Tic-Tac-Toe Game

A Python-based Game with AI and GUI Implementation

Welcome to the documentation for our Tic-Tac-Toe game, a classic reimagined with modern programming techniques. This presentation will guide you through the game's implementation, featuring an AI opponent and a user-friendly graphical interface.

Team Member's:

Bahaa Mahmoud Helmy

21-00550

Waseam Nashat Samir

21-01680

Selvana Atef Kadry Zekri

21-01588

Abdalla Shreif Mohammed

21-00242

Maram Mohamed Ahmed

21-00966

Introduction

Overview

A classic Tic-Tac-Toe game where the player competes against the computer.

Built using Python's Tkinter library for the graphical user interface.

Objective

Align three identical marks (X or O) horizontally, vertically, or diagonally to win.

The game can also result in a tie.

Key Features

Player vs Computer

Interactive gameplay against an AI opponent.

Minimax Algorithm

Ensures the computer makes optimal moves.

Simple GUI

Built using the Tkinter library for ease of use.

Restart Button

Allows restarting the game at any point.

Modules Used

Tkinter

For creating the graphical user interface.

Random

To simulate randomness in decision-making (if needed).

Core Functions

1. **next_turn(row, col)**

   Handles the player's move and updates the board.

2. **computer_turn()**

   Uses the Minimax algorithm to calculate the best move for the AI.

3. **minimax(board, depth, is_maximizing)**

   Recursively evaluates the board to decide the optimal move.

4. **get_best_move()**

   Determines the most strategic move for the AI.

5. **check_winner()**

   Checks for a winner or a tie after each move.

6. **start_new_game()**

   Resets the game board for a fresh start.

Game Flow

1

Initialization

The game starts with a 3x3 grid and the player's turn.

2

Player's Turn

Player clicks on a cell to place 'X'.

The system checks for a winner.

3

Computer's Turn

AI calculates the best move using Minimax.

4

Winner Check

After each move, checks for a winner or tie.

5

Restart

Players can reset the game anytime.

Example Use Case

1. **Start**
   Player begins by clicking a cell.

2. **AI Move**
   The computer calculates and places its move.

3. **Alternate Turns**
   The process repeats until there is a winner or a tie.

4. **Restart**
   Click the restart button to play again.

Code Structure Overview

Main Functions

next_turn(), computer_turn(), minimax(), get_best_move(), check_winner(), start_new_game()

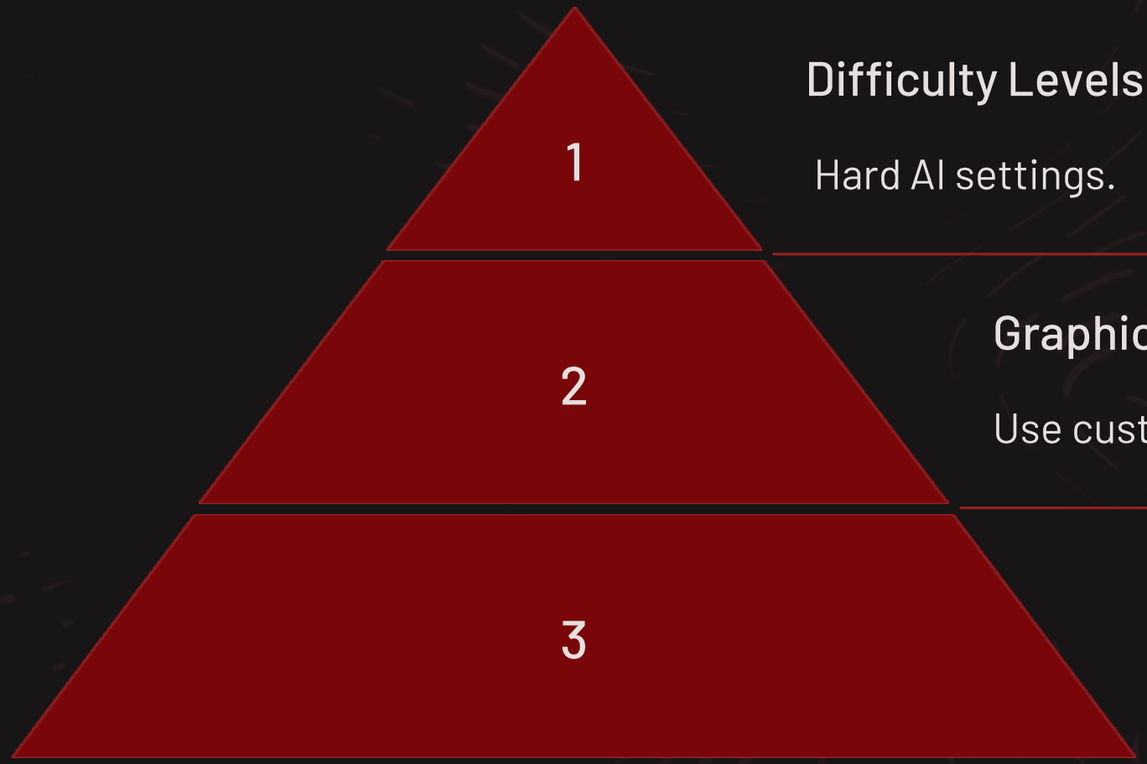
Tkinter GUI

Window creation and button configuration.

Minimax Algorithm

Decision-making for the AI.

Future Improvements



Difficulty Levels

Hard AI settings.

Graphics Enhancements

Use custom icons for X and O.

Multiplayer Mode

Play vs Ai gameplay.

Conclusion

Summary

Combines classic gameplay with AI and GUI.

A great demonstration of game logic and algorithmic decision-making.

Learning Points

Python programming, Tkinter for GUI, and Minimax for AI.

Thank YOU

Questions or feedback?