Graphical User Interface

By

Dr. R. R. Manza

Objectives

- Introduction
- GUI Layout Tools
- Dialog Box
- List Box
- Accessing Variables from workspace
- GUI Components
- Solved GUI Examples
- Summary

Introduction

This chapter explains how to design a form and how to place various components and their layouts. The main objective of this chapter is to introduce the user with GUI environment of MATLAB. Users interface with the graphical controls is known as Graphical User Interface (GUI). Graphical controls or objects are buttons, labels, popmenu, menu, text field, slider etc. These controls are used to build graphical user interface. To learn GUI and to implement them MATLAB has some commands and inbuilt functions. GUI contains various styles, uicontrol and objects to get action from each object. We must have to write a code for every object. The program with object can be saved and launched by GUI. All of these task are simplified by GUIDE i.e. MATLAB's **G**raphical **U**ser **I**nterface **D**evelopment **E**nvironment. This tool allows us to layout the GUI, selecting and aligning the GUI component.

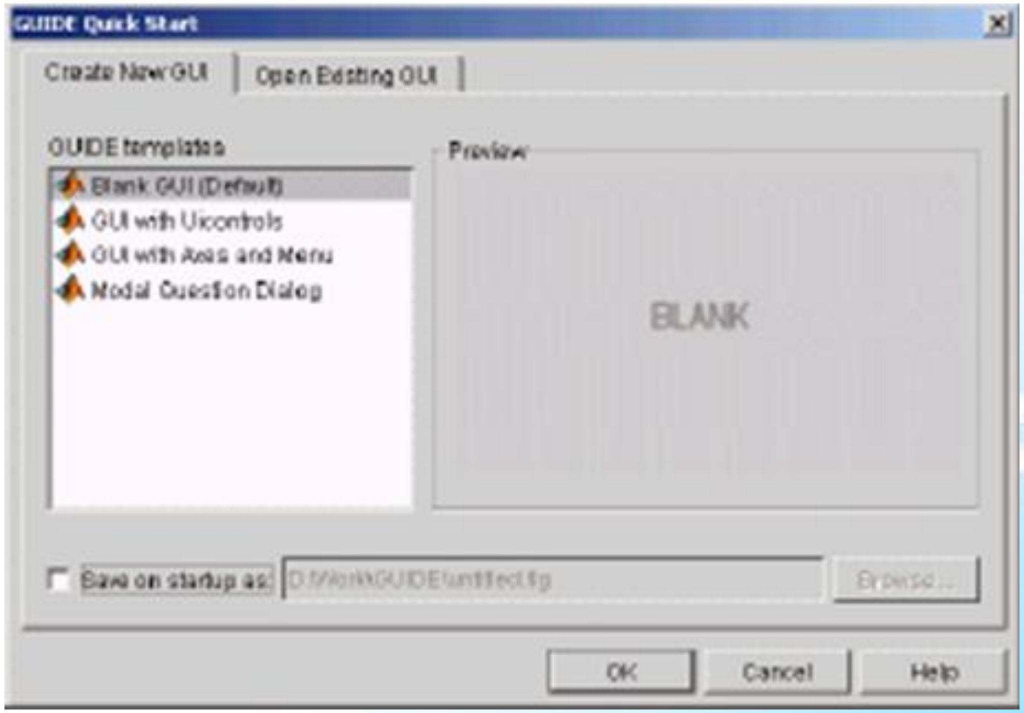
To implement GUI two basic tasks are necessary: 1) laying out the GUI components 2) programming the GUI components.

GUI layout tools

1. Layout editor: It adds and arranges objects in the figure window.
2. Alignment Tool: Align objects with respect to each other.
3. Property Inspector: Inspect and set property values.
4. Object browser: observe a hierarchical list of the Handle Graphics objects in the current MATLAB session.
5. Menu Editor: Create window menu and context menus.

Cont..

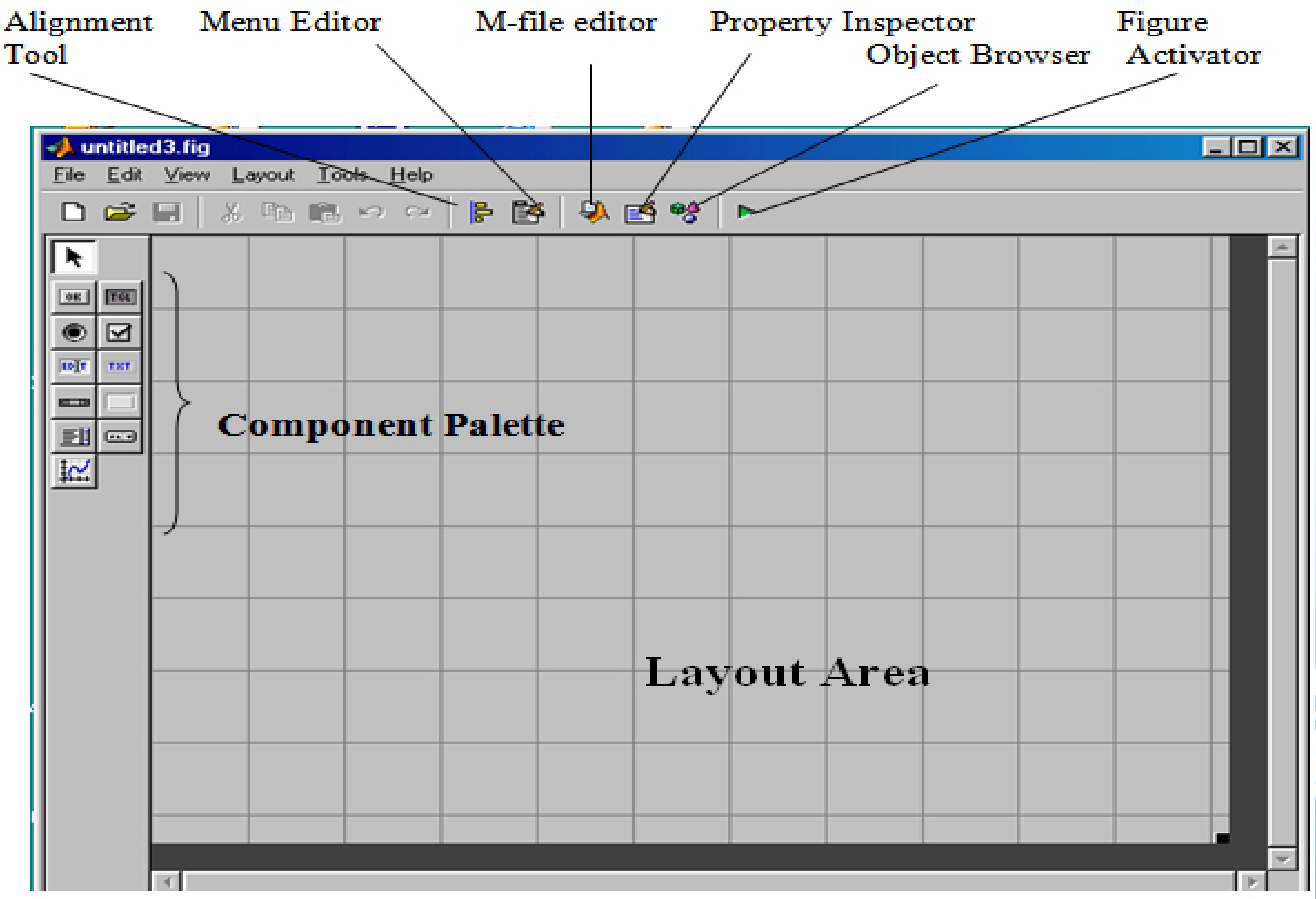
To start the layout editor, run guide command on command prompt of MATLAB; it displays an empty layout. To load and exit GUI for editing, type guide *mygui.fig or* use open option from the file menu on the layout editor or select the Open Existing GUI tab.



Cont..

In GUIDE quick start window displays two tags. The first tag is Create New GUI; it contains four options such as Blank GUI (default), GUI with Uicontrols, GUI with axes and Menu and Modal question dialog. The second tag is Open Existing GUI, which displays the list of all existing GUI. Select Blank GUI (default) and press ok button to display GUI layout editor as shown in the following figure.

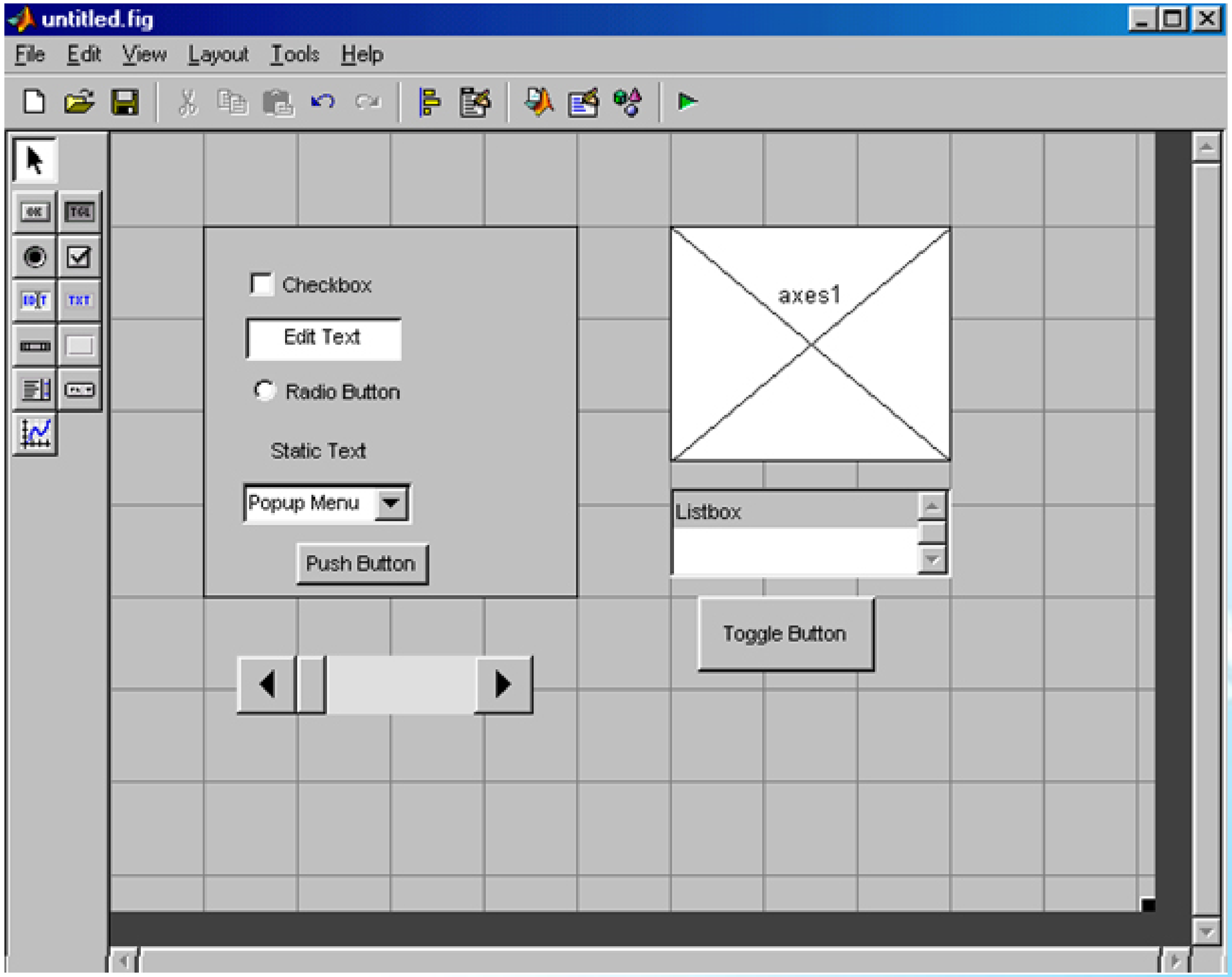
Cont..


Alignment Tool
Menu Editor
M-file editor
Property Inspector
Object Browser
Figure Activator
untitled3.fig
File Edit View Layout Tools Help
Component Palette
Layout Area

Cont..

GUI layout can save as FIG files i.e. a binary file that saves the component of the figure-using save or save as option of the file menu. GUIDE creates the application M-file automatically when we save or activate the figure. To display GUI figure use open or load command. These command load fig file into the workspace. The layout editor enables to select GUI components from a palette and arrange them in a figure window. The component palette contains the GUI components (Uicontrols Objects) that are used for user interface. The layout area becomes the figure window upon activation. From component palette, required components can be selected and placed in GUI as shown in following figure.

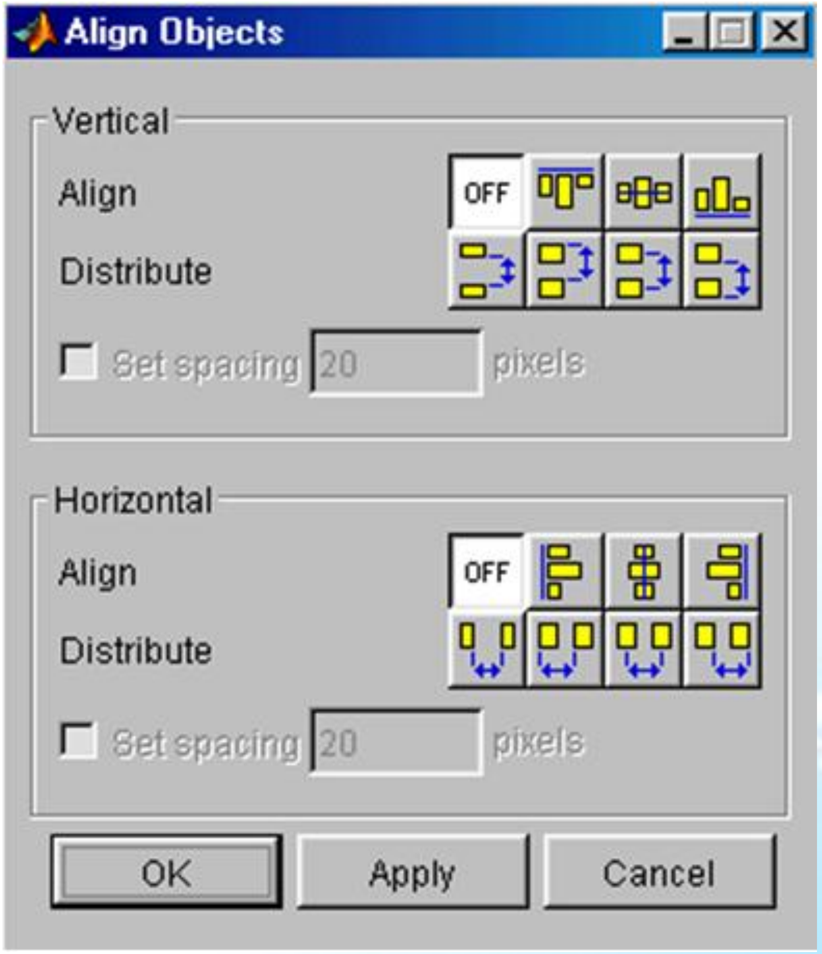
Cont..


untitled.fig
File Edit View Layout Tools Help
Checkbox
Edit Text
Radio Button
Static Text
Popup Menu
Push Button
axes1
Listbox
Toggle Button

Cont..

The layout editor provides a number of features for alignment of objects. Alignment tool aligns and distributes groups of components. Grid and Rulers align component on grid with optional snap to grid. Guide Lines: vertical and horizontal snap to Guide at arbitrary location. Bring to front, send to back, bring forward, send backward, and control the front to back arrangement of components. These alignment tools enable to position of objects and allow for adjusting the spacing between them. The specified alignment operation applies to all components that are selected when Apply button is pressed. Align and distribute components can be applied in vertical and horizontal directions. These alignment select components to a single reference line and uniformly with respect to each other respectively. Reference line is determined by the bounding box of the selected objects. The distribute option operates in two different modes: 1) equally space selected components within the bounding box. 2) Space selected components to a specified value in pixels.

Cont..



Align Objects

Vertical

Align: [✓ OFF]

Distribute

Set spacing 20 pixels

Horizontal

Align: [✓ OFF]

Distribute

Set spacing 20 pixels

Cont..

These options measure space with respect to vertical i.e. inner, top, center and bottom and horizontal i.e. inner, left, center and right edges. The layout area contains grids and rulers, which are used for component layout. Grid lines are spaced at fifty pixels intervals by default and also provide choice to select other space values ranging from 10 to 200 pixels.

Snap-to-grid space: any object, which is moved or resized within 9 pixels of a grid line to jump to another line. In this technique grids are either invisible or visible. Grid and Ruler dialog box select layout menu through control visibility of rulers, grid and guidelines, then set the grid spacing and make enable or disable snap to grid. Grid lines are essential to establish reference in component alignment at an arbitrary in the layout editor. To create a grid line click on the top or left ruler and drag the line into the layout area.

Cont..

About Property Inspector:

The property inspector enables to set the properties of the components in the layout. It also provides a list of all properties with current values. Each property in the list is associated with editing devices that are appropriate for the values accepted by the particular property. For example a text field to specify the callback string. Property inspector can be displayed by double clicking on a component in the layout editor or select property inspector from tool menu or select inspector properties from edit menu or right click on component and select inspector properties from the context menu.

Cont..

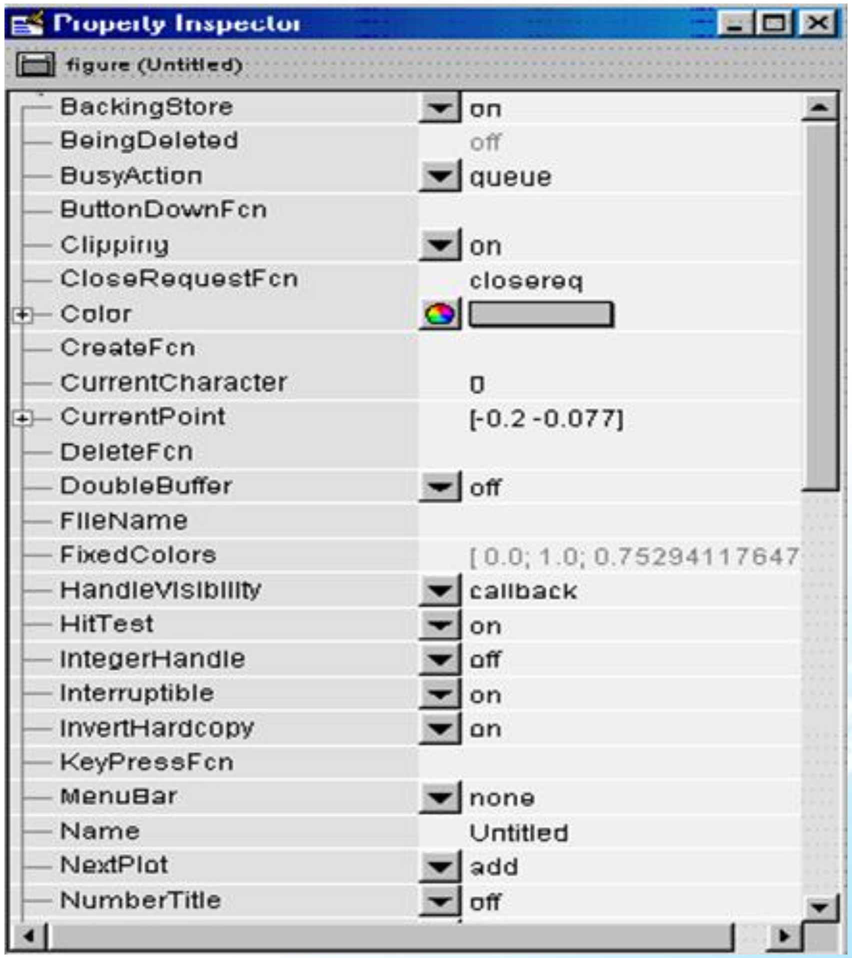

Property Inspector

figure (Untitled)

BackingStore	on
BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
Clipping	on
CloseRequestFcn	closereq
Color	▭
CreateFcn	
CurrentCharacter	0
CurrentPoint	[-0.2 -0.077]
DeleteFcn	
DoubleBuffer	off
FileName	
FixedColors	[0.0; 1.0; 0.75294117647
HandleVisibility	callback
HitTest	on
IntegerHandle	off
Interruptible	on
InvertHardcopy	on
KeyPressFcn	
MenuBar	none
Name	Untitled
NextPlot	add
NumberTitle	off

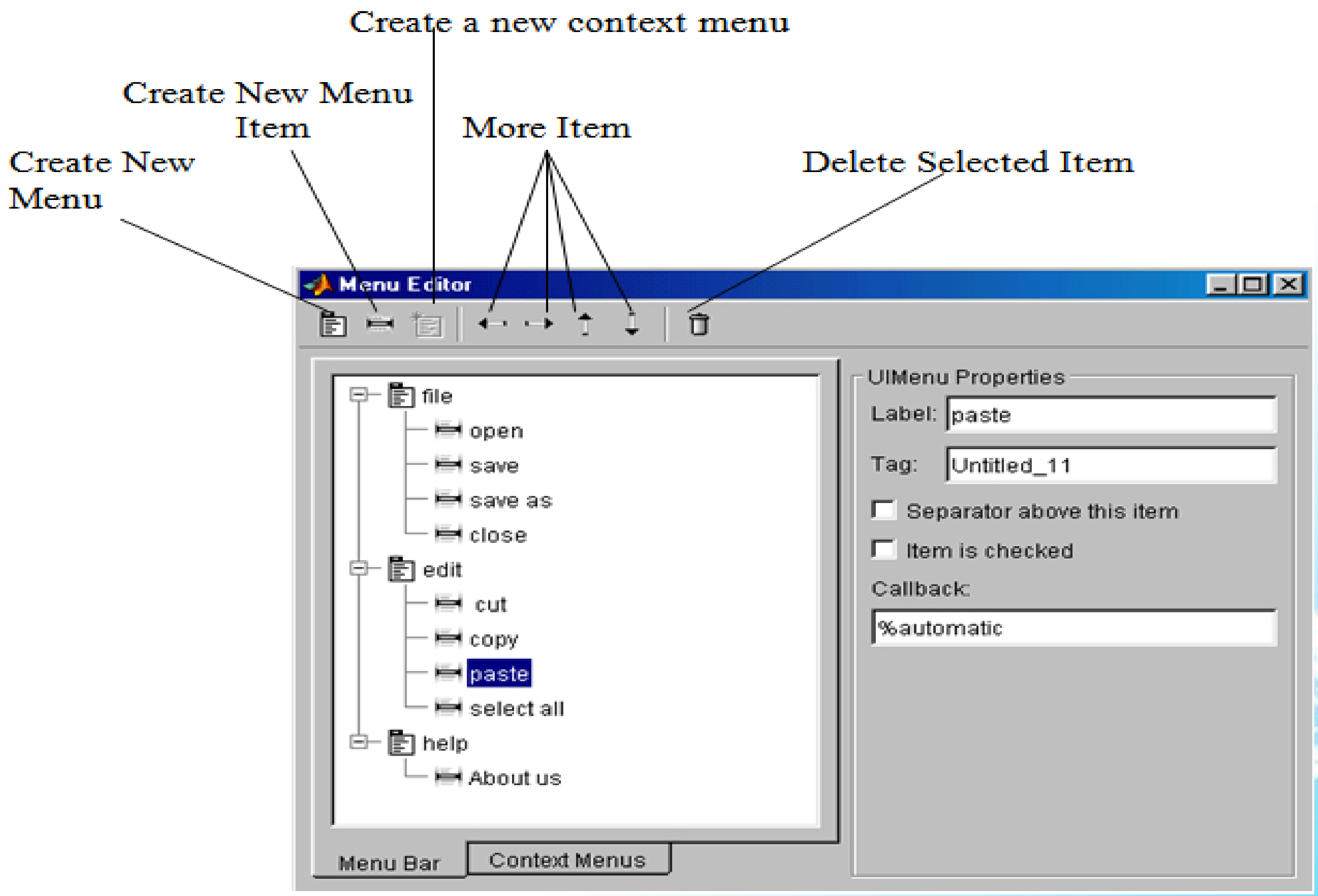
Cont..

The Menu Editor:

Menu editor enables to create two kinds of menus such as Menu bars objects and context menus. Menu bar objects displays on menu bar figure and context menu popup when the user presses the right button on graphics objects. These menus can access from the editor menu bar of layout menu and also from the layout editor tool bar.

Created menus are added to the figure menu bar. We can then create menu items for that menu. Each item can have their submenu items and these items can have submenus and so on.

Cont..



UIMenu Properties

Label: paste

Tag: Untitled_11

☐ Separator above this item

☐ Item is checked

Callback:

%automatic

Dialog Box

Dialog box to confirm an operation:

To protect the accidental event performed by users a GUI application allows to display a confirmation dialog box. This technique provides a chance to the user to confirm about the event by the accidental action taken. To implement this facility GUI provides the following steps:

1. Launching the dialog box: that asks for confirmation of the impending operation. This callback must wait for a dialog to return a value.
2. Wait for user input: To make the dialog wait for user input, selected function does not return until application window is dismissed in the GUIDE application option dialog. This option adds a call to uiwait in the dialog application M-file.
3. From property inspector select context menu to set the figures window style property to modal.

List Box

List Box Directory Reader:

It is used to display a file in a directory in a list box. If the user double clicks on list item then the following events happen:

1. If the item is file, the GUI opens the file appropriately for the file type.
2. If the item is directory, the GUI reads the contents of that directory into the list box.
3. If the item is a single dot (.), the GUI updates the display of the current directory.
4. If the item is a double dot (..), the GUI changes to the directory up one level and populates the list box with the contents of that directory.

Cont..

To implement the GUI the basic three steps are necessary:

1. Specifying the directory to list, it shows how to pass a directory path as input argument when the GUI is launched.
2. Loading list box describes the subfunction that loads the contents of the directory into the list box. This subfunction also saves information about the contents of the directory in the handles structure.
3. The list box callback explains how the list box is programmed to respond to user double clicks on items in the list box.

Accessing Variables from workspace

Accessing variables from workspace in list box:

The GUI uses a list box to display workspace variables. To do this the technique used is:

1. Populate the list box with the variable names that exist in the base workspace.
2. Display the list box with no items initially selected.
3. Enable multiple item selection in the list box.
4. Update the list items when the user presses a button.
5. Evaluate the plotting commands in the base workspace.

Cont..

Workspace variables can assign to the list box under string property and display their names. The subfunction used in the application M-file accomplishes this using *eval* in to execute *who* command in the base workspace. *who* command returns a cell array of strings, which can be used to populate the list box. The sub function used to do this task is as follows:

```
function update_Callback(hObject, eventdata, handles)
vars=evalin('base','who');
set(handles.listbox1,'String',vars)
```

GUI Components

This section introduces the common component of graphical user interface in MATLAB. It describes with an example, how to create and manipulate the GUI components.

- Static Text field
- Edit Box
- Push buttons
- Toggle buttons
- Checkboxes
- Radio buttons
- Popup menus
- Sliders
- Axis (call image in the axis)

Cont..

- **Static Text Fields**:

  Static text field is an important component of GUI, which displays one or more text string. Static text controls display lines of text. Its most important use is to give labels to other controls and provide information to user. The string property is used to assign the value to the text field. The string property accepts a string or a cell array of strings. It does not create the callbacks but the value displayed in the text field can be modified or updated from itself or another component's callback function by changing the text fields string property. By default the text box text is assigned in the center. But we can specify display text area by setting the horizontal alignment property. User cannot change it interactively and it does not have the callback routine.

Cont..

- **Edit box**

  Edit box is also an important factor in the MATLAB GUI Component, which obtains the input in the edit box and allows the developer to enter one or more text strings or numbers. To obtain the string a user types in an edit box, gets the String property in the callback. The structure of callback routine is like:

```
function edittext1_Callback(hObject, eventdata, handles)
user_string = get(hObject,'string');
% proceed with callback...
```

Cont..

For example, setting Max to 2, with the default value of 0 for Min, enables users to select multiple lines, If the min and max property are both set to zero1. For editing multiple lines use the **ctl+enter** for multi line edit text. The edit box will accept a single line of text and it will produce a callback when the user presses the **enter key** or **ESC** key after typing the text. When we obtain the numeric data from the Edit Text, MATLAB returns the value of the edit text String property as a character string; at that time we need to convert these characters to numbers by using *str2double* command, which converts string to double.

Cont..

- **Push buttons**:

  Push Button is a very important component in GUI building. It generates an action when clicking. When we press the button the callback routine will execute.

Cont..

- **Toggle buttons:**

  Toggle button is one kind of button, but it generates an action and indicates whether they are turned on or off. When we press the toggle button, it appears depressed i.e. set the Value property to max (default 1) and execute its routine and when we release the toggle button by clicking the mouse or pressing enter key, it sets the value property to min (default 0) and executes its routine. It means toggle button generates callback at each time. The structure of callback routine is as follows:

Cont..

```
function togglebutton1_Callback(hObject, eventdata, handles)

button_state = get(hObject,'Value');

if button_state == get(hObject,'Max')

% Toggle button is pressed

elseif button_state == get(hObject,'Min')

% Toggle button is not pressed

end
```

Cont..

- **Check boxes:**

  Check boxes are very helpful when the user provides independent choices. In that it sets a mode either checked or not checked. Check box generates an action when checked and indicates their state or mode. To generate an action use the value property by querying the state of its value, then execute callback routing.

```
 function checkbox1_Callback(hObject, eventdata, handles)
if (get(hObject,'Value') == get(hObject,'Max'))
% then checkbox is checked-take appropriate action
else
% checkbox is not checked-take appropriate action
end
```

Cont..

- **Radio buttons**

  It is also similar to check boxes, which is very useful to determine the current state of radio button from within its callback. It means you can select only one button at a time within a group of related radio buttons. By using the value property we can determine the current state of a radio button. The structure of callback routine as follows:

```
if (get(hObject,'Value') == get(hObject,'Max'))
% then radio button is selected to take appropriate action
else
% radio button is not selected-take appropriate action
end
```

Cont..

- **Popup menus:**

  Pop-up menu displays a list of options to select one option among them. Through pop-up menu callback, we can access the value, which consists of index of the item selected using the value property or we can obtain the actual string contained in the selected item. For selecting the indexed item it uses a switch statement to take action based on the value. If the contents of the popup menu are fixed then you can use this approach. The structure of callback routine is as follows:

Cont..

```
function popupmenu1_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
switch val
case 1
% The user selected the first item
case 2
% The user selected the second item
% proceed with callback...
```

Cont..

- **Sliders:**

  Slider accepts numeric input within a specific range by enabling the user to more sliding bar, and we can determine the current value of a slider from within its callback by using its value property. The location of slider indicates the percentage of the specified range. We use the mouse to change the position of slider.

Cont..

- **Axis** (call image in the axis)

  Axis allows us to display graphics i.e. graphs and images. We can set or get the axis graphics using its properties and axis is not uicontrol object. Like all graphics objects, axes have properties that you can set to control many aspects of its behavior and appearance like color, hold axes etc. For more details see the Axis Properties.

Cont..

- **Panels:**

  Panels are used for group GUI Components. Panels give us a good look for easier understanding of related controls or components and it has title and borders. If we move the panel automatically its children (components within the panel) move with it without changing its position on the panel.

Cont..

- **List box:**

  List box is used to display the list of items and enable us to select one or more items. MATLAB execute callback of list box when the mouse button is released or after certain key press events. The arrow keys change the value property and trigger callback execution. Enter and Space does not change the value property but triggers callback execution.

Cont..

- **Button Group:**

  Button groups are similar to panel but it can be used to manage exclusive selection behavior for radio or toggle button. User should enter the code to control these radio and toggle buttons that are managed by button group. The button group overwrites the callback properties of radio buttons and toggle buttons that it manages using selection property.

Cont..

- **ActiveX control:**

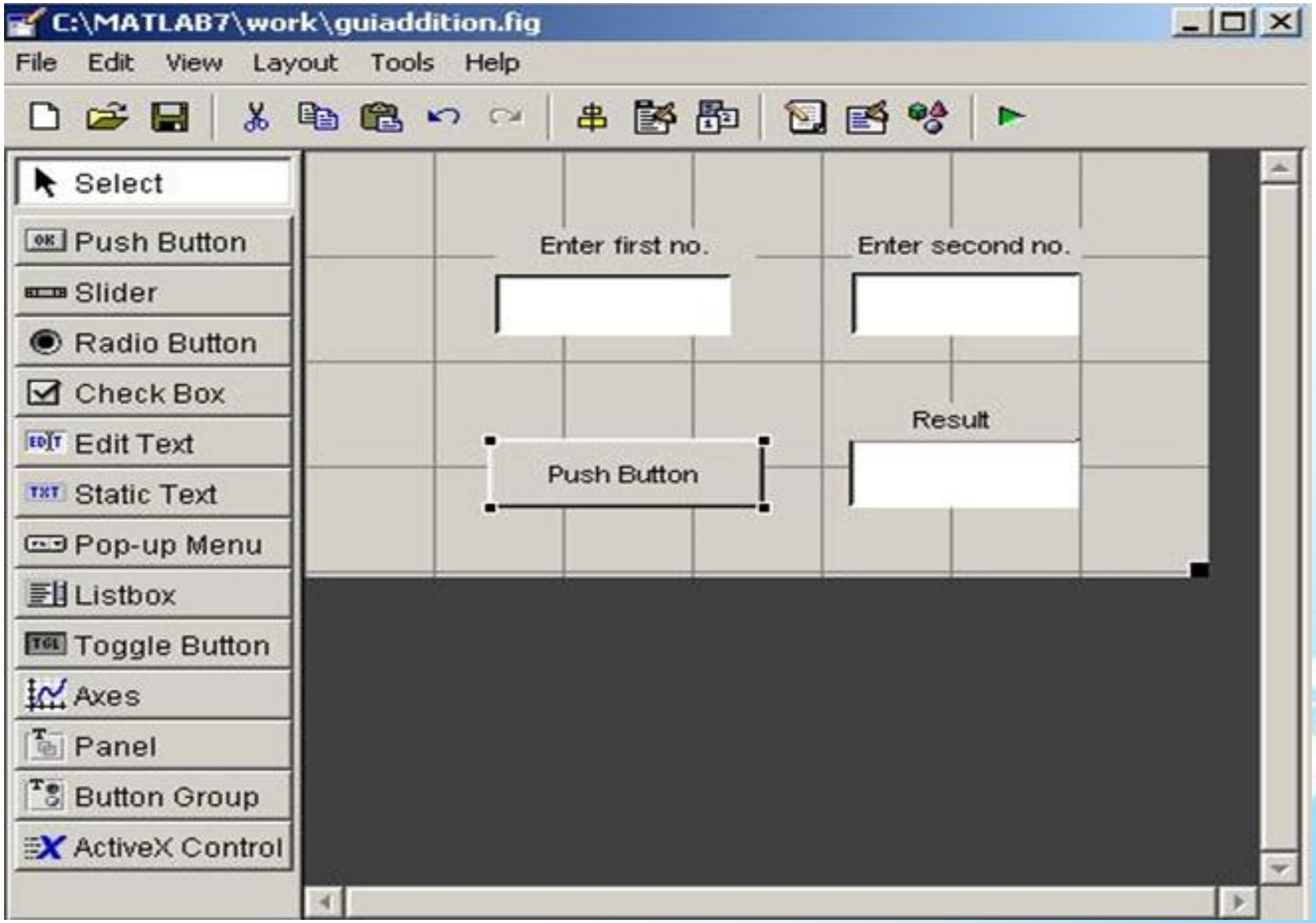
  MATLAB's ActiveX component allows us to add, view properties, and set properties etc. of ActiveX controls in our GUI. You can add an ActiveX control into your GUI if you are running MATLAB on Microsoft Windows.

Solved GUI Examples

Example 1: Build GUI for addition of two numbers

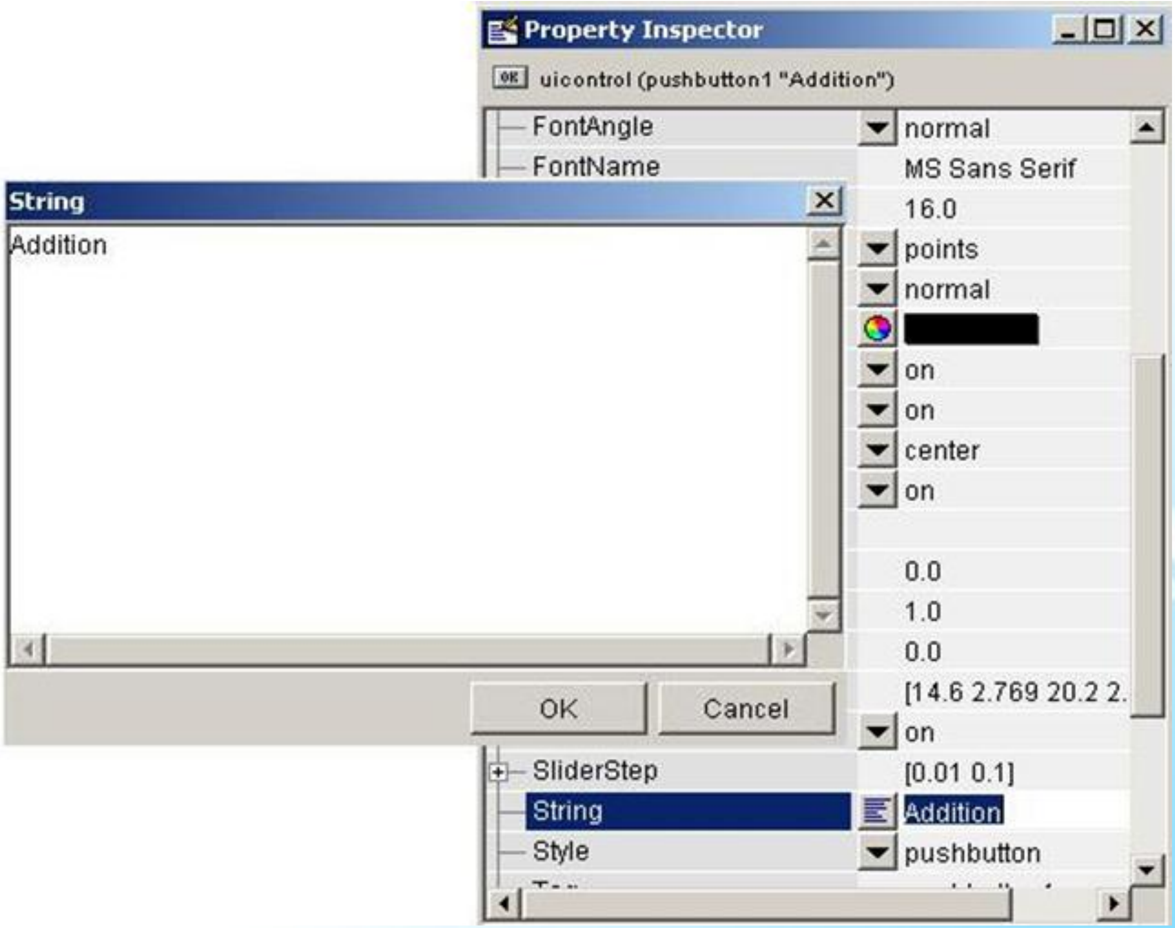
1. Design the Layout to add two numbers. In this example three components are used i.e. edit text, static text and push button and set the properties as described above.

Cont..


C:\MATLAB7\work\guiaddition.fig
File Edit View Layout Tools Help
Select
Push Button
Slider
Radio Button
Check Box
Edit Text
Static Text
Pop-up Menu
Listbox
Toggle Button
Axes
Panel
Button Group
ActiveX Control
Enter first no.
Enter second no.
Result
Push Button

Cont..

2. Set the properties by using the Property inspector like string, font size, font type etc of every GUI Component.



uicontrol (pushbutton1 "Addition")

FontAngle	normal
FontName	MS Sans Serif
	16.0
	points
	normal
	■
	on
	on
	center
	on
	0.0
	1.0
	0.0
	[14.6 2.769 20.2 2.
	on
SliderStep	[0.01 0.1]
String	Addition
Style	pushbutton

Addition

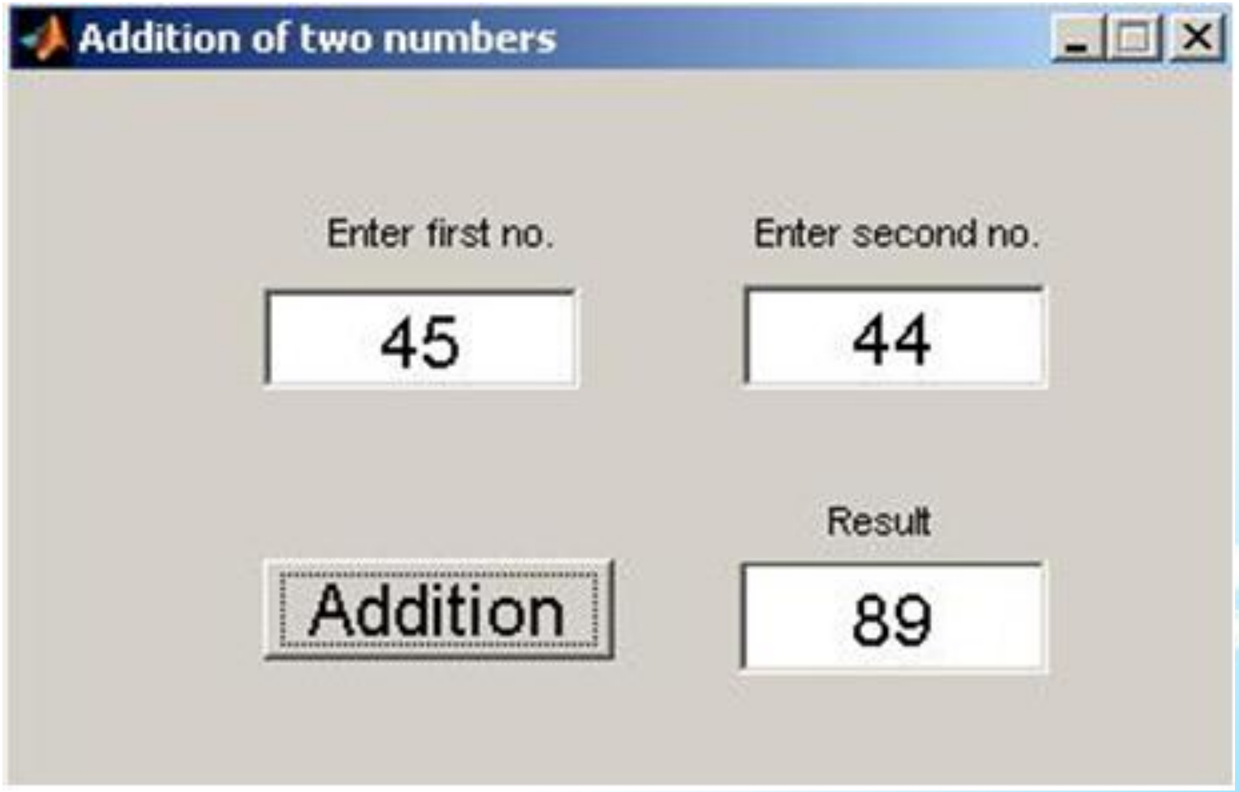
Cont..

3. Write the following code in the Pushbutton callback

```
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)
% set(handles.edit1,'String','')
% set(handles.edit2,'String','')
a=str2num(get(handles.edit1,'String'));
b=str2num(get(handles.edit2,'String'));
c=a+b;
d=num2str(c)
set(handles.edit3,'String',d)
```

Cont..

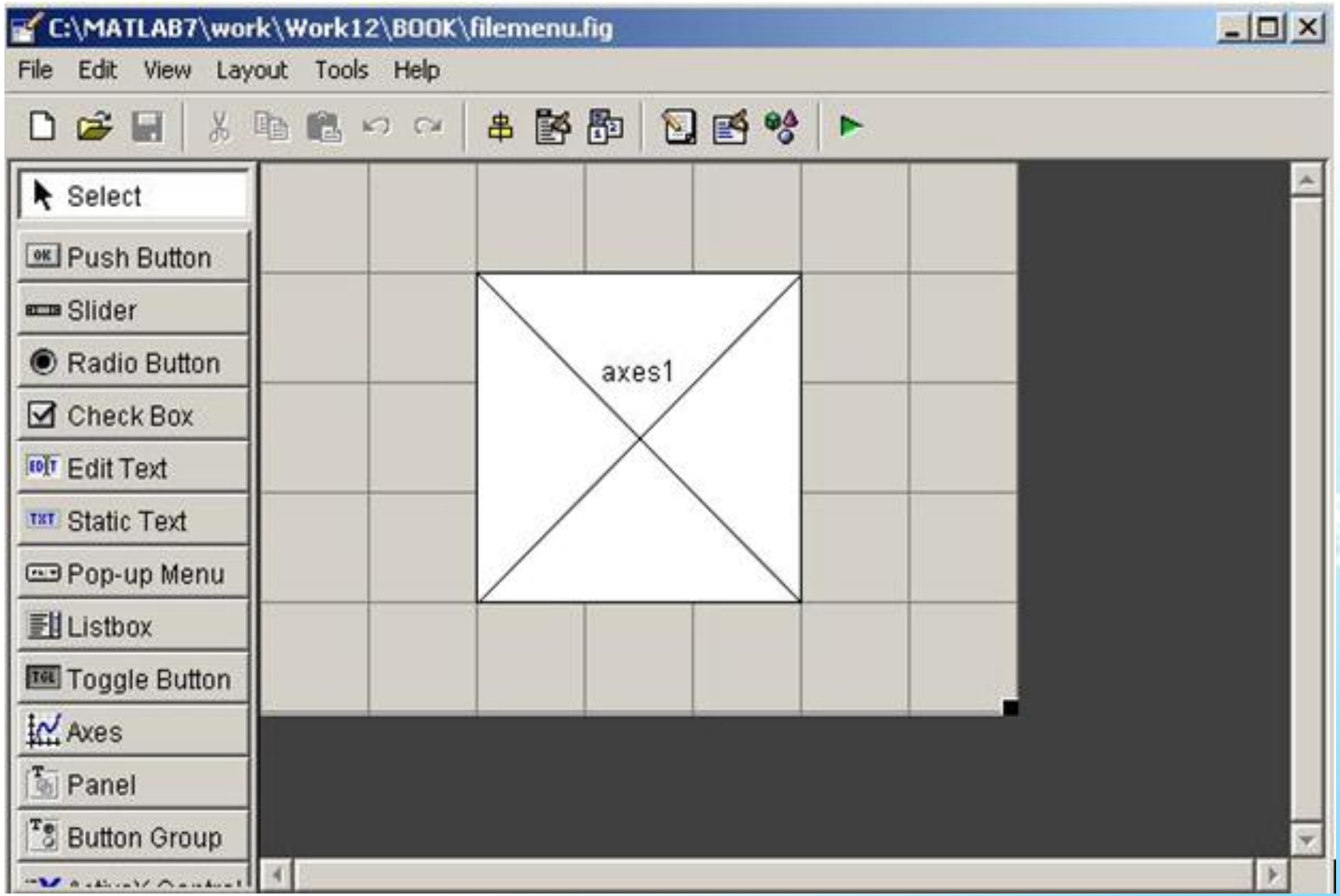
4. Run the created GUI and enter the parameters or values in edited text box and finally press the addition button, the output is shown as follows:



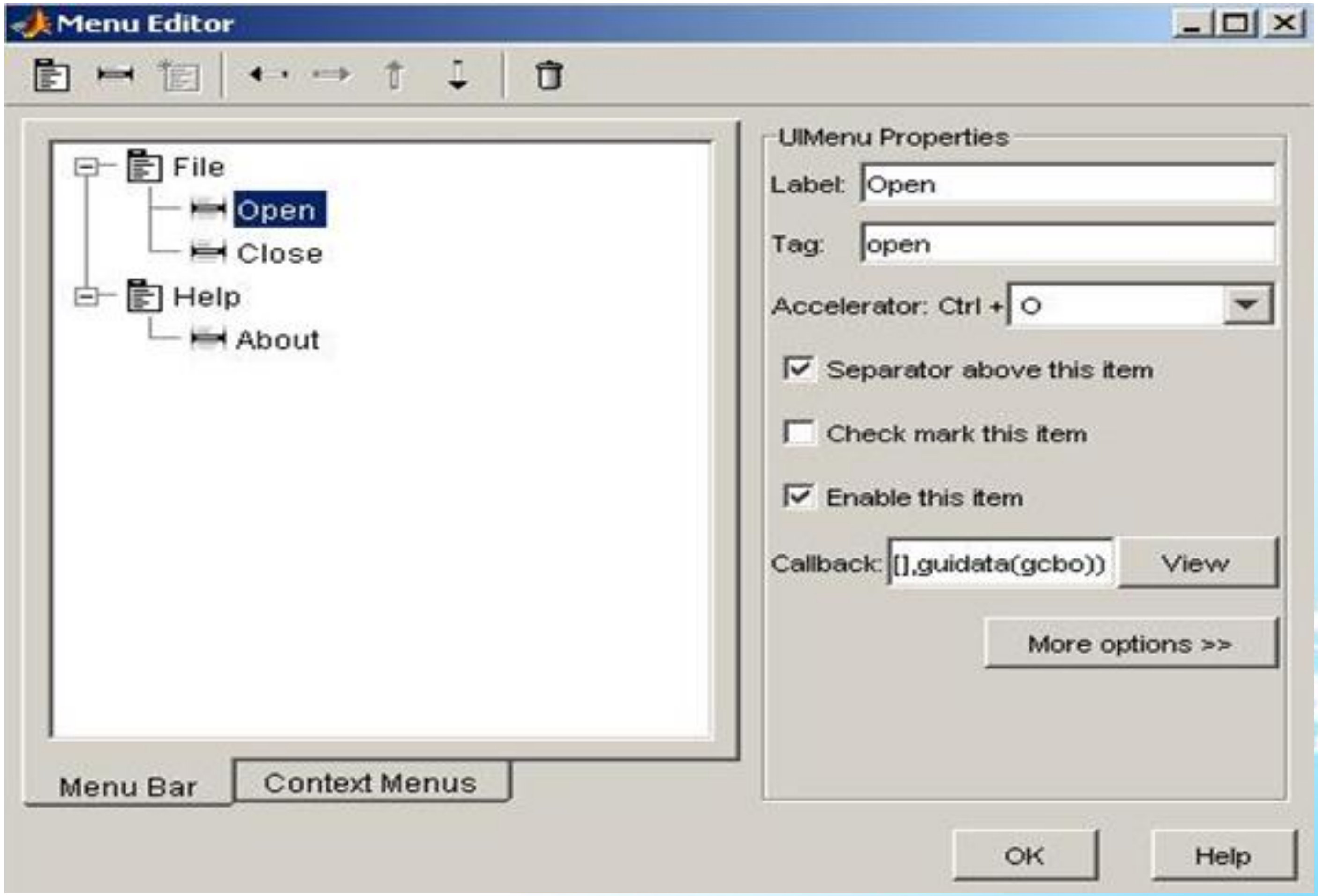
Cont..

Example 2:

Example of Menu editor: Build A GUI using menu editor to open an image file in axes component.



Cont..



UIMenu Properties

Label: Open

Tag: open

Accelerator: Ctrl + O

[✓] Separator above this item

[ ] Check mark this item

[✓] Enable this item

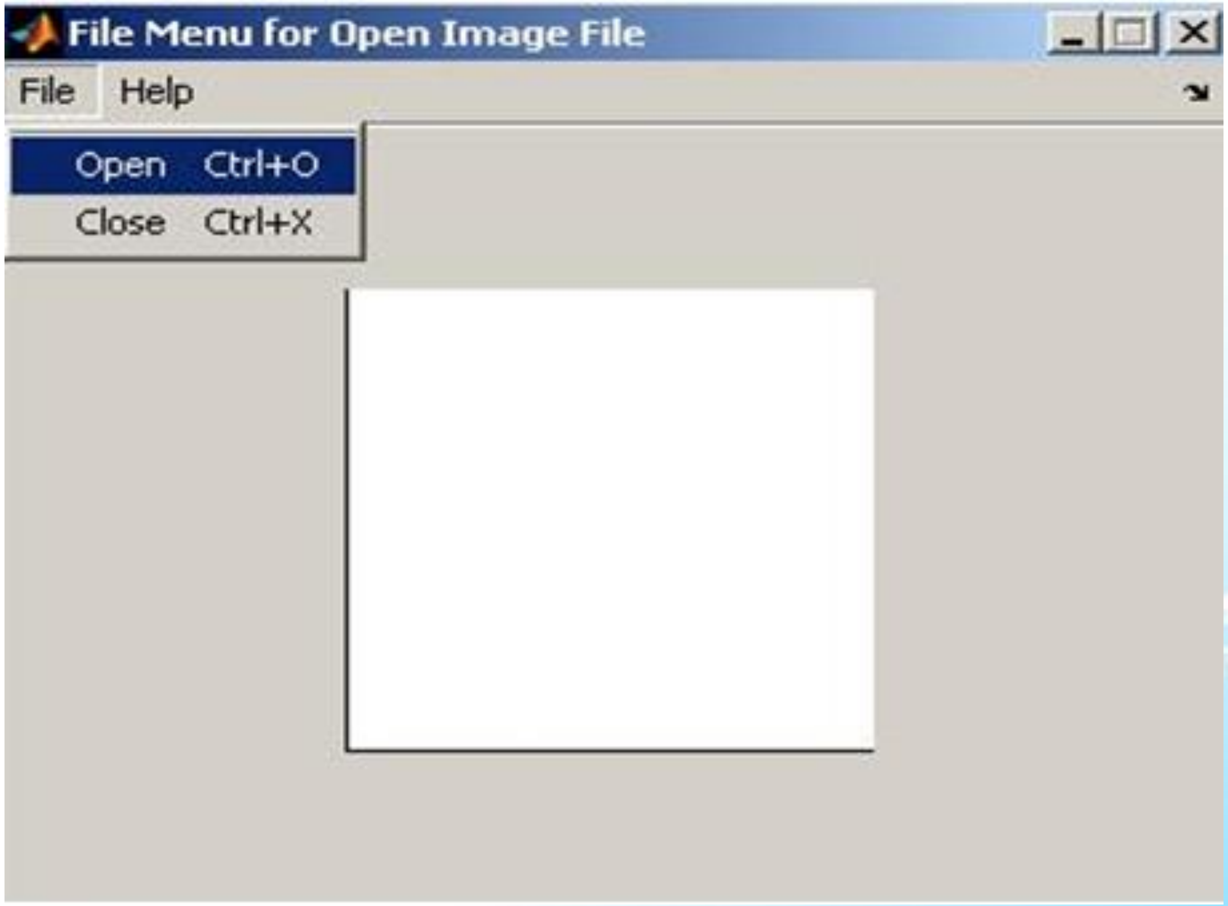
Callback: [],guidata(gcbo))

Cont..

Then execute the figure file
Callback of open

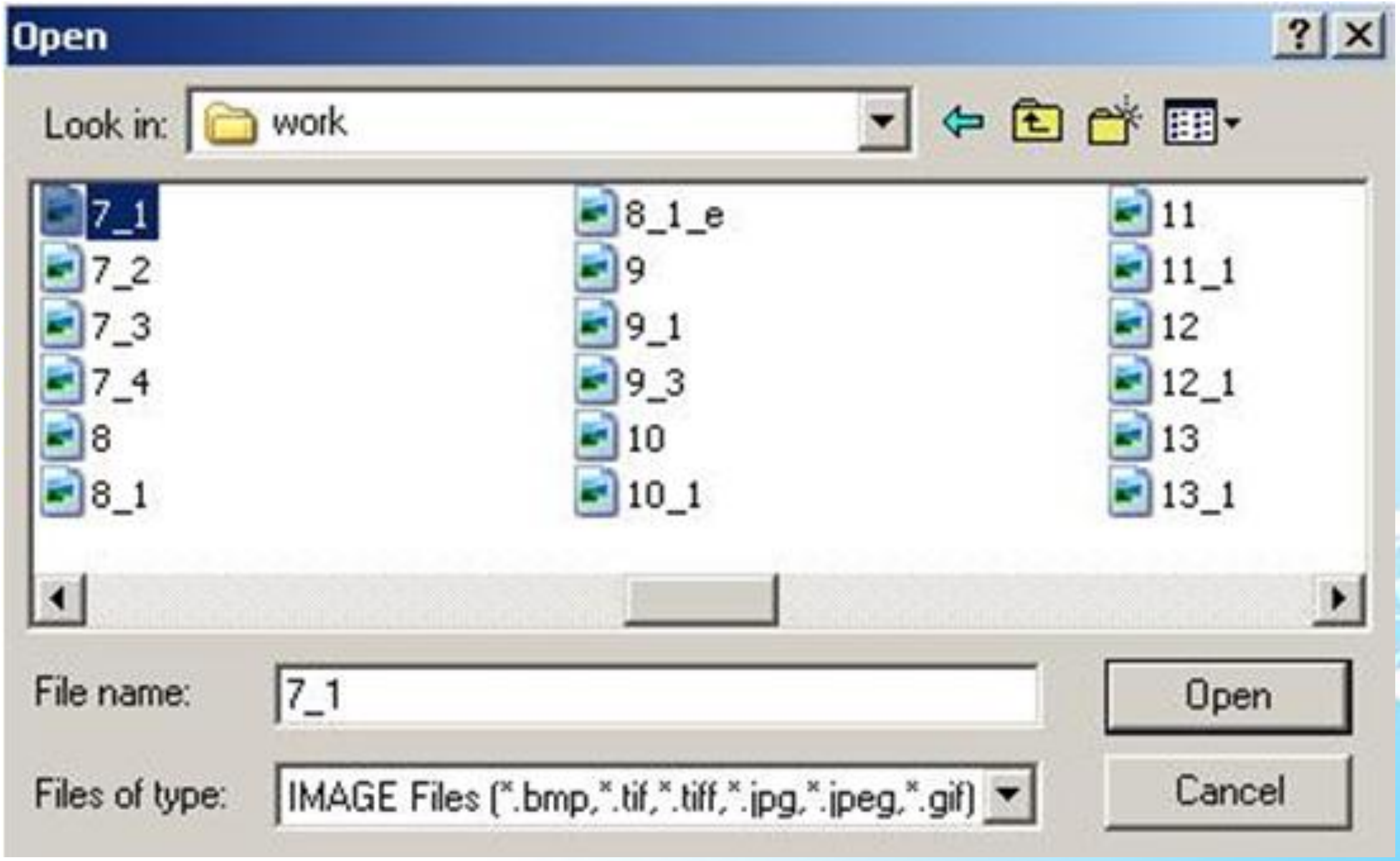
```
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;*.jpeg;*.gif',
'IMAGE Files (*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'},'Open');
i=namefile;
axes(handles.axes1)
imshow(i);
```

Cont..


File Menu for Open Image File
File
Help
Open Ctrl+O
Close Ctrl+X

Cont..

The open window will appear, select an image file which you want to show



File name: 7_1

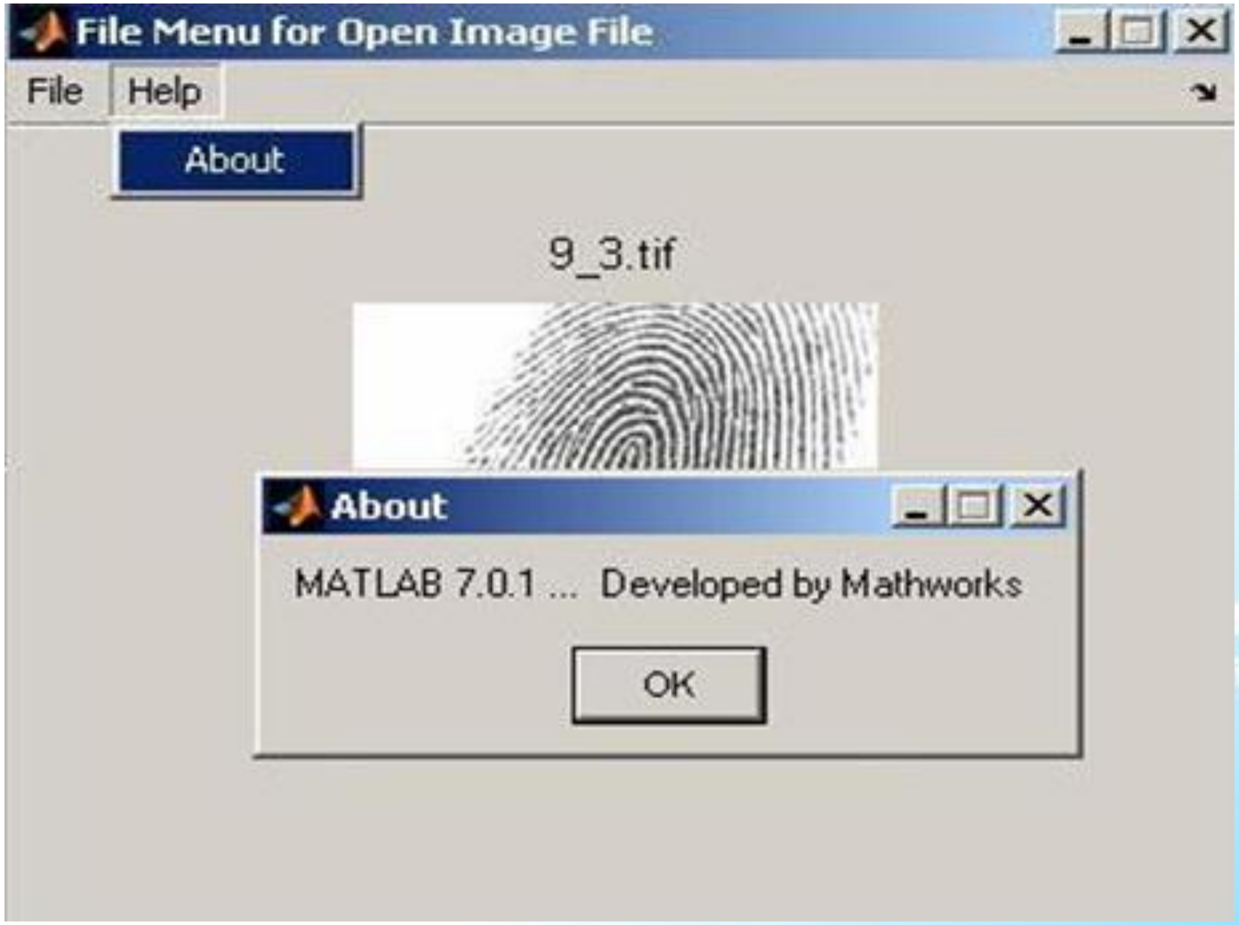
Files of type: IMAGE Files (*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)

Cont..

Callbacks for help and close are:

```
% Callback for help
function help_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
str= 'MATLAB 7.0.1 ...  Developed by Mathworks';
title = 'About';
msgbox(str,title);

%callback for close
function close_Callback(hObject, eventdata, handles)
% hObject    handle to close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close((gcbf));
```

Cont..

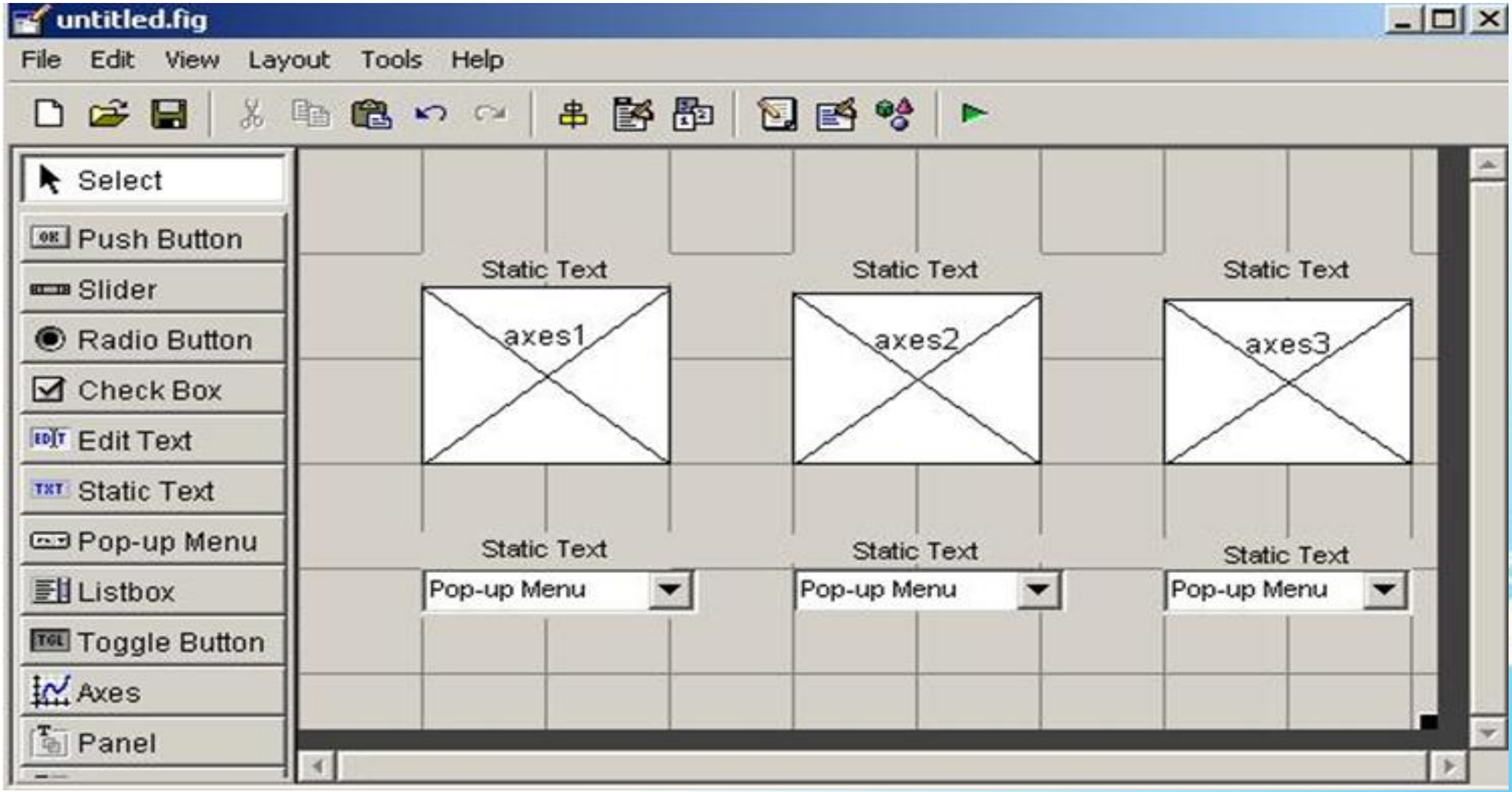


MATLAB 7.0.1 ... Developed by Mathworks

Cont..

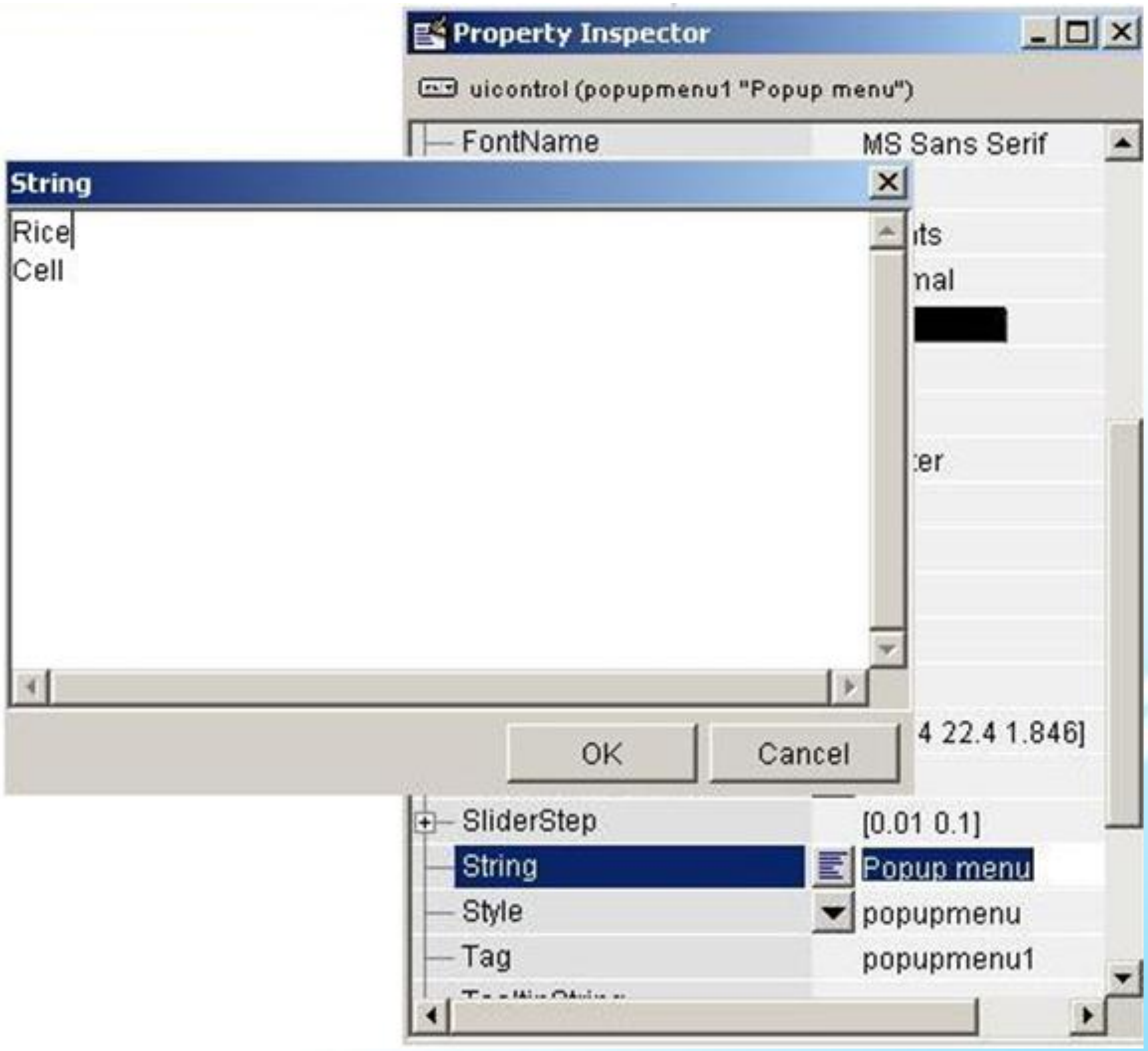
Example 3: Example of Image filtering demo using multiple axes and multiple popup menu.

1. Layout of the above example is



Cont..

2. Set the properties of each component by using Property Inspector



Property Inspector

uicontrol (popupmenu1 "Popup menu")

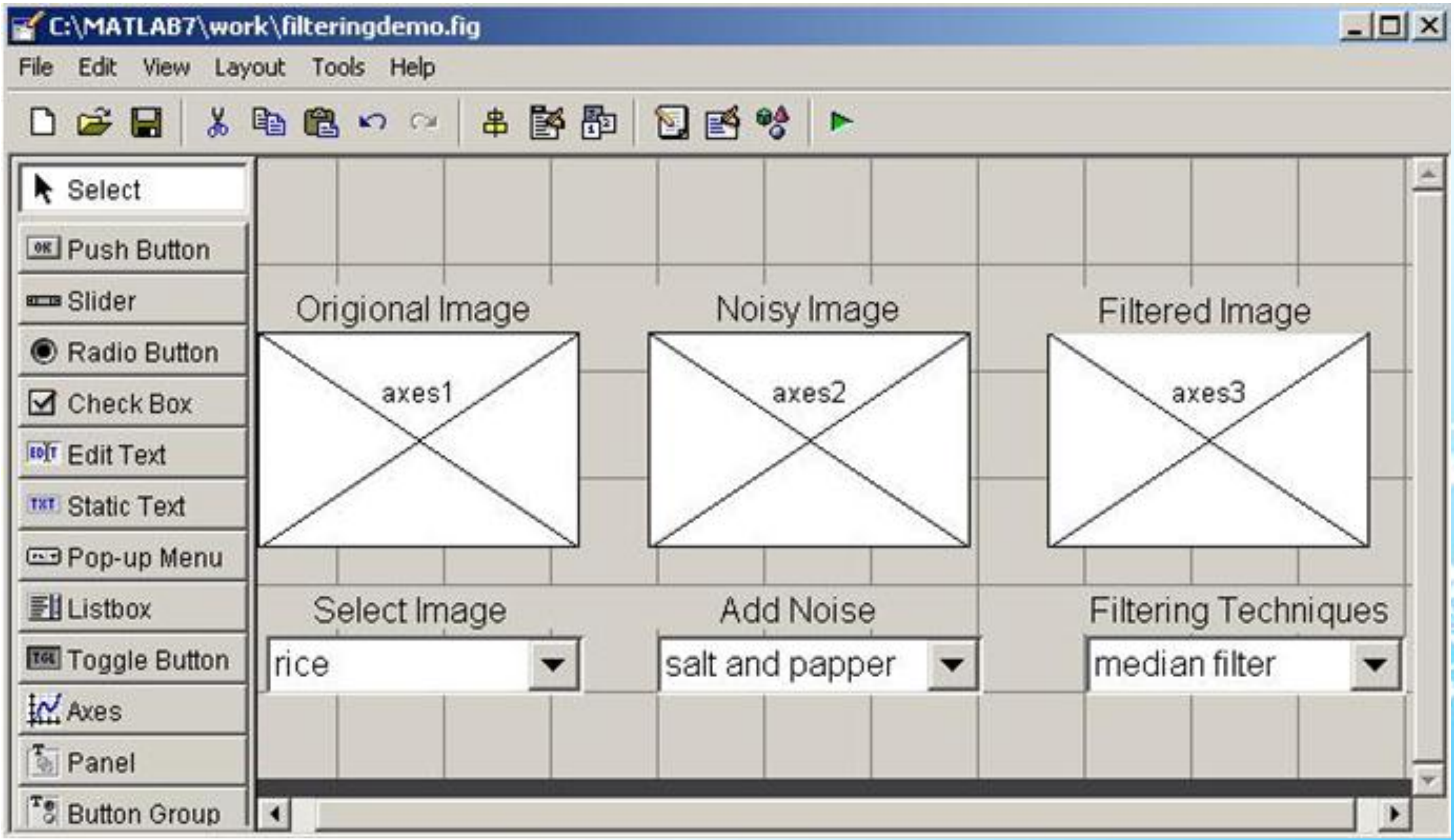
FontName	MS Sans Serif
	…4 22.4 1.846]
SliderStep	[0.01 0.1]
String	Popup menu
Style	popupmenu
Tag	popupmenu1

String

```
Rice
Cell
```

Cont..

3. Finally the layout appears as follows:



Cont..

4. The callback for above all component is as follows:

%Code for first popupmenu

```
 function popupmenu1_Callback(hObject, eventdata, handles)
 val1 = get(hObject,'Value');
switch (val1)
   case 1

       i=imread('rice.png');
       axes(handles.axes1);
       imshow(i)
       handles.i=i;
    case 2

       axes(handles.axes1);
       j=imread('cell.tif');
       imshow(j)
end
```

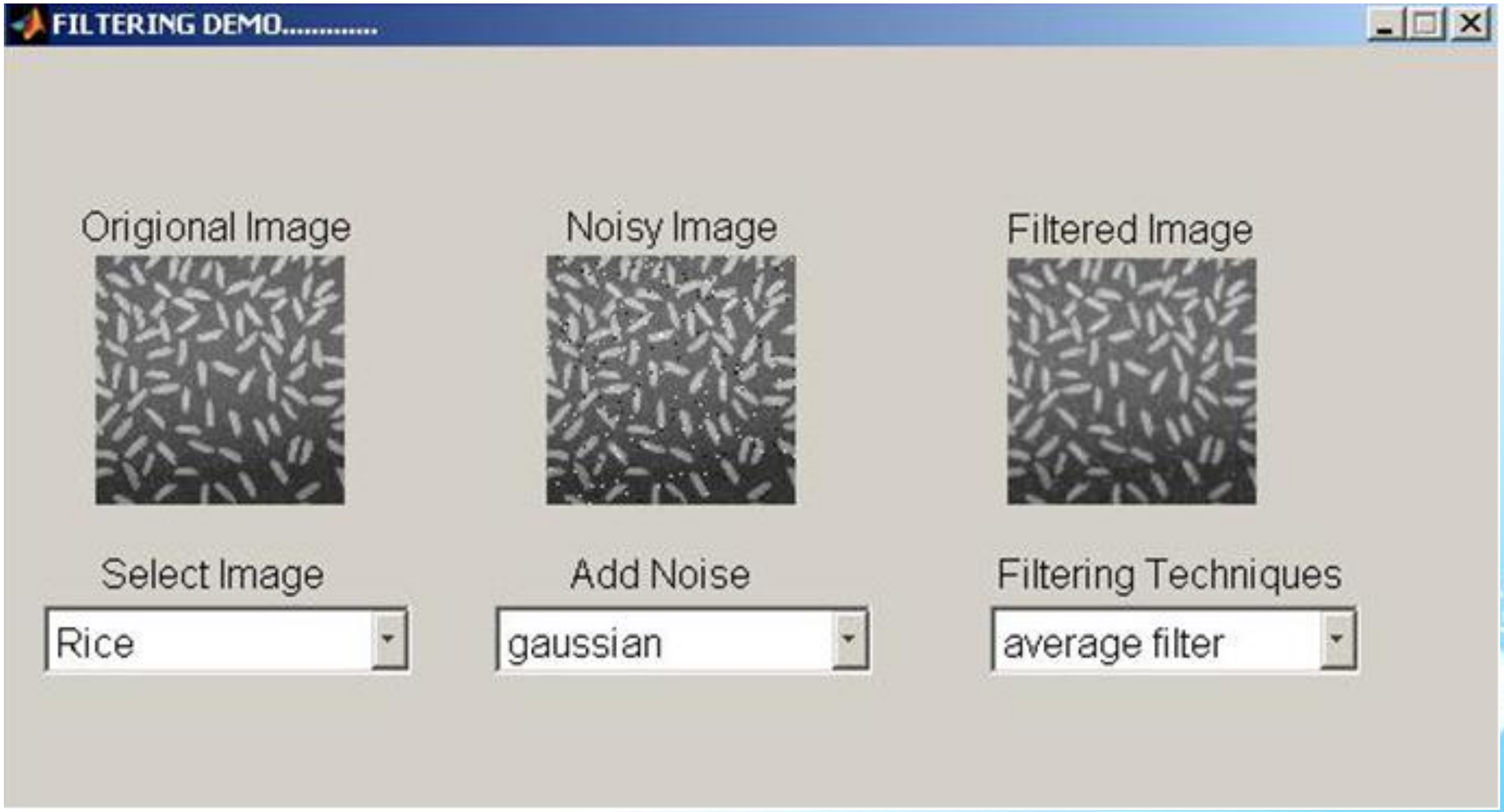
Cont..

```
%Code for first popupmenu
function popupmenu2_Callback(hObject, eventdata, handles)
val2 = get(hObject,'Value');
switch val2
   case 1
        i=getimage(handles.axes1);
       J = imnoise(handles.i,'gaussian');
       axes(handles.axes2);
       imshow(J);
        axes(handles.axes2);

   case 2
       i=getimage(handles.axes1);
       J = imnoise(i,'salt & pepper',0.02);
       axes(handles.axes2);
       imshow(J);
        axes(handles.axes4);
 end
```

Cont..

```
%Code for popupmenu3
function popupmenu3_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
switch val
    case 1
   i=getimage(handles.axes2);
   r=medfilt2(i);
   axes(handles.axes3);
   imshow(i)

case 2
   J=getimage(handles.axes2);
   K = filter2(fspecial('average',3),J)/255;
   axes(handles.axes3);
   imshow(K);
 end
```

Cont..


FILTERING DEMO.............

Origional Image

Noisy Image

Filtered Image

Select Image: Rice

Add Noise: gaussian

Filtering Techniques: average filter

Summary

This chapter has explained details about GUI layout tools like Layout editor, Alignment tools, Property Inspector, Object browser, Menu Editor, Dialog box, List Box, Accessing variables from workspace to creation and handling of users applications more user friendly and allow to have more easy interaction of user with system. MATLAB has a number of GUI Components such as Static text fields, Edit box, Push buttons, Toggle buttons, Checkboxes, Radio buttons, Popup menus, Sliders, Axis and so on. These components enhance the feature of user’s application layout. These features of MATLAB also facilitate to create application having maximum options and allow to feed more information within very few keys. At the end of this chapter the solved examples are also given which are useful for user to write their own applications easily.

Thank You