MSc. Baraa Hashim Kareem

MSc. Murtada A. Mahdi

Lecture 2: Introduction to GUI

What Is GUIDE?

GUIDE, the MATLAB® Graphical User Interface development environment, provides a set of tools for creating graphical user interfaces (GUIs). These tools greatly simplify the process of designing and building GUIs. You can use the GUIDE tools to

- Lay out the GUI

Using the GUIDE Layout Editor, you can lay out a GUI easily by clicking and dragging GUI components — such as panels, buttons, text fields, sliders, menus, and so on — into the layout area.

- Program the GUI

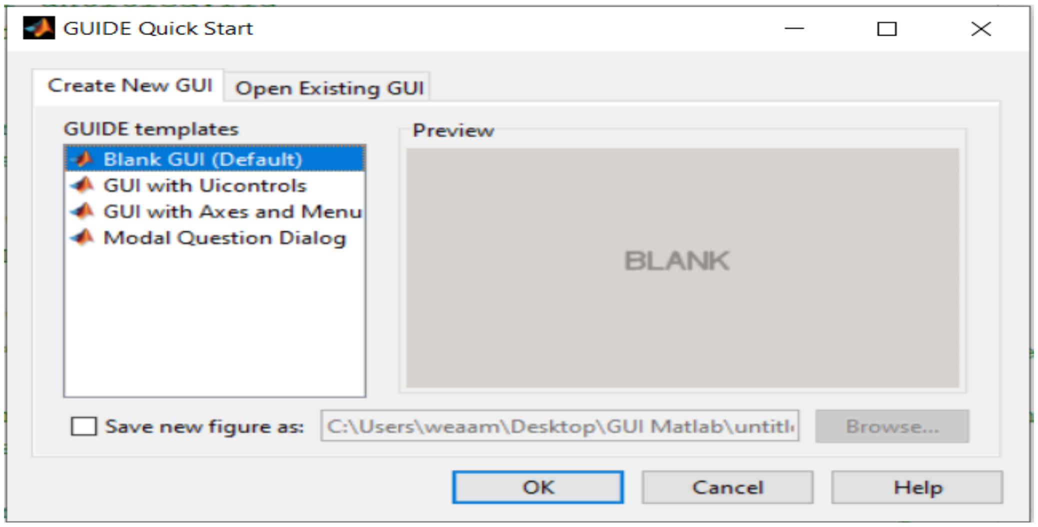
GUIDE automatically generates an M-file that controls how the GUI operates. The M-file initializes the GUI and contains a framework for all the GUI callbacks — the commands that are executed when a user clicks a GUI component. Using the M-file editor, you can add code to the callbacks to perform the functions you want them to.

Starting GUIDE

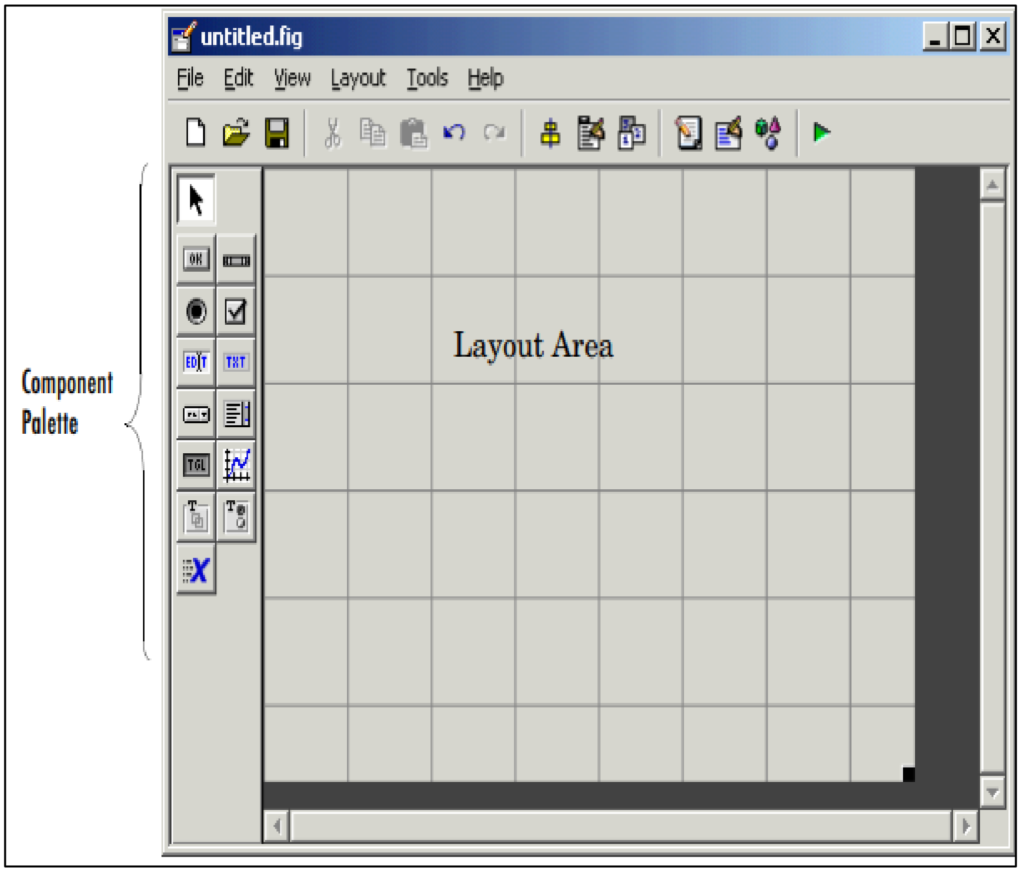
- To start GUIDE, enter **guide** at the MATLAB prompt. This displays the **GUIDE Quick Start** dialog as shown in the following figure.

MSc. Baraa Hashim Kareem

MSc. Murtada A. Mahdi



- Then, an untitled figure will pop up. You have some components on the left menu, which you can drag onto your interface.



Example1:

Create a simple GUI in MATLAB to display the message **(Hello World!) and** clear it.

- In this example we are going to use only two '**push buttons**' and one '**static text**'.
- Drag and drop a '**static text**' onto your MATLAB GUI. You can reduce or increase the size of your interface window by dragging its bottom-right corner, as it's done in other drawing programs.
- Double click on this **'static text'** and a '**Property Inspector**' window will appear. Scroll down, look for the 'String' property, and delete what's there.
- Make the '**Tag'** property of '**static text**' to be '**output_line'**. You can use whatever name you want. **Tag** means it's the object's name or identifier as it will be recognized in the rest of the code.
- drag-and-drop a '**push button**' onto your interface. Modify its '**String**' property to read '**Launch Message**'.
- Drag-and-drop another '**push button**'. Modify its '**String**' property to read '**Clear Message**'
- Save your figure. You'll be taken to the Matlab code (in the editor window) that will drive your interface. Matlab has **automatically** created functions related to your components.

The '**Callback'** functions are the instructions that will be executed when the user pushes the buttons or does something with the components that you have included in your Matlab GUI.

A '**set**' instruction sets the properties of the elements that you indicate. Do you remember that you have a '**static text**' with the tag (identifier) '**output_line**'?

We are going to modify ‘**output_line**’ when the user pushes the button ‘**Launch Message**’. This is accomplished with the **instruction**.

```
set(handles.output_line,'String','Hello World!!')
```

The **first parameter is the object** (component) that you’re going to modify. It starts with ‘**handles.**’. The **second argument is the object’s property** that you’re going to modify, and in this case, is the ‘**String**’ property. The **third argument is the value** that you want to assign to the property.

So, the result is that when the user presses the ‘**Launch Message**’ button, a message reading ‘**Hello World!!**’ will appear in the ‘**output line**’ (officially named ‘**handles.output_line**’). Add this single line to the code, so that it looks like this:

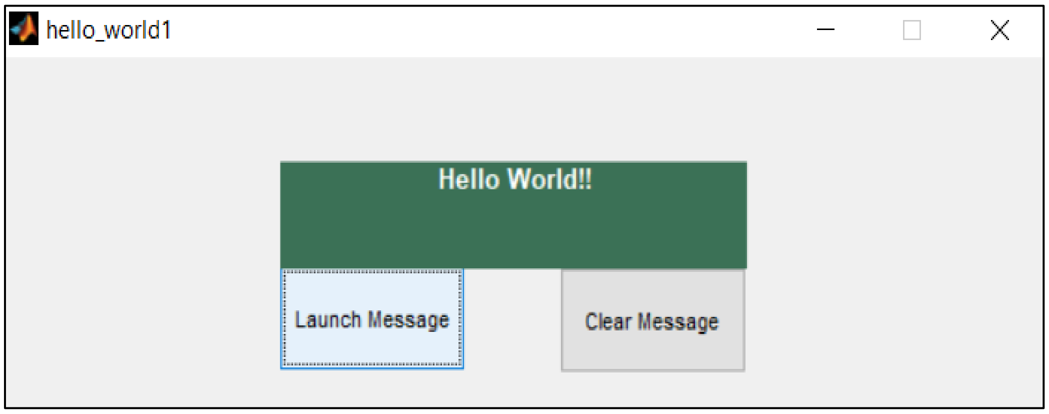
```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MAT
% handles structure with handles and user data (see GUIDATA)
set(handles.output_line,'String','Hello World!!')
```

We’ll do something similar to the ‘**callback**’ corresponding to the ‘**Clear Message**’ button.

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MAT
% handles structure with handles and user data (see GUIDATA)
set(handles.output_line,'String','')
```

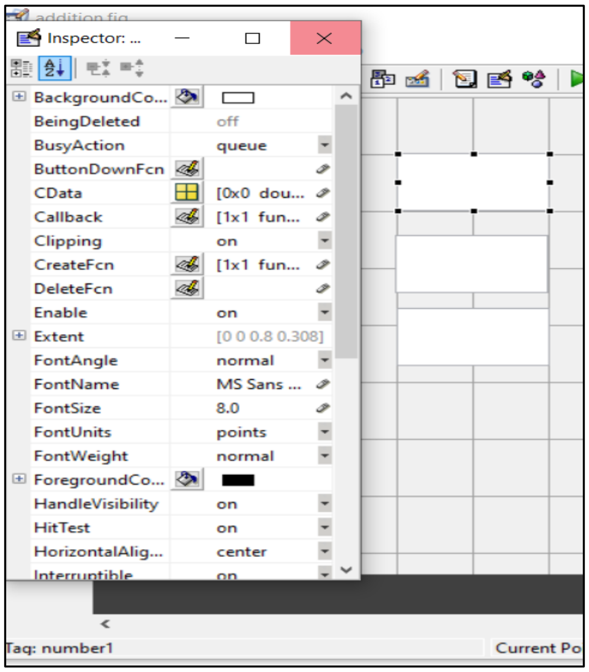
The result is that when the user presses the ‘**Clear Message**’ button, a blank message will appear in the ‘**output line**’

- Now, run your interface by clicking the '**run**' icon at the top of the editor window...



Setting Component Properties — The Property Inspector

The Property Inspector enables you to set the properties of the components in your layout. It provides a list of all settable properties and displays the current value. Each property in the list is associated with an editing device that is appropriate for the values accepted by the particular property. For example, a color picker to change the BackgroundColor, a pop-up menu to set FontAngle, and a text field to specify the Callback string.

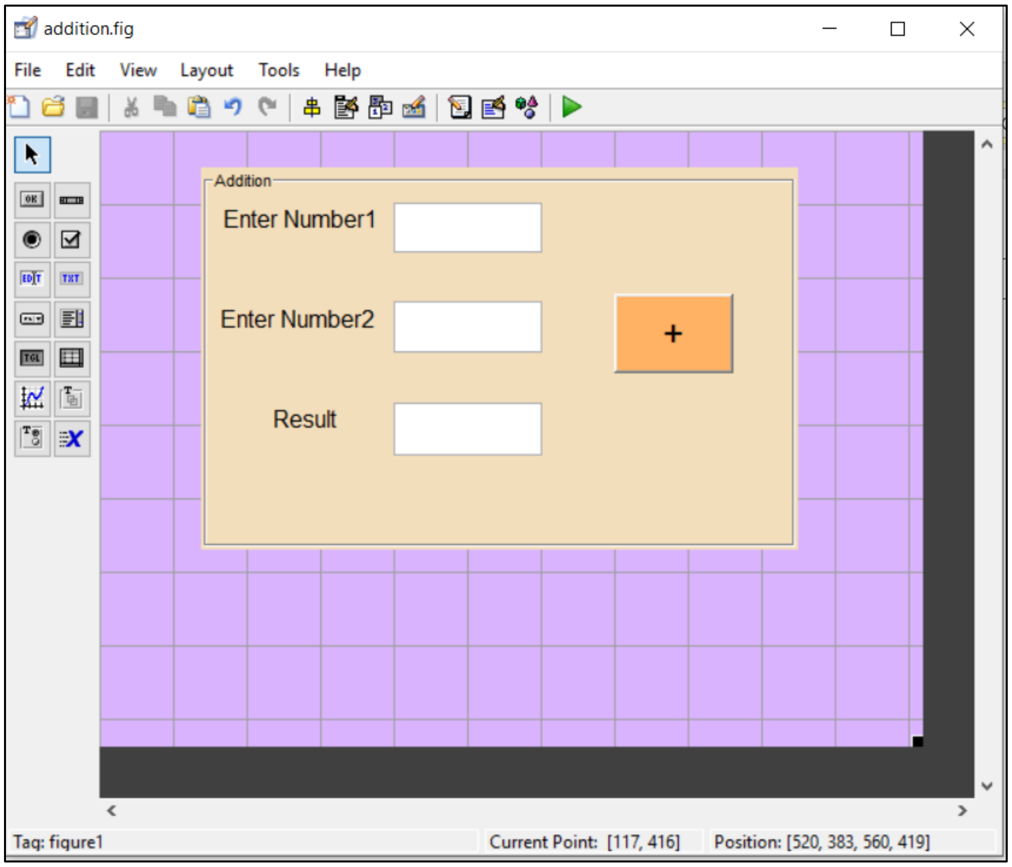


BackgroundCo...	▭
BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
CData	[0x0 dou...
Callback	[1x1 fun...
Clipping	on
CreateFcn	[1x1 fun...
DeleteFcn	
Enable	on
Extent	[0 0 0.8 0.308]
FontAngle	normal
FontName	MS Sans ...
FontSize	8.0
FontUnits	points
FontWeight	normal
ForegroundCo...	■
HandleVisibility	on
HitTest	on
HorizontalAlig...	center
Interruptible	on

Example2:

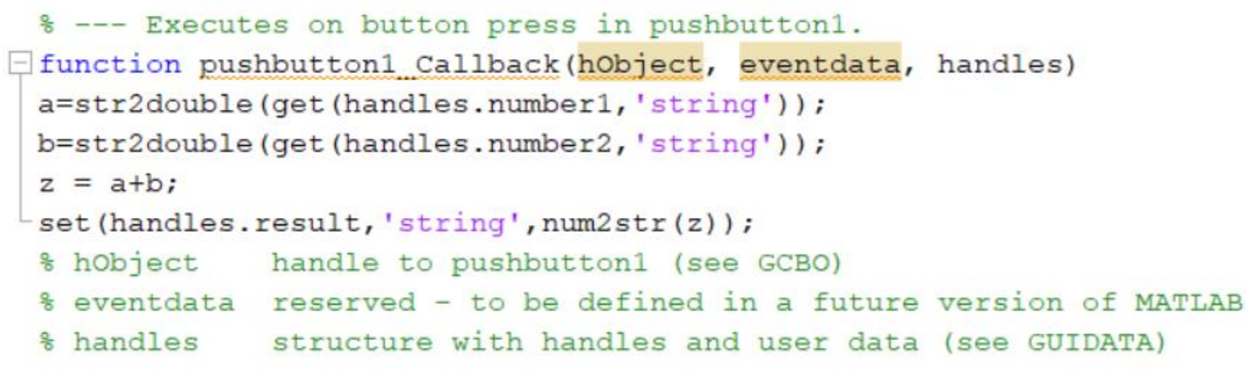
Create a small basic **GUI** that can perform arithmetic operations like addition, and a clear button that will clear the screen.

- In this example we are going to use only one '**push button**' and three '**static text**' and three '**Edit text**' and one panel.
- Labeling the three '**static text**' with names number 1, number 2, and Result. By changing the string in the property inspector list. And change the string of the pushbutton to (+).
- Give the panel title Addition.



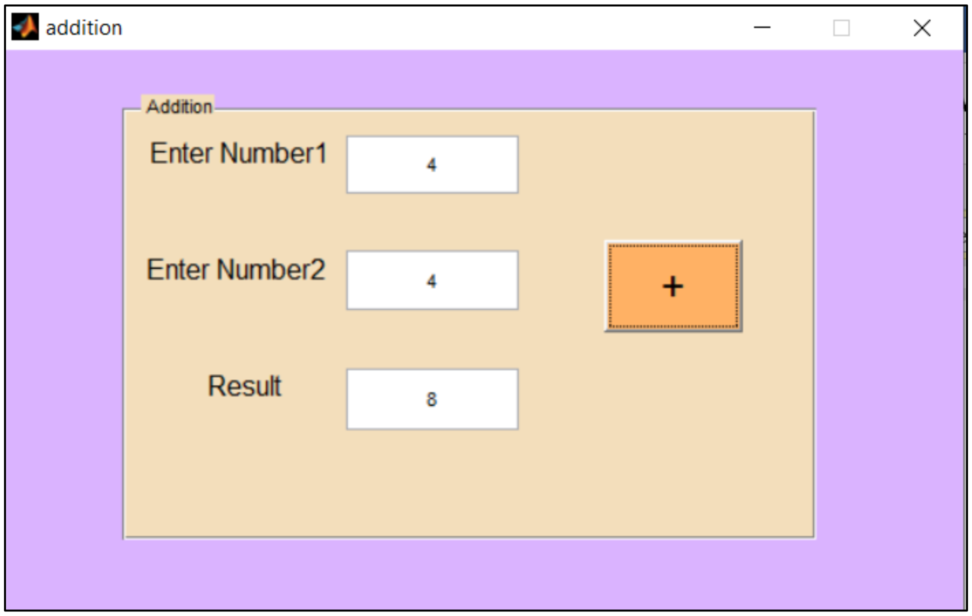
Tag: figure1 Current Point: [117, 416] Position: [520, 383, 560, 419]

Adding code:



```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
a=str2double(get(handles.number1,'string'));
b=str2double(get(handles.number2,'string'));
z = a+b;
set(handles.result,'string',num2str(z));
% 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)
```

➢ The result is



Enter Number1 4

Enter Number2 4

Result 8