Tic-Tac-Toe

Architecture

- `tictactoe.py` contains the game logic
- `test.py` contains the tests
- `runner.py` implements a graphical UI for playing Tic-Tac-Toe using the Pygame library.

Core Logic

- `pickMax(board, bestScore)`: Determines the maximum score the maximizing player can achieve by exploring possible actions and applying alpha-beta pruning.
- `pickMin(board, bestScore)`: Determines the minimum score the minimizing player can achieve by exploring possible actions and applying alpha-beta pruning.
- `minimax(board)`: The main entry point for the minimax algorithm. It decides whether to maximize or minimize the score based on the current player's role and returns the optimal action.

```
def pickMax(board, bestScore):
    if (terminal(board)):
        return utility(board)

    choices = actions(board)
    maxValue = -10
    for choice in choices:
        maxValue = max(maxValue, pickMin(result(board, choice), maxValue))
        # Alpha-beta pruning
        if maxValue > bestScore:
            break

    return maxValue


def pickMin(board, bestScore):
    if (terminal(board)):
        return utility(board)

    choices = actions(board)
    minValue = 10
    for choice in choices:
        minValue = min(minValue, pickMax(result(board, choice), minValue))
        # Alpha-beta pruning
        if minValue < bestScore:
            break

    return minValue

def minimax(board):
    """
    Returns the optimal action for the current player on the board.
    """
    # Decide to pick max or min according to the role
    role = player(board)
    choices = actions(board)
    action = None
    if role == X:
        maxScore = -10
```

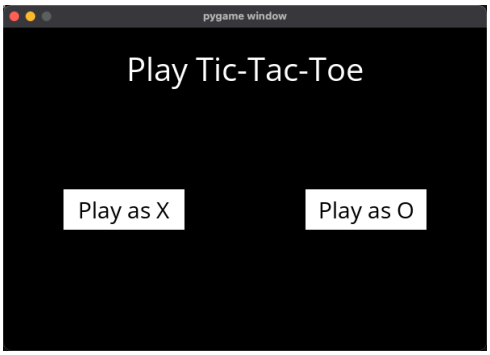
```
        for choice in choices:
            # After making the choice, O will pick the min score
            cur = pickMin(result(board, choice), maxScore)
            if cur > maxScore:
                maxScore = cur
                action = choice
    elif role == O:
        minScore = 10
        for choice in choices:
            # After making the choice, X will pick the max score
            cur = pickMax(result(board, choice), minScore)
            if cur < minScore:
                minScore = cur
                action = choice

    return action
```

There are other important functions such as `utility(board)`, `actions(board)` etc., but I will not go into the details here.

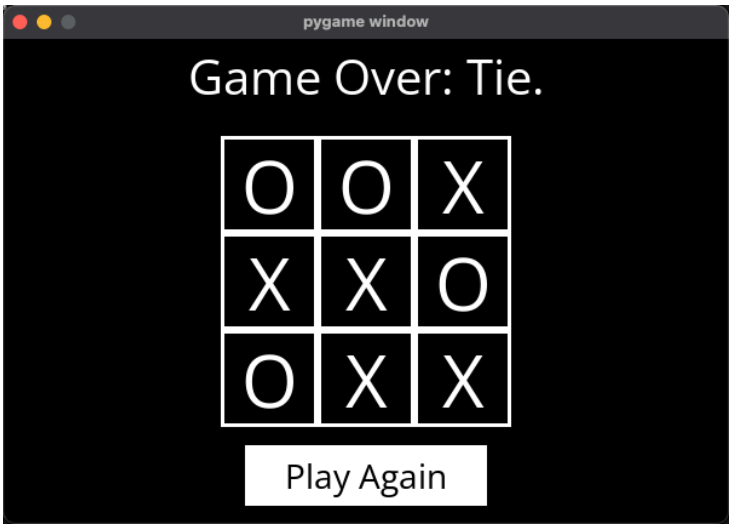
Demo

After installing the required dependencies in `requirements.txt`, you can run the game using the following command:

```
python runner.py
```



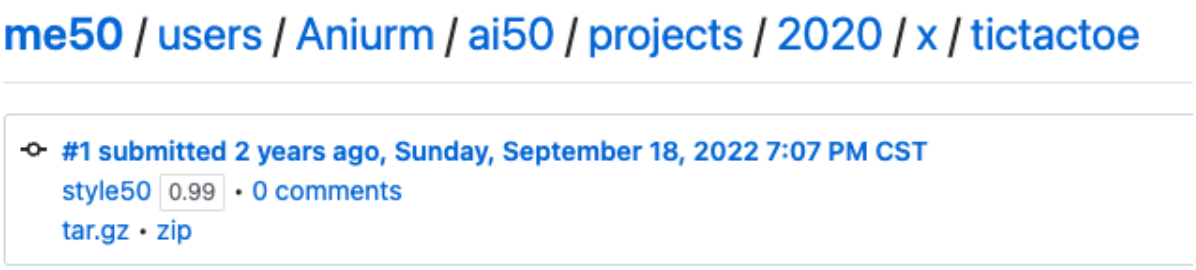
You will NEVER win against the AI.



Reference

Since the starter code provided by the instructor was too simplistic, I used the code framework from CS50AI. I had already implemented tic-tac-toe in the CS50AI course back in 2022. Below are the commit records and the URL for the CS50AI tic-tac-toe project.

My Commit Records



CS50AI Tic-Tac-Toe

https://cs50.harvard.edu/ai/2020/projects/0/tictactoe/