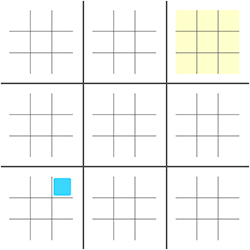
Ben Borders & James Gregory

C291 Final project Report

April 29th, 2016

We decided to do our final project on a mathematically interesting game called Tic-Tac-Toe-Ception. Tic-Tac-Toe-Ception is basically exactly what it sounds like. Normal tic tac toe is completely solved, and also completely boring, there are ways to always win or at least draw. However with Tic-Tac-Toe-Ception, hereby abbreviated as TTTC, there is no telling who will win and with what strategy. In each square of normal TTT, there is a whole game of TTT. So, TTTC consists of one big TTT game, that is 9 TTT games, one for each square. The basic rules are the same. If you get three in a row, horizontally, vertically, or diagonally in any one TTT square and that square counts as the tile that won, so if X gets 3 in a row on a TTT square then that square counts as an X in the overall big TTTC board.

The most important rule is that a player does not always get to pick which small board they play on. That is determined by whatever position the previous player played. Where the previous player played in a small board is where the next player will play in the big TTTC board. So if O plays on any small TTT board in the lower left hand square, then X has to then play in the lower left small TTT board.

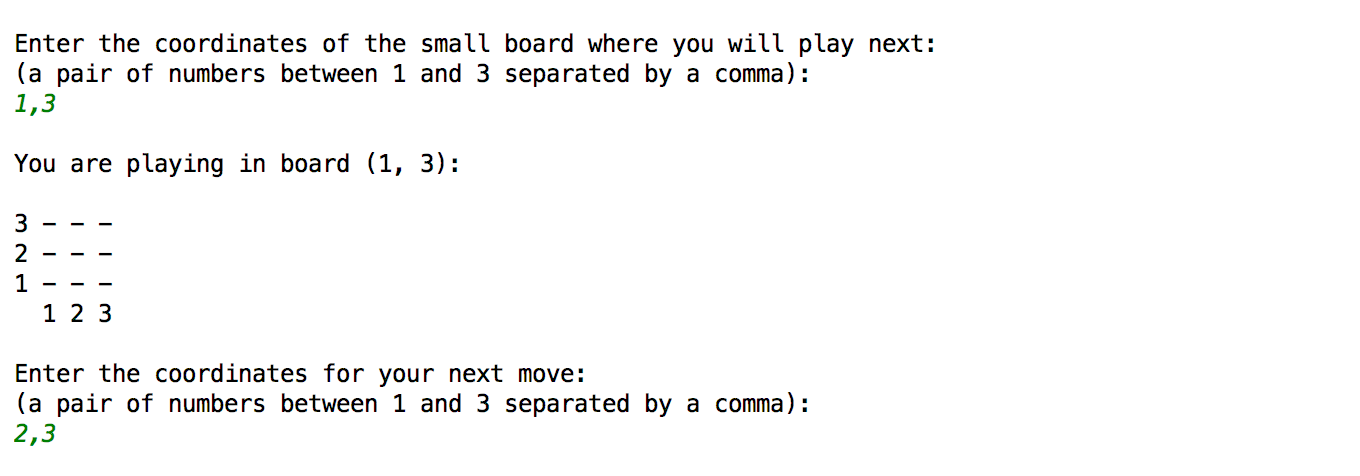


A player is also not allowed to play in a small board that has already been won or drawn, this is so that the other player cannot force the next player to keep playing in already won boards, which is too strong of a strategy to neglect. The only time a player can pick which small board they can play on is if it is the first move in the game, or if the opposing player sends them to a board that has already been won. We also made it so that a drawn or “catted” board counts for neither X nor O, to make it a bit harder. We wanted the game to be as impartial as possible.

As far as we know, this game has not been solved, nor are there any overwhelming algorithms or strategies that greatly affect who wins the game. There are many ways to play this game and many different strategies, but none grabbed our attention too much. At first we thought a good strategy would be to force the opponent to play in undesirable small boards, so the middle edges. However, this would mean you would have to play in the middle edge of the small board, which was also undesirable. There was no strategy that did not also have its downsides. We could also have played in the corners and middle of all small boards, but then this of course would force the opponent to play in desirable positions in the big game.

Overall, we figured instead of hardcoding a heuristic we would use a more foolproof technique that did not rely on any set strategy There aren’t many hard coded strategies I can think of that would improve the AIs decision, except weighting 3 in a row more than 2 and basics.

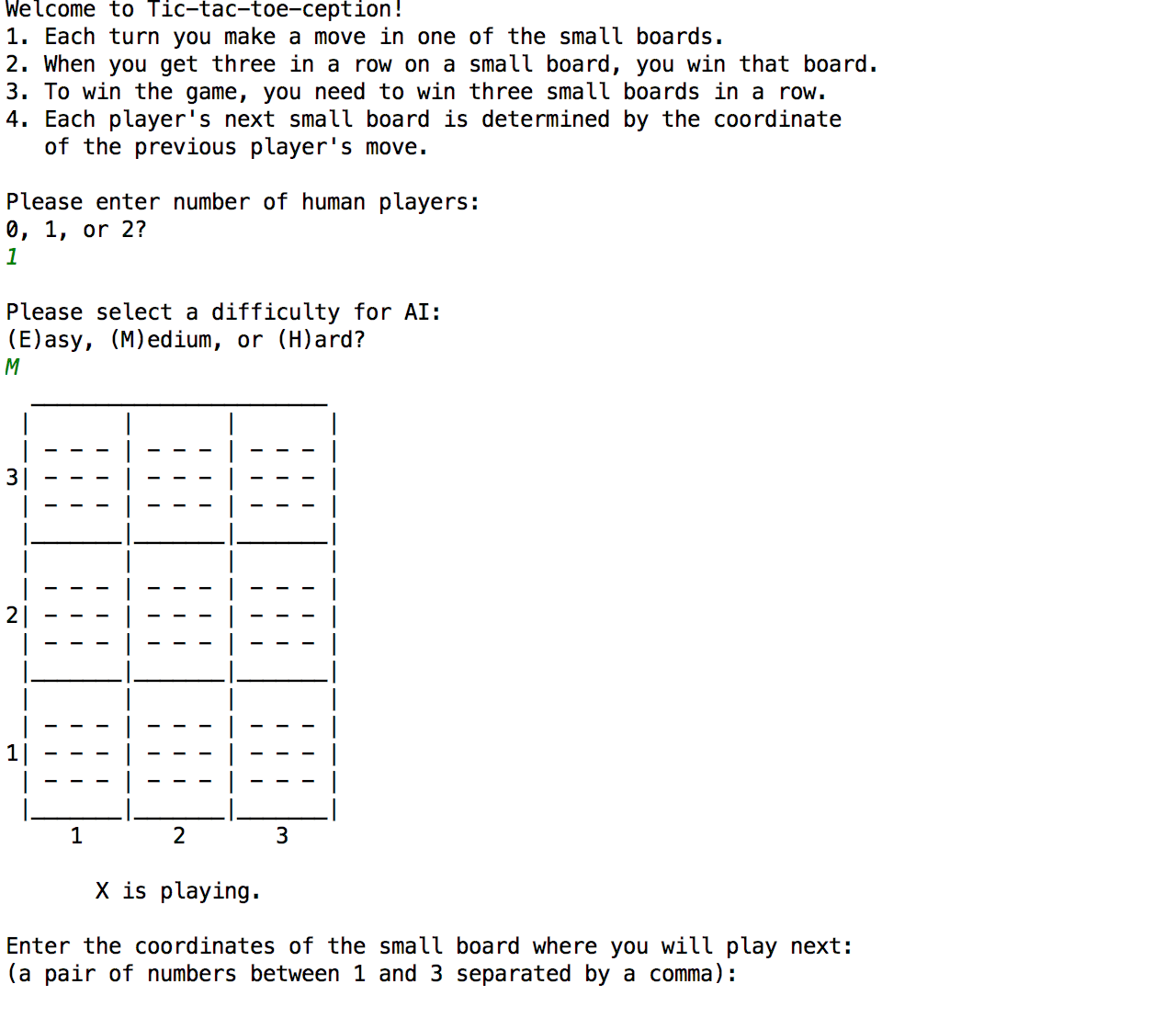
We represented the Tic Tac Toe Boards as dictionaries with keys as (i, j) pairs where i and j are both integers between 0 to 2 inclusive. So the large Tic Tac Toe Board dictionary would take these pairs as keys and return the small board dictionary corresponding to the coordinates passed to it. The small board’s dictionary would also take these (i, j) pairs. However, each small board starts off empty. Once a player makes a move on that small board, a corresponding pair key is added, assigned with a value of either 1 or -1, depending on which player played.



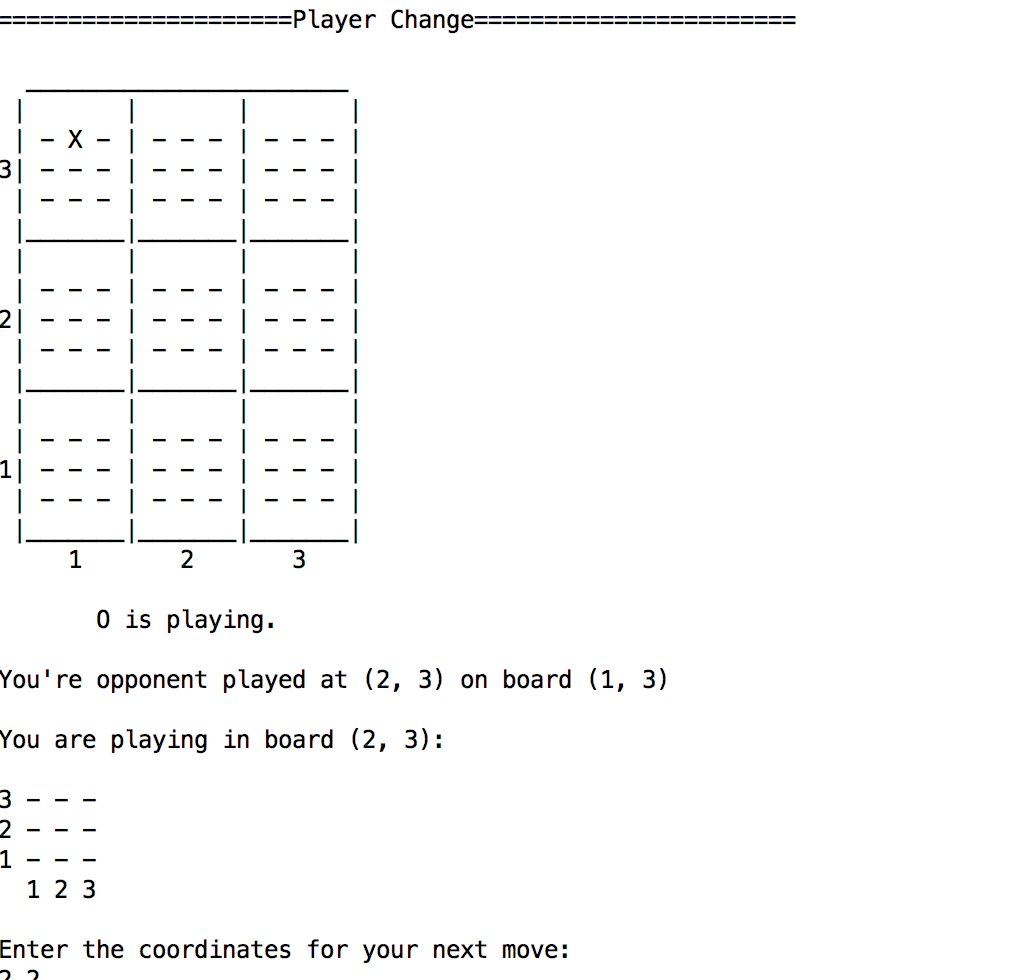
We used three different AIs for our project. We gave the user the freedom to choose whether he wanted to play versus another human, against an AI, or just watch 2 AIs battle each other. Within the AIs, we had three difficulties they could choose from: Easy, Medium, or Hard. The Easy AI was not sophisticated at all, and we didn’t want it to be. The easy AI simply enumerated a list of all available moves and chose randomly from the set and played it, so it was completely blind and dumb. Our medium difficulty AI is where things got a bit more sophisticated. For this, we used the Uniform Monte Carlo technique. To keep it from getting too difficult, we had it limit its method to the small boards only. For example: for each move, we would simulate a number of games played using that move and how many times it would win that board, not win the entire game. So the move that won the most simulations in the small board was the one that the medium AI chose. Our hard AI did the same thing, except it considered winning the ENTIRE board, and also it simulated itself against the medium AI.

Our results were great! Our Medium AI always beats the easy AI, and our Hard AI almost always beats the Medium AI. The only case where it hasn’t in the past has been when it goes second, as going second is still a disadvantage as it is in normal TTT, but we fixed it and made it more intelligent. Easy plays instantly, as is expected from a random move generator. The medium AI takes more time the more free space the board has because it has to generate more simulations, as does the hard, but neither take more than a few seconds each move.

To illustrate the game to the user so that it is clear what is going on, we draw the board. We went through a lot of stages in design, as we thought it was one of the most important parts in creating a usable and replayable game. The board is redrawn after every move, so that the players are always updated with the most up to date information.



First, if the player is to choose a small board to play on, we print that out and ask for coordinates, inputted in a user friendly format, (1-3, 1-3) instead of (0-2, 0-2). Then, once we have the information for which small board is being played on, we print out the small board as it is currently, and the user gives out an x, y, coordinates in the same format as before.



We learned that this game is pretty fair and it’s hard to have a specific strategy, though one that we did come up with was to focus on winning the boards in the corners and middle. At first, we also preferred to send our opponent to the least desirable small boards. Eventually, we realized that in doing so, we were setting up our opponent to block any board we had a winning position on. The most recent update to our hard difficulty was to, instead, send its opponent to the most neutral board. This would put our opponent in a position where they could only play the least offensively oriented move while also not allowing them to make any notably defensive moves.

We also learned about the usefulness of the Monte Carlo technique, especially when no strategy is completely apparent. It makes sense to utilize the speed and processing power of a computer to simulate many games and pick the best one. Another good amount of what we learned was Python, since neither of us were exactly experts in the language. We learned about the immutability of tuples, how python values are passed, and the importance of deep copy.

”Great job, guys. A+!”

