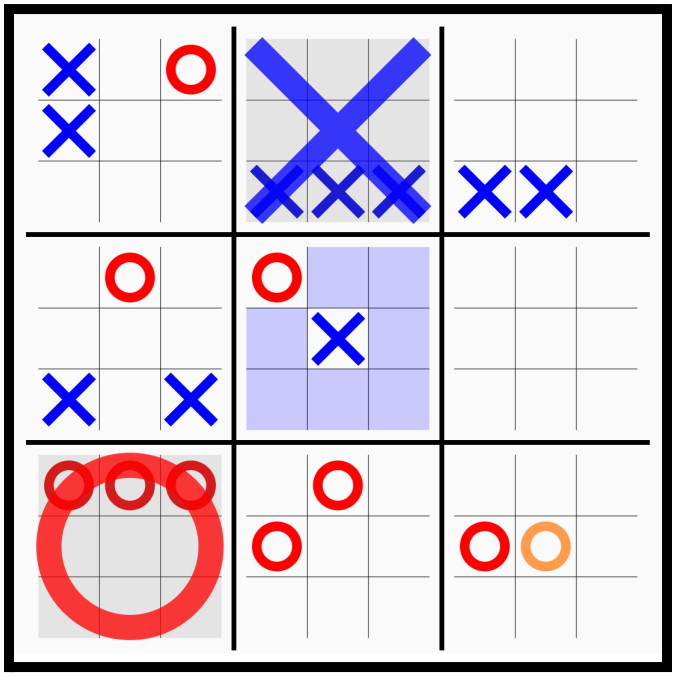
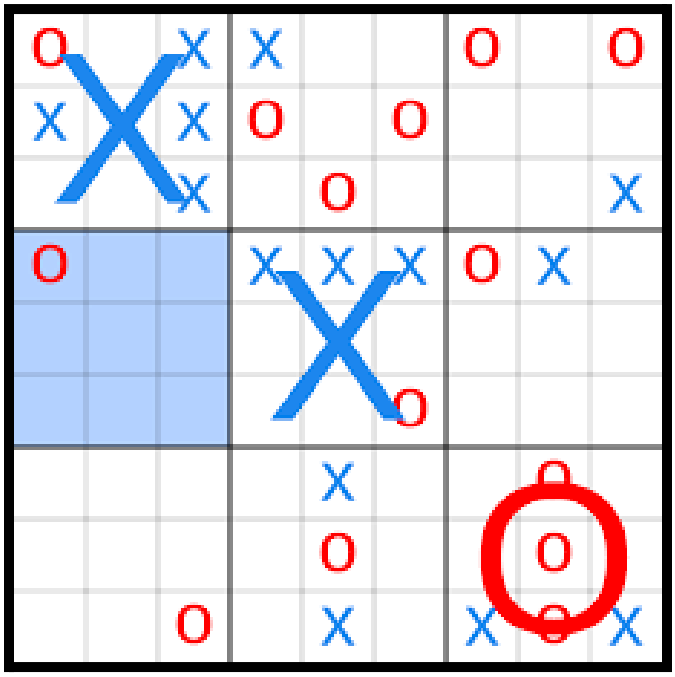
Ultimate Tic-Tac-Toe Specification

End Product Example:





Description:

- **Game:**

A 3x3 master grid, where each cell itself is a 3x3 grid tic tac toe board. Players alternate turns placing Xs or Os in any of the 9 tic tac toe games. Winning a ‘subgame’ results in the master cell for that game converting into a corresponding X/O. A player wins when they connect 3 of their type in a horizontal, vertical, or diagonal line on the master grid. A tie occurs only when **all spaces on a grid are entirely full** and a winner is not determined, in which case a T symbol should be drawn on the cell. This same logic applies to the master board as well.

- **Navigation:**

There must be a start screen to select which player (X or O) goes first, and a game ending screen where the game winner, or tie is displayed and restart(go back the the start screen) or quit option buttons are presented

- **Style:**

Full freedom to implement the game described above is given, as long as the core mechanics are clear and the game can be won, lost, or tied.

Library usage and some advice:

pygame:

This game will be implemented in pygame - a python library used to draw things onto a window and handle events such as clicks, drags, scroll etc. Learning to use this means you will need to be able to read through some documentation to find out how to do things, such as registering the mouse clicks, drawing shapes and lines to the screen, flipping between screens to actually display such as the start screen and the game screen etc.

Below are some links to get started with pygame. Before you get started I have some advice for intelligently parsing through code library documentation in general as it can be overwhelming at first and sometimes difficult to find what you are looking for/even know what you're looking for in the first place.

As with all projects, concepts become much easier when you break things down into simple digestible parts. As an example - consider this project here. The basic structure of the game board is simple ***lines.*** Looking at the big board grid at first can be confusing but breaking it down into just some lines makes things more manageable. Now in terms of figuring out how to do that - looking on the documentation page there's nothing about drawing lines or grids. Hm, we need to become more granular than just lines. We are drawing something onto our game screen. ***Drawing***. There is documentation pertaining to a “draw” functionality for pygame - perfect place to start looking. And within the page for draw functionality you will discover functionality to do what we wanted - drawing a line! From here you can look at what it takes to draw a line and even some examples of it being used. The main takeaway from this example is that by breaking problems into subproblems, and sometimes even those subproblems into even smaller problems, you can find some foothold to start. Reading documentation and experimenting is where most developers spend most of their time when working on any project.

Installation guide for the pygame library so you can use it when running your python can be found here. It is installed through pythons own package manager called ‘pip’:

- https://www.pygame.org/wiki/GettingStarted

The documentation for pygame can be found here:

- https://www.pygame.org/docs/

Codebase:

main.py - the entrypoint:

All software has a ‘main’ function that acts as the organizer/loop and pulls in the functionality from everywhere else in the code. These files/main functions are known as the **entry point.** This project's entry point has been provided as *main.py.* It has been filled out as a template with the basic pygame loop that you can find on the documentation's main page. Here is where you’ll instantiate a game manager and give it a master game board to take care of and manipulate it/draw it to the screen. It’s also where you’ll handle all the events that can occur in your pygame window. Keep things here as clean as possible - this is the ‘front door’ of your project codebase ‘house’; you wouldn't put your kitchen, bed, or toilet right at the front door of your house, would you? Some extremely large and successful projects have entry point files that are under 15-20 lines of code long - they keep it clean.

class organization:

I have defined some classes to help you organize yourself in the project. Each class has been defined and placed into its own .py file for clean code structure:

GameManager class - master hand:

The game logic controller - this is where to implement how the game functions and what/how conditions need to be met. All of this is done by manipulating a MasterBoard object that you will give the manager to control.

MasterBoard class - big picture board:

The bigger picture board should - it should contain a 3x3 grid of GameBoards and any functionality that the bigger board needs.

GameBoard class - single tictactoe board:

A basic tic-tac-toe board - it should contain a 3x3 grid of Cells and any functionality that a basic game board needs.

Cell class - individual cell:

The smallest unit of the game - the cell is what the game board is made up of and only needs to do a few things such as keep track of if it has an X or an O in it, and if it's already been played on.

Sidenotes:

I have defined this object structure because I believe it is simple to follow and lets you compartmentalize functionality into smaller pieces. Feel free to deviate from this structure if you think you have a better implementation or just don't want to use this structure. Below is an image that depicts how the objects I laid out relate to each other: