Assignment 5

Our code is mostly original except for the reinforcement & self learning part, meaning that most of the code is written by members of this group. The only things that aren't created by us are the API and libraries we imported from Python and Pygame. For example, on twitch/main.py, the code is written by Kedyn but we imported the pygame pyautogui library in order to implement it into our project. The other component that wasn't originally ours is the self learning that is based off of MENACE (Matchbox Educable Naughts and Crosses Engine) proposed by Donald Michie. We have provided a README.md file within our self-learning folder on our Github if you would like to learn more about it.

First, we created the framework for this project by setting up a director file and scene file that would handle how our game would be played. This meant including the controls and the rendering of our board. Second, we created the main parts of the program which included the AI, the board, the game scene itself, the menu, the pieces, and the player. The AI file would handle how the AI behaves through a minimax algorithm and the board would provide all of the components we would need to actually run the game. The minimax algorithm was created by ourselves using the knowledge of heuristics that we acquired through CPSC-481. We came up with the solution that whenever the AI wins, then we would assign a heuristic value of 7 and a -7 when the AI loses. The game scene file would handle the instances of the game where you are playing against the bot, and handle all of the win/loss/tie conditions. It would also play sounds and display messages according to the right conditions. The menu scene would handle the menu that shows up when the player starts the game and renders it.

The settings.py file within the Twitch folder is used to configure how we send data to our game using the Twitch.tv chat. We're not actually using any Twitch libraries, but instead we are using Twitch's socket to send data to. All you would need is the URL from Twitch, an "oauth" password, and your twitch channel name. Once we specified all of these fields within our code, when we run the main.py file, the Twitch Chat that's logged under your name will be able to send commands to the tic tac toe program that's running on your computer.

Finally, the game_test.py file was for testing purposes without having to run the twitch connection. main.py is the file that we would run officially when we are presenting. If you are running the main.py, you have to make sure you are on the correct twitch channel that is on your settings.py, otherwise you will be only be able to send commands directly to the game and not through twitch chat.

- Baseline Score: 90%
- Contribution
    - Ian Alvarez
        - 25%
    - Kedin Macedonio
        - 25%
    - Francis Nguyen
        - 25%
    - Joesh Bautista

        - 25%

- **Readme**
    - Make sure you have all of the source files from the Github page provided here:
        - https://github.com/Kedyn/tictactoe

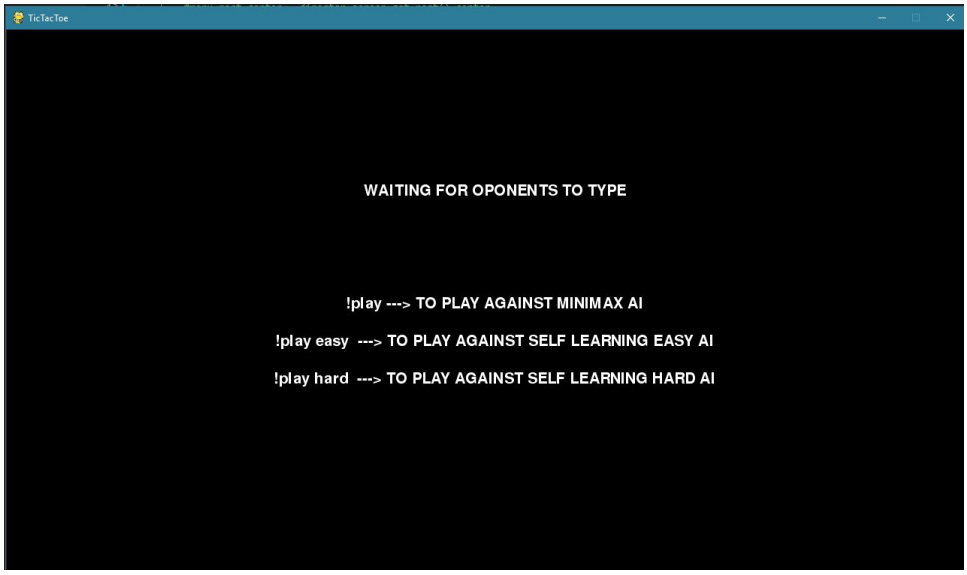
    - Softwares used:
        - Visual Studio code or any IDE you would like to use
            - https://code.visualstudio.com/download
            - Chosen IDE
        - Open Broadcast Software (OBS)
            - https://obsproject.com/
            - Used for streaming to twitch
        - Twitch
            - www.twitch.tv
            - Used for streaming the game live

    - Libraries that needs to be downloaded:
        - Pygame
            - https://www.pygame.org/wiki/GettingStarted
        - Python 3.7.x
            - https://www.python.org/downloads/windows/
            - Note: The group decided to use the 64-bit version
        - Pyautogui
            - https://pyautogui.readthedocs.io/en/latest/install.html

    - How to run a test:
        - Open Visual Studio Code (VSC)
        - Open the folder that has all of the source code
            - This is similar to creating a project in Visual Studio
            - Doing so will put all of the files in your folder inside your workstation
        - Create a new Terminal
            - In VSC, once you create a terminal it will automatically be in the right directory, although, still double check if you are in the right directory

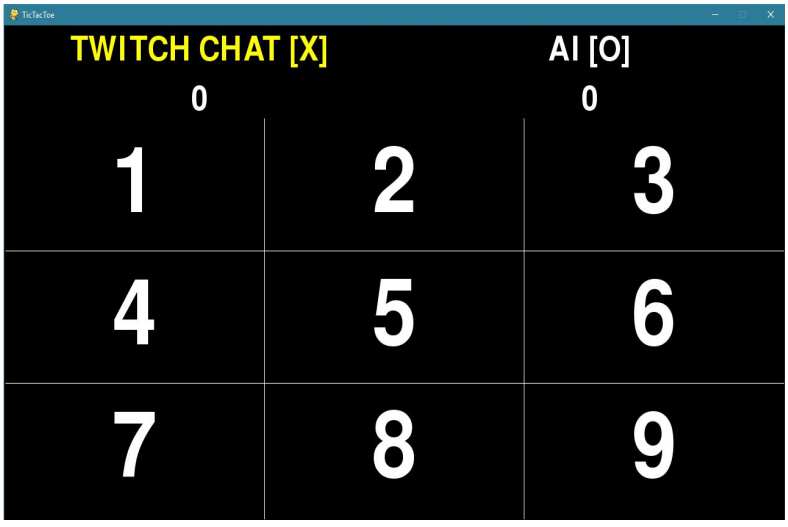
        - Run the test file called:
            - **game_test.py**
            - Do this by typing in the terminal:

- **python game_test.py**
- This will open a window



(fig. 1)

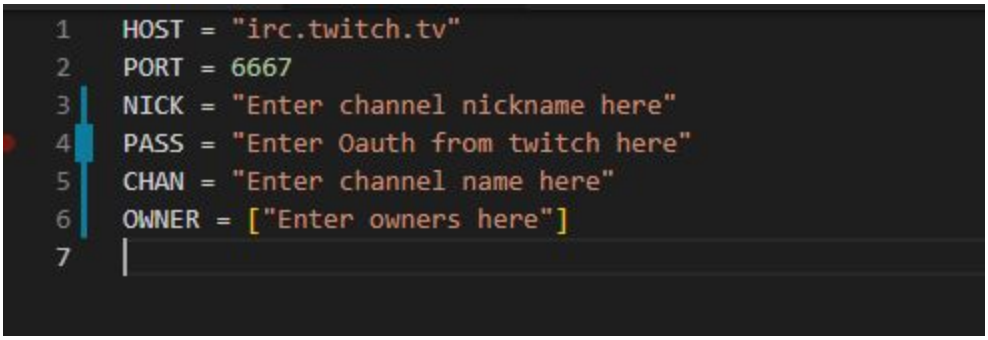
- For testing purposes, we implemented the following menu key presses:
  - ‘p’ -> to play against minimax
  - ‘e’ -> to play against easy SL-AI
  - ‘h’ -> to play against hard SL-AI
  - ‘q’ -> to exit the application
- Pressing ‘p’, ‘e’, or ‘h’ will then move to the game screen (fig.2)



(fig. 2)

        - When it is your turn simply type the number you want you put your piece in

- To run a full feature demo of the game:
    - Make sure that you have downloaded OBS
    - Make sure that you have inputted the proper user information in the file called (fig.3):
        - settings.py
        - This can be found in the folder labeled as twitch
        - You need to edit fields that have an arrow pointing to them
        - NICK
            - Channel nickname
        - PASS
            - You could get this password by **first** logging into your twitch account and clicking on this link:
                - **https://twitchapps.com/tmi/**
            - You will then paste that code in this parameter
        - CHAN
            - Your channel's name
        - OWNER
            - Owners that are allowed to perform special commands such as quitting the game



(fig. 3)

    - Go back to VSC and follow the same step as the demo except this time you type this in your terminal
        - **python main.py**

- This is what it should look like: