Project Report

for

Tic-Tac-Toe

Version 2.0

Prepared by Kiran Qaiser, Hafsah Shahbaz

Forman Christian College (A Chartered university)

21/01/2022

Report

Source:

https://www.geeksforgeeks.org/tic-tac-toe-game-with-gui-using-tkinter-in-python/

Changes:

We have changed the grid size of the game it was a 3x3 grid and we changed it to 4x4. While doing that we also had to change the logic of the game as it was set for a 3x3 matrix. The view of the game was also changed. All the changes are described briefly under:

- ## Color Change:

```
# 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()
```

Old Code for buttons

```
# Create the GUI of game board for play along with another player
def gameboard_pl(game_board, l1, l2):
    global button
    myFont = font.Font(family='Helvetica', size=20, weight='bold') # changed here
    button = []
    for i in range(4):
        m = 3+i
        button.append(i)
        button[i] = []
        for j in range(4):
            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,activeforeground = 'pink',
                  activebackground = "Blue", bg = "purple",
                  fg = "red",font=myFont) # changd here
            button[i][j].grid(row=m, column=n)
    game_board.mainloop()
```

New Code for buttons

So we added the *activeforgrouand and activebackground* parameter to change the background and foreground of the color.

- **Size change of X&O:**

```
# 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()
```

Old Code for buttons

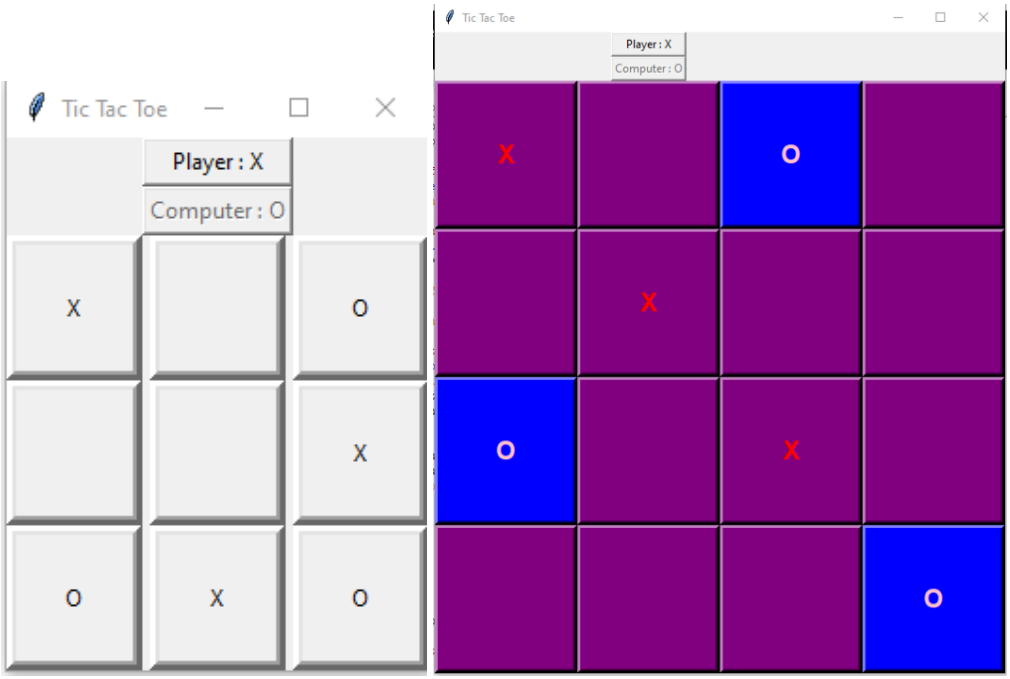
```
# Create the GUI of game board for play along with another player
def gameboard_pl(game_board, l1, l2):
    global button
    myFont = font.Font(family='Helvetica', size=20, weight='bold') # changed here
    button = []
    for i in range(4):
        m = 3+i
        button.append(i)
        button[i] = []
        for j in range(4):
            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,activeforeground = 'pink',
                  activebackground = "Blue", bg = "purple",
                  fg = "red",font=myFont) # changd here
            button[i][j].grid(row=m, column=n)
    game_board.mainloop()
```

New Code for buttons

We made *myfont* so that we could change the size of x and o and make them bolder so that it can be viewed properly.

- **3x3 to 4x4:**

  We changed the matrix for 3x3 to 4x4



Old Version *New version*

So for changing the game size we had to change the range in boardsize from 3 to 4.

Cause of which we also needed to change the winner algorithm and the possible moves the pc could take

```
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))
```

Old Code

```
def winner(b, l):
    # print(l)
     print(b[0][2])
     print(b[3][3])
     return ((b[0][0] == l and b[0][1] == l and b[0][2] == l and b[0][3] == l) or
             (b[1][0] == l and b[1][1] == l and b[1][2] == l and b[1][3] == l) or
             (b[2][0] == l and b[2][1] == l and b[2][2] == l and b[2][3] == l) or
             (b[3][0] == l and b[3][1] == l and b[3][2] == l and b[3][3] == l) or
             (b[0][0] == l and b[1][0] == l and b[2][0] == l and b[3][0] == l) or
             (b[0][1] == l and b[1][1] == l and b[2][1] == l and b[3][1] == l) or
             (b[0][2] == l and b[1][2] == l and b[2][2] == l and b[3][2] == l) or
             (b[0][0] == l and b[1][1] == l and b[2][2] == l and b[3][3] == l) or
             (b[0][3] == l and b[1][2] == l and b[2][1] == l and b[3][0] == l) or
             (b[0][3] == l and b[1][3] == l and b[2][3] == l and b[3][3] == l))
```

New Code

So we added the lines for the 4th row and column so the winner function could use it to check for the result.