TIC TAC TOE

Python OOP & GUI Project

PROJECT TYPE	Desktop Game Application
TECHNOLOGY	Python 3 + Tkinter
ARCHITECTURE	Object-Oriented Programming
AI	Easy / Medium / Hard + Minimax

A clean desktop Tic-Tac-Toe application demonstrating OOP design, GUI development, event handling, game-state management, and an introductory artificial intelligence algorithm.

1. Project Overview

Tic Tac Toe is a polished desktop X-O game developed in Python. The application separates the game rules, computer decision-making, and graphical interface into independent classes, making the project easy to understand, maintain, test, and extend.

Main Features

- Modern dark desktop graphical interface.
- Interactive 3×3 Tic-Tac-Toe board.
- Player vs Player mode.
- Player vs Computer mode.
- Three AI difficulty levels: Easy, Medium, and Hard.
- Hard mode uses the Minimax algorithm for optimal play.
- Automatic winner and draw detection.
- Winning cells are highlighted.
- Scoreboard for X wins, O wins, and draws.
- New Game and Reset Score controls.
- Hover effects and keyboard shortcuts: R for New Game and Esc to exit.
- No third-party packages are required.

Technologies Used

Technology	Purpose
Python 3	Core programming language
Tkinter	Desktop GUI
OOP	Separation of responsibilities into classes
Lists / Data Structures	Board and available-move management
Event Handling	Mouse clicks, buttons, and keyboard shortcuts
Minimax	Optimal AI decision-making in Hard mode

2. OOP Architecture

The project follows separation of concerns: each class has a focused responsibility instead of putting the complete application inside one file.

Class / File	Responsibility
TicTacToeGame — game.py	Stores the board and current player; validates moves; detects winners and draws; trac
TicTacToeAI — ai.py	Chooses computer moves using Easy random play, Medium tactical logic, and Hard M
TicTacToeGUI — gui.py	Builds the window, board, scoreboard, controls, mode/difficulty selectors, and event h
main.py	Application entry point that starts the GUI.

Game Flow

Player action → Game validation → Board update → Win/Draw check → Turn switch → AI move (if selected) → GUI refresh

Project Structure

```
tic_tac_toe/
■■■ main.py
■■■ game.py
■■■ ai.py
■■■ gui.py
■■■ README.md
■■■ requirements.txt
```

3. Artificial Intelligence

The computer opponent supports three difficulty levels. This gives the application a simple progression from random behavior to optimal game play.

Difficulty	Strategy	Behavior
Easy	Random move	Chooses one available cell at random.
Medium	Tactical rules	Wins when possible, blocks the player, prefers center/corners, then selects
Hard	Minimax	Searches future game states and selects the best available outcome.

How Minimax Works

- The AI treats O as the maximizing player and X as the minimizing player.
- For every available move, the algorithm simulates the move and recursively explores the resulting position.
- A winning O position receives a positive score, a winning X position receives a negative score, and a draw receives zero.
- The AI selects the move with the best score, allowing Hard mode to play optimally for Tic-Tac-Toe.

Representative AI Logic

```
if difficulty == "Easy":
    return random.choice(empty)

if difficulty == "Medium":
    # win, block, center, corners, fallback
    ...

# Hard
_, move = self._minimax(board[:], True)
return move
```

4. How to Run

Open a terminal inside the project folder and run the application with:

```
python main.py

# Windows alternative
py main.py
```

No pip installation is required because the project uses Python's built-in Tkinter package.

How to Play

- Choose Player vs Player or Player vs Computer.
- In Player vs Computer mode, select Easy, Medium, or Hard.
- X starts the round.
- Complete a row, column, or diagonal to win.
- Use New Game to start another round while keeping the scoreboard.
- Use Reset Score to clear the accumulated results.

5. Learning Outcomes

- Applied Object-Oriented Programming through multiple cooperating classes.
- Separated game logic from the graphical user interface.
- Implemented event-driven GUI programming with Tkinter.
- Managed game state using lists and structured methods.
- Implemented winner and draw detection.
- Implemented multiple levels of AI behavior.
- Applied the Minimax algorithm to a complete playable game.

6. Future Improvements

- Player names and customizable profiles.
- Sound effects and additional visual animations.
- Theme selection.
- Match history and statistics.
- Online multiplayer support.
- Expanded AI analytics.

Project Goal: Demonstrate how a simple game can be transformed into a clean Python application using OOP, GUI programming, event handling, data structures, separation of concerns, and an introductory AI algorithm.

Prepared as a professional project portfolio document.