Good morning

Today I want to present my Terminal Application and what I’ve built so far.

Here’s a quick presentation agenda: I’ll quickly go over the features of the application (2min)

I’ll do a live demo of my application.

And then the rest of the presentation I’ll be spending discussing the Application logic, the code, and the design process

Okay.. so … some of you may know this already.. But I am buiding a TicTacToe Game that has an unbeatable AI.

Now I’ve been a hobby game developer for a long time.. And we were Tic Tac Toe in class the other day and it inspired me to do my own version of TicTacToe.. where not only can you play with another player but also play against the computer

And I wanted to create something where the computer actually responds intelligently and tries to win…

So here are the features of my application:

There’s a single player and 2-player mode

Simulation Mode which I’ll discuss a bit later on

Instruction manual

Ability to save games to a file

Load previous games from a file

And last but not least, Unbeatable AI

So there was my first problem…

Whenever you’re creating a turn-based game or anything, In computer science there is something called Game-Tree Complexity.

It basically refers to how many possible moves are there after an X number of turns.

For example in Chess, if you try to look 6 moves ahead, that’s about 120 million possible combinations… And the number of possible Chess games also known as Shannon number is something like 10123 which is a number so large that it’s ridiculous

But TicTacToe has an advantage… it’s much smaller…

Now why is this relevant?

Do Live demo

Now I just wanna walk you through my code

It’s easier to show you in VS Code, but when you run tictactoe as an executable, the first thing it does is run command line arguments.

The game only runs if there are no command line arguments.

There are 3 classes and 1 module.

The one I wanna talk about is the computer response. To program the computer response, we have to go far back. To when I first started the project.

I was thinking how can I get the computer to spit out a value? How does the computer know which square to play?

So what I started with doing was just picking an empty square at random and just playing that. Just initially.

Here you can see a representation of the functions.

BTW, the rand() function I found that to be quite redundant because I found a much shorter and quicker way of doing exactly that and it’s by using Array sampling.

Here’s another function I wrote. This is what lets the computer predict whether the next move is going to be a win or a loss.

The way it does this is by taking in the board as an argument. Then it creates a duplicate of the board, then iterates through the board assigning a value to each square and checking if that leads a win.

Taking it beyond to Algorithms..

Okay… so.. so far my program was working quite okay… It was spitting out random values and was accurately predicting whether it was about to win or lose. But it still wasn’t good enough…

I wanted to create something that was unbeatable.

So I decided to look up algorithms…

I didn’t just want the computer to just spit out a random move… I wanted it to play intelligently…

So I used the Minimax algorithm…

What this basically does it creates a sort of tree… A tree of all possible moves and picks the move that will lead to a win…

So let me just demonstrate what that looks like here…

So lets say for example we have this board…3 X and 3 O

And it’s X’s turn to play

Now how does X know which square to pick?

So what this algorithm will do is find the empty squares on the board

Then it will iterate through each of them

X’s turn to play… Just checking to see what will happen… win or draw

O’s turn to play… same deal..