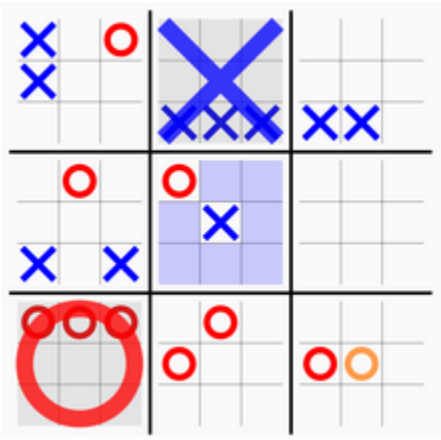
Jacob Hummer | Project proposal for COMPSI 170

Introduction

You know the default project of making a Tic-Tac-Toe game? I want to make Ultimate Tic-Tac-Toe https://en.wikipedia.org/wiki/Ultimate_tic-tac-toe complete with a GUI. The final product should look something like this:



This project could involve:

- Using a GUI framework
- Installing Python packages
- Using data structures to store the game state
- Receiving user input somehow
- Figuring out legal moves from the game state
- Saving stats in the filesystem somewhere

I know enough to know that there's a requirements.txt manifest-like file and a project-local venv that you need to set up to use Python libraries without installing them globally. I know that PyGame exists. I feel reasonably confident that I can figure out how to use some kind of GUI library and get something to show up on it. Getting the game logic 100% is going to be the tricky part. Adding the fancy stuff like a menu or a pause screen or a win screen or whatever is probably a bit much to ask. I **do not intend** to implement any AI bot player system. It is intended to be a two-player game.

Minimum viable product

1. A GUI that appears
2. 3x3 large Tic-Tac-Toe board
3. Each 3x3 large Tic-Tac-Toe cell has its own 3x3 mini Tic-Tac-Toe board
4. When the player attempts to place their symbol on a square it responds
5. Game detects the winning state and responds

Note that "game prevents players from making illegal moves" is not on there. The stretch goal is to somehow compute available moves and only allow legal moves to be made by the player.

Stretch goal

The stretch goal is to accurately implement the actual “what is a legal move from this board position” logic. That seems like it will be very complicated. The GUI stuff seems *relatively* easy. You “pip install” PyGame or something and draw some crude boxes on the screen and call it a day. The game logic seems like the biggest challenge; hence it is the stretch goal.

Topic coverage

✅ Various data types
Intend to use strings and numbers. That’s the basics. I’m sure that whatever library I end up using will also have me use some classes and objects and such. Who knows, I might even create my own!

✅ Input validation (not allowing irrational inputs)
Not CLI input per se, but I will need to detect whether or not the user is clicking on a blank space or on someone else’s existing symbol (and thus they can’t play there).

✅ Conditional statements (if/else)
Definitely. Checking if the game board is in a “someone won” state is an if check.

✅ Loops (while/for)
Yep. Looping over each big cell to render the mini 3x3 game inside it will be a thing.

✅ Lists
Yeah. There’s going to be a list of lists of game positions somewhere.

✅ Modular programming (defining functions)
Good idea. Functions will be used to stuff all the hyper-specific rendering code into procedures to avoid cluttering the pristine logic part of the code.

✅ File I/O
Want to stash the global stats like “you have played N games” or something into a file so that it persists across game restarts. The PyGame (or whatever GUI framework) probably also has a way to show images so that could count as “loading a file” (albeit through a library)