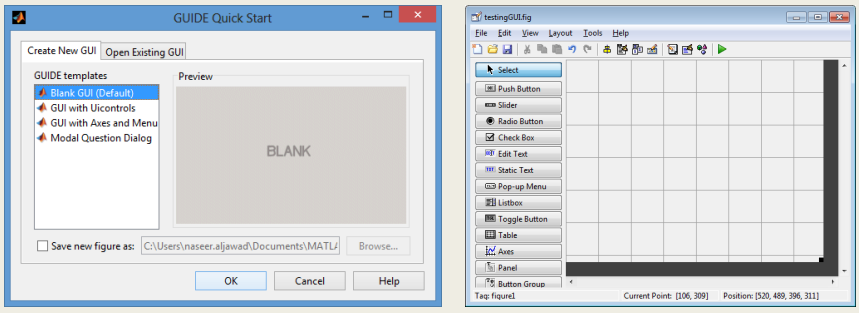
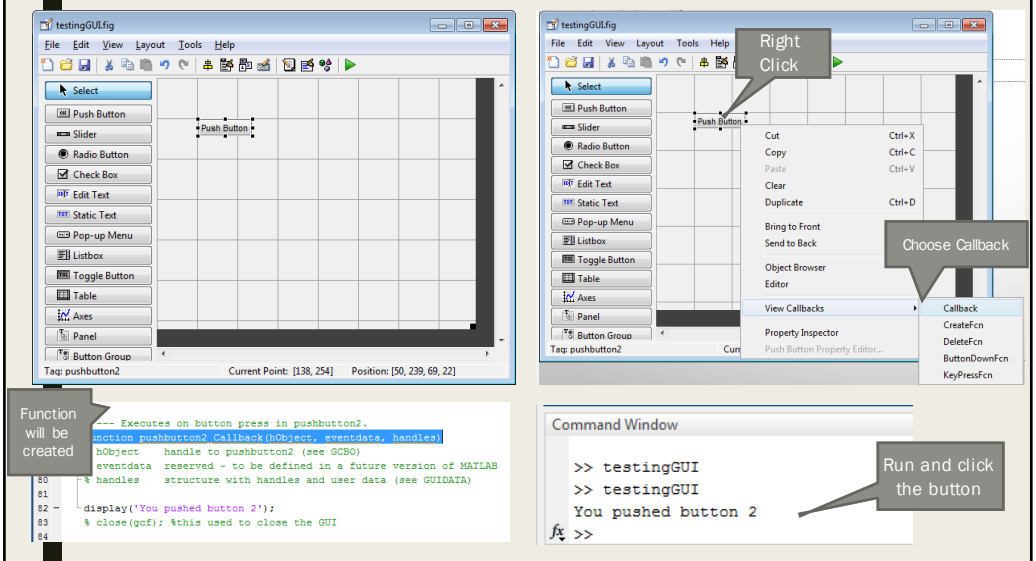
Introduction to Matlab's Graphical User Interface (GUI)

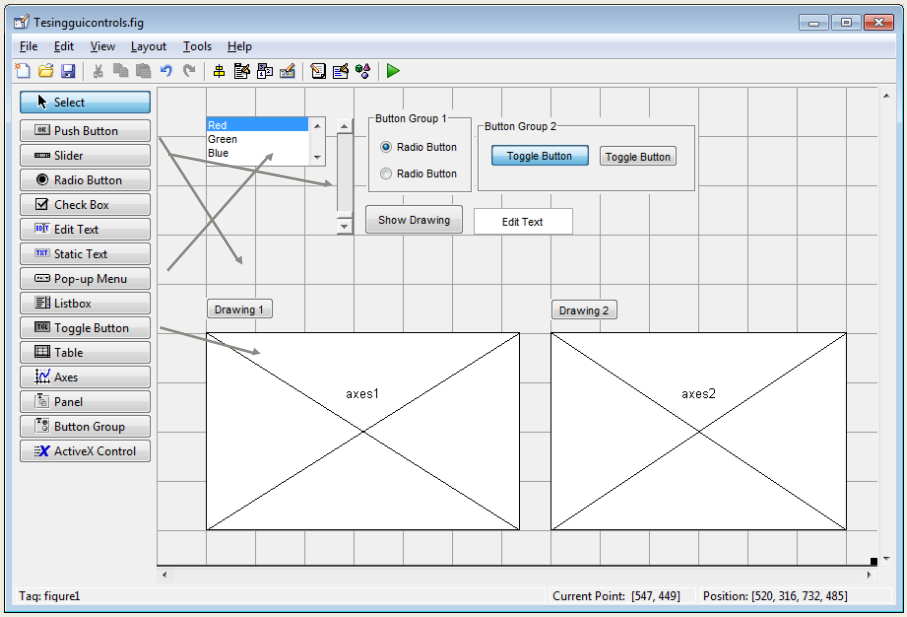
- Using "guide"
- "Guide" creates interface on a Figure window and code in an M-file. Some hidden code.
- The M-file and the Figure window usually have the same name.
- We add "**objects**" in the "Layout editor" and complete "**Callback** functions" in M-file editor.



Example of adding a push button …



Add components



Edit Text

Edit Text

This code is an example of an edit text box callback function in GUIDE. Associate this function with the edit text box `Callback` property to make it execute when the end user types characters in the text box.

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents as double
input = get(hObject,'String');
display(input);
```

When the end user types characters inside the edit text box and presses the **Enter** key, the callback function retrieves the string value and displays it in the Command Window.

To enable users to enter multiple lines of text, set the `Max` and `Min` properties to numeric values that satisfy `Max - Min > 1`. For example, set `Max` to 2, and `Min` to 0 to satisfy the inequality. In this case, the callback function triggers when the end user clicks on an area in the GUI that is outside of the edit text box.

Slider

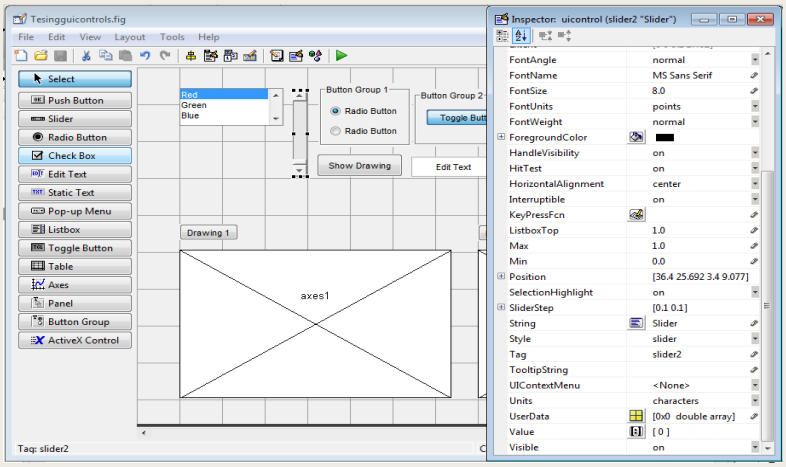
Slider

This code is an example of a slider callback function in GUIDE. Associate this function with the slider `Callback` property to make it execute when the end user moves the slider.

```
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine...
slider_value = get(hObject,'Value');
display(slider_value);
```

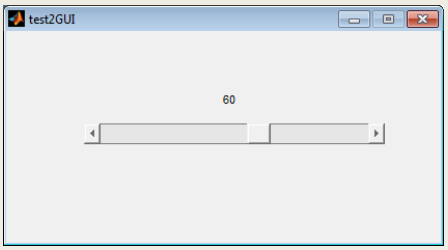
When the end user moves the slider, the callback function gets the current value of the slider and displays it in the Command Window. By default, the slider's range is [0, 1]. To modify the range, set the slider's `Max` and `Min` properties to the maximum and minimum values, respectively.

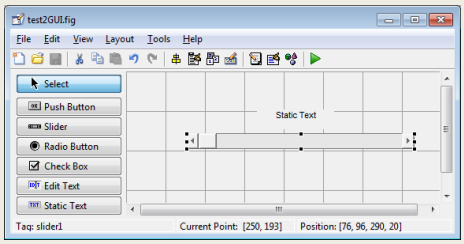
Slider property



Link Slider to static text

```
75
76      % --- Executes on slider movement.
77      function slider1_Callback(hObject, eventdata, handles)
78
79 -      Myval=get(hObject,'Value');
80 -      set(handles.text1,'String',Myval);
81
```





Using List Box

List Box

Populate Items in the List Box

If you are developing a GUI using GUIDE, use the list box `CreateFcn` callback to add items to the list box.

This code is an example of a list box `CreateFcn` callback that populates the list box with the items, `Red`, `Green`, and `Blue`.

```
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns

% Hint: listbox controls usually have a white background on Windows.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject,'String',{'Red';'Green';'Blue'});
```

The last line, `set(hObject,'String',{'Red';'Green';'Blue'})`, populates the contents of the list box.

Write the Callback Function

This code is an example of a list box callback function in GUIDE. Associate this function with the list box `Callback` property to make it execute when a selects an item in the list box.

```
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns contents
% contents{get(hObject,'Value')} returns selected item from listbox1
items = get(hObject,'String');
index_selected = get(hObject,'Value');
item_selected = items{index_selected};
display(item_selected);
```

When the end user selects an item in the list box, the callback function performs the following tasks:

- Gets all the items in the list box and stores them in the variable, `items`.
- Gets the numeric index of the selected item and stores it in the variable, `index_selected`.
- Gets the string value of the selected item and stores it in the variable, `item_selected`.
- Displays the selected item in the MATLAB® Command Window.

Radio Button

Radio Button

This code is an example of a radio button callback function in GUIDE. Associate this function with the radio button `Callback` property to make it execute when the end user clicks on the radio button.

```
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (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,'Value') returns toggle state of radiobutton1

if (get(hObject,'Value') == get(hObject,'Max'))
        display('Selected');
else
        display('Not selected');
end
```

The radio button's `Value` property matches the `Min` property when the radio button is not selected. The `Value` changes to the `Max` value when the radio button is selected. This callback function gets the radio button's `Value` property and then compares it with the `Max` and `Min` properties. If the button is selected, then the function displays `'Selected'` in the Command Window. If the button is not selected, then the function displays `'Not selected'`.

Toggle Button

Toggle Button

This code is an example of an example of a toggle button callback function in GUIDE. Associate this function with the toggle button `Callback` property to make it execute when the end user clicks on the toggle button.

```
function togglebutton1_Callback(hObject,eventdata,handles)
% hObject    handle to togglebutton1 (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,'Value') returns toggle state of togglebutton1
button_state = get(hObject,'Value');
if button_state == get(hObject,'Max')
        display('down');
elseif button_state == get(hObject,'Min')
        display('up');
end
```

Button Group

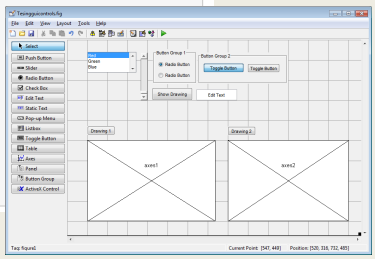
Button Group

Button groups are similar to panels, but they also manage exclusive selection of radio buttons and toggle buttons. When a button group contains multiple radio buttons or toggle buttons, the button group allows the end user to select only one of them.

Do not code callbacks for the individual buttons that are inside a button group. Instead, use the button group's `SelectionChangedFcn` callback to respond when the end user selects a button.

This code is an example of a button group `SelectionChangedFcn` callback that manages two radio buttons and two toggle buttons.

```
function uibuttongroup1_SelectionChangedFcn(hObject, eventdata,
handles)
% hObject    handle to the selected object in uibuttongroup1
% eventdata  structure with the following fields
%       EventName: string 'SelectionChanged' (read only)
%       OldValue: handle of the previously selected object or empty
%       NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
switch get(eventdata.NewValue,'Tag') % Get Tag of selected object.
    case 'radiobutton1'
        display('Radio button 1');
    case 'radiobutton2'
        display('Radio button 2');
    case 'togglebutton1'
        display('Toggle button 1');
    case 'togglebutton2'
        display('Toggle button 2');
end
```

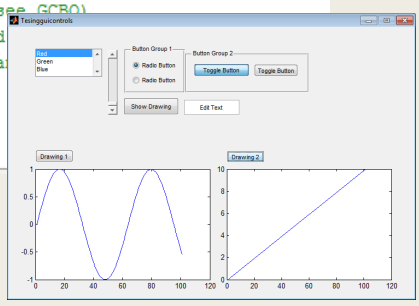


Using more than one axes

Editor - C:\Users\Naseer\Dropbox\MM-Northampton\Drawing\GUI\Tesingguicontrols.m

untitled.m | mygui.m | NaseerBlankFig.m | NasGUI.m | Tesingguicontrols.m

```
196    function pushbutton2_Callback(hObject, eventdata, handles)
197    % hObject    handle to pushbutton2 (see GCBO)
198    % eventdata  reserved - to be defined in a future version of MATLAB
199    % handles    structure with handles and user data (see GUIDATA)
200
201 -  axes(handles.axes1);
202 -  plot(sin(0:.1:10));
203
204    % --- Executes on button press in pushbutton3.
205    function pushbutton3_Callback(hObject, eventdata, handles)
206    % hObject    handle to pushbutton3 (see GCBO)
207    % eventdata  reserved - to be defined
208    % handles    structure with handles a
209
210 -  axes(handles.axes2);
211 -  plot(0:.1:10);
```



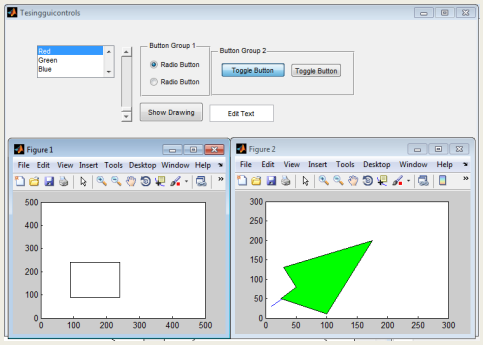
Run an outside script from a button

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

display('Hello you have pressed pushbutton');
Drawing;
%close(gcf);
```

Editor - C:\Users\Naseer\Dropbox\MM-Northampton\Drawing\GUI\Drawing.m*

untitled.m | mygui.m | NaseerBlankFig.m | NasGUI.m | Tesingguico

```
figure(1), rectangle('Position', [0 0 500 500]);
figure(1), MyRec=rectangle('Position', [90 90 150 150]);

x=[10 120];
y=[30 150];
figure(2), MyLine=line(x,y);
x1=[50 25 100 175 30];
y1=[80 50 10 200 130];
figure(2), rectangle('Position', [0 0 300 300]);
figure(2), MyPoly= patch (x1,y1,'g');
```



Try it.

- Type “guide”.
- The layout window appears.
- Add components one by one.
- Check the action for each component.
- Right click in the component, go down to “view callbacks”, and select “Callback”

References

- http://uk.mathworks.com/help/matlab/gui-development.html
- http://uk.mathworks.com/help/matlab/creating_guis/add-code-for-components-in-callbacks.html
- http://uk.mathworks.com/help/matlab/ref/uicontrol.html?nocookie=true
- http://uk.mathworks.com/help/matlab/gui-building-basics.html