3110 Final Project: Othello

Benjamin Edwards bje43, Yassin Mziya ymm22, Chengxuan Bai cb674, Joseph Nechleba jdn64

Plan for Regular Status Meeting: During the design phase, we will meet bi-weekly Mondays and Fridays at 5:00pm in Olin Café. We will then meet every other day during the implementation phase on Mondays, Wednesdays, and Fridays in the same place and at the same time. We also have a group chat set up that we can use to communicate at any time.

Main Idea: To build the board-game Othello in OCaml

Key Features:

- The user can play Othello against another human or a computer player with varied levels of difficulty
- The user will play on a GUI representation of the game-board
- The computer player will be implemented using a Minimax algorithm

Possible Features *(*these are features we may implement given time constraints)

- A story-mode in which the user can play against a series of increasingly difficult computer players.
- The ability to play on different sized square Othello boards

Description:

Othello is a two-player board game played on a square board (size n x n square grid) using bi-colored disks. The object of the game is to maximize the number of disks on the board with your assigned color. Players take turns placing disks of their color on the board until the game ends. This happens when it is no longer possible for either of the players to make a legal move or until the board is filled. A move is legal if the player can place their disk on the board such that one or more of their opponent's rows of disks is bordered at each end by a disk of the player's color. A row may be made up of one or more disks. When a player places a disk on the board, the disk changes the color of all the opposing player's disks lying on any straight line between the new disk and one of the player's disks. When the game ends, the player with more disks of their color on the board wins. The user interacts with the game on a GUI representation of the Othello board. The computer player's various levels of difficulty will be determined as follows:

Difficulty 0: the computer player randomly selects one of its possible moves
Difficulty 1: the computer player looks two moves ahead with Minimax
Difficulty 2: the computer player looks four moves ahead with Minimax
Difficulty 3: the computer player looks as many moves ahead as is computationally efficient with Minimax