Mini Project #3 – Tic Tac Toe

This will be the final mini project for the semester.

Due in Recitation the Week of 4/23-4/27

Important note on Plagiarism

As you may have heard, we had a lot of issues with plagiarism for Mini Project #2. Please note that I do use plagiarism software.

We expect that you will complete this assignment ***individually***. This means that you ***cannot***:

- Share code with friends.
- Find code on the internet.
- Write code together with a partner.
- Submit ***any*** code that was not written entirely by you.

This does leave room for limited collaboration. The following things ***are acceptable:***

- Talking to a partner about strategies you are using to code this project.
- Asking for help ***with code that you have written***.
- Offering to help a friend ***with code that they have written*** as long as you do not give them your code, or copy their code.
- Asking instructors for help.

If you collaborate with another student in any way, you must include their name(s) in a comment in your .m file.

Instructions

Getting Started

Download ticTacToe.m and ticTacToe.fig from the assignment link. You will not have to modify the .fig file in any way for this project.

Understanding the program

Read through the entirety of ticTacToe.m. Try to come to an understanding of what the code written in the program currently does. I have written an extensive amount of comments to help you understand what is going on.

What still needs to be written

The current `ticTacToe` program lacks the code required for players to put pieces on the board, and the code needed to determine whether the game has ended.

Game Rules

In Tic Tac Toe, two players take turns placing their pieces (‘o’ and ‘x’) in a 3x3 grid. The ‘o’ player goes first, and the ‘x’ player goes second. The game is won if a player creates a line of

three pieces horizontally, vertically, or diagonally. The game ends in a tie, also known as a Cat's Game, if all squares get taken up without a player winning.

Step 1: Writing a `buttonPress`

`buttonPress` is a function that places pieces on the `ticTacToe` board in the GUI. `buttonPress` is a function that should be called by every pushbutton callback function in `ticTacToe` **except** for the `newGame` button.

- `buttonPress` takes the `handles` structure for the GUI and the `hObject` structure for the button that was pressed as inputs
- and returns an updated `handles` structure as an output.

Hint: You can use `hObject` to determine which button was just pressed.

`buttonPress` should do the following:

- Check to see if the button that was clicked is a valid button to place a piece. Buttons are valid if they are empty, and invalid if they already contain a piece.
    - Invalid buttons should briefly flash red if they are clicked.
- If the button is valid, the button should be updated to contain the player's piece by changing the `String` property of the button.
- If a piece was placed, you should update any turn information data (such as `handles.turnNumber`) so that it is now the next player's turn.
- If a piece was placed, should also change the `String` of the static text box `whoseTurn` to display the next piece to be placed (the piece of the next player).

Step 2: Writing `checkWinner`

`checkWinner` is a function that determines whether the game of `ticTacToe` is over, either because somebody won or because the game has ended in a tie. `checkWinner` is a function that should be called by every pushbutton callback function in `ticTacToe` **except** for the `newGame` button.

- `checkWinner` takes the `handles` structure of the GUI as an input
- and returns an updated `handles` structure as an output.

`checkWinner` should do the following:

- Check whether either player has won the game.
- Check whether the game has ended in a tie (because all of the buttons have pieces on them).
- If the game is over, display the result of the game, either with a pop up message like the ones created with functions like `errordlg` and `msgbox`, or in one of the static text boxes on the GUI.

Step 3: Writing the callback functions

The code for each callback function should be the same:

- A call to `buttonPress`
- A call to `checkWinner`
- `guidata(hObjuect,handles)`

As it says in the comments of `ticTacToe`, writing anything more than that, or anything different, is wrong.

Submission

When you have completed this project, and have verified that your code has run correctly, you are ready to submit. Please do the following to submit:

- Copy and paste the code for `buttonPress` and `checkWinner` into a new .m file.
- Save that .m file.
- Submit that .m file on Canvas
- Print that .m file and give it to your recitation instructor.