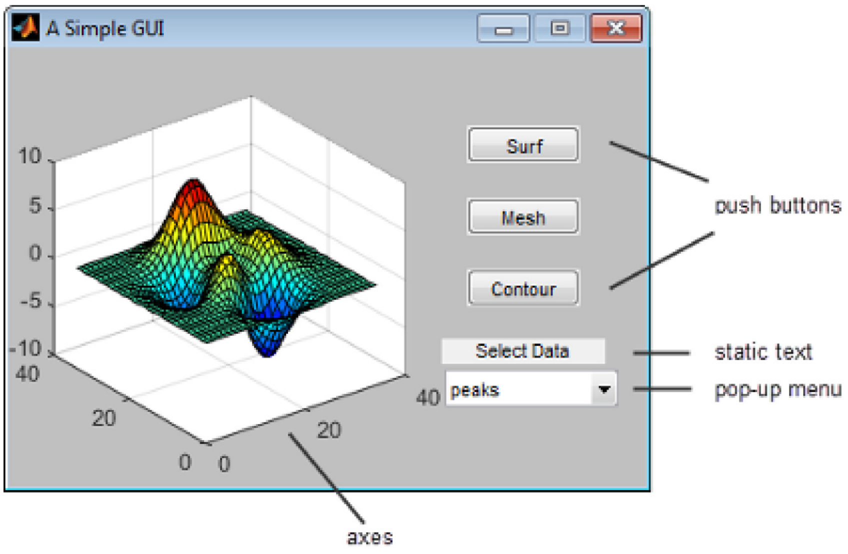
Create a Simple App Using GUIDE

Note: This topic applies to apps you create using GUIDE. For alternative ways to build apps, see Ways to Build Apps.

This example shows how to use GUIDE to create an app that has a simple user interface (UI), such as the one shown here.



Subsequent sections guide you through the process of creating this app.

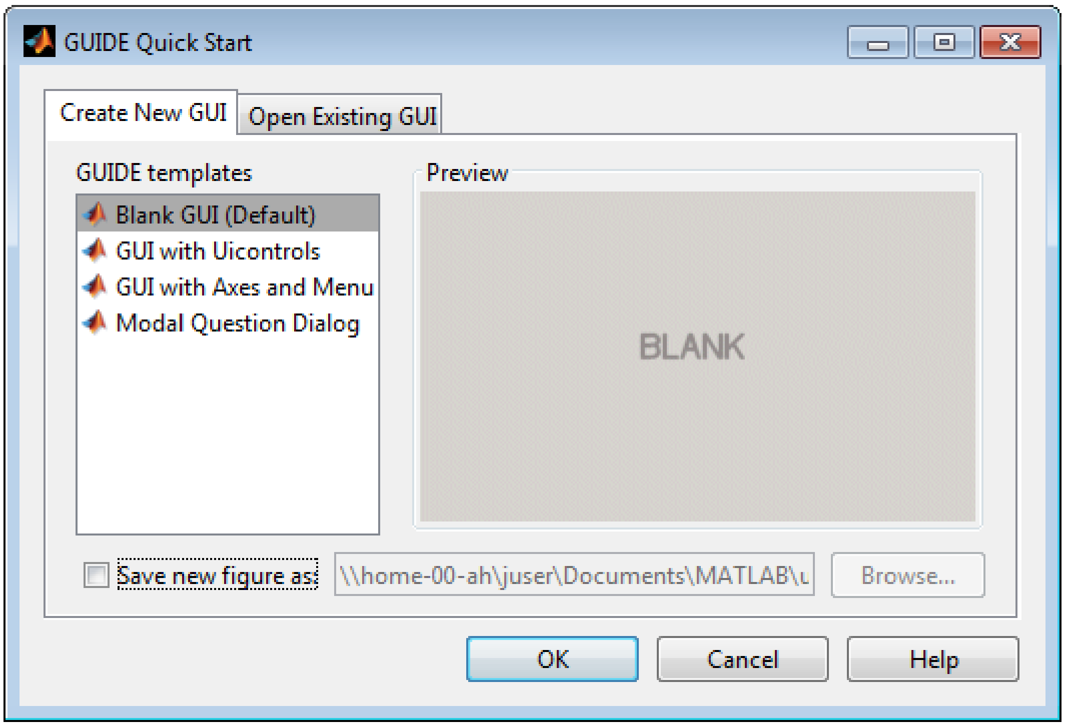
If you only want to view and run the code that created this app, set your current folder to one to which you have write access. Copy the example code and open it in the Editor by issuing the following MATLAB$^{®}$ commands:

```
copyfile(fullfile(docroot, 'techdoc','creating_guis',...
   'examples','simple_gui*.*')),fileattrib('simple_gui*.*', '+w');
guide simple_gui.fig;
edit simple_gui.m
```

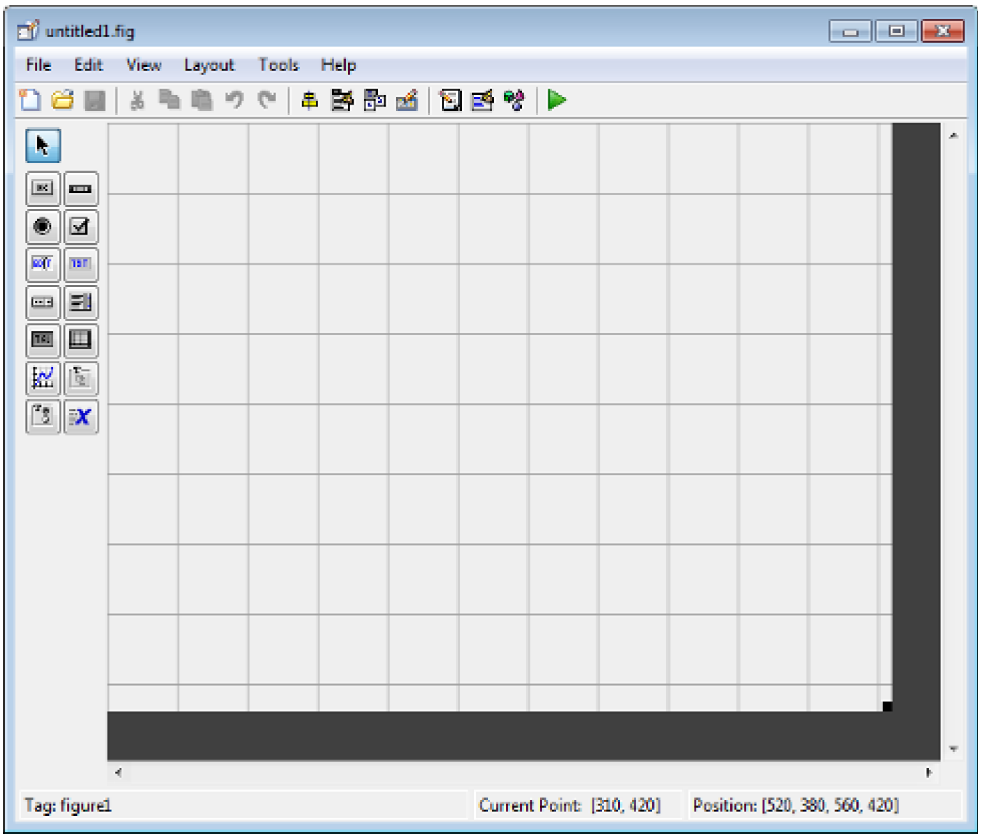
Click the **Run** button to run the app.

Open a New UI in the GUIDE Layout Editor

1. Start GUIDE by typing `guide` at the MATLAB prompt.

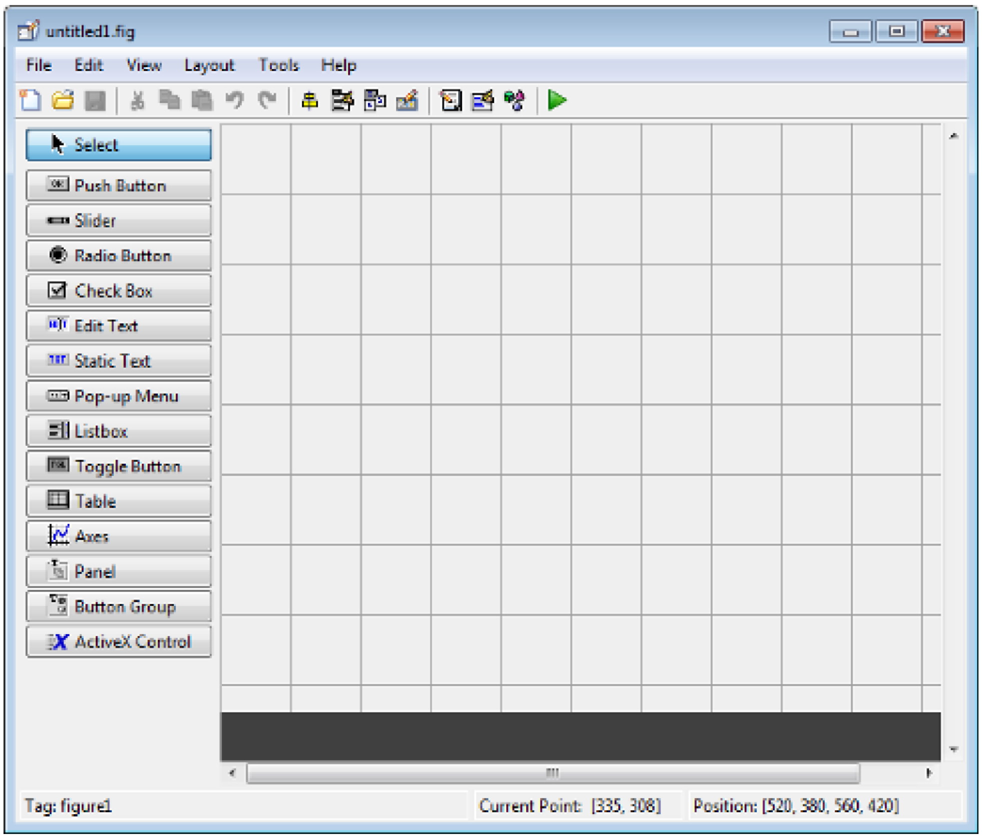


2. In the GUIDE Quick Start dialog box, select the **Blank GUI (Default)** template, and then click **OK**.



Tag: figure1 Current Point: [310, 420] Position: [520, 380, 560, 420]

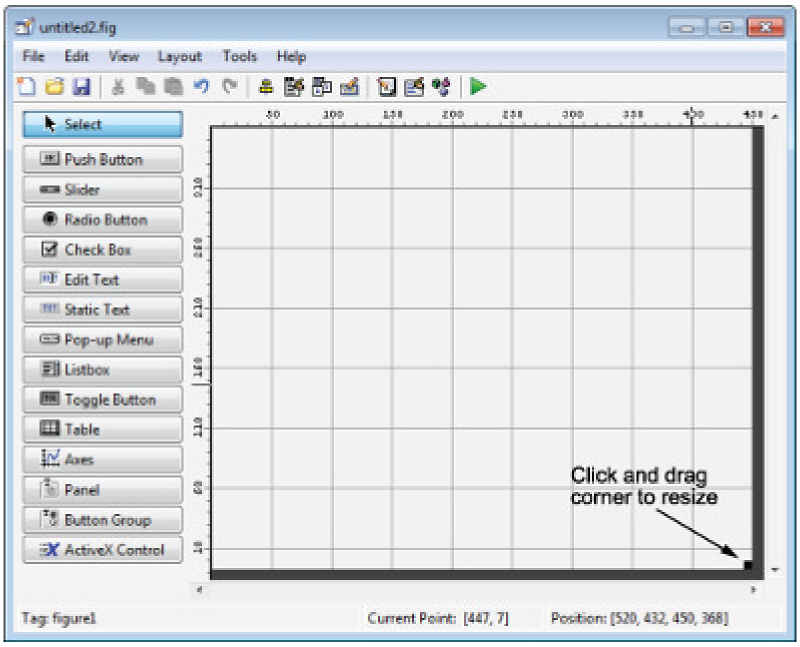
3. Display the names of the components in the component palette:
   a. Select **File** > **Preferences** > **GUIDE**.
   b. Select **Show names in component palette**.
   c. Click **OK**.



Tag: figure1 Current Point: [335, 308] Position: [520, 380, 560, 420]

Set the Window Size in GUIDE

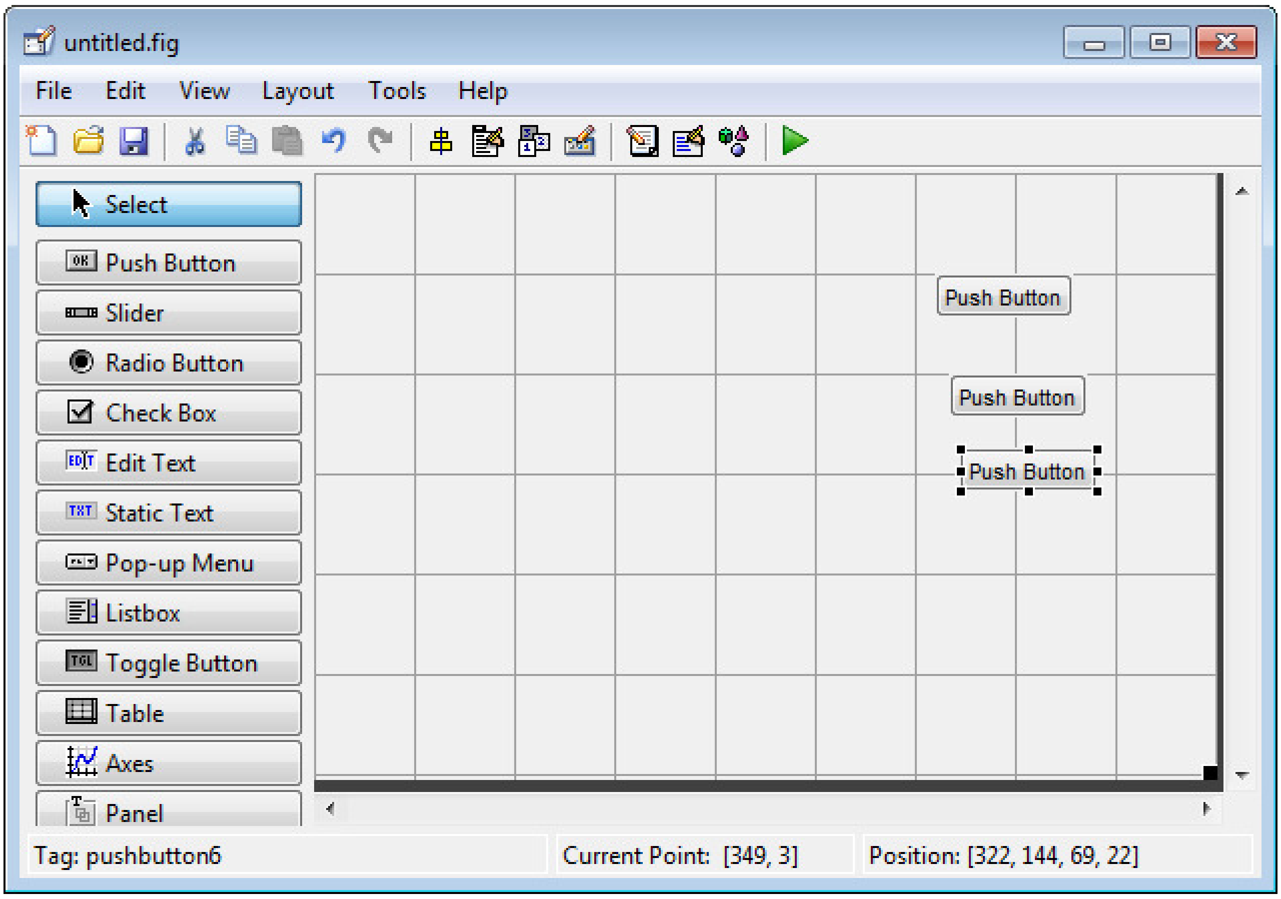
Set the size of the window by resizing the grid area in the Layout Editor. Click the lower-right corner and drag it until the canvas is approximately 3 inches high and 4 inches wide. If necessary, make the canvas larger.



Layout the UI

Add, align, and label the components in the UI.

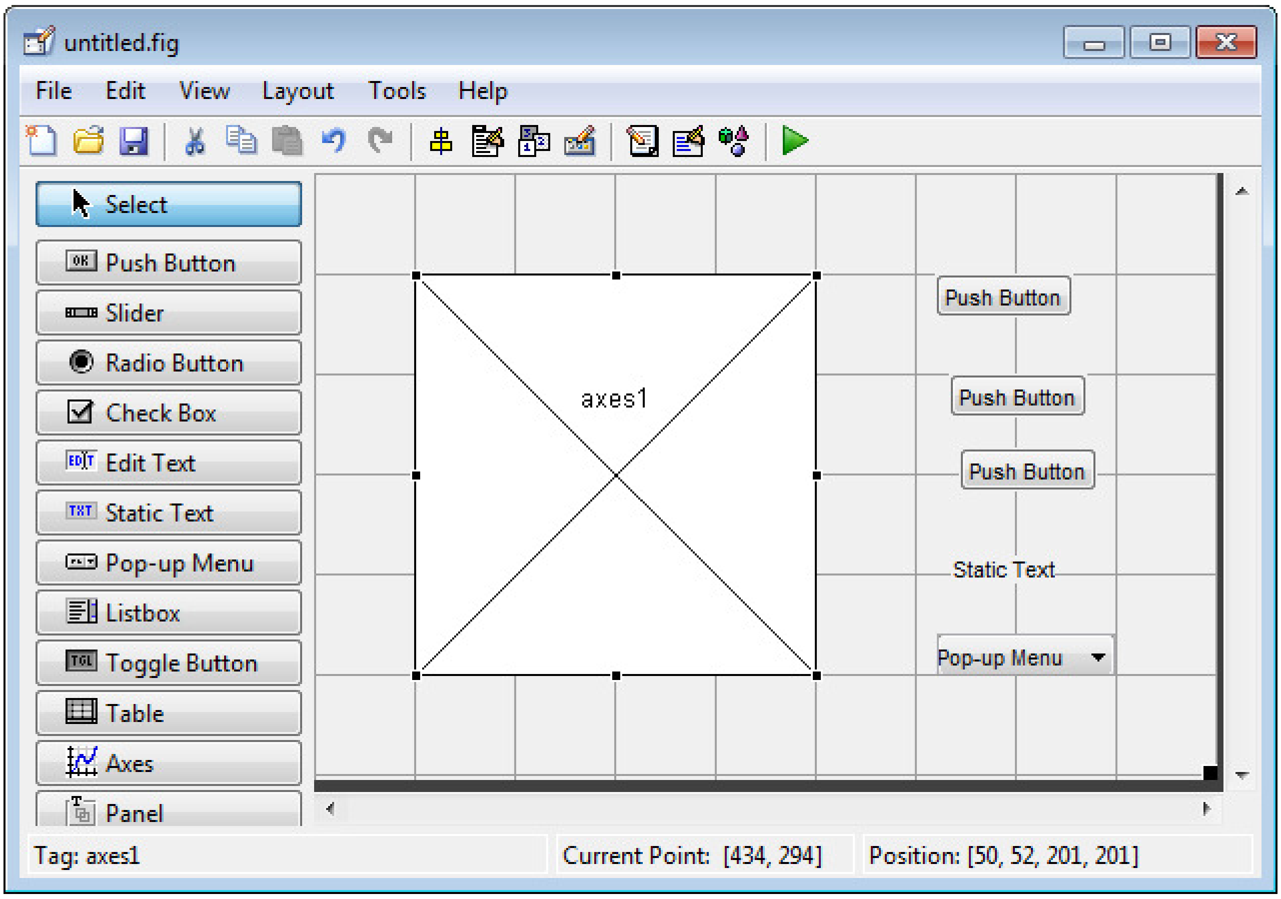
1. Add the three push buttons to the UI. Select the push button tool from the component palette at the left side of the Layout Editor and drag it into the layout area. Create three buttons, positioning them approximately as shown in the following figure.



2. Add the remaining components to the UI.

   - A static text area
   - A pop-up menu
   - An axes

   Arrange the components as shown in the following figure. Resize the axes component to approximately 2-by-2 inches.

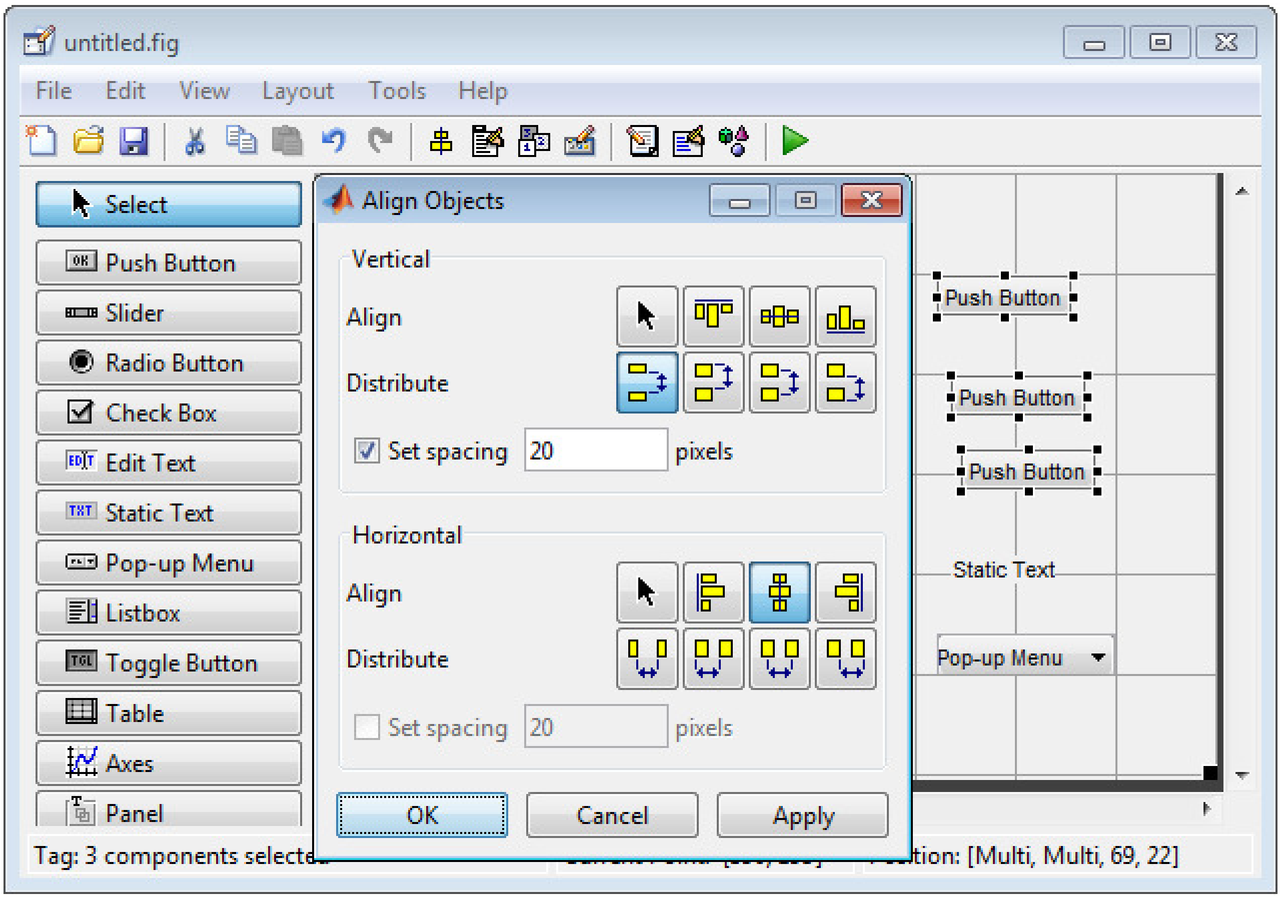


Align the Components

If several components have the same parent, you can use the Alignment Tool to align them to one another. To align the three push buttons:

1. Select all three push buttons by pressing **Ctrl** and clicking them.
2. Select **Tools** > **Align Objects**.
3. Make these settings in the Alignment Tool:

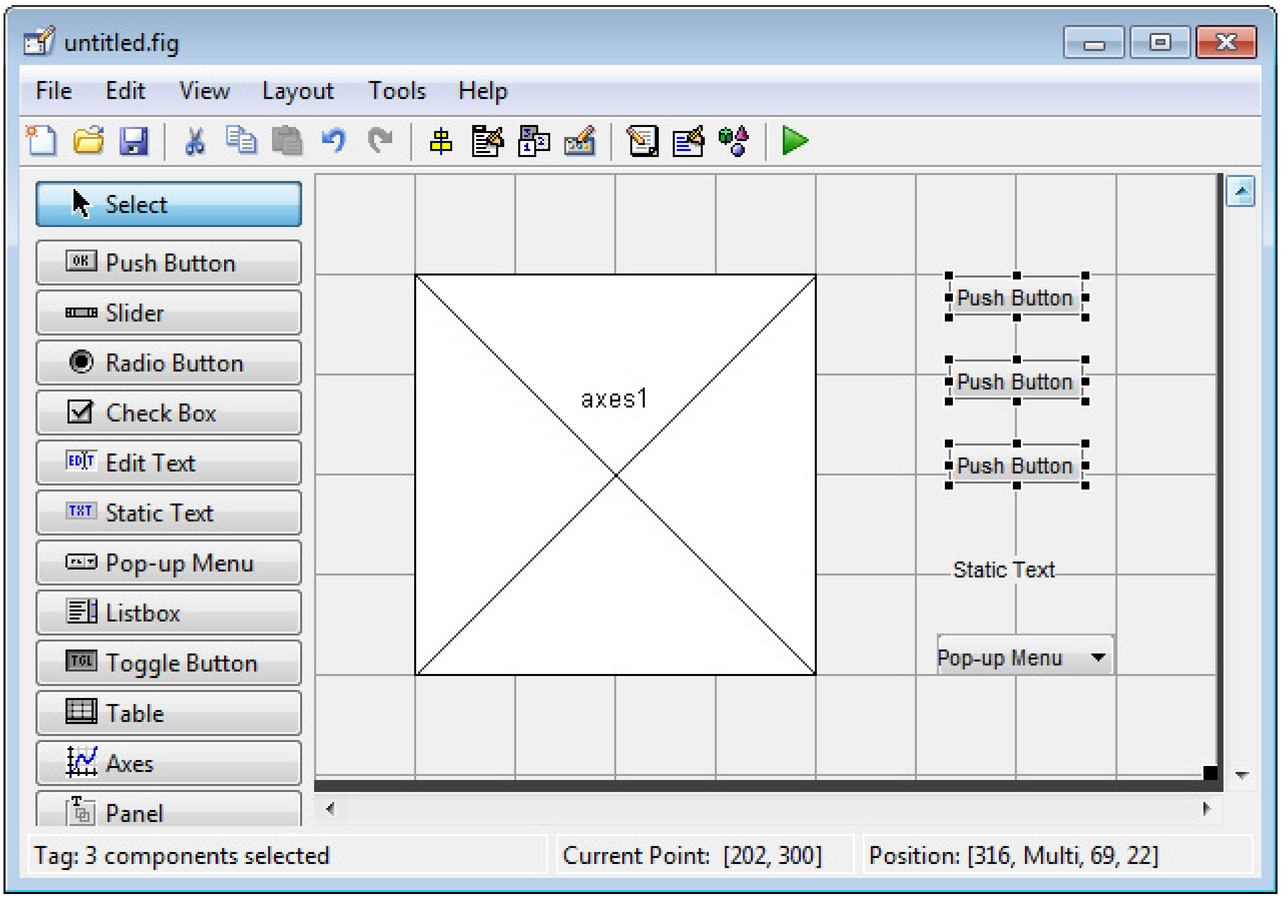
- Left-aligned in the horizontal direction.
- 20 pixels spacing between push buttons in the vertical direction.



Tag: 3 components selected

Position: [Multi, Multi, 69, 22]

4. Click **OK**.



Tag: 3 components selected

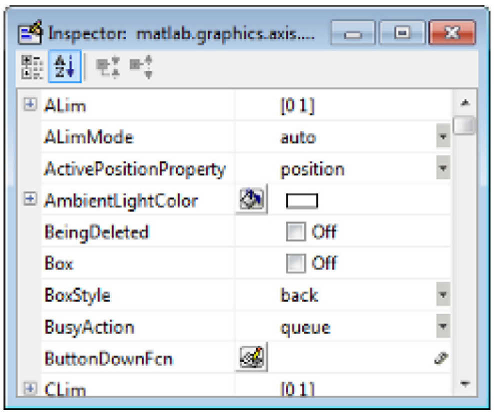
Current Point: [202, 300]

Position: [316, Multi, 69, 22]

Label the Push Buttons

Each of the three push buttons specifies a plot type: surf, mesh, and contour. This section shows you how to label the buttons with those options.

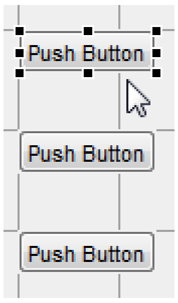
1. Select **View** > **Property Inspector**.



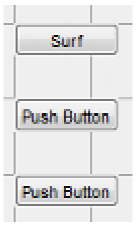
Inspector: matlab.graphics.axis....

ALim	[0 1]
ALimMode	auto
ActivePositionProperty	position
AmbientLightColor	▭
BeingDeleted	☐ Off
Box	☐ Off
BoxStyle	back
BusyAction	queue
ButtonDownFcn	
CLim	[0 1]

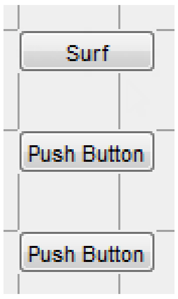
2. In the layout area, click the top push button.



3. In the Property Inspector, select the `String` property, and then replace the existing value with the word `Surf`.



4. Press the **Enter** key. The push button label changes to **Surf**.

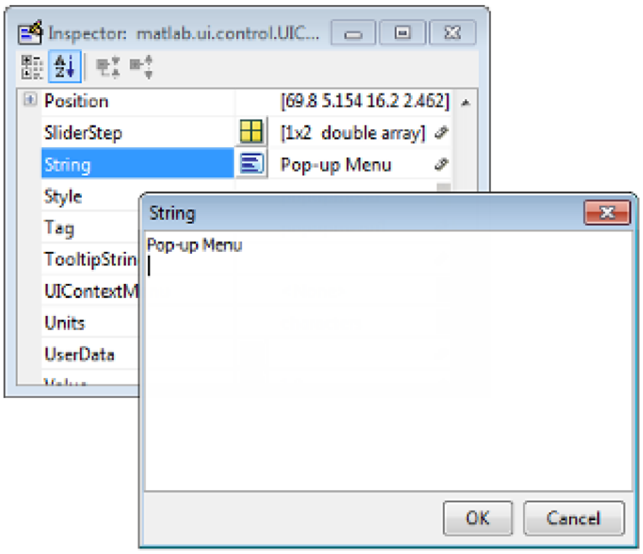


5. Click each of the remaining push buttons in turn and repeat steps 3 and 4. Label the middle push button **Mesh**, and the bottom button **Contour**.

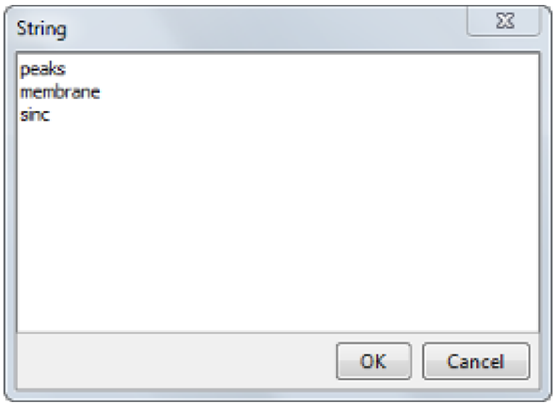
List Pop-Up Menu Items

The pop-up menu provides a choice of three data sets: peaks, membrane, and sinc. These data sets correspond to MATLAB functions of the same name. This section shows you how to list those data sets as choices in the pop-menu.

1. In the layout area, click the pop-up menu.
2. In the Property Inspector, click the button next to `String`. The String dialog box displays.

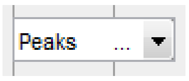


3. Replace the existing text with the names of the three data sets: peaks, membrane, and sinc. Press **Enter** to move to the next line.



4. When you finish editing the items, click **OK**.

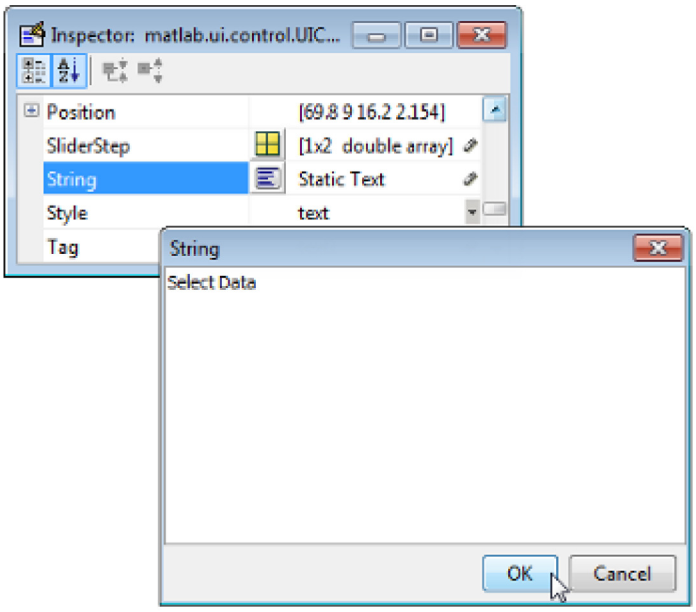
   The first item in your list, `peaks`, appears in the pop-up menu in the layout area.



Modify the Static Text

In this UI, the static text serves as a label for the pop-up menu. This section shows you how to change the static text to read `Select Data`.

1. In the layout area, click the static text.
2. In the Property Inspector, click the button next to `String`. In the String dialog box that displays, replace the existing text with the phrase `Select Data`.

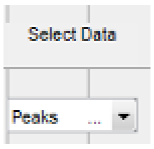


Position [69.8 9 16.2 2.154]
SliderStep [1x2 double array]
String Static Text
Style text
Tag

Select Data

3. Click **OK**.

   The phrase `Select Data` appears in the static text component above the pop-up menu.



Save the Layout

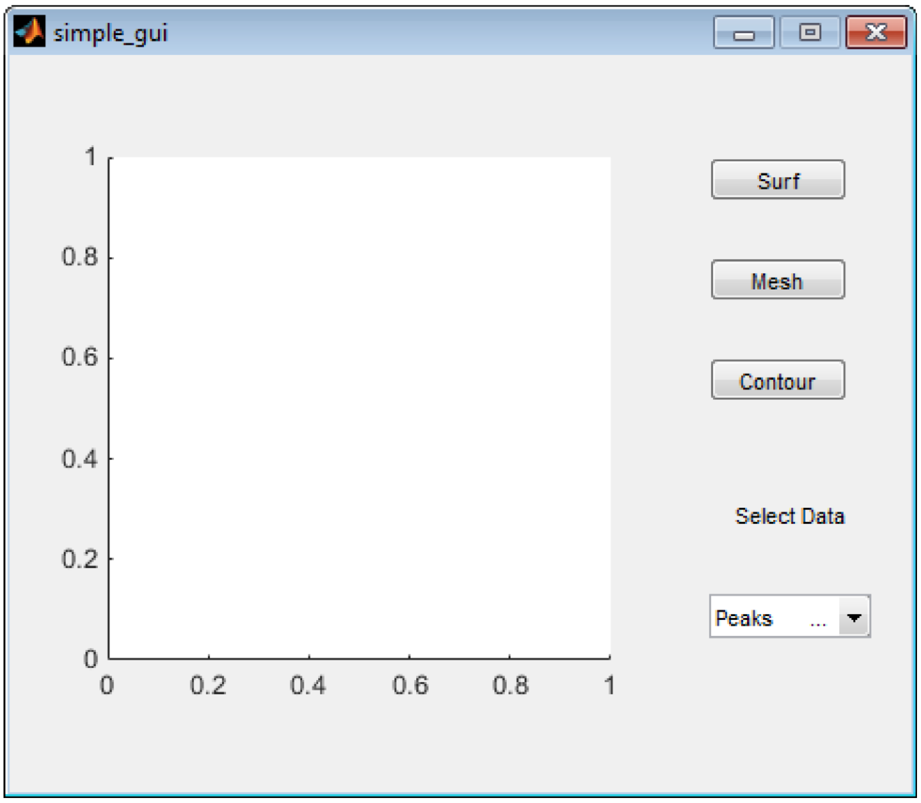
When you save a layout, GUIDE creates two files, a FIG-file and a code file. The FIG-file, with extension `.fig`, is a binary file that contains a description of the layout. The code file, with extension `.m`, contains MATLAB functions that control the app's behavior.

1. Save and run your program by selecting **Tools** > **Run**.
2. GUIDE displays a dialog box displaying: "Activating will save changes to your figure file and MATLAB code. Do you wish to continue?

   Click **Yes**.
3. GUIDE opens a **Save As** dialog box in your current folder and prompts you for a FIG-file name.
4. Browse to any folder for which you have write privileges, and then enter the file name `simple_gui` for the FIG-file. GUIDE saves both the FIG-file and the code file using this name.
5. If the folder in which you save the files is not on the MATLAB path, GUIDE opens a dialog box that allows you to change the current folder.
6. GUIDE saves the files `simple_gui.fig` and `simple_gui.m`, and then runs the program. It also opens the code file in your default editor.

   The app opens in a new window. Notice that the window lacks the standard menu bar and toolbar that MATLAB figure windows display. You can add your own menus and toolbar buttons with GUIDE, but by default a GUIDE app includes none of these components.

   When you run `simple_gui`, you can select a data set in the pop-up menu and click the push buttons, but nothing happens. This is because the code file contains no statements to service the pop-up menu and the buttons.



To run an app created with GUIDE without opening GUIDE, execute its code file by typing its name.

```
simple_gui
```

You can also use the `run` command with the code file, for example,

```
run simple_gui
```

Note: Do not attempt to run your app by opening its FIG-file outside of GUIDE. If you do so, the figure opens and appears ready to use, but the UI does not initialize and the callbacks do not function.

Code the Behavior of the App

When you saved your layout in the previous section, Save the Layout, GUIDE created two files: a FIG-file, `simple_gui.fig`, and a program file, `simple_gui.m`. However, the app is not responsive because `simple_gui.m` does not contain any statements that perform actions. This section shows you how to add code to the file to make the app functional.

Generate Data to Plot

This section shows you how to generate the data to be plotted when the user clicks a button. The *opening function* generates this data by calling MATLAB functions. The opening function initializes the UI when it opens, and it is the first callback in every GUIDE-generated code file.

In this example, you add code that creates three data sets to the opening function. The code uses the MATLAB functions `peaks`, `membrane`, and `sinc`.

1. Display the opening function in the MATLAB Editor.

   If the file `simple_gui.m` is not already open in the editor, open from the Layout Editor by selecting **View** > **Editor**.
2. On the **EDITOR** tab, in the **NAVIGATE** section, click **Go To**, and then select `simple_gui_OpeningFcn`.

   The cursor moves to the opening function, which contains this code:

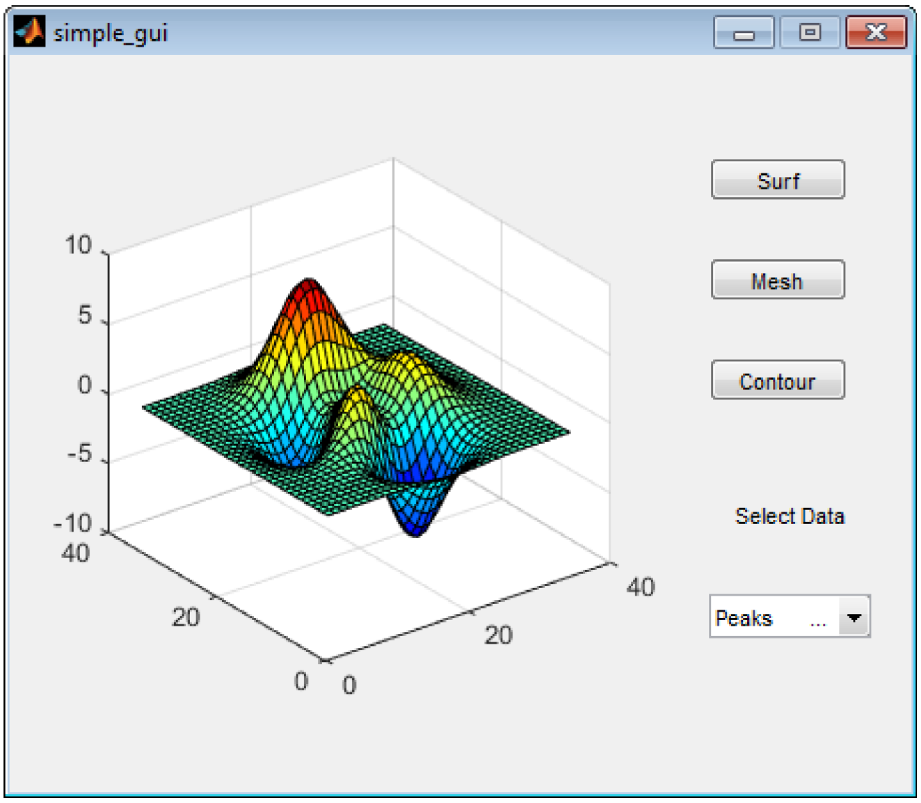
```
% --- Executes just before simple_gis made visible.
function simple_gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to simple_g(see VARARGIN)

% Choose default command line output for simple_gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes simple_gwait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

3. Create data to plot by adding the following code to the opening function immediately after the comment that begins `% varargin...`

```
% Create the data to plot.
handles.peaks=peaks(35);
handles.membrane=membrane;
[x,y] = meshgrid(-8:.5:8);
r = sqrt(x.^2+y.^2) + eps;
sinc = sin(r)./r;
handles.sinc = sinc;
% Set the current data value.
handles.current_data = handles.peaks;
surf(handles.current_data)
```

The first six executable lines create the data using the MATLAB functions `peaks`, `membrane`, and `sinc`. They store the data in the `handles` structure, an argument provided to all callbacks. Callbacks for the push buttons can retrieve the data from the `handles` structure.

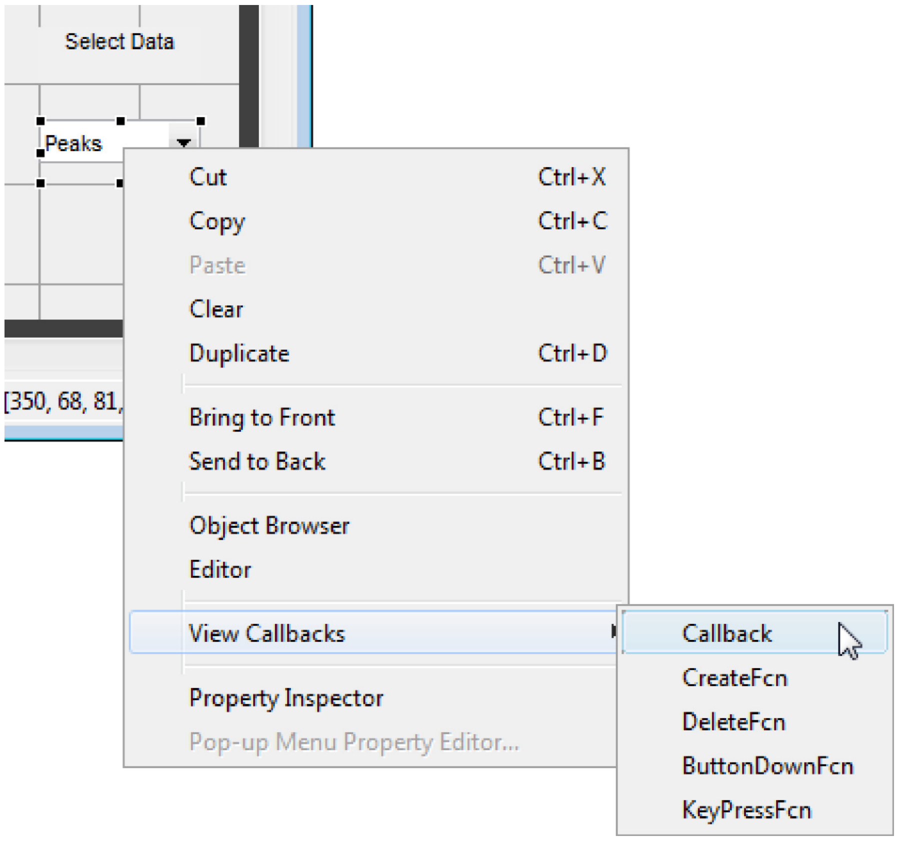
The last two lines create a current data value and set it to peaks, and then display the surf plot for peaks. The following figure shows how the app looks when it first displays.



Code Pop-Up Menu Behavior

The pop-up menu presents options for plotting the data. When the user selects one of the three plots, MATLAB software sets the pop-up menu `Value` property to the index of the selected menu item. The pop-up menu callback reads the pop-up menu `Value` property to determine the item that the menu currently displays, and sets `handles.current_data` accordingly.

1. Display the pop-up menu callback in the MATLAB Editor. In the GUIDE Layout Editor, right-click the pop-up menu component, and then select **View Callbacks** > **Callback**.

   GUIDE displays the code file in the Editor, and moves the cursor to the pop-menu callback, which contains this code:

   ```
   % --- Executes on selection change in popupmenu1.
   function popupmenu1_Callback(hObject, eventdata, handles)
   % hObject    handle to popupmenu1 (see GCBO)
   % eventdata  reserved - to be defined in a future version of MATLAB
   % handles    structure with handles and user data (see GUIDATA)
   ```

2. Add the following code to the `popupmenu1_Callback` after the comment that begins `% handles...`

   This code first retrieves two pop-up menu properties:

   - `String` — a cell array that contains the menu contents
   - `Value` — the index into the menu contents of the selected data set

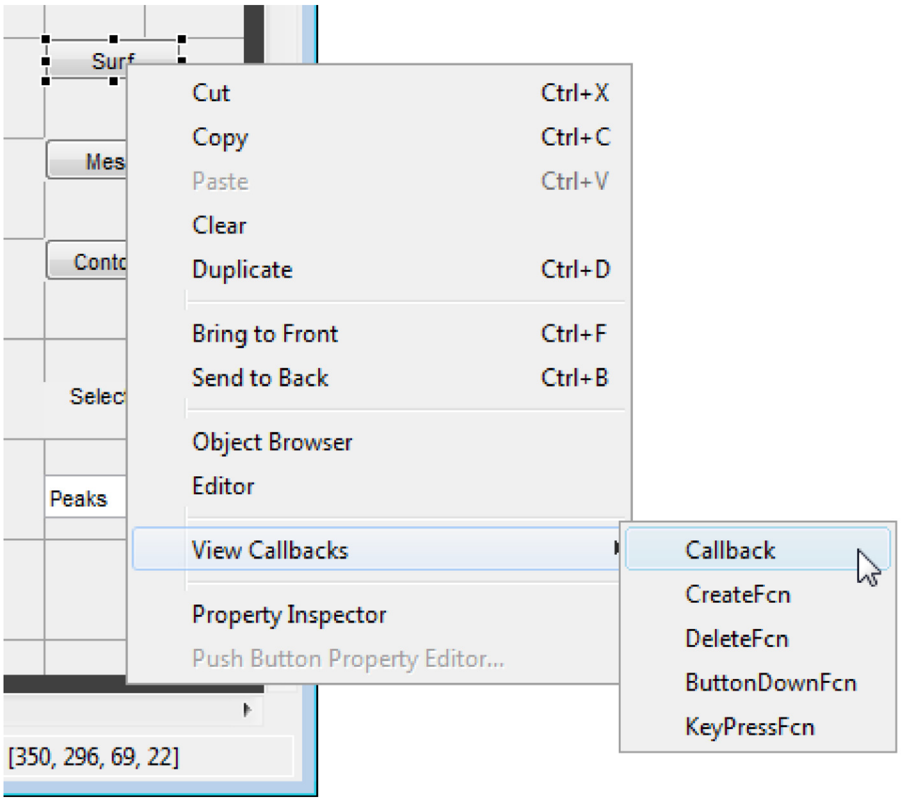
   The code then uses a `switch` statement to make the selected data set the current data. The last statement saves the changes to the `handles` structure.

```
% Determine the selected data set.
str = get(hObject, 'String');
val = get(hObject,'Value');
% Set current data to the selected data set.
switch str{val};
case 'peaks' % User selects peaks.
   handles.current_data = handles.peaks;
case 'membrane' % User selects membrane.
   handles.current_data = handles.membrane;
case 'sinc' % User selects sinc.
   handles.current_data = handles.sinc;
end
% Save the handles structure.
guidata(hObject,handles)
```

Code Push Button Behavior

Each of the push buttons creates a different type of plot using the data specified by the current selection in the pop-up menu. The push button callbacks get data from the `handles` structure and then plot it.

1. Display the **Surf** push button callback in the MATLAB Editor. In the Layout Editor, right-click the **Surf** push button, and then select **View Callbacks** > **Callback**.

   In the Editor, the cursor moves to the **Surf** push button callback in the code file, which contains this code:

   ```
   % --- 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 MATLAB
   % handles    structure with handles and user data (see GUIDATA)
   ```

2. Add the following code to the callback immediately after the comment that begins `% handles...`

   ```
   % Display surf plot of the currently selected data.
   surf(handles.current_data);
   ```

3. Repeat steps 1 and 2 to add similar code to the **Mesh** and **Contour** push button callbacks.

   - Add this code to the **Mesh** push button callback, `pushbutton2_Callback`:

```
% Display mesh plot of the currently selected data.
mesh(handles.current_data);
```

- Add this code to the **Contour** push button callback, `pushbutton3_Callback`:

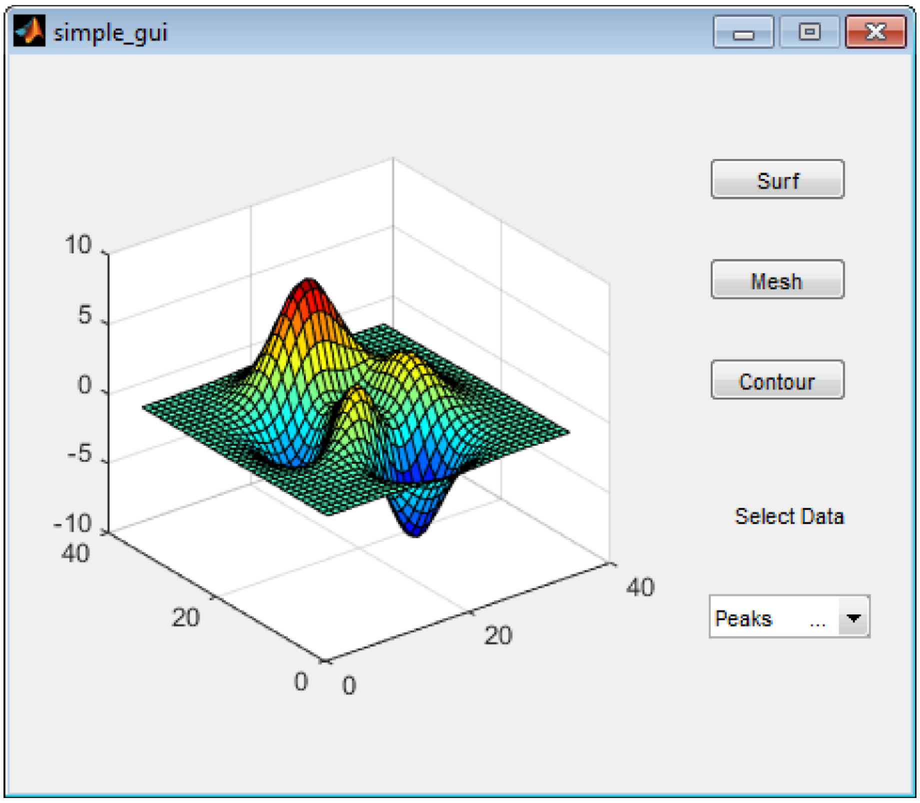
```
% Display contour plot of the currently selected data.
contour(handles.current_data);
```

4. Save your code by selecting **File** > **Save**.

Run the App

In Code the Behavior of the App, you programmed the pop-up menu and the push buttons. You also created data for them to use and initialized the display. Now you can run your program to see how it works.

1. Run your program from the Layout Editor by selecting **Tools** > **Run**.



2. In the pop-up menu, select **Membrane**, and then click the **Mesh** button. The app displays a mesh plot of the MathWorks® L-shaped Membrane logo.
3. Try other combinations before closing the window.

Related Topics

- Ways to Build Apps
- Create a Simple App Programmatically
- Create a Simple App Using App Designer