TIC-TAC-TOE

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology/Master of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

KG14

B. Sai Moesha (AP21110011261)

B. Vijaya Sravya (AP21110011224)

S. Jashvitha (AP2110010673)

G. Tarun Sai (AP21110010690)

Under the Guidance of

(Mohammad Miskeen Ali)

of Computer Science and Engineering

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[December, 2022]

CERTIFICATE

This is to certify that the work present in this Project entitled "**TIC-TAC-TOE**" has been carried out by **B.Sai Moesha (AP21110011261), B.Vijaya Sravya (AP21110011224), S.Jashvitha (AP21110010673), G. Tarun Sai (AP21110010690) students** of Department of Computer Science and Engineering, SRM University, in partial fulfilment of the requirement for the degree of "**B.Tech(CSE)" carried** out by her/his during the academic year 2021-2022.

Signature of the Supervisor	Signature of Head of the Dept.
MOHAMMAD MISKEEN ALI	**JATINDRA KUMAR DASH**

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success.

We are extremely grateful and express our profound gratitude and indebtedness to our project guide, **Mohammad Miskeen Ali**, Lecturer, Department of Computer Science & Engineering, SRM University, Andhra Pradesh, for his kind help and for giving us the necessary guidance and valuable suggestions in completing this project work.

B. Vijaya sravya (AP211100224)

B. Sai Moesha (AP21110011261)

S. Jashvitha (AP21110010673)

G. Tarun Sai(AP21110010690)

Table of contents:

1. Abstract:

Tic-Tac-Toe game can be played by two players where the square block (3 x 3) can be filled with a cross (X) or a circle (O). The game will toggle between the players by giving the chance for each player to mark their move. When one of the players make a combination of 3 same markers in a horizontal, vertical or diagonal line the circuit will display which player has won, whether X or O. The purpose of this documentation is to capture all requirements by which the user can play a game TIC-TAC-TOE in graphical user interface as well as they can actually develop a logic that what is actually happening.

2. Introduction:

One of the most universally played childhood games is TIC TAC TOE. An interactive TIC TAC TOE game is developed using GUI in python where two players will be able to play against each other in a suitable GUI by using proper mouse movements. This game will start by showing a simple display, prompt the user for a move and then prints the new board. The board looks like a large hash symbol (#) with nine slots that can each contain an X, a O, or a blank. There are two players, the X player and the O player. By default, players (the O player) take the initiative. game will end in two situations: a win, when one player gets three in a row, horizontally, vertically or diagonally. A draw if neither of them has won when there are no remaining places to choose and no one can logically create a straight line with its symbol.

3. Methodology:

Game is designed by using graphical user interfaces in Python by importing the tknter package.

Import all necessary libraries such as message box function to display message on screen.

variable called sign is used to decide which player goes first.

Global variables are used in program so that each part of program is accessible without needing extra lines of code for each part.

4. Proposed system

To overcome the drawbacks of the existing system, the proposed system has been evolved. Project aims to reduce the paper work and saving time to generate accurate results from the player's perspective. game has been made user friendly with proper use of graphical interface. The system can able to provide hundreds of TIC TAC TOE game without any interruption.

5. Requirements:

5.1 Hardware requirement:

Minimum RAM: - 1GB

Minimum Hard Disk: - 128GB

Processor: - Intel Pentium 4(1.50 GHz) or above

5.2 **Software requirement:**

Operating System: - Support for both LINUX and WINDOWS users

Back End: - Python 3.6.0 Interpreter

Front End Language: - Python3

Front Design: - Tkinterface

6. Algorithm:

We will now discuss the algorithm to write the code. This algorithm will help you to write code in any programming language of your choice. Let’s see how it’s done.

Create a board using a 2-dimensional array and initialize each element as empty.

You can represent empty using any symbol you like. Here, we are going to use a hyphen. '-'.

Write a function to check whether the board is filled or not.

Iterate over the board and return false if the board contains an empty sign or else return true.

Write a function to check whether a player has won or not.

We have to check all the possibilities that we discussed in the previous section.

Check for all the rows, columns, and two diagonals.

Write a function to show the board as we will show the board multiple times to the users while they are playing.

Write a function to start the game.

Select the first turn of the player randomly.

Write an infinite loop that breaks when the game is over (either win or draw).

Show the board to the user to select the spot for the next move.

Ask the user to enter the row and column number.

Update the spot with the respective player sign.

Check whether the current player won the game or not.

If the current player won the game, then print a winning message and break the infinite loop.

Next, check whether the board is filled or not.

If the board is filled, then print the draw message and break the infinite loop.

Finally, show the user the final view of the board.

You may be able to visualize what’s happening. Don’t worry, even if you didn’t understand it completely. You will get more clarity once you see the code.

So, let's jump into the code section. I assume you have Python installed on your PC to try the code.

7. Implementation:

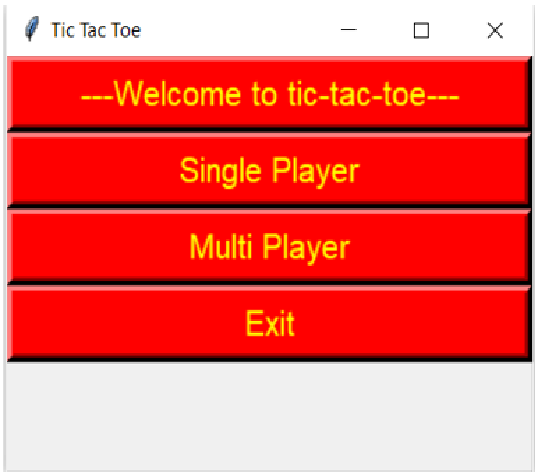
```
# Tic Tac Toe game with GUI
# using tkinter
# importing all necessary libraries
import random
import tkinter
from tkinter import *
from functools import partial
from tkinter import messagebox
from copy import deepcopy

# sign variable to decide the turn of which player
sign = 0

# Creates an empty board
global board
board = [[" " for x in range(3)] for y in range(3)]

# Check l(O/X) won the match or not
# according to the rules of the game
def winner(b, l):
    return ((b[0][0] == l and b[0][1] == l and b[0][2] == l) or
            (b[1][0] == l and b[1][1] == l and b[1][2] == l) or
            (b[2][0] == l and b[2][1] == l and b[2][2] == l) or
            (b[0][0] == l and b[1][0] == l and b[2][0] == l) or
            (b[0][1] == l and b[1][1] == l and b[2][1] == l) or
            (b[0][2] == l and b[1][2] == l and b[2][2] == l) or
            (b[0][0] == l and b[1][1] == l and b[2][2] == l) or
```

```
        (b[0][2] == l and b[1][1] == l and b[2][0] == l))

# Configure text on button while playing with another player
def get_text(i, j, gb, l1, l2):
    global sign
    if board[i][j] == ' ':
        if sign % 2 == 0:
            l1.config(state=DISABLED)
            l2.config(state=ACTIVE)
            board[i][j] = "X"
        else:
            l2.config(state=DISABLED)
            l1.config(state=ACTIVE)
            board[i][j] = "O"
        sign += 1
        button[i][j].config(text=board[i][j])
    if winner(board, "X"):
        gb.destroy()
        box = messagebox.showinfo("Winner", "Player 1 won the match")
    elif winner(board, "O"):
        gb.destroy()
        box = messagebox.showinfo("Winner", "Player 2 won the match")
    elif(isfull()):
        gb.destroy()
        box = messagebox.showinfo("Tie Game", "Tie Game")
```

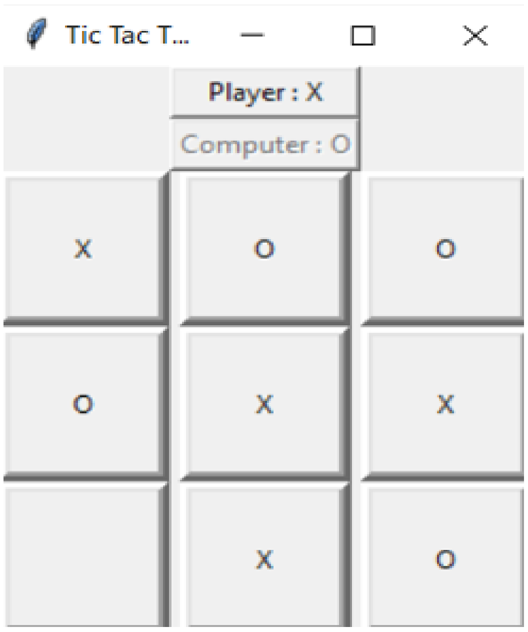
```
# Check if the player can push the button or not
def isfree(i, j):
    return board[i][j] == " "

# Check the board is full or not



def isfull():
    flag = True
    for i in board:
        if(i.count(' ') > 0):
            flag = False
    return flag

# Create the GUI of game board for play along with another player
def gameboard_pl(game_board, l1, l2):
    global button
    button = []
    for i in range(3):
        m = 3+i
        button.append(i)
        button[i] = []
        for j in range(3):
            n = j
            button[i].append(j)
            get_t = partial(get_text, i, j, game_board, l1, l2)
            button[i][j] = Button(
                game_board, bd=5, command=get_t, height=4, width=8)
            button[i][j].grid(row=m, column=n)
    game_board.mainloop()

# Decide the next move of system
def pc():
    possiblemove = []
    for i in range(len(board)):
        for j in range(len(board[i])):
            if board[i][j] == ' ':
                possiblemove.append([i, j])
    move = []
    if possiblemove == []:
        return
    else:
        for let in ['O', 'X']:
            for i in possiblemove:
                boardcopy = deepcopy(board)
                boardcopy[i[0]][i[1]] = let
                if winner(boardcopy, let):
                    return i
        corner = []
        for i in possiblemove:
            if i in [[0, 0], [0, 2], [2, 0], [2, 2]]:
                corner.append(i)
        if len(corner) > 0:
            move = random.randint(0, len(corner)-1)
            return corner[move]
        edge = []
        for i in possiblemove:
            if i in [[0, 1], [1, 0], [1, 2], [2, 1]]:
                edge.append(i)
        if len(edge) > 0:
            move = random.randint(0, len(edge)-1)
            return edge[move]

# Configure text on button while playing with system
def get_text_pc(i, j, gb, l1, l2):
    global sign
    if board[i][j] == ' ':
        if sign % 2 == 0:
            l1.config(state=DISABLED)
            l2.config(state=ACTIVE)
            board[i][j] = "X"
        else:
            button[i][j].config(state=ACTIVE)
            l2.config(state=DISABLED)
            l1.config(state=ACTIVE)
            board[i][j] = "O"
        sign += 1
        button[i][j].config(text=board[i][j])
    x = True
    if winner(board, "X"):
        gb.destroy()
        x = False
        box = messagebox.showinfo("Winner", "Player won the match")
    elif winner(board, "O"):
        gb.destroy()
        x = False
        box = messagebox.showinfo("Winner", "Computer won the match")
    elif(isfull()):
        gb.destroy()
        x = False
        box = messagebox.showinfo("Tie Game", "Tie Game")
    if(x):
        if sign % 2 != 0:
            move = pc()
            button[move[0]][move[1]].config(state=DISABLED)
            get_text_pc(move[0], move[1], gb, l1, l2)

# Create the GUI of game board for play along with system
def gameboard_pc(game_board, l1, l2):
    global button
    button = []
    for i in range(3):
        m = 3+i
        button.append(i)
        button[i] = []
        for j in range(3):
            n = j
            button[i].append(j)
            get_t = partial(get_text_pc, i, j, game_board, l1, l2)
            button[i][j] = Button(
            game_board, bd=5, command=get_t, height=4, width=8)
        button[i][j].grid(row=m, column=n)
    game_board.mainloop()

# Initialize the game board to play with system
def withpc(game_board):
    game_board.destroy()
    game_board = Tk()
    game_board.title("Tic Tac Toe")
    l1 = Button(game_board, text="Player : X", width=10)
    l1.grid(row=1, column=1)
    l2 = Button(game_board, text="Computer : O",
            width=10, state=DISABLED)
    l2.grid(row=2, column=1)
    gameboard_pc(game_board, l1, l2)

# Initialize the game board to play with another player
def withplayer(game_board):
    game_board.destroy()
    game_board = Tk()
    game_board.title("Tic Tac Toe")
    l1 = Button(game_board, text="Player 1 : X", width=10)
    l1.grid(row=1, column=1)
    l2 = Button(game_board, text="Player 2 : O",
            width=10, state=DISABLED)

    l2.grid(row=2, column=1)
    gameboard_pl(game_board, l1, l2)
```

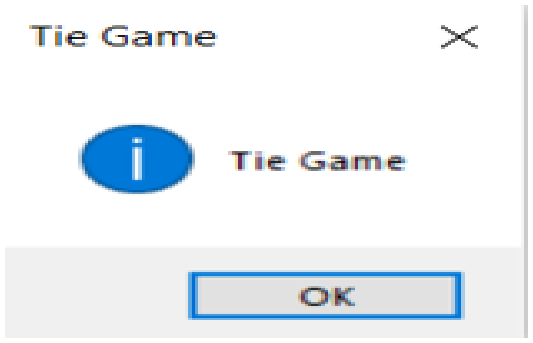
```
# main function
def play():
    menu = Tk()
    menu.geometry("250x250")
    menu.title("Tic Tac Toe")
    wpc = partial(withpc, menu)
    wpl = partial(withplayer, menu)

    head = Button(menu, text="---Welcome to tic-tac-toe---",
                  activeforeground='red',
                  activebackground="yellow", bg="red",
                  fg="yellow", width=500, font='summer', bd=5)

    B1 = Button(menu, text="Single Player", command=wpc,
                activeforeground='red',
                activebackground="yellow", bg="red",
                fg="yellow", width=500, font='summer', bd=5)

    B2 = Button(menu, text="Multi Player", command=wpl, activeforeground='red',
                activebackground="yellow", bg="red", fg="yellow",
                width=500, font='summer', bd=5)

    B3 = Button(menu, text="Exit", command=menu.quit, activeforeground='red',
                activebackground="yellow", bg="red", fg="yellow",
                width=500, font='summer', bd=5)
    head.pack(side='top')
    B1.pack(side='top')
    B2.pack(side='top')
    B3.pack(side='top')
    menu.mainloop()

# Call main function
if _name_ == '_main_':
    play()
```

8. Output:



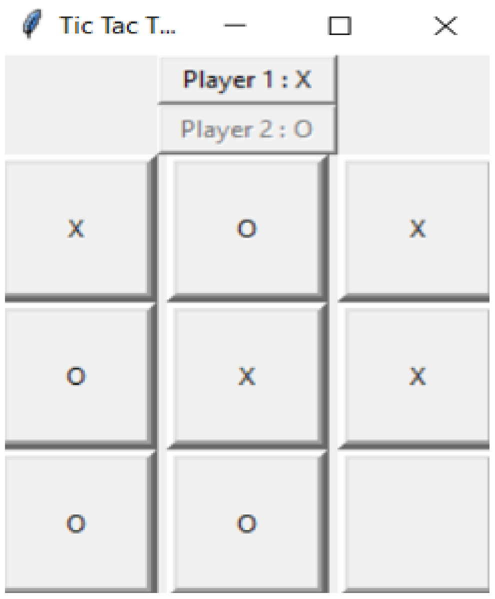
SINGLE PLAYER:

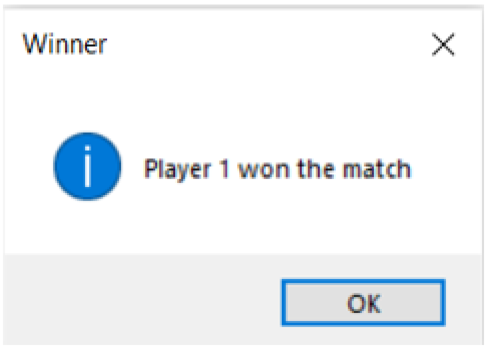



Tie Game

Tie Game

MULTI PLAYER:




Winner

Player 1 won the match

9. Future Work:

- Allow the players to choose the board size.
- Introduction of difficulty levels will make the game more interesting.
- One of the other enhancements could be that users can create their profiles and save their scores.
- Only mouse interface is implemented but it's good to activate the keyboard in the game.
- Introducing Bluetooth communication channel between two players. - report
- players should have option to edit their names-report

10. Conclusion:

The Tic Tac Toe game is most familiar among all the age groups ,used for entertaining also used in many technical events. Intelligence can be a property of any purpose-driven decision maker. This basic idea has been suggested many times. An algorithm of playing Tic Tac Toe has been presented and tested that works in efficient way. Overall, the system works without any bugs. Allow players to choose the mode of the game. Increasing the size of this game would create a huge time complexity issue with the same algorithm and techniques, for which other logics must be further researched.

11. References:

1. https://www.geeksforgeeks.org/
2. https://realpython.com/tic-tac-toe-python/
3. https://devdojo.com/jothin-kumar/tic-tac-toe-with-python-tkinter-part-1
4. https://geekflare.com/

5. Sixth Edition Herbert Schildt