while Reversi allows players to configure it.

2.1 Overview

Othello is a game for two players, played on an 8x8 grid. There are sixty-four identical game pieces called "disks", which are (traditionally) light on one side and dark on the other. Players take turns placing disks on the board, with their assigned color face-up. Once a piece is played, any disks of the opponents color that form a straight line between the newly-placed disk and another disk of the current player's color are flipped to the current player's color. All placements must result in at least one flipped disk. Play continues until a player can no longer take their turn, either because the board is full or there are no valid moves.

2.2 Initial Configuration

For Othello (but not Reversi), play always starts with four disks placed in a square in the middle of the board. The disks are arranged so that the north-east and south-west squares are dark-side-up, while the other two are light. The dark player moves first.

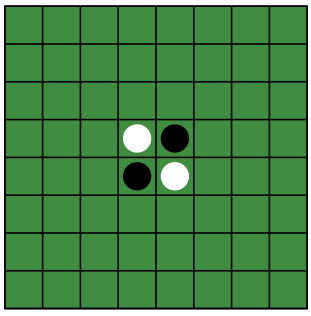

Figure 1: Initial configuration of Othello. A green background is traditional, but not required.

[1] aka Reversi: `https://en.wikipedia.org/wiki/Reversi`

2.3 Moves and Captures

To begin the game, the dark player must play a piece (dark side up) on the board. The piece must be positioned such that:

- It is adjacent to at least one light-colored piece (horizontally, vertically, or diagonally)
- There is a contiguous line from it, through at least one adjacent light piece, to another dark piece

The following figure shows possible first moves for the dark player:

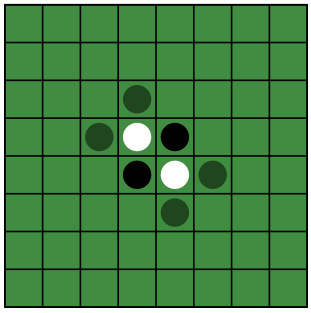

Figure 2: Possible first moves (translucent)

After placing the piece, dark turns over (flips, captures) all white pieces located on straight lines between the new piece and any existing dark disks. Thus, a valid move is one in which at least one piece is captured. The next figure shows the result of the dark player putting a piece in the topmost location.

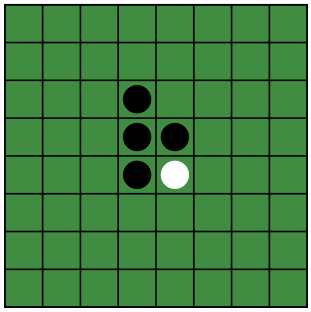

Figure 3: Result of dark playing in topmost location

The light player now makes a move, following the same rules but with the colors reverses. (Play a light piece adjacent to a dark one...) Figure 4 shows possible moves on the current board, and the result of taking the upper right square. Note that this sequence of moves leaves the dark player with three possible moves on their next turn, two of which will capture two tiles.

2.4 Passing

In standard Othello, if a player cannot make their move, play returns to the other player. You are not required to implement this, and may use a player being unable to move as the end of the game.

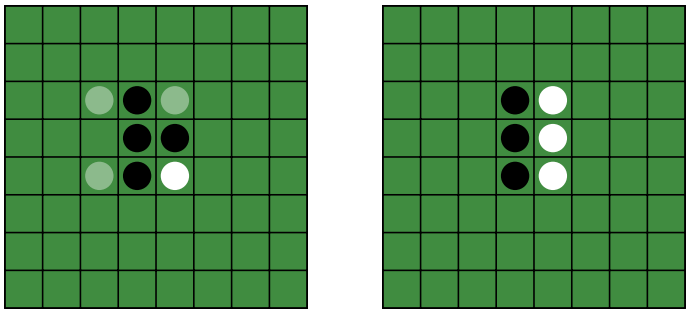

Figure 4: Possible moves for the white player, and result of taking the upper-right square

2.5 Ending the Game

Play ends when neither player can move, either because the board is full or no valid moves remain. (You may also stop play if a single player cannot take their turn.)

2.6 Scoring

A player's score is equal to the number of disks showing their color on the board. The winner of a game is the player with the higher score at the end of play – that is, the most disks showing their color. Games in which play ends with equal numbers of disks (almost always a 32-32 score on a full board) are considered ties.

3 Functionality Requirements

Your app must, at the minimum:

- Be neatly laid out and able to rescale for any phone
- Present a custom view which displays the Othello board and accepts taps
- Only accept legal moves
- Display the current score for each player
- Display whose turn it is
- Detect that the game is over and indicate the winner
- Allow the user to start a new game at any time

Your app need not:

- Support rotation
- Check whether a game is in progress before starting a new one
- Alert the player to legal / illegal moves, beyond not accepting illegal moves
- Have any particular color scheme, so long as your app is usable

4 Suggested Design

I would suggest structuring your app according to the Model-View-Controller pattern. This would involve creation of a class which maintains game state (whose turn it is, score, board state), a class that manages your view, and some support functionality to link them. As far as communication is concerned, your view class could then process a tap, figure out where it is, and send the move to the model, which would decide if the move is valid and update its internal board. The view class would then ask the model for the state of each grid cell (empty, dark, light) as part of its drawing method.

5 Submission

Generate a zip file as usual, and submit it via Caanvas.

6 Grading

This assignment is worth 50 points, broken down as follows:

10 pts	General Style / Compilation
05 pts	Neat, clear, scalable interface
05 pts	App displays current score and active player
05 pts	App supports starting new games
20 pts	App plays Othello correctly

You may earn some bonus points by going above and the requirements of the assignment, to a maximum of 5 bonus points. Here are examples of things you might do:

- Notify the user if they specify an illegal move (1pt)
- Display all currently legal moves (1pt)
- Confirm starting a new game if one is already in progress (1pt)
- Support players 'passing' their turn (i.e. if a player cannot move, return play to the other player). In this case, detect end of game by two consecutive passes, or a full board. (1pt)
- Implement AI (1-3pts, depending on AI complexity)

To be considered for extra credit, leave a note with your submission indicated what else the program does and how to use that functionality.

7 Alternate Games

If you'd like, you may implement a different game with similar heft. The general requirements are the same: you must have a rescaling custom view that accepts taps and allows the user to play a game. You must enforce the rules of the game, and display a turn indicator or current score as applicable. It is not required that your game be multiplayer, or, necessarily, a board game.

Games like Connect 4, Go, or Checkers would be acceptable alternatives. (A full implementation of the rules of Chess is more complex than intended here, but get in touch if you really want to do chess.) Tic-tac-toe is too simple. You may also implement puzzles like a tile puzzle (with at least 16 tiles) or sudoku. I have some existing sudoku puzzles you can use, if you would prefer not to generate your own.

Get in touch if you would like to do an alternate game and also earn some extra credit. This will be largely similar to the options for Othello, amended for the rules of your chosen game.