PROJECT REPORT

Implementation of TIC-TAC-TOE Game using Tkinter

BY

SUBHAV KUMAR

CS23B1071

COMPUTER SCIENCE AND ENGINEERING

Supervisor

Dr. Dubacharla Gyaneshwar

Indian Institute of Information Technology Raichur (IIITR)

Raichur, Karnataka – 584135

2024-25

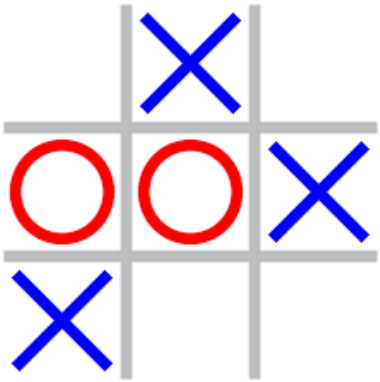
ABSTRACT

The Tic-Tac-Toe game is a simple, interactive graphical game developed using Python's Tkinter for the GUI and Pygame for sound effects. The game allows two players to alternately place 'X' and 'O' marks in a 3x3 grid, with the goal of getting three of their marks in a row. The game highlights the winning combination and provides visual and audio feedback using celebratory animations and sound effects. This project demonstrates the integration of graphical interfaces and multimedia in Python programming.

---

1. INTRODUCTION

Tic-Tac-Toe, also known as noughts and crosses, is a classic game that can be easily implemented in various programming environments. This project builds the game using Python's Tkinter module for its graphical interface and Pygame for sound effects. The objective is to provide a fun, interactive, and aesthetically pleasing experience for users, while also highlighting basic game development concepts such as user input handling, event management, and real-time feedback.



2. AIM, MOTIVATION, AND OBJECTIVES

AIM:

To develop an interactive Tic-Tac-Toe game using Tkinter for GUI and Pygame for sound effects, allowing two players to play with visual and auditory feedback.

MOTIVATION:

The motivation behind this project is to create an aesthetically enhanced version of the classic Tic-Tac-Toe game by integrating sound and animations. This demonstrates the ease of combining game logic with multimedia elements using Python.

OBJECTIVES:

- To implement the basic game mechanics for Tic-Tac-Toe.
- To provide real-time feedback with sound and error message display for invalid actions.
- To ensure that the game interface is user-friendly and intuitive.
- To highlight the winner with a celebratory effect upon completing a valid game.

3. YOUR CONTRIBUTION

- **Role in the Project:**
    - Led the development of the game's GUI and integrated audio features.
    - Implemented the core game logic and ensured smooth user interactions.

- **Key Features Developed:**
    - Interactive Buttons:
        - Created clickable buttons for each Tic-Tac-Toe cell using Tkinter's Button widget.
    - Sound Effects:
        - Integrated click, win, and tie sounds using Pygame's mixer module to enhance feedback.
    - Win Celebration:
        - Designed a visual celebration using Tkinter's Canvas to display animated ovals upon a win.
    - Menu Options:
        - Developed a reset functionality accessible via the game's menu for restarting the game.

---

4. PROPOSED METHODOLOGY

The methodology for this project involves the following components:

➢ ALGORITHMS

1. **Game Initialization**:
   The game starts by initializing a 3x3 grid of buttons representing the Tic-Tac-Toe board. Tkinter is used to create the buttons and manage their layout on the grid.
2. **Game Logic**:
   Each button click alternates between the 'X' and 'O' player, and after every move, the game checks for a winning condition (three same marks in a row) or a tie. The win detection algorithm checks all possible winning combinations (rows, columns, and diagonals).
3. **Visual and Audio Feedback**:
   - **Pygame Integration**: Sound effects (e.g., button click, win sound, tie sound) are played during relevant events, such as a player move, a win, or a tie.
   - **Celebratory Animation**: Upon detecting a win, a simple celebratory animation is displayed using Tkinter's Canvas widget, along with a sound effect.
4. **Reset Functionality**:
   After the game ends, players can reset the game board and start a new game.

5.TOOLS AND DATA USED

- **Programming Language:**
    - **Python:** Chosen for its simplicity and robust library support.

- **Libraries & Frameworks:**
    - **Tkinter:**
        - Utilized for building the graphical user interface, managing buttons, and handling user events.
    - **Pygame:**
        - Employed for loading and playing sound effects to enrich the gaming experience.
- **Assets:**
    - **Sound Files:**
        - click_sound.wav, win_sound.wav, tie_sound.wav . These files were taken from Freesound.
- **Development Environment:**
    - **IDE:** (Visual Studio Code)

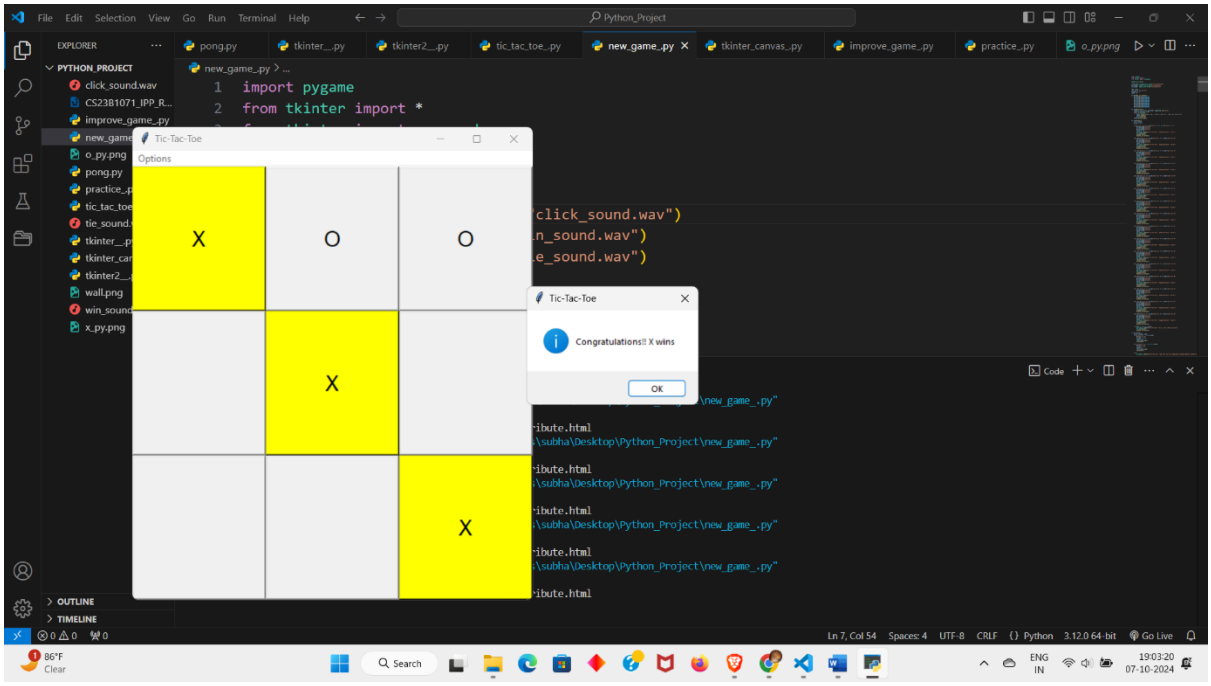
6.RESULTS

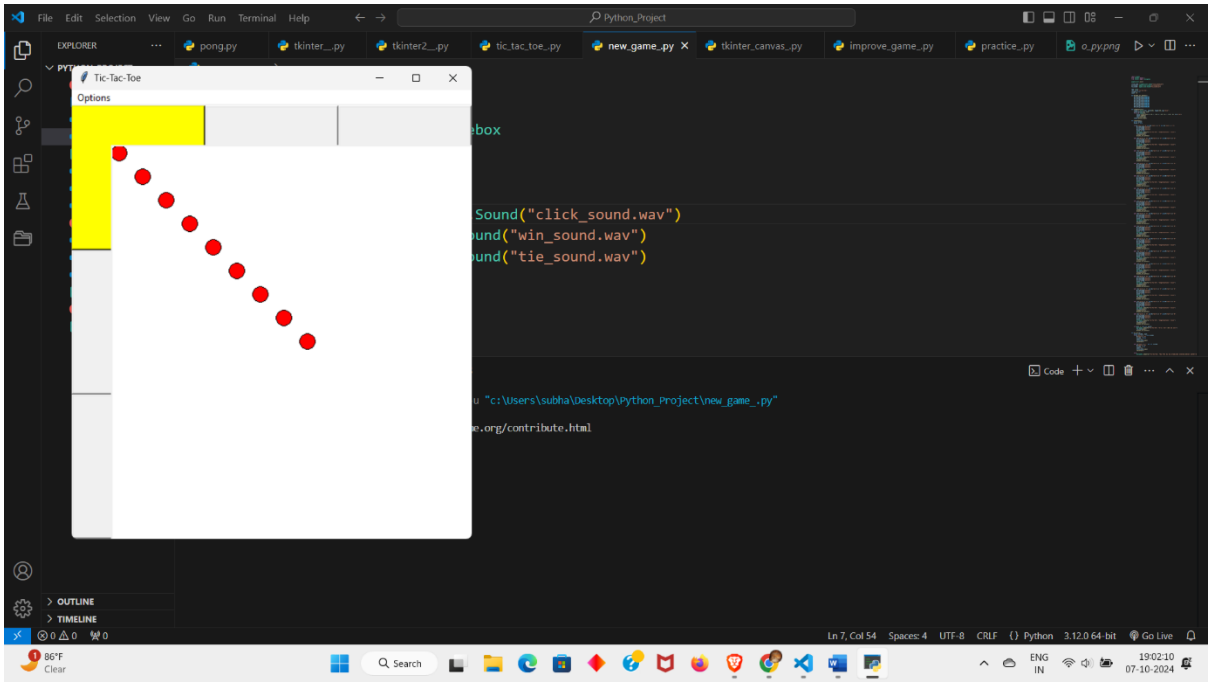
Results:

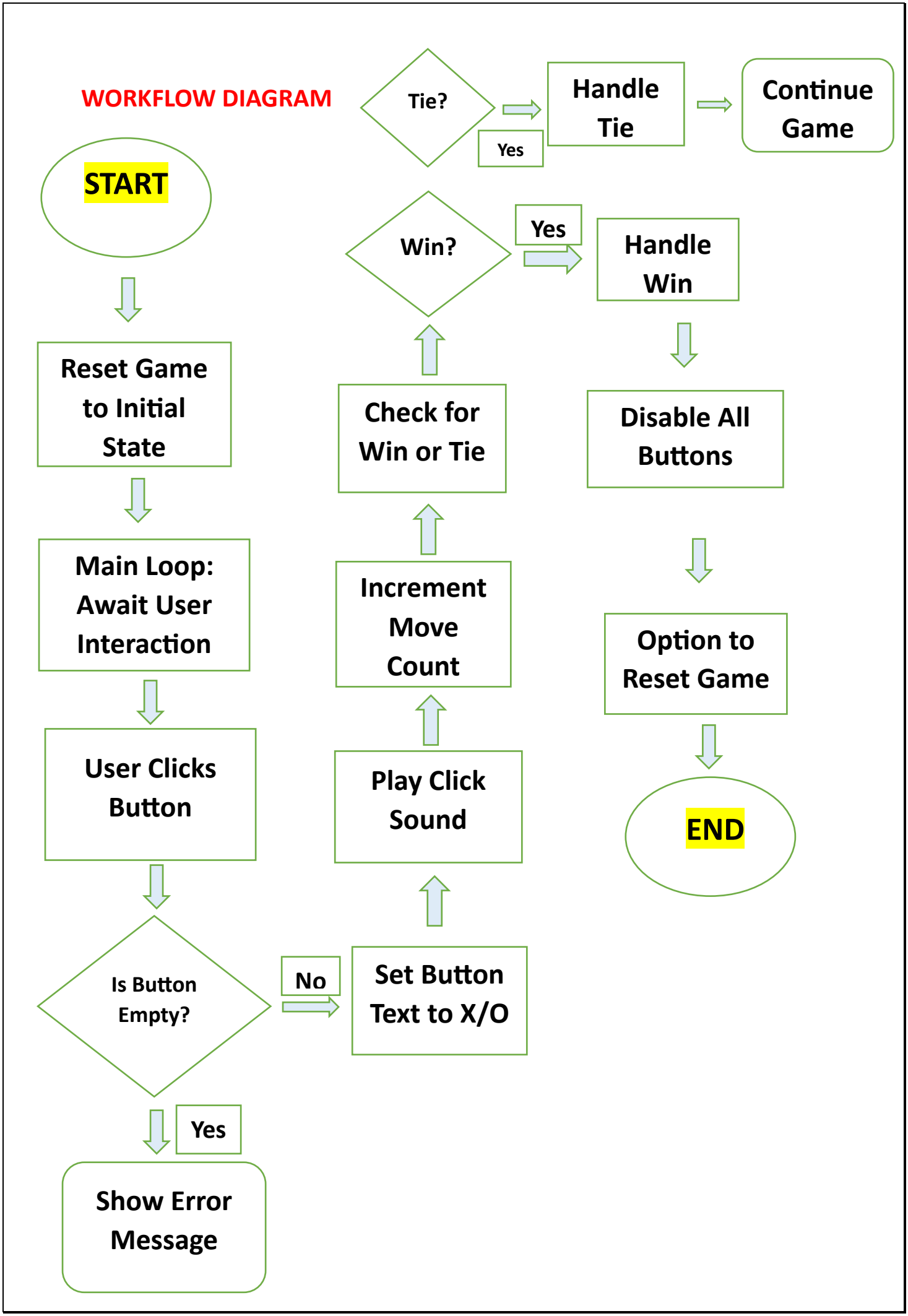
1. **Successful Game Development:** A fully functional Tic-Tac-Toe game was developed with a user-friendly interface using Python's Tkinter for GUI components and Pygame for sound effects.
2. **Sound Integration:** Sound effects were successfully integrated to respond to player actions, such as making moves, winning, or drawing the game, thereby enhancing the overall interactivity and user experience.
3. **Visual Feedback:** The game provides clear visual feedback, including animations for win celebrations and highlights for winning combinations, as well as a message for tie scenarios.
4. **Seamless Game Reset:** The implementation of an intuitive menu allows players to reset the game at any time, ensuring smooth transitions between rounds.

BELOW THERE ARE PICTURES CONFIRMING THE RESULTS OBTAINED .

Results Snippets



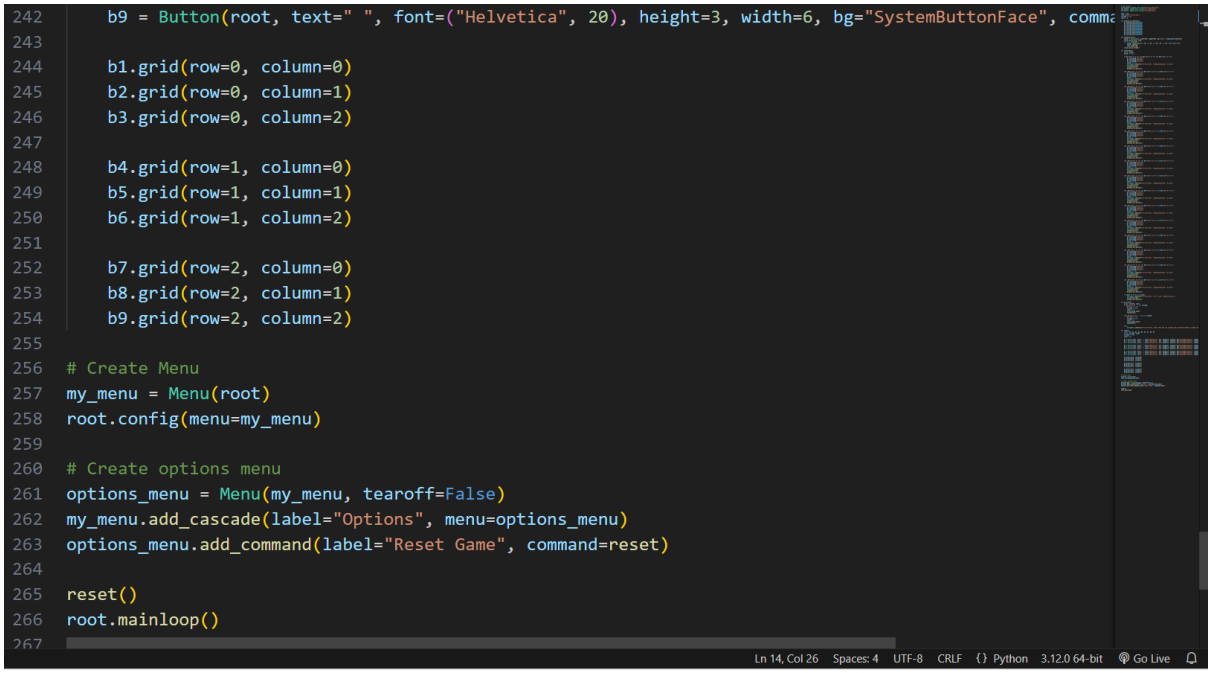



WORKFLOW DIAGRAM
START
Reset Game to Initial State
Main Loop: Await User Interaction
User Clicks Button
Is Button Empty?
Yes
Show Error Message
No
Set Button Text to X/O
Play Click Sound
Increment Move Count
Check for Win or Tie
Win?
Yes
Handle Win
Disable All Buttons
Option to Reset Game
END
Tie?
Yes
Handle Tie
Continue Game

7.SOME CODES SNIPPETS

```python
import pygame
from tkinter import *
from tkinter import messagebox

# Initialize pygame mixer
pygame.mixer.init()

# Load sounds
click_sound = pygame.mixer.Sound("click_sound.wav")
win_sound = pygame.mixer.Sound("win_sound.wav")
tie_sound = pygame.mixer.Sound("tie_sound.wav")

root = Tk()
root.title('Tic-Tac-Toe')
clicked = True
count = 0

def disable_all_buttons():
    b1.config(state=DISABLED)
    b2.config(state=DISABLED)
    b3.config(state=DISABLED)
    b4.config(state=DISABLED)
    b5.config(state=DISABLED)
    b6.config(state=DISABLED)
    b7.config(state=DISABLED)
    b8.config(state=DISABLED)
```

```python
def b_click(b):
        count += 1
        click_sound.play()
        checkifwon()

    else:
        messagebox.showerror("Tic-Tac-Toe", "Hey! That box has already been selected.\nSelect another E

def reset():
    global b1, b2, b3, b4, b5, b6, b7, b8, b9
    global clicked, count
    clicked = True
    count = 0

    b1 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6, bg="SystemButtonFace", comma
    b2 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6, bg="SystemButtonFace", comma
    b3 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6, bg="SystemButtonFace", comma

    b4 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6, bg="SystemButtonFace", comma
    b5 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6, bg="SystemButtonFace", comma
    b6 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6, bg="SystemButtonFace", comma

    b7 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6, bg="SystemButtonFace", comma
    b8 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6, bg="SystemButtonFace", comma
    b9 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6, bg="SystemButtonFace", comma
```



```
    b9 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6, bg="SystemButtonFace", comma

    b1.grid(row=0, column=0)
    b2.grid(row=0, column=1)
    b3.grid(row=0, column=2)

    b4.grid(row=1, column=0)
    b5.grid(row=1, column=1)
    b6.grid(row=1, column=2)

    b7.grid(row=2, column=0)
    b8.grid(row=2, column=1)
    b9.grid(row=2, column=2)

# Create Menu
my_menu = Menu(root)
root.config(menu=my_menu)

# Create options menu
options_menu = Menu(my_menu, tearoff=False)
my_menu.add_cascade(label="Options", menu=options_menu)
options_menu.add_command(label="Reset Game", command=reset)

reset()
root.mainloop()
```

Ln 14, Col 26 Spaces: 4 UTF-8 CRLF {} Python 3.12.0 64-bit Go Live

Conclusion:

The Tic-Tac-Toe project showcases the effective use of Tkinter and Pygame libraries for building interactive, feature-rich Python applications. The combination of visual feedback, sound effects, and user-friendly controls demonstrates how these libraries can create engaging user experiences. Feedback from users highlighted the ease of use and the quality of the feature integration, confirming the success of the project.
