Implementation of Tic-Tac-Toe Game Using Python and Tkinter on Raspberry Pi

Gayatri Tangudu
Student ID - 40221830
Information Systems Engineering
Concordia University
Montreal, Canada
gayatri.tangudu@concordia.ca

December 2023

Abstract

In order to create an interactive Tic-Tac-Toe game, this project aims to combine the capabilities of the Raspberry Pi, Python, and Tkinter. Connecting the dots between physical computing and programming instruction, the Raspberry Pi's smooth Python integration offers hobbyists an adaptable platform. Selecting a game that combines strategic complexity with ease of use, Tic-Tac-Toe pits two players against one another to strategically mark a three-by-three grid with the letters 'X' and 'O.' The project painstakingly puts together the necessary parts, using classes such as TicTacToeBoard and TicTacToeGame for the structural logic and visual design of the game. The game board appears as a dynamic interface using Tkinter widgets, utilizing a model-view-controller architecture. A Tkinter is used to transmit game status updates in real time.Label when using a tkinter-facilitated interactive cell grid.Button widgets that record and decipher player movements. Using Python and Tkinter, an immersive investigation of programming, gaming, and physical computing is encapsulated in this abstract, which is a Tic-Tac-Toe game created on a Raspberry Pi.

Contents

1 Introduction **4**
1.1 Project Overview 4
1.2 Motivation 5
1.3 Objectives 5

2 Background **6**
2.1 Overview of Tic-Tac-Toe 6
2.2 Introduction to Python, Tkinter, and Raspberry Pi 7
2.2.1 Python: 7
2.2.2 Tkinter: 7
2.2.3 Raspberry Pi: 7

3 System Architecture **8**
3.1 Hardware Components 8
3.1.1 Required Hardware 8
3.1.2 Raspberry Pi Board Overview: 8

4 Implementation **10**
4.1 Setting Up the Raspberry Pi 10
4.2 Running Python on the Raspberry Pi 10
4.3 Step by Step Guide for the Game Logic 11
4.3.1 Step 1: Import Required Modules 11
4.3.2 Step 2: Define the TicTacToe Class 11
4.3.3 Step 3: Define Button Click Callback 11
4.3.4 Step 4: Switch Players 12
4.3.5 Step 5: Check for a Winner 12
4.3.6 Step 6 : Check for a Draw 12
4.3.7 Step 7 : Handle Game Over 13
4.3.8 Step 8: Reset the Game 13
4.3.9 Step 9: Main Execution 13

5 Conclusion **14**

6 Future Work **15**
6.1 Future Work:Elevating Tic-Tac-Toe Game Project 15

List of Figures

2.1 Overview of Tic-Tac-Toe. 6

3.1 Raspberry Pi Board Overview. 9

4.1 Import required Modules 11
4.2 Tic Tac Toe Class 11
4.3 Game Board 11
4.4 Button Click Callback 11
4.5 Switch Players 12
4.6 Check for a Winner 12
4.7 Figure Out the Winning Combinations. 12
4.8 Check for a Draw 12
4.9 Game Over 13
4.10 Game Over Message Box 13
4.11 Activate the ”Play Again” 13
4.12 Reset the Game 13
4.13 Clear the board 13
4.14 Main Execution 13

Chapter 1

Introduction

1.1 Project Overview

Among physical computer boards, the Raspberry Pi is a platform that is popular with both amateurs and novices because it is so flexible. It allows for easy interaction with Python and may be used for a variety of applications, from instructional projects to do-it-yourself projects. The goal of this project is to use Python and the Tkinter GUI toolkit to build a Tic-Tac-Toe game by utilizing the Raspberry Pi's capabilities [3].

The simplicity of Tic-Tac-Toe makes it a great starting point for programming aficionados who want to get into game creation. The game basically consists of two players strategically placing their marks on a three-by-three grid in a strategic fight. Each player is given the role of "X" or "O." When three markers are effectively aligned horizontally, vertically, or diagonally, the player wins. When every cell is marked, the game ends graciously in a tie if there isn't a clear winner.

Starting this project entails coordinating important game elements. The TicTacToeBoard class is responsible for creating the game board, while the TicTacToeGame class is in charge of handling the game logic. By utilizing a Tkinter window, the game board transforms into a dynamic interface by utilizing a model-view-controller architecture. This window features two essential components:
Top Display: Providing instantaneous glimpses into the story as it develops in the game.
Grid of Cells: Marking accessible and marked places, providing a visual depiction of players' movements.

Visualization is achieved by skillfully using Tkinter widgets[4]. A tkinter is used to curate the top display.Label, as a tkinter array causes the cell grid to take shape.widgets with buttons. The latter becomes an interactive canvas that, when a player engages it to assess movements and determine the game's conclusion, activates the logic of the game. As the model, the game logic skillfully handles the complexities of the data, rules, and overall game flow.

1.2 Motivation

This project was chosen because it has two important applications: it will enhance teaching, and it will allow me to create a fun game just for the Raspberry Pi platform. The opportunity to dive into the nuances of Python programming and GUI creation using Tkinter is highlighted by the educational value, providing a hands-on learning experience. Investigating the incorporation of hardware—more especially, the Raspberry Pi—enhances the instructional value by offering insights into embedded systems and physical computing.

The attraction of developing an engaging and interactive game simultaneously gives the project a thrilling new dimension. Not only does designing a Tic-Tac-Toe game demonstrate programming skill, but it also gives one the joy of turning abstract concepts into a real, playable product. since of its adaptability, the Raspberry Pi is a perfect platform for this kind of work since it can combine hardware and software components in a way that is both small and easy to use.

Basically, the motivation is the desire to learn and employ programming abilities in a real-world setting, together with the excitement of creating a game that is entertaining and instructive for users of all skill levels and adding to the thriving ecosystem of the Raspberry Pi.

1.3 Objectives

The principal aims of this project are to create a playable and intuitive Tic-Tac-Toe game on the Raspberry Pi platform. These goals are carefully constructed to guarantee a thorough and satisfying project conclusion. The main objectives are as follows:

Functional Implementation: Using the Raspberry Pi's Python and Tkinter capabilities, we want to develop a stable and complete Tic-Tac-Toe game. This entails putting in place a sensible user interface, precise game mechanics, and a coherent game structure.

User-Friendly Design: To put the user's experience first by creating a visually appealing and simple gaming interface. This includes the smooth operation of the game, the unambiguous display of the current state of the game, and the user-friendly interactions that improve gaming in general.

Raspberry Pi Integration: Utilizing the Raspberry Pi platform's hardware capabilities, to smoothly integrate the game with it. In order to do this, the game must be optimized for the Raspberry Pi platform and made to take use of Python's compatibility with the environment.

Accessibility: To ensure that the Tic-Tac-Toe game is accessible to users of varying skill levels. This involves implementing clear instructions, user prompts, and error handling to enhance the overall accessibility and inclusivity of the gaming experience.

By accomplishing these objectives, the project aims to deliver not only a successful implementation of a classic game but also a valuable learning resource for those exploring the realms of programming and game development on the Raspberry Pi platform.

Chapter 2

Background

2.1 Overview of Tic-Tac-Toe

Take a fascinating trip as we explore how to create a Tic-Tac-Toe game in Python. This project demonstrates how to create an interactive graphical user interface (GUI) for our game by using the Tkinter toolkit from the Python standard library. The sample image that goes with this project shows off the intended outcome and gives an idea of how the finished output will look[2].

With an intuitive interface, our Tic-Tac-Toe game perfectly replicates the classic three-by-three gaming board. In turns, players will place their marks strategically on a shared gadget. Located prominently at the top of the window, the game display changes dynamically to show the player in charge at any given time.

The game display will joyfully declare the winner by displaying their mark (X or O) or name when they have won. In addition, to provide even more excitement, the winning combination of cells will be visibly highlighted on the game board at the same time.

The game features a File menu that allows players to reset the game for a rematch or to end their gaming session graciously, which improves user experience.

Let's go on this coding trip together if you're excited by the idea of utilizing Python and Tkinter to create a dynamic and eye-catching Tic-Tac-Toe game!

Figure 2.1: Overview of Tic-Tac-Toe.

2.2 Introduction to Python, Tkinter, and Raspberry Pi

2.2.1 Python:

Python is a well-known, high-level programming language that is easy to read and understand. Because of its syntax, which prioritizes code readability, it is a popular choice for a wide range of domains and is perfect for novices. Python's popularity may be attributed to its large standard library and active community, which allow developers to work effectively on a wide range of jobs. Python continues to be a strong and useful tool for a wide range of applications, from web development to data science and beyond[1].

2.2.2 Tkinter:

Python's default GUI (Graphical User Interface) toolkit, Tkinter, is a natural extension of the language. It's a great option for both new and seasoned developers since it provides an easy-to-use and simple way to create graphical interfaces. Tkinter makes it easier to create interactive programs by offering a variety of widgets for creating user interfaces. Tkinter's cross-platform compatibility guarantees consistent operation of programs across a range of operating systems [4].

2.2.3 Raspberry Pi:

The Raspberry Pi, a small single-board computer, has become a popular platform for professionals, students, and amateurs alike. Its little size, comparable to that of a credit card, belies a wide range of features, such as GPIO (General Purpose Input/Output) pins, which makes it the perfect option for experimentation and hardware projects. The Raspberry Pi, which runs on many operating systems, enables a wide range of applications, including robotics and home automation. Its versatility, affordability, and accessibility have made it a game-changer for physical computing and embedded systems education and implementation. The combination of Python, Tkinter, and Raspberry Pi creates a powerful ecosystem for creating interesting projects that combine hardware and software elements in a seamless manner [3].

Chapter 3

System Architecture

3.1 Hardware Components

3.1.1 Required Hardware

The following hardware is required for the initial setup of your Raspberry Pi :

Monitor We will need a monitor during the initial setup and configuration of the operating system.

microSD Card Raspberry Pi uses a microSD card to store the operating system and files.

Keyboard and Mouse A USB keyboard and mouse are required during the initial setup of the Raspberry Pi. Once the setup is complete, we can switch to using Bluetooth versions of these peripherals.

HDMI Cables and Power Supply: We need an HDMI cable to connect the Raspberry Pi to a monitor.The Raspberry Pi uses a USB connection to power the board. Again, different Raspberry Pi models have different USB connection and power requirements.

3.1.2 Raspberry Pi Board Overview:

The following parts are present on the Raspberry Pi 3 B+ board:

General-purpose input-output pins: The Raspberry Pi is connected to electronic parts via these pins.

Ethernet port: The Raspberry Pi is connected to a wired network using this connector. In addition, the Raspberry Pi features integrated Bluetooth and Wi-Fi for wireless connectivity.

Two USB 3.0 and two USB 2.0 ports: These USB ports are used to connect peripherals like a keyboard or mouse. The two black ports are USB 2.0 and the two blue ports are USB 3.0.

AV jack: This AV jack allows you to connect speakers or headphones to the Raspberry Pi.

Camera Module port: This port is used to connect the official Raspberry Pi Camera Module, which enables the Raspberry Pi to capture images.

HDMI ports: These HDMI ports connect the Raspberry Pi to external monitors. The Raspberry Pi 4 features two micro HDMI ports, allowing it to drive two separate monitors at the same time.

USB power port: This USB port powers the Raspberry Pi. The Raspberry Pi 4 has a USB Type-C port, while older versions of the Pi have a micro-USB port.

External display port: This port is used to connect the official seven-inch Raspberry Pi touch display for touch-based input on the Raspberry Pi.

microSD card slot (underside of the board): This card slot is for the microSD card that contains the Raspberry Pi operating system and files.

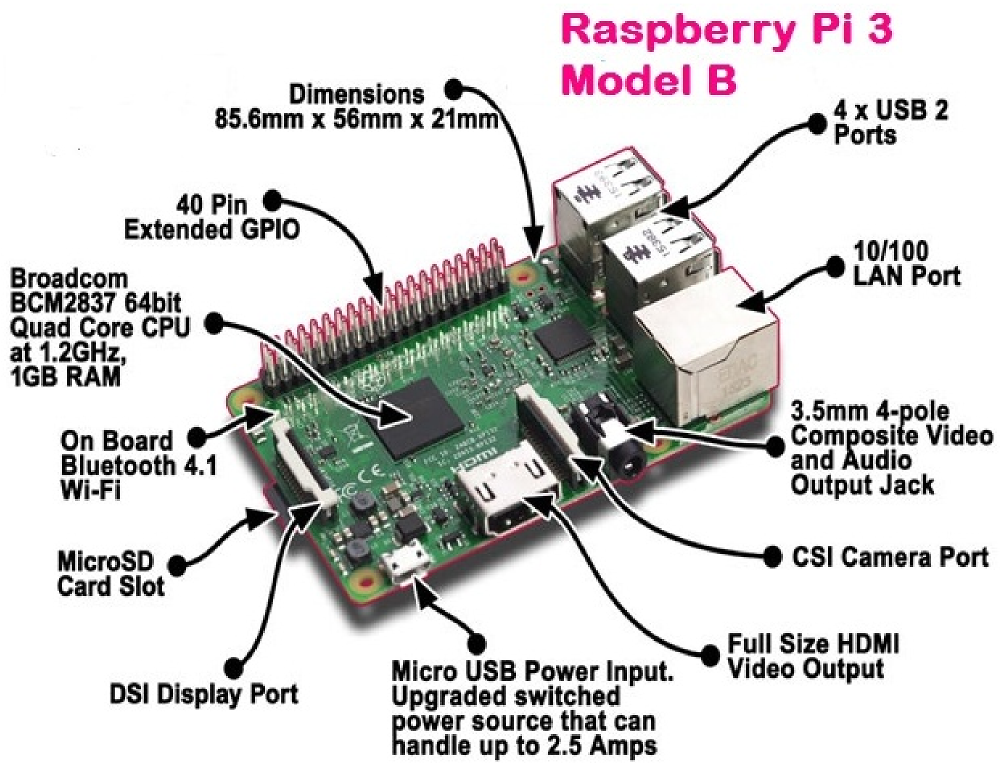

Raspberry Pi 3 Model B
Dimensions 85.6mm x 56mm x 21mm
4 x USB 2 Ports
40 Pin Extended GPIO
10/100 LAN Port
Broadcom BCM2837 64bit Quad Core CPU at 1.2GHz, 1GB RAM
On Board Bluetooth 4.1 Wi-Fi
3.5mm 4-pole Composite Video and Audio Output Jack
MicroSD Card Slot
CSI Camera Port
DSI Display Port
Micro USB Power Input. Upgraded switched power source that can handle up to 2.5 Amps
Full Size HDMI Video Output

Figure 3.1: Raspberry Pi Board Overview.

Chapter 4

Implementation

4.1 Setting Up the Raspberry Pi

This step is to connect everything and set up the operating system now that the microSD card and other components are ready. First, let's connect each and every peripheral:

1. Place the microSD card into the Raspberry Pi's bottom card slot.
2. Use any one of the four USB ports to connect the mouse and keyboard.
3. Using an HDMI cable designed specifically for your Raspberry Pi model. Connect a monitor to one of the HDMI ports.
4. Link a power source to the USB power outlet.

After connecting the peripherals, turn on your Raspberry Pi to set up the operating system [3].

Setup Wizard: Upon initial boot, Raspbian offers an installation process that assists with setting up your Wi-Fi network, password, locale, and operating system update. Proceed and carry out these actions as directed[3].

We may start learning Python on the Raspberry Pi by restarting the operating system when we have finished the instructions!

4.2 Running Python on the Raspberry Pi

Python is a first-class citizen on the Raspberry Pi, which is one of the greatest things about working with it. Python was chosen as the primary language by the Raspberry Pi Foundation due to its strength, adaptability, and user-friendliness. Raspbian ships with Python preloaded, so we'll be good to go right away [3].

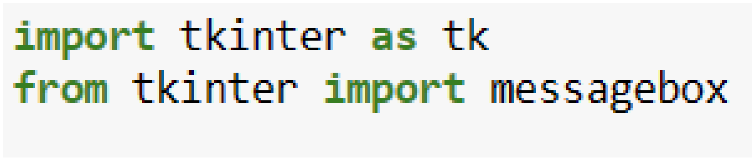
When it comes to programming Python on the Raspberry Pi, we have a wide range of choices. Throughout the project, we will utilize the widely-liked option is the Mu editor.

4.3 Step by Step Guide for the Game Logic

Let's break down the code into steps to understand its implementation:

4.3.1 Step 1: Import Required Modules

We will utilize a regular Python installation to do this project. Since there is no requirement for an external reliance, there is no need to construct a virtual environment. 'Tkinter' is the only package we will require, and it is included in the Python standard library[2]. And import 'messagebox' for displaying pop-up messages[2].

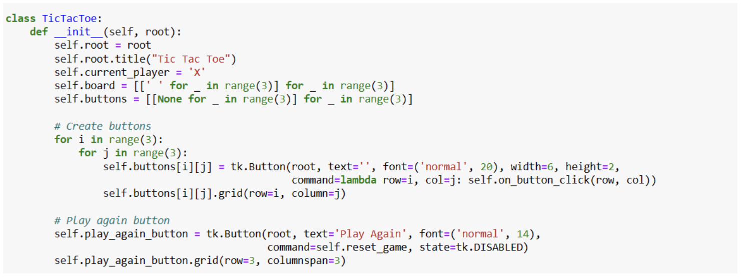


```
import tkinter as tk
from tkinter import messagebox
```

Figure 4.1: Import required Modules

4.3.2 Step 2: Define the TicTacToe Class

Now we have initialization method with the main window as a parameter.Set window title, initialize the current player, and create the game board. Create buttons and set their properties. Create the "Play Again" button and set its properties.



```
class TicTacToe:
    def __init__(self, root):
        self.root = root
        self.root.title("Tic Tac Toe")
        self.current_player = 'X'
        self.board = [[' ' for _ in range(3)] for _ in range(3)]
        self.buttons = [[None for _ in range(3)] for _ in range(3)]

        # Create buttons
        for i in range(3):
            for j in range(3):
                self.buttons[i][j] = tk.Button(root, text='', font=('normal', 20), width=6, height=2,
                                               command=lambda row=i, col=j: self.on_button_click(row, col))
                self.buttons[i][j].grid(row=i, column=j)

        # Play again button
        self.play_again_button = tk.Button(root, text='Play Again', font=('normal', 14),
                                           command=self.reset_game, state=tk.DISABLED)
        self.play_again_button.grid(row=3, columnspan=3)
```

Figure 4.2: Tic Tac Toe Class

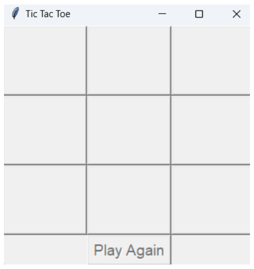
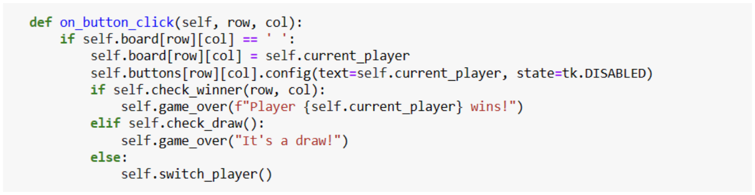


Figure 4.3: Game Board

4.3.3 Step 3: Define Button Click Callback

We will now notice the Callback function when we touch a button in this stage. Check to see if the chosen cell is vacant, update the board, then search for a winner or draw. Indicate when the game is over, turn off the button that was clicked, and switch players if necessary.



```
def on_button_click(self, row, col):
    if self.board[row][col] == ' ':
        self.board[row][col] = self.current_player
        self.buttons[row][col].config(text=self.current_player, state=tk.DISABLED)
        if self.check_winner(row, col):
            self.game_over(f"Player {self.current_player} wins!")
        elif self.check_draw():
            self.game_over("It's a draw!")
        else:
            self.switch_player()
```

Figure 4.4: Button Click Callback

4.3.4 Step 4: Switch Players

We will now create the code to switch the player that is currently in the game between 'X' and 'O'.

```
def switch_player(self):
    self.current_player = 'O' if self.current_player == 'X' else 'X'
```

Figure 4.5: Switch Players

4.3.5 Step 5: Check for a Winner

This section describes the fundamental principles and procedures that underpin the game of Tic Tac Toe. A winning combination is represented by examining the row, column, and diagonals.

You may now consider processing the movements made by the players after setting up your game board[2].

```
def check_winner(self, row, col):
    # Check row
    if all(self.board[row][i] == self.current_player for i in range(3)):
        return True
    # Check column
    if all(self.board[i][col] == self.current_player for i in range(3)):
        return True
    # Check diagonals
    if row == col and all(self.board[i][i] == self.current_player for i in range(3)):
        return True
    if row + col == 2 and all(self.board[i][2 - i] == self.current_player for i in range(3)):
        return True
    return False
```

Figure 4.6: Check for a Winner

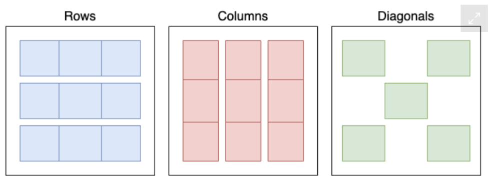


Figure 4.7: Figure Out the Winning Combinations.

4.3.6 Step 6 : Check for a Draw

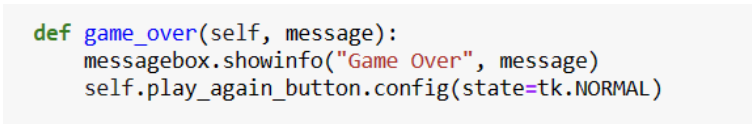
We will now examine every cell on the board to determine whether the game is a draw.

```
def check_draw(self):
    return all(self.board[i][j] != ' ' for i in range(3) for j in range(3))
```

Figure 4.8: Check for a Draw

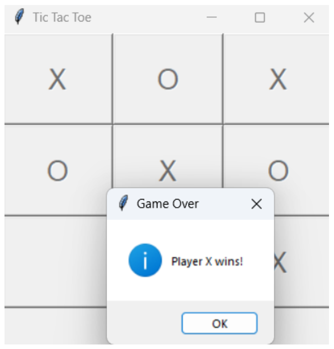
4.3.7 Step 7 : Handle Game Over

Right now Show the game outcome in a pop-up message. Activate the "Play Again" icon.



```python
def game_over(self, message):
    messagebox.showinfo("Game Over", message)
    self.play_again_button.config(state=tk.NORMAL)
```

Figure 4.9: Game Over



Player X wins!

Figure 4.10: Game Over Message Box

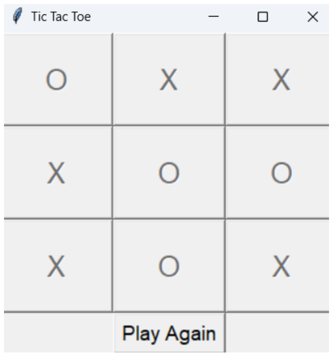
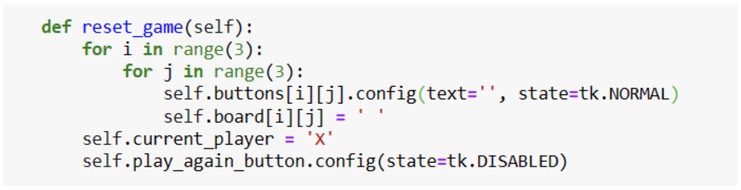


Figure 4.11: Activate the "Play Again"

4.3.8 Step 8: Reset the Game

A function that clears the board, activates buttons, and resets the player is required to reset the game. The following code sample illustrates this.



```python
def reset_game(self):
    for i in range(3):
        for j in range(3):
            self.buttons[i][j].config(text='', state=tk.NORMAL)
            self.board[i][j] = ' '
    self.current_player = 'X'
    self.play_again_button.config(state=tk.DISABLED)
```

Figure 4.12: Reset the Game

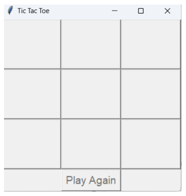
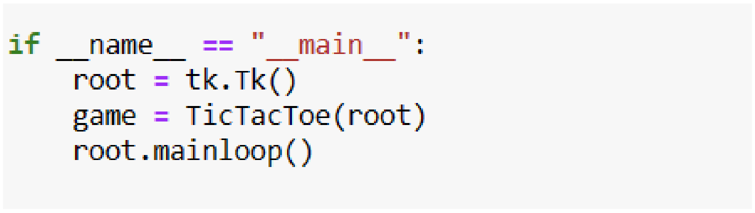


Figure 4.13: Clear the board

4.3.9 Step 9: Main Execution

The main window will now be created, and the TicTacToe game will be initialized.



```python
if __name__ == "__main__":
    root = tk.Tk()
    game = TicTacToe(root)
    root.mainloop()
```

Figure 4.14: Main Execution

Chapter 5

Conclusion

In this practical project, we have used the dynamic combination of Python, Tkinter, and Raspberry Pi gives us the ability to create games and opens us a world of creative possibilities. This project acts as a springboard, launching us into an innovative realm where hardware and code collide and laying the groundwork for fascinating future projects.

Along the way, we've refined a few crucial abilities:

- **Implementation of Game Logic:** Proficiency in implementing the complex logic underlying the beloved game tic tac toe, therefore strengthening our foundation in Python game programming.
- **Tkinter GUI Construction:** Skill in using Tkinter to create a visually appealing and user-friendly gaming board that improves the user experience overall with a well-designed graphical user interface.
- **Logic-GUI Integration:** The ability to seamlessly combine the logic of a game with its graphical user interface is essential for developing interactive applications that have user actions that make sense in relation to the underlying program logic.
- **Exploration with Raspberry Pi:** By delving into the world of Raspberry Pi, we have discovered how to run and modify our code for this flexible operating system. This experience extends our knowledge base and shows how flexible our abilities are in a range of hardware configurations.
- **Proficiency with Mu Editor:** Gaining knowledge of the Mu editor for Raspberry Pi, an essential tool for developing and exploring this distinctive environment.

Chapter 6

Future Work

6.1 Future Work:Elevating Tic-Tac-Toe Game Project

Although the existing Tic-Tac-Toe implementation is a good starting point, there are several intriguing potential improvements that may be made in the future:

- **Customizable Board Sizes:** Permit users to choose the size of the game board and add different degrees of difficulty to their play experience.
- **Computer Opponent with Difficulty Levels:** Introduce an AI opponent that may be adjusted in difficulty to provide gamers with a demanding solo experience.
- **Online Multiplayer Mode:** Include an online multiplayer component to broaden the game's appeal and encourage participation and competitiveness across borders.
- **Alternate Markers and Themes:** Add variation to the game by offering users the ability to customize it to their likings with changeable markers and theme options.
- **Persistent User Profiles:** Establish user profiles with customized data and settings to promote long-term interaction and a feeling of continuity.

With these improvements, the game Tic-Tac-Toe will become a dynamic, feature-rich program that provides players with a more engaging and customized gaming experience.

Bibliography

[1] Learning python — the hitchhiker's guide to python. https://docs.python-guide.org/intro/learning/, 2023. Accessed: 2023-12-03.

[2] Real Python. Build a tic-tac-toe game with python and tkinter – real python. https://realpython.com/tic-tac-toe-python/#project-overview, 2023. Accessed: 2023-12-03.

[3] Real Python. Build physical projects with python on the raspberry pi – real python. https://realpython.com/python-raspberry-pi/, 2023. Accessed: 2023-12-03.

[4] Real Python. Python gui programming with tkinter – real python. https://realpython.com/python-gui-tkinter/, 2023. Accessed: 2023-12-03.