blinkdagger

an Engineering and MATLAB blog

- Home
- Listchecker
- MATLAB
- Contact
- About

MATLAB GUI Tutorial - Sharing Data among Callbacks and Sub Functions

14 Nov 2007 *Quan Quach* 69 comments 14,141 views

Introduction

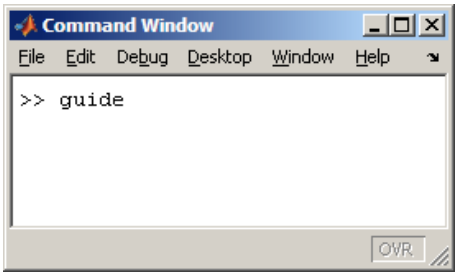
For the majority of GUIs that you create in Matlab, you will have to be able to share data among the component callbacks. One way of accomplishing this is to use the handles structure to store that data. In this tutorial, you will learn how to do so.

This tutorial is written for those with little experience creating a Matlab GUI (Graphical User Interface). If you're new to creating GUIs in Matlab, you should visit this tutorial first. Basic knowledge of Matlab is recommended. Matlab version 2007a is used in writing this tutorial. Both earlier versions and new versions should be compatible as well (as long as it isn't too outdated). Let's get started!

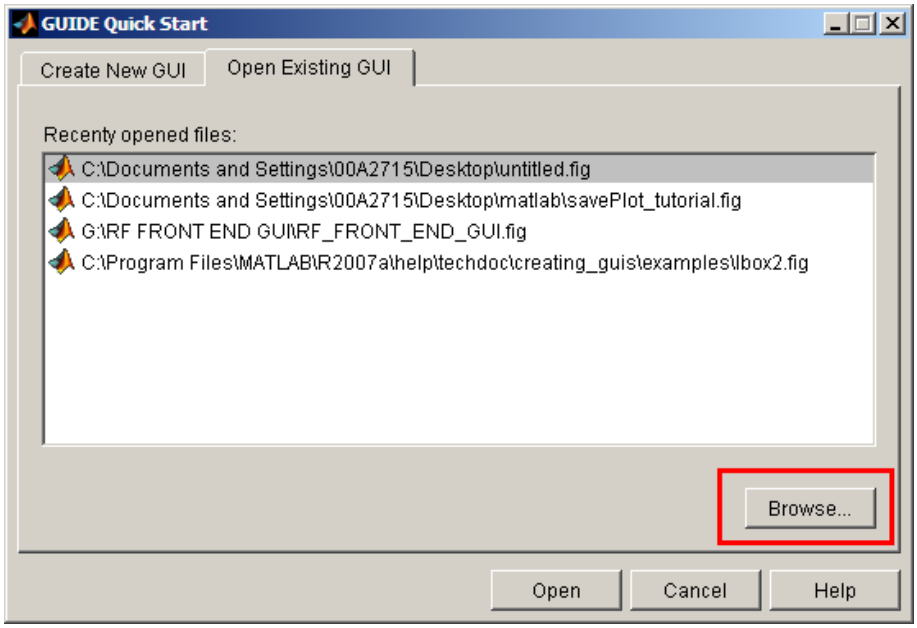
Sharing Data Between Callbacks

A simple example of sharing data among callbacks is to increment a numerical counter on the GUI each time ANY button on the GUI is pressed. How can this be done? This tutorial will explain how.

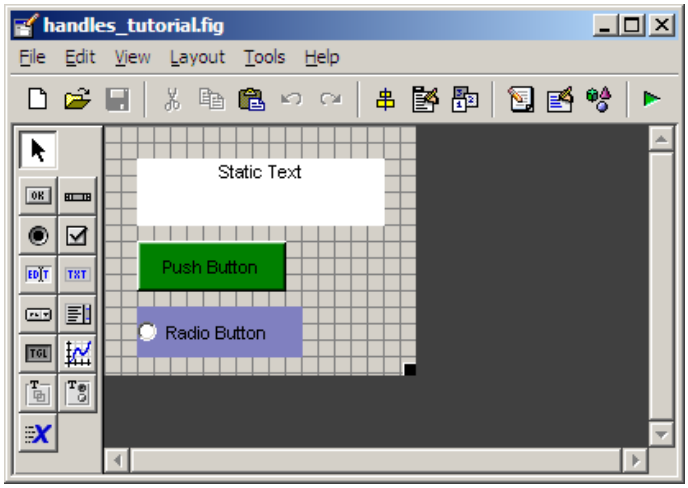
1. First, download the sample GUI here. Unzip the files and place them wherever you please.
2. Now, type guide at the command prompt.



3. Choose to open the sample GUI by clicking on "Open Existing GUI". Click on "Browse" to locate where you saved the GUI files.



4. Here is what the GUI should look like when you open it:



5. Click on the  icon on the GUI figure to bring up the accompanying .m file.

6. The first thing we need to do is to define a variable that will hold the value for our numerical counter. Place the following code in *handles_tutorial_OpeningFcn*.

```
handles.buttonCounter = 0;
set(handles.text1,'String','0');
```

Make sure you place it right before this line of code:

```
guidata(hObject, handles);
```

7. Notice that we defined the variable name as handles.buttonCounter, rather than buttonCounter. If we did not define the variable within the handles structure, then it would be considered a local variable. Being a local variable means that the variable does not exist outside of the function where it was defined. Thus we need to store it into the handles structure, so that it can be accessed and modified later in the other callbacks. So, if you want data to be available to ALL callbacks, you must store it in the handles structures.

8. Add this code to both *pushbutton1_Callback* and *radiobutton1_Callback*

```
handles.buttonCounter = handles.buttonCounter + 1;
set(handles.text1,'String',num2str(handles.buttonCounter) );
guidata(hObject, handles); %without this line, the handles structure would not update,
```

```
%and the counter would not increment.
```

9. Now, save the .m file and run the GUI. Try clicking on the buttons to increment the counter. Notice that the counter increments when you activate AND deactivate the radio button.

Sharing Data Between Callbacks and Sub Functions

It is good practice to make your functions modular so that your code is easily adaptable, robust, and flexible. Having said that, passing the entire handles structures to sub functions is something that I **don't** recommend. But there might be a time when you need to access the entire handles within a sub function. So here goes.

```
function [handles] = mySubFunction(handles)
%insert code here
%
%
```

This function will allow you to use the handles data within the sub function, and will also update any changes made within the sub function. You can add as many inputs and outputs as desired.

When you call this function, make sure to call it in the following manner:

```
[handles] = mySubFunction(handles);
```

This is the end of the tutorial.