Design And Analysis of Algorithm

PROJECT REPORT

ON

Tic-Tac-Toe Game in Python with Tkinter

SubjectCode:24CAP-612

Submitted By **SubmittedTo**

Name:Priyansh kumar Mr. Vedant Saraf

UID:24MCI10262

INDEX

ACKNOWLEDGEMENT

I would like to extend my sincere appreciation to the invaluable online resources, forums, and communities dedicated to programming, particularly the Python and Tkinter communities. Their contributions, including shared code snippets, best practices, and problem-solving discussions, have greatly enriched my understanding and made this Tic-Tac-Toe project possible. I am also grateful to all developers and enthusiasts who openly share their expertise, enabling beginners like me

to learn, create, and refine our programming skills. This project serves as a testament to the collaborative spirit and knowledge-sharing ethos that define these communities. Thank you for your generous support and guidance.

Abstract

This project presents a simple Tic-Tac-Toe game implemented using Python and Tkinter, the standard GUI toolkit for Python. The game allows two players to alternate turns, marking either "X" or "O" on a 3x3 grid. With built-in logic to check winning conditions and highlight the winning combination, the game provides an interactive and user-friendly experience. When a player wins, the game displays a congratulatory message and highlights the winning sequence. This project demonstrates fundamental programming concepts such as functions, conditional logic, loops, and GUI design, making it an excellent educational tool for beginners. The game's modular code structure allows for easy customization and enhancement, offering potential for expanding features, such as score tracking or player customization. By combining Python's logic capabilities with Tkinter's visual elements, this project provides an engaging and instructive introduction to GUI-based game development in Python.

Introduction

The Tic-Tac-Toe game, also known as Noughts and Crosses, is one of the most iconic and simple games played on a 3x3 grid. This project focuses on developing a two-player version of Tic-Tac-Toe using Python and the Tkinter library for graphical user interface (GUI) implementation. The objective of this project is to create an interactive, user-friendly game that allows players to alternately place

their respective marks ("X" or "O") on the grid and check for a winner once a player completes a winning combination of three marks in a row, column, or diagonal.

Game Board and GUI Design:

The board is designed as a 3x3 grid of buttons using Tkinter, each representing a square in the Tic-Tac-Toe game. Each button is clickable, and players can interact with it by placing their mark ("X" or "O"). Tkinter's widget management system is used to create and position these buttons.

Player Alternation and Mark Assignment:

Players alternate turns, with "X" starting the game. The current_player variable tracks whose turn it is, and each click on the grid updates the button's text to reflect the active player’s move. After each move, the game checks for a winner.

Winner Check and Notification:

The game logic includes a function that checks the current board for any winning combinations. It looks for three matching marks in a row, column, or diagonal. When a player wins, the game displays a congratulatory message and highlights the winning combination on the board.

Automatic Solver (Optional Feature):

Although not implemented in this basic version, an auto-solve feature could be added to make the game more challenging. The solver would employ a decision-making algorithm such as the **Minimax algorithm** to determine the best possible move for a player, making the game automated. This would allow a single player to play against the computer, where the computer follows an optimal strategy to either win or force a draw.

Game Restart/End:

After a winner is determined, or if the game reaches a draw, the game can be reset by clearing the grid and resetting the player turn indicator. The winner is announced, and the interface remains responsive for further games.

Technologies Used

Python:
The core language for this project is Python, a versatile and easy-to-learn language.

Python's simplicity allows for clear and concise code, which is perfect for this beginner-level project.

Tkinter:

Tkinter is the standard GUI library for Python and is used for creating the game's graphical interface. It provides all the necessary widgets like buttons, labels, and grids needed for the interactive gameplay.

Conditional Logic:

The game makes heavy use of conditional statements (e.g., if-else) to check player actions and determine the winner. The game's flow is controlled through conditionals, ensuring that the right player's turn is maintained and that winning conditions are accurately verified.

Event-driven Programming:

The game's interface and user interactions are handled in an event-driven manner. Button clicks trigger events, leading to updates in the UI and logic flow.

Message Boxes (for Alerts):

Tkinter's messagebox is used to notify the players of the game's outcome, such as a win or draw. This enhances the user experience by providing immediate feedback.

Design flow/Process

The design flow of the Tic-Tac-Toe game involves several key steps, from setting up the graphical user interface (GUI) to handling the game logic. Below is a step-by-step breakdown of the design process:

---

1. Initialize the Game Environment

Step 1: Initialize Tkinter Window

Create the main Tkinter window (root), which will hold all GUI elements.

Set the title of the window to "Tic-Tac-Toe".

Step 2: Initialize Game Variables

Define key variables, such as:

current_player: Tracks whose turn it is ("X" or "O").

winner: A flag that indicates whether the game has ended (used to disable further moves once the game ends).

buttons: A list to store references to all the buttons that represent the 3x3 grid.

label: Displays whose turn it is, using a Tkinter Label widget.

---

2. Create the Game Board

Step 3: Define the Grid

Create a 3x3 grid of buttons using a loop. Each button represents a cell on the Tic-Tac-Toe board.

Use the Tkinter Button widget with properties like font size, width, and height to make the buttons large and visible.

Each button is connected to a function (buttons_click) via its command property. This ensures that clicking a button triggers the logic to place a mark.

Step 4: Button Placement

Use the grid() method of Tkinter to position each button in the 3x3 layout.

---

3. Player Interaction

Step 5: Handle Button Clicks

The buttons_click function handles the player's move. When a button is clicked, it:

Checks if the button is empty (no mark has been placed).

If the button is empty, it places the current player’s mark ("X" or "O") on the button.

Calls the check_winner() function to verify if there’s a winning combination after the move.

Step 6: Alternate Turns

After each valid move, the current player switches using the toggle_player() function, which updates the player label and sets the current_player variable to the next player.

---

4. Check for Winning Condition

Step 7: Check Winning Combinations

The check_winner() function verifies if the current move resulted in a winning condition. The winning conditions are:

Three consecutive marks in a row (horizontal, vertical, diagonal).

This is done by checking all possible winning combinations (total 8 combinations).

If a winning combination is found:

The winning buttons are highlighted in green.

A pop-up message box (messagebox.showinfo()) is displayed to announce the winner.

The game stops further moves by setting the winner flag to True.

---

5. Game Termination

Step 8: Game End

Once a player wins, or the grid is filled with no winner (draw condition), the game ends.

The winner flag prevents any further moves from being made.

---

6. Optional Features (Future Enhancements)

Step 9: Auto-Solver / AI Player

Implement an AI opponent using a decision-making algorithm (e.g., Minimax) to play against a human player.

The AI can either play optimally or randomly based on difficulty level. This feature would involve:

Evaluating the board state.

Calculating the best possible move using a recursive algorithm.

The AI would automatically place its mark when it's its turn.

---

7. Restart the Game

Step 10: Reset the Game (Optional)

After the game ends, provide an option to restart the game, either by clearing the grid or by using a reset button.

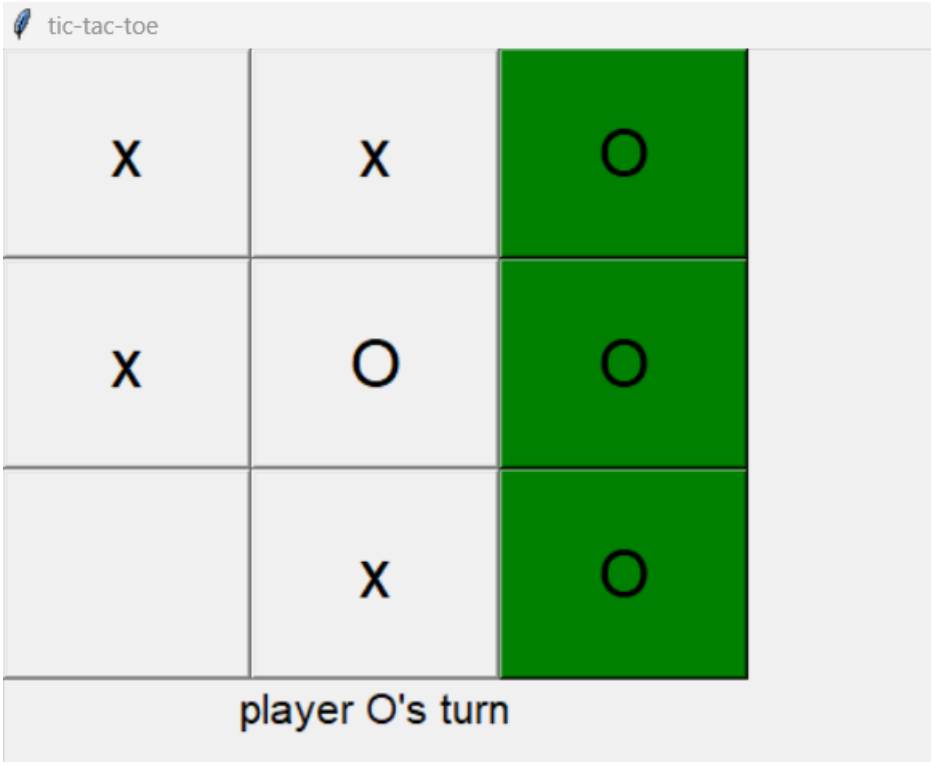
Reset all game variables (e.g., current_player, winner) and the visual interface (clear all button text and colors).

Code of project

```
import tkinter as tk
from tkinter import messagebox

def check_winner():
    for combo in [[0,1,2], [3,4,5], [6,7,8], [0,3,6],
[1,4,7], [2,5,8],[0,4,8],[2,4,6]]:
        if buttons[combo[0]]["text"] ==
buttons[combo[1]]["text"] == buttons[combo[2]]["text"]
!="":
            buttons[combo[0]].config(bg="green")
            buttons[combo[1]].config(bg="green")
            buttons[combo[2]].config(bg="green")
            messagebox.showinfo("Tic-Tac-Toe" , f"player
{buttons[combo[0]]['text']} wins!")
            root.quit()

def buttons_click(index):
    if buttons[index]["text"] == "" and not winner:
        buttons[index]["text"] = current_player
        check_winner()
        toggle_player()

def toggle_player():
    global current_player
    current_player = "x" if current_player == "O" else
"O"
    label.config(text=f"player {current_player}'s turn")

root = tk.Tk()
root.title("tic-tac-toe")

buttons = [tk.Button(root, text="", font=("normal", 25),
width=6, height=2, command=lambda i=i: buttons_click(i))
for i in range(9)]

for i, button in enumerate(buttons):
    button.grid(row=i //3, column=i % 3)
```

```
current_player = "x"
winner = False
label = tk.Label(root, text=f"player {current_player}'s
turn", font=("normal", 16))
label.grid(row=3, column=0, columnspan=3)

root.mainloop()
```

Chapter 4: OUTPUT OF PROJECT



Conclusion

This enhanced Tic-Tac-Toe game demonstrates the power of Python and Tkinter in creating interactive graphical user interfaces (GUIs) for simple games. Through this project, several key concepts were explored, including GUI development, event handling, game logic, and user interaction.

Key features of the enhanced version include:

Player Alternation: Players can take turns marking spaces on the grid, with the current player being displayed at the top.

Win Detection: The game automatically checks for winning combinations and highlights the winning line in green.

Draw Detection: If the board is filled without a winner, the game detects and announces a draw.

Game Reset: After a game ends, players can reset the board and start a new game, with a simple interface button for restarting.

Button Disablement: Once the game concludes, either with a win or a draw, all buttons are disabled to prevent further moves until the game is reset.

The project serves as a foundation for learning about event-driven programming and creating functional applications using Python's Tkinter library. It also opens up possibilities for further enhancements, such as adding an AI opponent, introducing difficulty levels, or customizing the appearance of the game.

Future Scope

This enhanced **Tic-Tac-Toe** game demonstrates the power of Python and Tkinter in creating interactive graphical user interfaces (GUIs) for simple games. Through this project, several key concepts were explored, including GUI development, event handling, game logic, and user interaction.

Key features of the enhanced version include:

- **Player Alternation**: Players can take turns marking spaces on the grid, with the current player being displayed at the top.
- **Win Detection**: The game automatically checks for winning combinations and highlights the winning line in green.
- **Draw Detection**: If the board is filled without a winner, the game detects and announces a draw.
- **Game Reset**: After a game ends, players can reset the board and start a new game, with a simple interface button for restarting.
- **Button Disablement**: Once the game concludes, either with a win or a draw, all buttons are disabled to prevent further moves until the game is reset.

The project serves as a foundation for learning about event-driven programming and creating functional applications using Python's Tkinter library. It also opens up possibilities for further enhancements, such as adding an AI opponent, introducing difficulty levels, or customizing the appearance of the game.

1. Complexifying the Maze

Bigger Grids and Varied Difficulty: Allow users to create bigger mazes or select from varied levels of difficulty. Making the grid larger and increasing obstacles should challenge advanced users more and more exciting mazes.

Law Options: Utilize an assortment of maze generation algorithms-Kruskal's and Recursive Backtracking-for the user to select the complexity and style of the maze.

2. Improved Epistemic and Graphic Elements

Active Menus and Options: Construct a start menu with options for difficulty, maze size, and game parameters. Additionally, pause menus or options to customize settings while playing would enhance the action.

Visuals: Include animations for both the player's movement and that of the maze as it is being generated. Upgrade the graphics, music in the background, and sound effects when the user does anything to further entice the gamer's attention.

3. Scoring and Time-Based Challenges

Scoring System: Introduce a scoring system based on achievements such as finishing time, number of moves, and/or difficulty levels. This will introduce a competitive edge to encourage players to improve their scores.

The Board With Timing: Display a timer along with best scores, or records that show the best time; they can be compared with another player or with themselves.

4. Additional Game Modes

Multiplayer Mode: Complement it with multiplayer mode-a race to be the first to get out from the maze, injecting social and competitive views into the game.

Adding Moving Obstacles, or an enemy in the maze, that AI controls; adding more and increasing the difficulty.

5. Adaptive Pathfinding Algorithms

Heuristic Methods are based on advanced pathing algorithms like Dijkstra's algorithm and so forth, if they could provide optimized routes & make it easy for auto-solving for larger or more complex designed mazes.

Dynamic pathfinding could mean real-time pathfinding that adjusts if the maze element changes such as moving obstacles or regenerating walls.

6. Mobile and Web Versions

Cross-Platform porting would involve the development of versions compatible with mobile devices or browsers so as to reach a wider audience. Pygame, with web and mobile deployment support, could be modified accordingly, or the entire project be recreated using tools like Unity for greater compatibility.

7. Educational Features

Algorithm visualization mode-Presenting the option of visualizing the maze generation and pathfinding algorithms on a step-to-step process; this would thereby make for good educational software for learning about algorithms and data structures.

Maze Creation Mode-Players can create and design mazes with certain shapes and styles and save, share, and solve them.

8. Enhanced Artificial Intelligence and Machine Learning Applications

AI maze solver-Involves developing an AI that can learn and decide how to solve mazes with any of the machine learning techniques and, thus, eventually adapt itself to different maze patterns. Player data analytics-Collect player data on maze-solving techniques, and apply such insights to optimize the designs of mazes for better engagement or for adaptive levels of difficulty.

REFERENCES:

- Tkinter Documentation: Official Python documentation for Tkinter, the library used to create the graphical interface. Available at: https://docs.python.org/3/library/tkinter.html
- Minimax Algorithm: A common algorithm used for decision-making in two-player games. For more information, refer to: https://en.wikipedia.org/wiki/Minimax
- Python Game Development: Tutorials and resources on Python game development with Tkinter. Example: https://realpython.com/python-gui-tkinter/
- Python Programming: General programming principles used in this project, available from Python's official site: https://www.python.org/doc/
- AI in Games: Introduction to AI techniques in gaming, including Tic-Tac-Toe and other strategy games. Available at: https://www.geeksforgeeks.org/mini-max-algorithm-in-game-theory-set-1-tic-tac-toe-game/