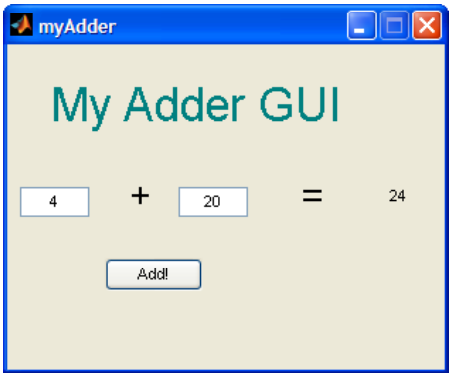
# Chapter 6
# MATLAB GUI

MATLAB GUI (Graphical User Interface) Tutorial for Beginners

By J.S. Park
University of Incheon

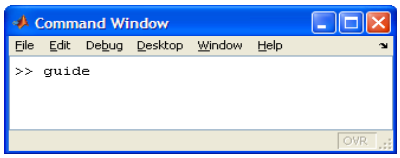
# Preprocessing Data

Why use a GUI in MATLAB?
It makes things simple for the end-users of the program.
The command line interface Vs. GUI

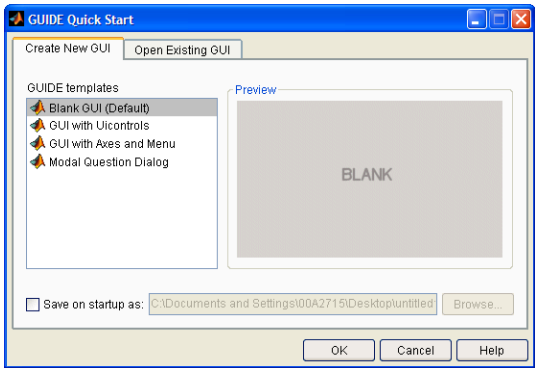


# Initializing GUIDE (GUI Creator)

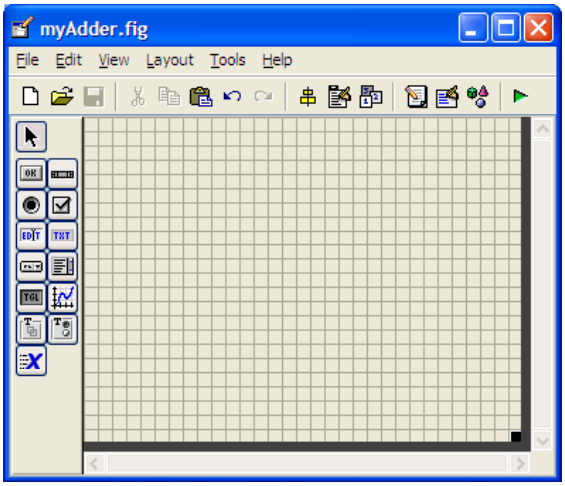
1. Open up MATLAB. Go to the command window and type in guide



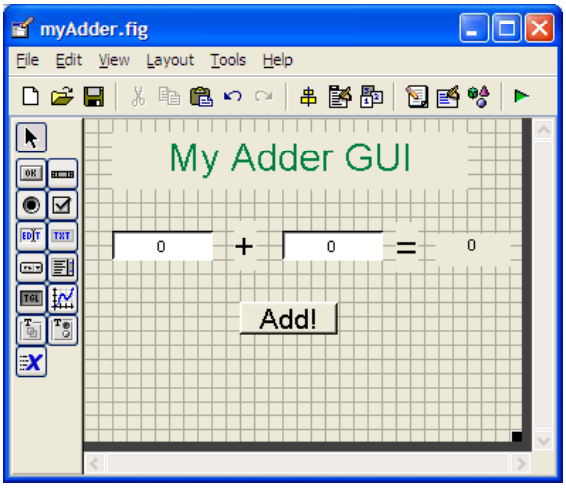

2. Choose the first option Blank GUI (Default)



3. You should now see the following screen.



4. Before adding components blindly, it is good to have a rough idea about how you want the graphical part of the GUI to look like.



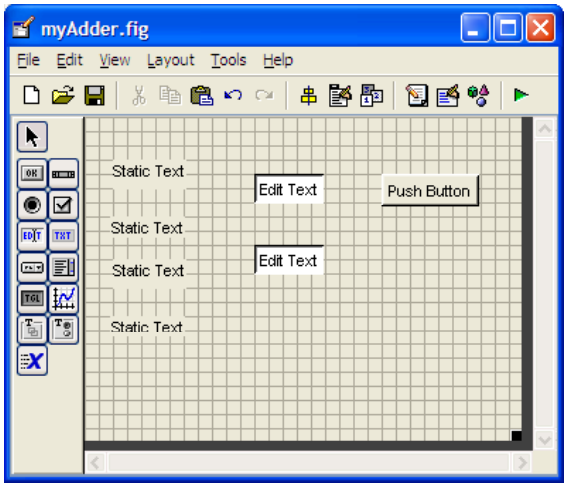
# Creating the Visual Aspect of the GUI: Part 1

1. For the adder GUI, we will need the following components


Two Edit Text components

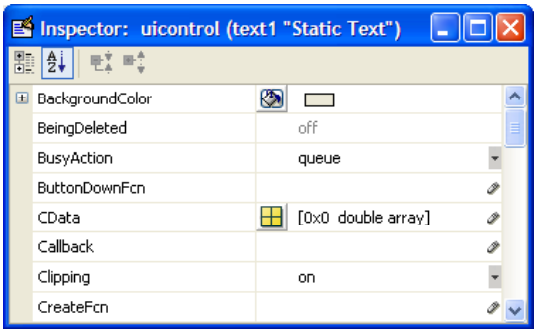

Four Static Text component

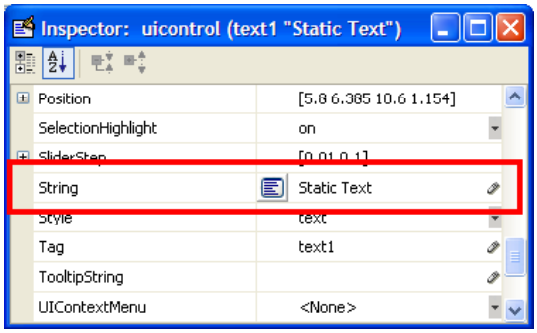

One Pushbutton component

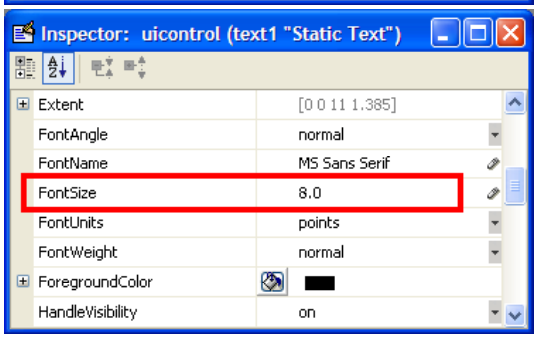


2. Edit the properties of these components.

   Double click one of the *Static Text* components. You should see the *property Inspector.*

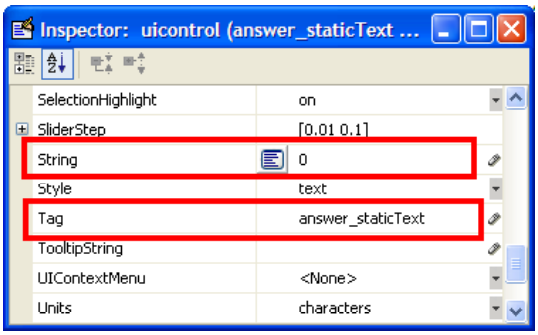


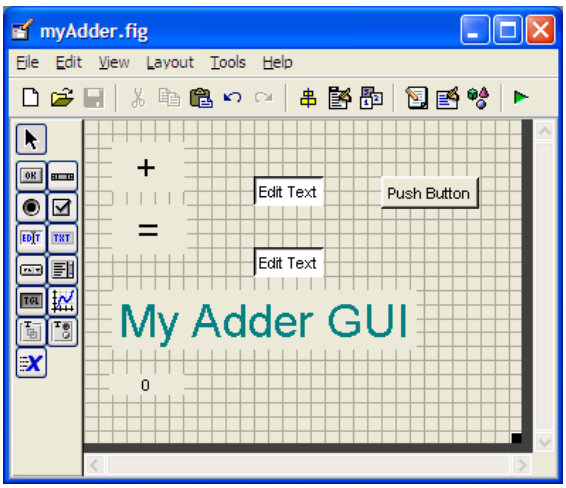



3. Do the same for the next *Static Text* component, but instead of changing the *String* pa rameter to +, change it to =, and another it to MyAdderGUI.

4. For *Static Text* component 0, modify the *Tag* parameter to answer_staticText.

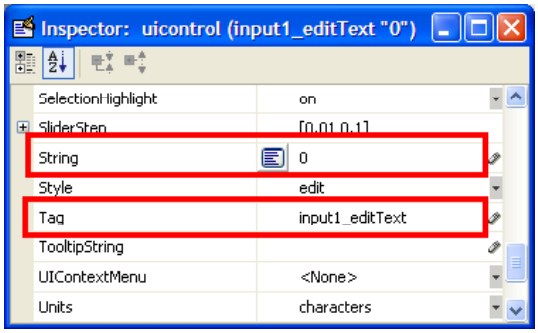



5. You should have something that looks like the following:



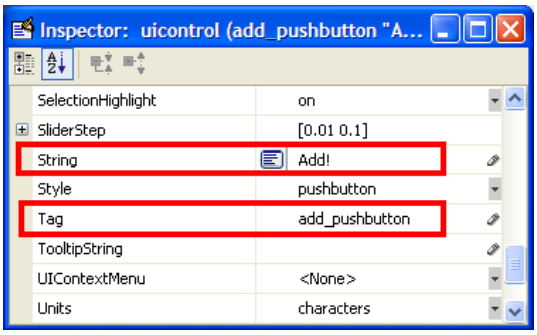
# Creating the Visual Aspect of the GUI: Part 2

1. Modify the *Edit Text* components. Double click on the first *Edit Text* component.
   Set the *String* parameter to 0
   Change the *Tag* parameter to input1_editText

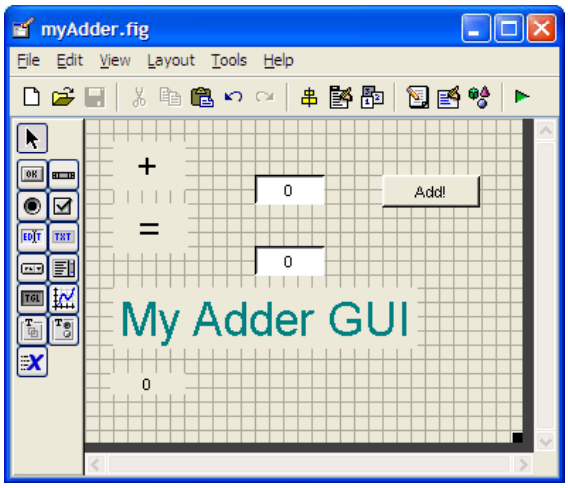



2. The second *Edit Text* component, set the *String* parameter to 0
   Set the *Tag* parameter input2_editText.

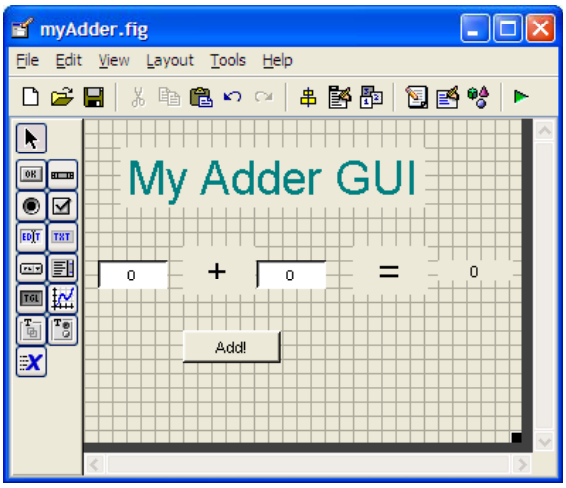
3. Modify the *pushbutton* component.
   Change the *String* parameter to Add!
   Change the *Tag* parameter to add_pushbutton.



4. You should have something like this:




You should have something like this:

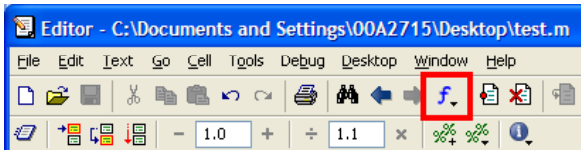



5. Save your GUI under any file name you please. I chose to name mine myAdder.
When you save this file, MATLAB automatically generates two files:
*myAdder.fig* and *myAdder.m*.
The .fig file contains the graphics of your interface.
The .m file contains all the code for the GUI.

# Writing the Code for the GUI Callbacks

1. Open up the .m file that was automatically generated when you saved your GUI.

2. In the MATLAB editor, click on the icon, which will bring up a list of the functions within the .m file. Select *input1_editText_Callback*.



3. The cursor should take you to the following code block:

```
function input1_editText_Callback(hObject, eventdata, handles)
% hObject    handle to input1_editText (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,'String') returns contents of input1_editText as text
%       str2double(get(hObject,'String')) returns contents of
%       input1_editText as a double
```

4. Add the following code to the bottom of that code block:

```
%store the contents of input1_editText as a string. if the string
%is not a number then input will be empty
input = str2num(get(hObject,'String'));

%checks to see if input is empty. if so, default input1_editText to zero
if (isempty(input))
    set(hObject,'String','0')
end
```

5. Add the same block of code to *input2_editText_Callback*.

6. Now we need to edit the *add_pushbutton_Callback*.

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

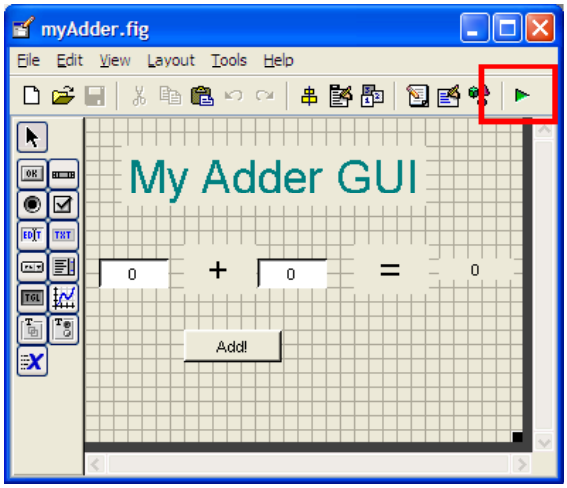
Here is the code that we will add to this callback:

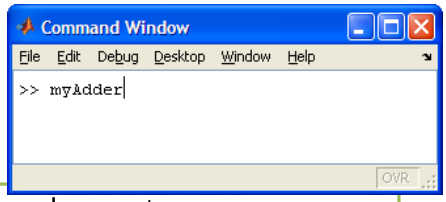
```
a = get(handles.input1_editText,'String');
b = get(handles.input2_editText,'String');
% a and b are variables of Strings type, and need to be converted
% to variables of Number type before they can be added together

total = str2num(a) + str2num(b);
c = num2str(total);
% need to convert the answer back into String type to display it
set(handles.answer_staticText,'String',c);
```

# Launching the GUI

7. There are two ways to launch your GUI.

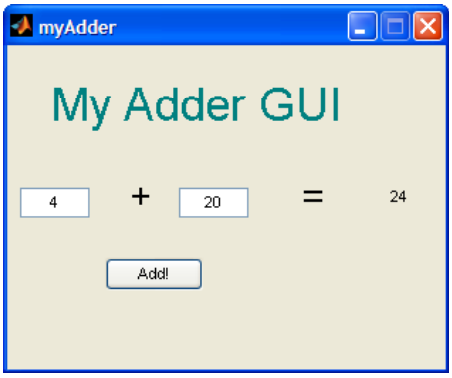
The first way: Press the  icon on the GUIDE editor.



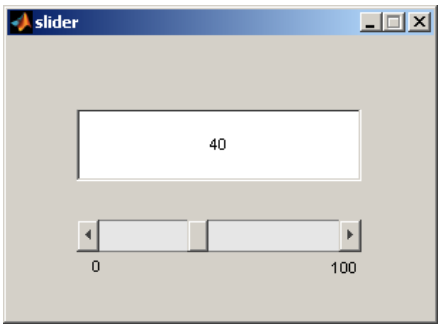

The second method : Launch the GUI from the MATLAB command prompt.
Type in the name of the GUI at the command prompt.

8. The GUI should start running immediately:



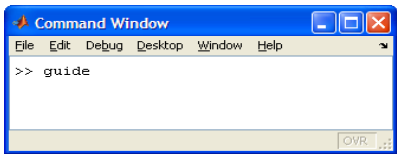
# MATLAB GUI Tutorial - Slider

In this Matlab GUI tutorial, you will learn how to create and use the slider component.
Sliders are useful controls for choosing a value in a range of values.
Common uses are volume controls, seekers for movie and sound files as well as color pic kers.
An example of a slider is shown below.

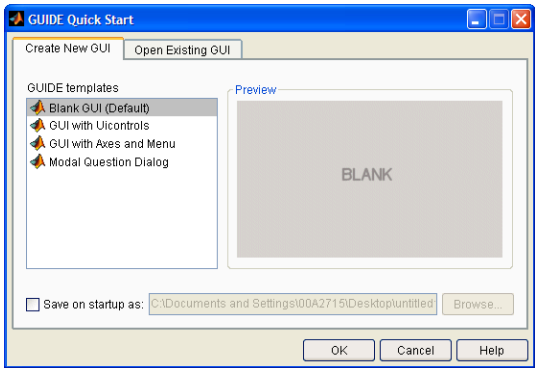


# Create the Visual Aspect of the GUI

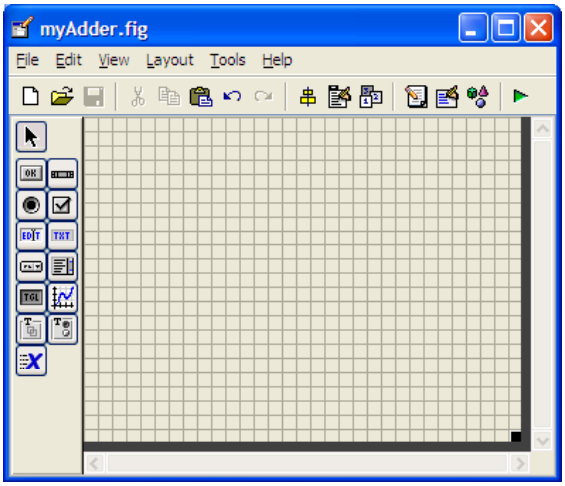
1. Open up MATLAB. Go to the command window and type in guide




2. Choose the first option Blank GUI (Default)



3. You should now see the following screen.

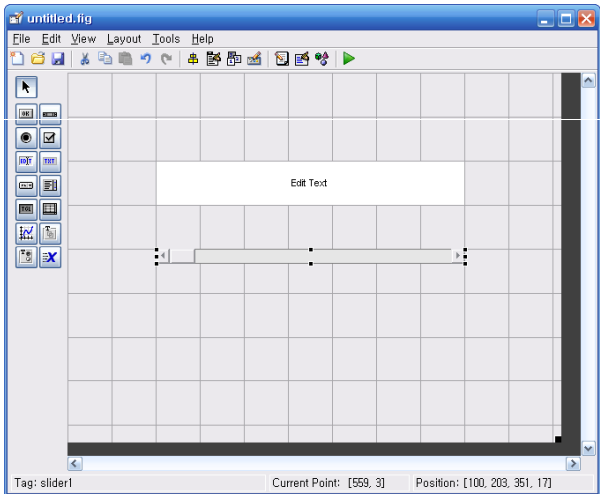


# Creating the Visual Aspect of the GUI: Part 1

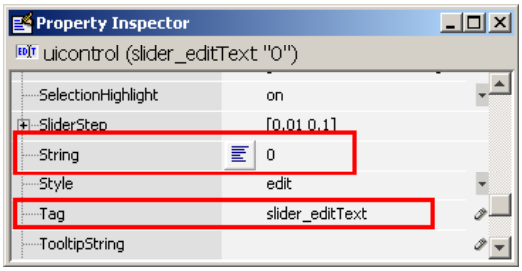
1. For the adder GUI, we will need the following components

Add an *Edit Text* component to the GUI figure.

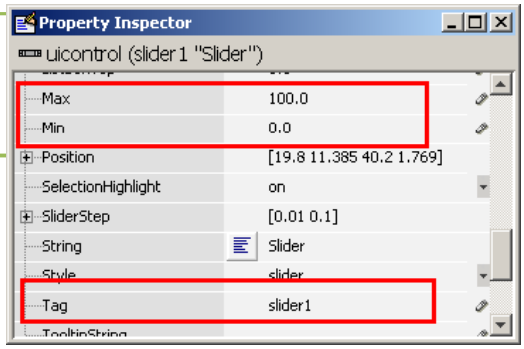
Add a *Slider* component onto the GUI figure.



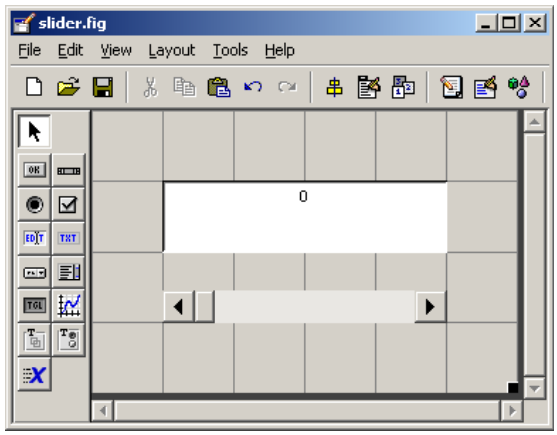
2. Edit the properties of these components.
   Double click the *Edit Text* component to bring up the Property Inspector.
   Change the *String* property to 0, and
   change the *Tag* property to sliderValue_editText.



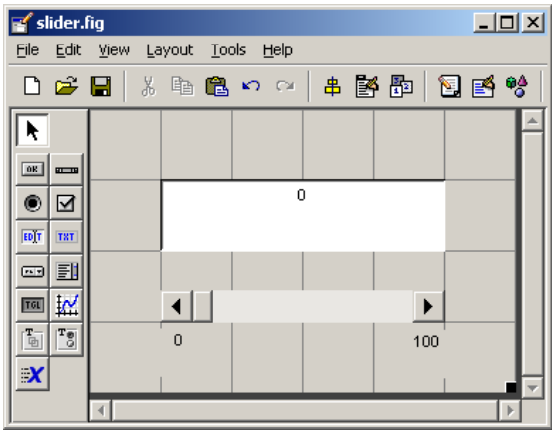

3. Modify the properties of the *Slider* component.
   Sit the *Min* property to 0, and
   the *Max* property to 100.
   Change the *Tag* property to slider1.



4. The figure should look like after you add the components and modify them.



5. Add some *Static Text* components to specify the min and max values of the slider.
   Modify their text by double clicking on the component and
   changing the *String* property.
   It's not required, but I highly recommend it.

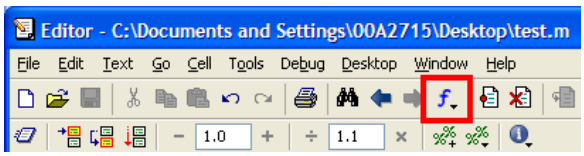



6. Save your GUI wherever you please with your desired filename.

# Writing the Code for the GUI Callbacks

1. Open up the .m file that was automatically generated when you saved your GUI.

2. In the MATLAB editor, click on the icon, which will bring up a list of the functions within the .m file.
   Select *slider1_Callback*.



3. Add the following code to the function:

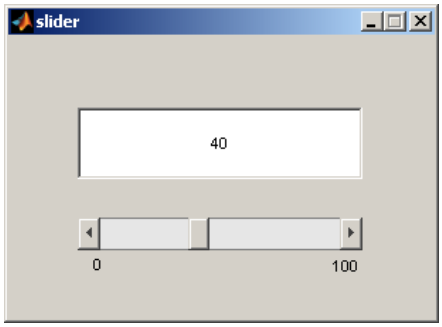
```
%obtains the slider value from the slider component
sliderValue = get(handles.slider1,'Value');

%puts the slider value into the edit text component
set(handles.slider_editText,'String', num2str(sliderValue));

% Update handles structure
guidata(hObject, handles);
```

4. Add the following code to the *slider_editText_Callback* function:

```
%get the string for the editText component
sliderValue = get(handles.slider_editText,'String');

%convert from string to number if possible, otherwise returns empty
sliderValue = str2num(sliderValue);

%if user inputs something is not a number,
%or if the input is less than 0
%or greater than 100, then the slider value defaults to 0
if (isempty(sliderValue) || sliderValue < 0 || sliderValue > 100)
    set(handles.slider1,'Value',0);
    set(handles.slider_editText,'String','0');
else
    set(handles.slider1,'Value',sliderValue);
end
```

# Run and Test the GUI

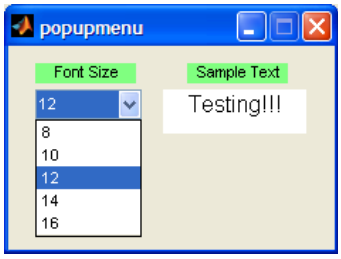
1. From the m-file editor, you can click on the icon  to save and run the GUI.
2. Alternatively, from the GUIDE editor, you can click on the  to launch the GUI.




Now, try to put in different types of inputs to test the GUI.
Any input that is not a number, less than zero,
or greater than 100 should default the slider to a value of zero.

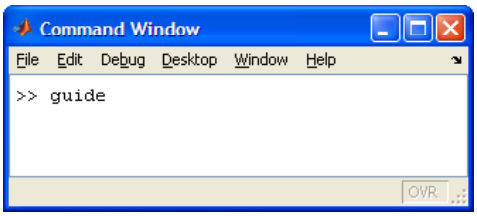
# Matlab GUI Tutorial - Pop-up Menu

In this Matlab GUI tutorial, you will learn how to create and use the *Pop-up Menu* component.
Pop-up menus are used as a control for choosing between a set of options.
When the user clicks on the Pop-up menu, the menu expands, revealing a set of choices that the user can pick.
A common use for Pop-up menus is a font size selector (shown below).

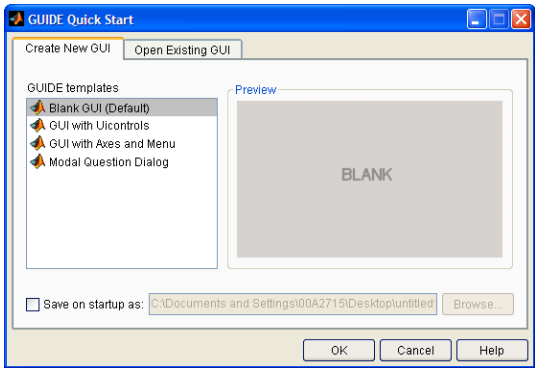


# Create the Visual Aspect of the GUI

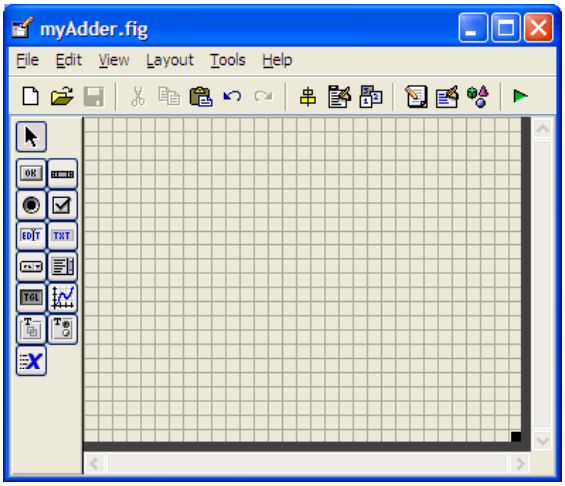
1. Open up MATLAB. Go to the command window and type in guide




2. Choose the first option Blank GUI (Default)



3. You should now see the following screen.

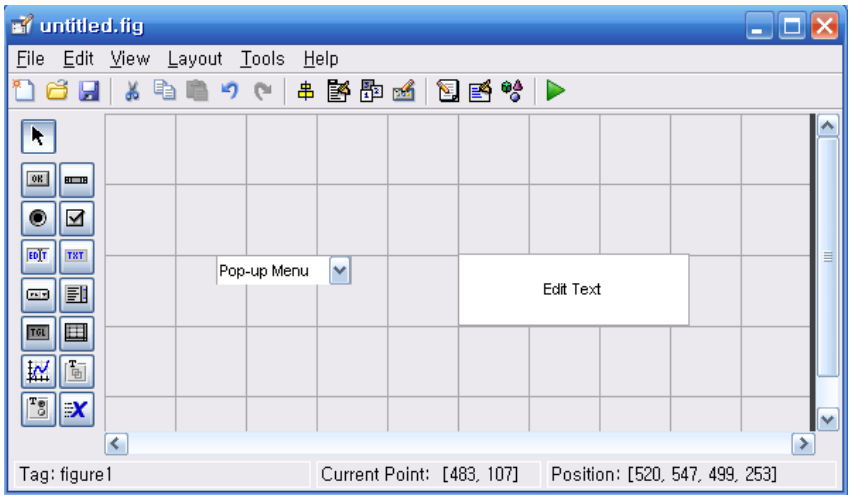


# Creating the Visual Aspect of the GUI: Part 1

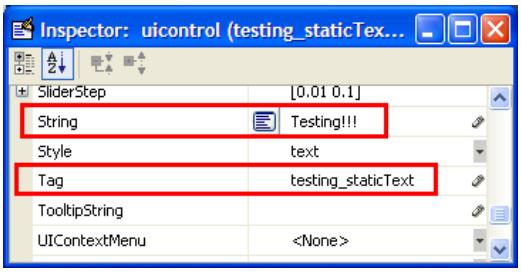
1. For the adder GUI, we will need the following components

Add an *Edit Text* component to the GUI figure.

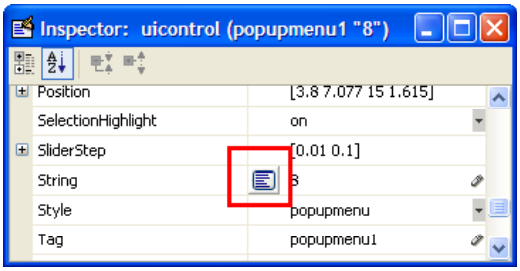
Add a *Pop-up Menu* component onto the GUI figure.



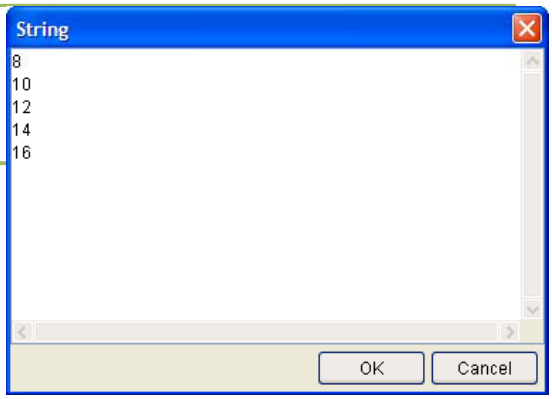
2. Double click the *Static Text* component to bring up the Property Inspector.
   Change the *String* property to Testing!!!, and
   change the *Tag* property to testing_staticText as shown in the figure below:



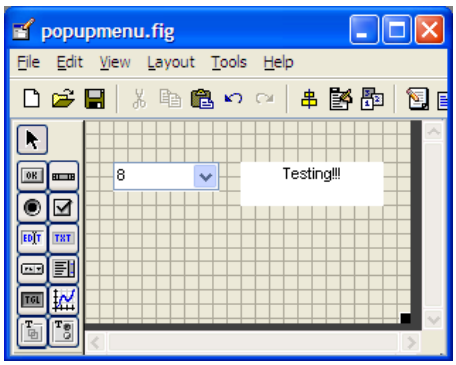
3. Modify the properties of the *Pop-up Menu* component.
   Click on the icon on the *String* property line as shown below.



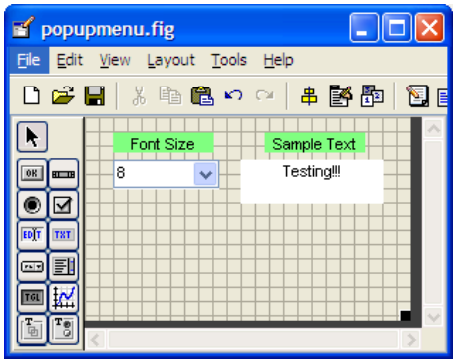

4. After clicking on the icon,
   you should now see the following window.
   Fill in the window as shown below:



5. The figure should look like after you add the components and modify them.



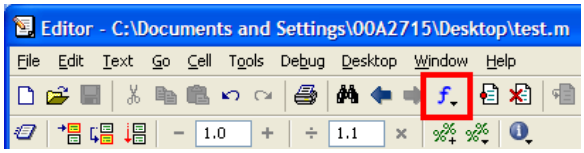
6. Add some *Static Text* components to add some description tags to the GUI. Modify their text by double clicking on the component and changing the *String* property.



# Writing the Code for the GUI Callbacks

1. Open up the .m file that was automatically generated when you saved your GUI.

2. In the MATLAB editor, click on the icon, which will bring up a list of the functions within the .m file.
   Select *popupmenu1_Callback*.

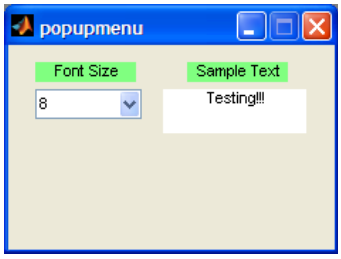


3. Add the following code to the function:

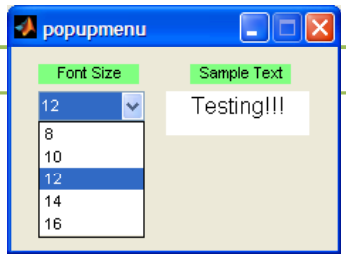
```
%gets the selected option
switch get(handles.popupmenu1,'Value')
    case 1
        set(handles.testing_staticText,'FontSize',8);
    case 2
        set(handles.testing_staticText,'FontSize',10);
    case 3
        set(handles.testing_staticText,'FontSize',12);
    case 4
        set(handles.testing_staticText,'FontSize',14);
    case 5
        set(handles.testing_staticText,'FontSize',16);
    otherwise
end
```

# Run and Test the GUI

1. From the m-file editor, you can click on the icon  to save and run the GUI.
2. Alternatively, from the GUIDE editor, you can click on the  to launch the GUI.

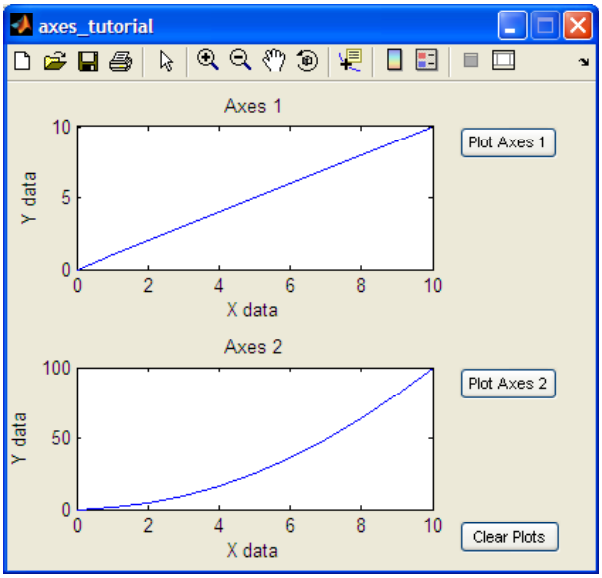



Go ahead and try selecting different font sizes.



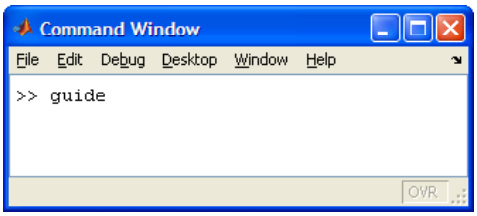
# MATLAB GUI Tutorial - Plotting Data to Axes

In this Matlab GUI tutorial, you will learn how to create and use the *Axes* component.
The *Axes* component allows you to display graphics, such as graphs and images on your GUI.
In this tutorial, we will create two axes on the GUI and plot some simple data onto it.
In addition, we will include a reset button to clear the axes and
we will also add the standard toolbar to allow the user to zoom, pan, and query the plot.

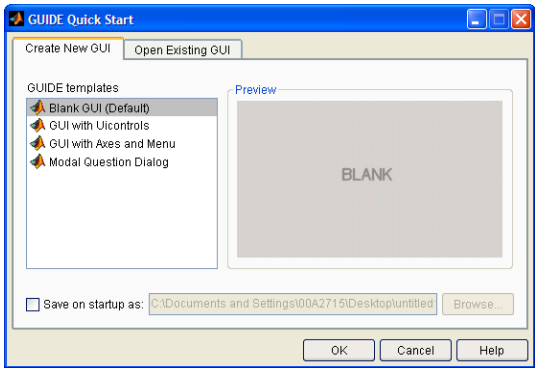


# Create the Visual Aspect of the GUI

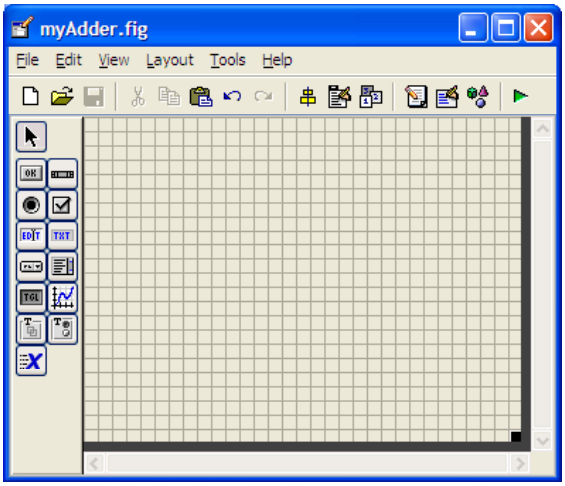
1. Open up MATLAB. Go to the command window and type in guide




2. Choose the first option Blank GUI (Default)



3. You should now see the following screen.



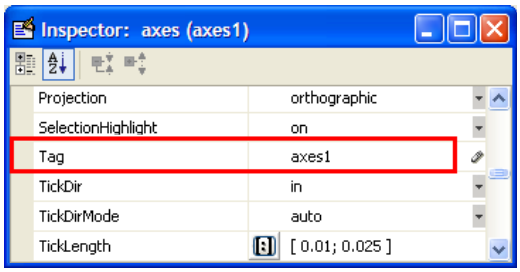
# Creating the Visual Aspect of the GUI: Part 1

1.For the adder GUI, we will need the following components.


Add two *Axes* components to the GUI figure.


Add three *Pushbutton* components onto the GUI figure.

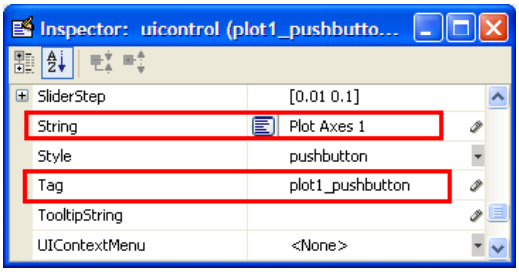
2. Double click the *Axes* component to bring up the Property Inspector.
   The *Tag* property is named axes1.
   The other *Axes* component's *Tag* property is named axes2.



3. Modify the properties of the *Pushbutton* components.

   Double click on one of the *Pushbutton* components.

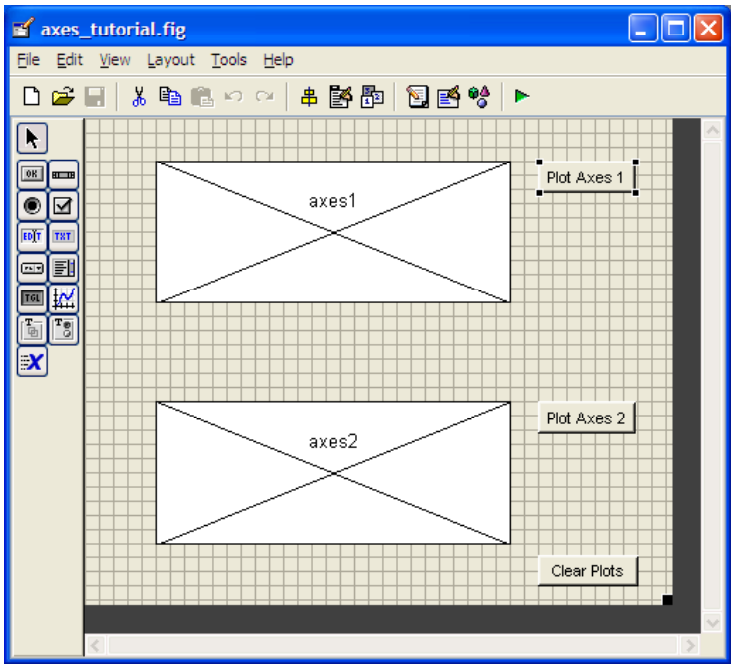
   Change the *String* property to Plot Axes 1, and the *Tag* property to plotAxes1_pushbutton




4. Double click on the next pushbutton and change the *String* property to Plot Axes 2 and change the *Tag* property to plotAxes2_pushbutton.

   Double click on the final pushbutton and change the *String* property to Clear Axes and change the *Tag* property to clearAxes_pushbutton.

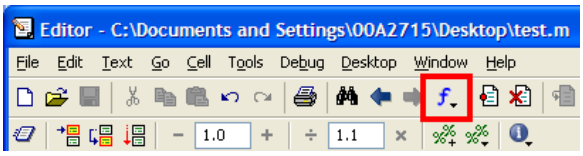
5. The figure should look like below after you add the components and modify them.



# Writing the Code for the GUI Callbacks

1. Open up the .m file that was automatically generated when you saved your GUI.

2. In the MATLAB editor, click on the icon, which will bring up a list of the functions within the .m file.
   Select *plot1_pushbutton_Callback*.



3. Add the following code to the function:

```
%selects axes1 as the current axes, so that
%Matlab knows where to plot the data
axes(handles.axes1)

%creates a vector from 0 to 10, [0 1 2 3 . . . 10]
x = 0:10;
%creates a vector from 0 to 10, [0 1 2 3 . . . 10]
y = 0:10;

%plots the x and y data
plot(x,y);
%adds a title, x-axis description, and y-axis description
title('Axes 1');
xlabel('X data');
ylabel('Y data');
guidata(hObject, handles); %updates the handles
```

4. Put the following code into the *plot2_pushbutton_Callback*:

```
%selects axes2 as the current axes, so that
%Matlab knows where to plot the data
axes(handles.axes2)

%creates a vector from 0 to 10, [0 1 2 3 . . . 10]
x = 0:10;
%creates a vector  [0 1 4 9 . . . 100]
y = x.^2

%plots the x and y data
plot(x,y);
%adds a title, x-axis description, and y-axis description
title('Axes 2');
xlabel('X data');
ylabel('Y data');
guidata(hObject, handles); %updates the handles
```

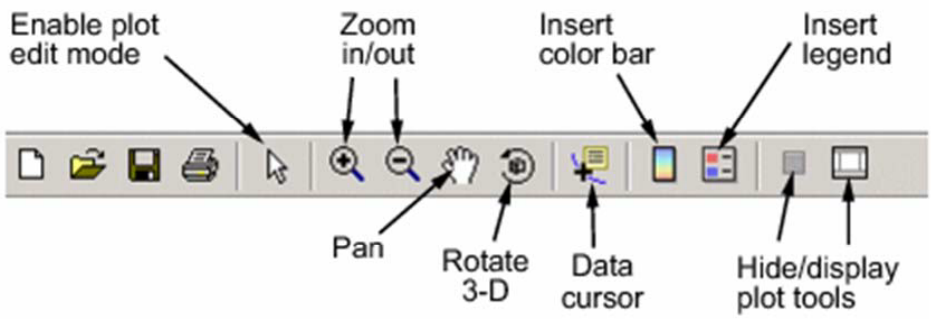
5. Add some code to the *clearPlots_pushbutton_Callback*:

```
%these two lines of code clears both axes
cla(handles.axes1,'reset')
cla(handles.axes2,'reset')
guidata(hObject, handles); %updates the handles
```

5. Add the following line of code to *axes_tutorial_OpeningFcn*:

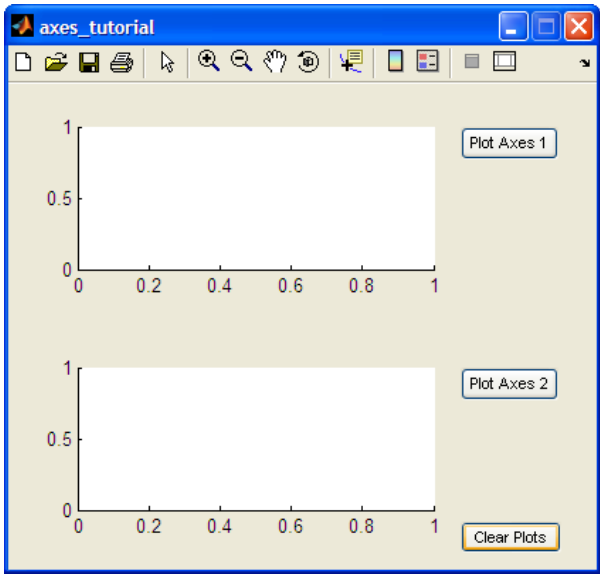
```
set(hObject,'toolbar','figure');
```

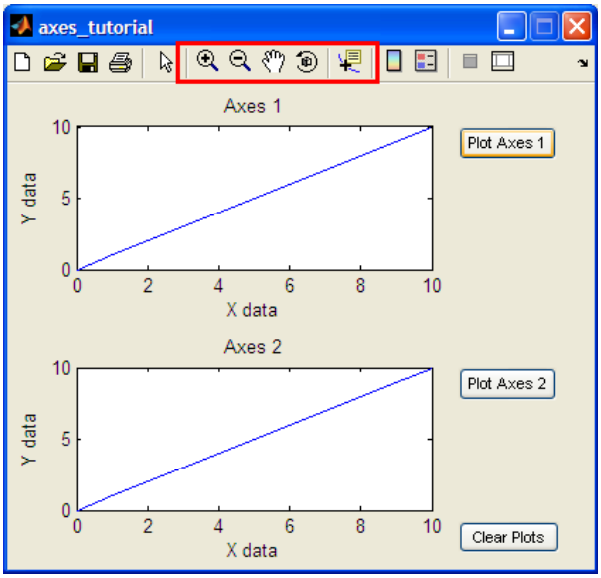
This line of code effectively adds the standard toolbar to the GUI, allowing the user to zoom, pan, query the plot, and more.



# Run and Test the GUI

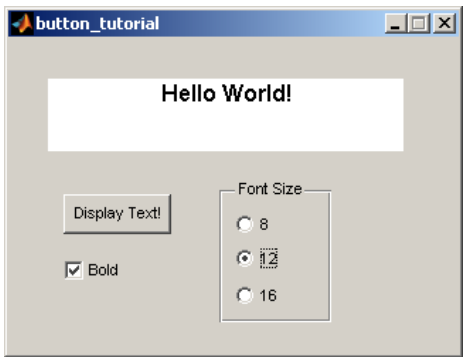
1. From the m-file editor, you can click on the icon to save and run the GUI.
2. Alternatively, from the GUIDE editor, you can click on the to launch the GUI.



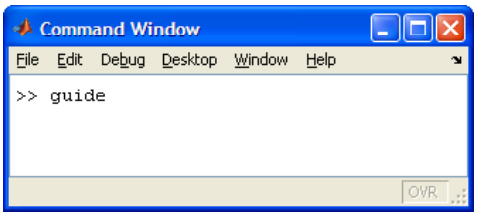
# MATLAB GUI Tutorial - Button Types and Button Group

You will learn how to use the different types of buttons available within Matlab GUIs.
These button types are: push button, radio button, check box, and toggle buttons.
In addition, you will learn how to use the button panel to control a group of buttons.

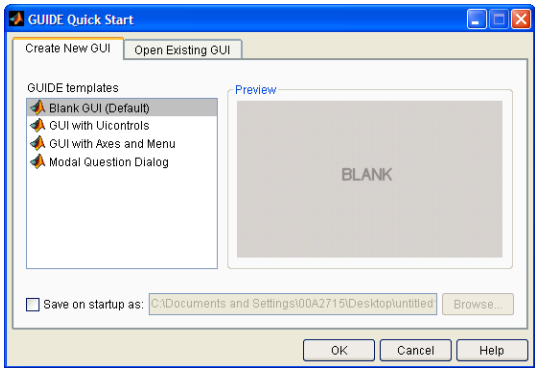


# Create the Visual Aspect of the GUI

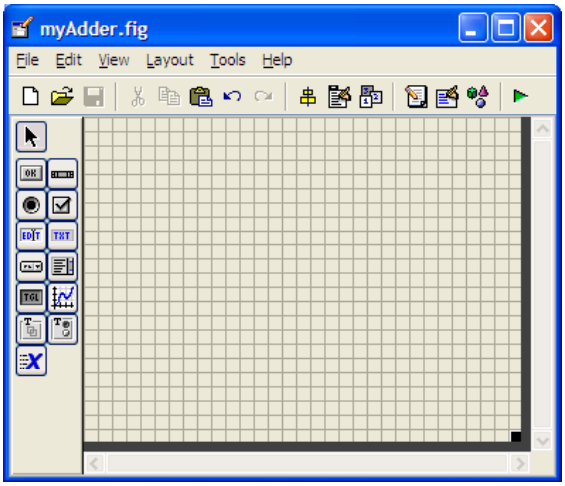
1. Open up MATLAB. Go to the command window and type in guide




2. Choose the first option Blank GUI (Default)



3. You should now see the following screen.



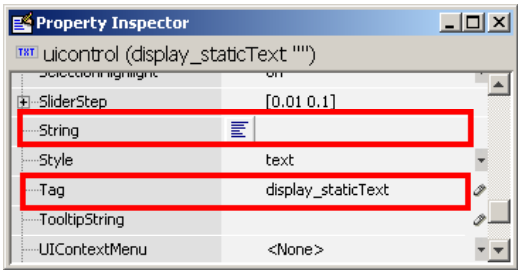
# Part One: The Pushbutton

1.For the adder GUI, we will need the following components.

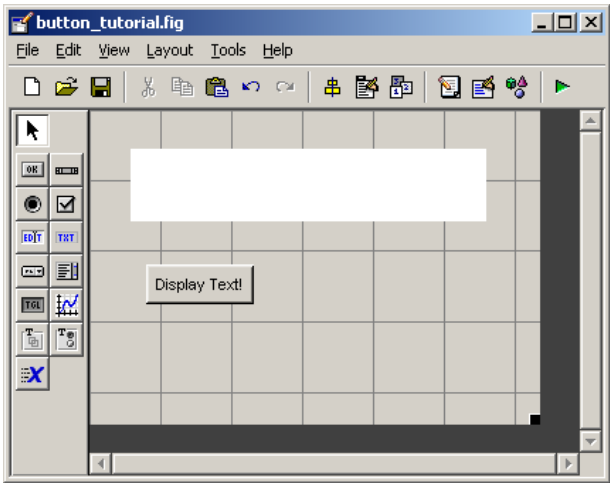

add one *Static Text* component to the GUI figure.


Add three *Pushbutton* components onto the GUI figure.

2. Double click the *Static Text* component to bring up the Property Inspector.
Change the *String* property so that there is nothing inside.
Change the *Tag* property to display_staticText.
Double click on the *Pushbutton* component and
change the *String* property to Display Text! and
change the *Tag* property to displayText_pushbutton.



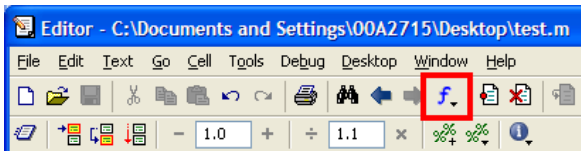
3. The figure should look like below after you add the components and modify them.



# Writing the Code for the GUI Callbacks

1. Open up the .m file that was automatically generated when you saved your GUI.

2. In the MATLAB editor, click on the icon, which will bring up a list of the functions within the .m file.
   Select *displayText_pushbutton_Callback*.

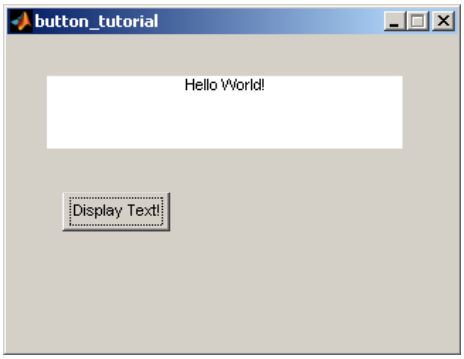


3. Add the following code to the function:

```
%display "Hello Wordl!" in the static text component when the
%pushbutton is pressed
set(handles.display_staticText,'String','Hello World!');
```

# Run and Test the GUI

1. From the m-file editor, you can click on the icon  to save and run the GUI.
2. Alternatively, from the GUIDE editor, you can click on the  to launch the GUI.

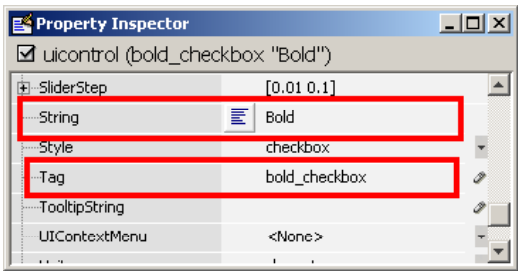


# Part Two: The Check Box

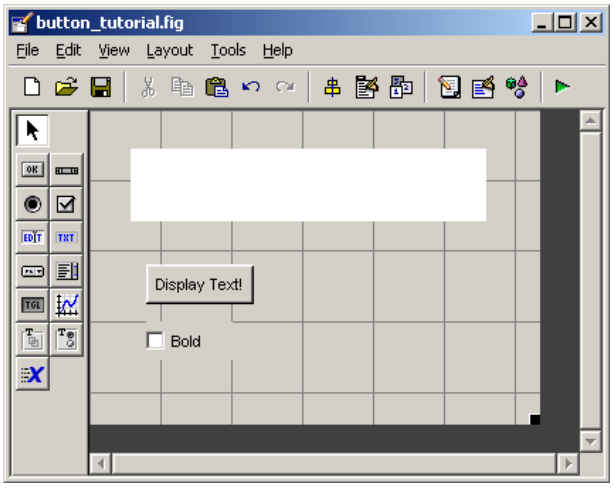
1.For the adder GUI, we will need the following components.

add one *Check Box* component to the GUI figure.

2. Double click the *Check Box* component to bring up the Property Inspector.
Change the *String* property to Bold.
Change the *Tag* property to bold_checkbox.



3. The figure should look like below after you add the *Check Box* component and modify it.

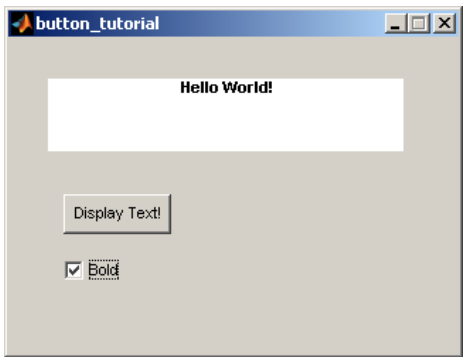


3. Add the following code to the *bold_checkbox_Callback* function:

```
%checkboxStatus = 0, if the box is unchecked,
%checkboxStatus = 1, if the box is checked
checkboxStatus = get(handles.bold_checkbox,'Value');
if(checkboxStatus)
    %if box is checked, text is set to bold
    set(handles.display_staticText,'FontWeight' , 'bold');
else
    %if box is unchecked, text is set to normal
    set(handles.display_staticText,'FontWeight', 'normal');
end
```

# Run and Test the GUI

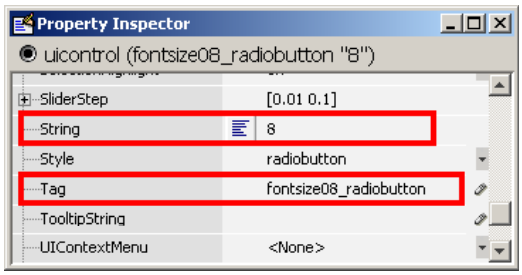
1. Run the GUI to make sure it works before we move on.
   Try checking and unchecking the *Check Box* component to make sure that the text "Hello World!" is being bolded and unbolded.



## Part Three: Radio Buttons, Toggle Buttons, and Button Group Panel

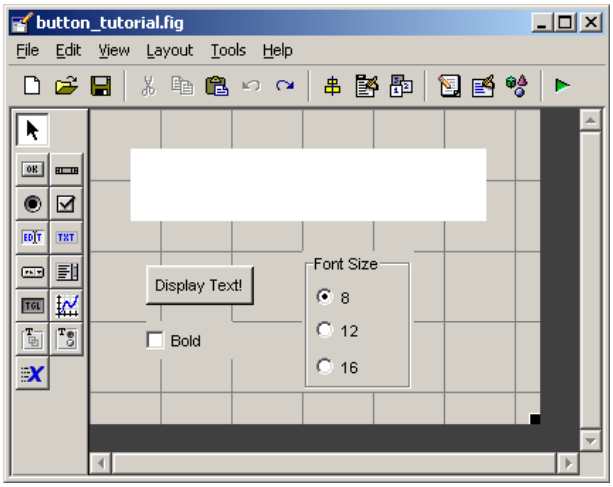
1. Closed GUIDE, reopen it again.
   
   add one *Button Panel* component to the GUI figure.
   
   Add three radio buttons onto the button group panel.

2. Double click on the first *Radio Button* component to bring up the Property Inspector.
   Change the *String* property to 8.
   Change the *Tag* property to fontsize08_radiobutton.

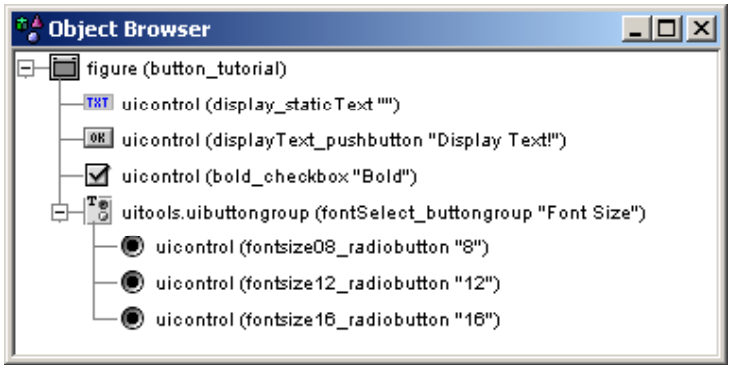


3. Double click on the second *Radio Button* component, and change the *String* property to 12. Change the *Tag* property to fontsize12_radiobutton.
   Double click on the third *Radio Button* component, and change the *String* property to 16. Change the *Tag* property to fontsize16_radiobutton.
   Double click on the button group panel and change the *Tag* property to fontSelect_buttongroup. Change the *String* property for the button group panel to Fontsize.

Here's what your figure should look like after you add the components and modify them.



3. Check the hierarchical structure of the GUI figure. Click on the  icon and the following should appear:




Make sure that the three radio buttons are one hierarchy below the button group icon.

3. Add the following line of code to the opening function.
   In this tutorial example, it is named *button_tutorial_OpeningFcn* function

```
set(handles.fontSelect_buttongroup,'SelectionChangeFcn', ...
    @fontSelect_buttongroup_SelectionChangeFcn);
```

3. Next, add the following function at the very end of the .m file.

```
function fontSelect_buttongroup_SelectionChangeFcn(hObject, eventdata)

%retrieve GUI data, i.e. the handles structure
handles = guidata(hObject);

switch get(eventdata.NewValue,'Tag')   % Get Tag of selected object
    case 'fontsize08_radiobutton'
      %execute this code when fontsize08_radiobutton is selected
      set(handles.display_staticText,'FontSize',8);

    case 'fontsize12_radiobutton'
      %execute this code when fontsize12_radiobutton is selected
      set(handles.display_staticText,'FontSize',12);

    case 'fontsize16_radiobutton'
      %execute this code when fontsize16_radiobutton is selected
      set(handles.display_staticText,'FontSize',16);
    otherwise
       % Code for when there is no match.

end
```

# Run and Test the GUI

Run the GUI.
Try clicking on all of the buttons to make sure they perform their function correctly.
Specifically, make sure that the font size changes accordingly.