MATLAB GUIs

We conclude our brief overview of MATLAB by looking at :

- Brief introduction to MATLAB GUI building.




Back
Close

MATLAB GUIs

Building a GUI in MATLAB is pretty straight forward and quick.

- You can create a GUI by hand.
- Use MATLAB's GUI Development Environment (GUIDE) to assist you

Predefined GUI Dialog Boxes

MATLAB Provides a variety of dialog boxes that are ready made for you to use:

Simple uicontrol objects : **errordlg** , **helpdlg**, **msgbox**, **warndlg** , **inputdlg** and **questdlg** — pretty self explanatory.

File/Directory Chooser : **uigetfile**

Font and Colour Choosers : **uisetfont** and **uisetcolor**



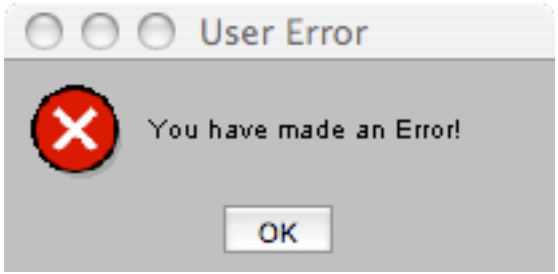

Back
Close

The Error Dialog box: `errordlg`

To create an error dialog you do something like this:

```
errfig = errordlg('You have made an Error!','User Error',...
     'replace');
```

This creates:



Note:

- The first string specifies the main error dialog text.
- The second string specifies the dialog window title text.
- The third string specifies as **CREATEMODE** which when set to `'replace'` forces MATLAB to use only one error window with the same title — do not create another one if exists.

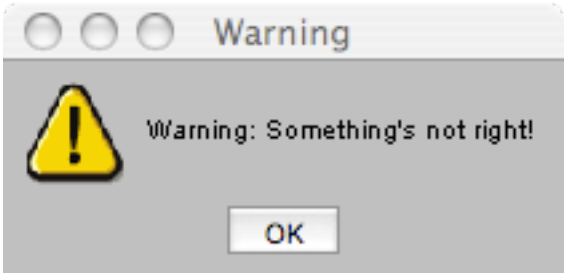



The Warning Dialog box: `warndlg`

To create a warning dialog you do something like this:

```
warnfig = warndlg('Warning: Something''s not right!', 'Warning');
```

This creates:



Note:

- The first string specifies the main error dialog text.
- The second string specifies the dialog window title text
- **Use** `''` to get a `'` character in a string

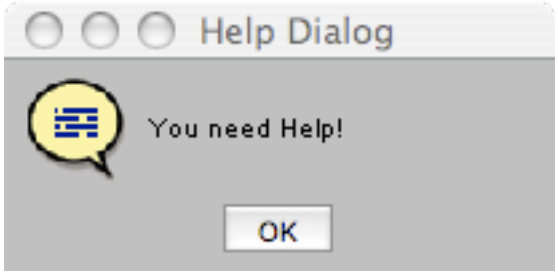



The Help Dialog box: `helpdlg`

To create a help dialog you do something like this:

```
helpfig = helpdlg('You need Help!');
```

This creates:



Note:

- The string specifies the main error dialog text.
- An optional second string could specify the dialog window title text — often unnecessary.




The Message Dialog box: `msgbox`

Error, Warning and Help dialogs are all special cases of a **msgbox**, *E.g.*:

```
errfig = msgbox('You have made an Error!','User Error','error');
warnfig = msgbox('Warning: Something''s not right!', 'Warning', ...
'warn');
helpfig = msgbox('You need Help!','Help Dialog','help')
```

All achieve same as above.

It is more general and can just create a general message:

```
msgfig = msgbox('This is a Message','Msg');
```

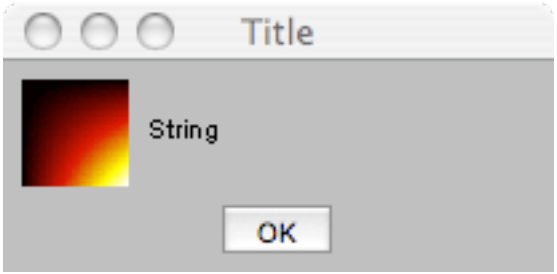
Customised Message Dialog Icons

You can even be used to create a message with a customised icon with the format:

```
msgbox(Message,Title,'custom',IconData,IconCMap)
```

E.g.:

```
Data=1:64;Data=(Data'*Data)/64;
msgfig =msgbox('String','Title','custom',Data,hot(64));
```

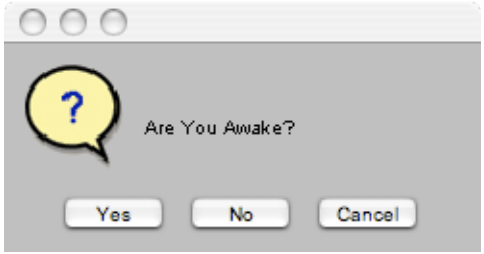


The Question Dialog Box: `questdlg`

To create a question dialog you do something like this:

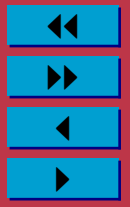
```
ret_string = questdlg('Are You Awake?');
```

This creates:



Note:

- The string specifies the main question dialog text.
- The **questdlg** is **modal** — MATLAB always waits for a response.
  - *Note*: **msgbox** dialogs can also be set to be `modal`/`non-modal` as well as `replace` (`non-modal` is the default behaviour)
- `ret_string` stores the text for the reply: `'yes'`,`'no'` or `'cancel'` in this case.



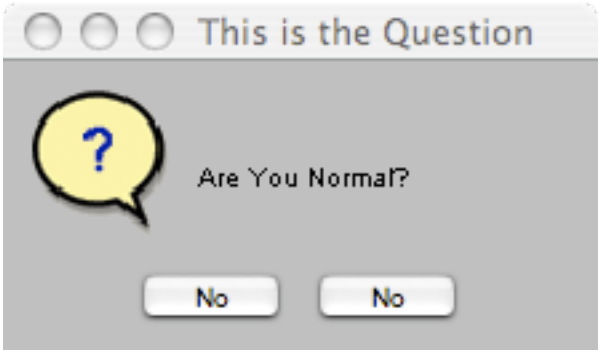
Customising The Question Dialog Box

The general form of the **questdlg** is:

```
ret_string = questdlg(QuestionString, ....
                      WindowTitleString, ...
                      Button_1_String,...
                      Button_2_String,...
                      Button_3_String,...
                      DefaultString);
```

For example:

```
ret_string = questdlg('Are You Normal?','This is the Question',...
                      'No','No','No');
```

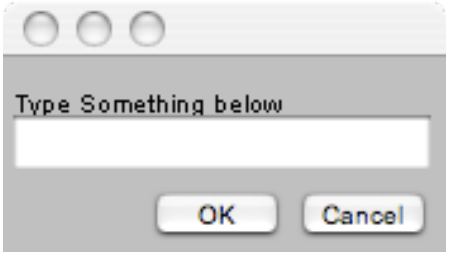


The Input Dialog Box: `inputdlg`

To create a input dialog you do something like this:

```
Answers = inputdlg('Type Something below');
```

This creates:



Note:

- The string specifies the main input dialog text.
- Answer stores the returned string
- If more than one input and array of strings returned
- This dialog is also **modal**
- Default answers maybe supplied — see `help inputdlg`




Multiple Input Dialogs

To create multiple inputs you do something like this:

```
Answers = inputdlg({'Q1: What Your Name?',...
                    'Q2: What is your Address?',
                    'Q3: What is your age?'},
                    'Questionnaire', [1 3 1]);
```

Note:

- A **cell array** (denoted by `{...}`) of strings specifies the set of questions
- Respective window sizes can be set with an array: `[1 3 1]`
- Answer stores the returned array of strings, e.g.

```
Answers =
    'Yukun'
    'COMSC'
    '??'
```

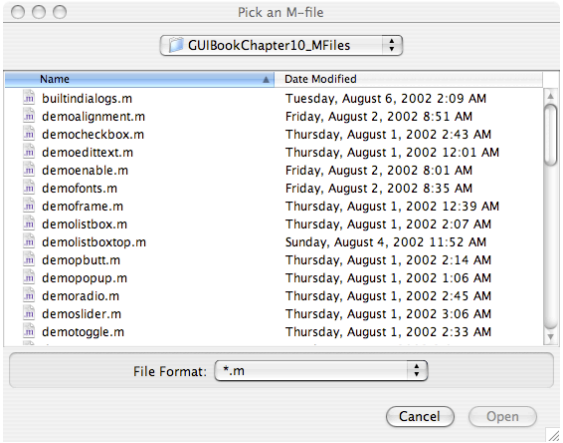
The File/Directory Selection Dialog Boxes

To create a input dialog you do something like this:

```
[filename, pathname] = uigetfile('*.m', 'Pick an M-file');
```

Note:

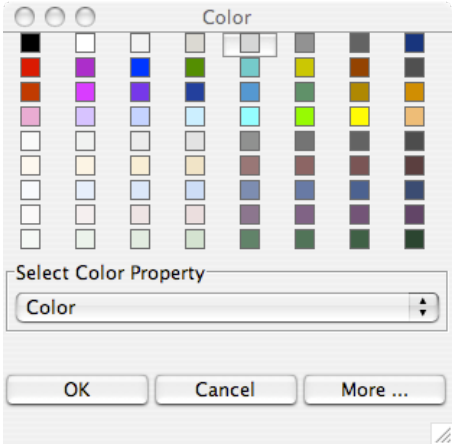
- The first string specifies a **file filter**
- The second string is the window title.
- `filename` and `pathname` store the returned respective values of the selected file
- More options — see `help uigetfile`
- **uiputfile** similar — see `help uiputfile`

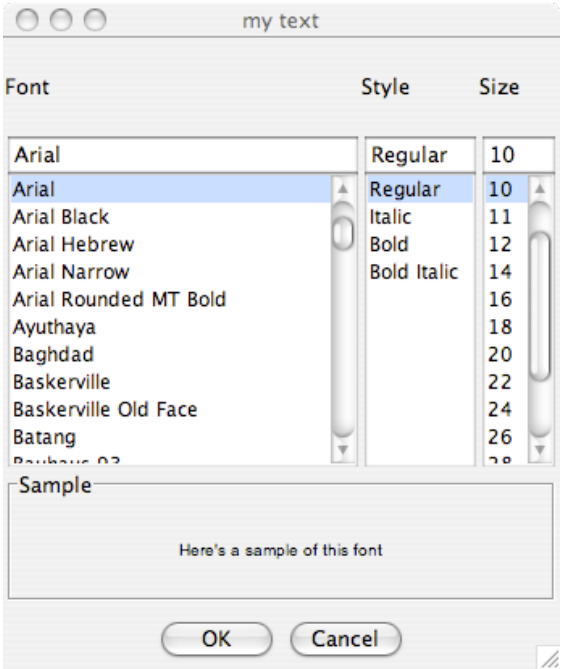


Setting Fonts and Colours

`uisetfont` and `uisetcolor` can be used to set properties of respective text and graphics objects. *E.g*:

```
myfig = figure(1);
xlbl = xlabel('x-axis');
uisetfont(xlbl, 'my text');
uisetcolor(myfig);
```

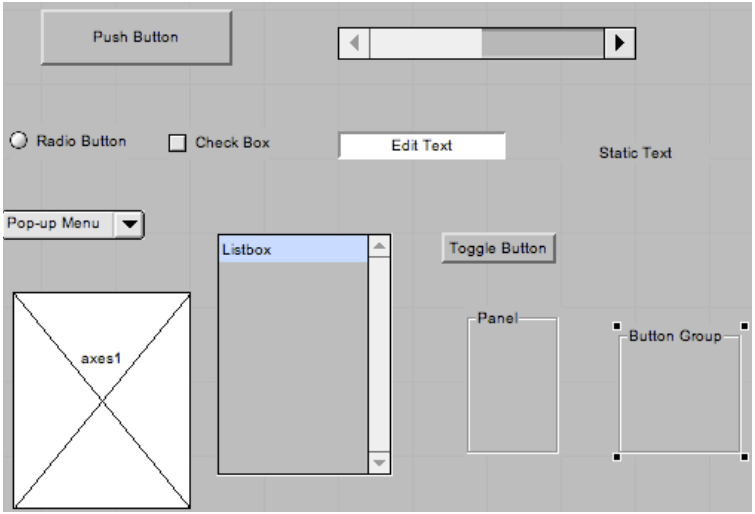




Uicontrol Elements

MATLAB provides a number basic GUI elements:

- Check boxes
- Editable text fields
- Frames
- List boxes
- Pop-up menus
- Push buttons
- Radio buttons
- Sliders
- Static text labels
- Toggle buttons



Manually Creating Uicontrol Elements

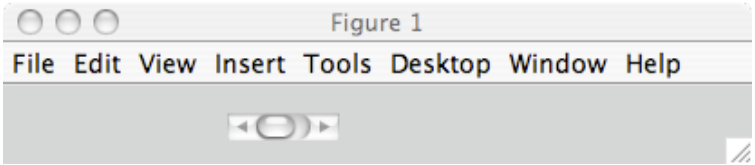
To create a `uicontrol` element, use the MATLAB command:

```
handle = uicontrol('Property1Name', Property1Value, ...
                   Property2Name', Property2Value, ...
                   .
                   .
                   );
```

- The first property name usually sets the style: Check box, slider, *etc.*
- Others specify attributes of that object.
- Simple Example:

```
h_slider = uicontrol('Style','slider',...
'units','normalized',...
'position',[.3 .6 .15 .05]);
```

- Use `doc uicontrol` and links to find detailed Uicontrol Properties



Uicontrol Callbacks

Having created a UI element such as a slider, we need to attach a callback to the element:

- Simply set the `'callback'` property value with an appropriate MATLAB function, e.g.

```
h_slider = uicontrol(h_fig,...
'callback','slidergui(''Slider Moved'');',...
```

- Callback can be a *self-referenced* function (as in example below) or an entirely new function (see GUIDE example later).
- Within the callback, you need to access the value of the Uicontrol element:
  - Store data in graphics handle `'userdata'`:

```
set(h_fig,'userdata', h_slider);
```

  - Retrieve values via a few `get`s:

```
h_slider = get(gcf,'userdata');
value = get(h_slider,'value');
```





Full Slide Callback Code Example

```
function slidergui(command_str)
% Slider
%
% Simple Example of creating slider GUIs.

if nargin < 1
command_str = 'initialize';
end

if strcmp(command_str,'initialize')
        h_fig = figure(1);  clf;

        h_slider = uicontrol(h_fig,...
             'callback','slidergui(''Slider Moved'');',...
             'style','slider',...
             'min',-100,'max',100,...
             'position',[25 20 150 20]);

        set(h_fig,'userdata',h_slider);
else
        h_slider = get(gcf,'userdata');
        value = get(h_slider,'value');
        disp(value);
end;
```





Back
Close

MATLAB's Graphical User Interface Development Environment — GUIDE

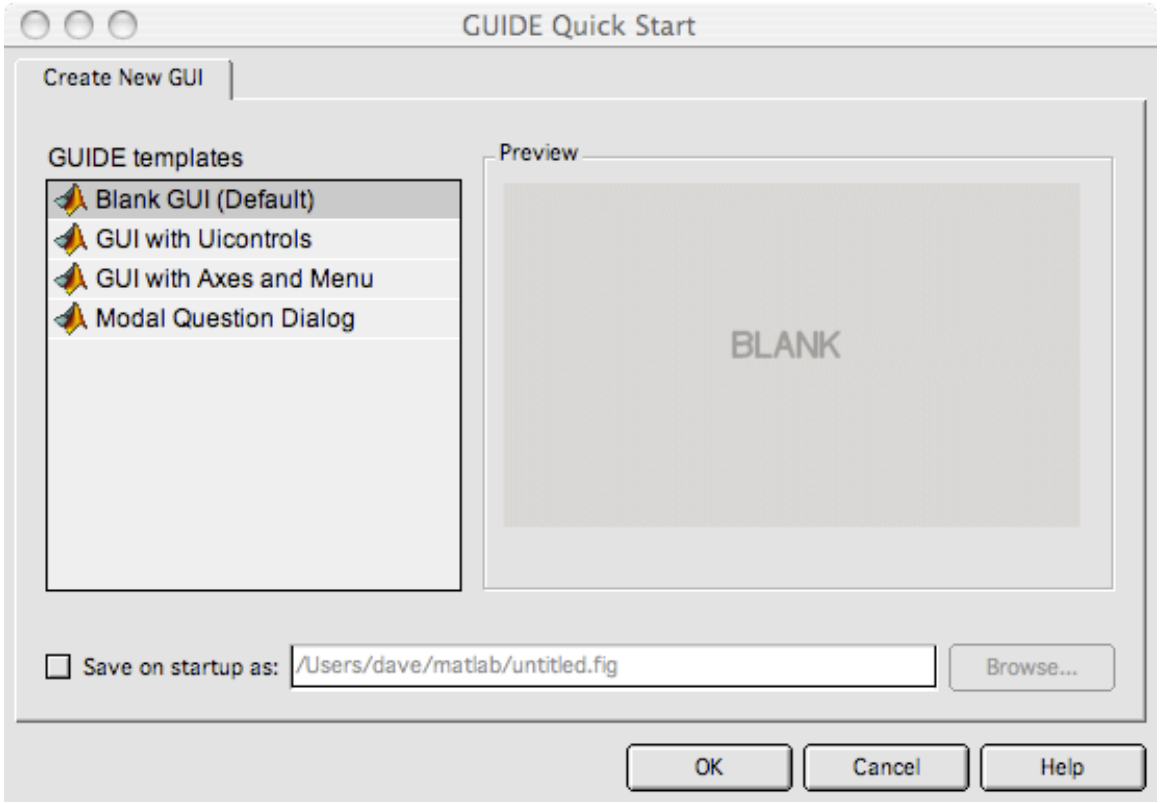
GUIDE provides a WYSIWYG way to assemble your GUI:

- Designing the overall layout and placement of UI elements is easy
- Editing UI element properties is easy
- Guide provides 4 templates with which to assemble your GUI:
  - A blank GUI (default)
  - GUI with Uicontrols
  - GUI with Axes and Menu
  - Modal Question Dialog
- Can also open existing GUIDE GUIs you have made

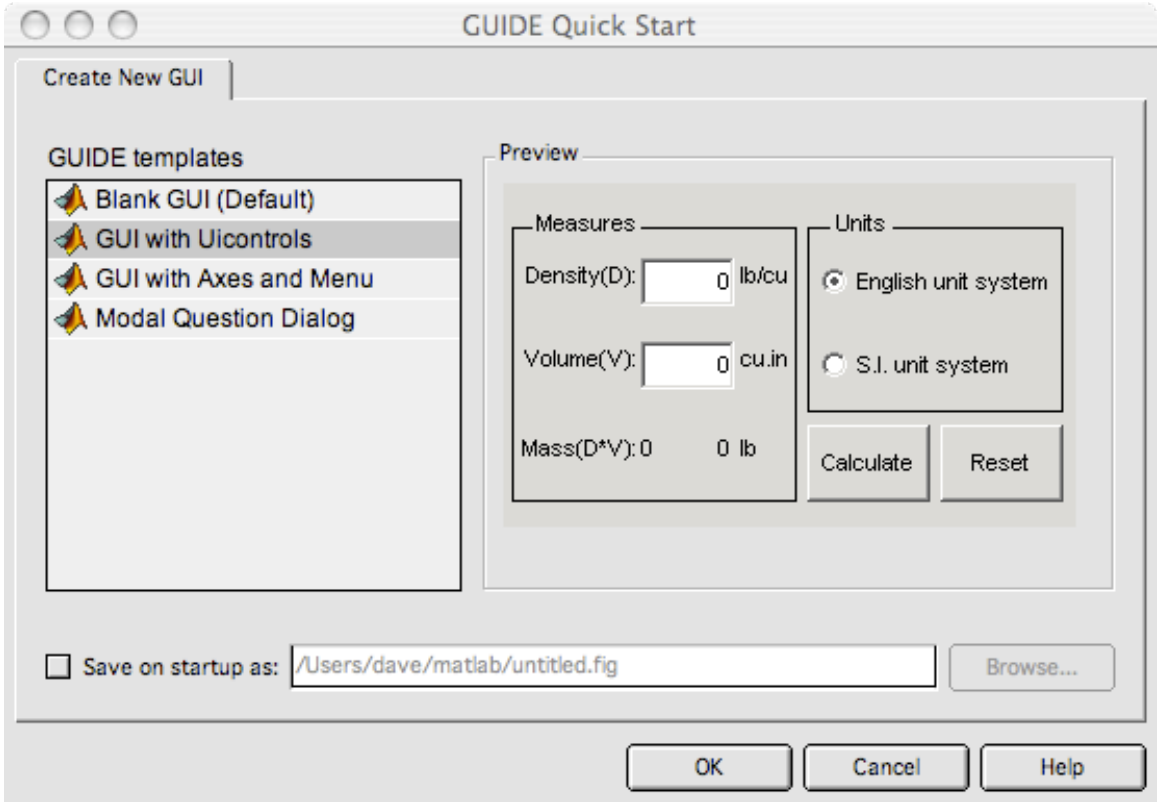
To invoke GUIDE: Type `guide` at command line.



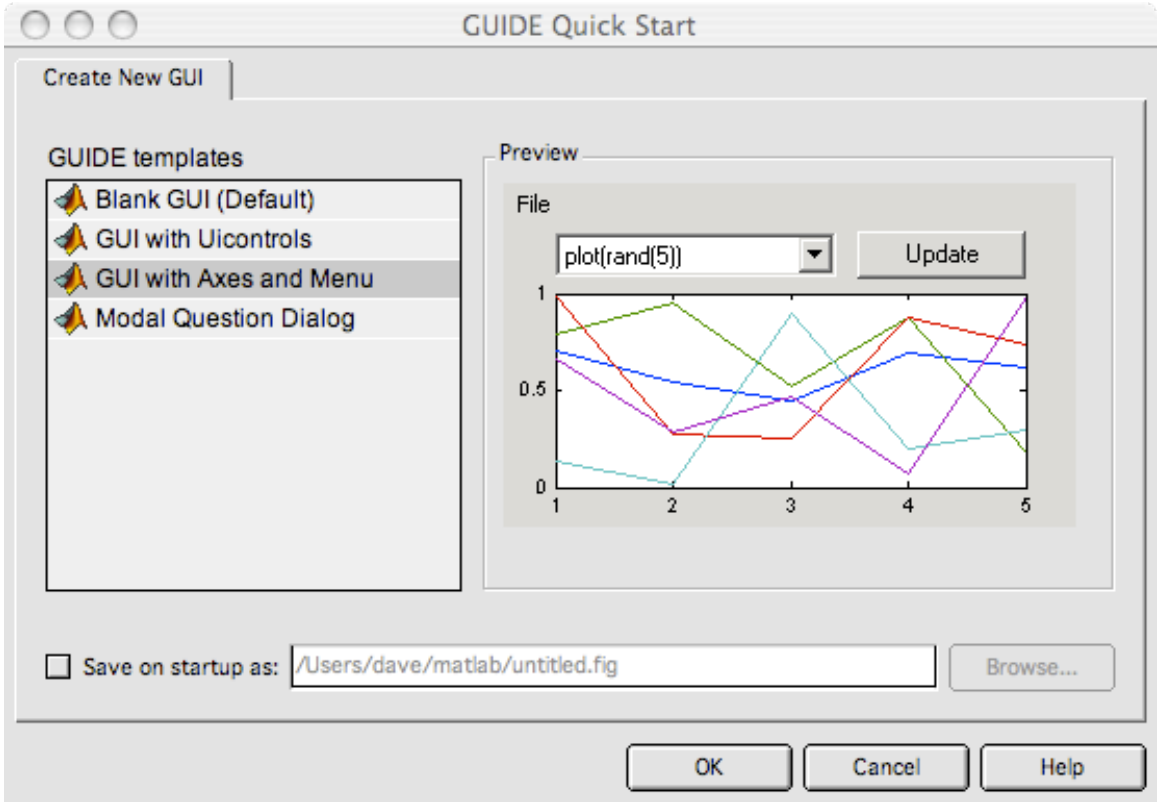

GUIDE: A blank GUI (default)



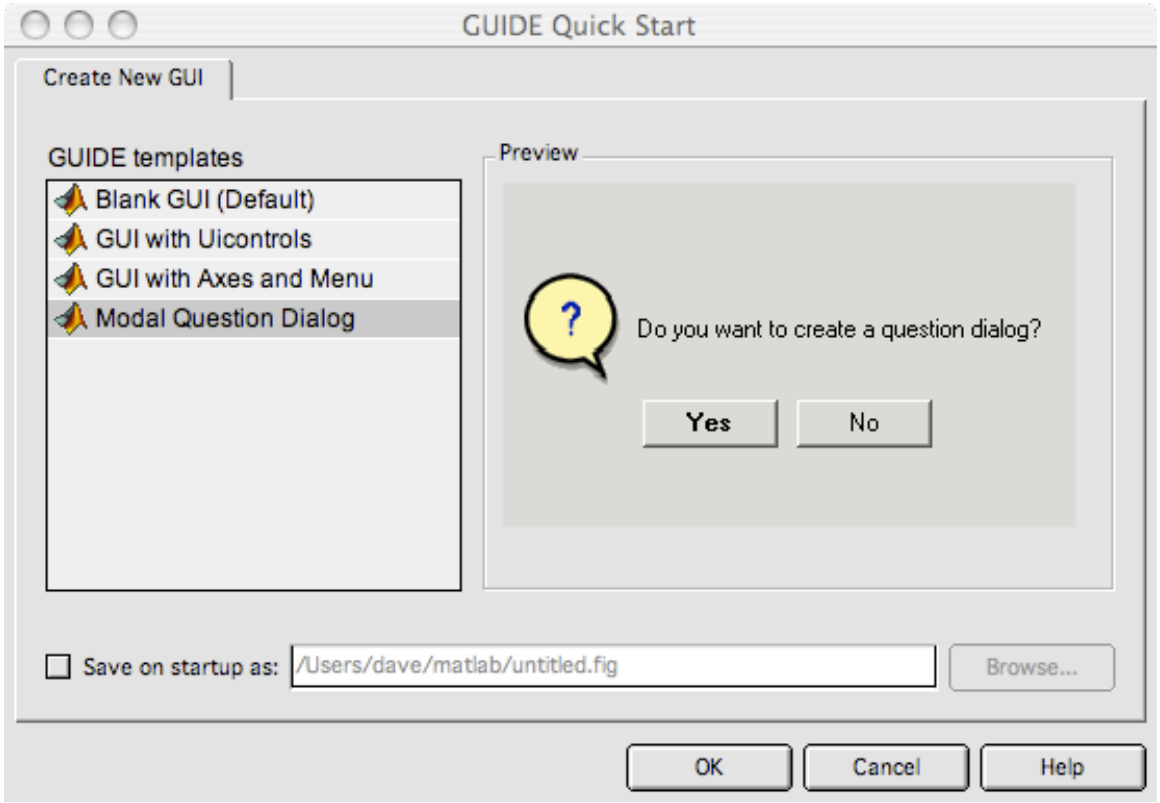
GUIDE: GUI with Uicontrols



GUIDE: GUI with Axes and Menu



GUIDE: Modal Question Dialog






GUIDE Layout Editor

Whichever GUIDE template you select:

- Click on **OK** button in chosen template

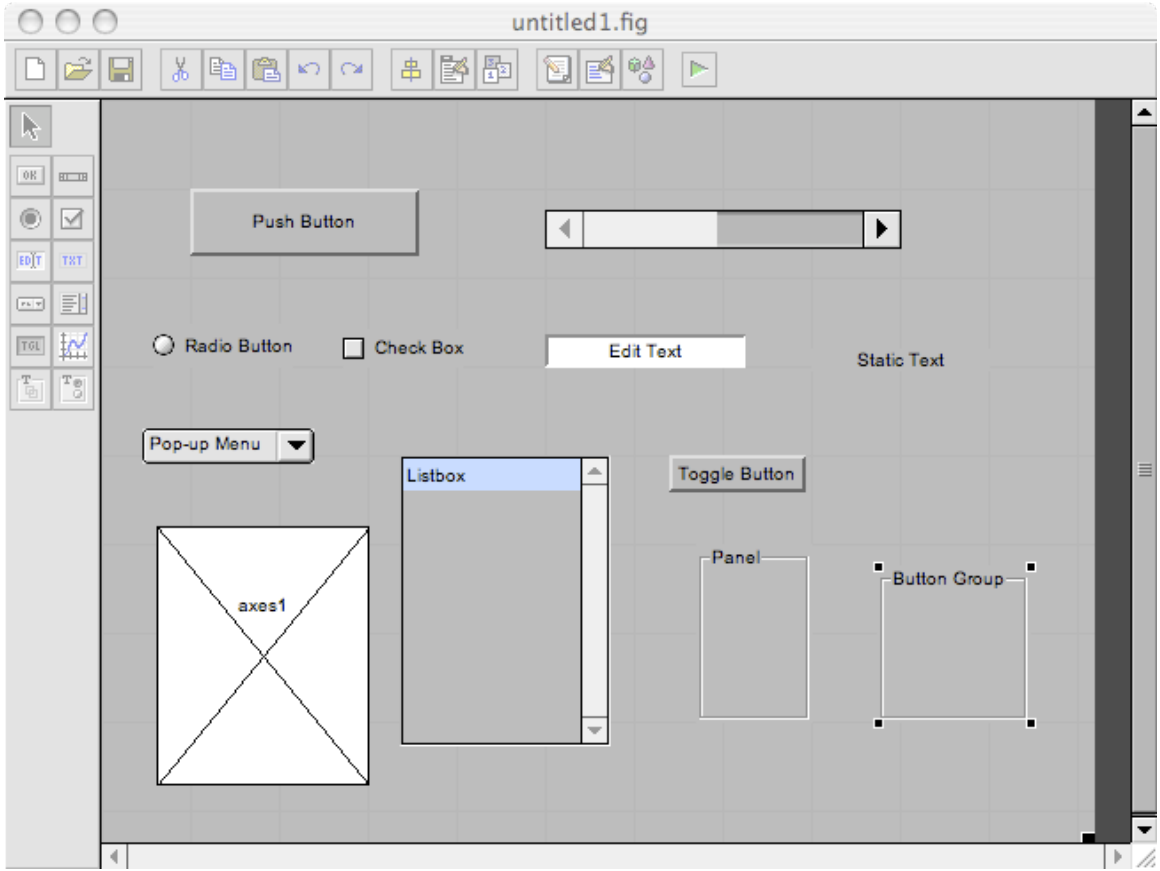
You get the Layout Editor:

- Choose Uicontrol elements on the left panel
- Use **select arrow** to move/resize *etc.*
- Double click on any Uicontrol element to see **Property Inspector** to edit the element — **Example soon**




Back
Close

Layout Editor with sample Uicontrol Elements



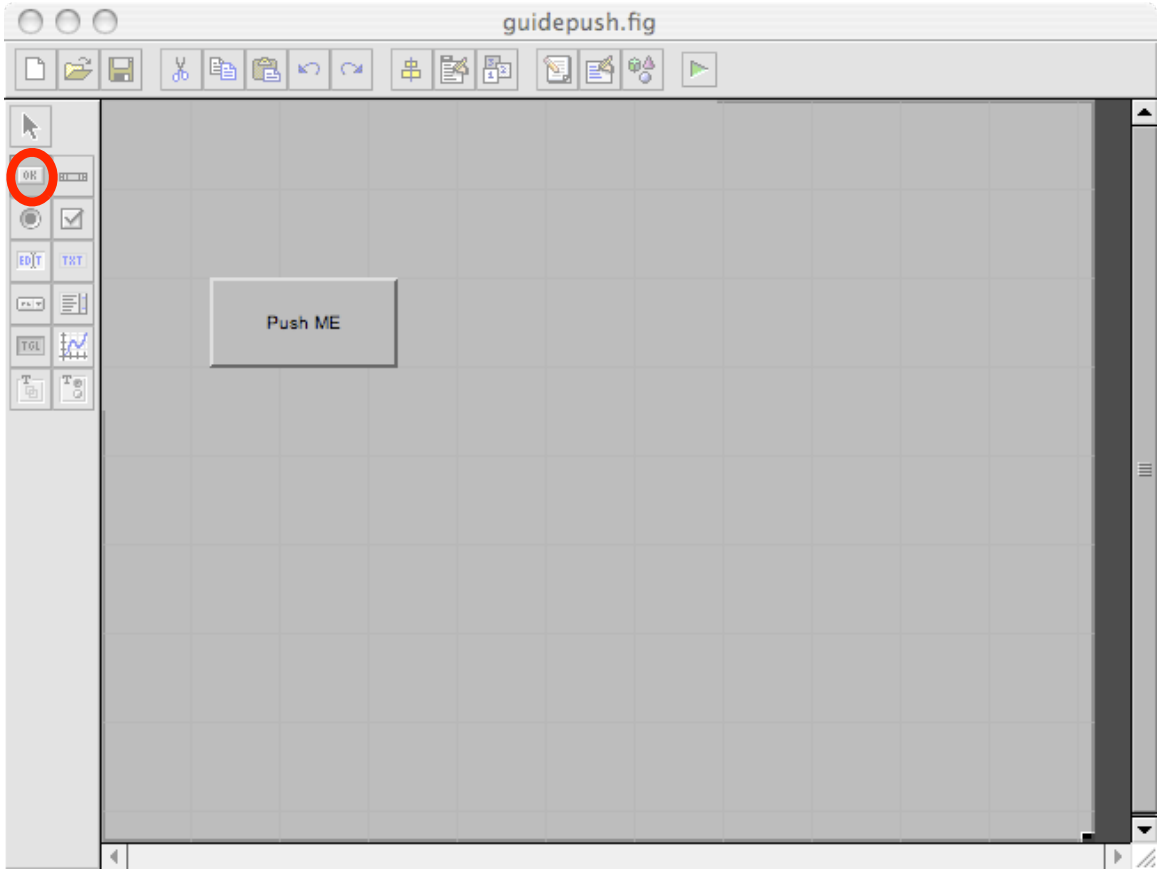
Creating a Simple GUI

Let's illustrate how we use GUIDE to create a simple push button GUI element:

- Start GUIDE: Type `guide` at command line.
- Select a blank GUI template
- Click on OK Button
- Select a Push Button
- Draw a Push Button
- Double click on the button to invoke **Property Inspector**
- Change the buttons text from `Push Button` to `Push ME`.
- Save session as `guidepush`, for example. **Two files created**
  - `guidepush.m` — run this from the command lin
  - `guidepush.fig` — (binary format) GUI data, read by `guidepush.m`.

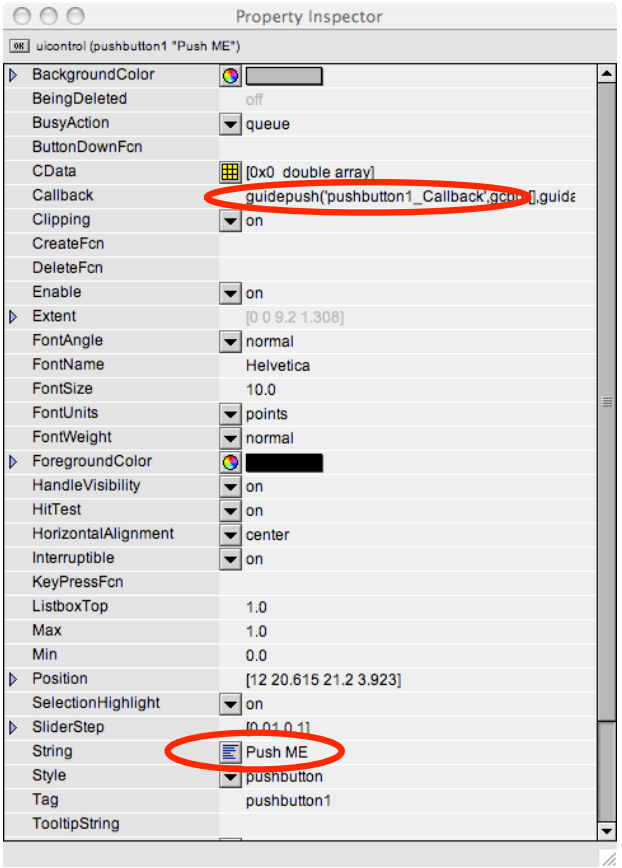



Example Push Button in Layout Editor



Push Button Property Inspector

- Note list of properties — useful for command programming reference
- **String** changed to `Push ME`
- Note callback function:
  - Can be changed
  - We edit this callback
- Other useful stuff to edit.



Adding Functionality to a GUIDE Callback

If you look at the guidepush.m file:

- Quite a lot MATLAB code
- **ONLY edit callback** — unless you know what you are doing
- Callback is pushbutton1_Callback()
- Let's add some simple functionality to this

```
function varargout = guidepush(varargin)
% GUIDEPUSH M-file for guidepush.fig
%      GUIDEPUSH, by itself, creates a new GUIDEPUSH or raises the existing
%      singleton*.
%
%      H = GUIDEPUSH returns the handle to a new GUIDEPUSH or the handle to
%      the existing singleton*.
%
%      GUIDEPUSH('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUIDEPUSH.M with the given input arguments.
%
%      GUIDEPUSH('Property','Value',...) creates a new GUIDEPUSH or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before guidepush_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to guidepush_OpeningFcn via varargin.

........

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @guidepush_OpeningFcn, ...
                   'gui_OutputFcn',  @guidepush_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before guidepush is made visible.
function guidepush_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 guidepush (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guidepush wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = guidepush_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- 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)
```

Editing the Push Button Callback

Initially the callback has no functioning code:

- Let's add a simple print statemen in traditional Hitchhikers Guide to the Galaxy mode:

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

disp('Dont Push Me!');
```

Clearly a lot more to GUIDE — check MATLAB built in docs and help and textbooks