Block-Tac-Toe: 5x5 Tic-Tac-Toe with Obstacles

Nguyễn Quốc Khánh
24520793
University of Information Technology

Nguyễn Thái Duy
24520390
University of Information Technology

Hoàng Quốc Duy
24520373
University of Information Technology

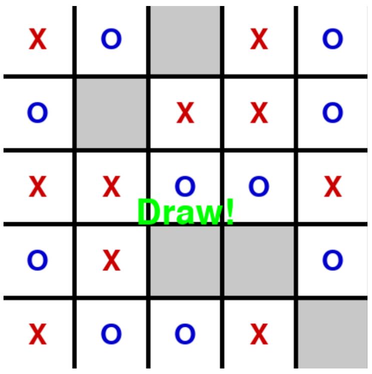
14/05/2025

Abstract

This project implements a 5x5 Tic-Tac-Toe game with obstacle pieces using object-oriented programming. It features a simple reinforcement learning agent trained to play the game. The application supports both human vs. human and human vs. AI game modes, providing a platform to explore AI behavior in a modified game environment.

1 Project Overview

This project presents an object-oriented programming (OOP) implementation of a modified 5x5 Tic-Tac-Toe game, we call it **Block-Tac-Toe** The game introduces an additional layer of complexity by including 'obstacle' pieces alongside the traditional 'X' and 'O' markers, with the winning condition being the first player to achieve four of their markers in a row horizontally, vertically, or diagonally. A core component of this project is the development and integration of a simple reinforcement learning (RL) agent designed to learn optimal strategies for playing this variant of the game. The application provides two distinct player modes: a classic two-player mode allowing human vs. human competition on the same device, and a human vs. AI mode where a human player can challenge the trained RL agent. This project serves to demonstrate the application of OOP principles in game development and explore the effectiveness of simple reinforcement learning techniques in creating an intelligent agent for a non-trivial game environment, offering an engaging platform for users to play and observe AI behavior.



2 Classes inside Block-Tac-Toe

2.1 Class Board

This class represents the game board, its state, obstacle placement and win condition checking

2.1.1 Attributes

- **rows**: number of rows
- **cols**: number of columns
- **win_len**: The number of consecutive marks required to win (e.g., 4 in a row)
- **cells**: A 2D list representing the board grid.
- **legal**: A set containing (row, col) tuples of all currently empty, non-obstacle cells where a move is possible.

2.1.2 Methods

- _init_: Define number of rows, columns, winning condition and number of obstacles being placed
- _init_board: Initializes cells with empty cells and places obstacles
- _place_obstacles : Randomly place obstacles
- reset: Reset the board
- is_legal_move: Check if the cell at row i and column j is valid
- place_mark: Attempt to place mark at row i and column j.
- check_winner: Check if either of the players have won
- draw: Check if game is draw.

2.2 Class Player

A base class representing a generic player.

2.2.1 Attributes

- symbol: Player X or O?

2.2.2 Method

- _init_: Initialize the player symbol.
- make_move: Define how the player move on the board.

2.3 Class HumanPlayer (inherits from class Player)

Represents a human player by makes moves via mouse input, from both

2.3.1 Attributes

- symbol: Inherited from class Player

2.3.2 Methods

- make_move: Take mouse position and attempt to place the player's mark on the board.

2.4 Class Bot (inherits from class Player)

Work in Progress, this is just the part for RL, the attributes and method may just be the same with HumanPlayer

2.5 Class Game

Manages the game loop, player turns, rendering, and event handling. Attributes:

- _rows : Number of rows on the board
- _cols : Number of columns on the board
- _win_len : Number of marks needed to win
- _num_obstacles : Number of obstacles to be placed
- _board : Instance of the Board class
- _players : List of player objects, typically two HumanPlayer instances
- _turn : Index (0 or 1) indicating which player's turn it is
- _cell_size : Size of each grid cell in pixels
- _margin : Margin between grid cells
- _screen : Pygame surface used for drawing the board
- _fonts : Dictionary containing fonts for rendering symbols and messages
- _game_over : Boolean flag indicating whether the game has ended
- _winner : The winning symbol ('X', 'O'), or None if the result is a draw

Methods:

- _init_() : Initializes all components including the board, players, fonts, and game window
- run() : Main loop for handling events, updating the game state, and managing turns
- _draw() : Renders the board, messages, and restart instructions on the screen