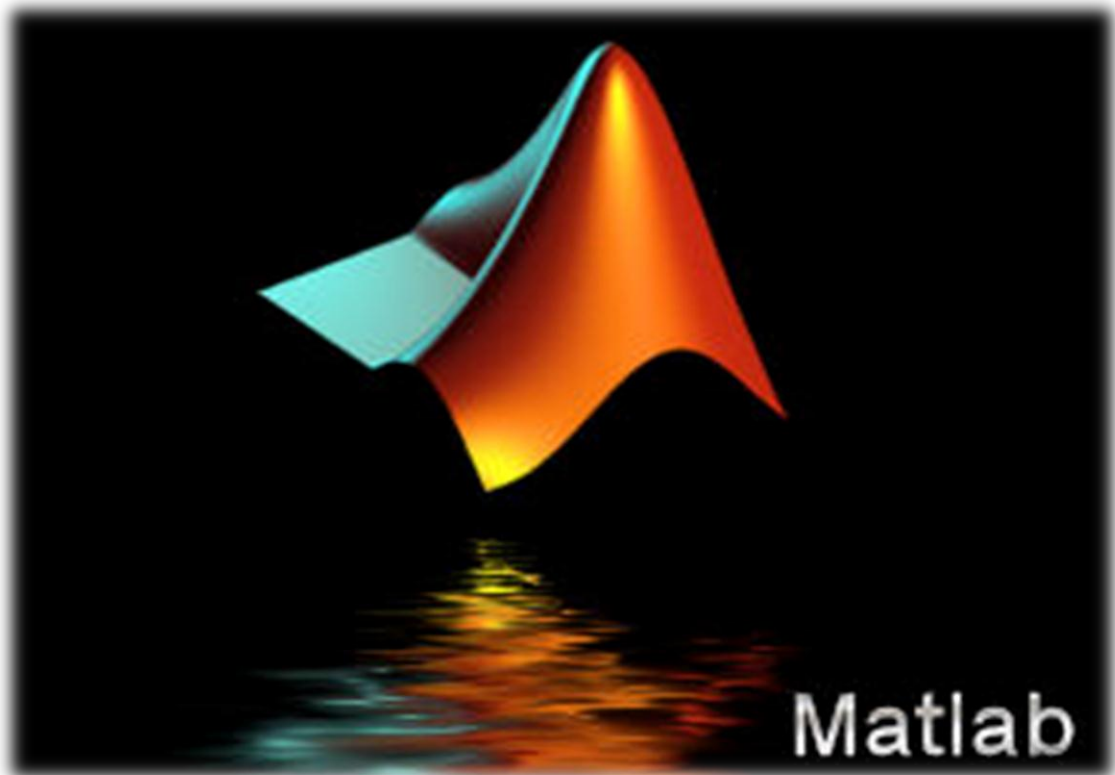


Experiment 5: GUI & SIMULINK



GUI

What Is a GUI?

A graphical user interface (GUI) is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. To perform these tasks, the user of the GUI does not have to create a script or type commands at the command line. Often, the user does not have to know the details of the task at hand. The GUI components can be menus, toolbars, push buttons, radio buttons, list boxes, and sliders. In MATLAB, a GUI can also display data in tabular form or as plots, and can group related components.

How Does a GUI Work?

Each component, and the GUI itself, is associated with one or more user-written routines known as callbacks. The execution of each callback is triggered by a particular user action such as a button push, mouse click, selection of a menu item, or the cursor passing over a component. This kind of programming is often referred to as event-driven programming. In event-driven programming, callback execution is asynchronous, controlled by events external to the software. In the case of MATLAB GUIs, these events usually take the form of user interactions with the GUI.

Ways to Build MATLAB GUIs

A MATLAB GUI is a figure window to which you add user-operated controls. You can select, size, and position these components as you like. Using callbacks you can make the components do what you want when the user clicks or manipulates them with keystrokes.

You can build MATLAB GUIs in two ways:

- Use GUIDE (GUI Development Environment), an interactive GUI construction kit.
- Create M-files that generate GUIs as functions or scripts (programmatic GUI construction).

The first approach starts with a figure that you populate with components from within a graphic layout editor. GUIDE creates an associated M-file containing callbacks for the GUI and its components. GUIDE saves both the figure (as a FIG-file) and the M-file. Opening either one also opens the other to run the GUI.

In the second, *programmatic*, GUI-building approach, you code an M-file that defines all component properties and behaviors; when a user executes the M-file, it creates a figure, populates it with components, and handles user interactions. The figure is not normally saved between sessions because the M-file creates a new one each time it runs.

As a result, the M-files of the two approaches look different. Programmatic M-files are generally longer, because they explicitly define every property of the figure and its controls, as well as the callbacks. GUIDE GUIs define most of the properties within the figure itself. They store the definitions in its FIG-file rather than in its M-file. The M-file contains callbacks and other functions that initialize the GUI when it opens.

MATLAB software also provides functions that simplify the creation of standard dialog boxes, for example to issue warnings or to open and save files. The GUI-building technique

you choose depends on your experience, your preferences, and the kind of application you need the GUI to operate.


You can combine the two approaches to some degree. You can create a GUI with GUIDE and then modify it programmatically. However, you cannot create a GUI programmatically and later modify it with GUIDE.

Before Designing a GUI

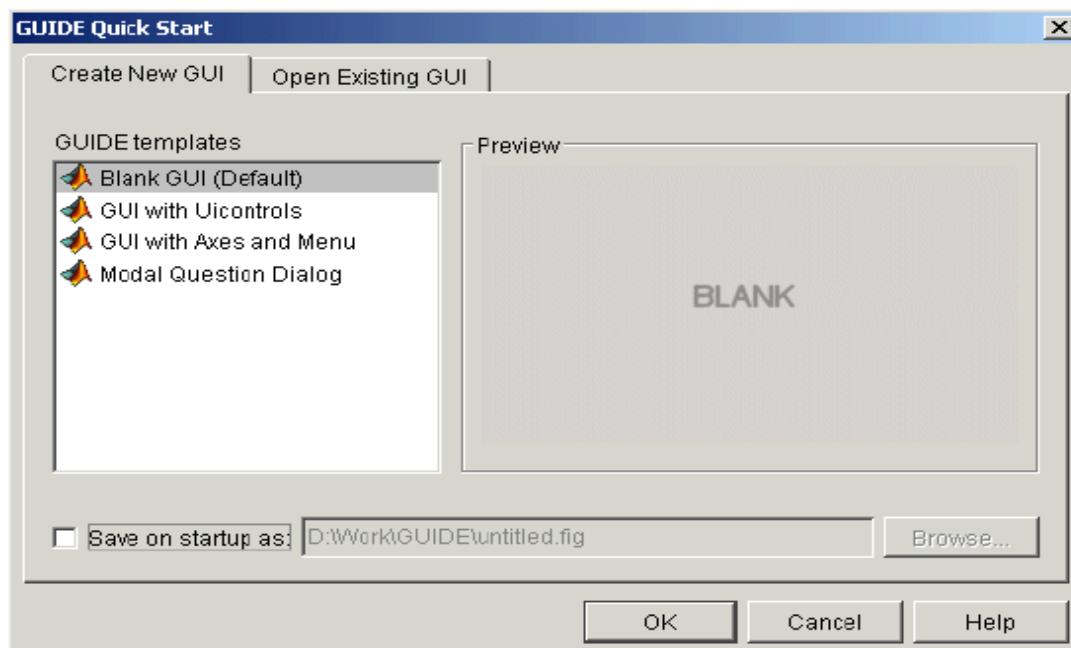
Before creating the actual GUI, it is important to decide what it is you want your GUI to do and how you want it to work. It is helpful to draw your GUI on paper and envision what the user sees and what actions the user takes.

Starting GUIDE

There are many ways to start GUIDE. You can start GUIDE from the:

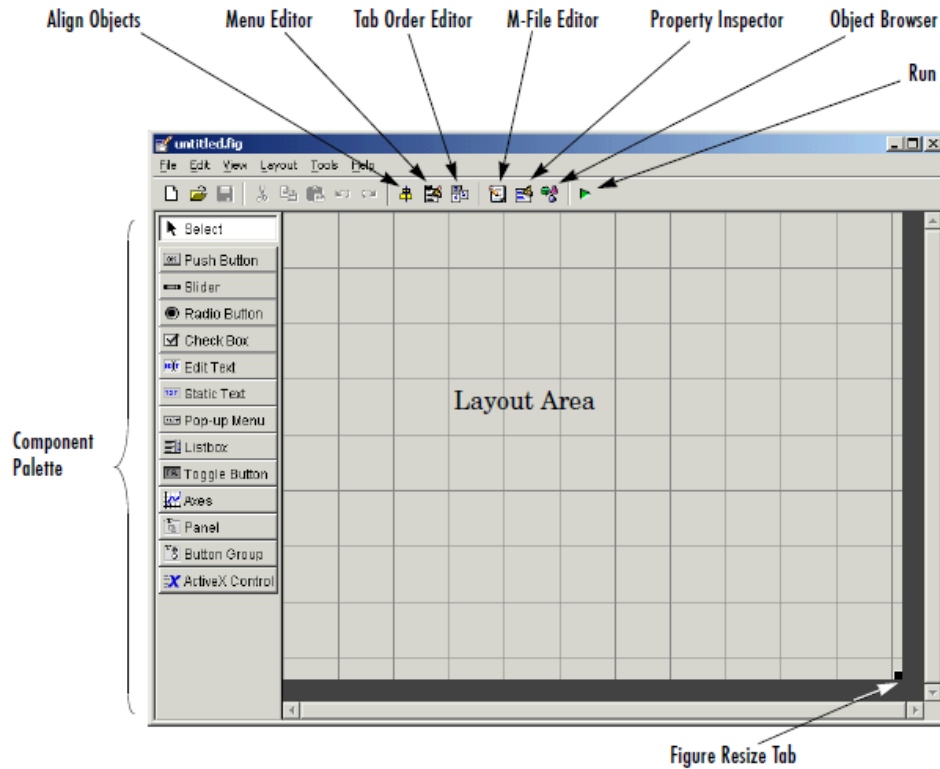
- Command line by typing `guide`
- **Start** menu by selecting **MATLAB > GUIDE (GUI Builder)**
- **MATLAB File** menu by selecting **New > GUI**
- MATLAB toolbar by clicking the **GUIDE** button 

However you start GUIDE, it displays the GUIDE Quick Start dialog box shown in the following figure.



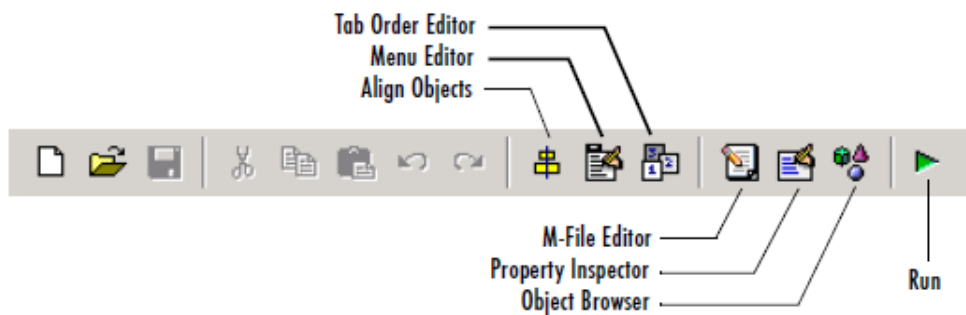
GUIDE Tools Summary

The GUIDE tools are available from the Layout Editor shown in the figure below. The tools are called out in the figure and described briefly below.



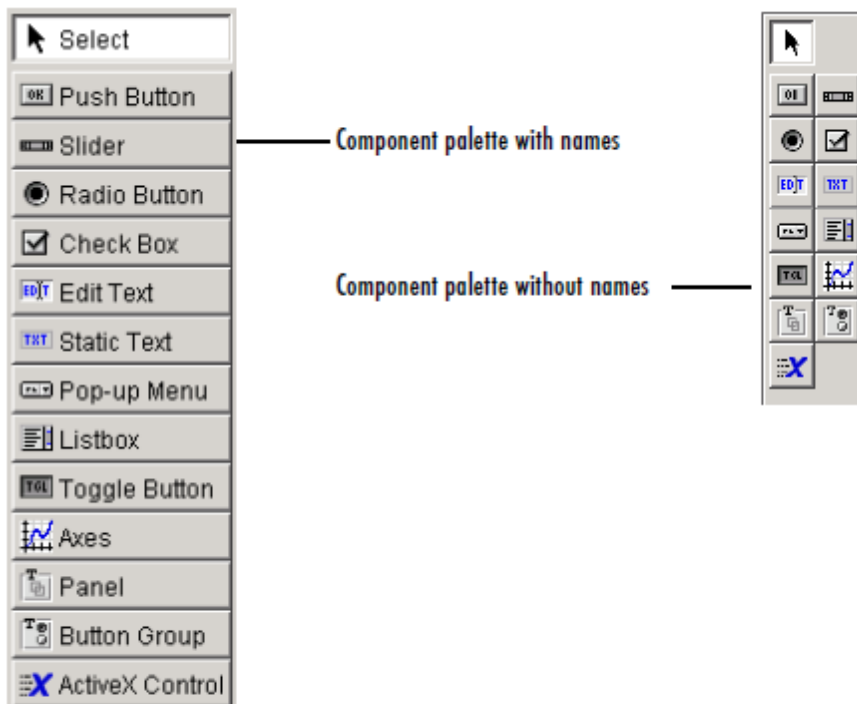
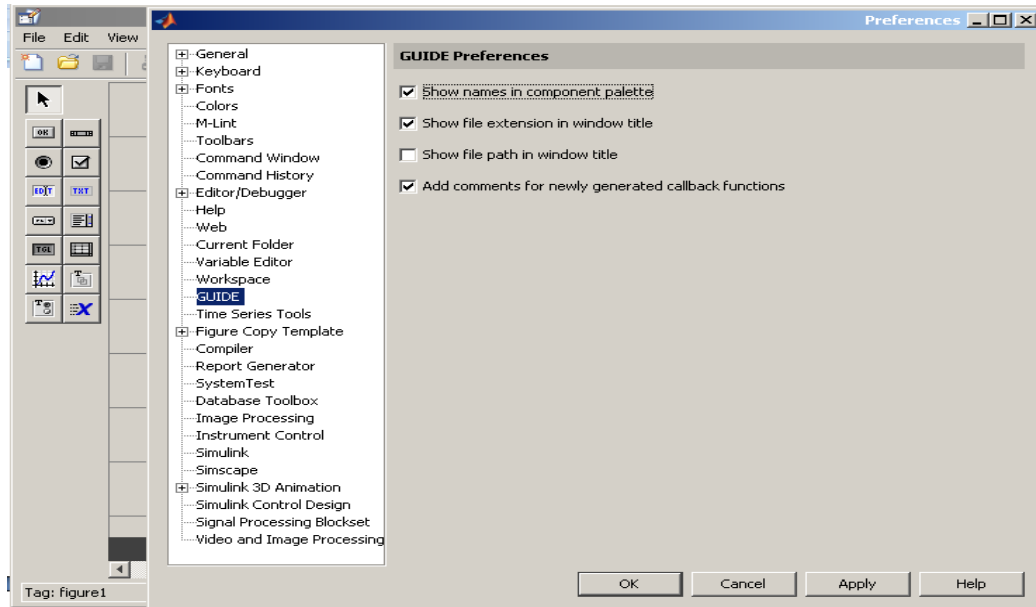
Show Toolbar

Displays the following toolbar in the Layout Editor window.



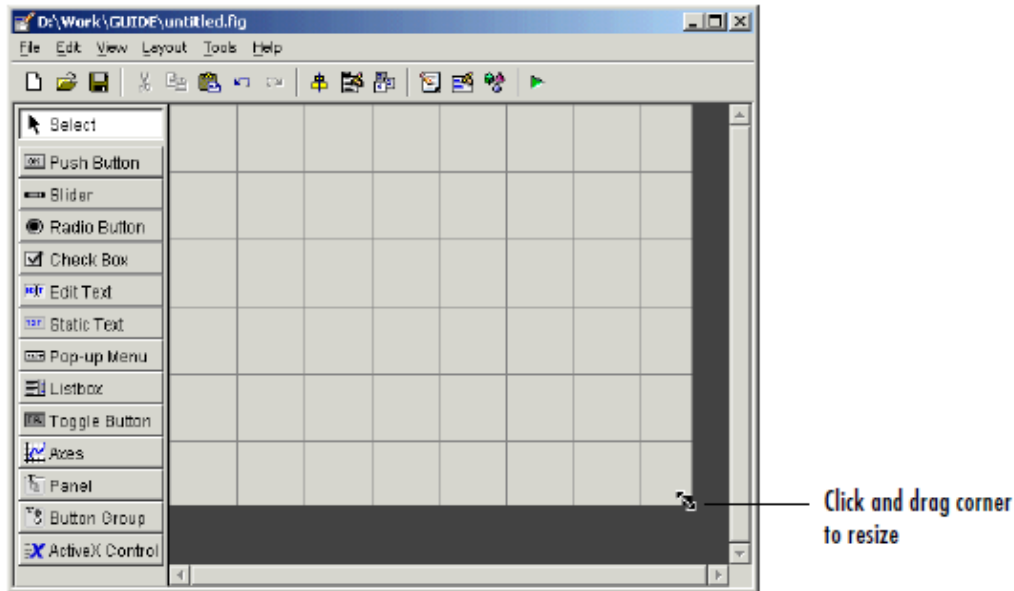
Show Names in Component Palette

When you first open the Layout Editor, the component palette contains only icons. To display the names of the GUI components, select **Preferences** from the **File** menu, check the box next to **Show names in component palette**, and click **OK**.







Setting the GUI Size









Set the size of the GUI by resizing the grid area in the Layout Editor. Click the lower-right corner and drag it until the GUI is the desired size. If necessary, make the window larger.



Available Components

The component palette at the left side of the Layout Editor contains the components that you can add to your GUI. You can display it with or without names.

Component	Icon	Description
Push Button		Push buttons generate an action when clicked. For example, an OK button might apply settings and close a dialog box. When you click a push button, it appears depressed; when you release the mouse button, the push button appears raised.
Toggle Button		Toggle buttons generate an action and indicate whether they are turned on or off. When you click a toggle button, it appears depressed, showing that it is on. When you release the mouse button, the toggle button remains depressed until you click it a second time. When you do so, the button returns to the raised state, showing that it is off. Use a button group to manage mutually exclusive toggle buttons.
Radio Button		Radio buttons are similar to check boxes, but radio buttons are typically mutually exclusive within a group of related radio buttons. That is, when you select one button the previously selected button is deselected. To activate a radio button, click the mouse button on the object. The display indicates the state of the button. Use a button group to manage mutually exclusive radio buttons.
Check Box		Check boxes can generate an action when checked and indicate their state as checked or not checked. Check boxes are useful

		when providing the user with a number of independent choices, for example, displaying a toolbar.
Edit Text		Edit text components are fields that enable users to enter or modify text strings. Use edit text when you want text as input. Users can enter numbers but you must convert them to their numeric equivalents.
Static Text		Static text controls display lines of text. Static text is typically used to label other controls, provide directions to the user, or indicate values associated with a slider. Users cannot change static text interactively.
Slider		Sliders accept numeric input within a specified range by enabling the user to move a sliding bar, which is called a slider or thumb. Users move the slider by clicking the slider and dragging it, by clicking in the trough, or by clicking an arrow. The location of the slider indicates the relative location within the specified range.
List Box		List boxes display a list of items and enable users to select one or more items.
Pop-Up Menu		Pop-up menus open to display a list of choices when users click the arrow.
Axes		Axes enable your GUI to display graphics such as graphs and images. Like all graphics objects, axes have properties that you can set to control many aspects of its behavior and appearance. See “Axes Properties” in the MATLAB Graphics documentation and commands such as the following for more information on axes objects: plot, surf, line, bar, polar, pie, contour, and mesh. See Functions — By Category in the MATLAB documentation for a complete list.
Panel		Panels arrange GUI components into groups. By visually grouping related controls, panels can make the user interface easier to understand. A panel can have a title and various borders. Panel children can be user interface controls and axes as well as button groups and other panels. The position of each component within a panel is interpreted relative to the panel. If you move the panel, its children move with it and maintain their positions on the panel.
Button Group		Button groups are like panels but are used to manage exclusive selection behavior for radio buttons and toggle buttons.

Callbacks: An Overview

After you have laid out your GUI, you need to program its behavior. The code you write controls how the GUI responds to events such as button clicks, slider movement, menu item selection, or the creation and deletion of components. This programming takes the form of a set of functions, called callbacks, for each component and for the GUI figure itself.

What Is a Callback?

A callback is a function that you write and associate with a specific GUI component or with the GUI figure. It controls GUI or component behavior by performing some action in response to an event for its component. This kind of programming is often called event-driven programming. When an event occurs for a component, MATLAB invokes the component's callback that is triggered by that event. As an example, suppose a GUI has a button that triggers the plotting of some data. When the user clicks the button, MATLAB calls the callback you associated with clicking that button, and the callback, which you have programmed, then gets the data and plots it. A component can be any control device such as a push button, list box, or slider. For purposes of programming, it can also be a menu or a container such as a panel or button group.

M-Files and FIG-Files

By default, the first time you save or run a GUI, GUIDE stores the GUI in two files:

- **A FIG-file**, with extension `.fig`, that contains a complete description of the GUI layout and the GUI components, such as push buttons, axes, panels, menus, and so on. The FIG-file is a binary file and you cannot modify it except by changing the layout in GUIDE.
- **An M-file**, with extension `.m`, that initially contains initialization code and templates for some callbacks that are needed to control GUI behavior. You must add the callbacks you write for your GUI components to this file. When you save your GUI the first time, GUIDE automatically opens the

M-file in your default editor. The FIG-file and the M-file, usually reside in the same directory. They

correspond to the tasks of laying out and programming the GUI. When you lay out the GUI in the Layout Editor, your work is stored in the FIG-file. When you program the GUI, your work is stored in the corresponding M-file.

GUI M-File Structure

The GUI M-file that GUIDE generates is a function file. The name of the main function is the same as the name of the M-file. For example, if the name of the M-file is `mygui.m`, then the name of the main function is `mygui`. Each callback in the file is a subfunction of the main function. When GUIDE generates an M-file, it automatically includes templates for the most commonly used callbacks for each component. The M-file also contains initialization code, as well as an opening function callback and an output function callback. You must add code to the component callbacks for your GUI to work as you want. You may also want to add code to the opening function callback and the output function callback. The major sections of the GUI M-file are ordered as shown in the following table.

Section	Description
Comments	Displayed at the command line in response to the help command. Edit these as necessary for your GUI.
Initialization	GUIDE initialization tasks. Do not edit this code.
Opening function	Performs your initialization tasks before the user has access to the GUI.
Output function	Returns outputs to the MATLAB command line after the opening function returns control and before control returns to the command line.
Component and figure callbacks	Control the behavior of the GUI figure and of individual components. MATLAB calls a callback in response to a particular event for a component or for the figure itself.
Utility/helper functions	Perform miscellaneous functions not directly associated with an event for the figure or a component.

Example1: Making Time shift and Time scale .

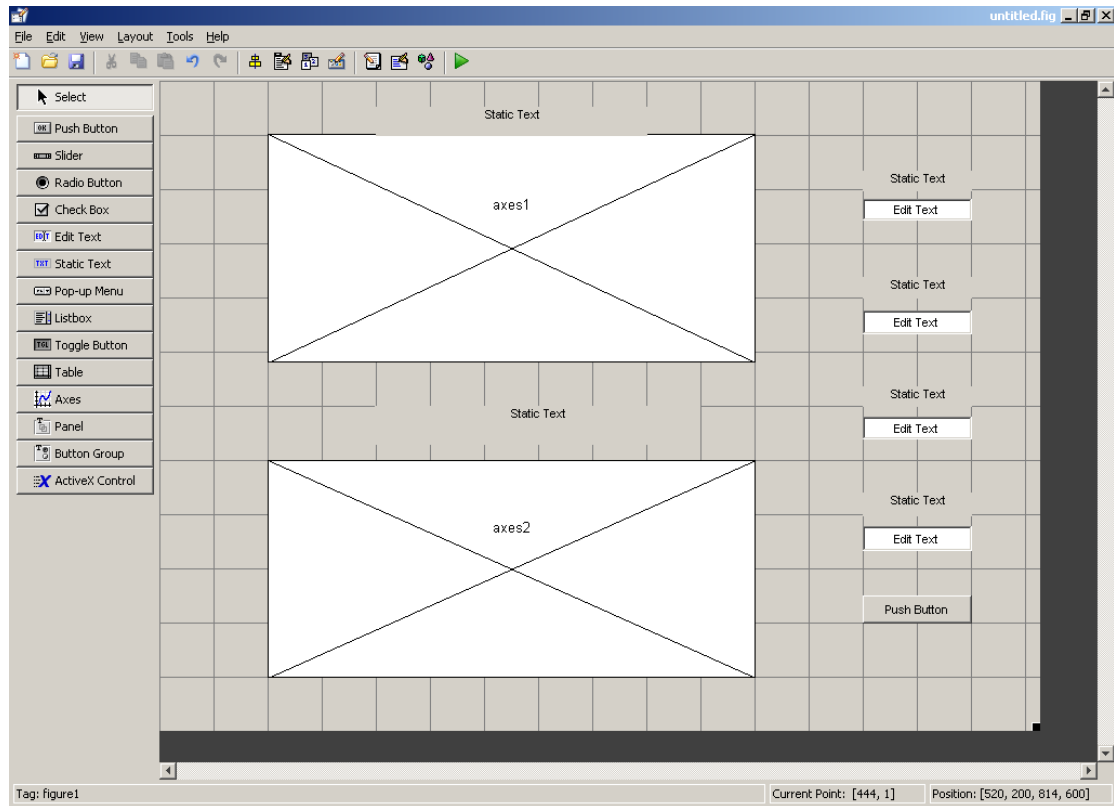
This GUI example plot a function $f(t)=t*[u(t+1)-u(t-1)]+u(t-1)-u(t-3)$ and plot a time shifted ,time scaled version of f(t) which has the general form $cf(at+b)$. The user can input variables values of a,b and c. The original function appears on GUI axis1 and the other on GUI axis2.

GUI-building techniques illustrated in this example include:

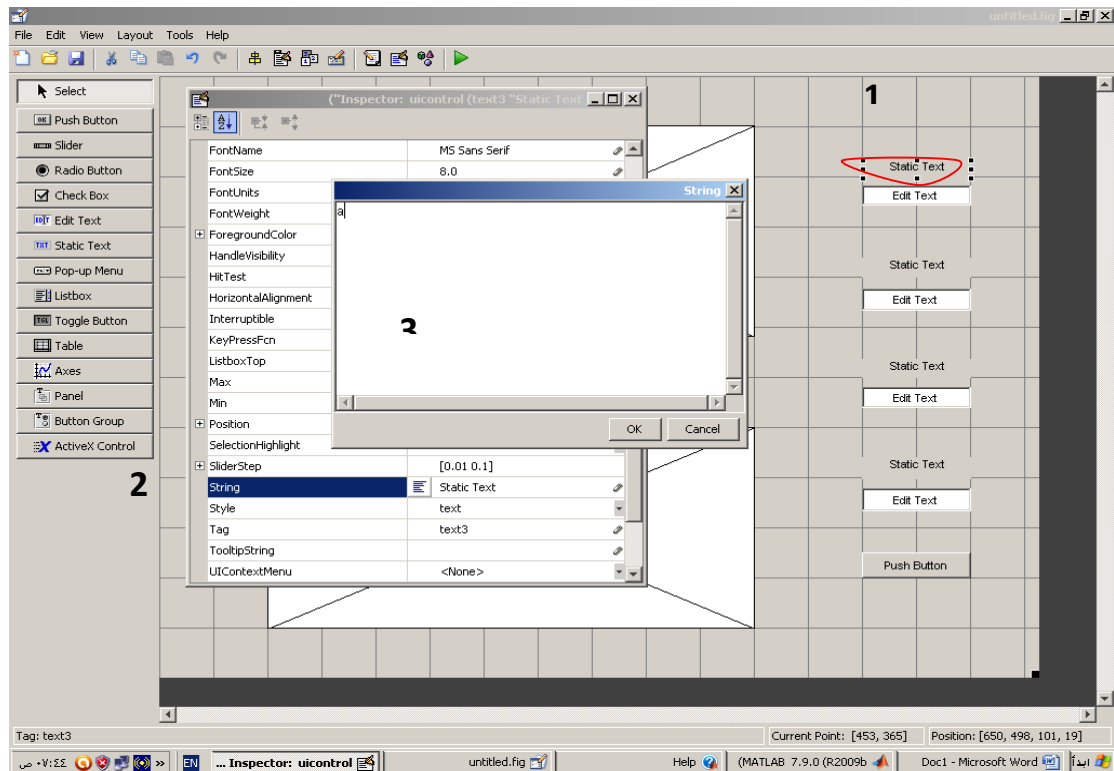
- Controlling which axes object is the target for plotting commands.
- Using edit text controls to read numeric input and MATLAB expressions.
- Converting user inputs from strings to numbers and validating the result.

Designing steps

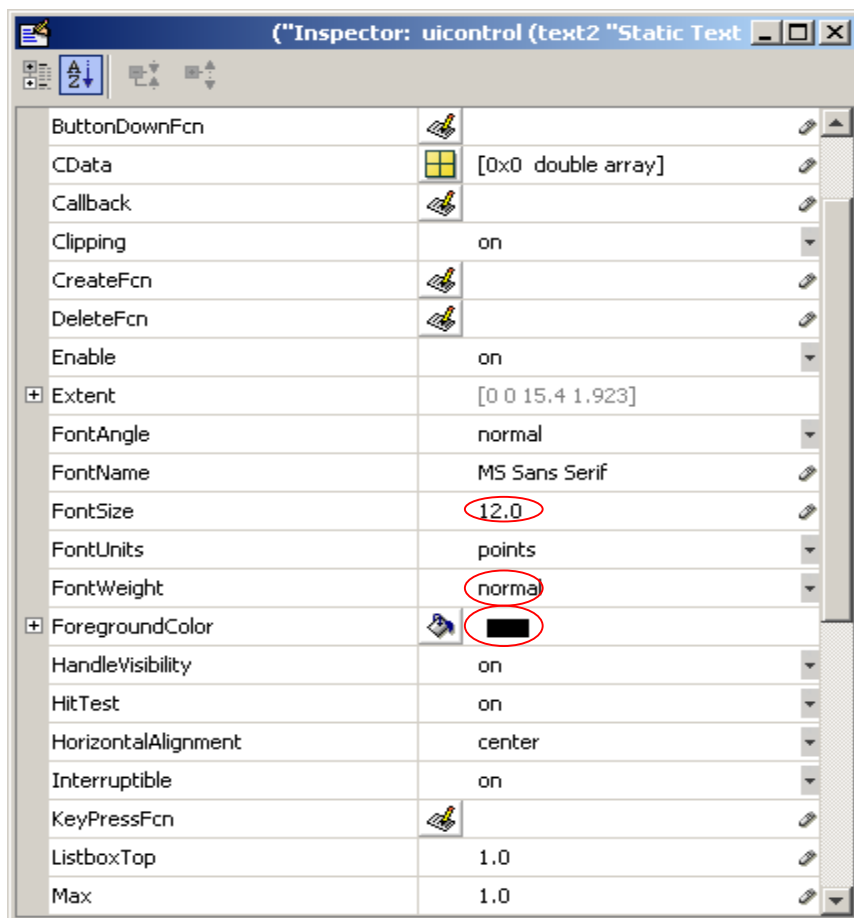
1-Put the following component in the figure .



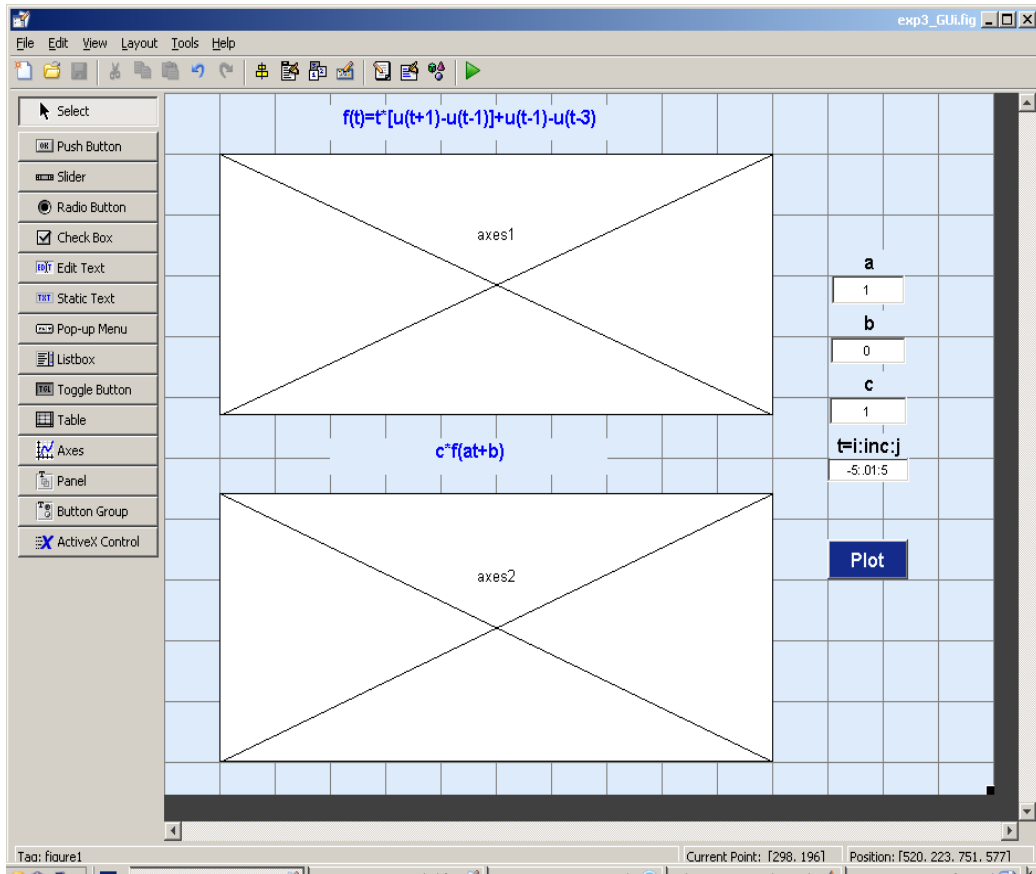
2. Change the name of the (*static text*) by double clicking on each one as follow



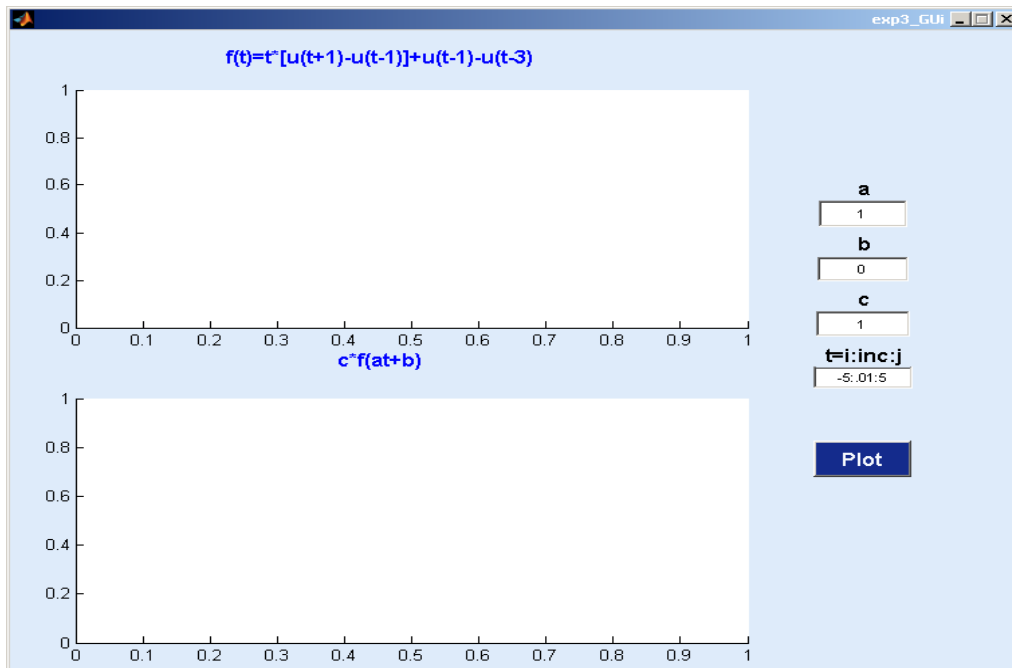
3-change the size ,colour and weight of the text as follow



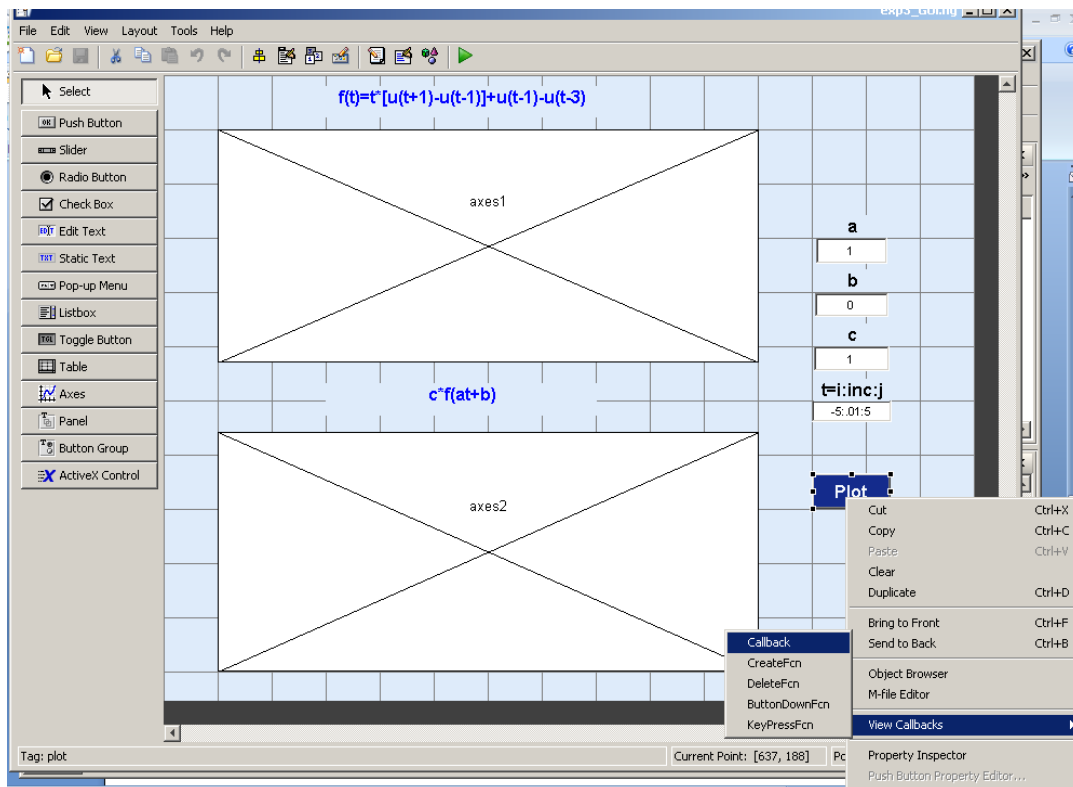
4- The final design will be as follow ,



5- push on the green arrow (Run) and save the design .



6-Right click on the Plot button and select *view callback* and choose *callback* .



7-The M file will open as follow

```

137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155

```

```

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

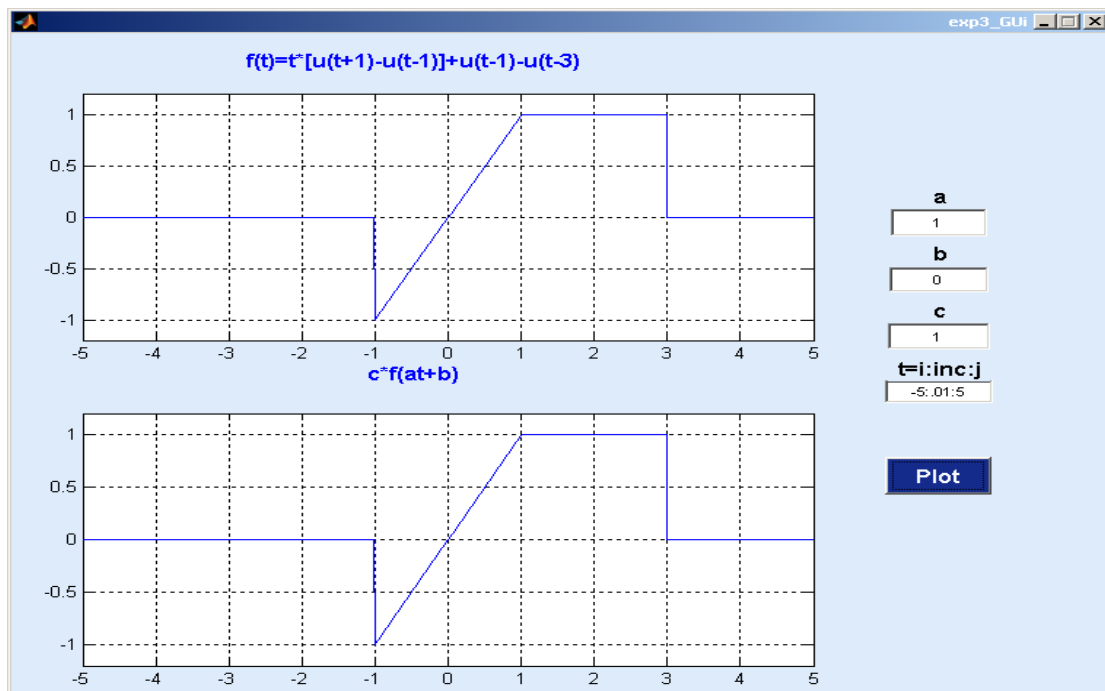
% --- Executes on button press in plot.
function plot_Callback(hObject, eventdata, handles)
% hObject handle to plot (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get user input from GUI
axes(handles.axes1)
a = str2double(get(handles.a,'String'));
b = str2double(get(handles.b,'String'));
c = str2double(get(handles.c,'String'));
t = eval(get(handles.t,'String'));

```

8-Write the following code under the Plot_callback function

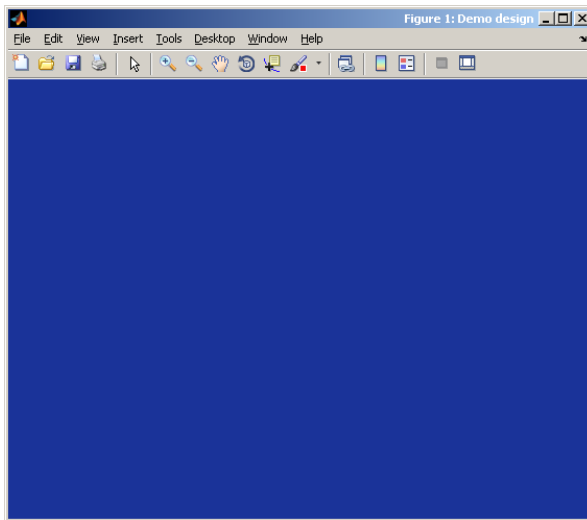
```
axes(handles.axes1)
a = str2double(get(handles.a, 'String'));
b = str2double(get(handles.b, 'String'));
c = str2double(get(handles.c, 'String'));
t = eval((get(handles.t, 'String')));
%plot the first function
f=inline('(t>=1)&(t<3)', 't');
plot(t, f(t))
ylim ([ min(f(t))-0.2 max(f(t))+0.2])
grid on
%plot the second function
f1=c.*f(a*t+b);
axes(handles.axes2)
plot(t, f1)
ylim ([ min(f1)-0.2 max(f1))+0.2])
grid on
```

9-Run and enjoy

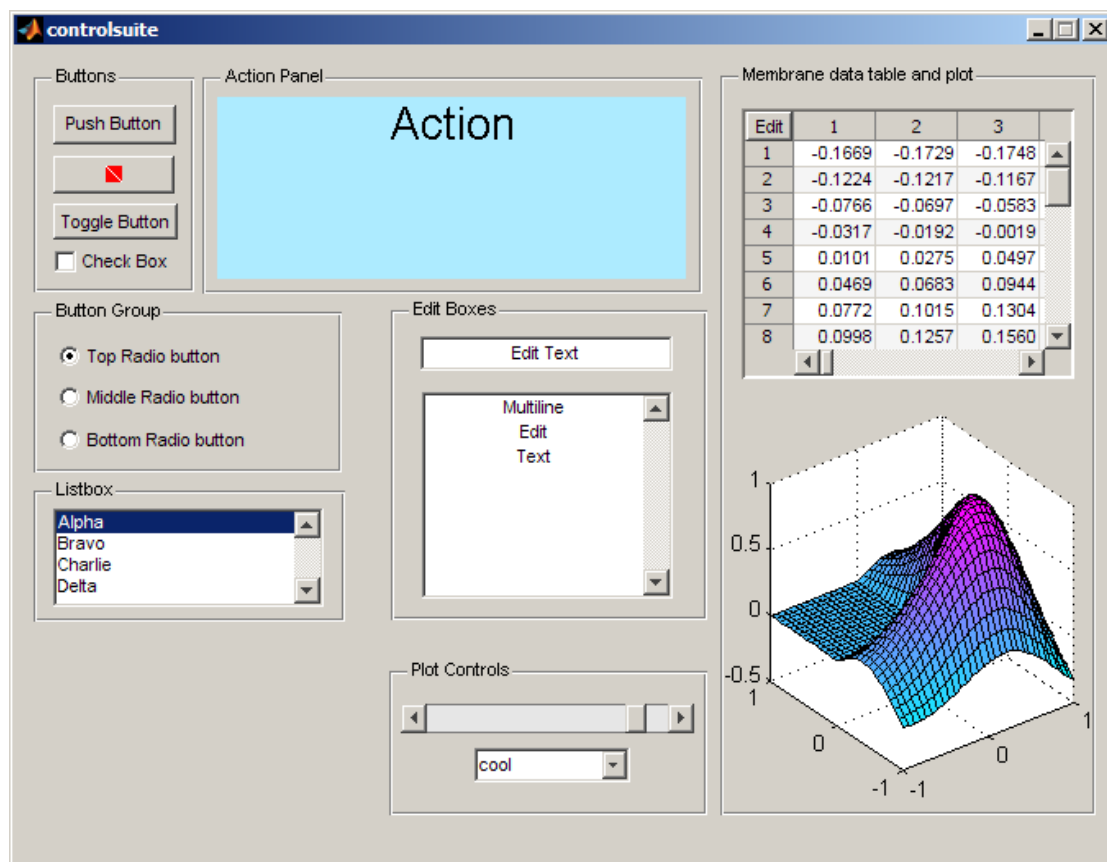


Example 2: Design programmatically

```
MainFigure = figure('Color',[0.1 0.2 0.6], 'Name', 'Demo design')
```



Try this (controlsuite in the help)



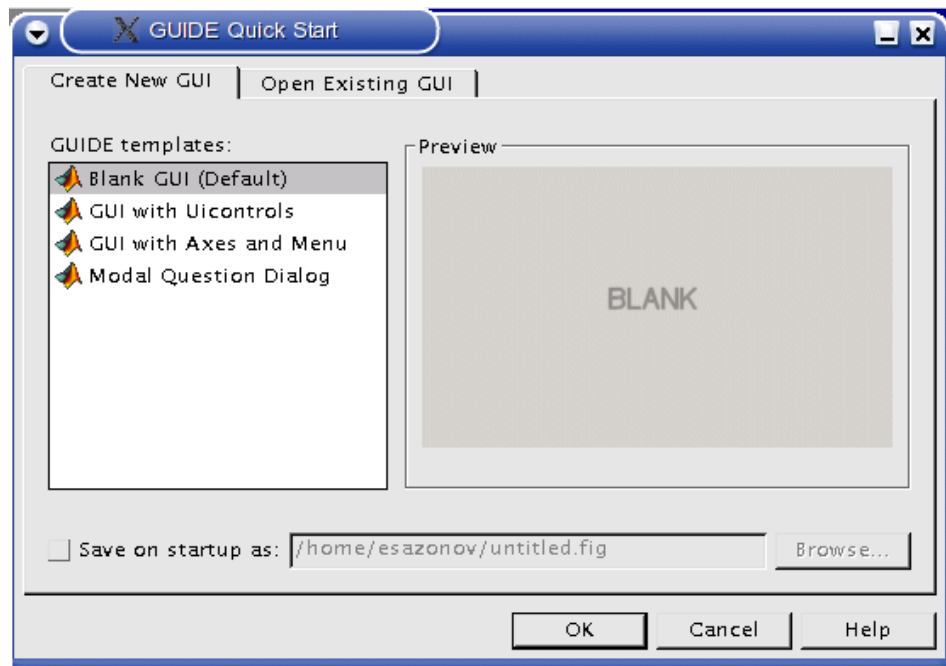
Exercise

Building GUI interfaces in Matlab

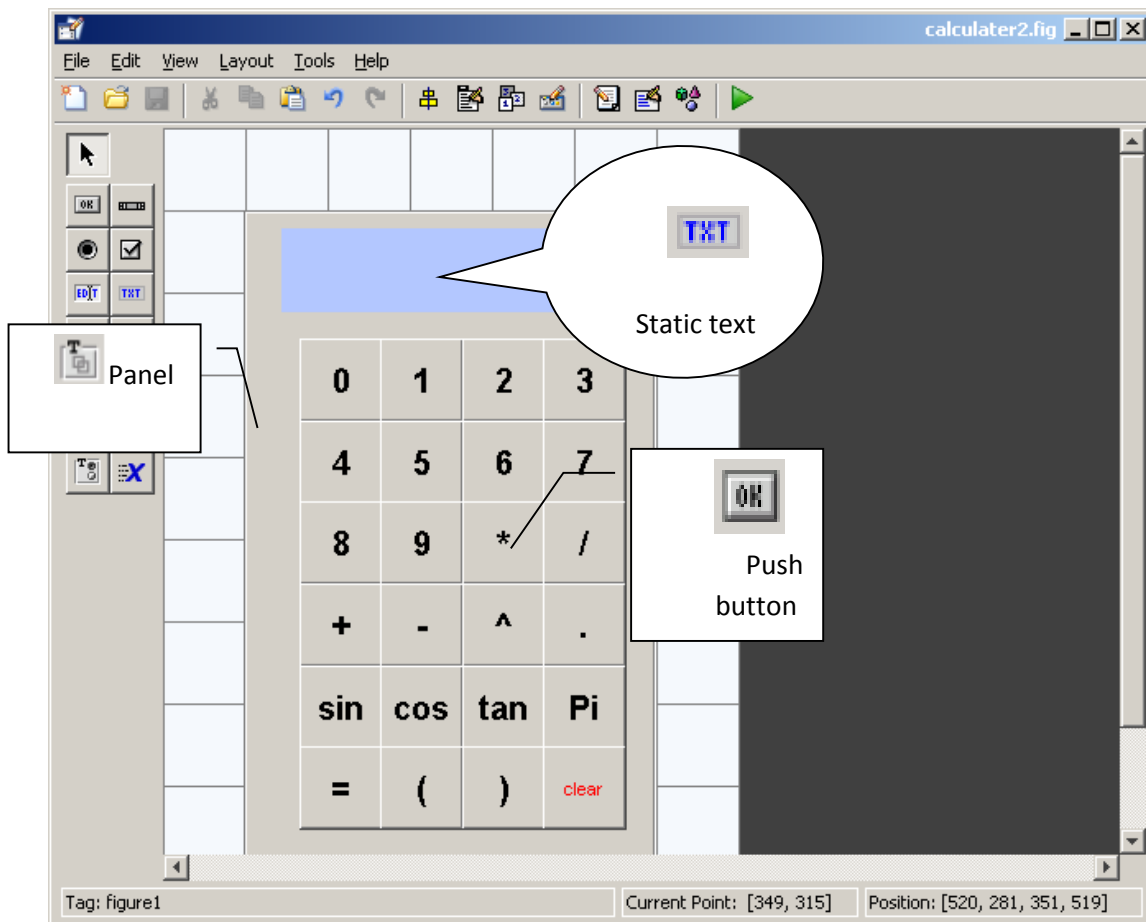
We will build a simple calculator to do that.

Start gui builder by typing

>>guide



1: insert the following component and rename it as in the figure



2-Rename the tage of each push button such that it indicate it in order to make thing more easy for example (0) called its tag as (zero) and so on ...

3-under the callback of zero button write this code

```
% --- Executes on button press in zero.
function zero_Callback(hObject, eventdata, handles)
% hObject    handle to zero (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% هذه برمجة للرقم صفر
% حتي لا تمسح شاشة التكتست عند اضافة رقم جديد نعمل التالي
OLDstring=get(handles.text1,'string');% نستخدم هذا الامر
لتخزين محتوى التكتست القديم
NEWstring=('0'); هذا النص الجديد المراد اظهاره مع النص القديم

textstring=strcat(OLDstring, NEWstring);
يقوم هذا الامر بوضع النص القديم والنص الحالي في تكتست واحد

set(handles.text1,'string',textstring); هذا الامر الذي يظهر
النص لكلي
```

Tray to put this code instead of the above and see what happen

```
zero=get(handles.zero,'string')
set(handles.text1,'string',zero)
```

Note the name of the static text tag is (text1)

4- apply the same code for all the button **except the (=) and clear button**.just change the ('0') to ('1') , ('2'), ('3'), ('4') and so on

5-for the *sin* ,*cos* ,and *tan* it consider the angle in radian and to convert it to degree you must multiply by (pi/180)

So the code is

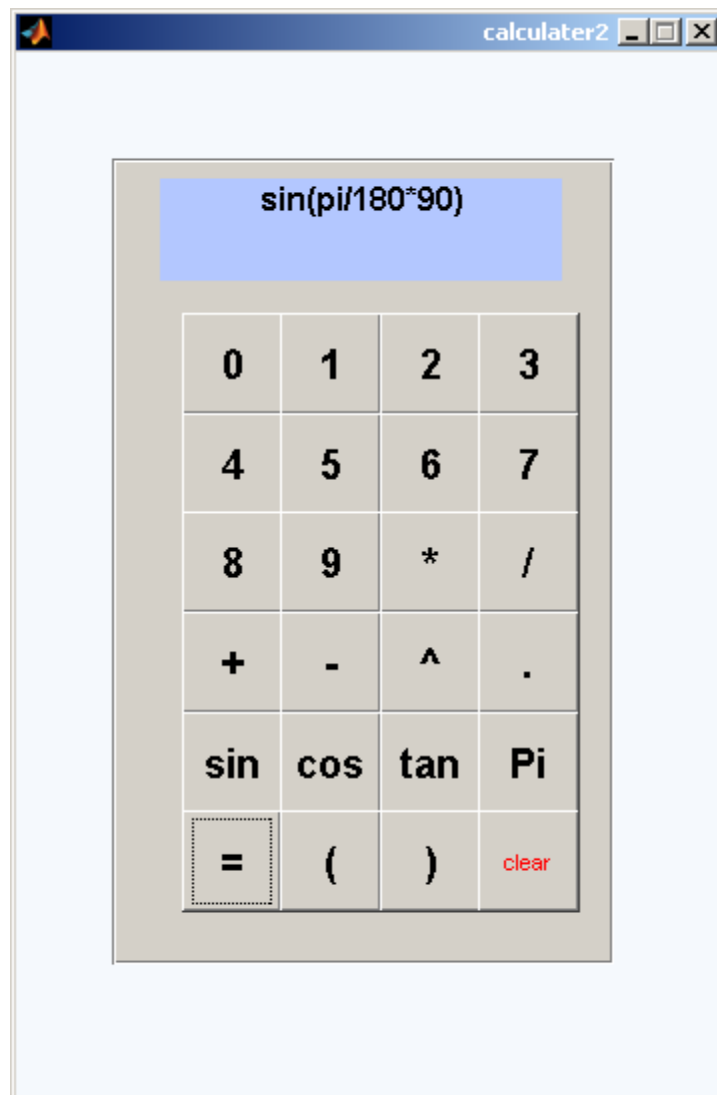
```
textstring=get(handles.text1,'string');
textstring=strcat(textstring,'sin(pi/180*');
set(handles.text1,'string',textstring);
```

6 -For the clear button write this code

```
set(handles.text1,'String',' ');
```

7-For the equal button write this code

```
textstring=get(handles.text1,'string')
textstring=eval(textstring)
set(handles.text1,'string',textstring);
```



Any addition to the calculator will be considered