Experiment No. (13)

Introduction to Graphical User Interface (GUI)

Aim of experiment:

To introduce: what is GUI , how it can allow the user to interact with your program and principal elements required to create a MATLAB Graphical User Interface (GUI).

Introduction:

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.

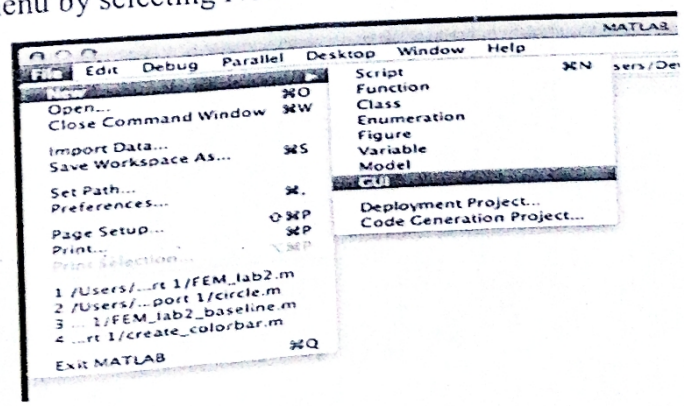
How Does a GUI Work?

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.

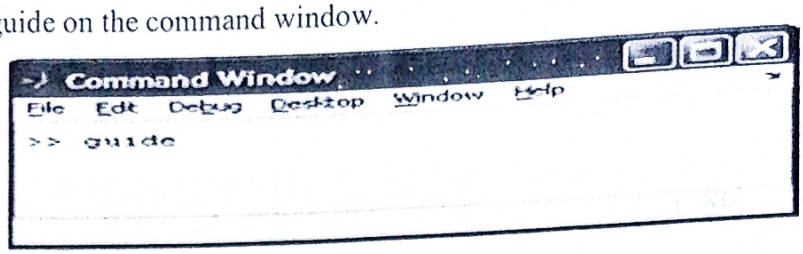
Opening Matlab GUI:

Two ways to open the GUI window

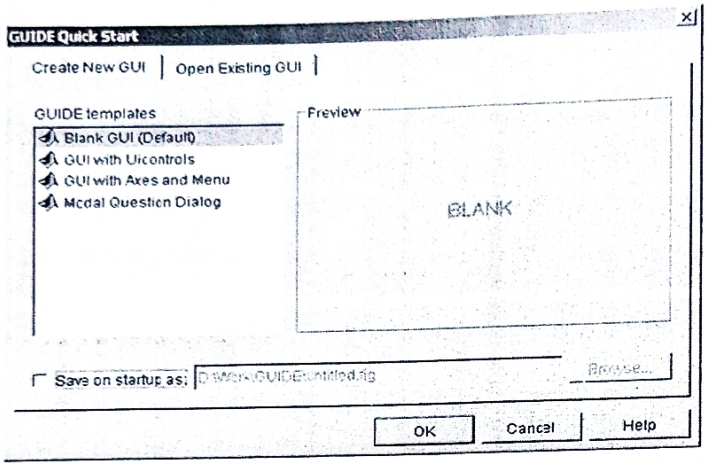
- Click **MATLAB File** menu by selecting **New > GUI**



- Type guide on the command window.

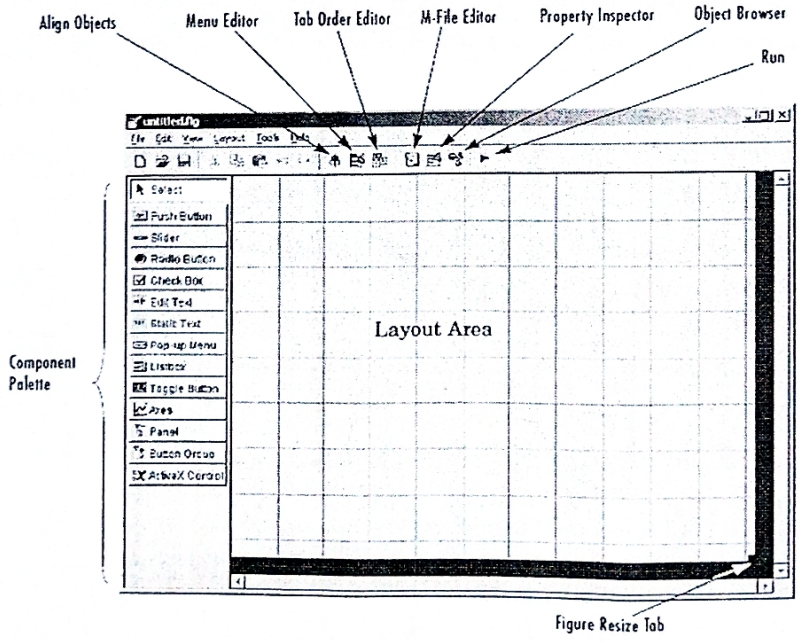


- Choose the first option Blank GUI (Default)



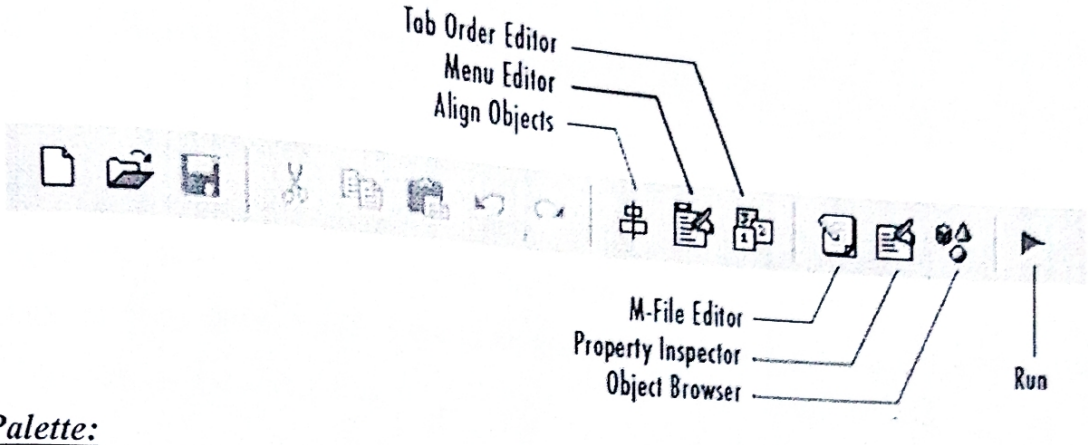
GUIDE Tools Summary:

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



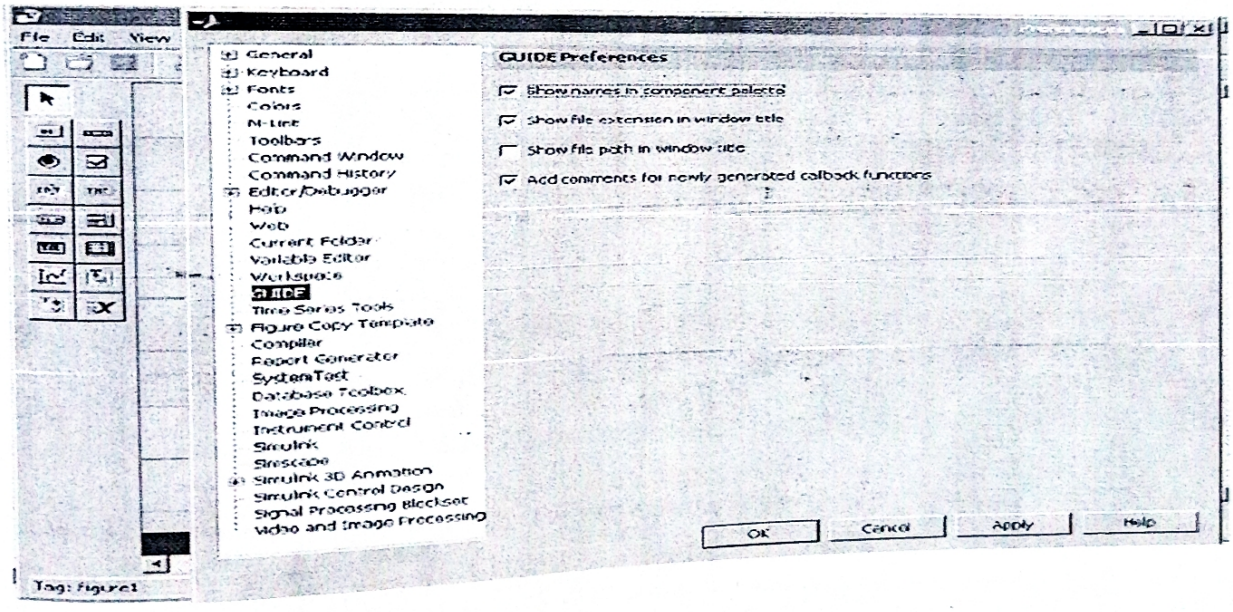
Toolbar:

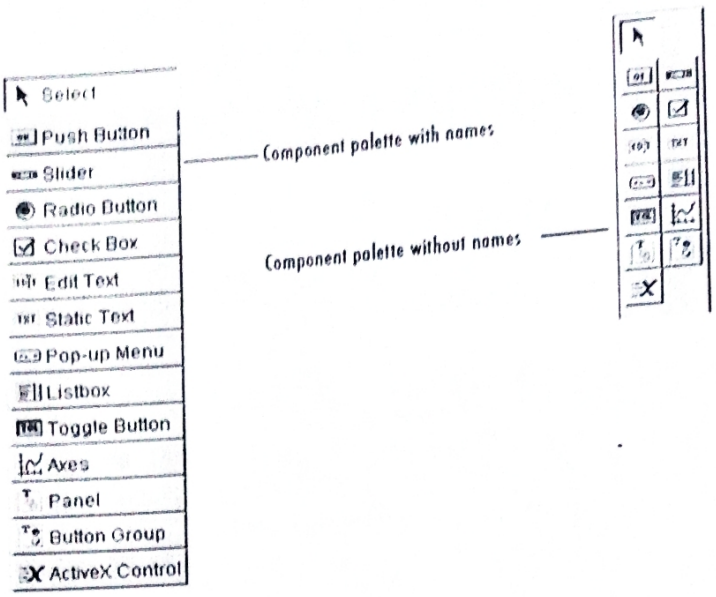
Displays the following toolbar in the Layout Editor window.



Component Palette:

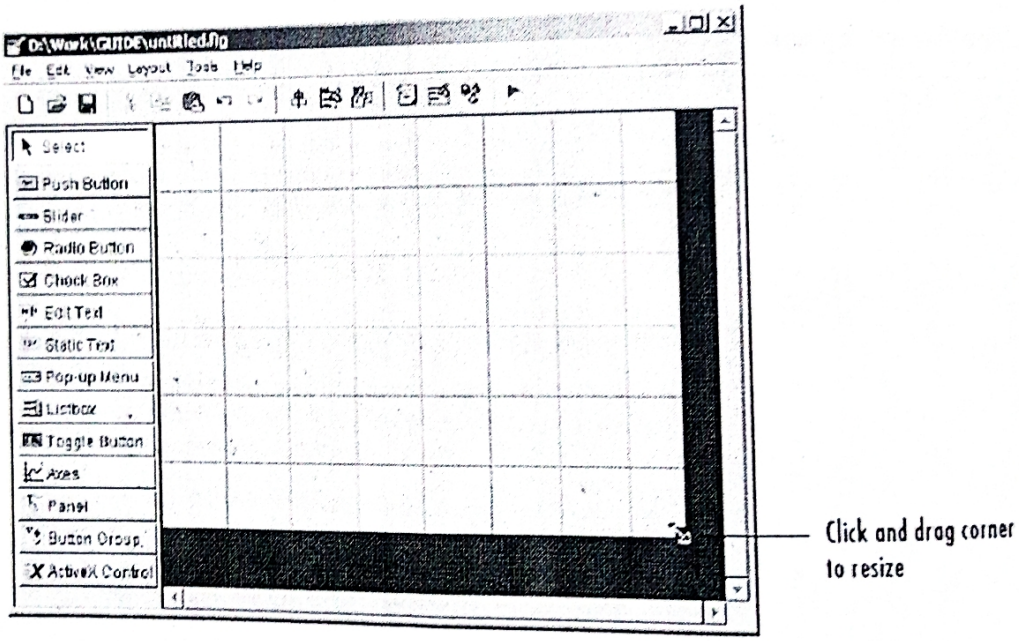
When you first open the Layout Editor, the component palette contains only icons. To display the names of the GUI components, select **Preferences** from the **File** menu, check the box next to **Show names in component palette**, and click **OK**.





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.



Exercises:

1. How many ways to build MATLAB GUIs?
2. State the steps that be used to show names component palette?
3. How we can set the GUI size?

Experiment No. (14)

Design Graphical User Interface (GUI) Part (II)

Aim of experiment:

To introduce the basics elements that require for designing GUI .

Introduction

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.

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.

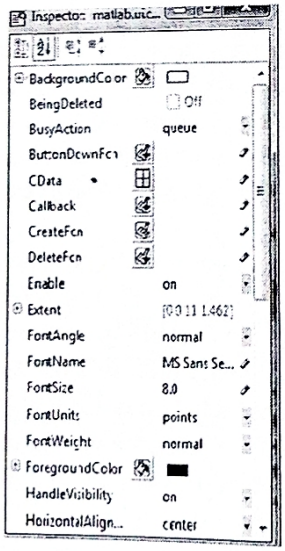
The following table describes each of these components:

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 togglebuttons.
Radio Button		Radio buttons are similar to check boxes, but radio buttons are typically mutually exclusive within a ngroup 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.
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.

Property inspector

The Property Inspector (built into guide) to give each component a name (a "tag") and to set the characteristics of each component, such as its color, the text it displays, and so on. Double click on any components. You should see the following table appear which is called *Property Inspector*



Having located the control-elements of your GUI in the layout panel, you can now customize the properties, which includes the appearance of these item.

- Each element in the layout panel is treated as an object that is contained in the figure which defines the GUI.
- Following from that, each object will possess a defined 'and limited' set of properties.
- Some of these properties determine the way that object looks, such as its
- color, its title or the fonts used for text output associated with that object.
- There are also properties that determine the action that needs to be performed when a particular interaction is applied to an object,such as mouse-click on a button, slider-dragging with the mouse so on .

Saving your GUI

- Once you are happy with the layout of your GUI, you should save it for future use.
- Save the GUI using the <file> <save> menu or by clicking the save icon.
- Graphical User Interfaces generated by GUIDE are saved into two closely linked files-namely;
- your_gui_name.fig and your_gui_name.m
- The figure (.fig) file contains all the information related to the layout and appearance.
- The script (.m) file contains all the programming logic aspects of the GUI.
- Most of the programming components of interest in the .m file will be contained in the set of function-stubs ready for use as call-back functions to various control objects.
- The task of programming the GUI will be one of finding the suitable code that need to be inserted into these call-back functions.

Exercises:

1. State the available components that will be used in GUI design?
2. What are the functions of property inspector?
3. What is the important step we should be done before creating the actual GUI?

~~Experiment No. (15)~~

Experiement No. 15

Programming the GUI components

Aim of experiment:

The aim of this experiment is to know how we can program the GUI components ,give practical examples and identify the practical steps for designing and programming simple GUI.

Introduction

The most commonly properties used in *Property Inspector* to program the GUI components are:

- String

  Text or title to display(e.g. button titles) .

- Tag

  To identify the component during programming

- Callback

  To specify the action to take when control activated (.e.g what to do when mouse Clicked on button)

- Value

  Indicates the state of the control object numerically(e.g. position of the slider , button on/off

String Properties:

String property of the most GUI control items determine the titles to be printed on the screen to identify these controls.

- For buttons, check boxes, edit text, and static text, the String text is displayed on or next to the component. These titles will not normally be changed by the user during the execution of the gui.
- For pop-up menus and list-boxes all the items on the menu or the list-box should be entered as a column of n number of strings where; n is the number of options on the menu.
- For an edit text, the String property contains a list of strings that is displayed in the text box. When a user edits the text, the String property is updated.

Tags:

Tags are used for identifying the GUI components during programming the interface. Every object created on the GUI panel will have a tag property associated with it. By default, Guide gives simple descriptive values to these TAG properties, such as pushbutton1, slider3 so on and uses these values when it generates the call back functions in the auto-generated .m-file associated with that GUI. For easier understanding and programming of this mfile, we recommend that you define more meaningful tag values for the control components 'buttons, sliders etc.' of your GUI such as 'rescale', 'convert' etc.

Callback:

This is the most important property of the control objects. Callback property determines what action needs to be taken when the user activates that object (for example by clicking on the object, selecting an item from the listbox or choosing a menu item etc.) Most, if not all, the programming related to your GUI will be contained in the callback functions. Callback property will normally be set to the name of a function 'as a string variable' that needs to be invoked, but it can also contain a Matlab expression enclosed in a string (for simple tasks) .

Programming your GUI:

After laying out the GUI and setting its component properties, the next step will be to program its behavior.

The code contained in the callback functions will control how the GUI responds to events such as button clicks, menu item selection, window resizing as well as creation and deletion of components.

There will normally 'but not always' be one callback function per component on the GUI interface plus a couple of extra callback functions relating to the initialization and output onto Matlab window. All these functions will be contained in a single .m file generated by GUIDE when you save your gui. (*your_gui.m*)

Callback function names

Names of the callback functions for each GUI component will also be automatically generated by GUIDE using the convention;

function <objects_tag>_<event_to_handle>

For examples;

The function that handles the listbox selection events for a listbox with tag <listbox2> will be named :

function listbox2_Callback

A function that handles a key-press over a button with tag 'redbox1' will be named

function redbox1_KeyPressFcn

There will also be the following automatically generated callback functions namely:

- ***<guiname>_OpeningFcn :*** For use in initialization just before the GUI is made visible.
- ***<guiname>_OutputFCn :*** For use to output messages onto the Matlabs command output window.

Programming the callback functions:

Programming the behaviour of your GUI will involve inserting your own Matlab code into the call back functions in the GUI's .m file.

Initially GUIDE would have generated all the commonly needed callback functions but they will only be simple stubs, i.e. functions that contain very little or no executable code. Never-the-less this code is ready and functioning from the very start that can be run to confirm that all the mouse-clicks, slider-bars, pop-up menus etc. operate as expected.

Without your own code not yet being in place, all your interactions will be generating calls to the corresponding callback functions that simply return control without doing any real work.

Transferring data between the callback functions:

Although all the callback functions are contained within the same .m file, they are semantically separate pieces of code that do not share any data except the data passed to them via the input parameters.

To make the task of passing data easier, GUIDE inserts code for the creation of a handles structure that contains all the data for the GUI objects such as the control buttons, menus, axes.

The handles structure (named: **handles**) is passed as an input parameter to all the callback functions that make up the GUI.

Access to data is achieved by accessing the components of this **handles** structure using the rules relating to structures. See Matlab part II lecture notes.

Accessing the GUI data from the handles structure:

The **handles** structure contains fields that are named with the tag names of all the components in your gui. As these structure fields must be unique it is important that all the object 'tags' have unique names.

For example if you have a radio_button which has the 'tag' property of 'mediumwave' then the handle of that radio_button will be accessible via the field **handles.mediumwave** .

Most of the property related information stored in the handles structure can be altered by using the **get()** and **set()** combination of functions.

For example:

get (handles.mediumwave , 'Value') will return whether the radio button is currently pressed (1.0) or not (0.0) Note that 'Value' is an important property of the radio button that indicates if it is currently in pressed position or not.

Using the example of the radio button, if we decide to make this button invisible we can do so by setting its property named 'Visible' to string 'off' by using the set function in the following manner:

set (handles.mediumwave , 'Visible' , 'off')

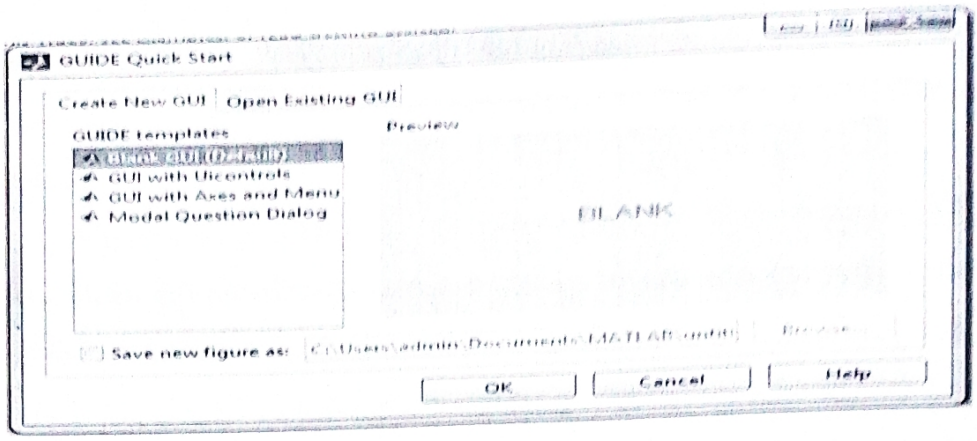
Practical examples:

Example1:

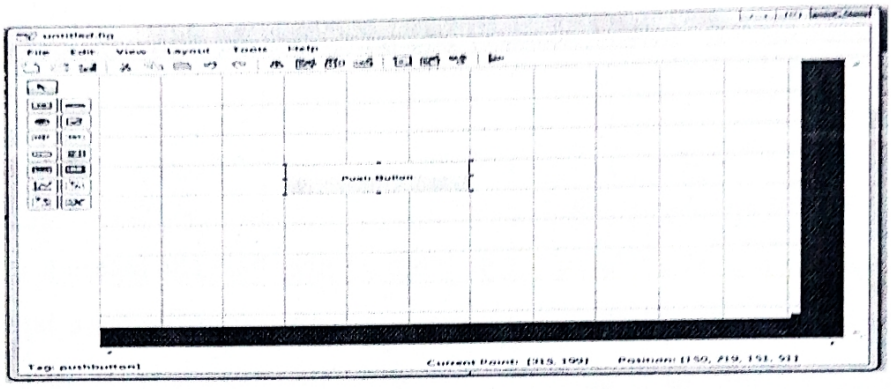
Changing the text of pushbutton component to 'hello world'

Steps:

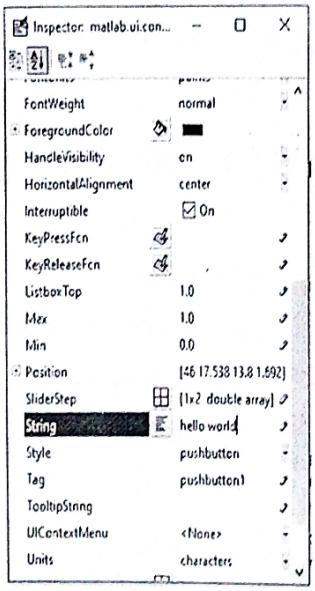
1. Type 'guide' on the command window. A window created as:



2. Click Blank GUI –> OK
3. Drag and drop pushbutton component to design layout



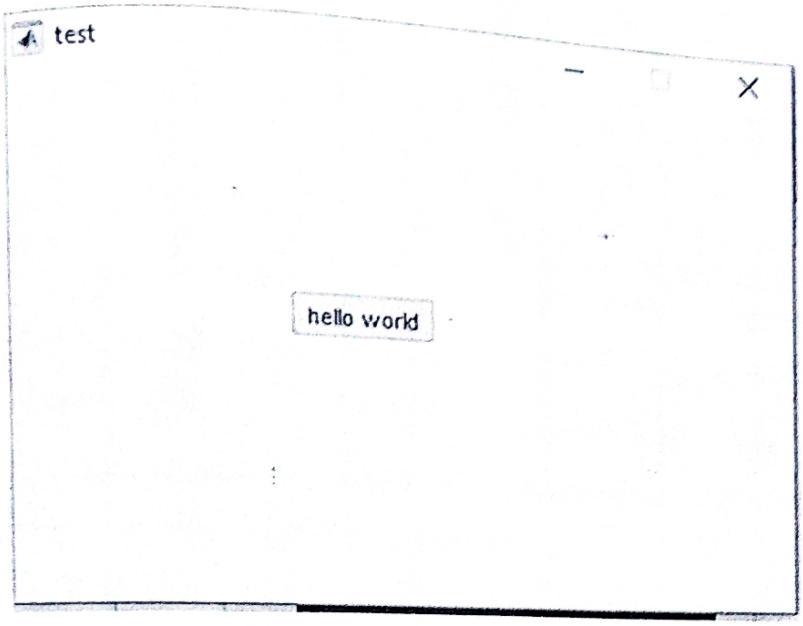
4. Double click on the pushbutton and change the string property inside *Property Inspector* to 'hello world' as shown:



5. Save the GUI window.

When a GUI is saved, it is saved in two forms:

1. in .fig
2. in .m

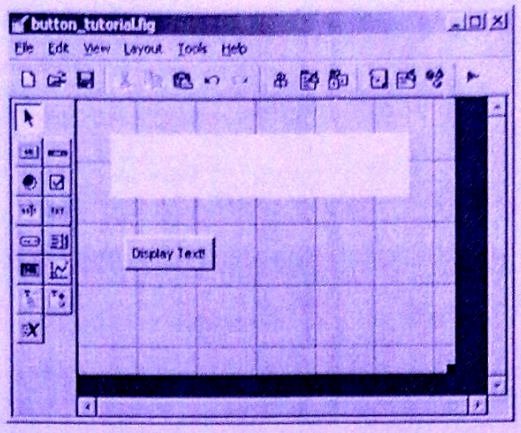


Example2:

In this example we will program the push button to display some text when it is pressed.

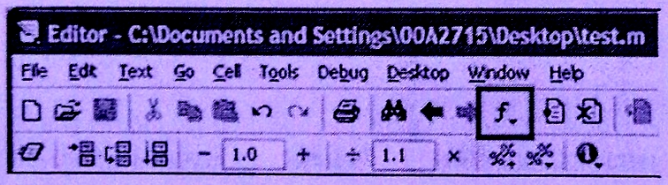
1. Go to the command window and type in guide.
2. Choose the first option Blank GUI.
3. Drag and drop Static Text component t to design layout.
4. Drag and drop pushbutton component to design layout .
5. Double click the Static Text component to bring up the Property Inspect
6. Change the String property so that there is nothing inside.
7. Change the Tag property to display staticText.
8. Similarly, double click on the Pushbutton component and change the String property to Display Text! and change the Tag property to display Text_pushbutton.

Figure should look like after you add the components and modify them as the following:



9. Save the GUI window with your desired filename.
10. Now, we are going to write the code for the GUI. When you save your GUI, Matlab automatically generates an .m file to go along with the figure that you just put together. The .m file is where we attach the appropriate code to the callback of each component.

    Open up the .m file that was automatically generated when you saved your GUI. In the Matlab editor, click on the $f_{\cdot}$ icon, which will bring up a list of the functions within the .m file. Select "Display Text"_pushbutton_Callback.

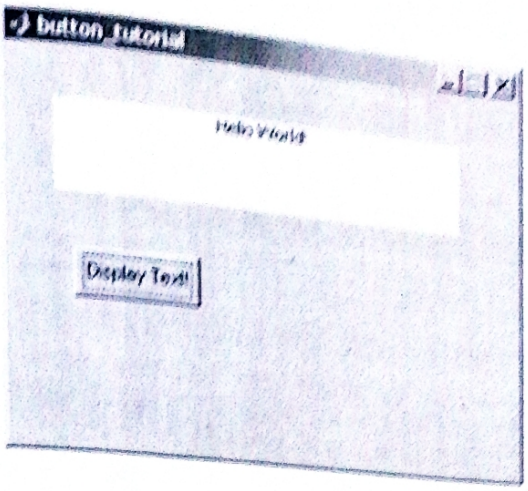


    Add the following code to the function:

    %display "Hello Wordl!" in the static text component when the pushbutton is pressed

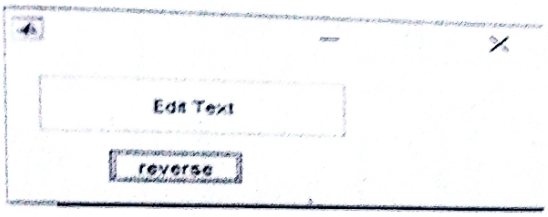
set(handles.display_staticText,'String','Hello World!');

11. Now that we've completed both the visual and code aspects of the GUI, its time to run the GUI to make sure it works before we move on. From the m-file editor, you can click on the ▶ icon to save and run the GUI. The GUI should appear once you click the icon. Now try clicking on the button to make sure that Hello World! appears on the GUI.

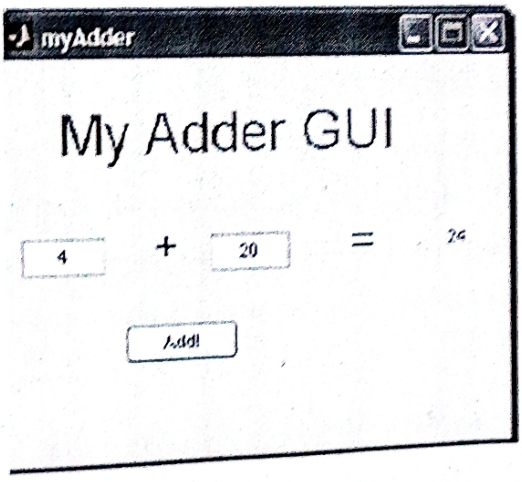


Exercises:

1. Design the following figure to reverse the string inside the editor whenever the pushbutton named (reverse) is pressed?



2. Design the following figure to add two numbers getting from Edit Text and putting the result on Static Text after pressing Push Button named(Add) is pressed?



3. Design the following figure to sum the element the matrix diagonal entered inside the editor whenever the pushbutton (sum) is pressed and print the result to the Static Text named (0)?



4. Design the following figure to plot selected function with specified color whenever the pushbutton (plot) is pressed. Use editor to enter the function period. NOTE: Any Radiobutton is selected will unselect the other and vice versa?

