Jacob Dunning

CS2

Individual Summary

In this program I used several of the different coding techniques discussed in class. I created a Board class with private data members, a constructor function, and various behaviors. I used a recursive function to search through the entire tree of possible games to find the highest scoring moves. I also used file output to generate a txt file showing the final outcome of the tic-tac-toe game.

I had to research various algorithmic approaches to solving competitive strategy games. I thought I could use a minimax algorithm to find the best AI moves assuming the human player makes the strongest moves. The implementation of this program proved too difficult, however. I also researched more about the use of recursive functions to know if searching through the tree of tic-tac-toe games would prove to be computationally burdensome.

I was most surprised by how difficult it is to program an AI that always plays the best move, even in a game as simple and predictable as tic-tac-toe. I thought that by assigning scores to terminal outcomes (cat games or three in a row), the computer could anticipate the best human moves in response to its moves and would therefore be able to make the best move. Implementing this algorithm turned out to be far more challenging and complicated than I anticipated.

If I were to design this program again, I would do several things differently. I would have designed the Board class with different (and fewer) member functions, as some of the functionality could be simplified. Although I still think that the minimax approach should work, I would have to dedicate much more time to figuring out how to avoid its pitfalls. Also, I would better include a low depth preference in the AI algorithm. It should strongly prefer winning moves that come earlier in the tree of possibilities because it has the potential to miss winning moves because alternative moves have many winning terminal outcomes, but those alternatives may not guarantee a win.