VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

UNIVERSITY OF SCIENCE

REPORT TICTACTOE

COURSE: ARTIFICIAL INTELLIGENCE

TEACHER :	Nguyen Ngoc Thao
ASSISTANT TEACHING:	Nguyen Thai Vu

LIST OF STUDENT:	Hoang Huu Minh An	20127102
	Nguyen Vo Minh Tri	20127364
	Pham Minh Xuan	20127395

TABLE OF CONTENTS

I. STUDENT'S INFORMATION

ID	NAME	MAIL
20127102	Hoàng Hữu Minh An	20127102@student.hcmus.edu.vn
20127364	Võ Nguyễn Minh Trí	20127364@student.hcmus.edu.vn
20127395	Phan Minh Xuân	20127395@student.hcmus.edu.vn

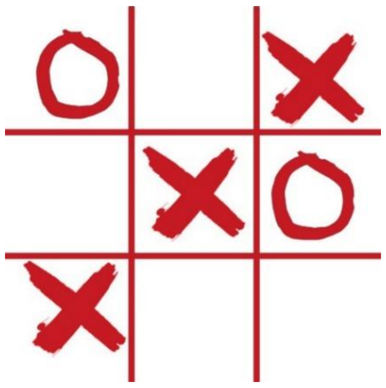
II. INTRODUCE

1. Introduce requirement

- In this project, students research and implement the adversarial searching algorithm.
- In addition, students implement an application (tic-tac-toe problem) and apply the adversarial technique to solve that tic-tac-toe.
- Programming language: Python (for visualization, we recommend students use tkinter library or Pygame of Python).

2. Introduce Tic-Tac-Toe Game

- Tic-Tac-Toe is one of the paper-and-pencil games. This game requires two players in 3x3 grid with Player 1 acts as "O" and Player 2 acts as "X", or vice versa. The objective of this game is to take place of three connecting grids in a horizontal, vertical, or diagonal way/fork.
- Tic-tac-toe can be also be played on a 5-by-5 grid with each player trying to get five in a row.
- The game can also be played on larger grids, such as 10-by-10 or even 20-by-20. For any grid of 6-by-6 or greater, it might be best to make your goal to get five in a row. This turns the basic game of tic-tac-toe into a much more complex game with similarities to the board game Pente, meaning "five" in Greek. Similarly, the goal of Pente is for a player to score five marks in a row.



III. INTRODUCE ABOUT THE ALGORITHM

1. Adversarial Search

- Adversarial search is a search, where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us.
- we have studied the search strategies which are only associated with a single agent that aims to find the solution which often expressed in the form of a sequence of actions.
- But, there might be some situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.
- The environment with more than one agent is termed as multi-agent environment, in which each agent is an opponent of another agent and playing against each other. Each agent needs to consider the action of the other agent and the effect of that action on their performance.
- So, Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches, often known as Games.
- Games are modeled as a Search problem and heuristic evaluation function, and these are the two main factors that help to model and solve games in AI.

2. Implement application (Tic-Tac-Toe problem)

a. Idea

- To solve games using AI, we will introduce the concept of a game tree followed by minimax algorithm. The different states of the game are represented by nodes in the game tree, very similar to the above planning problems. The idea is just slightly different. In the game tree, the nodes are arranged in levels that correspond to each player's turns in the game so that the "root" node of the tree (usually depicted at the top of the diagram) is the beginning position in the game. In tic-tac-toe, this would be the empty grid with no Xs or Os played yet. Under root, on the second level, there are the possible states that can result from the first player's moves, be it X or O. We call these nodes the "children" of the root node.

b. Implement Algorithm

Implement different classes with specific roles. Includes the following classes: board to implement attributes related to the board, XAT to implement attributes related to the players specifically is AI and another classes.

`__init__()`: It specifies how the game is set up at the start.

`terminalTest()`: Terminal test is `-inf` if the player is win, if BOT is win, terminal test return `inf`. The state where the game ends is called terminal state.

`action()`: A list of (move, state) pairs specifying legal moves.

`utility()`: A numerical value of a terminal state s for a player p

- ✓ The evaluation is calculated by the sum of the evaluations of the moves on the board.
    - The evaluation is added or subtracted a value which 1 divided by the distance of this position to the center position + 1 for BOT or player.
    - If the next step is on the main diagonal or minor diagonal of the chessboard. For BOT, the evaluation is added for 0,15. By opposite, the evaluation is subtracted 0,15 for player.

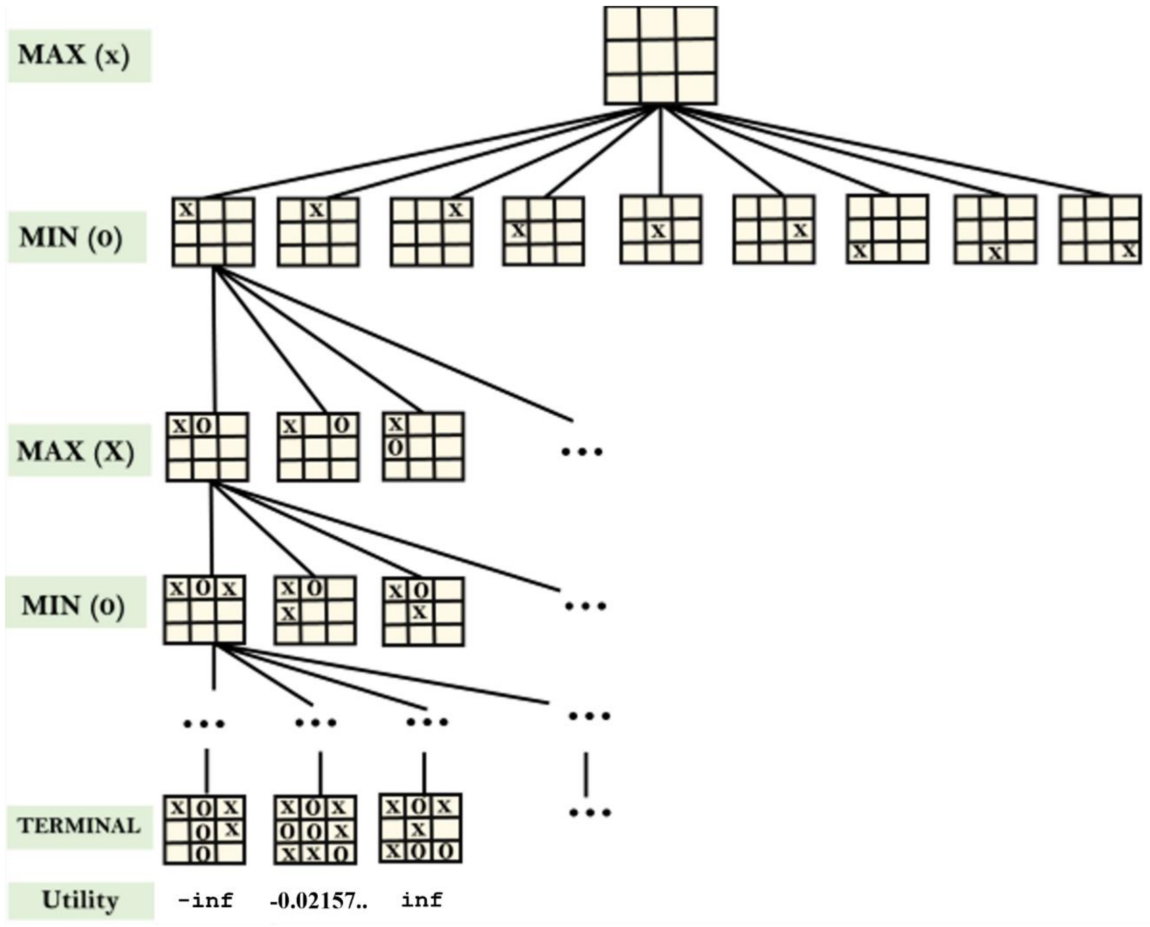
`possibleMove()`: List of cells that can be typed in the next step.

`minimaxAlgorithm()`: Implement minimax algorithm.

- ➢ Firstly, set the stop state by check depth of tree = 0 or the match is draw. If it's true, return the evaluation of current state of board. If player is win, return `-inf`. If BOT is win, return `inf`.
- ➢ After that, call recursive of Min-Max Algorithm with current depth equals to previous depth value to find the evaluations of all possible move. For Bot state is True return max value which max value of above list evaluations and opposite that to return min value which min value of above list evaluations.

❖ Tic-Tac-Toe game tree:

- There are two players MAX and MIN.
- Players have an alternate turn and start with MAX.
- MAX maximizes the result of the game tree.
- MIN minimizes the result.



❖ Hence adversarial Search for the minimax procedure works as follows:

- It aims to find the optimal strategy for MAX to win the game.
- It follows the approach of Depth-first search.
- In the game tree, optimal leaf node could appear at any depth of the tree.
- Propagate the minimax values up to the tree until the terminal node is discovered.

❖ In a given game tree, the optimal strategy can be determined from the minimax value of each node, which can be written as MINIMAX(n). MAX prefer to move to a state of maximum value and MIN prefer to move to a state of minimum value then:

$$\text{MINIMAX}(s) = \begin{cases} \boxed{\text{UTILITY}(s)} \text{ For MAX} & \text{if TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

3. Time/space complexity

a. Completeness

- ✓ Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.

b. Time complexity

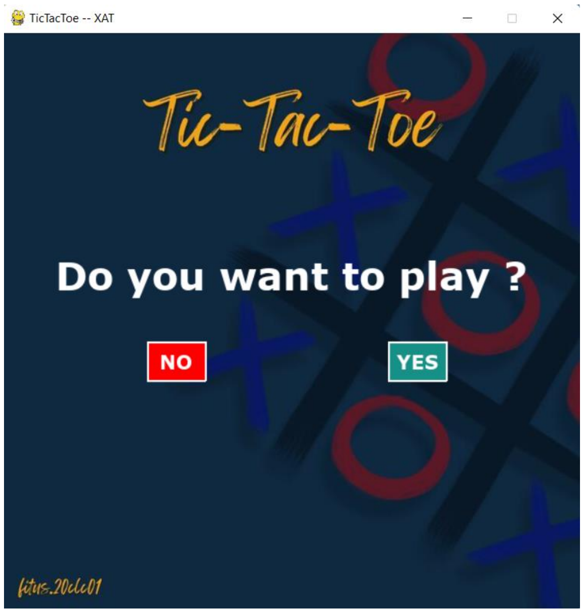
- ✓ O(b^m): As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **O(b^m)**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- ➢ Survey with different depths.
  - Given the depth of the tree is 3, the average time for a step:
    - For 3x3 board: 0,009s
    - For 5x5 board: 0,3462s
    - For 7x7 board: 6,9728s
  - Given the depth of the tree is 5, the average time for a step:
    - For 3x3 board: 0,0232s
    - For 5x5 board: 12,1324s
    - For 7x7 board: time-consuming
  - ➢ With the depth of the tree is 5, the algorithm takes a long time for board of size 5x5 and 7x7.
  - ➢ It's faster for the depth of the tree is 3.

c. Space complexity

- ✓ O(bm) (Depth-first exploration)

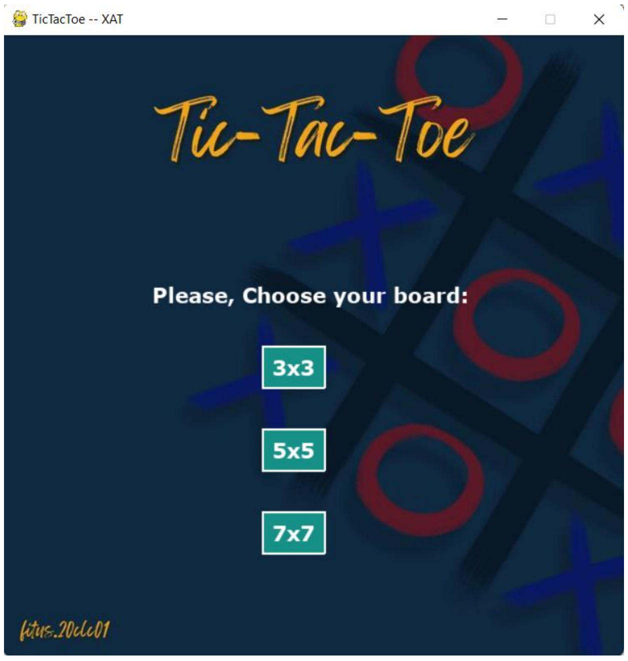
4. Game manual

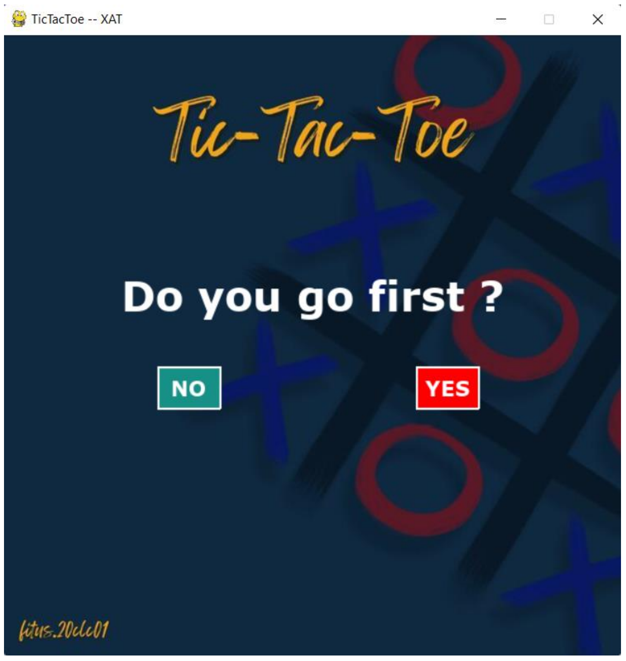
❖ When open, the game will ask you whether you want to play. Pick your option with control key on key board then click space to select.



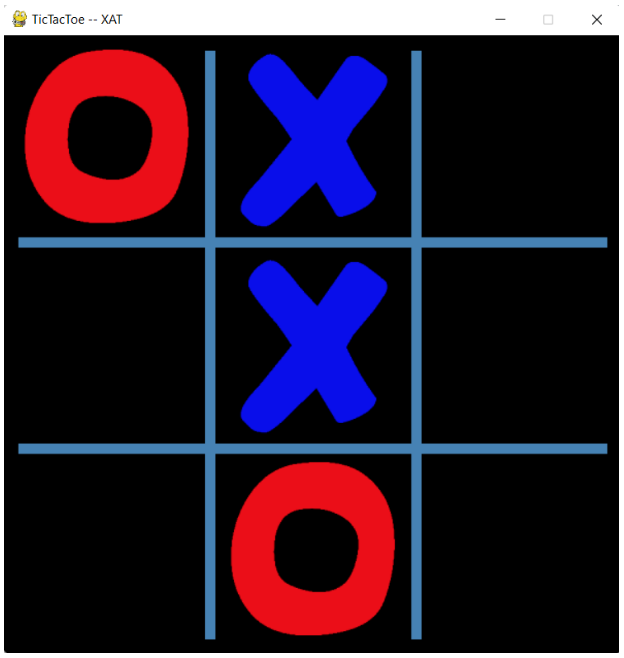
❖ Then you will moved to the start menu. Here we have 3 option to choose the size of board:

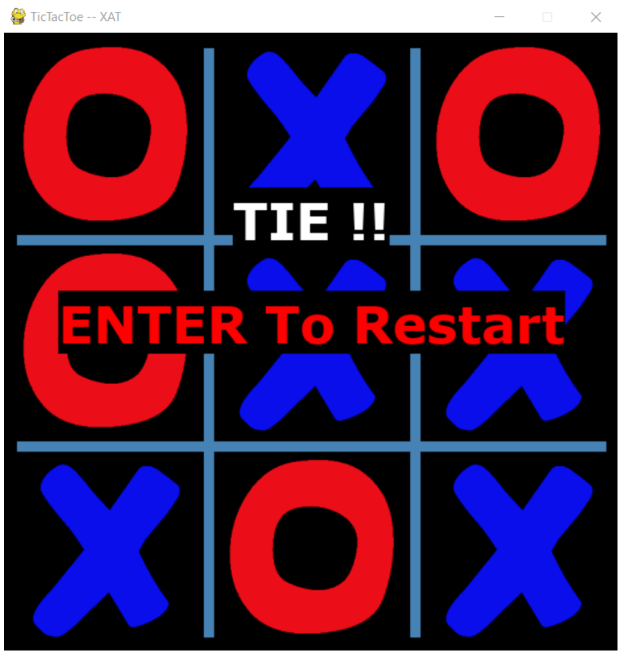
- ➢ 3x3
- ➢ 5x5
- ➢ 7x7

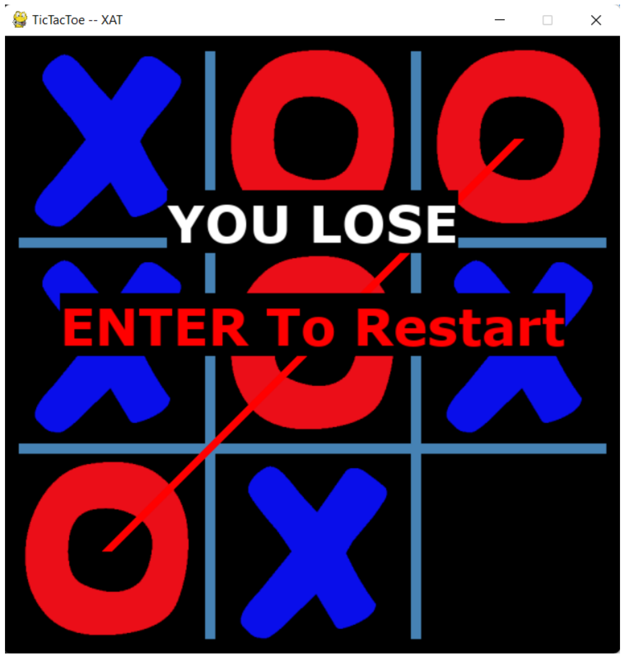




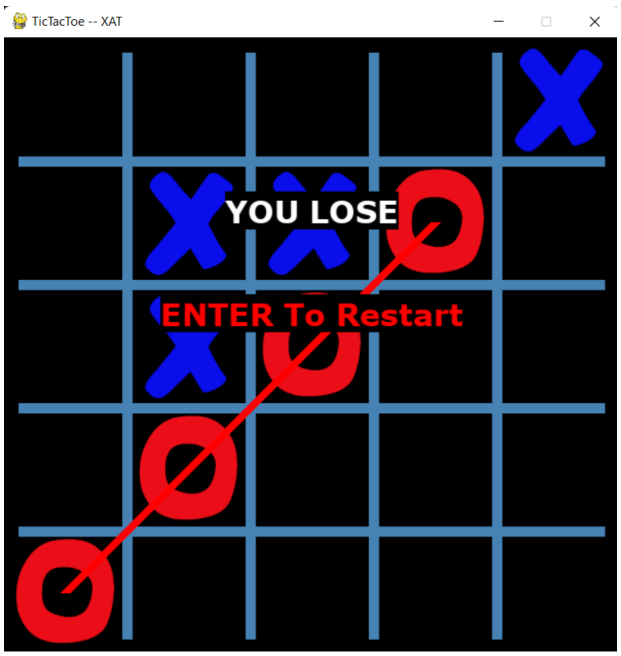
- While playing:
  - Click on the box that you want to type.
  - Board 3x3

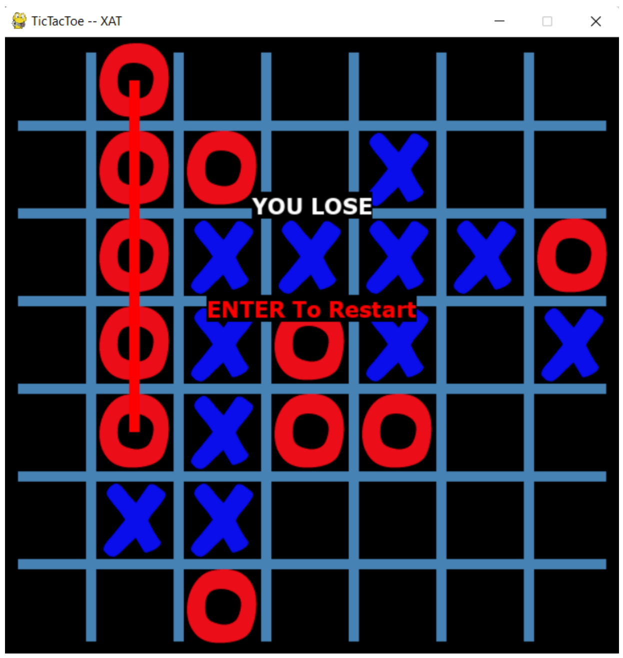






➢ Board 5x5 and 7x7





❖ After the game is over, press Enter to restart.

IV. LINK DEMO APPLICATION

https://studenthcmusedu my.sharepoint.com/:f:/g/personal/20127395_student_hcmus_edu_vn/EhKN4RwmvtBCgu7PEdQXXlEBLNeB8nAUIHakID2uP20QfQ?e=xj2m9x

V. REFERENCES

Teacher Nguyen Ngoc Thao
https://drive.google.com/drive/folders/1OmW_a959zfyCfWP__agipFzfEZX_eV1S
Use Pygame
https://codelearn.io/sharing/lap-trinh-game-co-ban-voi-pygame

ꕥ END ꕥ