How to Run:

Simply open up the index.html file in your favorite browser. (Tested on Chrome) You will see buttons to run the different games, as well as a speed input. Speed controls the amount of time between turns. You can run bulk games by clicking the bottom row of buttons. The board output during bulk games will not be displayed, but you will see the aggregated results.

1.) Architecture

There is a “Game” object. This game object instantiates the players, keeps an internal representation of the game board, draws to the game board, controls the turns, and checks for winning moves or draws.

Then there are two types of player objects. One is the Naive Player, the other is the Thoughtful Player. Either can be passed into the Game, which will assign them an ‘X’ or ‘O’. The commonality between them is that they both possess the *move* method. This method is called by the game on their turn, and it returns a board position determined by some rule internal to the player.

Naive agents just randomly choose open board positions. This seemed like the most naive I could get, but it could also potentially get lucky.

“Smart” agents have a series of rules that they run through. If the first rule is not met, then it moves on to the second, and so on.

The rules I made for the “smart” agent are the following, in this priority:

1.) If the player can make a winning move, go ahead and take the win.
2.) If the other player could make a winning move, block them.
3.) If you have a piece down, and you can win in a direction (vertical/horizontal/diag) off that piece, place a piece in an open space on that vertical/horizontal/diag
4.) Move at random. (Last resort/Opening move)

These rules struck me as the most important to keep in mind while playing Tic-Tac-Toe. If you can win the game immediately, obviously it is ideal to do so. If you can’t, and your opponent can, then it is clear that to prevent losing you must block them. Also, you should build towards victory by placing pieces on an adjacent row/diag/column as another one of your pieces, if that row/diag/column has a chance of victory.

I evaluate these rules via a “scoring” heuristic for each row/diag/column. Blank spaces do not contribute to the score, your pieces add 1 to the score, and your opponents pieces take 1 away from the score. Rows/diags/columns with a score of “2” with respect to your own pieces, for example, are rows that are ripe for winning, satisfying rule 1. Rows/diags/columns that have a score of “2” with respect to your opponents pieces, however, are rows that you must block, matching rule 2. Rows/diags/columns that have a score of 1 are rows with one of your pieces, and none of your opponents. These rows can contribute to your victory. These rows satisfy rule 3.

2.) Results and Analysis

Naive vs Naive: A fairly uninteresting matchup. X tends to win first, simply because it moves first. By moving first in a 9 turn game, you end up having one more turn than your opponent, increasing the chances to win slightly if you are picking at random. Out of 100 games, I received the results 19 draws, 53 wins for X, and 28 wins for O.

Naive vs Smart: Smart almost always wins. Sometimes there is a draw if the naive agent gets lucky. I’ve seen Naive win as well, rarely, but it comes down to making a specific sequence of moves that ends up trapping the Smart agent in a no-win scenario. This was unexpected, but makes sense as I review the move sequence. The smart agent does not perform any look ahead, it simply applies rules to deal with the situation at the moment, so it could be trapped by a lucky set of moves. Out of 100 games, I received the results 10 Draws, 6 wins for X (Naive), and 84 wins for O (Smart)

Smart vs Naive: Smart almost always wins. The naive agent can almost never make enough moves to bumble into trapping the smart agent. The smart agent destroys the naive agent mercilessly. The naive agent must be extremely lucky to make it past the first 5 moves. The results for 100 games were 2 draws, 97 wins for X (Smart), and 1 win for O (Naive).

Smart vs Smart: Many more draws here. A decent amount of wins as well. O tends to win more than X. As I looked through the move sequences, it looks like in the process of blocking each other, one agent may “trap” the other, leading to an unblockable scenario for the opposing agent. This is not an intentional goal of the smart agent, however, and was surprising to me. For 100 games, the results were 49 draws, 17 X Wins, and 34 O Wins.

To improve the performance of the smart agent, I added a rule to prioritize the center, which will nullify most traps. This rule was added to the chain before “random move”. After adding this rule, I saw the gap between *Smart vs Smart* wins get smaller, to 69 draws, 10 X Wins, and 21 O Wins. In *Naive vs Smart*, I saw a decrease in Naive wins with this new rule, with results of 11 draws, 1 X Win, and 88 O wins.

Then I added a rule to prioritize an open corner if no center was available. This was added to the rule chain after “prioritize center”. After this rule, the *Smart vs Smart* games became mired in stalemate, with draws shooting up to 80, no X wins, and 20 O wins.