

目录

第十章 用户图形界面.....	3
10.1 用户图形界面是如何工作的.....	3
10.2 创建并显示用记图形界面.....	3
10.2.1 盖头下的一瞥.....	11
10.2.2 一个响应子函数的结构.....	12
10.2.3 给图象增加应用程序数据.....	13
10.2.4 一些有用的函数.....	14
10.3 对象属性.....	14
10.4 图形用户界面组件.....	15
10.4.1 文本域(Text Fields).....	16
10.4.2 编辑框(Edit Boxes).....	16
10.4.3 框架(Frames).....	17
10.4.4 按钮(Pushbuttons).....	17
10.4.5 开关按钮(Toggle Buttons).....	17
10.4.6 复选和单选按钮(Checkboxes and Radio Buttons).....	18
10.4.7 下拉菜单(Popup Menus).....	20
10.4.8 列表框(List Boxes).....	21
10.4.9 滑动条(Sliders).....	23
例 10.1.....	24
10.5 对话框.....	26
10.5.1 错误和警告对话框.....	26
10.5.2 输入对话框.....	27
10.5.3 打开与保存对话框.....	28
10.6 菜单.....	28
10.6.1 禁用默认菜单.....	30
10.6.2 创建自定义菜单.....	31
10.6.3 加速键与键盘助记键.....	31
10.6.4 创建上下文菜单.....	32
例 10.2 绘制数据点.....	32
测试 10.1.....	36
10.7 创建高效 GUIs 的技巧.....	36
10.7.1 工具提示.....	37
10.7.2 伪代码 (p 码, pcode).....	37
10.7.3 附加提高.....	37
例 10.3.....	38
10.8 总结.....	41
10.8.1 好的编程习惯总结.....	42
10.8.2 MATLAB 总结.....	42
10.9 练习.....	43

第十章 用户图形界面

用户图形界面（GUI）是程序的图形化界面。一个好的 GUI 能够使程序更加容易的使用。它提供用户一个常见的界面，还提供一些控件，例如，按钮，列表框，滑块，菜单等。用户图形界面应当是易理解且操作是可以预告的，所以当用户进行某一项操作，它知道如何去做。例如，当鼠标在一个按钮上发生了单击事件，用户图形界面初始化它的操作，并在按钮的标签上对这个操作进行描述。

本章将向大家 **MATLAB** 用户图形界面的基本元素。本章不会对部件和 GUI 特性进行全部的描述，但是它将为你的程序提供必须的 GUI 元素。

10.1 用户图形界面是如何工作的

用户图形界为用户提供了一个熟悉的工作环境。这个环境包括按钮，列表框，菜单，文本框等等，所有的这些控件对用户来说非常地熟悉。所以能够应用它操作应用程序，而不用直接调用操作函数。但是，对于程序员来说，GUI 比较难的，因为程序的每一个控件都必须为鼠标单击做好准备。像鼠标单击这样的输入就是我们熟知的事件，而对事件有反应的程序，我们称之为事件驱动。

创建 **MATLAB** 用户图形界面必须由三个基本元素：

1. 组件 在 **MATLAB** GUI 中的每一个项目(按钮，标签，编辑框等)都是一个图形化组件。组件可分为三类：**图形化控件**(按钮，编辑框，列表，滑动条等)，**静态元素**(窗口和文本字符串)，**菜单和坐标系**，图形化控件和静态元素由函数 `uicontrol` 创建，菜单由函数 `uimenu` 和 `uicontextmenu` 创建，坐标系经常用于显示图形化数据，由函数 `axes` 创建。

2. 图象窗口 GUI 的每一个组件都必须安排图象窗口中。以前，我们在画数据图象时，图象窗口会被自动创建。但我们还可以用函数 `figure` 来创建空图象窗口，空图象窗口经常用于放置各种类型的组件。

3. 响应 最后，如果用户用鼠标单击或用键盘键入一些信息，那么程序就要有相应的动作。鼠标单击或键入信息是一个事件，如果 **MATLAB** 程序运行相应的函数，那么 **MATLAB** 函数肯定会有所反应。例如，如果用户单击一按钮，这个事件必然导致相应的 **MATLAB** 语句执行。这些相应的语句被称为响应。只要执行 GUI 的单个图形组件，必须有一个响应。

基本的 GUI 元素被总结在表 10.1 中，一些元素的例子被显示在图 10.中。我们将会学习这些例子，并通过它们说明 GUI 是如何工作的。

10.2 创建并显示用记图形界面

我们用工具 `guide` 来创建 **MATLAB** 用户图形界面，`guide` 是 GUI 集成开发环境。此工具允许程序员安排用读图形界面，选择和放置所需的 GUI 组件。一旦这些组件放置成功，程序员就能够编辑它们的属性：名字，颜色，大小，字体，所要显示的文本等等。当 `guide` 保存了这个用户图形界面之后它将会自动创建一个包括有骨干函数的工作程序，程序员可以利用这些程序执行用户图形界面的执行动作。

当执行 `guide` 语句时，**MATLAB** 将会创建一个版面编辑器（layout editor），如图图 10.2 所示。带有网格线的大空白区域被称之为布局区（the layout area）。用户可以通过单击

你需要的组件创建任意的目的 **MATLAB** 组件，然后通过拖动它的轮廓线，把它放置在布局区内。在这个窗口的上部用一个带有许多有用工具的工具条，它允许用户分配和联合 GUI 组件，修改每一个 GUI 组件的属性，在用户图形界面中添加菜单等。

创建一个 **MATLAB** 用户图形界面的基本步骤为：

1. 决定这个用户图形界面需要什么样的元素，每个元素需要什么样的函数。在纸上手动粗略地画出组件的布局图。

表 10.1 一些基本的 GUI 组件

元素	创建元素的函数	描述
图形控件		
按钮 (pushbutton)	uicontrol	单击它将会产生一个响应
开关按钮 (togglebutton)	uicontrol	开关按钮有两种状态 “on”，“off”，每单击一次，改变一次状态。每一个单击一次产生一个响应。
单选按钮 (radiobutton)	uicontrol	当单选按钮处于 on 状态，则圆圈中有一个点
复选按钮 (checkbox)	uicontrol	当复选按钮处于 on 状态时，复选按钮中有一个对号
文本编辑框 (editbox)	uicontrol	编辑框用于显示文本字符串，并允许用户修改所要显示的信息。当按下回车键后将产生响应
列表框 (listbox)	uicontrol	列表框可显示一系列文本字符串，用于可用单击或双击选择其中的一个字符串。当用户选择了其中一个字符串后，它将会有有一个响应。
下拉菜单 (popup Menus)	uicontrol	下拉菜单用于显示一系列的文本字符串，当单击时将会产生响应。当下拉菜单没有点击时，只有当前选择的字符串可见
滑动条 (slider)	uicontrol	每改变一次滑动条都会有一次响应。
静态元素		
框架 (frame)	uicontrol	框架是一个长方形，用于联合其他控件。而它则不会产生反应
文本域 (textfield)	uicontrol	标签是在图像窗口内某一点上的字符串。
菜单和坐标系		
菜单项 (menuitems)	Uimenu	创建一个菜单项。当鼠标在它们上单击时，它将会产生一个响应
右键菜单 (contextmenus)	Uicontextmenu	创建一个右键菜单
坐标系 (axes)	Axes	用来创建一个新的坐标系。

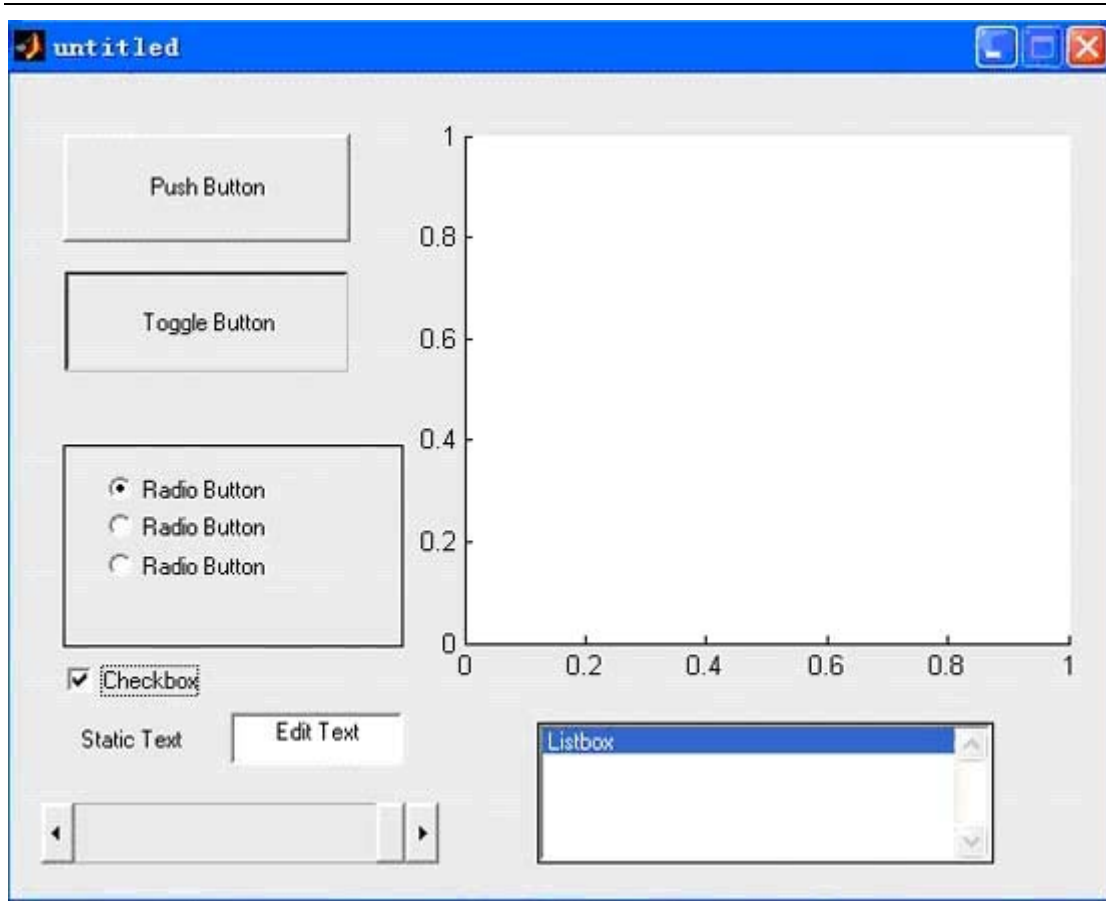


图 10.1 本图显示的是 MATLAB 用户图形界面元素的例子。按从上到下，从左向右的顺序依次为：（1）按钮（2）处于“on”状态的开关按钮。（3）在一个框架中的三个单选按钮（4）复选按钮（5）一个文本域和编辑框（6）滑动条（7）坐标系（8）列表框

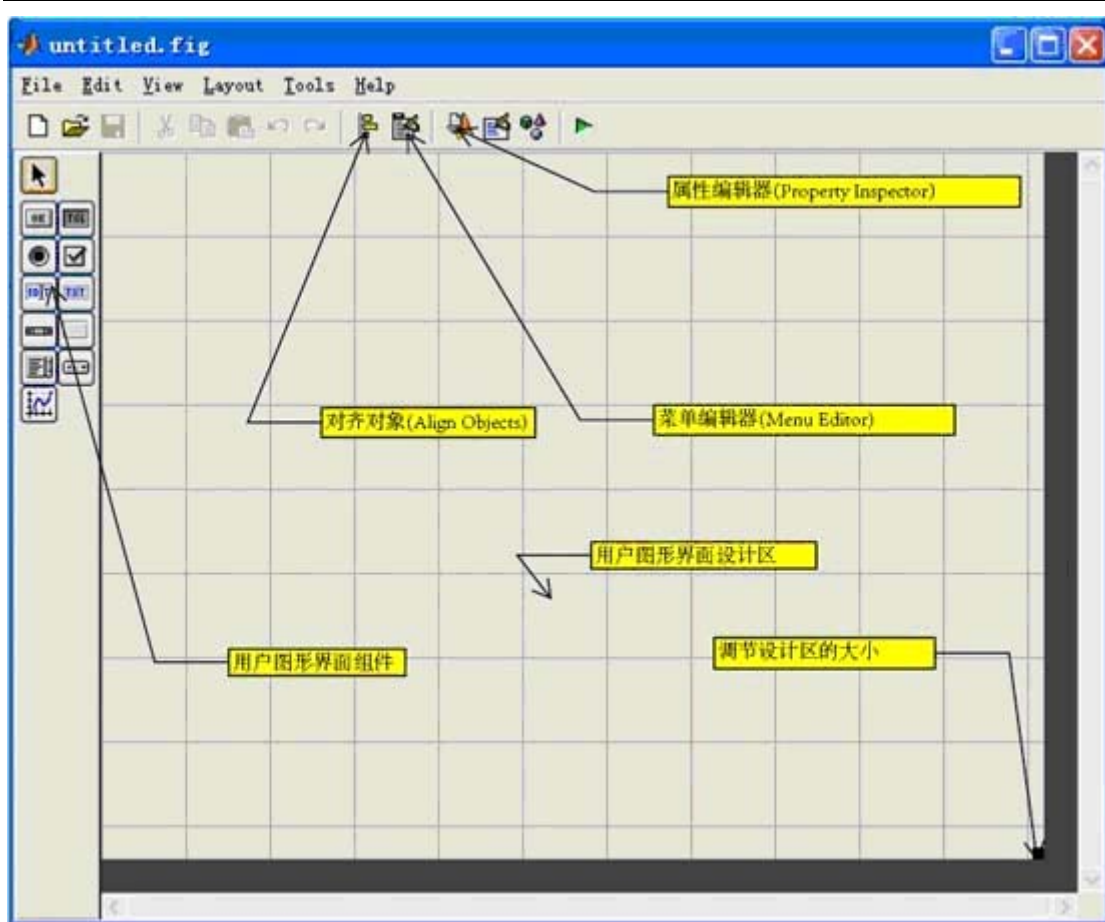
2. 调用 **MATLAB** 工具 **guide** 对图象中的控件进行布局。图象窗口的大小，排列和其中的控件布局都可以利用它进行控制。

3. 我们可以用 **MATLAB** 属性编辑器(property inspector)(内置于 **guide**)给每一个控件起一个名字(标签)，还可以设置每一个控件的其他特性，例如颜色，显示的文本等等。

4. 把图象保存到一个文件中。当文件被保存后，程序将会产生两个文件，文件名相同而扩展名相同。**fig** 文件包括你创建的用户图形界面，**M** 文件包含加载这个图象的代码和每个 GUI 元素的主要响应。

5. 编写代码，执行与每一个回调函数相关的行为。

作为这些步骤的一个简单例子，让我们考虑一个简单的用户图形界面，它包括一个按钮和一个文本框。每单击一次按钮，文本字符串就更新一次，它用于显示用户图形界面启动后的单击总数。



第一步：这个用户图形界面是非常简单的。它包含一个简单的按钮和一个单个的文本域。这个按钮的响应为文本域中的当数字增加 1。这个用户图形界面的草图为图 10.3。

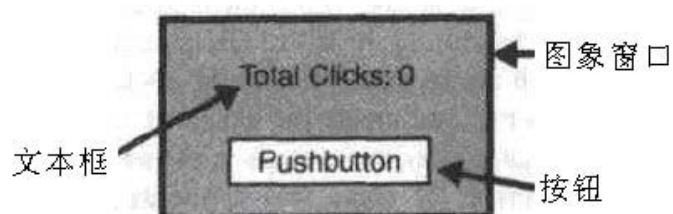


图 10.3 控件布局草图

第二步：对 GUI 控件进行布局，运行函数 `guide`。当运行 `guide` 执行时，它将产生如图 10.2 所示的窗口。

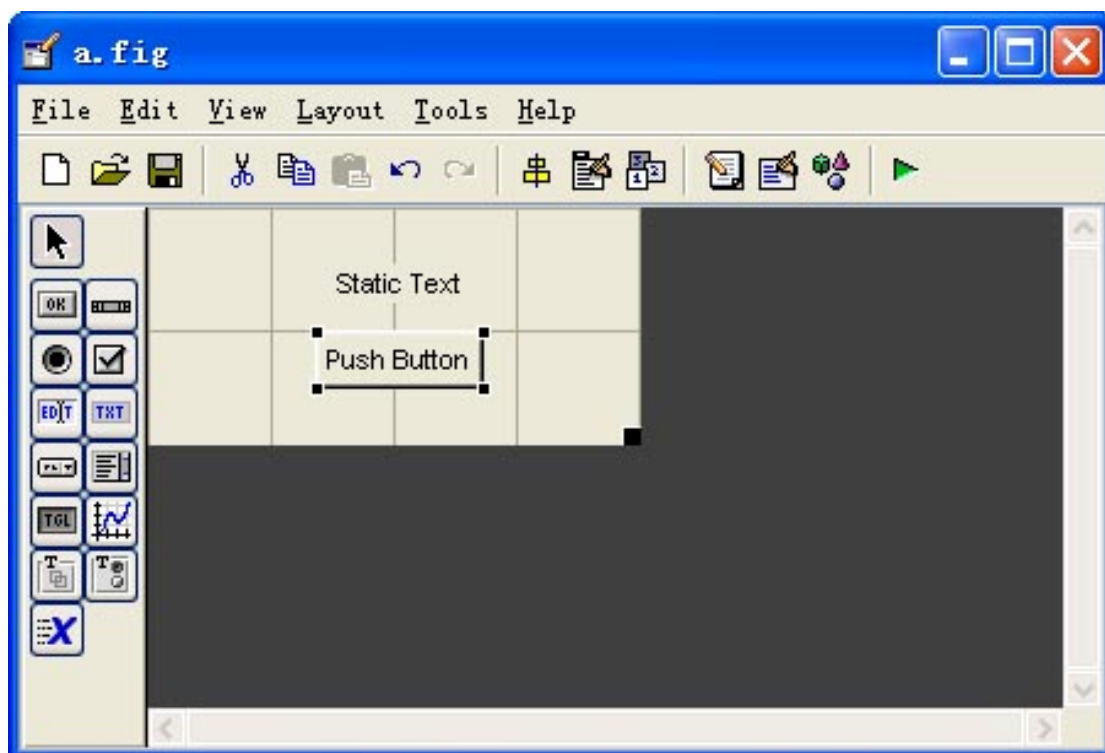



图 10.4 guide 窗口中的用户图形界面的布局

首先，我们必须设置布局的大小，它将生成最终用户图形界面的大小。我们可以通过拖动窗口右下角的小正方形调节布局区的大小和形状。然后单击“push button”按钮然后拖动在布局区创建一个按钮。最后单击“text”按钮，然后拖动在布局区创建一个文本字符串。这些步骤产生的结果如图 10.4 所示。如果我们想让两个控件对齐的话，那么可以用对齐工具 (Alignment Tool) 达到此目的。

第三步：为了设置按钮的属性，右击按钮并选择“Inspect Properties”(编辑属性 )。属性编辑窗口如图 10.5 所示。注意这个属性编辑器列出这个按钮的所有可以得到的属性，并允许我们改变用户图形界面的属性值。属性编辑器运行得到的结果和第九章中介绍 `get` 和 `set` 函数得到的结果相同，但是属性编辑器是一种非常容易使用的形式。

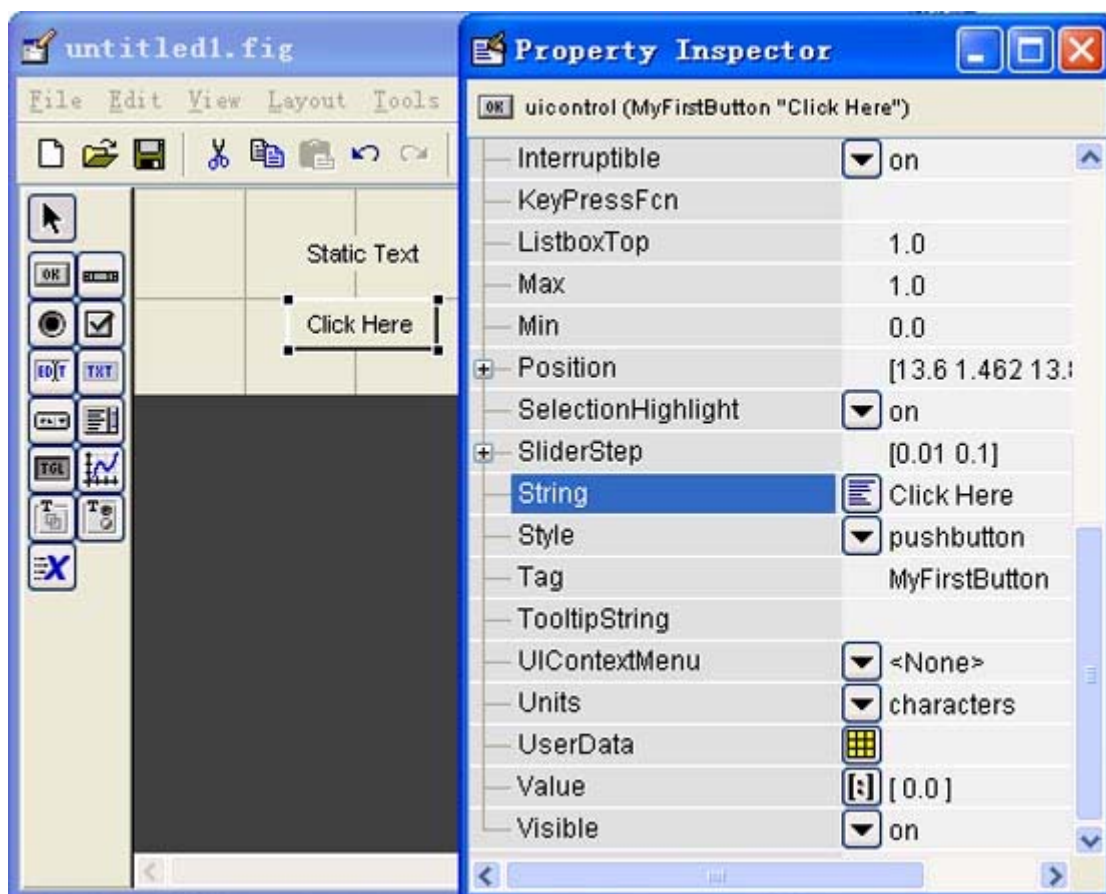


图 10.5 属性编辑器显示的按钮的属性。注意 string 被设置成“Click Here”，Tag 被设置成“MyFirstButton”

对于这个按钮来说，我们可以设置它的许多属性，例如，颜色，大小，字体，文本对齐等属性。但是有两个必须设置的属性：String 属性，它包含的属性值是所要显示的文本；Tag 属性，它的属性值为按钮的名字。在这个情况下，String 被设置为‘ClickHere’，Tag 属性被设置成‘MyFirstButton’。

对于文本域来说，也有两个必须设置的属性：String 属性，它包含的属性值是所要显示的文本；Tag 属性，它的属性值为文本域的名字。回叫函数需要这个名字，并更新的它的文本域。在这个情况下，String 被设置为‘Total Click’，Tag 属性被设置成‘MyFirstText’。经过了这些步骤，布局区如图 10.6 所示。

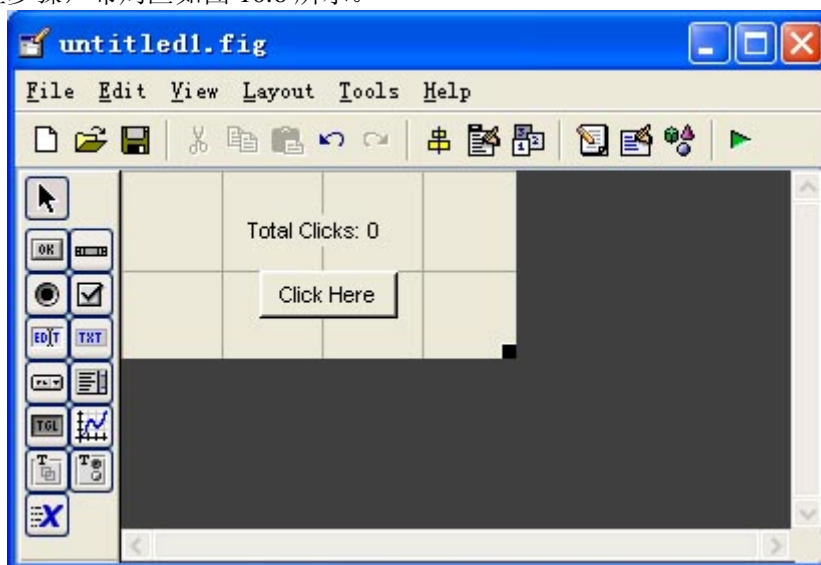


图 10.6 按钮和文本域被修改后的设计区域

我们可以通过单击鼠标在布局编辑区的空白点来调用属性编辑器，然后用属性编辑器检测并设置图象窗口的属性。即使不需要，设置图象窗口的名字是一个非常好的方法。当运行行程序时，在 `name` 属性中的字符串将会显示在用户图形界面的标题栏中。

第四步：我们以 `MyFirstGUI` 为名保存布局区。选择菜单项“File/Save as”，并键入名字 `MyFirstGUI`，然后单击“save”。**MATLAB** 将会产生两个文件，`MyFirstGUI.flg` 和 `MyFirstGUI.m`。图象文件由我们创建的用户图形界面构成。`M` 文件由加载图象文件和创建用户图形界面的代码组成，还包括每一个活动的用户图形界面组件的若干函数。

这时，我们已经拥有了一个完整的用户图形界面，但是它不能完成我们所要求的工作。在命令窗口内输入 `MyFirstGUI`，即可启动你的用户图形界面，如图 10.7 所示。如要你在这个主用户图形界面上单击这个按钮，在命令窗口中将会出现下面的信息。

`MyFirstButton Callback not implemented yet.`



图 10.7 在命令窗口中键入 `MyFirstGUI`，启动对应的用户图形界面

`Guide` 自动创建的一部分 `M` 文件显示在图 10.8 中。这个文件包含函数 `MyFirstGUI`，还有对每一个活动的用户图形界面组件执行响应的哑元子函数，如果函数 `MyFirstGUI` 被调用时无参数，那么这个函数将显示出包含在文件 `MyFirstGUI` 中的用户图形界面。如果函数 `MyFirstGUI` 调用时带有参数，那么函数将会假设第一个参数是子函数的名字，并用 `feval` 调用这个函数，把其它的参数传递给这个函数。

每一个回叫函数(call function)都控制着来自对应的用户图形界面组件的事件。如果鼠标单击事件发生在这个用户图形界面组件上，那么这个组件的回叫函数将自动被 **MATLAB** 调用。这个回叫函数的名字是这个用户图形界面组的 `Tag` 属性值加上字符串“_Callback”，所以，回叫函数 `MyFirstButton` 的名字为“`MyFirstButton_Callback`”。

由 `guide` 创建的 `M` 文件包括了每个活动的用户图形界面组的响应。但是这些响应只是简单的显示一条消息：回叫函数还没有被执行。

第五步：现在，我们需要执行这个按钮对应的子函数。这个函数包括一个 `persistent` 变量，这个变量经常被用于对点击次数进行计数。当单击次数发生在这个按钮上，**MATLAB** 将会调用带有第一个参数 `MyFirstButton_Callback` 的函数 `MyFirstGUI`。然后函数 `MyFirstGUI` 将会调用函数 `MyFirstButton_Callback`，如图 10.9 所示。

这个函数将会对单击的次数增加 1。并创建一个新的包含有这个次数的字符串，并存储在对象文本域中 `String` 属性的新字符串中。在这个步骤运行的函数如下所示：

```
function MyFirstButton_Callback(hObject, eventdata, handles)
% hObject    handle to MyFirstButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Declare and initialize variable to store the count
persistent count
if isempty(count)
    count = 0;
end
% Update count
count = count + 1;
% Create new string
str = sprintf('Total Clicks: %d',count);
```

```
% Update the text field
set(handles.MyFirstText,'String',str);
```

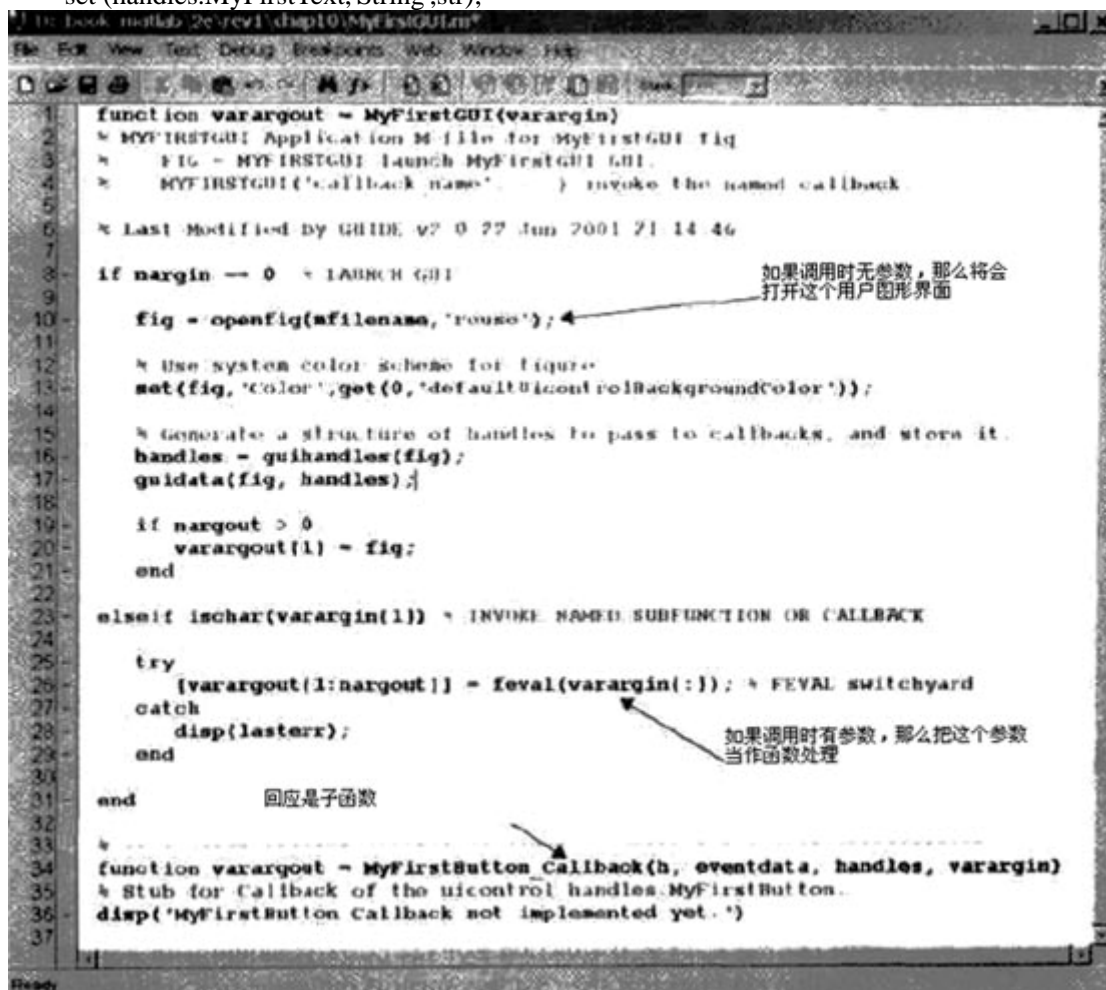


图 10.8 MyFirstGUI 的 M 文件, 是由 guide 自动创建的

程序 MyFirstGUI 的事件运行过程。当一用户用鼠标在按钮进行单击, 带有参数 MyFirstButton_Callback 的函数 MyFirstGUI 就会被自动调用。函数 MyFirstGUI 将会调用子函数 MyFirstButton_Callback。这个函数用来增加次数, 然后把新的次数存入用户图形界面的文本域中。

注意这个函数定义了一个持久变量 count, 并把它初始化为 0。这个函数每一次调用, count 都会增加 1, 这个函数就会创建一个含有 count 的新字符串, 然后, 函数更新在文本域 MyFirstText 中的字符串。

在命令窗口中键入 MyFirstGUI, 运行产生的程序。当用户单击这个按钮时, MATLAB 就会自动调用带有参数 MyFirstButton_Callback 的函数 MyFirstGUI。然后函数 MyFirstGUI 就会调用子函数 MyFirstButton_Callback。这个函数就把变量 count 加 1, 并把这个值显示在文本域中。三次单击产生的用户图形界面如图 10.10 所示。

好的编程习惯

用 guide 对一个新的用户图形界面进行布局, 并用属性编辑器对每一个组件的初始属性进行设置, 例如显示在组件上的文本, 组件的颜色, 还有回叫函数的名字。

好的编程习惯

用 guide 创建完一个用户图形界面后, 人工编辑产生的函数, 增加注释, 描述这个函数的目的和组件, 执行回叫函数的代码。

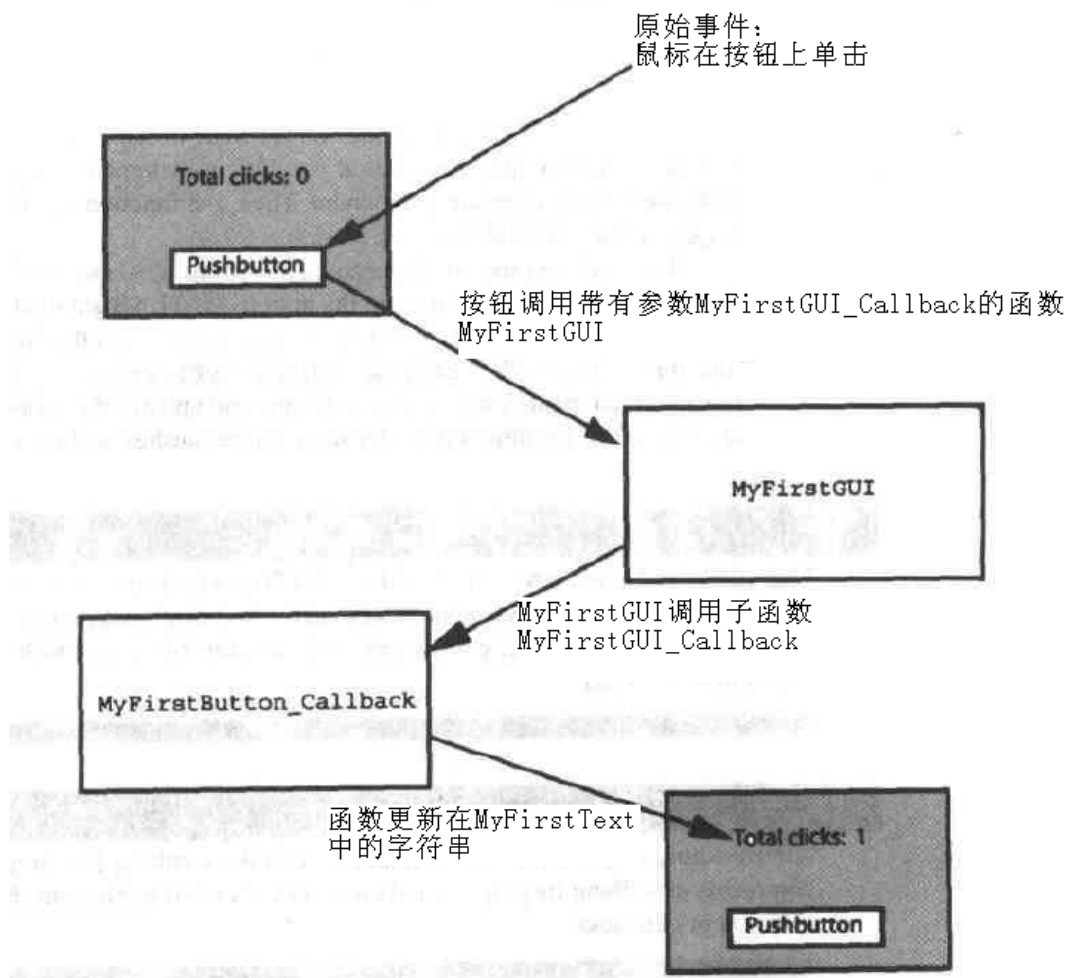


图 10.9



图 10.10 在三次单击按钮之后产生的程序

10.2.1 盖头下的一瞥

图 10.8 显示的 M 文件可以由 `guide` 为 `MyFirstGUI` 产生的。我们现在更加细致地检查这个 M 文件，以了解它是如何工作的。

首先，让我们看一下这个函数是如何声明的。注意这个函数用 `varargin` 来代表它的输

入参数，用 `varargout` 代表它的输出结果。正如我们在第七章学到的，函数 `varargin` 能代表任意数目的输入参数，函数 `varargout` 能代表可变数目的输出参数。所以，用户可以用任意数目的参数调用函数 `MyFirstGUI`。

- 无参数 M 文件的调用

如果用户调用无参数函数 `MyFirstGUI`，由 `nargin` 返回的值将为 0。在这种情况下，程序用函数 `openfig` 从图象文件 `MyFirstGUI.fig` 中加载用户图形界面。这个函数的形式为

```
fig = openfig(mfilename, 'reuse');
```

其中 `mfilename` 是指所要加载的图象文件的名字。第二个参数用于指定是一个拷贝运行指定的次数，或指定同时运行多个拷贝。如果这个参数是 "reuse"，那么这个图象窗口只能有一个窗口在运行，如果调用带有 "reuse" 参数的函数 `openfig`，而且指定的图象窗口已经存了，那么原来的窗口将会被置于屏幕的顶端并被重复利用。相反地，如果这个参数为 "new"，那么这个函数的多个拷贝就可以运行。如果调用带有 "reuse" 参数的函数 `openfig`，那么每一次运行都会创建一个新的拷贝。默认地，用 `guide` 创建的用户图形界面是参数 "reuse"，所以无论在什么时候，图象窗口只会有一个拷贝。如果你想有多个用户图形界面，你必须人工更改函数的参数。

在图象窗口被加载之后，函数将会执行下面的语句

```
set(fig, 'Color', get(0, 'defaultUicontrolBackgroundColor'));
```

这个函数用于设置图象窗口的背景色，当 **MATLAB** 运行时，背景色就是计算机默认的背景色。这个函数把这个用户图形界面的颜色设置为计算机本地窗口的颜色。所以在 windows 操作系统上编写的用户图形界面，也可以用在 UNIX 系统的计算机上，反之亦然。在不同的操作系统下，看起来非常的自然。

下面的两个语句创建了一个结构，这个结构的元素为当前窗口中所有对象的句柄，这两个语句把这个结构当作应用程序数据来存储。

```
handles = guihandles(fig);
```

```
guidata(fig, handles);
```

函数 `guihandles` 创建了一个结构，这个结构的元素为指定窗口中所有对象的句柄。结构的每一个元素的名字与组件的 `Tag` 属性相同，它的值就是每一个组件的句柄。例如在 `MyFirstGUI` 中返回的句柄结构为

```
>> handles = guihandles(fig)
```

```
handles =
```

```
figure1: 99.0005
```

```
MyFirstText: 3.0021
```

```
MyFirstButton: 100.0007
```

在本图中有三个用户图形界面组件——图象窗口本身，一个文本域和一个按钮。函数 `guidata` 把句柄结构当作应用程序数据来存储，这里我们将用函数 `setappdata`（在第九章有介绍）。

如果有指定一个输出参数给 `MyFirstGUI`，那么在这里最后的一系列语句将返回这个图象句柄给这个调用者。

```
if nargin > 0
```

```
varargout{1} = fig;
```

```
end
```

- 带有参数的 M 文件的调用

如果调用带有参数的函数 `MyFirstGUI`，那么由 `nargin` 返回的值将会比 0 大。在这个种情况下，程序把第一个参数当作一个回叫函数的名字，并用 `feval` 执行这个函数。这个函数执行函数 `narargin(1)`，并把剩余的函数传递给这个函数。这种机制使子函数变为回叫函数，而这个子函数不能直接被程序其他部分直接调用。

10.2.2 一个响应子函数的结构

每一个调用子函数都有它的标准形式

```
function varargout = ComponentTag_Callback(h, eventdata, handles, varargin)
```

其中 ComponentTag 是产生响应的组件的名字(在 Tag 属性中的字符串)。子函数的参数为。

- h 父图象的句柄
- eventdata 当前不用的数组
- handles 由所有组件句柄构成的结构
- varargin 对回叫函数的增补函数。如果程序员需要的话, 可以利用这个特性为回叫函数提供更多的附加信息。

注意每一个回叫函数都用全部的权限访问 handles 结构, 所以每一个回叫函数可以修改图象文件中的每一个用户图形界面组件。我们可以充分利用按钮的回叫函数的结构, 其中按钮的回叫函数修改了在文本域中所要显示的文本。

```
% Update the text field
set(handles.MyFirstText, 'String', str);
```

10.2.3 给图象增加应用程序数据

应用程序需要把应用程序指定的信息存储到 handles 结构中, 而不存储到全局内存和持久内存中。产生的用户图形界面将更加的耐用, 因为其他 **MATLAB** 程序不能突然更改全局用户图形界面数据, 还因为多个相同的用户图形界面之间不相互冲突。

为了增加本地数据到 handles, 在用 guide 创建了用户图形界面之后, 我们必须人工地修改 M 文件。程序员, 在调用应用数据到 guidata 之后和到 guide 之前, 添加应用到 handles 结构。例如, 为了添加鼠标单击的次数 count 到 handles 结构中, 我们将修改这个程序如下:

```
% Generate a structtrue of handles to pass to callbacks
handles = guidata(fig);
% Add count to the structure.
handles.count = 0;
% Store the structure
guidata(fig, handles);
```

当用到这个应用数据时, 那么这个应用数据将会通过 handles 结构传递给每一个回叫函数。用到 count 值的按钮回叫函数如下所示。注意如要在结构中的任意一个信息被修改了, 那么我们必须把结构存储到 guidata。

```
function varargout = MyFirstButton_Callback(h, eventdata, handles, varargin)
% Update count
handles.count = handles.count + 1;
% Save the updated handles structure
guidata(h, handles);
% Create new string
str = sprintf('Total Clicks: %d', handles.count);
```

```
% Update the text field
set(handles.MyFirstText, 'String', str);
```

好的编程习惯

把 GUI 应用程序数据存储到 handles 结构中, 以便任意的一个回叫函数都可以应用它。

好的编程习惯

如果你修改了 handles 结构中的任何 GUI 应用数据, 确保在函数退出之前保存了调用 guidata 的结构。

10.2.4 一些有用的函数

在设计回叫函数过程中有三种特殊函数经常被使用：`gcbo`，`gcbf` 和 `findobj`。虽然这些函数在 MATLAB 6 GUIs 中的没有像以前版本那么频繁，它们还是非常有用，做为程序员，肯定会碰到。

`gcbo` 函数（获得回叫对象）返回产生回叫函数的对象的句柄，`gcbf` 函数（获得回叫图形）返回包含该对象的图形的句柄。这些函数可以被回叫函数用来确定产生回叫的对象或图形，以便可以修改图形中的对象。

`findobj` 函数搜索父对象中的所有子对象，查找那些指定属性具有特定值的对象，它返回任何特征匹配的对象句柄。`findobj` 最常用的格式是

```
Hndl = findobj(parent, 'Property', Value);
```

其中 `parent` 是父对象（如图形）的句柄，`Property` 是要检查的属性，而 `Value` 是要查找的值。

例如，假设程序员要更改名称（`tag`）为“`Button1`”的按钮的文字，该程序可能先查找这个按钮，然后用下面的语句替换该文字：

```
Hndl = findobj(gcbf, 'Tag', 'Button1');  
set(Hndl, 'String', 'New text');
```

10.3 对象属性

每个 GUI 对象都包含一系列可以自定义该对象的扩展属性，各种类型的对象（如图形、坐标轴，`uicontrols` 控件等）之间只有轻微的差别，所有类型的对象的所有属性都可以通过帮助浏览器在线找到它们的介绍文档，但是图形对象和 `uicontrols` 控件的一些较重要属性在表 10.2 和表 10.3 中粗略列出。

对象的属性值可以通过使用对象检查器或者是使用 `get` 和 `set` 函数进行修改，虽然对象检查器在 GUI 设计过程中可以很方便的修改属性，我们必须在程序运行过程中使用 `set` 和 `get` 函数动态地修改属性值，例如在回叫函数中进行修改。

表 10.2 图形的重要属性

属性	描述
Color	设定图形的颜色。值要么是预定义的颜色如“r”、“g”或“b”，要么是一个有 3 个元素的向量，这 3 个元素分别代表红、绿和蓝，范围从 0-1 之间。比如洋红色为[1 0 1]。
MenuBar	设定是否在图形上显示默认菜单。可以设为“figure”表示显示默认菜单，而设为“none”则删除菜单。
Name	设定要图形标题栏显示的名称
NumberTitle	设定是否在标题栏显示图形数量，可以设为“on”或者“off”。
Position	设定图形在屏幕上的位置，单位为“units”。这个值接受一个 4 元素的向量，前 2 个元素表示图形左下角的 x 和 y 坐标。而后 2 个元素则表示图形的宽和高。
SelectionType	设定鼠标在图形上的最后一次击键时选择的类型。单击返回“normal”，双击返回“open”。还有一些附加选项，请查阅 MATLAB 在线文档。
Tag	图形的名称，可以用来选中定位图形。
Units	用来描述图形位置的单位，可选的值为“inches”、“centimeters”、“normalized”，“points”、“pixels”或“characters”。默认为“pixels”。
Visible	设定图形是否可见，可选值为“on”或“off”。

表 10.2 图形的重要属性

属性	描述
WindowStyle	设定图形风格是普通的还是模式的（参阅对话框的讨论），可选的值为“normal”或“modal”。

表 10.3 uicontrol 控件的重要属性

属性	描述
BackgroundColor	设定对象的背景颜色。值要么是预定义的颜色如“r”、“g”或“b”，要么是一个有 3 个元素的向量，这 3 个元素分别代表红、绿和蓝，范围从 0-1 之间。比如洋红色为[1 0 1]。
Callback	设定对象被键盘或文本输入激活时被叫函数的名称和参数。
Enable	设定对象是否可选，如果不能，则对鼠标或键盘输入没有响应。可选值为“on”或“off”。
FontAngle	设定在对象上显示的字符串的角度，可选值为“normal”，“italic”和“oblique”。
FontName	设定对象上显示的字符串的字体名称。
FontSize	设定对象上显示的字符串的字号大小。默认是单位是 points。
FontWeight	设定对象上显示的字符串的字体粗细，可选值为“light”、“normal”、“demi”和“bold”。
ForegroundColor	设定对象的前景色。
HorizontalAlignment	设定对象中的字符串的水平对齐情况，可选值为“left”、“center”和“right”。
Max	设定对象 value 属性的最大值。
Min	设定对象 value 属性的最小值。
Parent	包含本对象的图形的句柄。
Position	设定图形在屏幕上的位置，单位为“units”。这个值接受一个 4 元素的向量，前 2 个元素表示对象相对于图形在左下角的 x 和 y 坐标。而后 2 个元素则表示对象的宽和高。
Tag	对象的名称，可以用来选中定位对象。
TooltipString	设定鼠标光标指到该对象时显示的帮助文本。
Units	用来描述图形位置的单位，可选的值为“inches”、“centimeters”、“normalized”，“points”、“pixels”或“characters”。默认为“pixels”。
Value	uicontrol 控件的当前值。对开关按钮、复选按钮和单选按钮来说，选中时，这个值为 max，没有选中时值为 min。对其它控件来说有不同的意义。
Visible	设定对象是否可见。可选值为“on”或“off”。


10.4 图形用户界面组件

本节概述了常见图形用户界面组件的基本特性，讨论了如何创建和使用每种组件，同时也讨论了每种组件能产生的事件类型。本节讨论的组件有：

- 文本域
- 编辑框
- 框架
- 按钮


- 开关按钮
- 复选按钮
- 单选按钮
- 下拉菜单
- 列表框
- 滑动条

10.4.1 文本域(Text Fields)

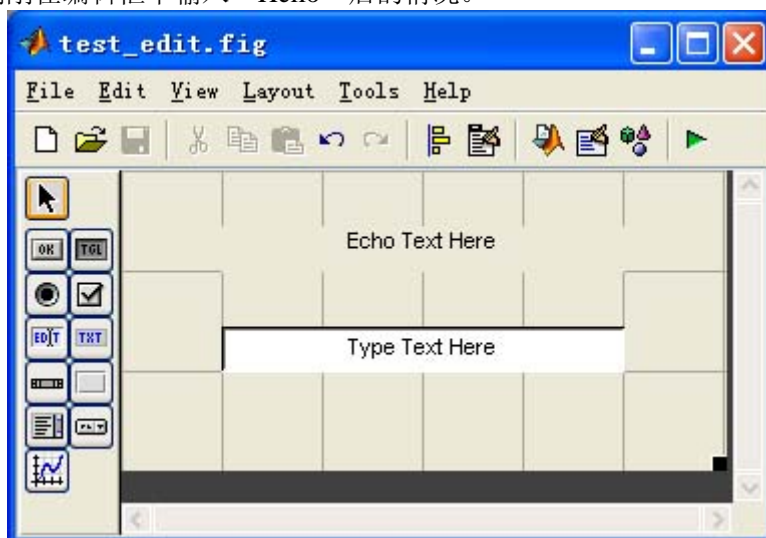
文本域是显示文本的图形对象，可以通过设定文本域的水平对齐属性改变显示文本时的对齐方式，创建时默认是水平居中。文本域是通过创建风格为“edit”的 uicontrol 控件来创建的。可以通过使用版面编辑器中的文本域工具（）把文本域添加到 GUI 中。

文本域并不产生回叫，不过文本域中显示的文本可以在回叫函数中通过设定 String 属性来更改，如 10.2 节所示。

10.4.2 编辑框(Edit Boxes)

编辑框是允许用户输入文本的图形对象，当用户在文本框中输完文本后按回车 Enter 时会产生回叫。文本域是通过创建风格为“edit”的 uicontrol 控件来创建的。可以通过使用版面编辑器中的编辑框工具（）把编辑框添加到 GUI 中。

如图 10.11a 所示，该图是一个简单的 GUI，包含了名为“EditBox”的编辑框和名为“TextBox”的文本域各一个。当用户在编辑框中输入字符串后，它将自动调用 EditBox_Callback 函数，如图 10.11b 所示。这个函数使用 handles 结构定位编辑框获得用户的输入内容，然后再定位文本域并把字符串在文本域中显示出来。图 10.12 显示了 GUI 启动之后用户刚刚在编辑框中输入“Hello”后的情况。



(a)

```
function EditBox_Callback(hObject, eventdata, handles)
% Find the value typed into the edit box
str = get ( handles.EditBox,'String');
% Place the value into the text field
set (handles.TextBox,'String', str);
```

(b)

图 10.11 (a) 只有一个编辑框和文本域的布局 (b) GUI 的回叫函数

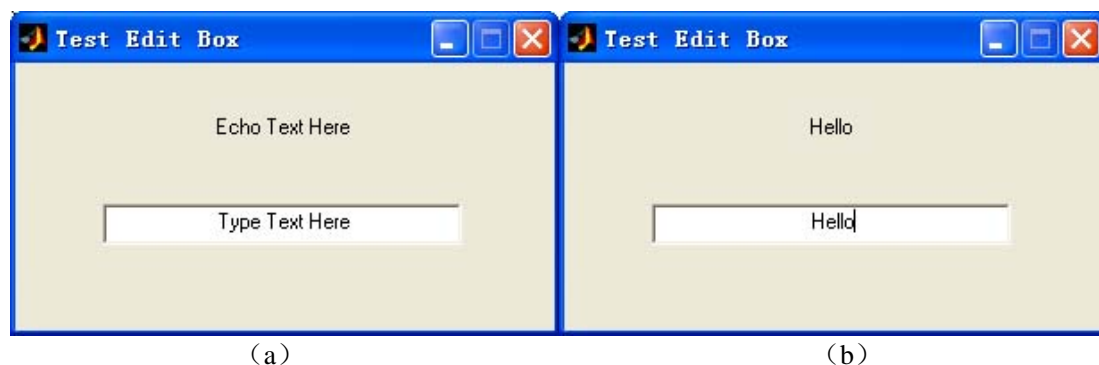



图 10.12 (a) 由程序 test_edit 产生的 GUI (b) 用户输入 Hello 并回车后的 GUI

10.4.3 框架(Frames)

框架是在 GUI 界面上显示一个矩形框，你可以把逻辑上相联系的对象用框架框起来。如图 10.1 所示，框架可以把一组单选按钮组合起来。

框架是通过创建风格为 “frame” 的 uicontrol 控件来创建的。可以通过使用版面编辑器中的框架工具 () 把框架添加到 GUI 中。框架不会产生回叫。

10.4.4 按钮(Pushbuttons)


按钮是当用户在其上面单击时能触发特定行为的一种组件。当用户用鼠标在其单击时，按钮会产生回叫。按钮是通过创建风格为 “pushbutton” 的 uicontrol 控件来创建的。可以通过使用版面编辑器中的按钮工具 () 把按钮添加到 GUI 中。

图 10.10 中的 MyFirstGUI 函数演示了如何使用按钮。

10.4.5 开关按钮(Toggle Buttons)

开关按钮是有两种状态的一类按钮：on（被按下去）和 off（没有被按下去）。当鼠标单击时，开关按钮在两种状态之间切换，并且每次都产生回叫。当开关按钮为 on（被按下去）时，“Value” 属性的值被设为 max（通常是 1），当为 off（没有被按下去）时，“Value” 属性的值被设为 min（通常是 0）。


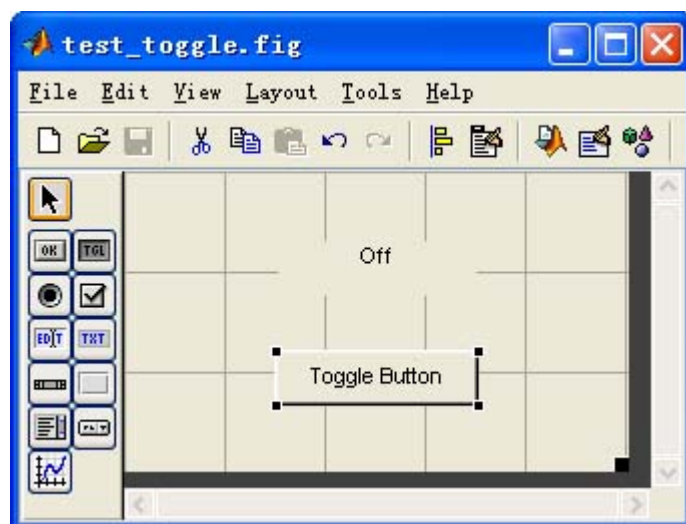
开关按钮是通过创建风格为 “Togglebutton” 的 uicontrol 控件来创建的。可以通过使用版面编辑器中的开关按钮工具 () 把开关按钮添加到 GUI 中。

图 10.13a 是一个含有开关按钮（名为 “ToggleButton”）和文本域（名为 “TextBox”）的简单 GUI。当用户在开关按钮上单击时，将自动调用 ToggleButton_Callback 函数（如图 10.13b 所示）。这个函数使用 handles 结构定位开关按钮并从 “Value” 属性获得按钮的状态，然后再定位文本域并把按钮状态在文本域中显示出来。图 10.14 显示了 GUI 启动之后用户刚刚在开关按钮上单击后的情况。

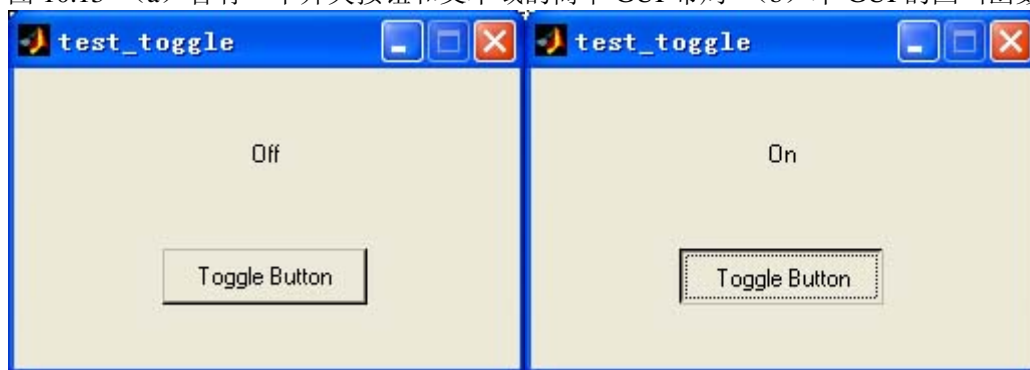


(a)

```
function ToggleButton_Callback(hObject, eventdata, handles)
% Find the state of the toggle button
state = get(handles.ToggleButton,'Value');
% Place the value into the text field
if state == 0
    set(handles.TextBox,'String','Off');
else
    set(handles.TextBox,'String','On');
end
```

(b)

图 10.13 (a) 含有一个开关按钮和文本域的简单 GUI 布局 (b) 本 GUI 的回叫函数



(a)

(b)



图 10.14 (a) 由程序 test_toggle 产生的当开关按钮为 off 时的 GUI

(b) 开关按钮为 on 时的 GUI

10.4.6 复选和单选按钮(Checkboxes and Radio Buttons)

复选和单选按钮在本质上是与开关按钮一样的，除了它们的外观不同之外。与开关按钮一样，复选与单选按钮有两种状态：on 和 off。当鼠标单击时，按钮会在两种状态之间切换，每次产生回叫。当按钮状态为 on 时，复选或单选按钮的“Value”被设为 max（通常为 1），当按钮状态为 off 时，“Value”值被设为 min（通常为 0）。复选与单选按钮的形状如图 10.1 所示。

复选按钮是通过创建风格为“checkbox”的 uicontrol 控件来创建的，而单选按钮则是通过创建风格为“radiobutton”的 uicontrol 控件来创建的。可以通过使用版面编辑器中的

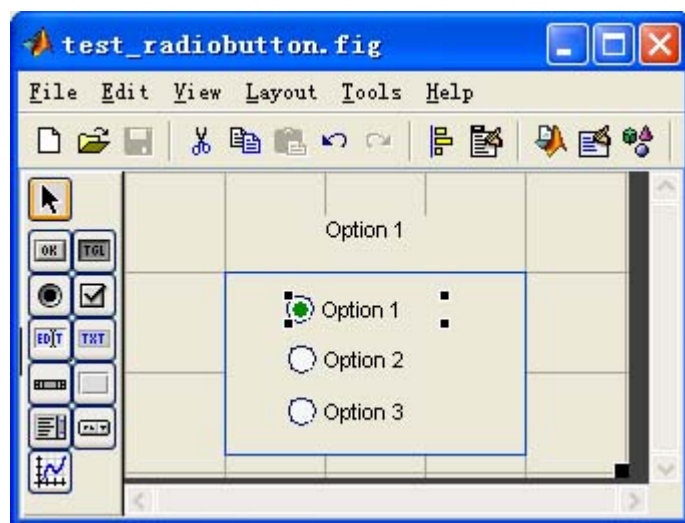
复选按钮工具 () 把复选按钮添加到 GUI 中, 可以通过使用版面编辑器中的单选按钮工具 () 把单选按钮添加到 GUI 中。

复选按钮通常用来作为“开/关”选择, 而单选按钮通常用在一组排它性选项中选一。

图 10.15a 演示了如何用单选按钮创建一组排它性选项。这个图形的 GUI 创建了三个单选按钮, 标为“Option 1”、“Option 2”和“Option 3”, 每个单选按钮拥有独自的参数, 使用相同的回叫函数。

相应的回叫函数如图 10.15b 所示。当用户单击一个单选按钮时, 相应的回叫函数被执行, 函数在文本框中显示当前选项同, 选中相应的单选按钮, 并把其它单选按钮设为未选状态。

注意 GUI 使用框架把单选按钮组合到一起, 使它们看起来更像是一组。图 10.6 显示了“Option 2”被选中后的情况。



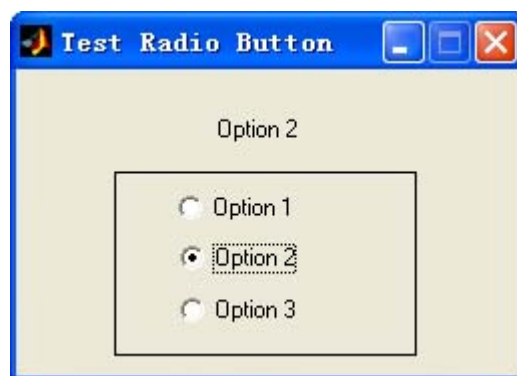
(a)

```
% --- Executes on button press in radiobutton1.  
function radiobutton1_Callback(hObject, eventdata, handles)  
set(handles.Label1,'String','Option 1');  
set(handles.radiobutton1,'Value',1);  
set(handles.radiobutton2,'Value',0);  
set(handles.radiobutton3,'Value',0);
```

```
% --- Executes on button press in radiobutton2.  
function radiobutton2_Callback(hObject, eventdata, handles)  
set(handles.Label1,'String','Option 2');  
set(handles.radiobutton1,'Value',0);  
set(handles.radiobutton2,'Value',1);  
set(handles.radiobutton3,'Value',0);
```

```
% --- Executes on button press in radiobutton3.  
function radiobutton3_Callback(hObject, eventdata, handles)  
set(handles.Label1,'String','Option 3');  
set(handles.radiobutton1,'Value',0);  
set(handles.radiobutton2,'Value',0);  
set(handles.radiobutton3,'Value',1);
```

(b)



(c)

10.4.7 下拉菜单(Popup Menus)


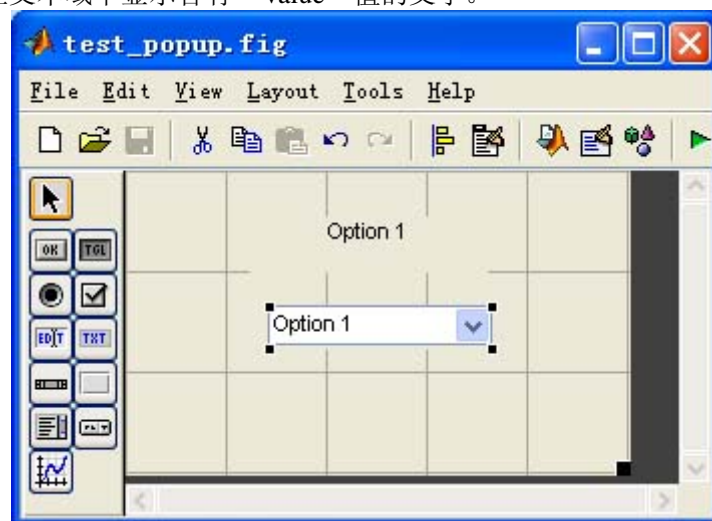
下拉菜单允许用户选择一组排它性列表选项中的一项目。用户可以选择的列表选项是由一个字符串胞数组指定的。“Value”属性的值指示了当前哪个字符串被选中。下拉菜单可以通过版面编辑器上的下拉菜单工具 () 添加到 GUI 中。

图 10.17a 是一个使用下拉菜单的例子。图形中的 GUI 创建了有五个选项的下拉菜单，依次标为“Option 1”、“Option 2”……

相应的回叫函数如图 10.17b 所示。回叫函数通过检查菜单“Value”参数的值取得当前选中项，然后在文本域中显示含有“Value”值的文字。



(a)

```
% --- Executes on selection change in Popup1.
function Popup1_Callback(hObject, eventdata, handles)
% Find the value of the popup menu
value = get(handles.Popup1,'Value');
% Place the value into the text field
str = ['Option ' num2str(value)];
set(handles.Label1,'String',str);
```

(b)

图 10.17 (a) 含有下拉菜单和用来显示当前选择的项的文本域 GUI 布局 (b) 本 GUI 的回叫函数

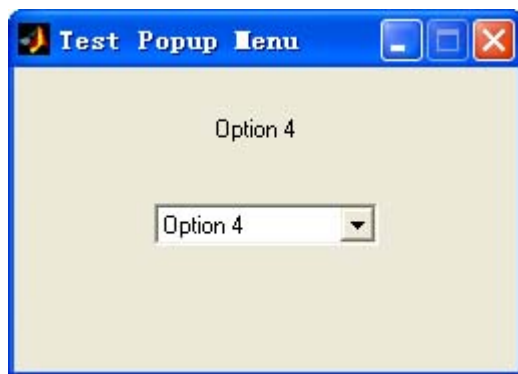



图 10.18 由程序 test_popup 产生的 GUI

10.4.8 列表框(List Boxes)

列表框可以显示多行文本并允许用户选择其中一行或多行的图形对象，如果文本的行数比刚好填满列表框还要多，则将会在列表框中出现滑动条以使用户可以上下滚动列表项。用户能够选择的文本行由一字符串数组指定，“Value”属性的值指示了当前哪些字符串被选中。

列表框由风格属性为“listbox”的 uicontrol 创建，用户可以使用版面编辑器中的列表框工具（）把它添加到 GUI 中。

列表框可以用来从几个可能选项中选项其中的一项。在平常的 GUI 使用中，在列表框中单击选择某项并不产生动作，而是要等到某些外部触发器（如按钮）来产生动作。不过，鼠标双击通常立即产生动作。可以通过设置图形对象的 SelectionType 属性的不同使单击与双击事件有差异：单击设置为“normal”，双击设置为“open”。

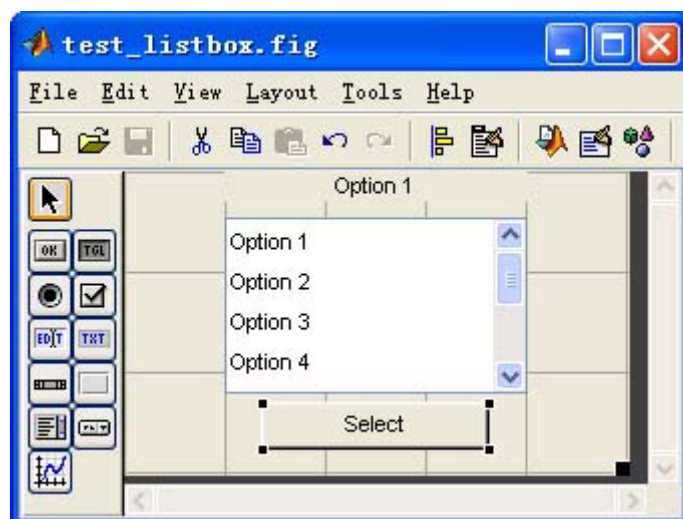
有时也允许列表框多选。如果 max 属性值与 min 属性值的差大于 1，那么就允许多选，否则则仅允许单选。

图 10.19a 是列表框仅允许单选的例子。图形中的 GUI 创建了有八个选项的列表框，这八项分别标为“Option 1”、“Option 2”……“Option 8”。此外，GUI 创建了一个按钮用来完成选择并显示选项。列表框和按钮都会产生回叫。

相应的回叫函数如图 10.19b 所示。如果列表框中的项被选中，则函数 ListBox1_Callback 将会被执行，这个函数会检查产生回叫的图形（使用 gcbf 函数），看看选择动作是单击还是双击，如果是单击，则什么也不做，双击则取得列表框中的选中值，然后在文本域中显示适当的文字。

如果是按钮被选中，则执行 Button1_Callback 函数，取得列表框中的选中值，然后在文本域中显示适当的文字。程序 test_listbox 创建的 GUI 如图 10.20 所示。

在本章练习中，你将会被要求修改本例，使得列表框可以复选。



(a)

```
% --- Executes on button press in Button1.
function Button1_Callback(hObject, eventdata, handles)
% Find the value of the popup menu
value = get(handles.Listbox1,'Value');
% Update text label
str = ['Option ' num2str(value)];
set(handles.Label1,'String',str);

% --- Executes on selection change in Listbox1.
function Listbox1_Callback(hObject, eventdata, handles)
% if this was a double click, update the label.
selectiontype = get(gcf,'SelectionType');
if selectiontype(1) == 'o'
    % Find the value of the popup menu
    value = get(handles.Listbox1,'Value');
    % Update text label
    str = ['Option ' num2str(value)];
    set(handles.Label1,'String',str);
end
```

(b)

图 10.19 (a) 放置列表框、按钮和文本域的界面 (b) 本 GUI 的回叫函数，注意单击选择和双击选择都能引发回叫。



图 10.20 程序 test_listbox 产生的 GUI

10.4.9 滑动条(Sliders)

滑动条是允许用户通过用鼠标移动滑块来在最大值和最小值之间选择一个值的一种图形对象。设置到“Value”中的值在 min 和 max 之间，具体取决于滑块的位置。


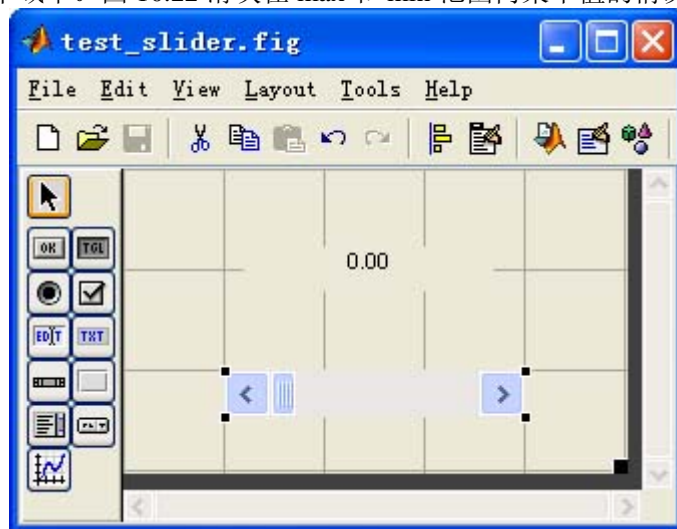
滑动条是由风格为“slider”的 uicontrol 控件创建的。可以通过版面编辑器中的滑动条工具（）把滑动条添加到 GUI 中。

图 10.21a 是版面放置了一个滑动条和一个文本域的情况，滑动条的“min”（最小值）属性设为 0，“max”（最大值）属性设为 10。当用户拖动滑块时，将自己调用 Slider1_Callback 函数（如图 10.21b 所示），这个函数从滑动条的“Value”属性取得当前值并把它显示在文本域中。图 10.22 滑块在 max 和 min 范围内某个值的情况。



(a)

```
% --- Executes on slider movement.  
function Slider1_Callback(hObject, eventdata, handles)  
% Find the value of the slider  
value = get(handles.Slider1,'Value');  
% Place the value in the text field  
str = sprintf('%.2f',value);  
set(handles.Label1,'String',str);
```

(b)

图 10.21 (a) 带有滑动条和文本域的简单界面 (b) 本 GUI 的回叫函数

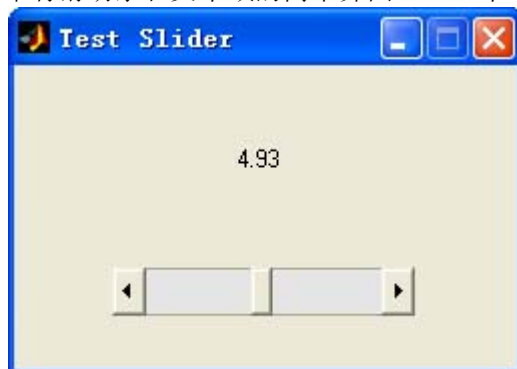


图 10.22 由程序 test_slider 产生的 GUI

例 10.1

温度转换

写一个程序，在 0-100°C 内进行华氏温度和摄氏温度之间的相互转换，要求使用 GUI 接受数据和显示结果。程序必须有一个编辑框用来输入华氏温度，另一个编辑框用来输入摄氏温度，一个滑动条用来连续调节温度值。用户可以在编辑框中输入直接温度值，也可以通过移动滑块来输入。GUI 在中所有元素必须与值相一致。

解决方案

要创建这个程序，华氏温度需要一个文本域和编辑框，摄氏温度也同样需要一个文本域和编辑框，还需要一个滑动条。需要两个函数：一个把华氏温度转换成摄氏温度，另一个把摄氏温度转换成华氏温度。最后还需要编写回叫函数来支持用户的输入。

要转换的值范围在 32-212°F 或 0-100°C，因此把滑动条的范围设计 0-100 是比较合适的，并把滑动条设成摄氏温度。

这个过程的第一步是使用 guide 向导来设计 GUI，使用 guide 创建五个必需的 GUI 元素，并把它们放在合适的位置。然后使用属性编辑器实现下面的步骤：

1. 为每个 GUI 元素取合适的名称并编辑相应的 Tag 值。这里取 “Label1”、“Label2”、“Edit1”、“Edit2” 和 “Slider1”。
2. 修改两处标签的 “String” 值为 “Degrees F” 和 “Degrees C”。
3. 把滑动条的 min 属性和 max 属性分别设为 0 和 100。
4. 保存初始值到编辑框的 “String” 中和滑动条的 “Value” 中，它们相应为 “32°F”、“0°C” 和 “0”。
5. 把本图形的标题栏名称 (Name) 设为 “Temperature Conversion”。

上面的步骤完成后把 GUI 保存为 temp_conversion.fig，会产生一个图形文件和配对的 M 文件。M 文件中保存了编辑框和文本框的回叫函数的存根。界面布局如图 10.23 所示。

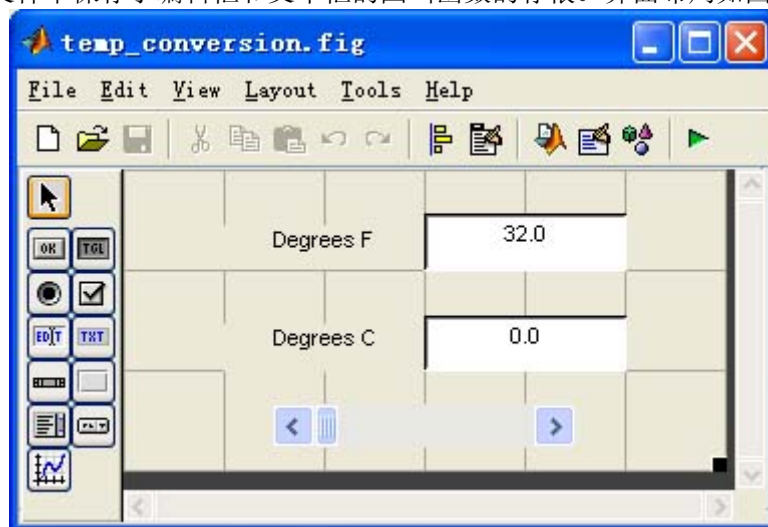


图 10.23 温度转换 GUI 界面布局

第二步是创建函数把华氏温度转换成摄氏温度。函数 to_c 把华氏温度转换成摄氏温度，它必须以下面的公式计算

$$\text{deg } C = \frac{5}{9} (\text{deg } F - 32) \quad (10.1)$$

这个函数的代码是：

```
function deg_c = to_c(deg_f)
% Convert degrees Fahrenheit to degrees C.
deg_c = (5/9) * (deg_f - 32);
```

函数 to_f 把摄氏温度转成华氏温度，用下在的公式计算

$$\text{deg } F = \frac{9}{5} (\text{deg } C) + 32 \quad (10.2)$$

这个函数的代码如下：

```
function deg_f = to_f(deg_c)
% Convert degrees Celsius to degrees Fahrenheit
deg_f = (9/5) * deg_c + 32;
```

最后，我们必须编写回叫函数把它绑在一起。这些函数必须能对三个按钮作出反应。（注意我们既要难对用户输入在编辑框的输入做出更新，也要以一致的格式来显示数据，同时，如果用户输入的值落在允许范围之外，还必须能够更正它。）

这里还有另外一个因素要考虑：我们在编辑框中输入的值是字符串 `strings`，但我们要以数字来对待它。如果用户在编辑框输入 100，则实际上得到的是字符串 “100”，并不是数字 100。回叫函数必须把字符串转换成数字，以便用来计算。这种转换用 `str2num` 函数来完成，它把字符串转换成数字。

同时，回叫函数必须限制用户输入的值在有效范围之内，即 0-100°C 和 32-212°F 之间。

回叫函数如图 10.24 所示。

```
function Edit1_Callback(hObject, eventdata, handles)
% Update all temperature value
deg_f = str2num(get(hObject,'String'));
deg_f = max([32 deg_f]);
deg_f = min([212 deg_f]);
deg_c = to_c(deg_f);

% Now update the fields
set(handles.Edit1,'String',sprintf('%.1f',deg_f));
set(handles.Edit2,'String',sprintf('%.1f',deg_c));
set(handles.Slider1,'Value',deg_c);

function Edit2_Callback(hObject, eventdata, handles)
% Update all temperature value
deg_c = str2num(get(hObject,'String'));
deg_c = max([0 deg_c]);
deg_c = min([100 deg_c]);
deg_f = to_f(deg_c);

% Now update the fields
set(handles.Edit1,'String',sprintf('%.1f',deg_f));
set(handles.Edit2,'String',sprintf('%.1f',deg_c));
set(handles.Slider1,'Value',deg_c);

% --- Executes on slider movement.
function Slider1_Callback(hObject, eventdata, handles)
% Update all temperature values
deg_c = get(hObject,'Value');
deg_f = to_f(deg_c);

% Now update the fields
set(handles.Edit1,'String',sprintf('%.1f',deg_f));
set(handles.Edit2,'String',sprintf('%.1f',deg_c));
set(handles.Slider1,'Value',deg_c);
```

图 10.24 温度转换 GUI 的回叫函数

程序完成，执行并分别向文本框或滑动条输入几个不同的值试试，注意使用范围之外的值试试。它是否正确执行了？

10.5 对话框

对话框是一种通常用来显示信息或从用户处获得输入的特殊图形，对话框通常用来显示出错、提供警告、提问问题或获取输入。它们也用来选取文件或打印机属性。

对话框可以是有模式的或无模式的。模式对话框在它消失之间不允许程序中的其它窗口被访问，而无模式对话框则不存在这些限制。模式对话框典型的应用是用来显示错误或警告，而这些信息通常是需要引起重视，不可忽略的。默认地，大多数对话框是无模式的。

MATLAB 包含了很多类型的对话框，大多数对话框已经在表 10.4 中列出。这里我们只讨论其中的几种，关于其它对话框的内容请参阅 MATLAB 在线文档。

表 10.4 挑选的对话框

属性	描述
dialog	创建一个通用对话框
errordlg	在对话框中显示错误信息，用户必须点击 OK（确定）按钮才能继续。
helpdlg	在对话框中显示帮助信息，用户必须点击 OK（确定）按钮才能继续。
inputdlg	显示需要输入数据，并接受输入的数据。
printdlg	显示打印机选择对话框。
questdlg	提问。这种对话框可以包含二到三个按钮，默认是 Yes（是），No（否）和 Cancel（取消）。
uigetfile	显示打开文件对话框。这类对话框允许用户选择要打开的文件，但并不打开文件。
uiputfile	显示保存文件对话框。这类对话框允许用户选择要保存的文件，但并不保存文件。
uisetcolor	显示颜色选择对话框。
uisetfont	显示字体选择对话框。
warndlg	在对话框中显示警告信息，用户必须点击 OK（确定）按钮才能继续。

10.5.1 错误和警告对话框

错误与警告对话框有相似的调用参数与行为，实际上，这两种对话框仅仅是显示的图标不同而已。最常见的调用格式如下：

```
errordlg(error_string,box_title,create_mode);  
warndlg(warning_string,box_title,create_mode);
```

其中 error_string 或 warning_string 就是要显示给用户的信息，box_title 是对话框的标题，create_mode 可以是“modal”或“non_modal”，这取决于你要何种对话框。

例如，下面的语句创建一个用户无法忽略的模式对话框，这个语句产生的对话框如图 10.25 所示。

```
errordlg('Invalid input values!','Error Dialog Box','modal');
```



图 10.25 错误对话框

10.5.2 输入对话框

输入对话框提示用户输入程序需要的一个或多个值，可以用下面格式创建：

```
answer = inputdlg(prompt)
answer = inputdlg(prompt, title)
answer = inputdlg(prompt, title, line_no)
answer = inputdlg(prompt, title, line_no, default_answer)
```

这里的 `prompt` 是一个字符串数组，数组的每个元素就是要求用户回答的问题，`title` 指定对话框的标题，`line_no` 限定用户输入的行数，最后，`default_answer` 是一个胞数组——包含了具体某个问题用户没有回答时的默认答案。注意有多少个问题，就必须提供多少个默认答案。

当用户单击对话框上的 `OK` 按钮时，答案将在字符串胞数组的形式保存到变量 `answer` 中。

作为输入对话框的例子，假设我们要用输入对话框允许用户指定图形的位置，下面的代码将能做到：

```
prompt{1} = 'Starting x position:';
prompt{2} = 'Starting y position:';
prompt{3} = 'Width:';
prompt{4} = 'Height:';
title = 'Set Figure Position';
default_ans = {'50', '50', '180', '100'};
default = inputdlg(prompt, title, 1, default_ans);
```

结果如图 10.26 所示。

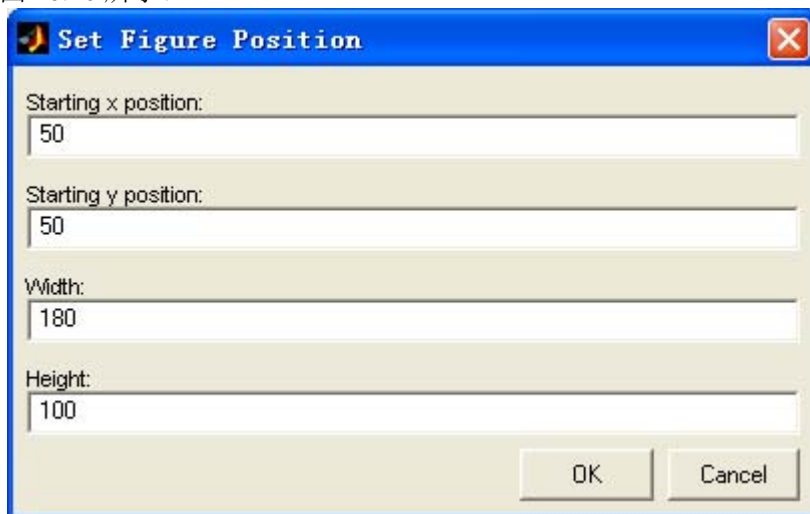


图 10.26 输入对话框

10.5.3 打开与保存对话框

`uigetfile` 和 `uisetfile` 对话框是设计用来允许用户交互地选择要打开或保存的文件，这些对话框返回文件名及路径，但并不实际读取或保存它。程序员负责写额外的代码。

这两个对话框的形式如下：

```
[filename, pathname] = uigetfile(filter_spec, title);
```

```
[filename, pathname] = uisetfile(filter_spec, title);
```

参数 `filter_spec` 是指定在对话框中显示的文件类型的字符串，如 “*.m”、“*.mat” 等等。参数 `title` 指定对话框的标题。对话框执行后，`filename` 包含了所选择的文件名，而 `pathname` 包含了文件的路径。如果取消对话框，则 `filename` 被设为 0。

下面的脚本文件演示了如何使用这些对话框，它提示用户输入 MAT 文件名，读取文件的内容，图 10.27 是在 Windows Xp 系统中运行的结果（不同的操作系统显示的界面会有不同）。

```
[filename, pathname] = uigetfile('*.mat', 'Load MAT File');
```

```
if filename ~= 0
```

```
    load([pathname filename]);
```

```
end
```

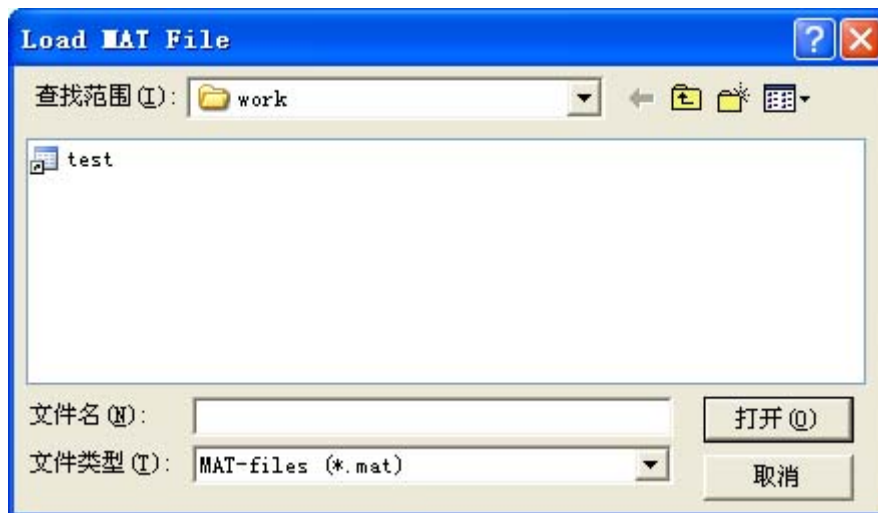


图 10.27 Windows xp 系统下 `uigetfile` 的打开文件对话框

好的编程习惯

在基于 GUI 编程中，使用对话框来提供信息或要求输入数据，如果信息紧迫且不可忽略，则把对话框设为模式对话框。

10.6 菜单

菜单也可以添加到 MATLAB 的 GUI 中。使用菜单可使 GUI 显示界面上没有附加组件用户即可选择行动。

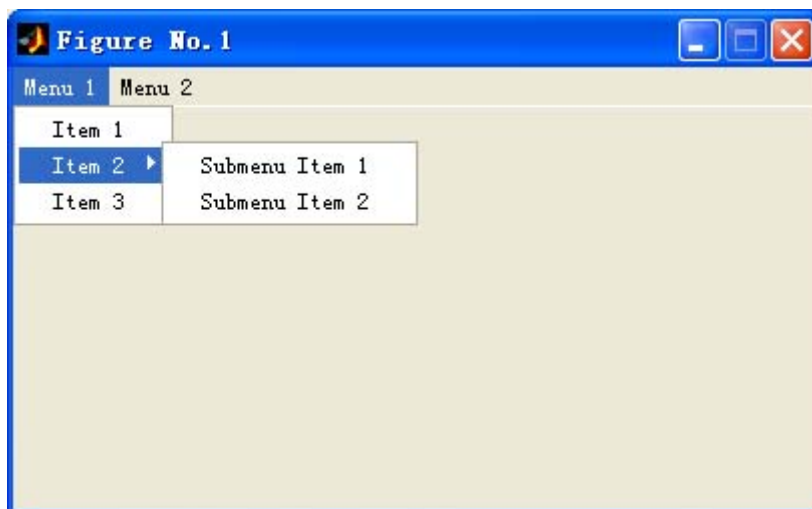
MATLAB 中有两种菜单：在图形界面顶部的菜单栏下拉的**标准菜单**与在某对象上右击的弹出的**上下文菜单**。本节我们将学习如何创建这两类菜单。

标准菜单是由 `uimenu` 对象创建的，菜单中的每一项及子菜单都是一个独立的 `uimenu` 对象，这些 `uimenu` 对象与 `uicontrol` 对象相似，有许多相同的属性，例如 `Parent`，`Callback`，`Enable` 等等，一些 `uimenu` 较重要的属性在表 10.5 中列出。

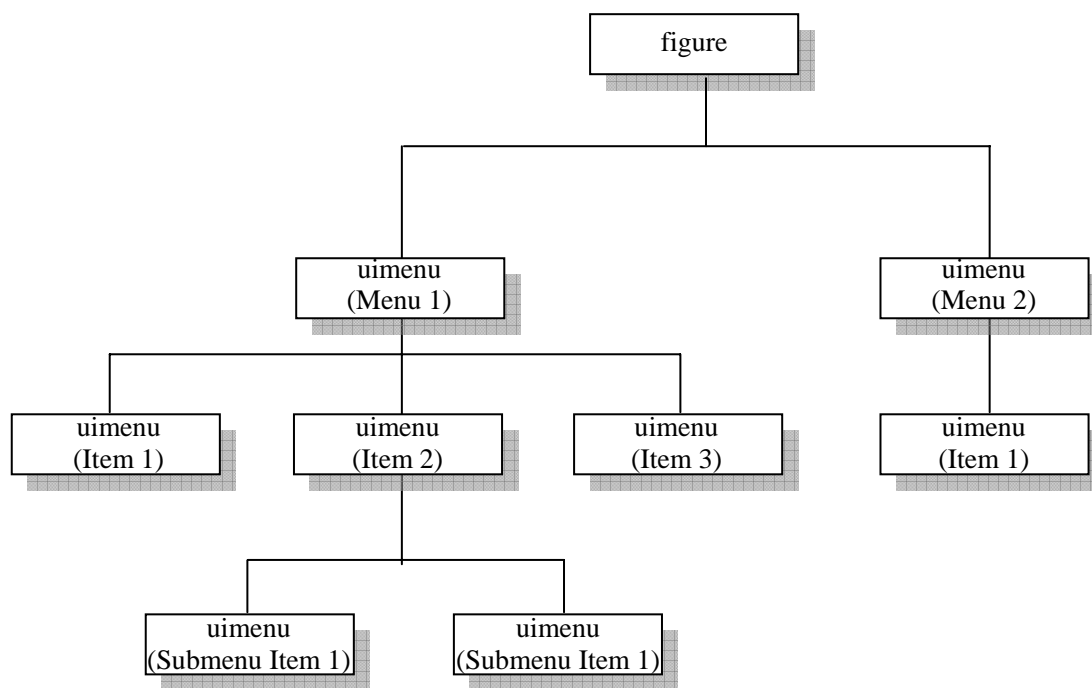
表 10.5 重要的 `uimenu` 属性

属性	描述
Accelerator	指定键盘上某个字符与该菜单项相当，用户可以使用键盘组合键 CTRL + key 激活该项。
Callback	设定该菜单项激活时调用的函数的名称与参数。如果该项是一个子菜单，则该函数在子菜单显示之前执行；如果该项没有子菜单，则该函数在鼠标释放时执行。
Checked	如果该项为 on，则在该项左边会有一个选择标记，这种属性可以用来指示菜单项在两种状态之间转换在情况，可选值为“on”和“off”。
Enable	设定该菜单项是否可选，如果不能，则不会对任意鼠标单击与键盘加速键做出响应，可选值为“on”或“off”。
Label	设定在菜单项上显示的文本，可以用“&”符号指定本项的键盘助记键并在菜单上显示出来。例如，字符串“&File”将在菜单上显示“File”并对 F 键作出响应。
Parent	本菜单项的父对象的句柄。父对象可以是一个图形或另一个菜单项。
Position	设定菜单项在菜单栏或菜单上的位置，1 代表在顶级菜单中的最右边或在子菜单中的最上边。
Separator	如果属性为“on”，则在这项上画一条分隔线，可选的值为“on”或“off”。
Tag	菜单项的名称，可用来定位菜单项。
Visible	设定本菜单项是否可见，可选值为“on”或“off”。

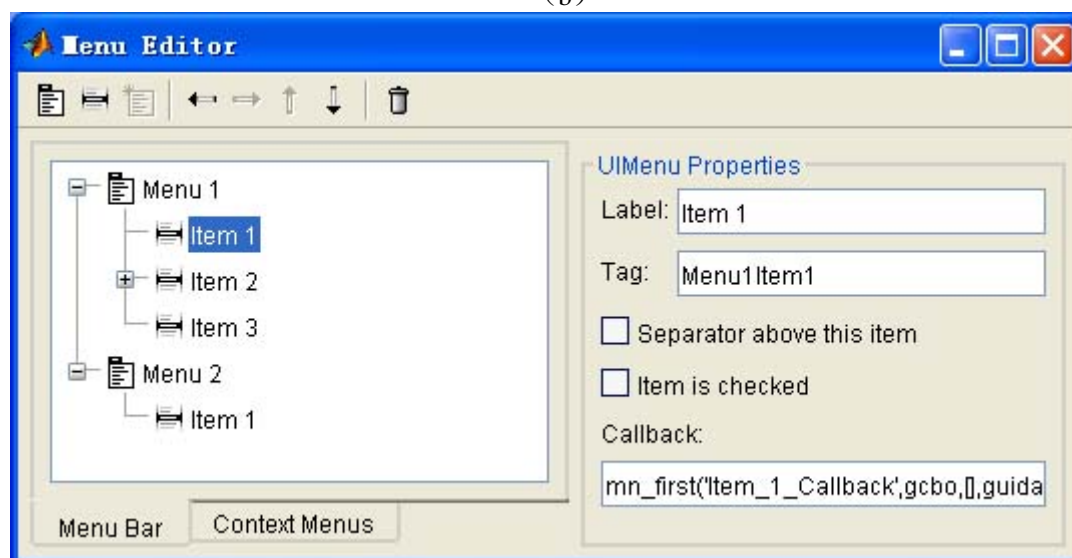
每个菜单项都是都隶属于某个父对象，而图形本身是一个最顶端的菜单。所有的隶属于同一父对象的 uimenu 显示在相同的菜单上，各项叠起来形成一个树形子菜单。图 10.28a 显示了运行中的一个典型 MATLAB 菜单，而图 10.28b 显示了组成该菜单的各项之间的关系。



(a)




(b)



(c)

图 10.28 (a) 典型的菜单结构 (b) 创建菜单的 uimenu 项之间的关系 (c) 菜单编辑器中的结构

MATLAB 使用菜单编辑器创建菜单，可以通过上界面编辑器向导顶端的  图标打开。图 10.28c 显示了产生菜单上各项的结构。附加属性在表 10.5 中列出。菜单编辑器可以用属性编辑器设置 (propedit)。

顶级上下文菜单由 `uicontextmenu` 对象创建，上下文菜单中的下一级菜单项则由 `uimenu` 对象创建，上下文菜单基本上与标准菜单相同，除了可以隶属于任何对象（坐标轴，线条，文本，图形等等）外。表 10.6 列出了 `uicontextmenu` 一些重要的属性。

10.6.1 禁用默认菜单

每个 MATLAB 图形默认都会出现一组标准菜单，如果你要删除这些菜单并创建自己的菜单，首先就必须把默认菜单关闭掉，是否显示默认菜单是由图形的 `Menubar` 的属性控

制的。或选值为“figure”和“none”。如果设为“figure”，那么将显示菜单，如果设为“none”，默认菜单将会抑制。创建 GUIs 时你可以使用属性编辑器来设置。

表 10.6 uicontextmenu 几个重要属性

属性	描述
Callback	设定上下文菜单激活时调用的函数的名称和参数。在菜单显示之前执行。
Parent	上下文菜单的父对象的句柄。
Tag	上下文菜单的名称，可以用来定位它。
Visible	设定上下文菜单是否可见。这个属性会被自动设置，通常不需要修改。

10.6.2 创建自定义菜单

为 GUI 创建自定义标准菜单基本上分三步处理：

1. 第一，在菜单编辑器中创建一个新的菜单结构。使用菜单编辑器定义结构，给每个菜单项填上 Label 标签和独一的 Tag 名称，同时也可以为菜单项手动创建回叫字符串，这需要些技巧——为菜单项创建回叫的最好办法是对 uicontrol 自动创建的回叫进行研究，并把它当成例子学习。uimenu 回叫字符串的正确形式如下：

```
MyGui('MenuItemTag_Callback', gcbo, [], guidata(gcbo));
```

这里你要把 MyGui 替换成你自己的 GUI 名称，把 MenuItemTag 设为菜单项的名称。

2. 第二，使用属性编辑器（propedit）编辑每个菜单项的属性，最重要的几个需要设置的是 Callback，Label 和 Tag，这些都可以通过菜单编辑器进行设置，因此通常并不需要属性编辑器。当然，如果你要设置表 10.5 中列出的其它属性，就必须使用属性编辑器了。

3. 第三，为菜单项编写代码实际回叫函数所必须完成的功能。你必须为菜单项手动创建回叫函数。

在本章节末尾，将以一个例子演示创建菜单的过程。

编程隐患

与 GUI 对象不同，MATLAB 并不自动为菜单项创建回叫字符串与存根，必须手动完成。

编程隐患

菜单项的属性中只有 Label，Tag，Callback，Checked 和 Separator 等属性可以在菜单编辑器中设置，如果要设置其它属性，就必须使用图形中的属性编辑器，选择正确的菜单项进行编辑。

10.6.3 加速键与键盘助记键

MATLAB 菜单支持键盘加速键与助记键。加速键是“CTRL+key”组合键，不需要先打开菜单就能使某菜单项执行。例如，加速键“o”可能分配给菜单中的“File/Open（文件/打开）”，如果这样，键盘组合键“CTRL+o”将会使“File/Open”项的回叫函数执行。

有一些组合键盘为操作系统所保留，因操作系统（PC 或 UNIX）而异，查阅 MATLAB 在线文档，看看哪些组合键适合你的计算机。

加速键在 uimenu 对象的 Accelerator 属性中设定。

键盘助记键仅是一个字母，在菜单被打开后敲击该字母就会执行该菜单项。分配菜单项的助记键被加以下划线。如果是顶级菜单，则必须按住键盘的 ALT 键再同时按下该字母，一旦顶级菜单已经打开，则只需敲击该字母就能执行该项。

图 10.29 演示了键盘助记键的使用。**F**ile（文件）菜单使用 ALT+f 打开，一旦菜单打开，只需敲“x”即可执行 Exit（退出）项。

助记键是通过在 Label 属性中的目标字母前面加上“&”字符来定义的。“&”不会被显示出来，后面是字符会被加以下划线，它就成了助记键。例如，图 10.29 中的 Exit（退出）菜单项的 Label 属性值是“E&xit”。

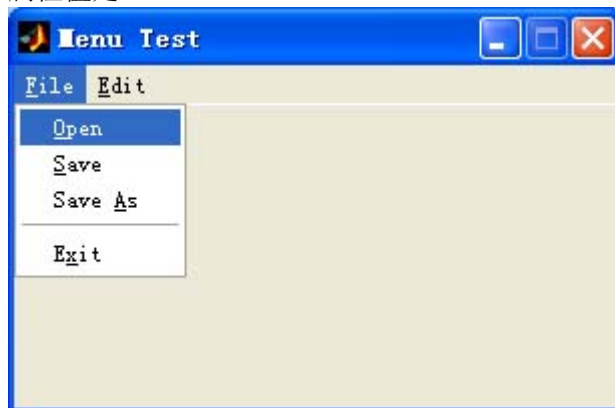


图 10.29 通过 ALT+f 打开显示的菜单，要退出只需单击“x”即可

10.6.4 创建上下文菜单

上下文菜单与创建普通菜单的方式相同，除了顶级菜单项是一个 `uicontextmenu` 外。`uicontextmenu` 的父对象必须是一个图形。但上下文菜单也可以与任何图象对象绑定起来并对鼠标右键做出响应。上下文菜单通过在菜单编辑器选择“Context Menu”项来创建。一旦上下文菜单创建后，就可以在它下面创建任意数量的菜单项。

要把上下文菜单与特定对象绑定起来，你必须把对象的 `UiContextMenu` 属性设置成 `uicontextmenu` 的句柄。这通常通过属性编辑器来完成，但也可以使用下面的 `set` 命令来做。假如 `Hcm` 是一个上下文菜单的句柄，下面的语句将把这个菜单与用 `plot` 命令创建的线条绑定起来。

```
H1 = plot(x,y);  
set(H1, 'UiContextMenu', Hcm);
```

在下面的例子中，我们创建一个上下文菜单，并把它与一个图对象联合起来。

例 10.2 绘制数据点

编写程序，打开用户指定的数据文件，然后把文件中数据绘点画线。程序必须包含一个文件（File）菜单，有打开（Open）和退出（Exit）项，还必须包含一个绑定到线条上的上下文菜单，菜单的内容可以改变线条的风格。假设文件的数据是以（x，y）对的形式出现，每行一对。

解决方案

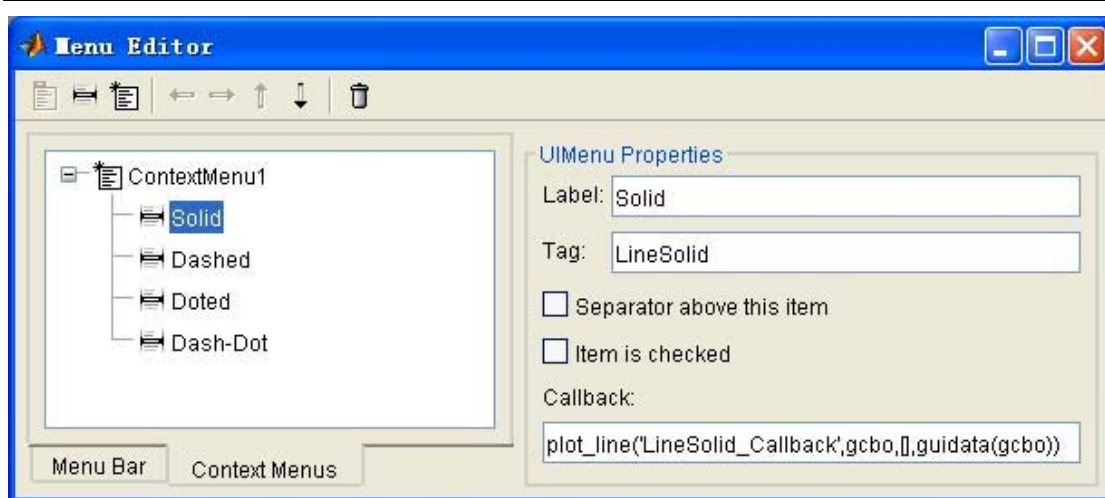
本程序必须包含一个有打开（Open）和退出（Exit）菜单项的标准菜单和一个用来绘制数据的坐标轴，也必须包含一个用来设定不同线条风格的上下文菜单，还必须把上下文菜单绑定到描绘的线条中，上下文菜单项中应该包含实线、虚线、点线和虚点线等风格。

创建本程序的第一步是使用向导创建所需要的 GUI，本例中界面只有一组坐标轴（见图 10.30a）。然后，我们用菜单编辑器创建文件菜单，这个菜单包含了 Open（打开）和 Exit（退出）项，如图 10.30b。注意我们必须使用菜单编辑器设置 Label，Tag 和每个菜单项的 callback 字符串。还必须为 File（文件）指定助记键“F”，为 Open（打开）指定“O”及为 Exit（退出）指定“x”，并在 Exit 与 Open 之间设置分隔符，如图 10.30 所示。

The screenshot shows the MATLAB Menu Editor interface. The title bar is blue and reads "Menu Editor". Below it is a toolbar with icons for file operations (new, open, save, print, delete) and navigation (back, forward, up, down). The main workspace is divided into two panes: "Menu Bar" and "Context Menus". The "Menu Bar" pane displays a hierarchical tree structure. The root node is "&File", which has two sub-items: "&Open" and "E&xit". The "E&xit" item is currently selected and highlighted with a blue background. The "Context Menus" pane is currently empty. To the right of the workspace is the "UI Menu Properties" panel. It contains the following settings:

- Label:** E&xit
- Tag:** mFileExit
- ☒ Separator above this item
- ☐ Item is checked
- Callback:** plot_line('mFileExit_Callback',gcbo,[],gui

图 10.30 (a) plot_line 的界面 (b) 菜单编辑器中的文件菜单



(c)

图 10.30 续 (c) 菜单编辑器中的上下文菜单

下一步，我们采用菜单编辑器创建上下文菜单。这个菜单以 `uicontextmenu` 对象开始，带有四个选项（如图 10.30c），同样，我们设置各项的 `Label`，`Tag` 和 `callback` 字符串。

到这里，我们把 GUI 保存为 `plot_line.fig`，`plot_line.m` 将被自动创建。然而，每项哑元回叫函数不会被自动创建，必须手工完成。

GUI 创建后，必须为菜单建立六个回叫函数。最难的回叫函数是 `File/Open` 菜单项的函数，这个回叫函数必须提示用户输入文件名（使用 `uigetfile` 对话框），打开文件，读取数据，把它保存到 `x` 和 `y` 数组中，最后关闭文件。接着，绘制线条，以应用程序数据保存线条句柄以便后面我们可以修改线条风格。最后，回叫函数必须把上下文菜单联结在一起。`mFileOpen_Callback` 函数如图 10.31 所示。注意函数使用对话框告知用户文件打开出错。

剩下的回叫函数相当简单，`mFileExit_Callback` 函数仅仅是关闭图形，线条风格函数仅仅设置线条风格。当用户鼠标在线条上面右击时，上下文菜单会弹出。如果用户从弹出的菜单选择一项，相应的回叫函数将使用保存的线条句柄更改属性。这五个函数如图 10.32 所示。

程序最终如图 10.33。在你自己的电脑上实践，确保它正确运行。

```
% -----
function mFileOpen_Callback(hObject, eventdata, handles)
% Get the file to open
[filename, pathname] = uigetfile('*.dat','Load Data');
if filename ~= 0
    % Open the input file
    filename = [pathname filename];
    [fid, msg] = fopen(filename, 'rt');
    % Check to see if the open failed.
    if fid < 0
        % There was an error -- tell user.
        str = ['File' filename ' could not be opened.'];
        title = 'File Open Failed';
        errordlg(str, title, 'modal');
    else
        % File opened successfully, Read the (x,y) pairs from
        % the input file. Get first (x,y) pair before the
        % loop starts.
        [in, count] = fscanf(fid, '%g', 2); %%%% 读取数据
        ii = 0;
        while ~feof(fid)
            ii = ii + 1;
            x(ii) = in(1);
```

```

        y(ii) = in(2);
        %Get next (x,y) pair
        [in, count] = fscanf(fid, '%g', 2); %%%% 读取数据
    end
    % Data read in. Close file.
    fclose(fid);
    % Now plot the data.
    hline = plot(x,y,'LineWidth',3); %%%% 绘线条
    xlabel('x');
    ylabel('y');
    grid on;
    % Associate the context menu with line
    set(hline,'Uicontextmenu', handles.ContextMenu1); %%%% 设置上下文菜单
    % Save the line's handle as application data
    handles.hline = hline; %%%% 把句柄保存为应用程序数据
    guidata(gcbf, handles);
end
end

```

图 10.31 File/Open 回叫函数

```

% -----
function mFileExit_Callback(hObject, eventdata, handles)
close(gcbf);
% hObject    handle to mFileExit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function LineSolid_Callback(hObject,eventdata,handles)
set(handles.hline,'LineStyle','-');

function LineDashed_Callback(hObject,eventdata,handles)
set(handles.hline,'LineStyle','--');

function LineDotted_Callback(hObject,eventdata,handles)
set(handles.hline,'LineStyle',':');

function LineDashDot_Callback(hObject,eventdata,handles)
set(handles.hline,'LineStyle','-');
% -----

```

图 10.32 plot_line 中的其它函数

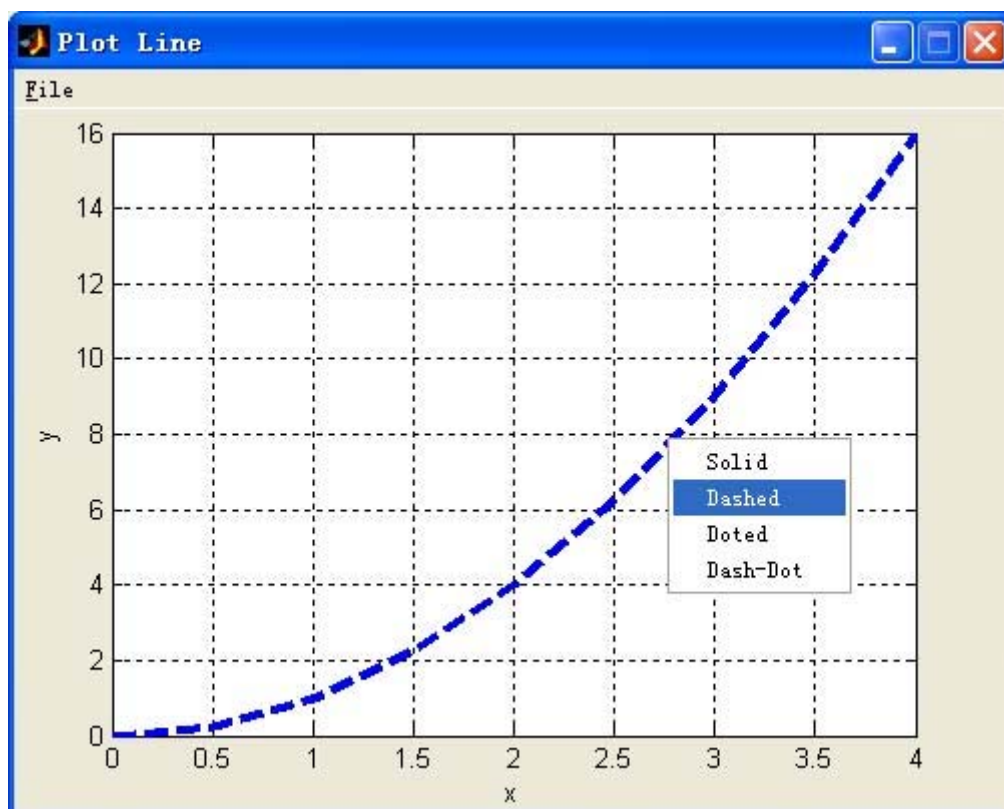


图 10.33 plot_line 产生的 GUI

测试 10.1

本测试检查你是否已经明白了 10.1 节到 10.6 节介绍的概念。如果你完成本测试有困难，重读以上章节，问你的老师，与你的同学讨论。可在本书后面找到本测试的答案。

1. 列出本章中介绍到的图形组件的类型。每一个目的是什么？
2. 什么是回叫函数？MATLAB 的 GUIs 是如何使用回叫函数的？
3. 描述创建基于 GUI 程序的步骤。
4. 描述数据结构句柄的目的。
5. MATLAB GUI 中是如何保存应用程序数据的？为什么要保存 GUI 中的应用程序数据？
6. 如何使图对象不可见？如何关闭图对象对鼠标的单击或键盘输入的响应？
7. 本章中哪些 GUI 组件能对鼠标作业响应？哪些能对键盘输入作出响应？
8. 什么是对话框？如何创建对话框？
9. 模式对话框与无模式对话框之间的区别是什么？
10. 标准菜单与上下文菜单之间的区别是什么？
11. 什么是加速键？什么是助记键？

10.7 创建高效 GUIs 的技巧

本节列出一些创建高效图形用户界面（GUI）的技巧。

10.7.1 工具提示

MATLAB 从 5.2 版本开始支持工具提示——即当鼠标在某对象上面停留一小会儿后弹出的一个小帮助窗口，它是 `uicontrol` GUI 对象中的一个属性，通常用来对 GUI 上的对象提供快速帮助。

工具提示通过在对象的 `TooltipString` 属性中设置你想要显示的内容来定义。在本章末练习中，你会被要求创建工具提示。

好的编程习惯

对 GUI 组件设置工具提示，为用户提供关于该组件功能的有用线索。

10.7.2 伪代码（p 码，pcode）

在程序执行过程中第一次执行函数时，MATLAB 把函数编译（或解析）成称为 `pcode`（伪代码的简称）的中间代码，然后执行它的运行时解析程序中执行伪代码。一旦函数被编译，它就留中 MATLAB 的内存中，能够被重复执行而无需重新编译。不过，当 MATLAB 下次执行时，函数又得重新编译。

这种初期编译带来的开销（不利因素）相对较小，不过随着函数越来越大，这种开销变得越来越重要。由于定义 GUI 的函数通常相当大，基于 GUI 的程序编译开销与其它类型的程序相比也相对较大。换句话说，由于初期编译，GUI 程序运行得更慢。

幸运的是，我们可以避免这种开销：把 MATLAB 函数及脚本文件编译成伪代码，保存的伪代码文件可以在将来立即执行。执行伪代码文件节省了初期编译时间，使程序运行得更快。

MATLAB 采用 `pcode` 命令创建伪代码文件，这个命令采用下面的形式之一：

```
pcode fun1.m fun2.m fun3.m ...
```

```
pcode *.m
```

第一种形式编译给定名称的文件，第二种形式编译当前目录下所有的 M 文件。编译结果以“p”保存。例如，你编译了文件 `foo.m`，那么输出将保存在 `foo.p` 文件中。

如果同一函数既存在于 M 文件中也存在于 p 文件中，MATLAB 将自动执行 p 文件中的版本，这是由于该版本更快。然而，如果你修改了 M 文件，你一定要记得重新编译，否则程序将仍然执行旧代码。

把文件编译成伪代码也有其它优点。在伪代码的形式把发布给其他人可以保护你在源代码上的投资。它们可以自由执行，但别人就没那么容易重建文件得到你的（设计）理念。

好的编程习惯

一旦程序工作正常，用 `pcode` 命令预编译 M 文件，以便提高程序运行速度。

编程隐患

如果更改了已经编译成伪代码的 M 文件，记得要重新编译。否则，你仍然是那些老的、没有修改的代码。

10.7.3 附加提高

基于 GUI 的程序可能比我们在本章所简单介绍的更复杂。作为本章使用过的 `Callback`

属性的附加内容，`uicontrols` 支持三种类型的回叫：`CreateFcn`、`DeleteFcn` 和 `ButtonDownFcn`。MATLAB 图形同样支持三种重要类型的回叫：`WindowButtonDownFcn`、`WindowButtonMotionFcn` 和 `WindowButtonUpFcn`。

`CreateFcn` 属性定义了对对象创建时自动调用的回叫。这种属性允许程序员在程序运行期间对象创建时自定义该对象。由于这种回叫在对象完成定义之前执行，程序员必须在对象创建之前指定描述对象根属性的函数。例如，下面的语句将会使得 `function_name` 函数在每次 `uicontrol` 对象创建之前执行。函数将会在 MATLAB 创建对象的属性之后被调用，因此它们（对象的属性）在函数执行之前都是可用的。

```
Sset(0,'DefaultUicontrolCreateFcn','function_name')
```

`DeleteFcn` 属性定义了对对象被销毁时自动调用的函数，它在对象的属性被销毁之前执行，所以它们（对象的属性）在函数执行之前仍可用。这个回叫使程序员有机会去做一些自定义的清理工作。

`ButtonDownFcn` 属性定义了当鼠标在 `uicontrol` 周围 5-pixel 内按按钮时自动调用的回叫。如果鼠标按钮按在 `uicontrol` 上，执行 `Callback` 函数，否则，如果是在边界附近，则执行 `ButtonDownFcn`。如果 `uicontrol` 不可用，此时即使鼠标击在控件上，也会执行 `ButtonDownFcn`。

`WindowButtonDownFcn`、`WindowButtonMotionFcn` 和 `WindowButtonUpFcn` 等图形级的回叫函数允许程序员实现动画或拖放等特性，这是由于这些函数可以检测鼠标按下后的初始、中间和最终位置。这些函数跳出本书讨论的范围，但是值得学习知道。参阅 MATLAB 用户文档《创建图形用户界面》一卷获得这些回叫函数的描述。

例 10.3

创建柱状图

编写程序，打开用户指定的数据文件，按文件中的数据计算柱状图。程序应该能计算文件中数据的平均差，中位差和标准差。程序必须包含文件菜单（含打开和退出项），还必须提供用户更改图形柱数的方法。

选择另一种颜色替换图形和文本标签的背景色，为菜单项设置键盘加速键，在适当的地方添加工具提示。

解决方案：

本程序应包含带有打开和退出项的标准菜单、一个用来绘制柱状图的坐标轴，六个文本域，其中三个用来显示平均差、中位差和标准差，三个用来做为标签。程序还必须包含一个标签和一个编辑框，用来让用户更改柱状图中的柱数。

我们选取淡蓝色[0.6 1.0 1.0]作为 GUI 的背景色，为了达到目的，应在设计时在属性编辑器中把这个代表颜色的向量加进图形的“Color”属性和文本标签的“BackgroundColor”属性。

第一步，使用向导 `guide` 设计 GUI（如图 10.34a 所示），然后，用属性编辑器编辑界面中的文本域和编辑框。必须为这些控件设置独一的名称，以便在后面的回叫函数中定位。下一步，使用菜单编辑器创建文件菜单（如图 10.34b）。最后，把 GUI 保存为 `histGUI`，向导会自动创建 `histGUI.fig` 和 `histGUI.m`。

`histGUI.m` 保存后，必须编辑它，在句柄结构中添进表示柱状图的柱数的应用程序数% Choose default command line output for histGUI

```
handles.output = hObject;
```

```
% Add the number of bins to this struct.
```

```
handles.nbins = str2num(get(handles.NBins,'String'));
```

```
% Update handles structure
```

```
guidata(hObject, handles);
```

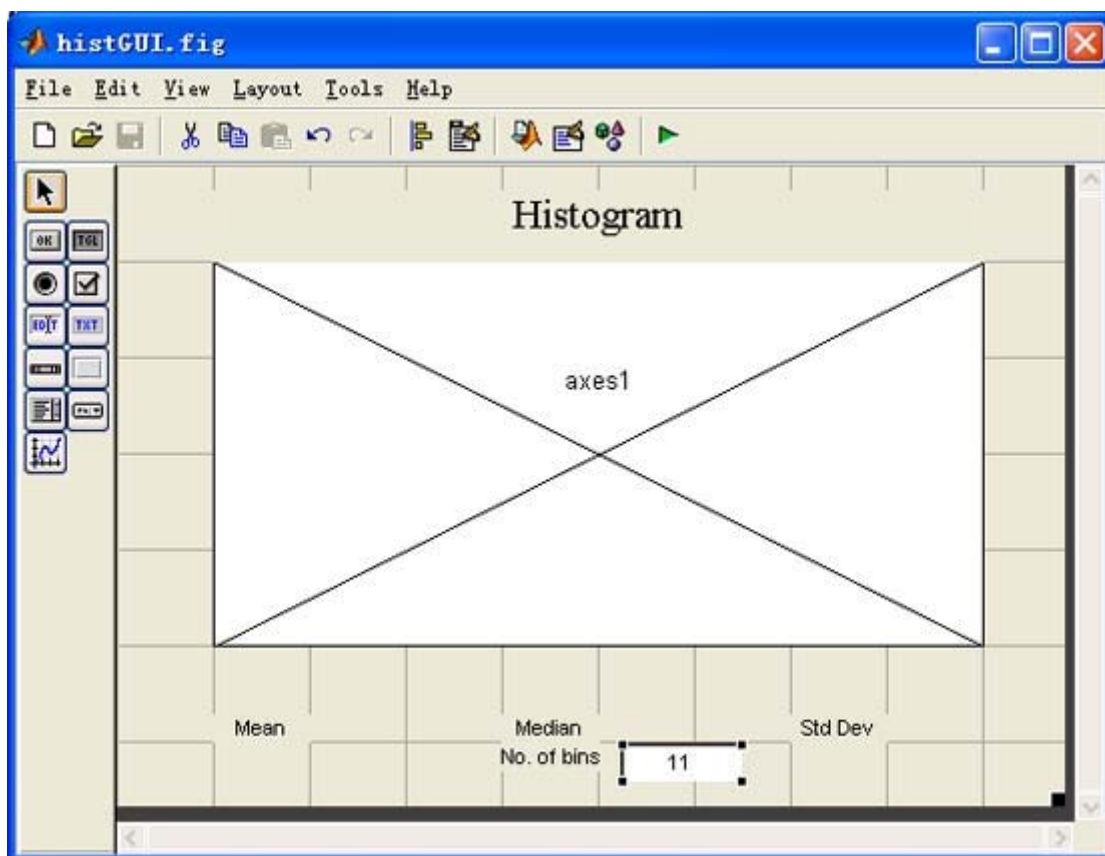


图 10.34 (a) histGUI 的布局

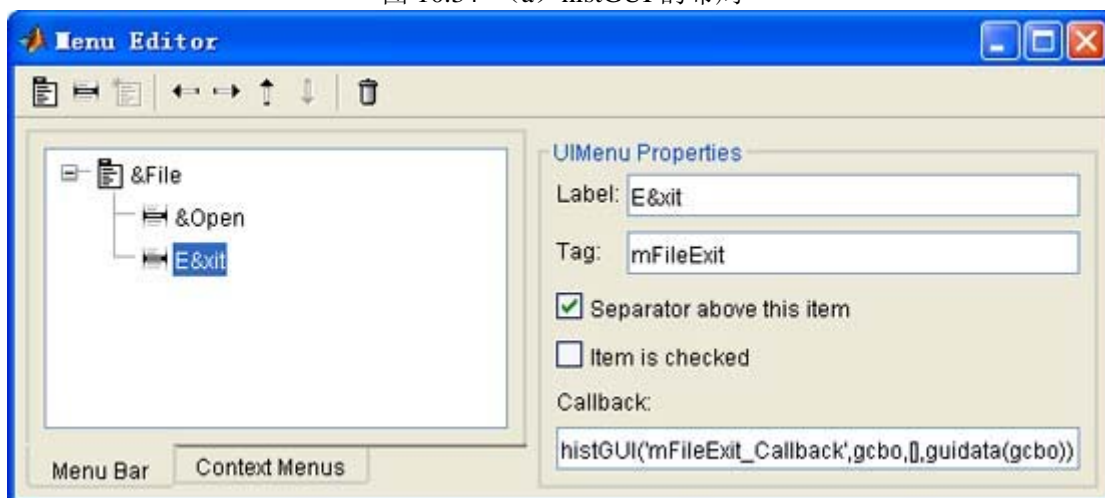


图 10.34 (b) 菜单编辑器中的文件菜单

接着，为“文件/打开”、“文件/退出”菜单项和“柱数”编辑框创建回叫函数。只有最后一个函数才会自动创建——菜单项的回叫必须手工添加。

“文件/打开”回叫函数提示用户选择文件，然后从文件中读取数据，计算并显示柱状图，更新统计文本域。注意文件中的数据也必须保存到 handles 结构中，以便用户更改柱数时可以重新计算。处理这些的步骤的函数如下：

```
function mFileOpen_Callback(hObject,eventdata, handles)
% Get file name
[filename, path] = uigetfile('*.dat;*.abc', 'Load Data File');
if filename ~= 0
    % Read data
    x = textread([path filename], '%f');
    % Save in handles structure
```

```

handles.x = x;
guidata(gcbf, handles);
% Create histogram
hist(handles.x, handles.nbins);
% Set axis labels
xlabel('\bfValue');
ylabel('\bfCount');
% Calculate statistics
ave = mean(x);
med = median(x);
sd = std(x);
n = length(x);

% Update fields
set(handles.MeanData,'String',sprintf('%7.2f',ave));
set(handles.MedianData,'String', sprintf('%7.2f',med));
set(handles.StdDevData,'String', sprintf('%7.2f',sd));
set(handles.TitleString,'String',['Histogram (N = ' int2str(n) ' )']);
end

```

“文件/退出”很简单，仅是关闭图形而已。

```

function mFileExit_Callback(hObject, eventdata, handles)
close(gcbf);

```

NBins 的回叫函数必须读取输入的数字，取最接近的整数，然后又把它显示在编辑框中，并重新计算和显示柱状图。注意柱数必须重新保存到 `handles` 结构中，以使用户打开新的数据文件时可以使用。实现这些步骤的回叫函数如下：

```

function NBins_Callback(hObject, eventdata, handles)
% Get number of bins, round to integer, and update field
nbins = str2num(get(gcbo,'String'));
nbins = round(nbins);
if nbins < 1
    nbins = 1;
end
set(handles.NBins, 'String', int2str(nbins));
% sprintf('nbins = %d' nbins);
% Save in handles structrue
handles.nbins = nbins;
guidata(gcbf, handles);

```

```

% Re-display data, if available
if handles.nbins > 0 & ~isempty(handles.x)

```

```

    % Create histogram
    hist(handles.x, handles.nbins);

```

```

end

```

程序最终如图 10.35 所示，试验该程序能否正确运行。

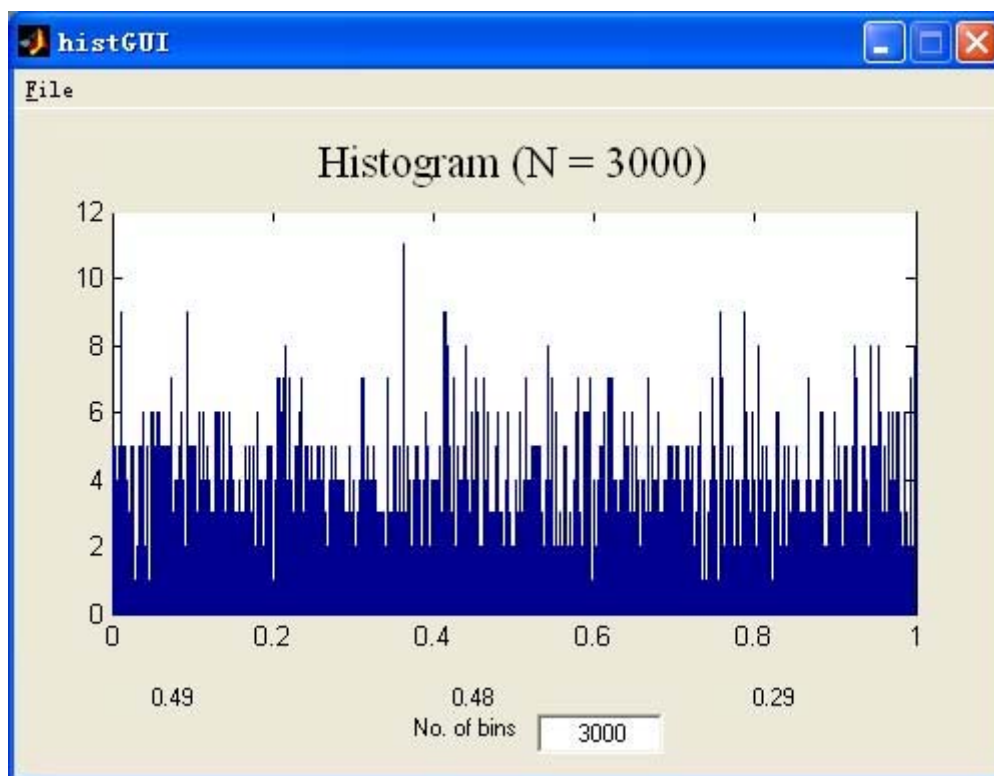


图 10.35 histGUI 程序产生的 GUI（数据采用随机产生的 3000 个 0-1 之间的数）

10.8 总结

本章我们学习了如何创建 MATLAB 图形用户界面，一个 GUI 的三个基本部分是**组件**（uicontrols, uimenu 和 uicontextmenus），包含它们的**图形**和以鼠标及键盘作出响应实现动作的**回叫**。

标准 GUI 组件由 uicontrol 创建，它包括文本域，编辑框，框架，按钮，形状按钮，复选框，单选按钮，下拉菜单，列表框和滑动条。由 uimenu 和 uicontextmenu 创建的标准 GUI 组件是标准菜单和上下文菜单。

所有的这些组件都可以使用向导 guide（GUI 开发环境工具）放置到图形上。一旦 GUI 布局完成，用户必须用属性编辑器编辑对象属性，然后为每个 GUI 对象编写实现其动作的回叫函数。

对话框是用来显示信息或从用户处获取输入的特殊图形。对话框通常用来显示错误，提供警告，询问问题或获取用户输入。它们也用来选择文件与打印机属性。

对话框有有模式与无模式之分。模式对话框在它消失之前不允许用户访问应用程序的其它窗口，而无模式对话框没有这个限制。模式对话框习惯上用来显示需要用户急切注意、不可忽略的警告或错误信息。

菜单也可以添加到 MATLAB 的 GUI 中。菜单可以使界面没有太多组件又可以实现动作。菜单在处理那些不会经常用到的选项而又不想把 GUI 堆得乱七八糟时非常有用。菜单采用菜单编辑器来创建，对每个菜单项，用户必须使用菜单编辑器设置其 Label 标签、Tag 名称和回叫字符串。与 GUI 组件不同，guide 不会自动为菜单项创建回叫字符串，用户要完全负责定义回叫字符串和实现其函数体。

加速键与键盘助记键可以用来加快窗体操作。

10.8.1 好的编程习惯总结

当使用 MATLAB 的 GUI 时，应该遵循下面的指导语：

1. 用 `guide` 对一个新的用户图形界面进行布局，并用属性编辑器对每一个组件的初始属性进行设置，例如显示在组件上的文本，组件的颜色，还有回叫函数的名字。
2. 用 `guide` 创建完一个用户图形界面后，人工编辑产生的函数，增加注释，描述这个函数的目的和组件，执行回叫函数的代码。
3. 把 GUI 应用程序数据存储到 `handles` 结构中，以便任意的一个回叫函数都可以应用它。
4. 如果你修改了 `handles` 结构中的任何 GUI 应用数据，确保在函数退出之前保存了调用 `guidata` 的结构。
5. 在基于 GUI 编程中，使用对话框来提供信息或要求输入数据，如果信息紧迫且不可忽略，则把对话框设为模式对话框。
6. 对 GUI 组件设置工具提示，为用户提供关于该组件功能的有用线索。
7. 一旦程序工作正常，用 `pcode` 命令预编译 M 文件，以便提高程序运行速度。

10.8.2 MATLAB 总结

下面的总结列出了本章中讨论到的 MATLAB 命令与函数，还有它们的简要介绍。同时，请参阅表 10.2，10.3，10.5 和 10.6 等图形对象属性总结。

命令与函数

<code>axes</code>	创建一组坐标轴的函数
<code>dialog</code>	创建一个通用对话框
<code>errordlg</code>	显示错误信息
<code>helpdlg</code>	显示帮助信息
<code>findobj</code>	查找一个或多个属性匹配的 GUI 对象
<code>gcbf</code>	取得回叫图形
<code>gcbo</code>	取得回叫对象
<code>guidata</code>	保存图形中的 GUI 应用程序数据
<code>guihandles</code>	从保存在图形中的应用程序数据中取得 <code>handles</code> 结构
<code>guide</code>	GUI 开发环境工具
<code>inputdlg</code>	从用户处获取输入数据的对话框
<code>printdlg</code>	打印对话框
<code>questdlg</code>	询问问题的对话框
<code>uicontrol</code>	创建 GUI 对象的函数
<code>uicontextmenu</code>	创建上下文菜单的函数
<code>uigetfile</code>	获取输入文件的对话框
<code>uimenu</code>	创建标准菜单、或者在标准菜单或上下文菜单中创建菜单项的函数
<code>uiputfile</code>	选择输出文件对话框
<code>uisetcolor</code>	显示颜色选择对话框
<code>uisetfont</code>	显示字体选择对话框
<code>warndlg</code>	显示警告对话框

10.9 练习

10.1 描述 MATLAB 中创建 GUI 所需的步骤。

10.2 回叫函数如何工作？回叫函数如何定位其所要操作的对象或图形。

10.3 创建一个 GUI 程序，使用下拉菜单选择 GUI 的背景颜色。

10.4 创建一个 GUI 程序，使用标准菜单选择 GUI 的背景颜色。

10.5 写一个 GUI 程序，绘制函数 $y(x) = ax^2 + bx + c$ 的图象。程序应该包含一组坐标轴，包含用来输入 a , b , c , x 的最大值和最小值的 GUI 元素，还必须有 GUI 元素的工具提示。

10.6 修改 10.5 的 GUI，加入菜单，菜单中包含两个子菜单，一个用来选择图象线条的颜色，一个用来选择图象线条的风格。必须有选择标记显示所选择的项。同时，还应该有一个“退出”项，当用户选择退出时，弹出一个模式询问对话框，显示“Are You Sure?”以便进一步确认。

10.7 修改 10.4.8 节中的列表框例子，允许对列表框中的项多选，当点击“Selected”按钮时，文本域中显示列表框中所有选择的项。

10.8 **随机数分布** 创建一个 GUI，显示不同的类型的随机数分布。程序应该能够产生 200,000 个随机数并用 hist 创建柱状图显示出来。注意正确设置标题和坐标轴的标签。

程序应能支持统一、高斯和瑞利分布，用下拉菜单来选择分布类型，此外，还必须提供一个编辑框，以便用户可以修改柱状图的柱数。确保用户的输入合法（必须为正整数）。

10.9 修改例 10.1 中的温度转换 GUI，添加一支“温度计”，温度计的红色“液柱”显示了当前摄氏温度值，范围在 0-100°C 之间。

10.10 修改练习 10.9 中温度转换 GUI，允许用鼠标调节显示的温度。（警告：本练习需要用到本章中没有讨论到的材料，在线参阅 figure 图形对象的 CurrentPoint 属性。）

10.11 **最小二乘方拟合** 创建一个 GUI，从文件中读取数据对，对数据进行最小二乘方拟合。数据以 (x, y) 格式储存在磁盘文件中，每行一组数据。用 MATLAB 的 playfit 函数绘制最小二乘方拟合图线，同时还绘制原始数据图线。包含两个菜单：“文件”和“编辑”，文件菜单含有“打开”和“退出”两项，在退出程序之间用户必须收到“Are You Sure?”信息。编辑菜单允许用户自定义显示，包括线条风格、线条颜色和网格显示状况。

10.12 修改练习 10.11，在编辑菜单添加“首选项”，允许用户取消退出时提示“Are You Sure?”。

10.13 修改练习 10.12，允许程序读写初始化文件。该文件应包含程序前一次运行时的线条风格，线条颜色，网格选择 (on/off) 和退出时的提示设置。这些设置应该在用户从“退出”菜单中退出程序时自动保存，下次程序再运行时被自动加载读取。