ConnectFour Design

Manaar Hyder, James Lee, Maya Ramamurthy,

Randa Mohsen, Danielle Shwed

Comp Sci 2ME3, Group 2

# Classes and Modules Used

## ConnectFour Class

Module()

## CheckWin Class

The main purpose of this class is to serve for a way for the game to check if the user has won based on the moves that have been made in the game as of yet, by using an array of given locations to figure out if there currently is four in a row through a series of tests. This was placed into a class of its own for the purpose of determining the win without having to be placed in the same class as the user interface, however due to the interaction, it is not able to be kept completely private. The methods that interact with the ConnectFour class are left public to allow for this interaction, however the remaining methods are not.

duplicate ()

This class interacts with the checkWin () method, in which it receives the input of the positions of arrays, with the format of three possible values stored in it to mark the colour of the counter if applicable: 1 if it is a red counter, and a 2 if it is a blue counter; 0 otherwise.

Using this, it will create a new array of the same size and type (integer), which will now contain a 1 if it is red, and a -1 if it is blue; 0 otherwise. This is done through two nested for-loops to easily access index positions and to allow easier access when writing in the positions.

The purpose of this method was to provide an easy way to quickly total up the value of four positions in a row to see if it added up to a sum of 4 or -4, in which there would be a win, all without having to modify the original array of positions, which would be trickier to use when it came to checking the positions, as two blues in a row would equal 4, which would throw off the calculation of the game. It also would be far less efficient if we simply checked each time if the positions array at a certain index contained a 1 or a 2 and added a 1 or -1 to the array used to check the wins, thus why we made this decision.

This method is private to keep it hidden from the user, so it is not accessible from the user interface class. This will ensure the array cannot be tampered with from the outside and will remain intact due to this reasoning.

getPos()

The purpose of this is to be something to interact with the ConnectFour class once it has checked to see if a winning move has been made. If this is the case, then it will return the winning positions obtained through both the setPos() method and the checkWin() method. This will then, once called upon, deliver the array, which is later used to draw the dots on the screen to show that a win has been made. It has no input variable, but receives the array winPos which is an array of coordinates of any winning moves, which is the output for this method.

**setPos()**

The purpose of this method is to obtain the positions array found in the checkWin() method, and return it to the getPos() method which can use it to find the winning moves, as checkWin() merely returns if either the red player or blue player has won, or if the game is still in progress. There is no or output for this method, as it merely sets the variable winPos to the input variable win, an array which receives it’s input from the checkWin class after determining that either red or blue has won.

checkWin()

This is the method that determines if any winning moves have been made. This works by using the input of the array of filled positions, and in doing so, calling upon the duplicate() method to receive the array of 1 and -1 values to figure out if there is a winning move being made. There are four possible ways for there to be four in a row: by rows, columns, left diagonal, and right diagonal. We decided to divide it up this way to check systematically if a winning move has been made, and if it has, automatically return the winning player, as opposed to checking all possible combinations and returning it at the end.

This has the input of the array of positions to calculate the wining moves, and will return the variable total, which will either be a 1, if red, or a 2 if blue. If no win has been calculated at the end, it will return a 0 to show that the game is in a draw state.

## currentPlayer Class

# Public Entities

## ConnectFour Class

Module()

## CheckWin Class

## currentPlayer Class

# The Uses Relationship

# Private Entities

## ConnectFour Class

Module()

## CheckWin Class

## currentPlayer Class

# Review Of Our Design

# Testing