blinkdagger

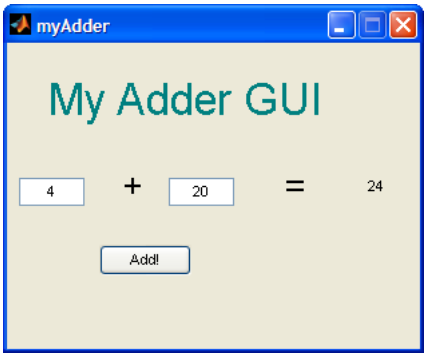
an Engineering and MATLAB blog

- Home
- Listchecker
- MATLAB
- Contact
- About

MATLAB GUI (Graphical User Interface) Tutorial for Beginners

23 Oct 2007 *Quan Quach* 341 comments 106,587 views

Why use a GUI in MATLAB? The main reason GUIs are used is because it makes things simple for the end-users of the program. If GUIs were not used, people would have to work from the command line interface, which can be extremely difficult and fustrating. Imagine if you had to input text commands to operate your web browser (yes, your web browser is a GUI too!). It wouldn't be very practical would it? In this tutorial, we will create a simple GUI that will add together two numbers, displaying the answer in a designated text field.

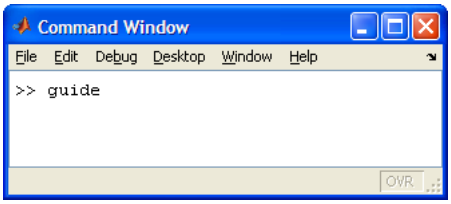


This tutorial is written for those with little or no experience creating a MATLAB GUI (Graphical User Interface). Basic knowledge of MATLAB is not required, but recommended. MATLAB version 2007a is used in writing this tutorial. Both earlier versions and new versions should be compatible as well (as long as it isn't too outdated). Lets get started!

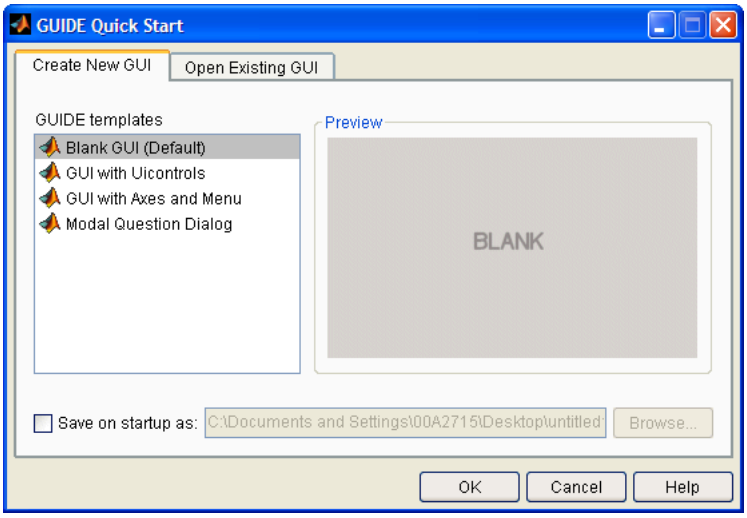
Contents

Initializing GUIDE (GUI Creator)

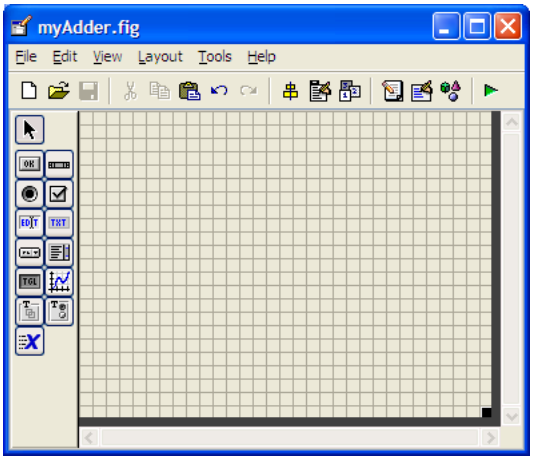
1. First, open up MATLAB. Go to the command window and type in guide.



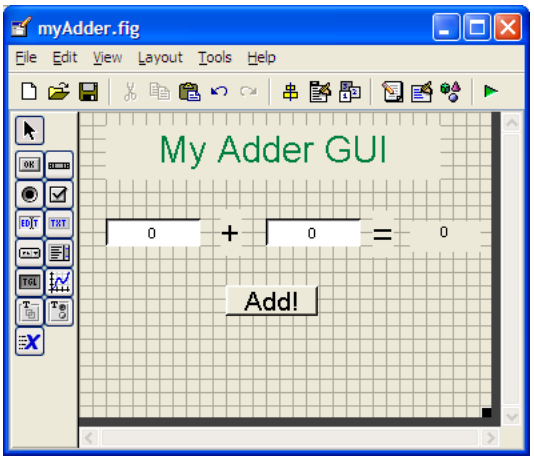
2. You should see the following screen appear. Choose the first option Blank GUI (Default).



3. You should now see the following screen (or something similar depending on what version of MATLAB you are using and what the predesignated settings are):



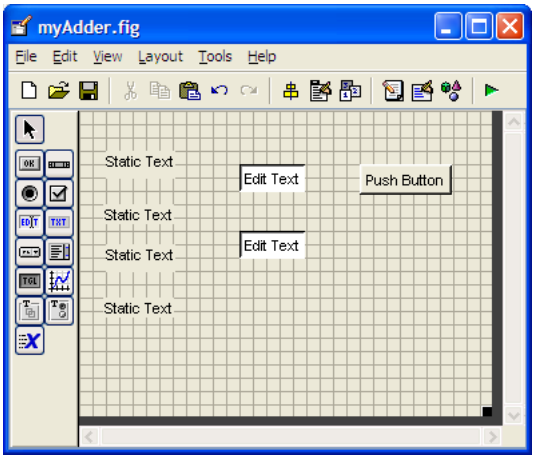
4. Before adding components blindly, it is good to have a rough idea about how you want the graphical part of the GUI to look like so that it'll be easier to lay it out. Below is a sample of what the finished GUI might look like.



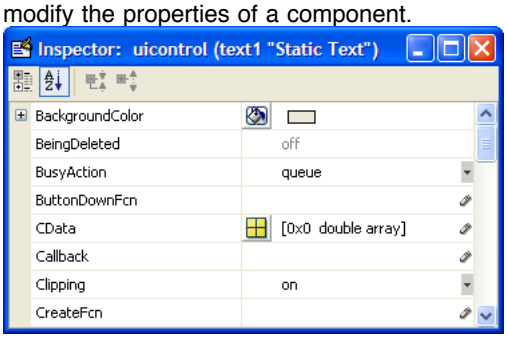
Creating the Visual Aspect of the GUI: Part 1

1. For the adder GUI, we will need the following components
   - Two Edit Text components
   - Three Static Text component
   - One Pushbutton component

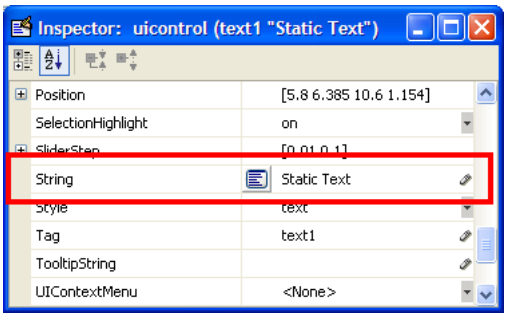
   Add in all these components to the GUI by clicking on the icon and placing it onto the grid. At this point, your GUI should look similar to the figure below :



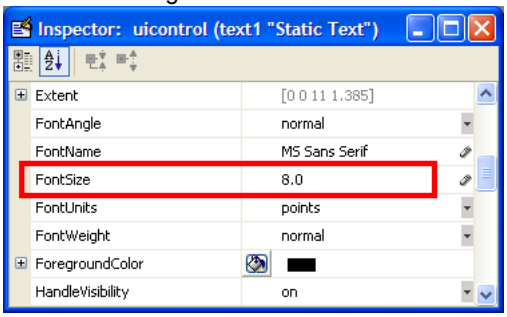
2. Next, its time to edit the properties of these components. Let's start with the static text. Double click one of the *Static Text* components. You should see the following table appear. It is called the *Property Inspector* and allows you to modify the properties of a component.



3. We're interested in changing the *String* parameter. Go ahead and edit this text to +.



Let's also change the font size from 8 to 20.



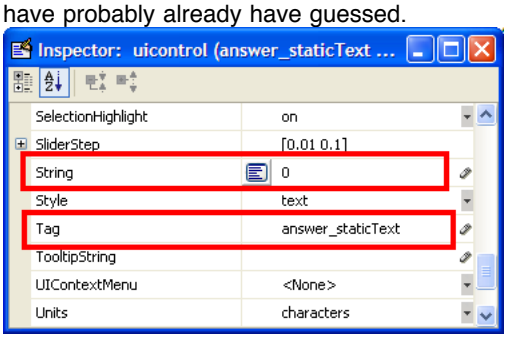
After modifying these properties, the component may not be fully visible on the GUI editor. This can be fixed if you resize the component, i.e. use your mouse cursor and stretch the component to make it larger.

4. Now, do the same for the next *Static Text* component, but instead of changing the *String* parameter to +, change it to =.

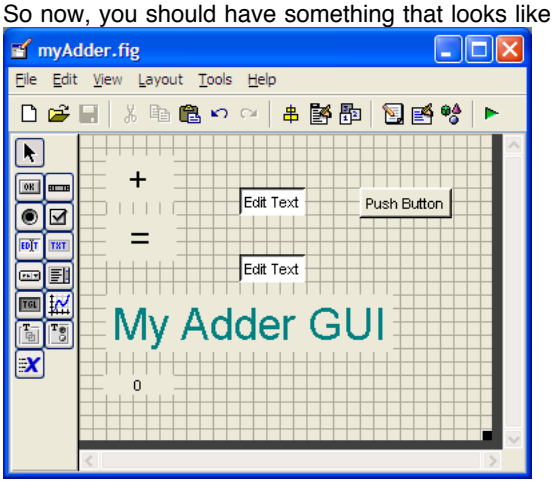
5. For the third *Static Text* component, change the *String* parameter to whatever you want as the title to your GUI. I kept it simple and named it MyAdderGUI. You can also experiment around with the different font options as well.

6. For the final *Static Text* component, we want to set the *String* Parameter to 0. In addition, we want to modify the *Tag* parameter for this component. The *Tag* parameter is basically the variable name of this component. Let's call it answer_staticText. This component will be used to display our answer, as you

have probably already have guessed.

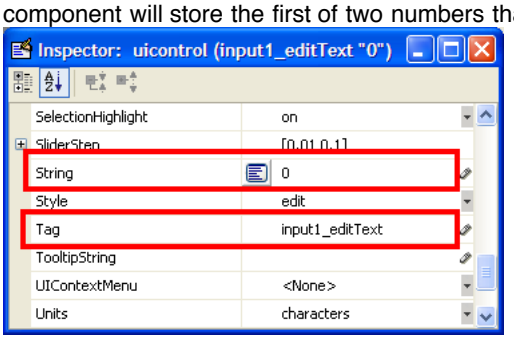


7. So now, you should have something that looks like the following:



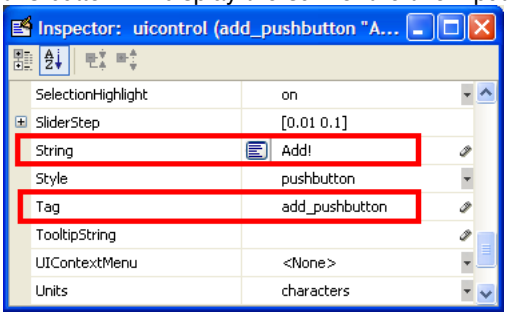
Creating the Visual Aspect of the GUI: Part 2

1. Next, lets modify the *Edit Text* components. Double click on the first *Edit Text* component. We want to set the *String* parameter to 0 and we also want to change the *Tag* parameter to input1_editText, as shown below. This component will store the first of two numbers that will be added together.

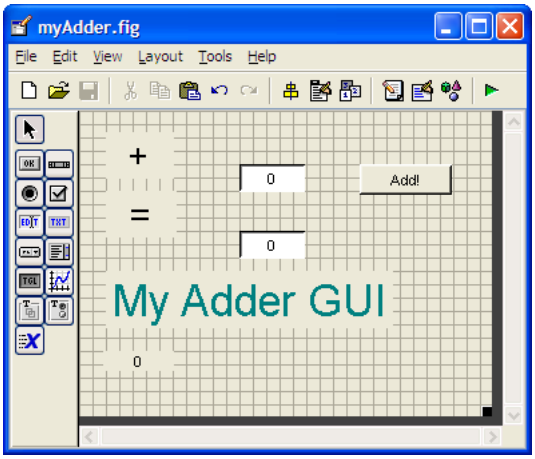


2. For the second *Edit Text* component, set the *String* parameter to 0 **BUT** set the *Tag* parameter input2_editText. This component will store the second of two numbers that will be added together.

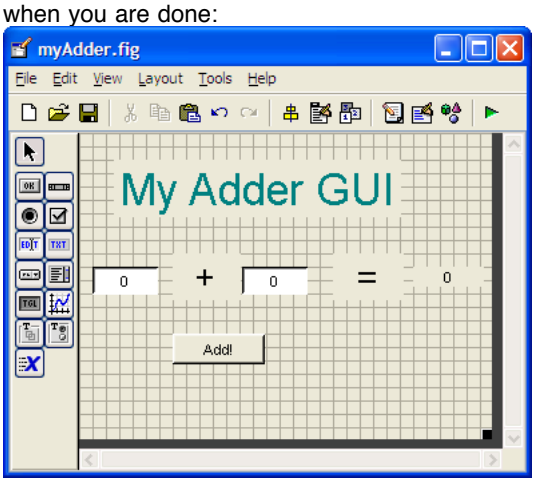
3. Finally, we need to modify the *pushbutton* component. Change the *String* parameter to Add! and change the *Tag* parameter to add_pushbutton. Pushing this button will display the sum of the two input numbers.



4. So now, you should have something like this:



Rearrange your components accordingly. You should have something like this when you are done:

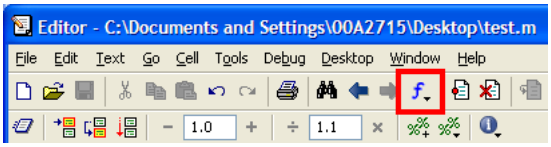


5. Now, save your GUI under any file name you please. I chose to name mine myAdder. When you save this file, MATLAB automatically generates two files: *myAdder.fig* and *myAdder.m*. The .fig file contains the graphics of your interface. The .m file contains all the code for the GUI.

Writing the Code for the GUI Callbacks

MATLAB automatically generates an .m file to go along with the figure that you just put together. The .m file is where we attach the appropriate code to the callback of each component. For the purposes of this tutorial, we are primarily concerned only with the *callback* functions. You don't have to worry about any of the other function types.

1. Open up the .m file that was automatically generated when you saved your GUI. In the MATLAB editor, click on the  icon, which will bring up a list of the functions within the .m file. Select *input1_editText_Callback*.



2. The cursor should take you to the following code block:

```
function input1_editText_Callback(hObject, eventdata, handles)
% hObject    handle to input1_editText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'String') returns contents of input1_editText as text
%       str2double(get(hObject,'String')) returns contents of
%       input1_editText as a double
```

Add the following code to the bottom of that code block:

```
%store the contents of input1_editText as a string. if the string
%is not a number then input will be empty
```

```
input = str2num(get(hObject,'String'));

%checks to see if input is empty. if so, default input1_editText to zero
if (isempty(input))
    set(hObject,'String','0')
end
guidata(hObject, handles);
```

This piece of code simply makes sure that the input is well defined. We don't want the user to put in inputs that aren't numbers! The last line of code tells the gui to update the handles structure after the callback is complete. The handles stores all the relevant data related to the GUI. This topic will be discussed in depth in a different tutorial. For now, you should take it at face value that it's a good idea to end each callback function with guidata(hObject, handles); so that the handles are always updated after each callback. This can save you from potential headaches later on.

3. Add the same block of code to *input2_editText_Callback*.

4. Now we need to edit the *add_pushbutton_Callback*. Click on the  icon and select *add_pushbutton_Callback*. The following code block is what you should see in the .m file.

```
% --- Executes on button press in add_pushbutton.
function add_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to add_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Here is the code that we will add to this callback:

```
a = get(handles.input1_editText,'String');
b = get(handles.input2_editText,'String');
% a and b are variables of Strings type, and need to be converted
% to variables of Number type before they can be added together

total = str2num(a) + str2num(b);
c = num2str(total);
% need to convert the answer back into String type to display it
set(handles.answer_staticText,'String',c);
guidata(hObject, handles);
```

5. Let's discuss how the code we just added works:

```
a = get(handles.input1_editText,'String');
b = get(handles.input2_editText,'String');
```

The two lines of code above take the strings within the *Edit Text* components, and stores them into the variables *a* and *b*. Since they are variables of *String* type, and not *Number* type, we cannot simply add them together. Thus, we must convert *a* and *b* to *Number* type before MATLAB can add them together.

6. We can convert variables of *String* type to *Number* type using the MATLAB command str2num(String argument). Similarly, we can do the opposite using num2str(Number argument). The following line of code is used to add the two inputs together.

```
total= (str2num(a) + str2num(b));
```

The next line of code converts the *sum* variable to *String* type and stores it into the variable *c*.

```
c = num2str(total);
```

The reason we convert the final answer back into *String* type is because the *Static Text* component does not display variables of *Number* type. If you did not convert it back into a *String* type, the GUI would run into an error when it tries to display the answer.

7. Now we just need to send the sum of the two inputs to the answer box that we created. This is done using the following line of code. This line of code populates the *Static Text* component with the variable *c*.

```
set(handles.answer_staticText,'String',c);
```

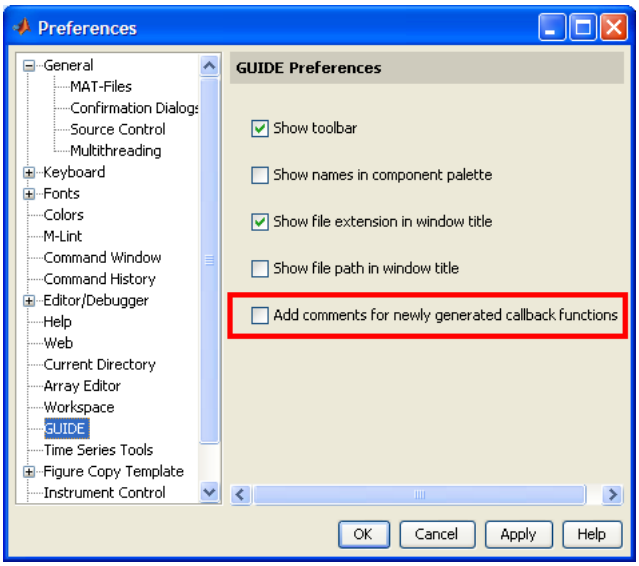
The last line of code updates the handles structures as was previously

mentioned.

guidata(hObject, handles);

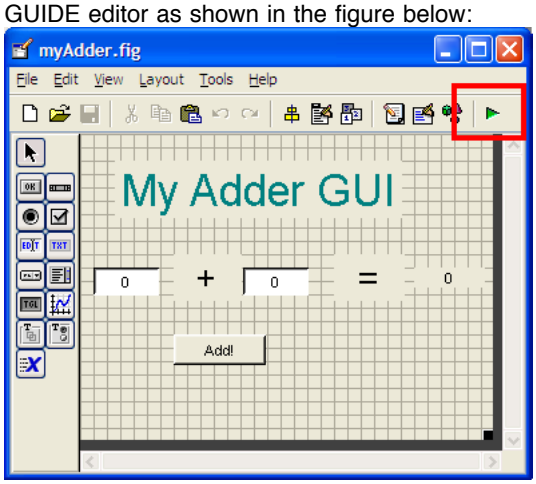
Congratulations, we're finished coding the GUI. Don't forget to save your m-file. It is now time to launch the GUI!

8. If you don't want MATLAB to automatically generate all those comments for each of the callbacks, there is a way to disable this feature. From the GUI editor, go to ***File***, then to ***Preferences***.

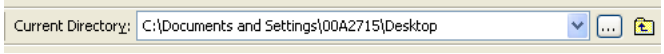


Launching the GUI

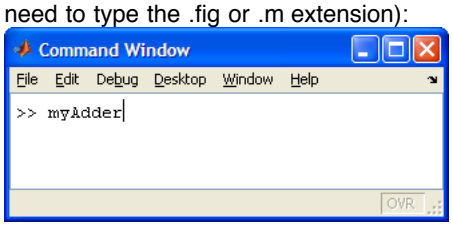
1. There are two ways to launch your GUI.
   - The first way is through the GUIDE editor. Simply press the icon on the GUIDE editor as shown in the figure below:



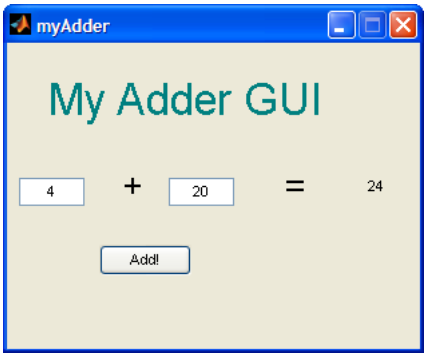
   - The second method is to launch the GUI from the MATLAB command prompt. First, set the MATLAB current directory to wherever you saved your .fig and .m file.



     Next, type in the name of the GUI at the command prompt (you don't need to type the .fig or .m extension):



2. The GUI should start running immediately:



Try to input some numbers to test out the GUI. **Congratulations** on creating your first GUI!

Troubleshooting and Potential Problems

So your GUI doesn't work and you don't know why. Here are a couple of tips that might help you find your bug:

1. If you can't figure out where you error is, it might be a good idea to quickly go through this tutorial again.
2. The command line can give you many hints on where exactly the problem resides. If your GUI is not working for any reason, the error will be outputted to the command prompt. The line number of the faulty code and a short description of the error is given. This is always a good place to start investigating.
3. Make sure all your variable names are consistent in the code. In addition, make sure your component Tags are consistent between the .fig and the .m file. For example, if you're trying to extract the string from the *Edit Text* component, make sure that your get statement uses the right tag! More specifically, if you have the following line in your code, make sure that you named the *Edit Text* component accordingly!

   a = get(handles.input1_editText,'String');
4. The source code is available here, and could be useful for debugging purposes.
5. If all else fails, leave a comment here and we'll try our best to help.

Related Posts and Other Links

MATLAB GUI Tutorial - Slider
MATLAB GUI Tutorial - Pop-up Menu
MATLAB GUI Tutorial - Plotting Data to Axes
MATLAB GUI Tutorial - Button Types and Button Group
MATLAB GUI Tutorial - A Brief Introduction to handles
MATLAB GUI Tutorial - Sharing Data among Callbacks and Sub Functions
Video Tutorial: GUIDE Basics
More GUI Tutorial Videos From Doug Hull

This is the end of the tutorial.

341 Responses to "MATLAB GUI (Graphical User Interface) Tutorial for Beginners"

1. *on 20 Nov 2007 at 10:04 am 1Mike*

   Thanks for the tutorial - its nice and clear