Group6ProjectFeb2025

March 2, 2025

1 Tic-Tac-Toe Group Project

1.1 Group 6- Caitlyn Carmabella Nayeli,Tahrin Tasmim, Anuradha Lakmali Wickramage Wickramage, Abdelaziz Bouraqqadi, Shima Azizi, and Don Ravindu Sanoj Hapuarachchi

1.1.1 Feb 2025

Please note that this code uses tk interface, which is a standard python library. If this code does not run the first time, please make sure you have it installed. If you are not sure how, please refer to this document: https://docs.python.org/3/library/tkinter.html

For this project we chose to make a game of Tic-Tac-Toe. Tic-Tac-Toe is a two-player game and consists of a nine-square grid. Each player chooses their move and with O or X and marks their square one at each chance. The player who succeeds in making their marks all in one line whether diagonally, horizontally, or vertically wins. The challenge for the other player is to block the game for their opponent and also to make their chain.

2 Import standard python library tk interface in order to make a small gui

[5]:
```
from tkinter import *
from tkinter import messagebox
```

3 Instatiate global variables

[7]:
```
complete = False
player = "X"
```

4 Define function to quit out of the gui window when the game is over

[9]:
```
def quit(self):
    self.destroy()
```

5 Change turn function, switches player from X to O and vice versa

[11]:
```
def changeTurn():
    global player
    if (player == "X"):
        player = "O"
    else:
        player = "X"
```

6 Define on-click function for the button on the grid of the gui

[13]:
```
def click(row,col):

    global player

    if player == "X" and states[row][col] == 0 and complete == False:
        states[row][col] = "X"
        b[row][col].configure(text = "X")

    elif player == "O" and states[row][col] == 0 and complete == False:
        states[row][col] = "O"
        b[row][col].configure(text = "O")

    checkWin()
```

7 Define function to check if there is a winning combination

[15]:
```
def checkWin():

    global complete
    global player

    i = 0

    while (i < 3):
        # horizontal cases
        if (states[i][0] == states[i][1] == states[i][2] !=0):
            complete = True
            messagebox.showinfo(player + " Wins!")
            quit(root)
            break

        # vertical cases
        elif (states[0][i] == states[1][i] == states[2][i] != 0):
            complete = True
            messagebox.showinfo(player + " Wins!")
            quit(root)
            break

        # diagonal case 1
        elif (states[0][0] == states[1][1] == states[2][2] !=0):
            complete = True
            messagebox.showinfo(player + " Wins!")
            quit(root)
            break
        # diagonal case 2
        elif (states[0][2] == states[1][1] == states[2][0] !=0):
            complete = True
            messagebox.showinfo(player + " Wins!")
            quit(root)
            break
        # every box complete with no win instance
        elif (states[0][0] and states[0][1] and states[0][2] and states[1][0]␣
 ↪and states[1][1] and states[1][2] and states[2][0] and states[2][1] and␣
 ↪states[2][2] != 0):
            complete = True
            messagebox.showinfo("It's a Tie!")
            quit(root)
            break
        else:
            i = i + 1

    changeTurn()
```

8 Define and instatiate the window of the gui

[17]:
```
root = Tk()
root.title("Tic Tac Toe: Group 6")
messagebox.showinfo("X Starts!")
#Button
b = [
     [0,0,0],
     [0,0,0],
     [0,0,0]]

states = [
     [0,0,0],
     [0,0,0],
     [0,0,0]]
```

9 Run the loop that creates and runs the instance of the game

[19]:
```
for i in range(3):
    for j in range(3):
        b[i][j] = Button(height = 4, width = 8, font = ("Helvetica","24"),␣
 ↪command = lambda row = i, col = j : click(row,col))
        b[i][j].grid(row = i, column = j)

mainloop()
```

Conclusion: in order to logic out what we needed to solve the problem, we made an algorithm and referenced other works to check its validity. Basically we knew that we would have to set up a way to click a button and mark either X or O, change turns, and determine a win, so we made individual functions we could call for those problems. We wrote and tested each by themselves, and later also added the quit function to destroy the window once the game was finished. We used tkinterface because it is a common tool included with most python installations that we could use to implement a simple gui. We had to look up how to incorporate it, but it wasn't too difficult. The hardest part was actually figuring out how to instantiate the window and the main loop themselves. The only significant user issue is that if you click on a taken asquare, it will let you and invalidate your turn instead saying you can't do it or giving you a warning. I considered implementing a warning, but decided it was outside the scope of the project.