Anthony Zaccaria and Lexi Eastman CMSCI 256 B
3/22/23 Project 1

Dots and Boxes

The game we chose to code using a Table class is Dots and Boxes. We also utilized a Line class and a separate driver file. All lines a Line objects. The game board and the list of all possible lines are Table objects. The rules of the game are as follows. There are 2 players. The game, in our case for a reason discussed later, must be played on a 4 by 4 game board. Players take turns entering lines. If a player completes a square with a line they entered, they get a point. Turns alternate no matter what. In some variations, if a player makes a square they get to go again, but not in our variation. Whoever has the most points in the end wins. A line can only be a distance of one. For example and line cannot go from (0,0) to (0,2) since this is a distance of 2. Lines must also be vertical or horizontal, no diagonal lines are allowed. For example, a line from (0,0) to (1,1) is not allowed since this would indicate a diagonal line. There is a slight problem with the aesthetics of the game. It is difficult to tell what lines specifically are used by looking at the game board alone.

The gameplay works first by creating a coordinate system on the game board, which is a Table object. Each spot on the grid is a coordinate. The top leftmost spot would be (0,0) and the bottom right-most spot is (3,3). The first coordinate is the vertical location and the second is the horizontal location. Players enter the coordinate for the start of the line and the coordinate for the end of the line. This input becomes a Line object. Their input is protected using conditionals and try/except. If a player enters a line that does not fit the rules or is not possible, they lose their turn. Players can quit by entering q when prompted to enter the vertical location of the start of their line. The program can tell when a square is completed through the countSquares4by4()

function, which uses conditionals to determine how many squares have been completed so far in the game. This only works for a 4 by 4 game board because the method focuses on 4 by 4 grids and does not account for any other size boards. The turn() method runs the gameplay of each turn. Within it is contained the newLine() function which gets the user's input for the line. All of the lines that are input throughout the game are stored in an array named lines. This is displayed for the user each turn so they know what lines have already been used. There is a used() function that determines if a user entered a line that has already been used, if so then they have to reenter a line. After a user enters an unused line, the updateTable() and changeSymbol() functions are run which updates the gameboard according to the user’s input. The scores are then each turn based on if any new squares have been completed, determined by the countSquares4by4 function. The countSquares4by4() function depends on all the possible lines that can be entered on the specific game board, generated by the genPossLines() method. The possible lines is a Table object.