A

Artificial Intelligence Project REPORT

ON

"Tic-Tac-Toe"

OF

T.E.(AI & DS)

(Academic Year: 2023-2024)

SUBMITTED BY

Shivam Awasare

Roll No :- T14110004

GUIDED BY

Prof . Dikshendra Sarpate

Department of AI & DS

Zeal Education Society's

Zeal College of Engineering & Research

Narhe, Pune-411041

A Project REPORT

ON

"Tic-Tac-Toe"

Submitted By

Shivam Awasare

in partial fulfilment for the award of the degree of

Bachelor of Engineering
of
Savitribai Phule Pune University

IN

AI & DS

Zeal College Of Engineering and Research,Narhe, Pune
2022 - 2023

Zeal Education Society’s

Zeal College of Engineering & Research

Department of AI & DS

CERTIFICATE

This is to certify that DBMS Project entitled

”Tic-Tac-Toe”

have successfully completed by **”Shivam Awasare”** of TE (AI & DS) in the academic year 2023-2024 in partial fulfillment of the third Year of Bachelor degree in “AI & DS Engineering” as prescribed by the Savitribai Phule Pune University.

Prof. D. D. Sarpate	**Dr.A. M. Kate**
HOD	Principal

Place: ZCOER, Pune.
Date: 31/10/2023

ACKNOWLEDGEMENT

We take this opportunity to thank our guide **Prof. Smruti Vyavahare** and Head of the Department **Prof. D. D. SARPATE** for their valuable guidance and for providing all the necessary facilities, which were indispensable in the completion of this project report. We are also thankful to all the staff members of AI & DS the Department for their valuable time, support, comments, suggestions and persuasion. We would also like to thank the institute for providing the required facilities, Internet access and important books.

Shivam Awasare

Roll No. T1411004

TE

ABSTRACT

Tic-Tac-Toe This project presents the development of a graphical Tic-Tac-Toe game with an artificial intelligence (AI) opponent. The game is implemented in Python using the 'tkinter' library for the graphical user interface and features a challenging AI opponent that uses the minimax algorithm with alpha-beta pruning. Players can enjoy a classic game of Tic-Tac-Toe while testing their skills against the AI.

The project explores the implementation of the game's core logic, AI strategy, and graphical user interface. It allows players to make moves (X) on the game grid, and the AI opponent responds with intelligent moves (O). The game checks for various win conditions, including player wins, AI wins, and draws, providing a complete gaming experience.

The code is open-source and can be customized or extended for various purposes. The project is a great example of integrating AI algorithms into a simple game, and it serves as a foundation for further exploration and experimentation with AI-based gaming applications.

This report provides an overview of the project's features, requirements, and usage instructions, making it accessible to developers and enthusiasts interested in building similar AI-based games.

Keyword - Tic-Tac-Toe, Artificial Intelligence, Minimax Algorithm, Graphical User Interface

Contents

1 Introduction

Tic-Tac-Toe, often known as "X and O" or "Noughts and Crosses," is a classic and widely recognized two-player game. It is traditionally played on a 3x3 grid, where players take turns marking empty cells with their respective symbols, 'X' and 'O.' The objective is to be the first to form a horizontal, vertical, or diagonal line of their symbol or to fill the entire grid without achieving a winning pattern, resulting in a draw.

This project presents the development of a modern and interactive version of Tic-Tac-Toe, enhanced with artificial intelligence (AI). While traditional Tic-Tac-Toe can be easily mastered, the introduction of an AI opponent makes the game more challenging and engaging for players.

The core goal of this project is to create a game that showcases AI capabilities in decision-making, strategy, and intelligent gameplay. The AI opponent is designed to provide players with a tough gaming experience, utilizing the minimax algorithm with alpha-beta pruning for making its moves.

In this report, we will explore the features, dependencies, installation instructions, and usage of this AI-powered Tic-Tac-Toe game. We will also delve into the project's structure, present code snippets, and showcase the game's output. Through this project, we aim to demonstrate the fusion of classic games with modern AI techniques, opening up possibilities for future gaming applications.

Let's embark on a journey through this AI-enhanced Tic-Tac-Toe game and discover how it combines the charm of a timeless classic with the intelligence of modern technology.

2 Features

The AI-enhanced Tic-Tac-Toe project offers several notable features that make it an engaging and challenging gaming experience. These features include:

1. **Interactive Tic-Tac-Toe Gameplay:** Players can enjoy the classic Tic-Tac-Toe game on a 3x3 grid with an interactive graphical user interface.
2. **AI Opponent with Minimax Algorithm:** The AI opponent uses the minimax algorithm with alpha-beta pruning to make intelligent moves, providing a challenging gaming experience for players.
3. **Player vs. AI Mode:** The game allows players to compete against the AI opponent, testing their skills and strategic thinking.
4. **Win Condition Detection:** The game checks for various win conditions, including player wins, AI wins, and draws, ensuring a fair and competitive environment.
5. **New Game Option:** Players can start a new game at any time, resetting the grid and scores for a fresh gaming experience.
6. **Customizable and Extensible:** The open-source code can be customized and extended for various purposes, making it a valuable learning resource for developers.
7. **User-Friendly GUI:** The game features a user-friendly graphical user interface with clear grid cells and responsive buttons.

These features combine the classic charm of Tic-Tac-Toe with the intelligence of artificial intelligence, providing an enjoyable and educational gaming experience.

3 Dependencies

The development of the AI-enhanced Tic-Tac-Toe project relies on several dependencies and technologies to create a smooth gaming experience. The primary dependencies include:

1. **Python Programming Language:** The project is implemented in Python, a versatile and widely-used programming language. Ensure you have Python installed on your system to run the game.
2. **tkinter Library:** The graphical user interface of the game is built using the 'tkinter' library, which is included in Python's standard library. This library is used for creating windows, buttons, and other graphical elements.
3. **Basic Understanding of Minimax Algorithm:** While not a software dependency, a basic understanding of the minimax algorithm and its usage in game theory is helpful to comprehend the AI's decision-making process.
4. **Code Editor or IDE:** You can use any code editor or integrated development environment (IDE) of your choice to work with the project source code. Popular options include Visual Studio Code, PyCharm, and Jupyter Notebook.

These dependencies are relatively straightforward, and the project's code is designed to be beginner-friendly, making it accessible to a wide range of users. Make sure to have Python and 'tkinter' available on your system to run and explore the Tic-Tac-Toe game.

4 Installation

Setting up and running the AI-enhanced Tic-Tac-Toe project is a straightforward process. Follow these steps to install and run the game:

1. **Clone the Repository:** Start by cloning the project repository from its source on GitHub. You can use the following command in your terminal or command prompt:

   $https : //github.com/zargon01/AI - based - Tic - tac - toe$

2. **Navigate to the Project Directory:** Change your working directory to the project folder: $cdyour - tic - tac - toe - repo$

3. **Run the Game:** Once you are in the project directory, run the Tic-Tac-Toe game using Python:

   pythontic_tac_toe.py

   This command will launch the game, and you can start playing against the AI opponent.

4. **Enjoy the Game:** Have fun playing Tic-Tac-Toe against the AI. You can make your moves by clicking on the grid cells, and the AI will respond with its intelligent moves.

The game is designed to be beginner-friendly, and the provided code is open-source, allowing you to explore, modify, and learn from it. The installation process is simple, and you can start enjoying the AI-enhanced Tic-Tac-Toe experience in no time.

5 Uses

The AI-enhanced Tic-Tac-Toe project serves several purposes and offers a range of uses, including:

1. **Entertainment and Gaming:** The primary use of the project is to provide an entertaining and engaging gaming experience. Players can enjoy classic Tic-Tac-Toe with the added challenge of competing against an AI opponent.
2. **Educational Resource:** The project is an educational resource for those interested in game development and artificial intelligence. It showcases the implementation of the minimax algorithm and provides insights into AI decision-making in gaming.
3. **Open-Source Learning:** The project's code is open-source and can be freely accessed, modified, and extended. It serves as a valuable learning tool for individuals looking to explore game development and AI integration.
4. **Algorithm Exploration:** Developers and AI enthusiasts can use this project to explore the minimax algorithm and its application in game theory. It offers a practical example of how this algorithm can be implemented in a game.
5. **Customization and Extension:** The open-source nature of the project allows users to customize and extend the game's features. You can add new functionalities, improve the AI, or adapt it for specific purposes.

Whether you're a gamer looking for a challenging opponent, a developer interested in AI algorithms, or an educator seeking a practical resource, the AI-enhanced Tic-Tac-Toe project has uses that cater to a variety of interests and objectives.

6 Technologies and Algorithms Used

The AI-enhanced Tic-Tac-Toe project incorporates various technologies and algorithms to provide an interactive gaming experience. The primary technologies and algorithms used in the project include:

1. **Python:** The project is developed using the Python programming language. Python's simplicity and versatility make it an ideal choice for both game development and AI implementation.
2. **tkinter Library:** The graphical user interface (GUI) of the game is built using the 'tkinter' library. This library is a standard part of Python and facilitates the creation of windows, buttons, and other GUI elements.
3. **Minimax Algorithm:** The core AI strategy of the project is based on the minimax algorithm. The minimax algorithm is a decision-making algorithm used in two-player games to determine the best move for a player while assuming that the opponent plays optimally.
4. **Alpha-Beta Pruning:** To enhance the efficiency of the minimax algorithm, alpha-beta pruning is used. This technique reduces the number of nodes that need to be evaluated in the game tree, leading to faster decision-making by the AI.

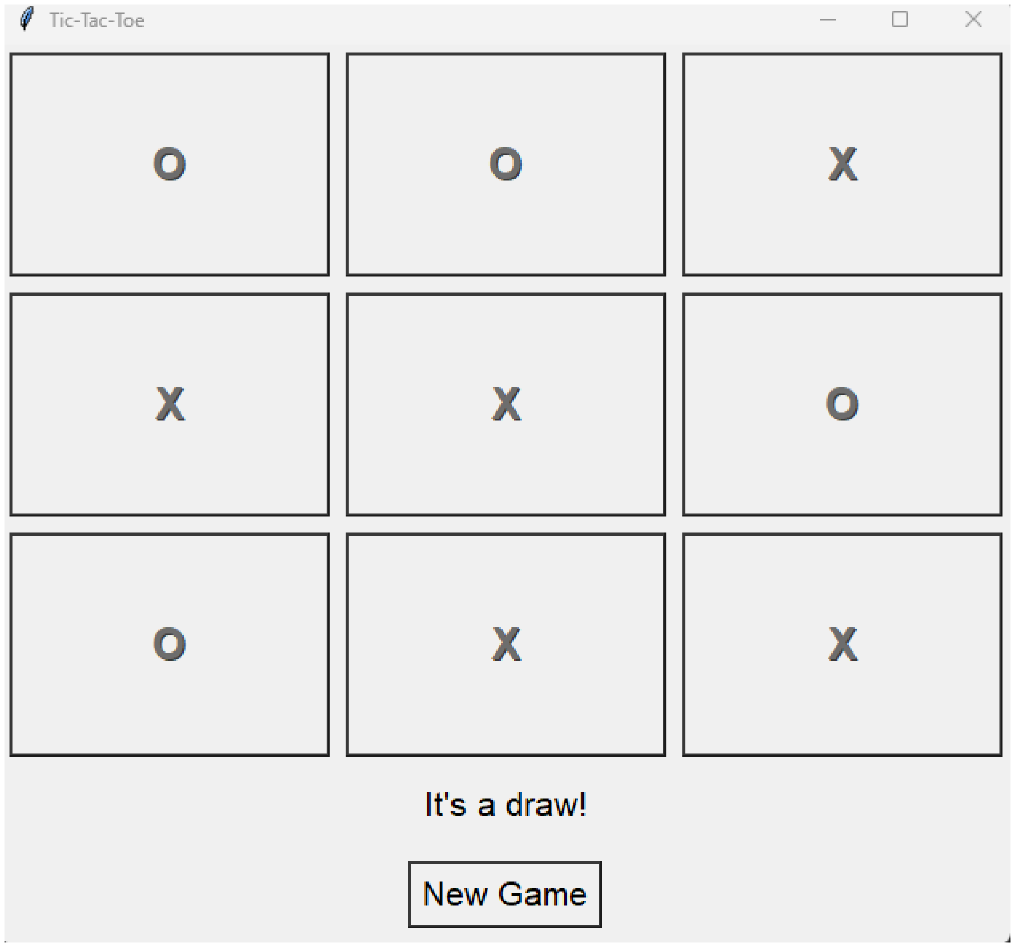
These technologies and algorithms work in harmony to provide a challenging and engaging gaming experience for players. The minimax algorithm with alpha-beta pruning, in particular, is responsible for the intelligent decision-making of the AI opponent, ensuring that players have to strategize to win the game.

7 Code

```python
import tkinter as tk

# Function to check if a player has won
def check_win(board, player):
    # Check rows
    for row in board:
        if all(cell == player for cell in row):
            return True

    # Check columns
    for col in range(3):
        if all(board[row][col] == player for row in range(3)):
            return True

    # Check diagonals
    if all(board[i][i] == player for i in range(3)) or all(board[i][2 - i] == player for i
    in range(3)):
        return True

    return False

# Function to check if the board is full
def is_full(board):
    return all(cell != ' ' for row in board for cell in row)

# Function to handle player's move
def player_move(row, col):
    if board[row][col] == ' ':
        board[row][col] = 'X'
        update_button(row, col)
        if check_win(board, 'X'):
            result_label.config(text="Player wins!")
        elif is_full(board):
            result_label.config(text="It's a draw!")
        else:
            ai_move()

# Function to handle AI's move
def ai_move():
    best_move, _ = minimax(board, 'O', 'O', -float('inf'), float('inf'))
    row, col = best_move
    board[row][col] = 'O'
    update_button(row, col)
    if check_win(board, 'O'):
        result_label.config(text="AI wins!")
    elif is_full(board):
        result_label.config(text="It's a draw!")

# Minimax algorithm with alpha-beta pruning
def minimax(board, currentPlayer, aiPlayer, alpha, beta):
    if check_win(board, aiPlayer):
        return None, 1
    elif check_win(board, 'X'):
        return None, -1
    elif is_full(board):
        return None, 0

    empty_cells = [(row, col) for row in range(3) for col in range(3) if board[row][col] ==
    ' ']
    best_move = None

    if currentPlayer == aiPlayer:
        best_score = -float('inf')
        for row, col in empty_cells:
            board[row][col] = currentPlayer
            _, result = minimax(board, 'X', aiPlayer, alpha, beta)
            board[row][col] = ' '
            if result > best_score:
                best_score = result
                best_move = (row, col)
            alpha = max(alpha, best_score)
            if alpha >= beta:
                break
    else:
        best_score = float('inf')
        for row, col in empty_cells:
            board[row][col] = currentPlayer
            _, result = minimax(board, aiPlayer, aiPlayer, alpha, beta)
            board[row][col] = ' '
            if result < best_score:
                best_score = result
                best_move = (row, col)
            beta = min(beta, best_score)
            if beta <= alpha:
                break

    return best_move, best_score

# Function to start a new game
def new_game():
    global board
    for row in range(3):
        for col in range(3):
            board[row][col] = ' '
            buttons[row][col].config(text=' ', state='active')
    result_label.config(text="")

# Function to update the button text
def update_button(row, col):
    buttons[row][col].config(text=board[row][col], state='disabled')

# Create the main window
root = tk.Tk()
root.title("Tic-Tac-Toe")

# Initialize the game board
board = [[' ' for _ in range(3)] for _ in range(3)]

# Create buttons for the game grid with styling
buttons = [[None for _ in range(3)] for _ in range(3)]
button_font = ('Helvetica', 20, 'bold')
for row in range(3):
    for col in range(3):
        buttons[row][col] = tk.Button(root, text=' ', width=10, height=3, font=button_font,
    command=lambda r=row, c=col: player_move(r, c))
        buttons[row][col].grid(row=row, column=col, padx=5, pady=5, ipadx=10, ipady=10)

# Label to display the game result with styling
result_label = tk.Label(root, text="", font=("Helvetica", 16))
result_label.grid(row=4, column=0, columnspan=3, padx=10, pady=10)
```

```

# New Game button with styling
new_game_button = tk.Button(root, text="New Game", font=("Helvetica", 16), command=new_game)
new_game_button.grid(row=5, column=0, columnspan=3, padx=10, pady=10)

root.mainloop()
```

8 Output



It's a draw!