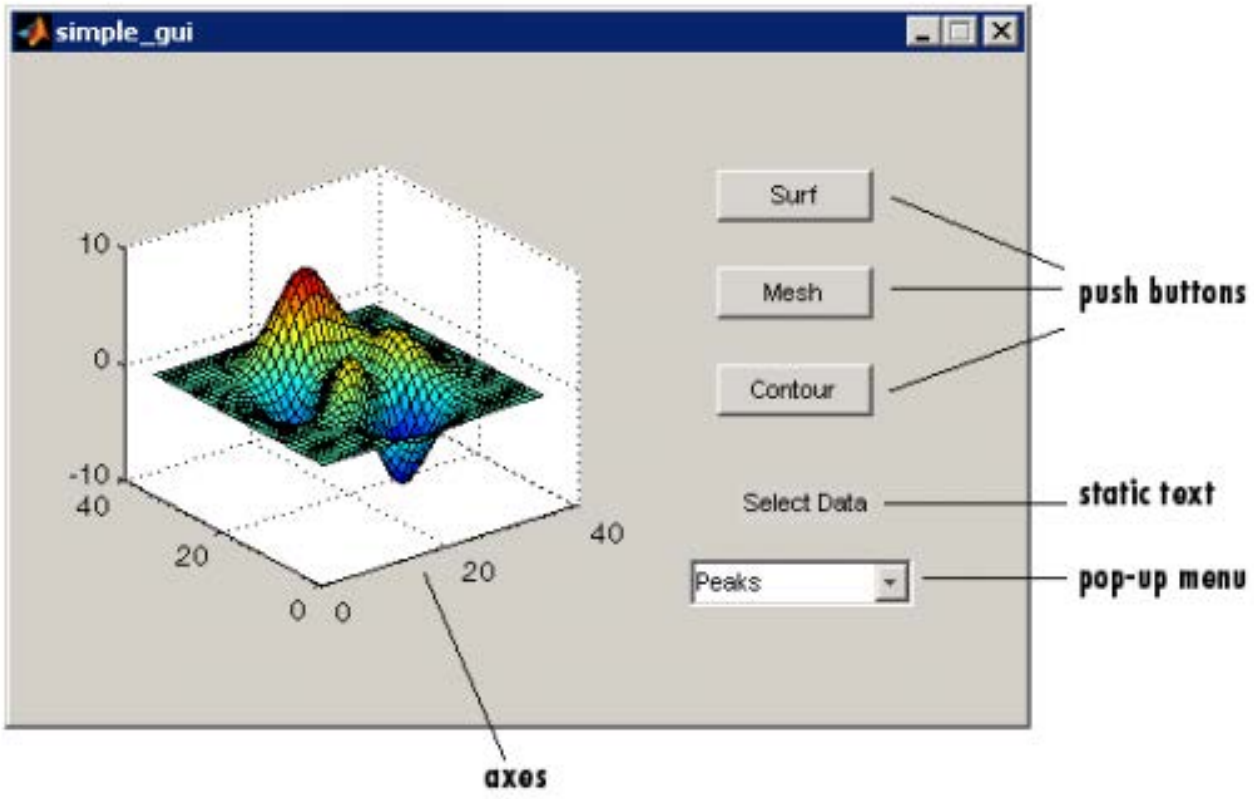
Lab 6: Graphical User Interface

What is 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.
- Each component, and the GUI itself, is associated with one or more user-written routines known as callback functions.
- 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. You, as the creator of the GUI, provide these callbacks.
- This kind of programming is often referred to as event-driven programming.
- The writer of a callback has no control over the sequence of events that leads to its execution or, when the callback does execute.



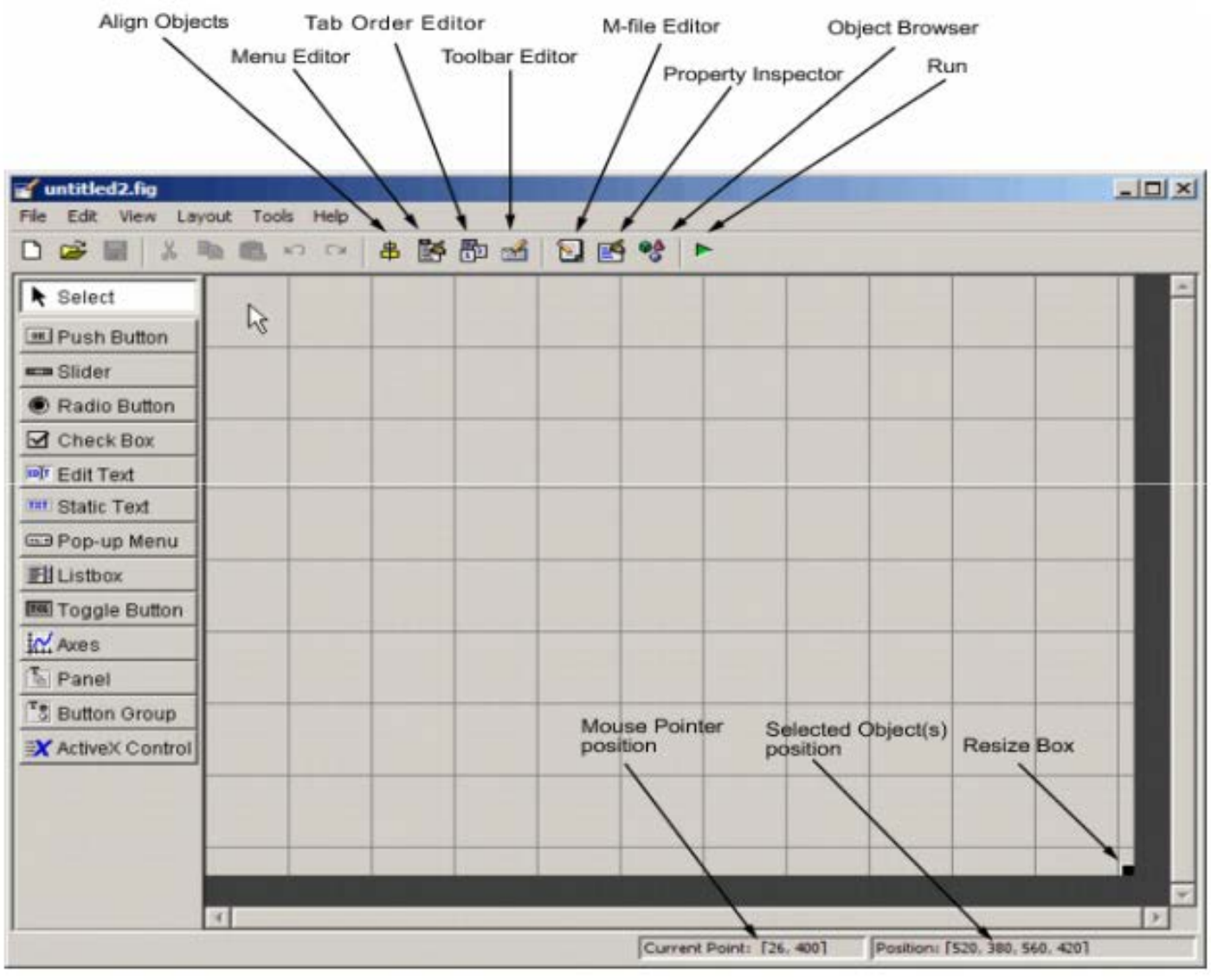
Structure of GUI in Matlab:

- Any GUI in MATALB is associated with two files:

1. Figure file with extension .fig that contains all the components or controls to be used in the GUI application. The FIG -file is a binary file and you cannot modify it except by changing the layout in GUIDE.
2. M-file that contains the code for the GUI initialization and the callbacks/functions for different components in the GUI layout and other hidden functions of the GUI.

Creating GUI using GUIDE:

- GUIs in Matlab can be created either programmatically or by using the GUIDE tool.
- Type guide on the command prompt to start the GUIDE tool.
- With this tool, you can use the mouse to add different components and set their properties (size, text, callbacks)



Use This Tool...	To...
Layout Editor	Select components from the component palette, at the left side of the Layout Editor, and arrange them in the layout area. See Adding Components to the GUI for more information.
Figure Resize Tab	Set the size at which the GUI is initially displayed when you run it. See Setting the GUI Size for more information.
Menu Editor	Create menus and context, i.e., pop-up, menus. See Creating Menus for more information.
Align Objects	Align and distribute groups of components. Grids and rulers also enable you to align components on a grid with an optional snap-to-grid capability. See Aligning Components for more information.
Tab Order Editor	Set the tab and stacking order of the components in your layout. See Setting Tab Order for more information.
Toolbar Editor	Create Toolbars containing predefined and custom push buttons and toggle buttons. See Creating Toolbars for more information.
Icon Editor	Create and modify icons for tools in a toolbar. See Creating Toolbars for more information.
Property Inspector	Set the properties of the components in your layout. It provides a list of all the properties you can set and displays their current values.
Object Browser	Display a hierarchical list of the objects in the GUI. See Viewing the Object Hierarchy for more information.
Run	Save and run the current GUI. See Saving and Running a GUIDE GUI for more information.
M-File Editor	Display, in your default editor, the M-file associated with the GUI. See GUI Files: An Overview for more information.
Position Readouts	Continuously display the mouse cursor position and the positions of selected objects

Structure of the GUI m-file:

- The GUI M-file that GUIDE generates is a function file. It has the same name as the GUI figure file.
- It contains the callbacks for the GUI components which are subfunctions 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.

Components Properties

- Each GUI component has its own properties that can be changed by the property inspector or by the set function.
- Some of these properties are:
  - 'String', 'fontWeight', 'fontSize', 'fontName', 'fontAngle', which are used to format and specify the component label.
  - 'value' property depends on component type.
  - 'Tag' is the name of the component that is used to get and set its properties in the code.

- 'Enable' is used to disable or enable the component.
- 'Visible' controls if the component is visible or not. ◦ 'Resize' enable or disable resizing fo components or the GUI figure.

The Get Function:

- The get function is extensively used in programming GUIs. It is used to get properties and values of the GUI component.
- Syntax `get(h,'PropertyName')`
- h is a handle to the component and propertyName is the name of the property that we want to get for the component specified by the handle.
- Components handles can specified using the handles.tagValue, where tagValues is the value of the tag property for the component.

The Set Function:

- The set function is the opposite of the get function.
- It is used to change the value of certain property of a component.
- Syntax `set(h,'PropertyName',PropertyValue,...)`
- The properties differ from component to component.

Example:

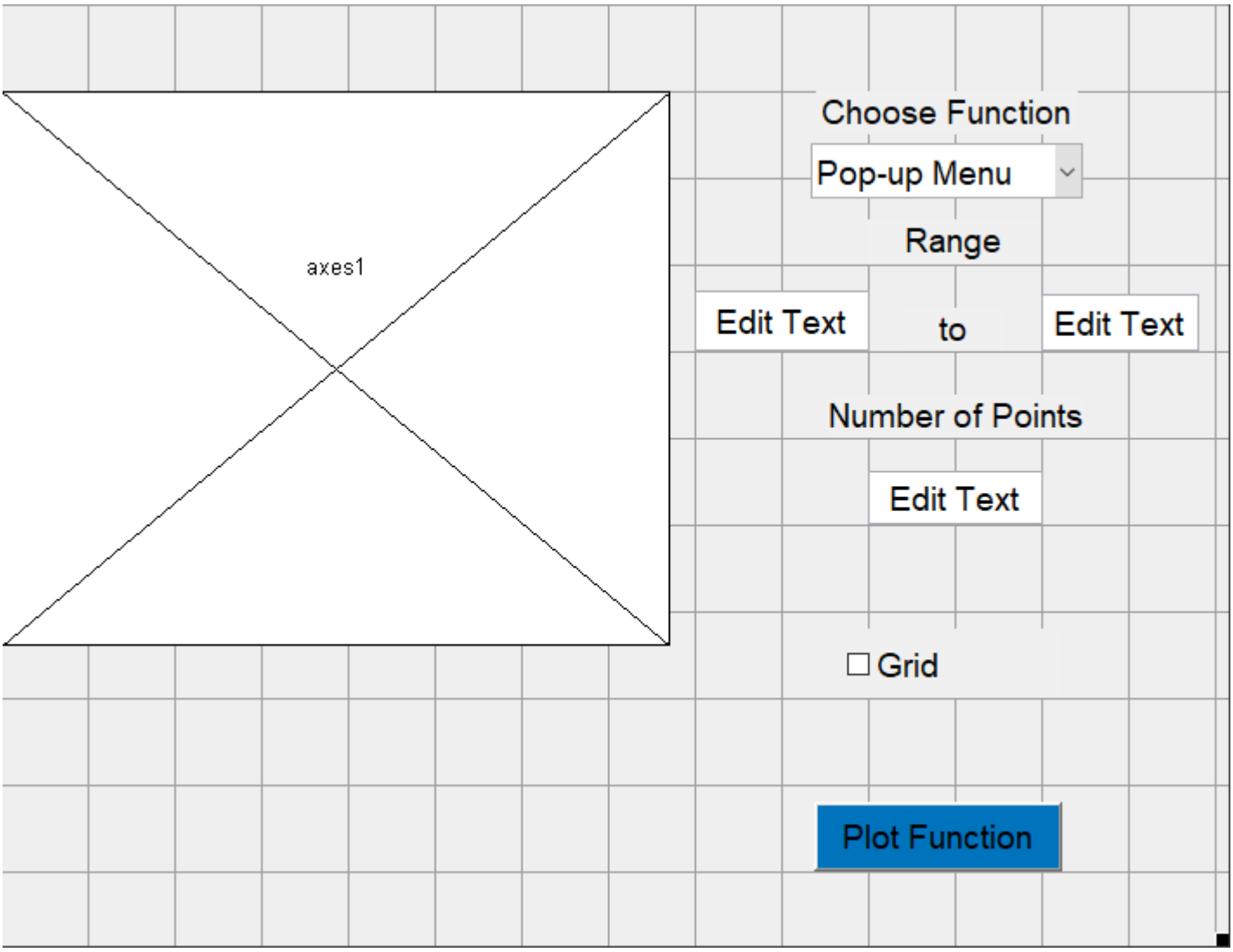
We want to create a GUI that plots a sine or a cosine function over a range that is specified by the user. When the GUI is started, the default plot is a sine wave over [-π,π] with 100 points. The user should have the option to show or hide gridline.

Required components

- Axes to plot the function.
- Popmenu to allow the user to choose which function to plot.
- Two text boxes to specify the minimum and the maximum of the range.
- One text box to specify the number of points to be used in the plot
- A push button to update the plot based on the user input.
- A check box to show or hide gridlines

Steps to do that:

1. Start the GUIDE tool by typing guide on the command prompt.
2. The guide quick start tool is started.
3. Select blank GUI then click OK.
4. Use the GUI preferences from the File menu to display the names of the components.
5. Using the mouse, drag and drop the required components in the figure.
6. Use the property editor to change the labels of different components. Change the property 'String' for each component.
7. Save your GUI with the name guiExample by clicking on save from the File menu.
8. Matlab will create two files guiExample.fig and guiExample.m
9. Matlab will open the editor to show the contents of the guiExample.m file.
10. The m-file contains the common callbacks for different components in the GUI.



Initialization Part:

- We want the GUI to generate a default plot when it is first started. This code is added to the opening function of the GUI.
- In the M-file editor, click on show function icon, A menu will appear listing all functions in the M-file.
- Select guiExample_OpeningFcn. The editor goes to the line where this function starts.
- Add the following code to opening function:

```
% Set the values for the text boxes to default values
% access the handles for the text boxes from the handles
structure
set(handles.xMin,'string',num2str(-pi))
set(handles.xMax,'string',num2str(pi))
set(handles.nPoints,'string',num2str(100))
```

```
% define default range, points, and function to be plotted
xMin = -pi , xMax = pi , nPoints = 100 ;
% define default range, points, and function to be plotted
x = xMin: (xMax-xMin)/(100-1):xMax;
y = sin(x) ;
plot(x,y), xlabel('x'),ylabel('y');
% Update handles structure
guidata(hObject, handles);
```

Programming the Push Button:

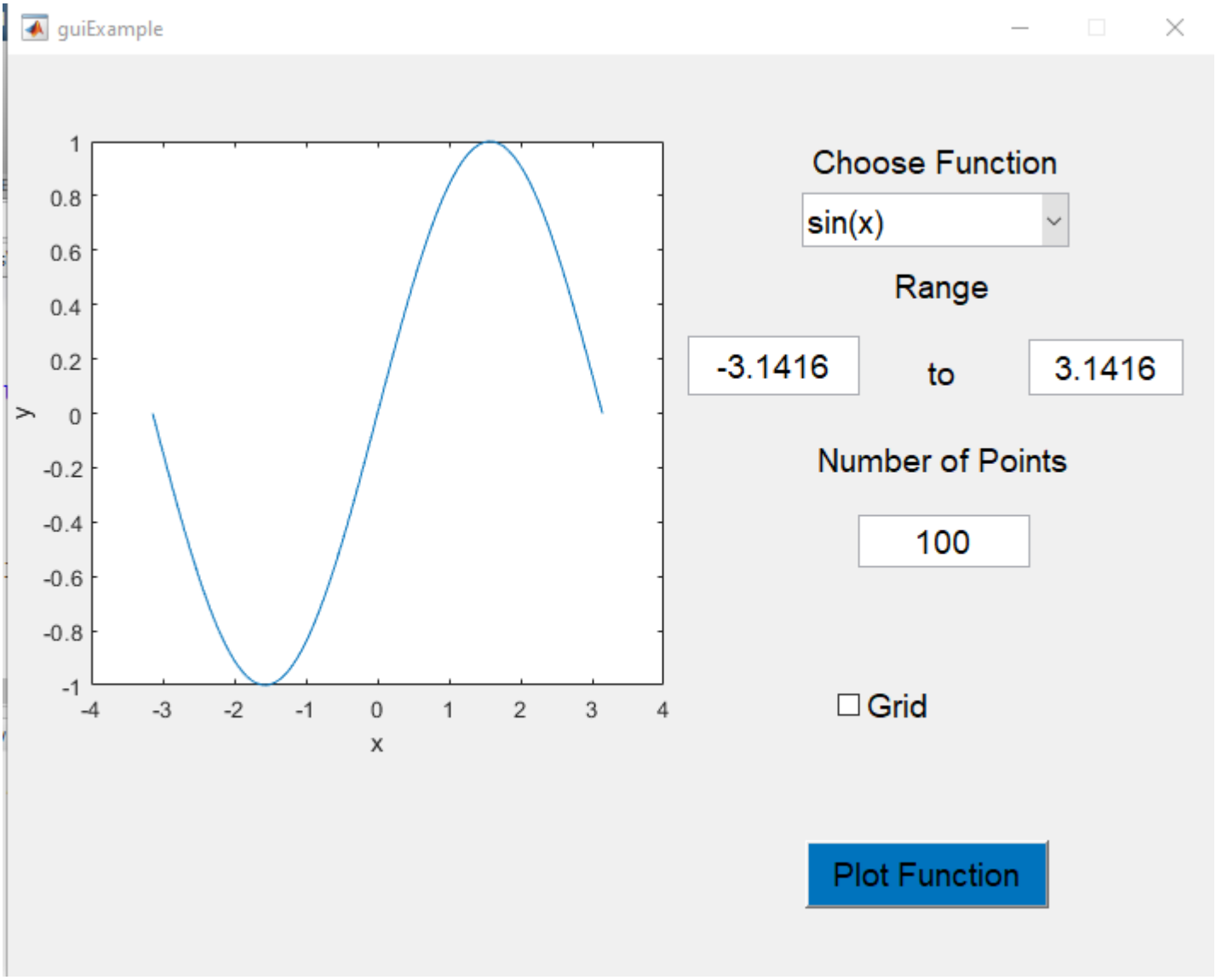
- The callback function of the push button should perform the following tasks:
  - Read the minimum and maximum of the plotting range from xMin and xMax textboxes.
  - Read the number of points to be used from nPoints textbox.
  - Read the function to be plotted from the popmenu.
  - Plot the required function based no user selection.
- Select the plotButton callback function from the function icon . . Insert the following code in the function.

```
% when this key is pressed, we have to read the values from the
text boxes
% and the function to be plotted, the plot.
xMin = str2num(get(handles.xMin,'string'));
xMax = str2num(get(handles.xMax,'string'));
nPoints = str2num(get(handles.nPoints,'string'));
func = get(handles.funcMenu,'value')
% generate x vector and plot the function based on the value of
the popmenu
x = xMin : (xMax-xMin)/(nPoints-1) : xMax ;
if func == 1
plot(x,sin(x));
title('Sin(x)'), xlabel('x'),ylabel('y');
else
plot(x,cos(x));
title('cos(x)'), xlabel('x'),ylabel('y');
end
%clear selection of the grid checkbox
set(handles.gridChkBox,'value',0)
```

Programming the Checkbox

- When this checkbox is on, gridlines should appear on the axes of the GUI. When it is not selected, no gridlines are shown.

- To program this, just add the grid command in the body of the checkBox callback function (the tag name is gridChkBox).
- Each time the checkbox is selected or deselected, the function is executed and the use of the grid command toggles the gridlines.

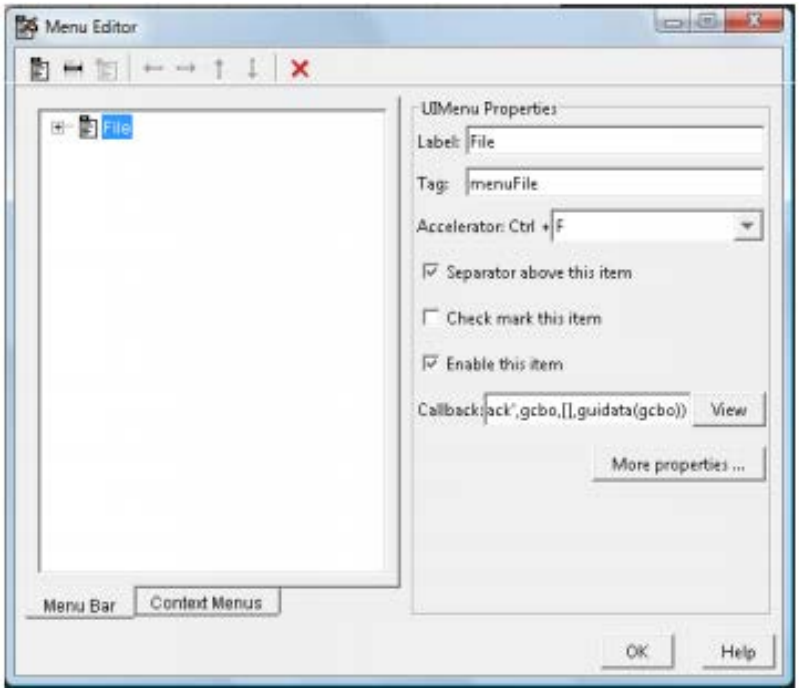


Notes:

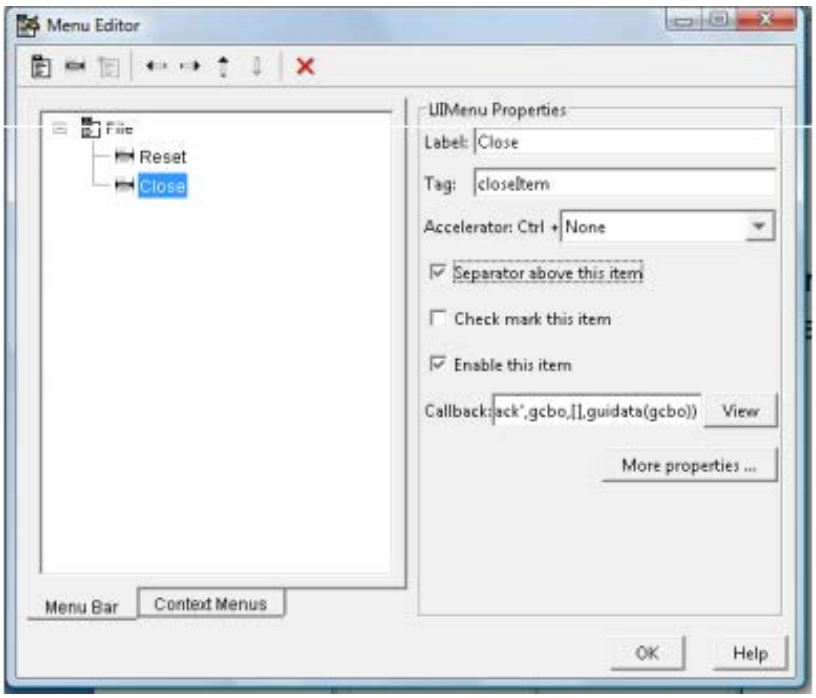
- To run your GUI, type its name on the command prompt. Or click on the run icon in GUIDE.
- To open your GUI, start the guide tool then click on the open existing GUI tab and select your file.
- To close your GUI from your code, use the close command.
- If you have more than one axes in your GUI and you want activate one of them for plotting, use axes(h) where h is a handle to axes. Example axes(handles.axes1) set axes1 in the GUI as the current axes.

Adding Menus:

- We can add menus to our GUI similar to those found in most GUI application. Let's add a File menu to the previous example. The menu should contain two items: close, to close the GUI, and reset to clear the axes and the text boxes.
- This can be done using the menu editor.
- Click on the menu editor icon to start the menu editor.
- Click on the new menu icon to insert a new menu. Click on the new menu and update its properties as shown.



- Click on the new menu item icon and add two items with labels Reset and Close, and tags resetItem and closeItem.



- Now if you run you GUI by clicking on the run icon in GUIDE, you will see the created menu.
- We need now to define the callbacks for the menus items.
- In the editor, select resetItem_callback from the function list and add the following line of code

```
cla(handles.axes1)
set(handles.xMin,'string','')
set(handles.xMax,'string','')
set(handles.nPoints,'string','')
set(handles.gridChkBox,'value',0)
```

- Similarly, add the close command to the callback function of closeItem to close the GUI.

The msgbox Command

- We can generate message windows by using the msgbox command.
- Syntax

```
msgbox(Message,Title,Icon)
```

- Message is the message to be displayed as string.
- Title is the title of the window as a string.
- Icon is the icon to be displayed on the message. It can be 'error','help', or 'warn'.

```
msgbox('Division by Zero','My Message','Warn')
```