# Noughts and Crosses Report

## Introduction

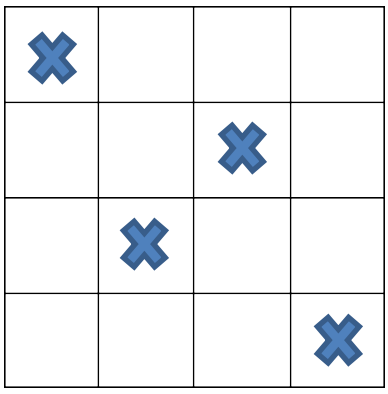
For this project, a Noughts and Crosses player for a 4x4 grid has been created along with a game of Connect Four using functions from a Noughts and Crosses game. The functions used have also been altered slightly so to accommodate for some of the elements of the game to function.

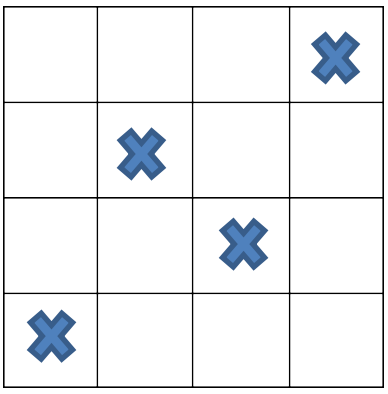
## Changes to GameRunnerV3

The main change made to the original GameRunner is having the player and number of rounds given as arguments on the command line. Giving no commands will ask the player how many games they would like to play before playing it against the ExpertPlayer. If only the number of rounds is given on the command line, it will play the ExpertPlayer first against the HumanPlayer. Putting in only one of the players will pit that player first against the human player, for example, if the arguments given are "10 SimpleComputerPlayer" it will play 10 games with the SimpleComputerPlayer vs the HumanPlayer. If the number of rounds is not given but the players, an error will be given as the code checks for an integer. The players will alternate turns at the end of each round.

## ExpertPlayer

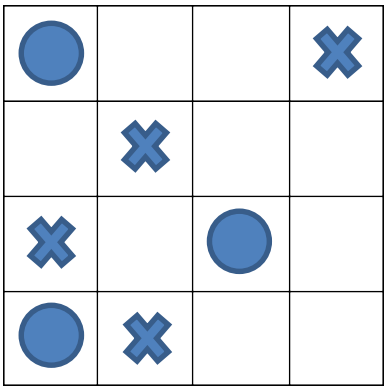
Since it is possible to draw in Noughts and Crosses, the best a player can do against an opponent with perfect information and makes no mistakes is to draw. This player aims to do this by playing a strategy in which it will play to block the other player from winning in as few moves as possible. This is done in two different ways by only using 4 moves:





These two layouts block every possible line from being taken and so by a player taking these posistions they are guaranteed not to lose so this player makes it their priority to take these posistions. If the player goes second and the other player goes in the top left corner, this player will switch to using the other corner and follow the second plan. There are also parts of the plan if the

other player goes in the opposite corner to the first chosen corner, the ExpertPlayer will choose a space which is empty on the two adjacent lines in order to block the opposing player.



An example of said situation with the corner taken by nought. The player will also fill out the bottom row.

ExpertPlayer will then fill out any of the center squares before moving to check if any of the corners have two of the same symbol and filling in the squares which are empty between them before using the algorithm used in SimpleComputerPlayer to fill in any empty squares but it rarely gets that far.

One of the other functions used in the ExpertPlayer class is the checkLines function. This functions checks all the lines in the function and counts the number of noughts and crosses in each line. If there are three of either symbol in a line, it returns an integer depending on if its 3 crosses or 3 noughts. The beginning of the main function in the ExpertPlayer uses this and checks for an empty square in the line and returns the grid coordinate and saves that along with the integer in the ArrayList listOfSquares. The player then checks the ArrayList and first checks if there are lines with 3 of its symbols (there is an obsolete method in the code which wasn't removed) and choosing that line so it wins the game. If not, ExpertPlayer will then block the other player's line. ExpertPlayer will make this check at the beginning of every turn.

When following these instructions, the ExpertPlayer will beat the SimpleComputerPlayer 100% of the time. It will also never lose to the RandomComputerPlayer and wins ~60% of the games.

## Connect Four

For the extended project, a game of connect four has been created on the gird using elements from the original game. I've added a few new functions to NCGridV3 as well as a new class to run connect four. The new class is very similar to the original GameRunner. The changes I have made include a change to the variables when calling the grid and the calling of a new function for getting the next move for connect four.

One of the main changes made to the NCGridV3 class is the change of the gridSize into two different variables, maxRow and maxCol. This allows me to use a grid that doesn't have equal sides as is in connect four. This means that the variables for most of the functions in NCGridV3 have had to be

changed as both the number of columns and rows were the same, namely gridSize has had to be replaced. A function I have added is the setSquareConnect4 function which takes only the column from a player's move and puts it in the lowest possible space on that column.

Because of the new dimensions of the grid, a function has been added for reading all the possible lines on the grid. The function starts by reading each column before reading both diagonals from each column. It then moves on to going down rows before getting the diagonals from the two ends rows. This function calls the modified checkForLineOfN, getGameStatusConnect4. The change has been to see if 4 of the same symbols have been counted and returns a game win for that player. Connect four cannot end in a draw so the draw enumerator has been removed.

## Conclusion

The main issue in this project was finding the algorithm for making the computer player that never lost. Choosing the initial squares and finding the positions where you were guaranteed not to lose were easy but it was hard once the other player had chosen one of those squares as well and there would be many situations where the ExpertPlayer would player 100000 games against the RandomComputerPlayer and lose 1 or 2 games. Also implementing a computer player for the game of connect four was difficult because there were a lot more factors to consider such as both sides of a line of three could be taken and would result in a winning game. Also adapting my functions from ExpertPlayer into the ConnectFourPlayer proved challenging because of the new dimensions. Because of these issues an AI wasn't created for Connect four which would been competitive against the RandomComputerPlayer that wasn't just going vertical in the middle row (which won 90% of games).