MATLAB GUI Tips

Collated advice on construction of user interfaces

R. S. Schestowitz
Imaging Science and Biomedical Engineering
Stopford Building
University of Manchester
United Kingdom

May 18th, 2004

WARNING: Written on a sole afternoon (roughly 2.5 hours). Typos are expected as well as solutions that are not optimal.

Contents

1 Introduction

From a command-line-driven operation of MATLAB code, many of us migrate to perform tasks while hiding the unnecessary clutter and technicality that lies underneath. Interfacing is concerned with simplification of entry points to the code and encapsulation of its functional behaviour. And what better way to do so than by employing a graphical front-end? This is what this document will focus on – the ability to embed functions within a graphical user interface (GUI).

The document is intended to provide a step-by-step description while concentrating on useful tips and tricks that make GUI construction more trivial. With the aid of GUIDE (the GUI development environment), even GUI construction itself is controlled by a GUI. However, no prior knowledge of GUIDE will be assumed and principles will form the majority of arguments and suggestions.

On the issue of technicality and formality, this document intentionally avoids the use of jargon and should be readable by people who have no background of computer science or mathematics. The instruction are meant to remain clear and concise so that GUI's can be constructed most rapidly.

2 Starting Point

Let us assume that a function $myfunction$ has been created and it takes a series of arguments (e.g., $arg1$,$arg2$ and so on). A call to this function will either produce a result (visual or numerical typically) or return data of some sort to the command-line (or command window). What is the difference between the two? Simply, the former makes the function a stand-alone list of commands that cannot be easily combined with other code. In the case of GUI's, these usually appear the appropriate entity to start off with. The exception is the case where our GUI should be used by another auxiliary application; but this case will not be considered here any further.

Let us illustrate with an example. The function *draw_some_plot* is is set to produce a plot and do nothing more. The code for such a file (*draw_some_plot.m*) may be:

```
plot(rand(10));
```

Now all we wish to do is to call this function from a graphical window which may, for example, contain just a button to invoke our function. The rather intuitive GUIDE (it can be simply started by typing in *guide*) that is based on Java, makes the placement of GUI elements an obvious step to follow. For information on the use of any GUI generation tool, see the relevant documentation and help files.

This then leads us to the concept of *callbacks*. These are functions that will be called once an event in some GUI occurs. Once a callback has been assigned to an object, the control flow of the program will be passed to the assigned callback. If a button has been assigned the callback named *do_action*, then once we press that button, the code under *do_action* will be executed. Going back to the example above, sensible code to put under this function will be just the statement:

```
draw_some_plot;
```

which will then draw the plot, assuming paths have been defined properly. We now have covered some dry and embarrassingly fundamental grounds. The next few section describe some of the interesting things that can be done to increase the functionality of the user interface quickly and wisely (in practice, I never got that far).

3 Commonly Used Code

The following aims to present some code which is repeatedly used and establishes what is expected from objects.

3.1 Radio Buttons

Where a group of buttons is inter-related, the following code should do:

```
set(handles.option1, 'Value', 1);
set(handles.option2, 'Value', 0);
set(handles.option3, 'Value', 0);
```

where *option1* is the source of the callback. For the other options, the only part which needs changing is the latter one where the state of the button is defined by a zero or a one.

3.2 Check Boxes

A naive yet useful implementation of those will involve an element *state* which hold some information about the state of the checkbox or its meaning.

```
if (get(handles.state,'Value') == 0),
   set(handles.checkbox, 'Value', 0);
   set(handles.state, 'String', '0');
else
   set(handles.checkbox, 'Value', 1);
   set(handles.state, 'String', '1');
end
```

3.3 Drop-down Menus

In the following, the menu object needs to first be identified using:

```
contents = get(hObject,'String');
```

Subsequently, the menu entry needs to be checked for extraction and setting of information.

```
if (strcmp(contents{get(hObject,'Value')},'MenuEntry1')),
   set(handles.data, 'String', 'Data1');
elseif (strcmp(contents{get(hObject,'Value')},'MenuEntry2')),
   set(handles.data, 'String', 'Data2');
else
   msgbox('Error with menu callback. Parameter passed is not recog
end
```

This should record the menu selection appropriately.

3.4 Sliders

The following code will fetch the quantised value of the slider and assign it to a new object called *slider_value*.

```
set(handles.slider_value, 'String', num2str(ceil(get(handles.slide
```

3.5 Menu Items

For items that need to be ticked and unticked, the following can be useful.

```
set(handles.menuitem1, 'Checked', 'off');
set(handles.menuitem2, 'Checked', 'off');
set(handles.menuitem3, 'Checked', 'off');
set(handles.menuitem4, 'Checked', 'on');
set(handles.menu_selection, 'String', 'item4');
```

4 GUI Editing

More efficient work on the code can be done by opening all relevant files in advance. I personally write M-files to open all relevant windows at the start. Set up a file which includes the following lines:

```
edit <m-file1> <m-file2>...
guide <fig-file1> <fig-file2>...
```

where of course the files listed are these which are frequently worked upon.

5 Final Word

The document was adapted from notes whose essence is given in Section 3 onwards. As I earlier stated, it is meant to be useful more than eloquent and hopefully assist people who are new to callback function constructions. Comments are very welcome, most preferably via E-mail.