第9章
图形用户界面设计（GUI）

图形用户界面简介

9.1 图形用户界面开发环境(GUIDE)

9.2 控件对象及属性

9.3 编写 GUI 代码

9.4 程序菜单设计

9.5 程序对话框设计

9.6 程序设计实例

用户界面是用户与计算机进行信息交流的方式。计算机在屏幕显示图形、文本、声音等。用户通过输入设备（如：键盘、鼠标、跟踪球、绘制板或麦克风），与计算机通讯。用户界面设定了如何观看和如何感知计算机、操作系统或应用程序。通常，多是根据悦目的结构和用户界面功能的有效性来选择计算机或程序。

图形用户界面（**GUI**）是指由窗口、菜单、图标、光标、按键、对话框和文本等各种图形对象组成的用户界面。它让用户定制用户与**Matlab**的交互方式，而命令窗口不是唯一与**Matlab**的交互方式。

用户通过鼠标或键盘选择、激活这些图形对象，使计算机产生某种动作或变化。

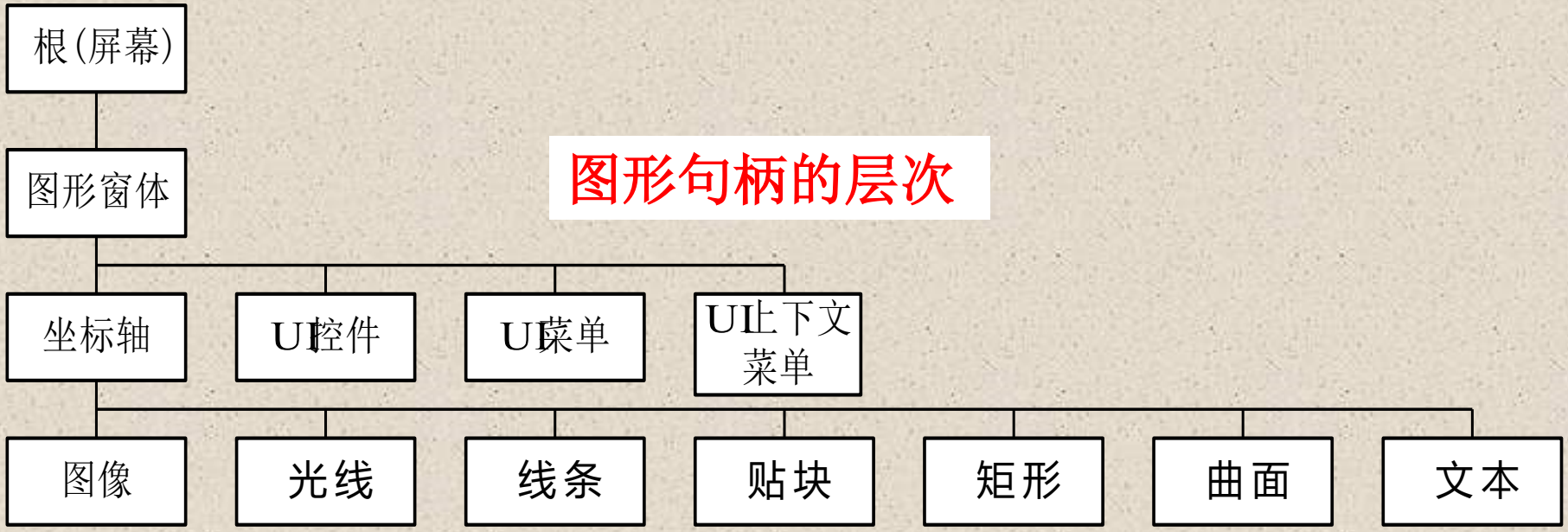
基本图形对象分为控件对象和用户界面菜单对象，简称控件和菜单。

用户菜单、用户控件和对话框是和坐标轴处于同一层次的，都是图形窗口的子对象。

图形句柄入门

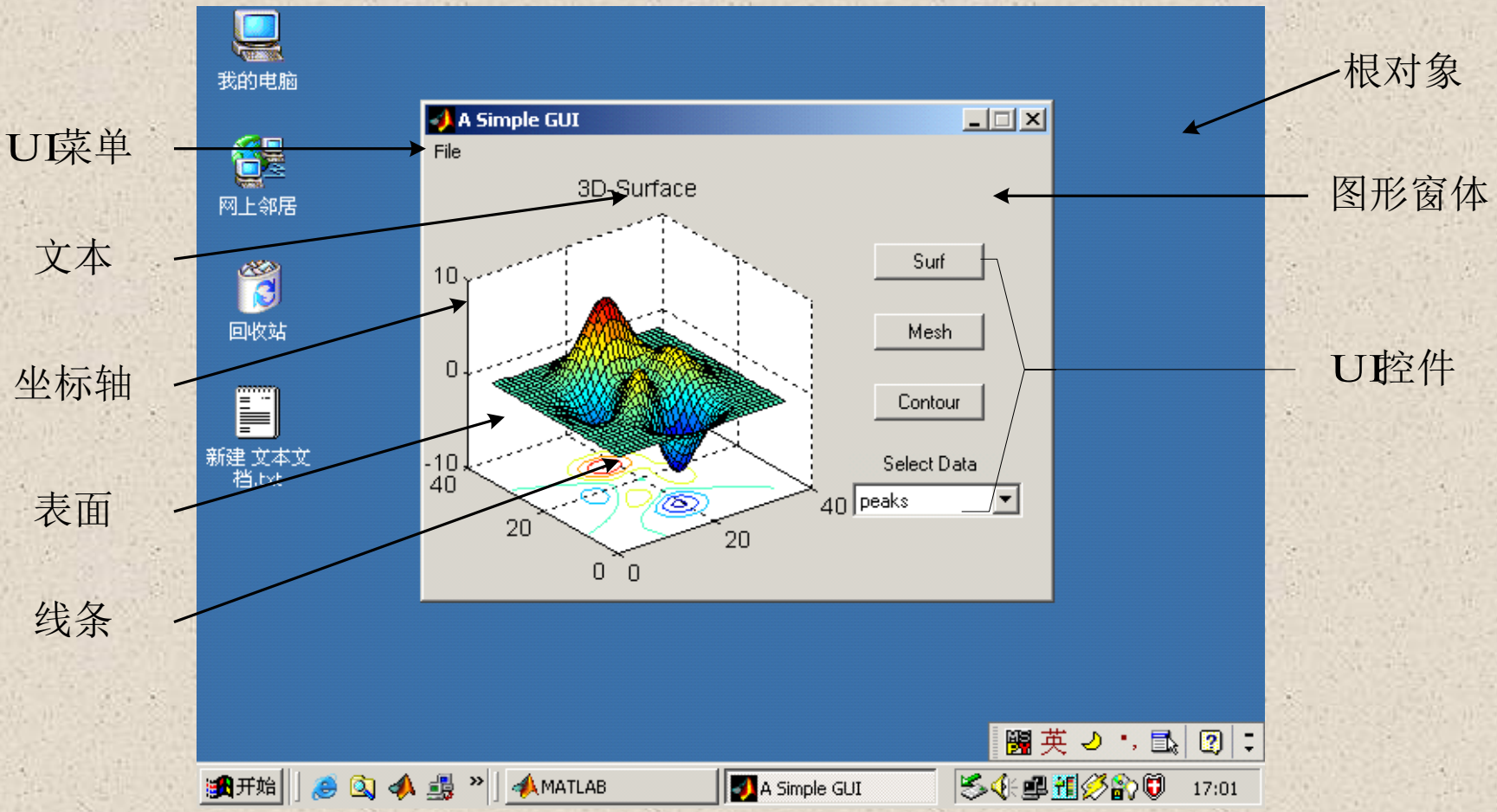
之前我们介绍了很多MATLAB可视化函数，这些函数都是将不同的曲线或者曲面绘制在图形窗体中，而图形窗体也就是由若干图形对象组成的可视化的图形界面。在 MATLAB 环境中每一个图形对象都有一个相应的句柄，这些句柄帮助系统标识这些对象，获取或者设置它们的属性。理解图形对象句柄也是进行图形界面创建的前提之一，所以首先简要介绍图形对象句柄的概念，以及图形句柄的使用方法。

MATLAB的图形对象是按照一定的层次排列的，如下图所示



图形句柄的层次

在上图中，除了最上一层的屏幕(root)对象以外，每一种对象都具有自己的父层次对象，即对象的上一层次的对象，而自己下一层次的对象都被称为子对象。


UI菜单
文本
坐标轴
表面
线条
根对象
图形窗体
UI控件
我的电脑
网上邻居
回收站
新建 文本文档.txt
A Simple GUI
File
3D-Surface
Surf
Mesh
Contour
Select Data
peaks
10
0
-10
40
20
0
0
20
40
开始
MATLAB
A Simple GUI
17:01

图形句柄的具体层次

Matlab中设计图形用户界面的方法有两种：

- 使用可视化的界面环境
- 通过编写程序。

➢ 图形用户界面设计工具的启动

图形用户界面设计工具的启动方式：

1. 命令方式

图形用户界面GUI设计工具的启动命令为guide，格式为:

① guide

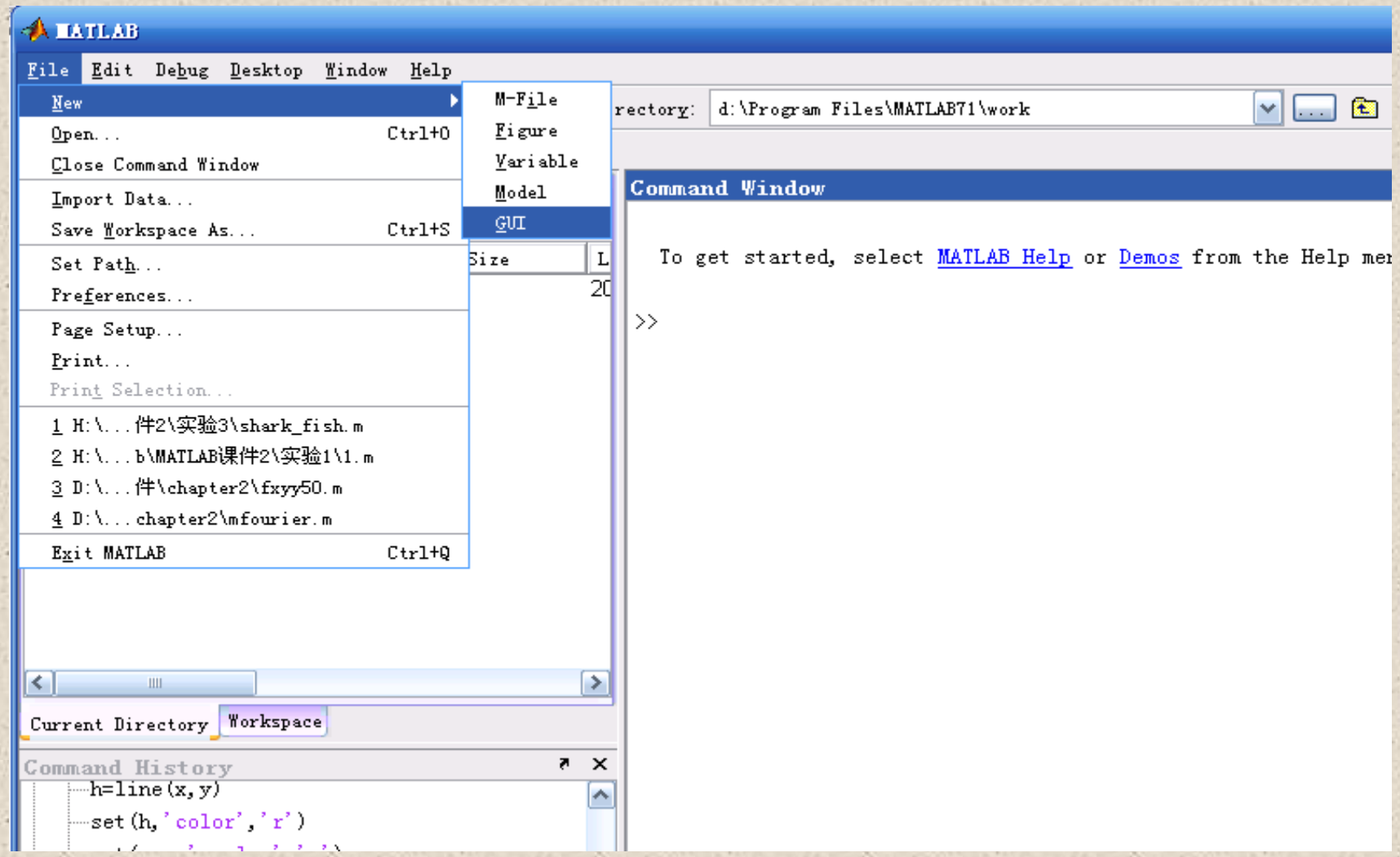
功能：启动GUI设计工具，并建立名字为untitled.fig的图形用户界面。

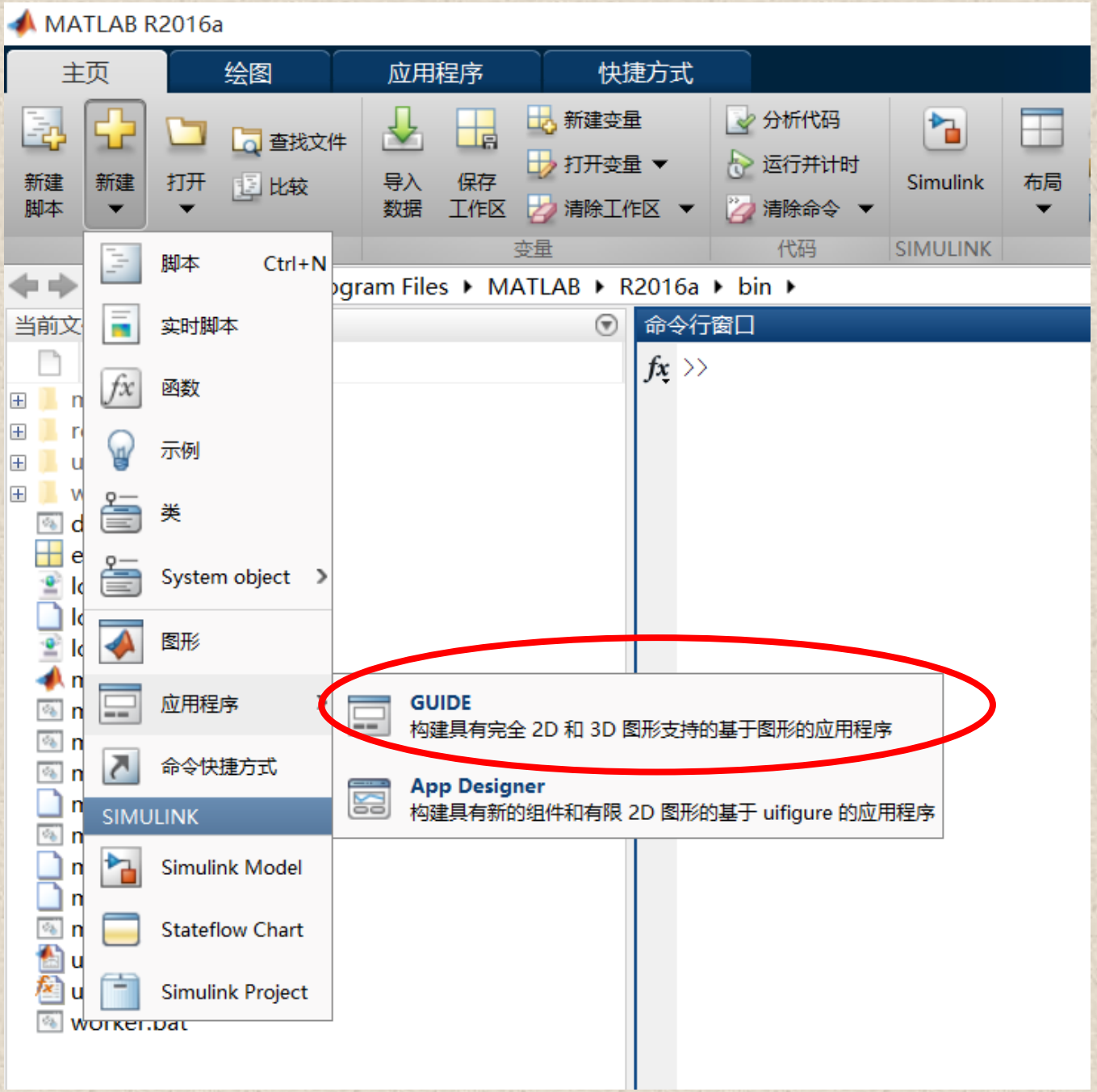
② guide filename

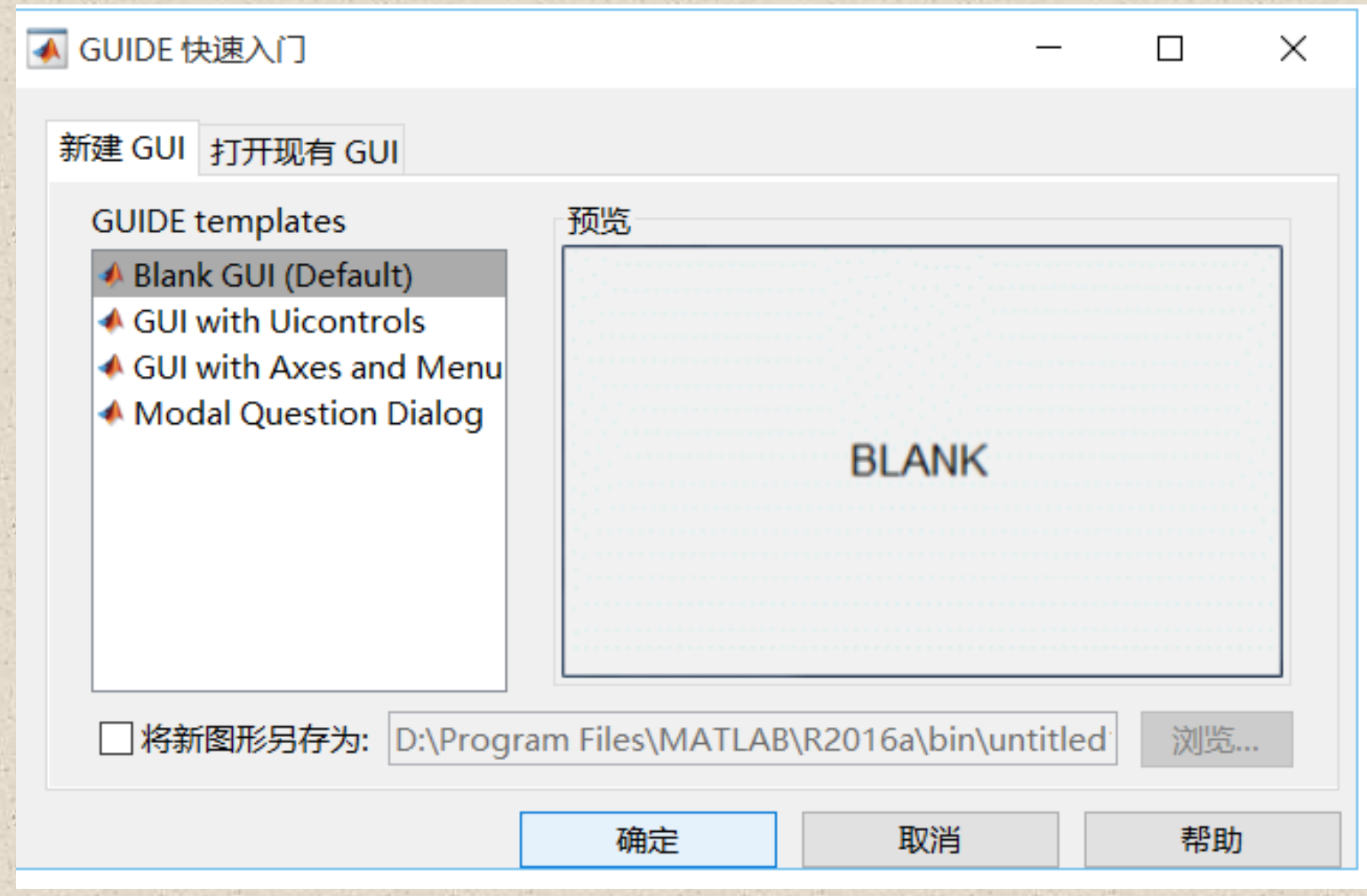
功能：启动GUI设计工具，并打开已建立的图形用户界面filename。

2. 菜单方式

在Matlab的主窗口中，选择File菜单中的New菜单项，再选择其中的GUI命令，就会显示GUI的设计模板。




MATLAB R2016a
主页
绘图
应用程序
快捷方式
新建脚本
新建
打开
查找文件
比较
导入数据
保存工作区
新建变量
打开变量
清除工作区
分析代码
运行并计时
清除命令
Simulink
布局
变量
代码
SIMULINK
gram Files ▸ MATLAB ▸ R2016a ▸ bin ▸
命令行窗口
>>
脚本 Ctrl+N
实时脚本
函数
示例
类
System object
图形
应用程序
命令快捷方式
SIMULINK
Simulink Model
Stateflow Chart
Simulink Project
GUIDE
构建具有完全 2D 和 3D 图形支持的基于图形的应用程序
App Designer
构建具有新的组件和有限 2D 图形的基于 uifigure 的应用程序



图形用户界面设计工具启动时模板选择对话框

Matlab为GUI设计一共准备了4种模板，分别是:

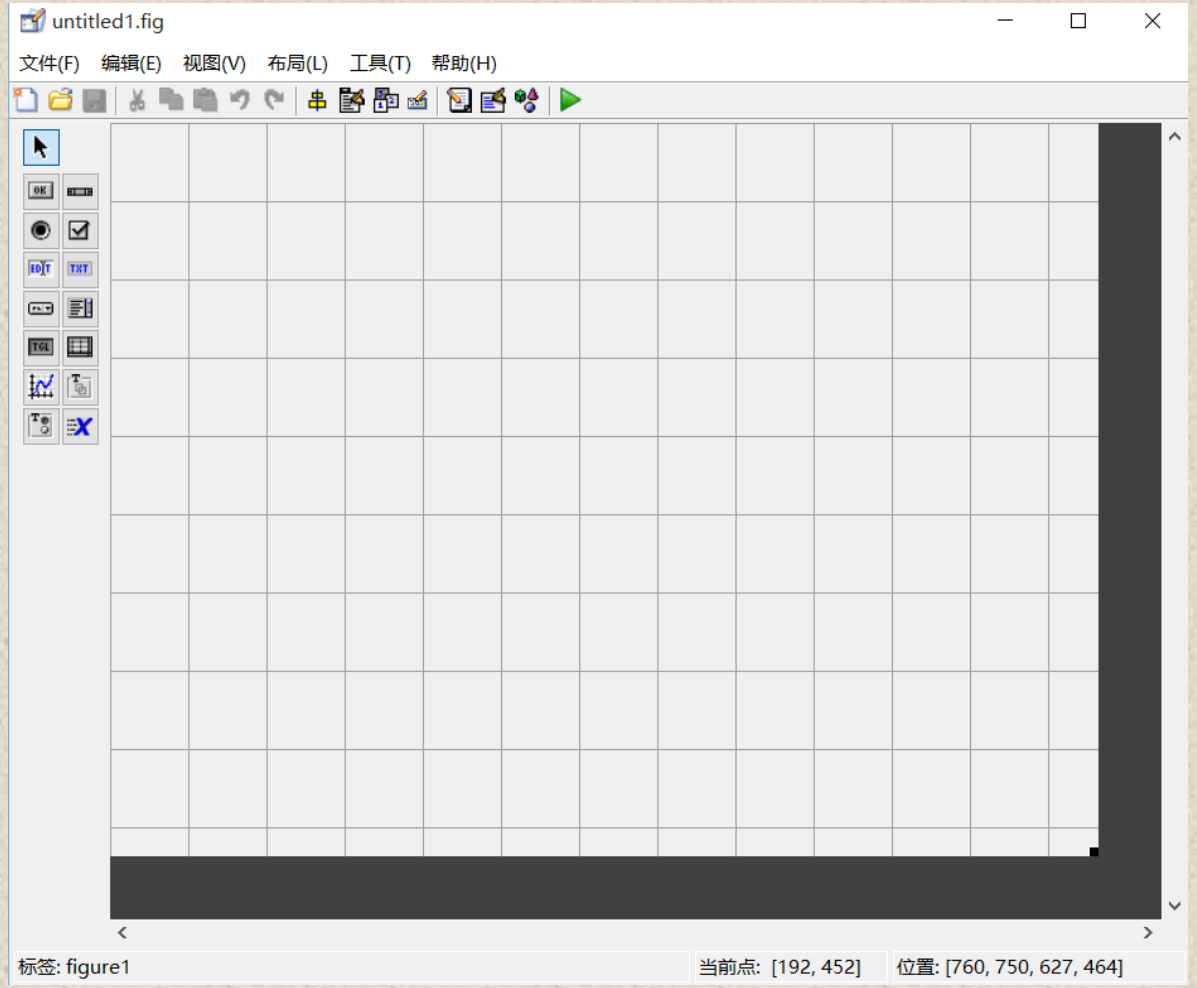
- **Blank GUI（Default）(空白模板，默认);**
- **GUI with Uicontrols(带控件对象的GUI模板);**
- **GUI with Axes and Menu(带坐标轴与菜单的GUI模板);**
- **Modal Question Dialog(带模式问题对话框的GUI模板)。**

当用户选择不同的模板时，在GUI设计模板界面的右边就会显示出与该模板对应的GUI图形。

图形用户界面设计窗口

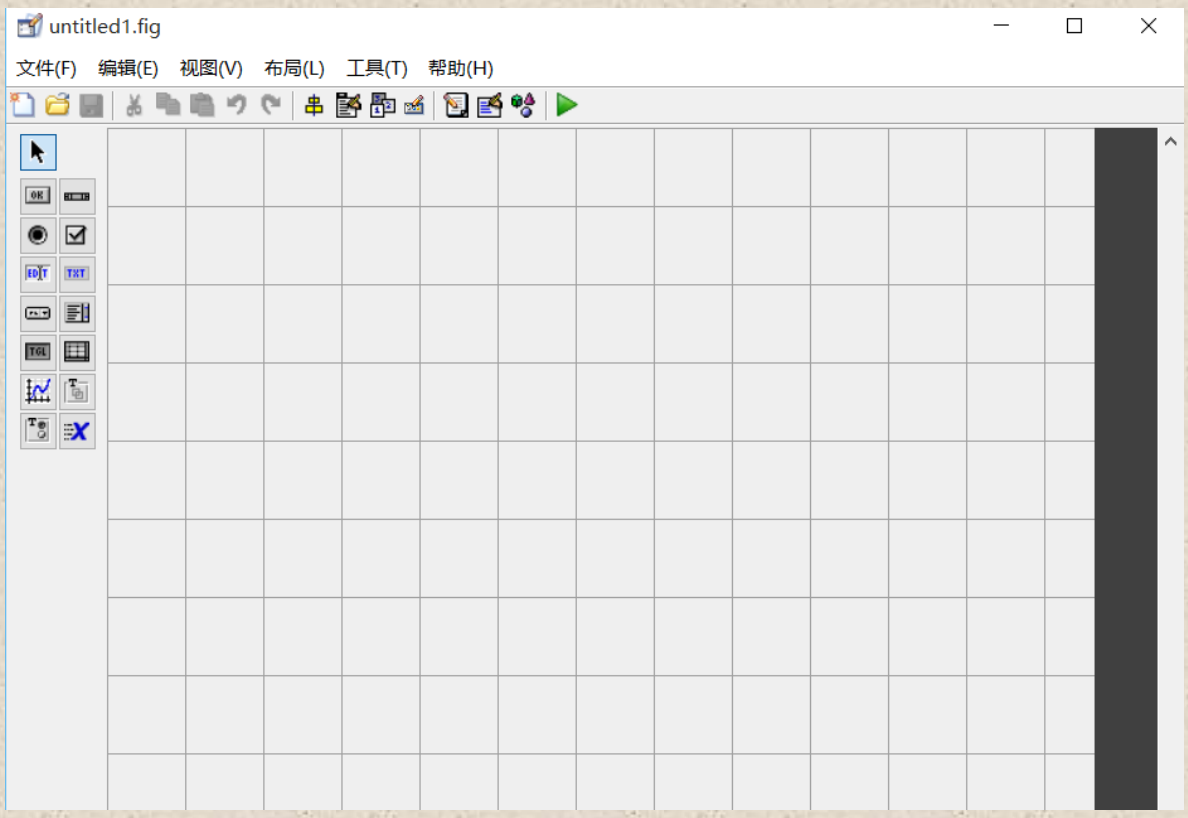
在GUI设计模板中选中一个模板，然后单击OK按钮，就会显示GUI设计窗口。选择不同的GUI设计模式时，在GUI设计窗口中显示的结果是不一样的。

图形用户界面GUI设计窗口由菜单栏、工具栏、控件工具栏以及图形对象设计区等4个功能区组成。



右图为空白GUI模板情形

GUI设计窗口的菜单栏有File、Edit、View、Layout、Tools和Help 6个菜单项，使用其中的命令可以完成图形用户界面的设计操作。



编辑工具在菜单栏的下方，提供了常用的工具；
设计工具区位于窗口的左半部分，提供了设计GUI过程中所用的用户控件；
空间模板区是网格形式的用户设计GUI的空白区域。

在GUI设计窗口创建图形对象后，通过双击该对象，就会显示该对象的属性编辑器。

一、图形用户界面开发环境(GUIDE)

Matlab提供了一套可视化的创建图形窗口的工具，使用图形用户界面开发环境可方便地创建GUI应用程序，它可以根据用户设计的GUI布局，自动生成M文件的框架，用户使用这一框架编制自己的应用程序。

Matlab提供了一套可视化的创建图形用户接口（GUI）的工具，包括：

◆布局编辑器(Layout Editor)——在图形窗口中创建及布置图形对象。布局编辑器是可以启动用户界面的控制面板，上述工具都必须从布局编辑器中访问，用guide命令可以启动，或在启动平台窗口中选择GUIDE来启动布局编辑器；

◆几何排列工具(Alignment Tool)——调整各对象相互之间的几何关系和位置；

◆属性查看器(Property Inspector)——查询并设置属性值；

◆对象浏览器(Object Browser)——用于获得当前Matlab图形用户界面程序中的全部对象信息，对象的类型，同时显示控件的名称和标识，在控件上双击鼠标可以打开该控件的属性编辑器；

◆菜单编辑器(Menu Editor)——创建、设计、修改下拉式菜单和快捷菜单；

◆Tab顺序编辑器（Tab Order Editor）——用于设置当用户按下键盘上的Tab键时，对象被选中的先后顺序。

在Matlab 5中， GUI的设计是以 M文件的编程形式实现的，GUI的布局代码存储在M文件和MAT文件中，

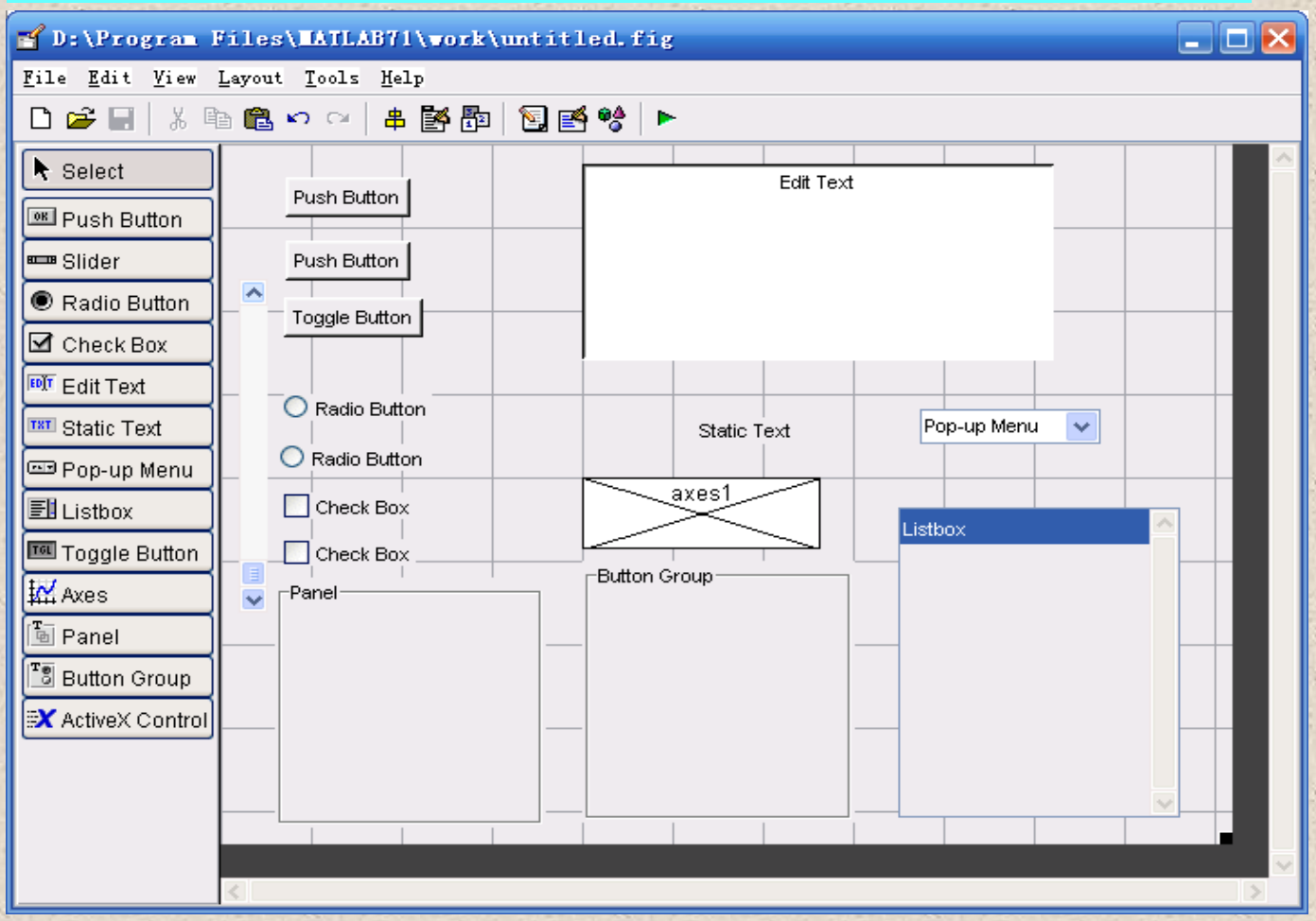
在Matlab 6中有了很大的改变，Matlab 6将GUI的布局代码存储在 FIG文件中，同时还产生一个M文件用于存储调用函数，在M文件中不再包含GUI的布局代码，在开发应用程序时的代码量大大减少。

1. 布局编辑器(Layout editor)

用于从控件选择板上选择控件对象并放置到布局区去，布局区被激活后就成为图形窗口。 在命令窗口输入GUIDE命令或点击工具栏中的guide图标都可以打开空白的布局编辑器，在命令窗口输入 GUIDE filename 可打开一个已存在的名为filename 图形用户界面。

① 将控件对象放置到布局区

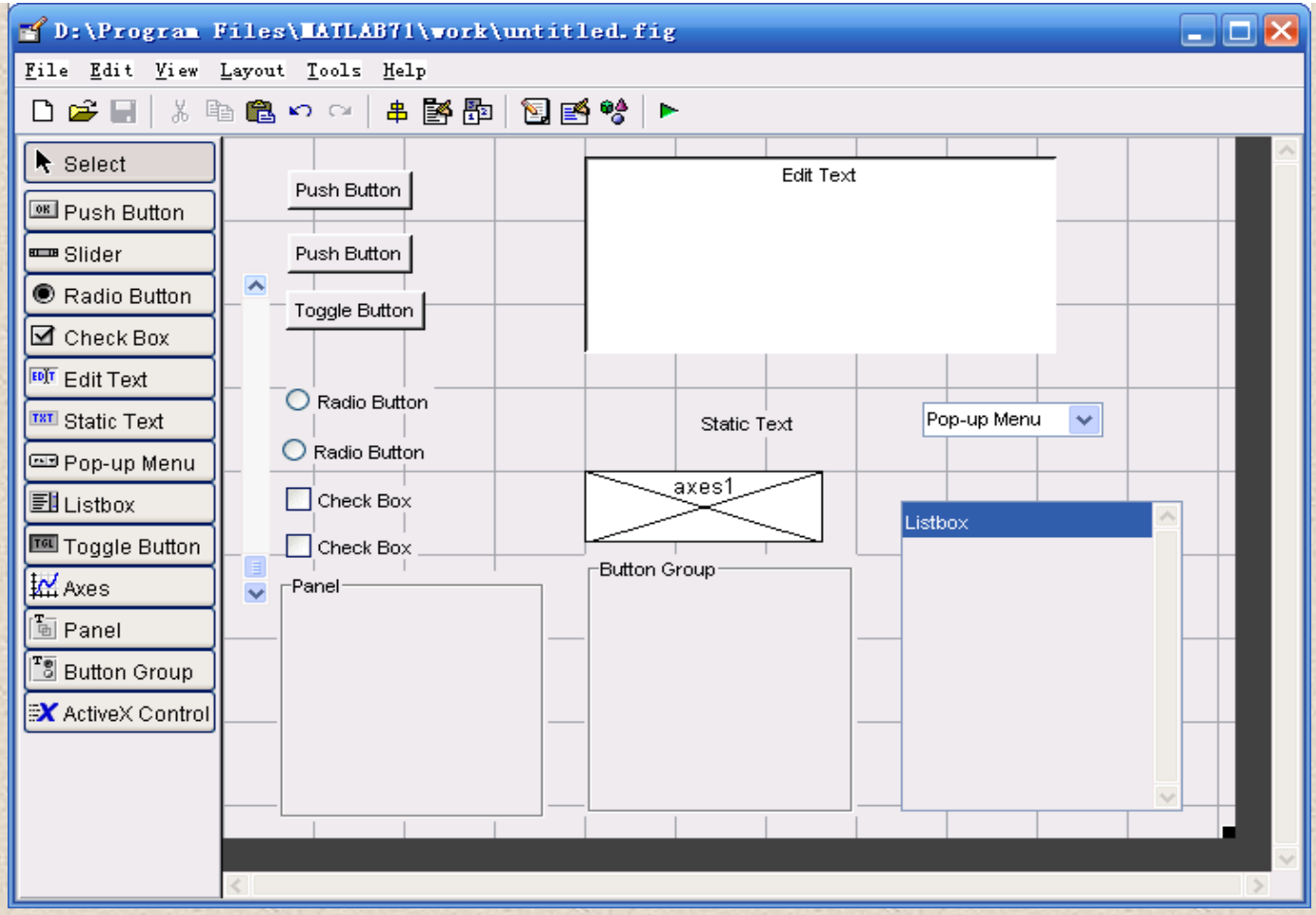
- 用鼠标选择并放置控件到布局区内；
- 移动控件到适当的位置；
- 改变控件的大小；
- 可同时选中多个对象。



一个简单的布局示例

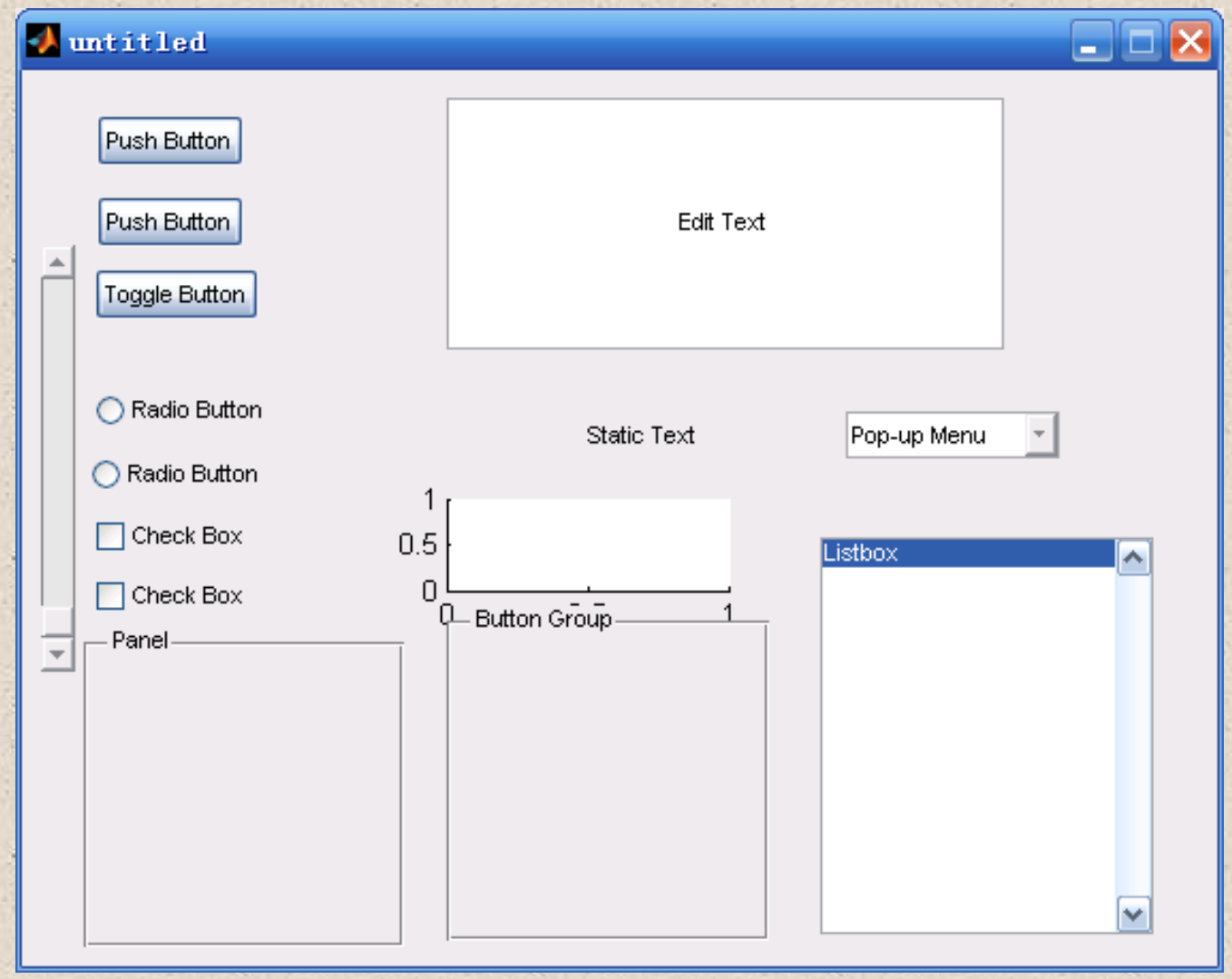
② 激活图形窗口

如所建立的布局还没有进行存储，可用File菜单下的Save As菜单项（或工具栏中的对应项），按输入的文件的名字，在激活图形窗口的同时将存储一对同名的M文件和带有.fig扩展名的FIG文件。



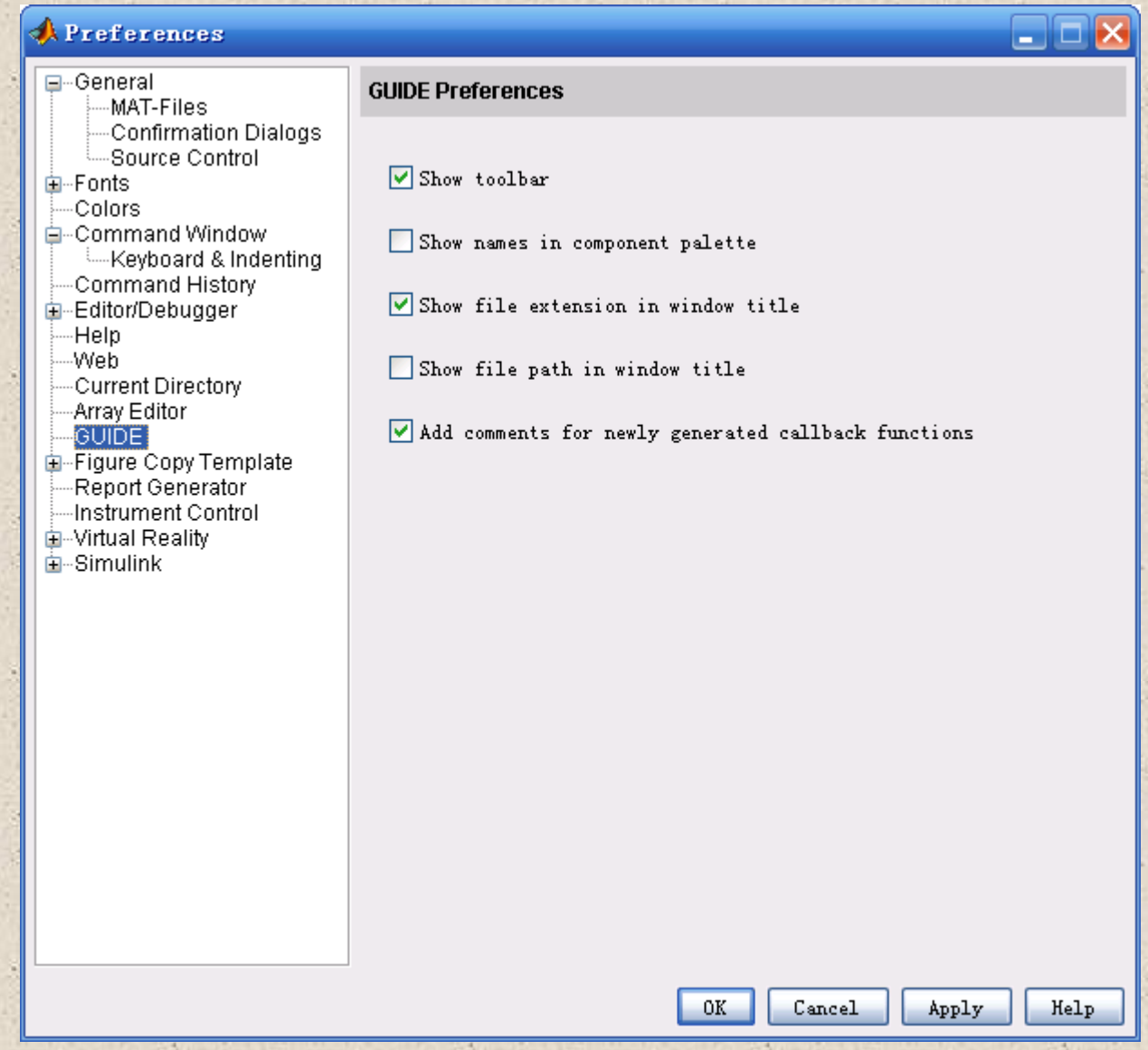
③ 运行GUI程序

在命令窗口直接键入文件名或用openfig, open或hgload命令运行GUI程序。



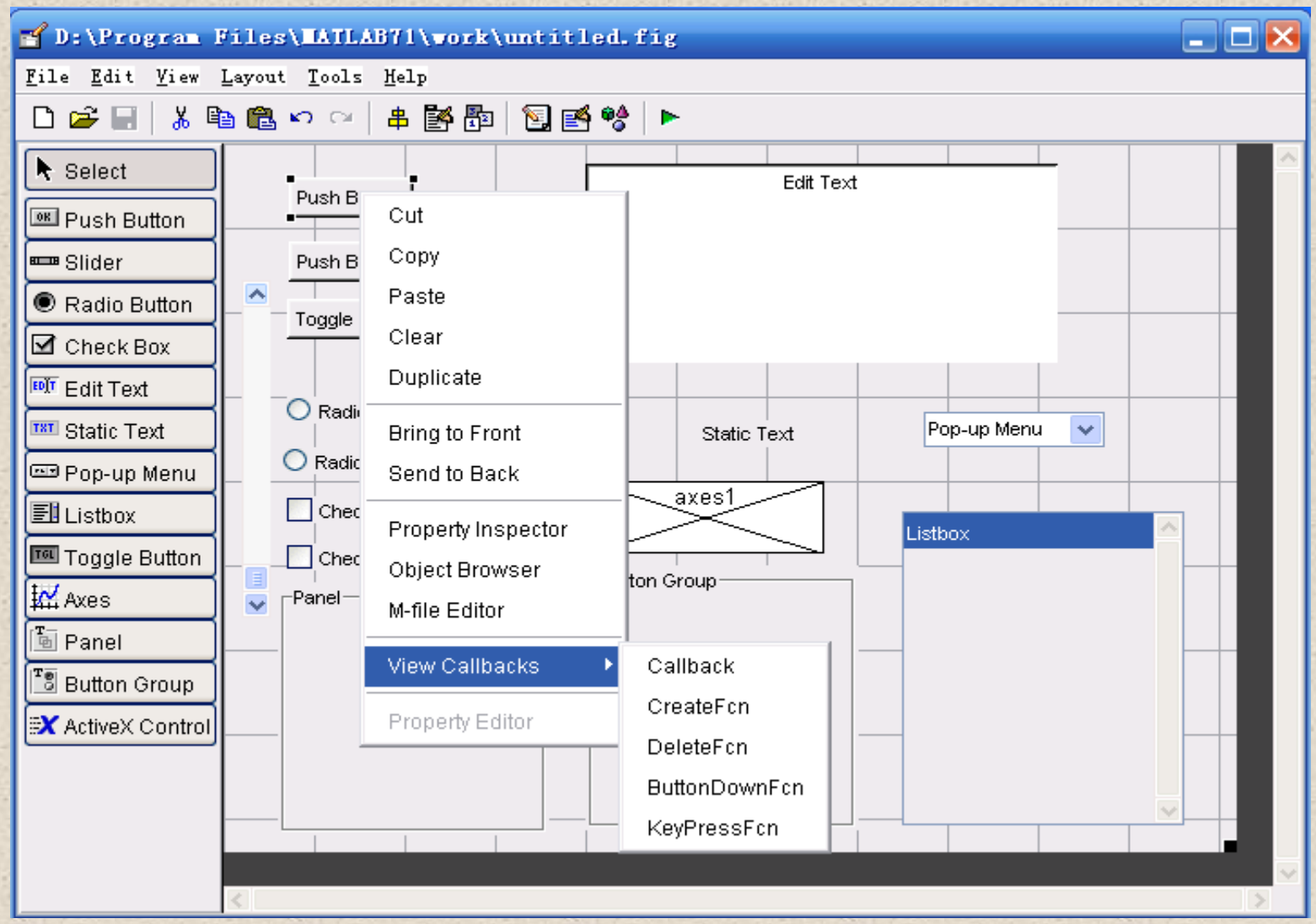
④ 布局编辑器参数设置

选File菜单下的Preferences菜单项打开参数设置窗口，点击树状目录中的GUIDE，即可以设置布局编辑器的参数。



⑤ 布局编辑器的弹出菜单

在任一控件上按下鼠标右键，会弹出一个菜单，通过该菜单可以完成布局编辑器的大部分操作。



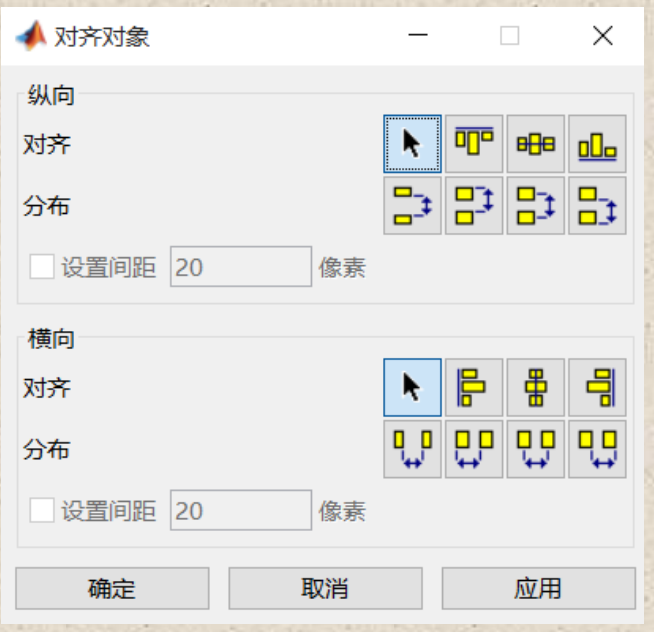
2. 位置调整工具(Alignment tool)

利用位置调整工具，可以对GUI对象设计区内的多个对象的位置进行调整。

位置调整工具的打开方式有两种：

① 从GUI设计窗口的工具栏上选择Align Objects命令按钮;

② 选择Tools菜单下的Align Objects...菜单项，就可以打开对象位置调整器。



对象位置调整器中的第一栏是垂直方向的位置调整，第二栏是水平方向的位置调整。

在选中多个对象后，可以方便的通过对象位置调整器调整对象间的对齐方式和距离。

3. 用属性查看器设置控件属性

利用对象属性查看器，可以查看每个对象的属性值，也可以修改、设置对象的属性值。

① 打开属性查看器(Opening Property Inspector)

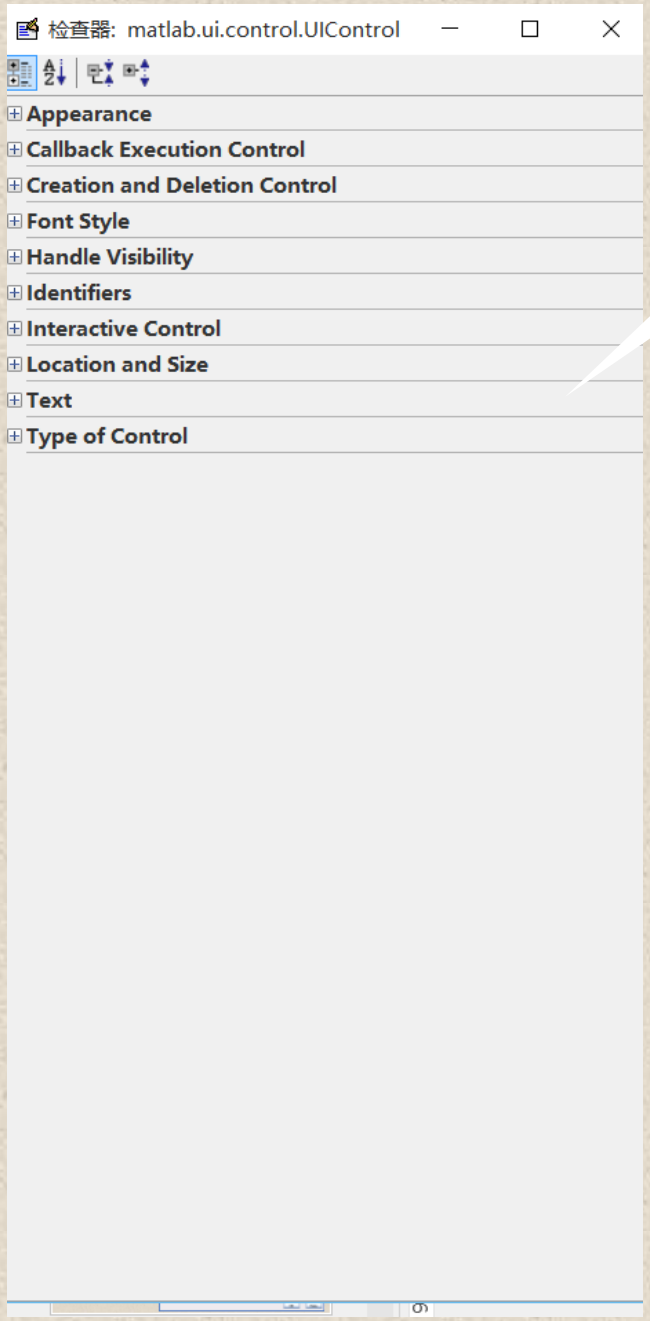
对象属性查看器的打开方式有四种：

◆从GUI设计窗口工具栏上选择Property Inspector命令按钮;

◆选择View菜单下的Property Inspector菜单项;

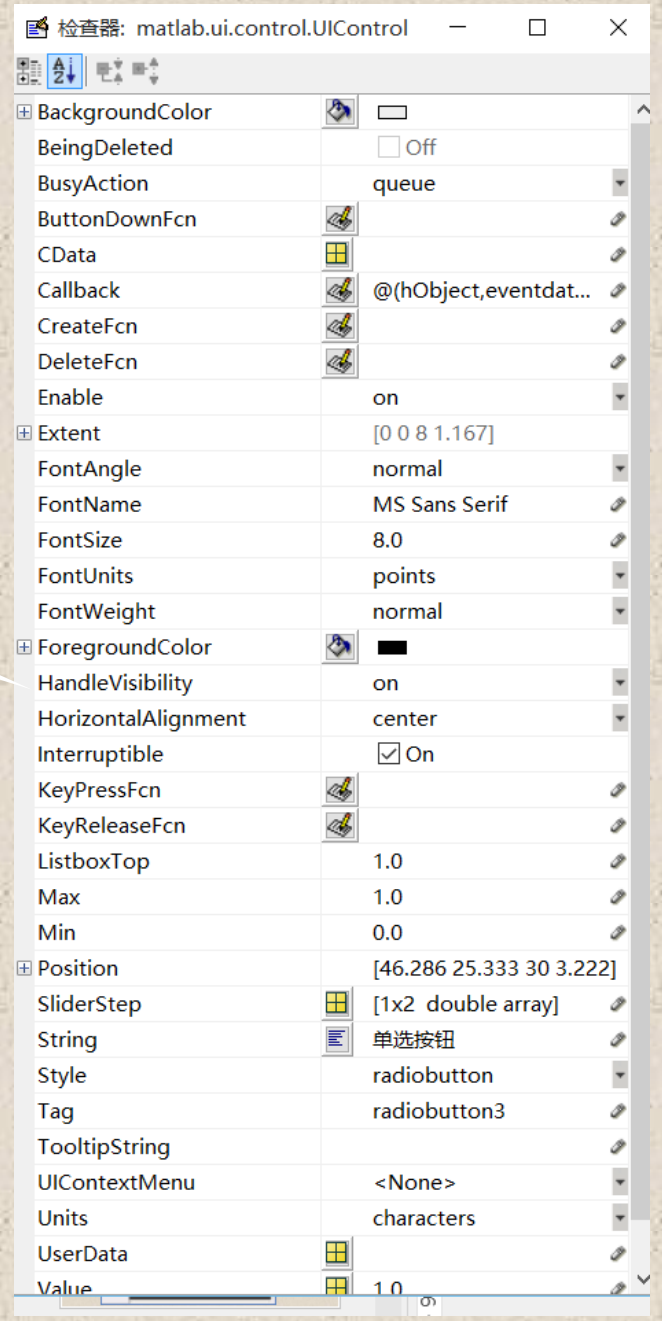
◆在命令窗口中输入inspect;

◆在控件对象上单击鼠标右键，选择弹出菜单的 Property Inspector菜单项。



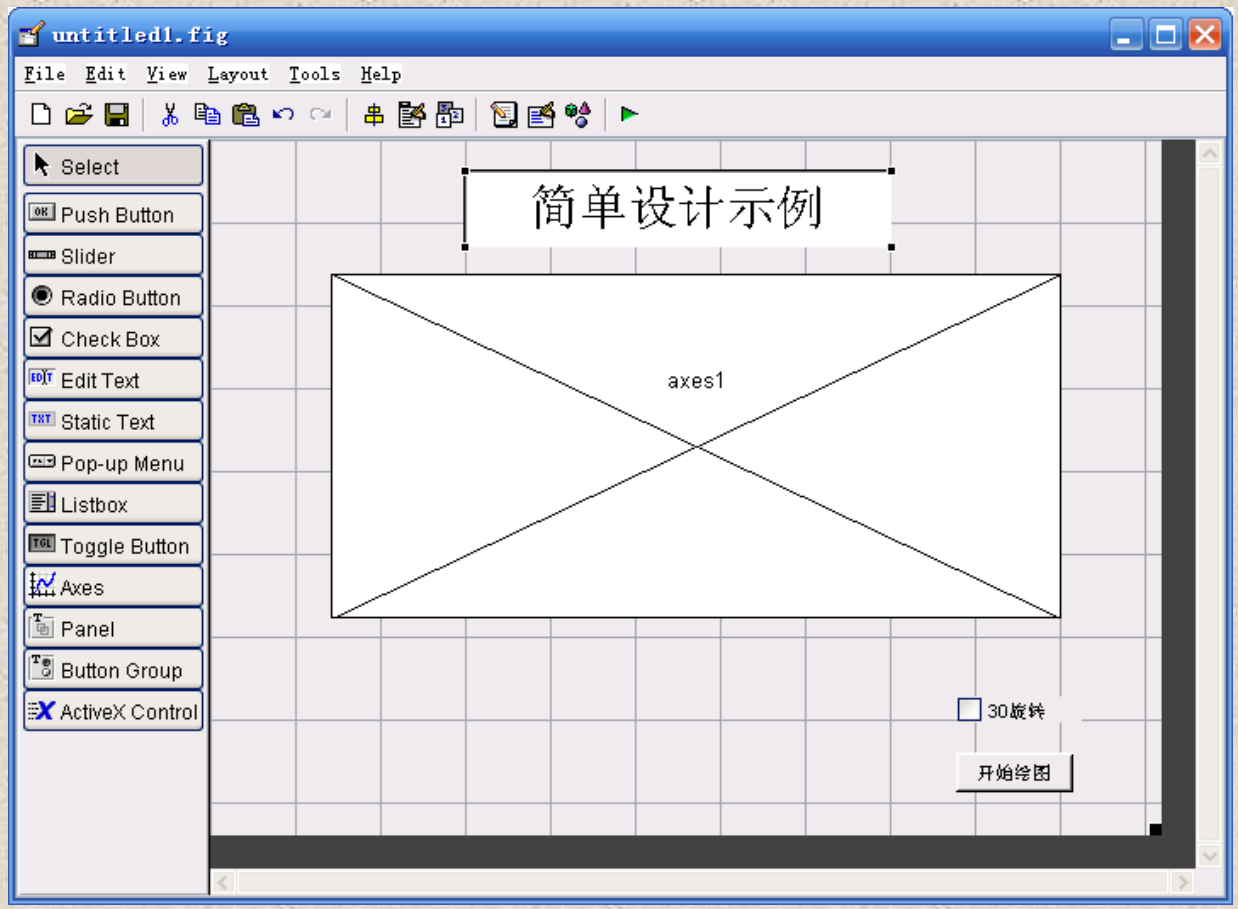
按照属性分类查看

按照字母顺序查看



② 使用属性查看器(Using Property Inspector)

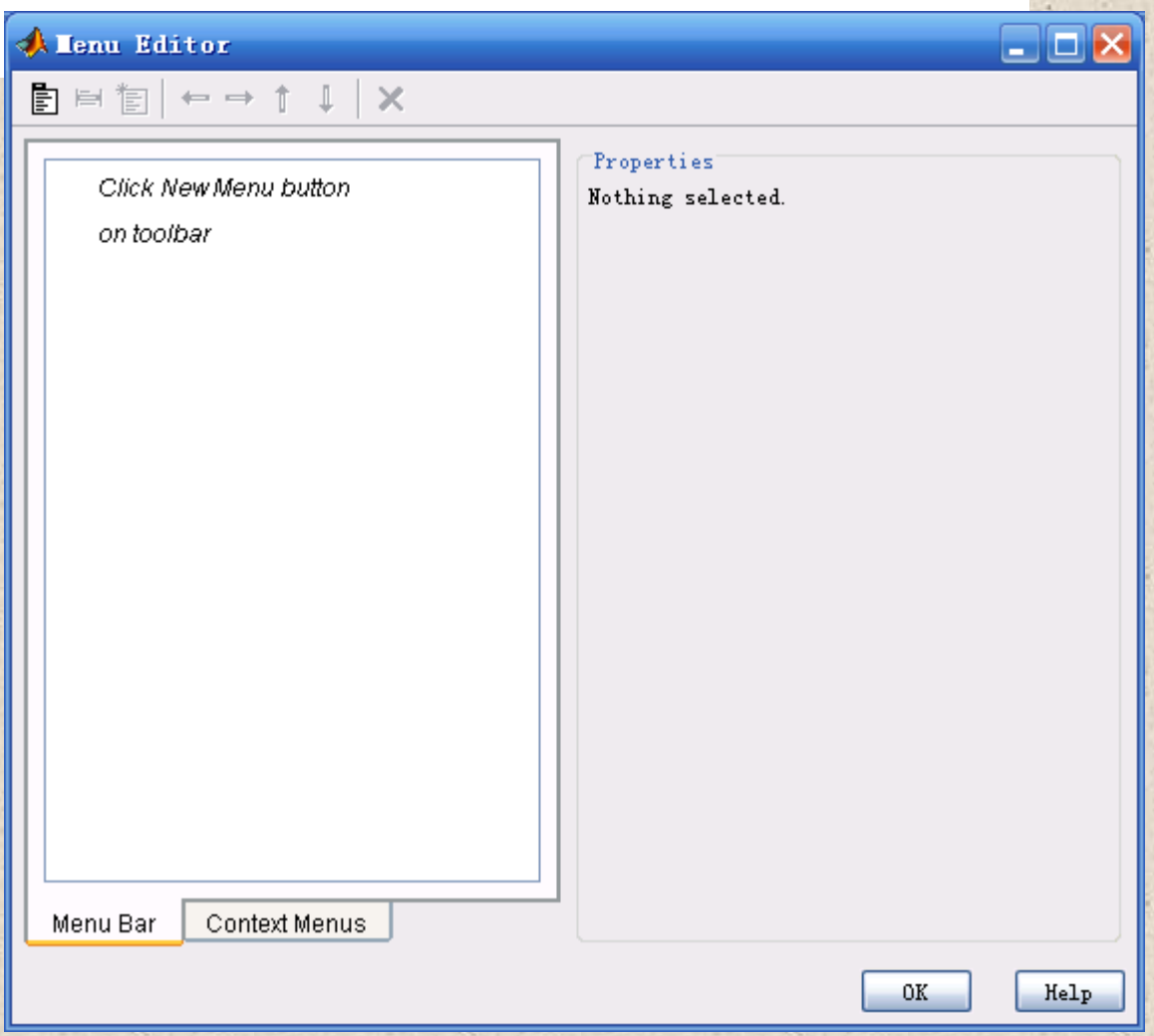
- 布置控件；
- 定义文本框的属性；
- 定义坐标轴的属性；
- 定义按钮的属性；
- 定义复选框。

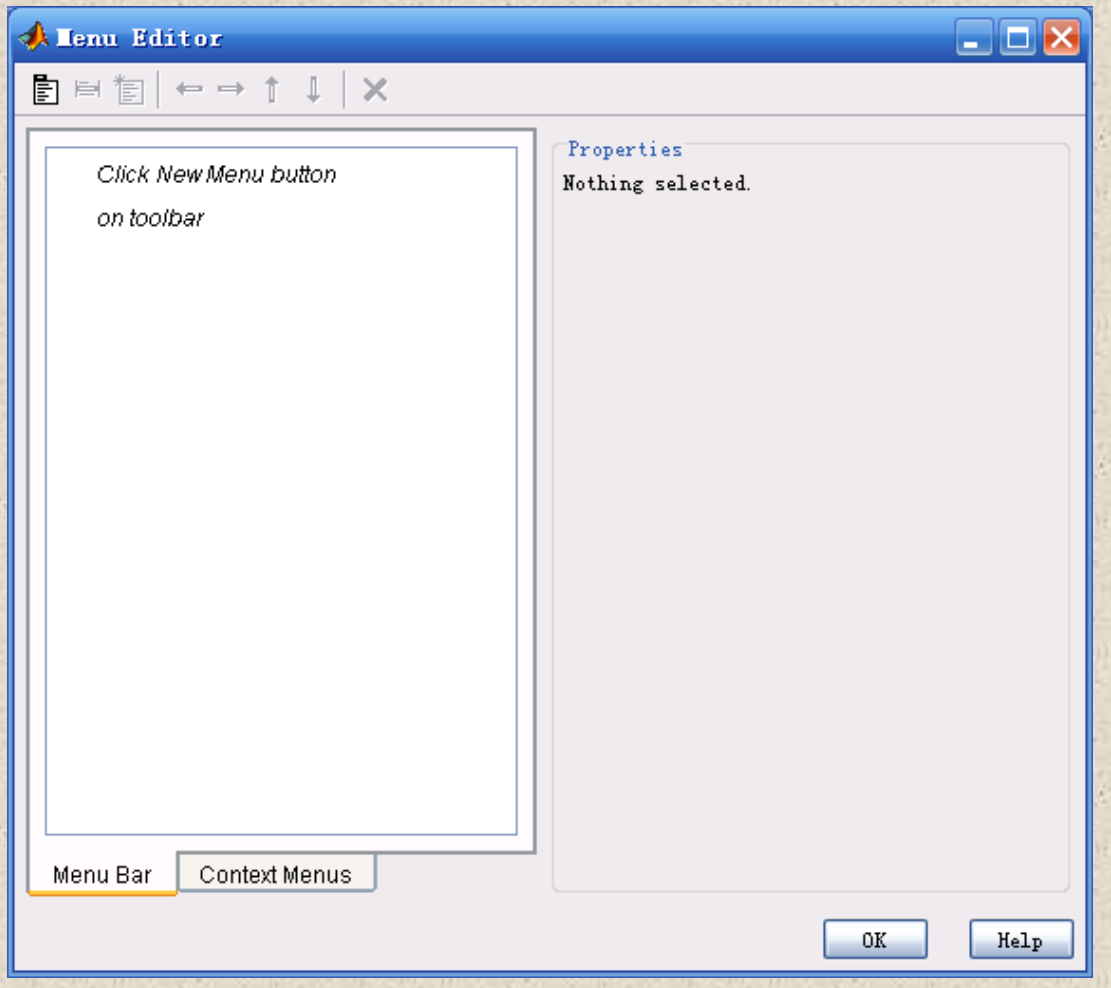


4. 菜单编辑器(Menu Editor)

利用菜单编辑器，可以创建、设置、修改下拉式菜单和快捷菜单。选择 Tools 菜单下的 Menu Editor...子菜单，即可打开菜单编辑器。

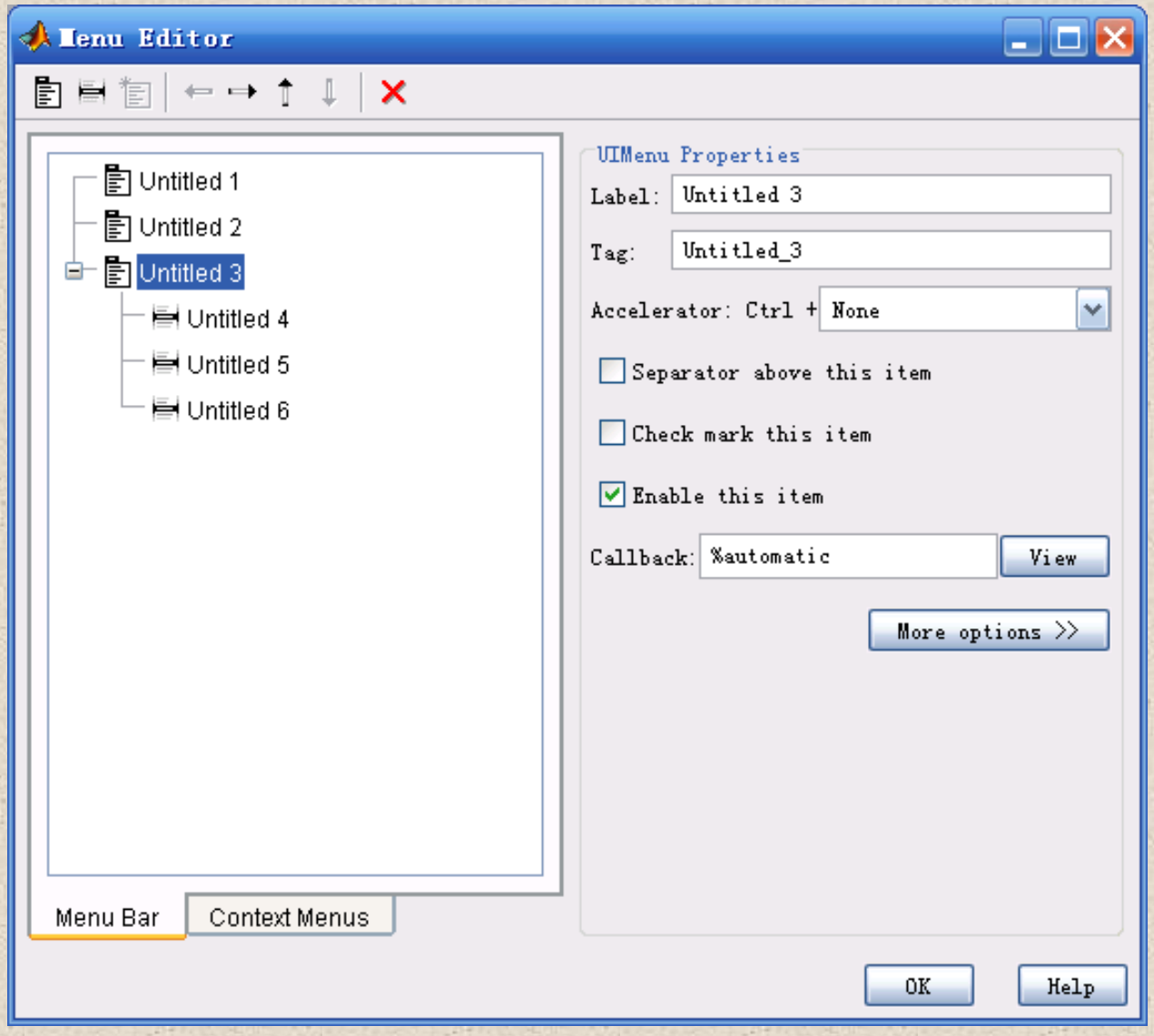
菜单也可以通过编程实现，方法为从GUI设计窗口的工具栏上选择Menu Editor命令按钮，打开菜单编辑程序。



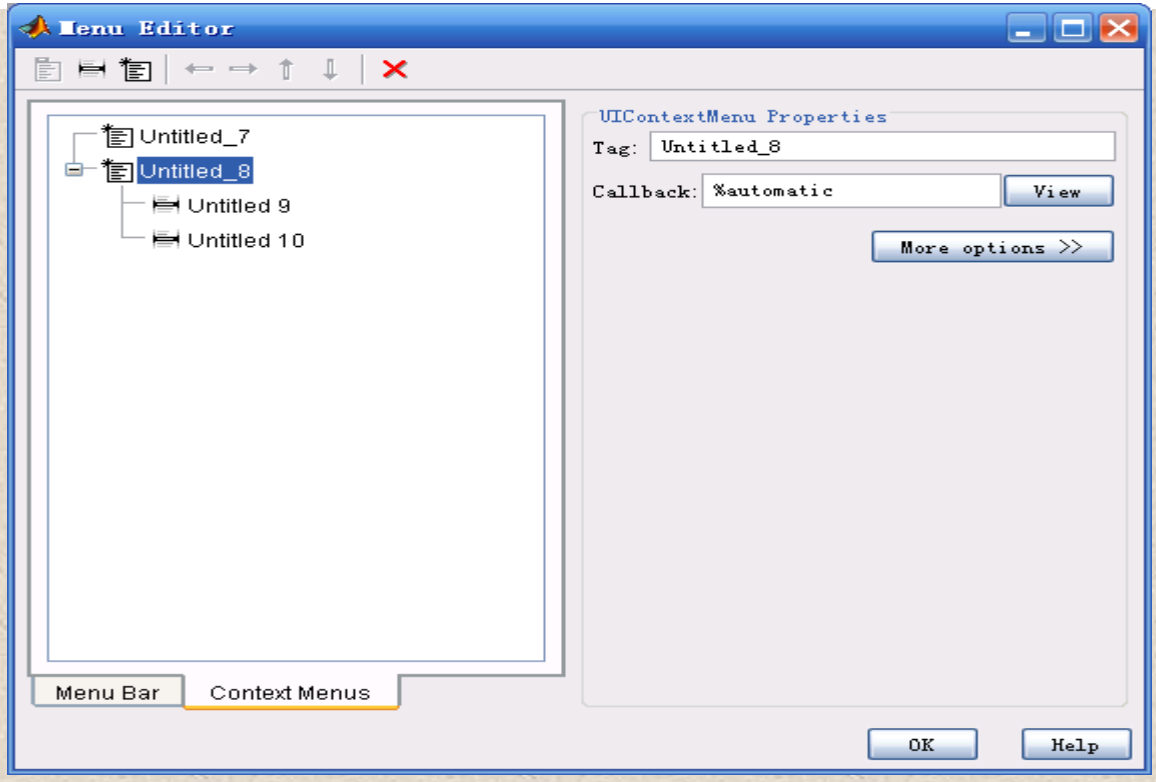


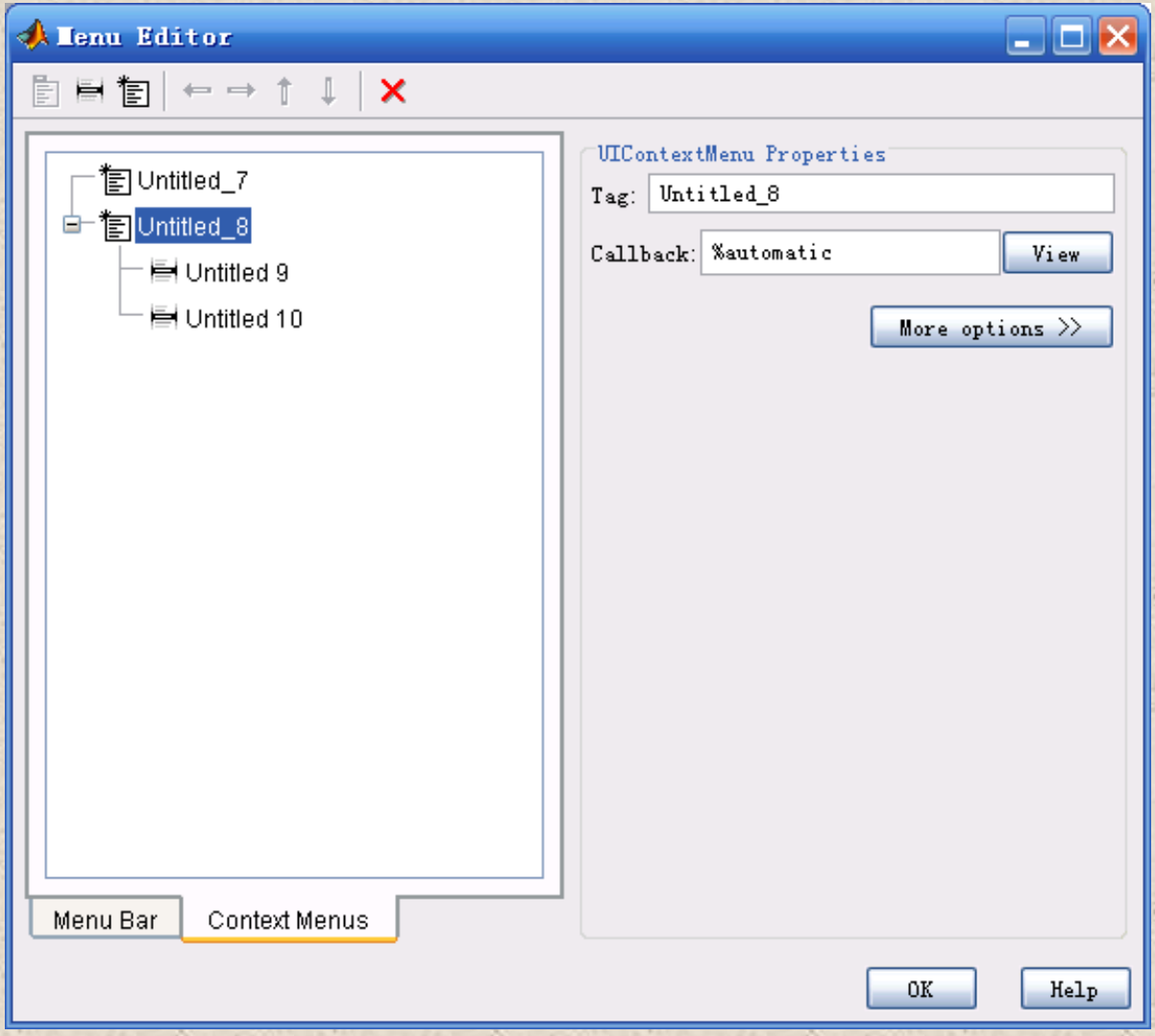
菜单编辑器包括菜单的设计和编辑，菜单编辑器有八个快捷键，可以利用它们任意添加或删除菜单，可以设置菜单项的属性，包括名称(Label)、标识(Tag)、选择是否显示分隔线(Separator above this item)、是否在菜单前加上选中标记(Item is checked)、调用函数(Callback)。

菜单编辑器左上角的第一个按钮用于创建一级菜单项。第二个按钮用于创建一级菜单的子菜单。



菜单编辑器的左下角有两个按钮，选择第一个按钮，可以创建下拉式菜单。选择第二个按钮，可以创建Context Menu菜单。选择它后，菜单编辑器左上角的第三个按钮就会变成可用，单击它就可以创建Context Menu主菜单。在选中已经创建的Context Menu主菜单后，可以单击第二个按钮创建选中的Context Menu主菜单的子菜单。与下拉式菜单一样，选中创建的某个Context Menu菜单，菜单编辑器的右边就会显示该菜单的有关属性，可以在这里设置、修改菜单的属性。





菜单编辑器左上角的第四个与第五个按钮用于对选中的菜单进行左移与右移，第六与第七个按钮用于对选中的菜单进行上移与下移，最右边的按钮用于删除选中的菜单。

5. 对象浏览器(Object Browsers)

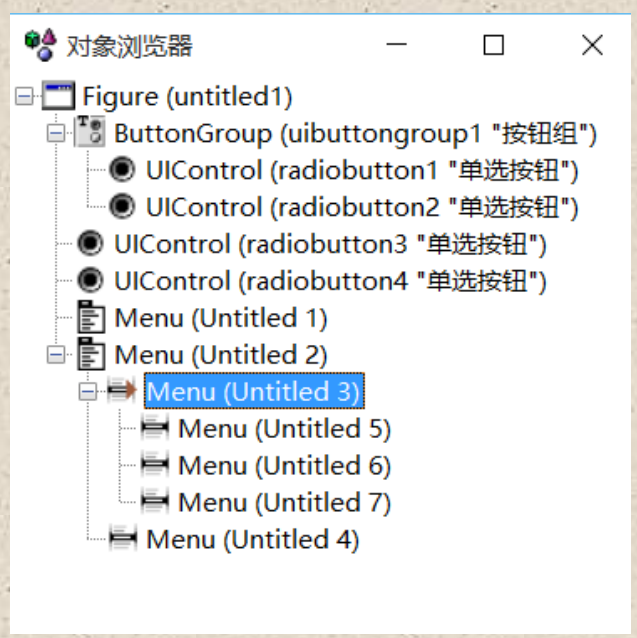
利用对象浏览器，可以查看当前设计阶段的各个句柄图形对象。可以在对象浏览器中选中一个或多个控件来打开该控件的属性编辑器。

对象浏览器的打开方式有：

① 从GUI设计窗口的工具栏上选择Object Browser命令按钮.

② 选择View菜单下的Object Browser子菜单；

③ 在设计区域单击鼠标右键，选择弹出菜单的 Object Browser。



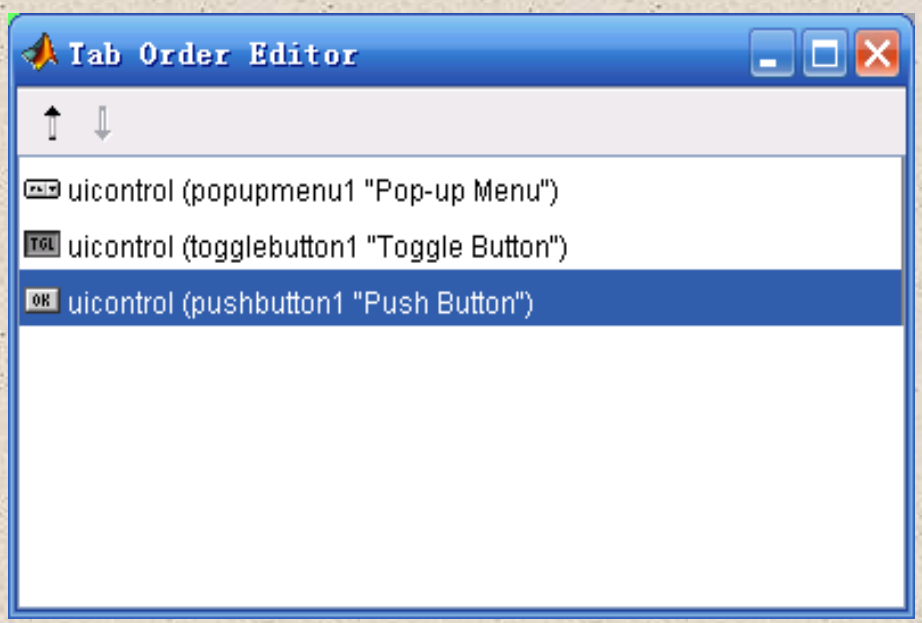
6． Tab顺序编辑器（Tab Order Editor）

利用Tab顺序编辑器(Tab Order Editor)，可以设置用户按键盘上的Tab键时，对象被选中的先后顺序。

Tab顺序编辑器的打开方式为：

① 选择Tools菜单下的Tab Order Editor...菜单项，就可以打开Tab顺序编辑器。

② 从GUI设计窗口的工具栏上选择Tab Order Editor...命令按钮。



二、控件对象及属性

1. GUI控件对象类型

控件对象是事件响应的图形界面对象。当某一事件发生时，应用程序会做出响应并执行某些预定的功能子程序（Callback）.

2. 控件对象的描述

Matlab中的控件大致可分为两种，一种为动作控件，鼠标点击这些控件时会产生相应的响应。一种为静态控件，是一种不产生响应的控件，如文本框等。

每种控件都有一些可以设置的参数，用于表现控件的外形、功能及效果，即属性。属性由两部分组成：属性名和属性值，它们必须是成对出现的。

① **按钮(Push Buttons)：** 执行某种预定的功能或操作；

② **切换按钮(Toggle Button)：** 产生一个动作并指示一个二进制状态（开或关），当鼠点击它时按钮将下陷，并执行callback（回调函数）中指定的内容，再次点击，按钮复原，并再次执行callback 中的内容；

③ **单选按钮(Radio Button)：** 单个的单选框用来在两种状态之间切换，多个单选框组成一个单选框组时，用户只能在一组状态中选择单一的状态，或称为单选项；

④ **复选框(Check Boxes)：** 单个的复选框用来在两种状态之间切换，多个复选框组成一个复选框组时，可使用户在一组状态中作组合式的选择，或称为多选项；

⑤ **文本编辑器(Editable Texts)：** 用来使用键盘输入字符串的值，可以对编辑框中的内容进行编辑、删除和替换等操作；

⑥ **静态文本框(Static Texts):** 仅用于显示单行的说明文字;

⑦ **滚动条(Slider)：** 可输入指定范围的数量值;

⑧ **面板(Frames)：** 在图形窗口圈出一块区域;

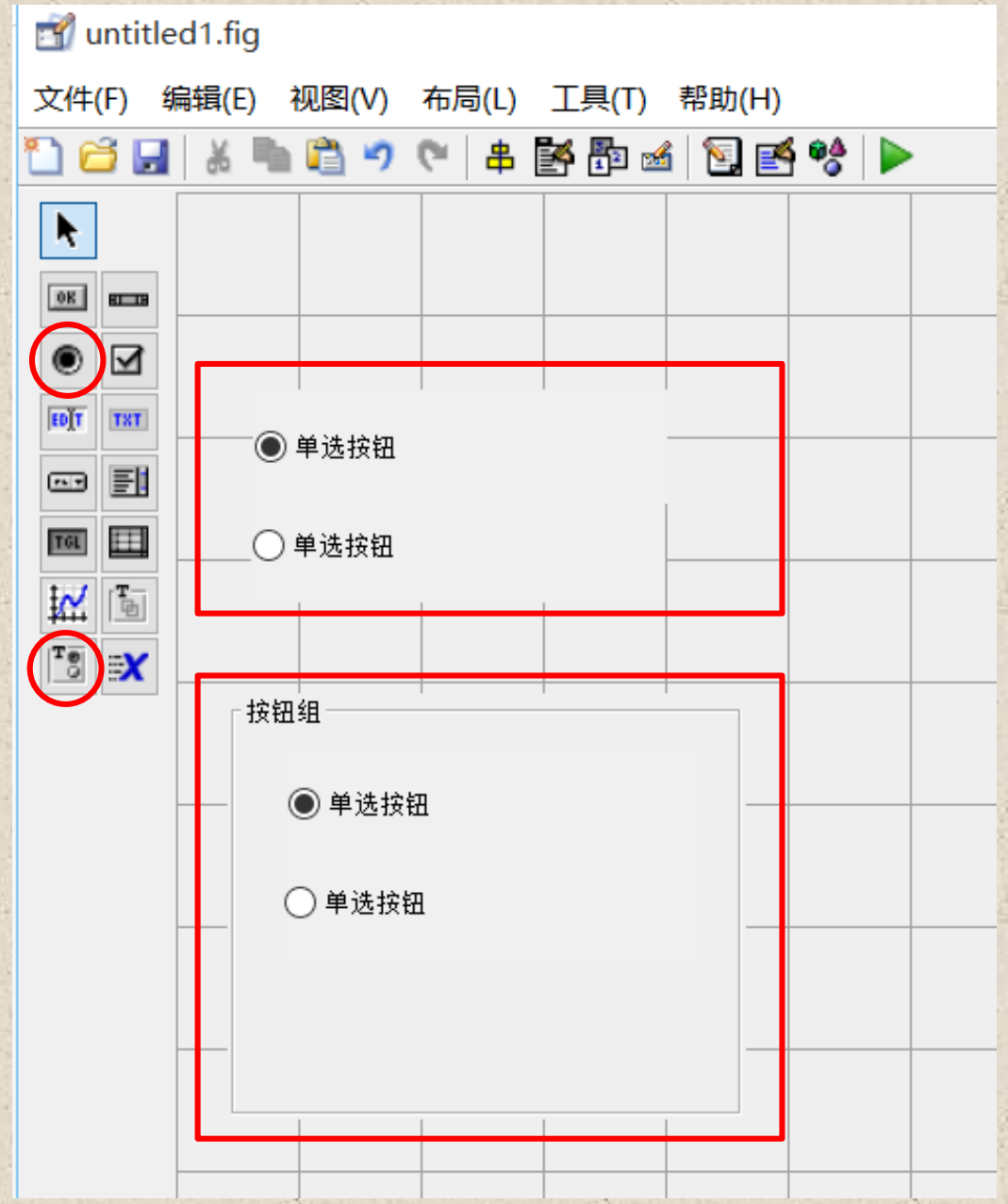
⑨ **列表框(List Boxes)：** 在其中定义一系列可供选择的字符串;

⑩ **弹出式菜单(Popup Menus)：** 让用户从一列菜单项中选择一项作为参数输入;

⑪ **坐标轴(Axes)：** 用于显示图形和图象。

⑫ **按钮组(Axes)：** 与单选框按钮配合实现多个选项的单选功能。

将单选框放入按钮组中，运行后即可实现单选按钮之间的互斥（只能选中其中一个）



3. 控件对象的属性(Attributes of controller object)

用户可以在创建控件对象时，设定其属性值，未指定时将使用系统缺省值。

两大类控件对象属性：

- 第一类是所有控件对象都具有的公共属性
- 第二类是控件对象作为图形对象所具有的属性。

① 控件对象的公共属性

◆**Children** 取值为空矩阵，因为控件对象没有自己的子对象；

◆**Parent** 取值为某个图形窗口对象的句柄，该句柄表明了控件对象所在的图形窗口；

◆**Tag** 取值为字符串，定义了控件的标识值，在任何程序中都可以通过这个标识值控制该控件对象；

◆**Type** 取值为uicontrol，表明图形对象的类型；

◆**UserDate** 取值为空矩阵，用于保存与该控件对象相关的重要数据和信息；

◆**Visible** 取值为on 或off。

② 控件对象的基本控制属性

◆**BackgroundColor** 取值为颜色的预定义字符或RGB数值；缺省值为浅灰色；

◆**Callback** 取值为字符串，可以是某个M文件名或一小段Matlab语句，当用户激活某个控件对象时，应用程序就运行该属性定义的子程序；

◆**Enable** 取值为on（缺省值）, inactive和off；

◆**Extend** 取值为四元素矢量[0, 0, width, height]，记录控件对象标题字符的位置和尺寸；

◆**ForegroundColor** 取值为颜色的预定义字符或RGB数值，该属性定义控件对象标题字符的颜色；缺省值为黑色；

② 控件对象的基本控制属性

◆**Max，Min** 取值都为数值，缺省值分别为1和0；

◆**String** 取值为字符串矩阵或块数组，定义控件对象标题或选项内容；

◆**Style** 取值可以是pushbutton(缺省值), radiobutton, checkbox, edit, text, slider, frame, popupmenu 或listbox；

◆**Units** 取值可以是pixels (缺省值), normalized（相对单位）, inches, centimeters（厘米）或points（磅）；

◆**Value** 取值可以是矢量，也可以是数值，其含义及解释依赖于控件对象的类型。

③ 控件对象的修饰控制属性

◆**FontAngle** 取值为normal（正体，缺省值）, italic（斜体）, oblique（方头）；

◆**FontName** 取值为控件标题等字体的字库名；

◆**FontSize** 取值为数值；

◆**FontUnits** 取值为points（缺省值）, normalized, inches, centimeters或pixels；

◆**FontWeight** 取值为normal（缺省值）, light，demi和bold，定义字符的粗细；

◆**HorizontalAligment** 取值为left，center（缺省值）或right，定义控件对象标题等的对齐方式。

④ 控件对象的辅助属性

◆**ListboxTop** 取值为数量值，用于listbox控件对象；

◆**SliderStep** 取值为两元素矢量[minstep, maxstep]，用于slider控件对象；

◆**Selected** 取值为on 或off（缺省值）；

◆**SlectionHighlight** 取值为on 或off（缺省值）。

⑤ Callback管理属性

◆**BusyAction** 取值为cancel或queue（缺省值）；

◆**ButtDownFun** 取值为字符串，一般为某个M文件名或一小段Matlab程序；

◆**Creatfun** 取值为字符串，一般为某个M文件名或一小段Matlab程序；

◆**DeletFun** 取值为字符串，一般为某个M文件名或一小段Matlab程序；

◆**HandleVisibility** 取值为on（缺省值）, callback或off；

◆**Interruptible** 取值为on 或off（缺省值）。

4. 控件对象的建立

在对话框上有各种各样的控件，利用这些控件可以实现有关控制。

Matlab提供了用于建立控件对象的函数uicontrol，其调用格式为：

对象句柄=uicontrol(图形窗口句柄，属性名1，属性值1，属性名2，属性值2，…)

其中各个属性名及可取的值和后面将介绍的uimenu函数相似，但也不尽相同。

例　建立数制转换对话框。在左边输入一个十进制整数和2～16之间的数，单击“转换”按钮能在右边得到十进制数所对应的2～16进制字符串，单击“退出”按钮退出对话框。

程序如下：

所建立的数制转换对话框如下：

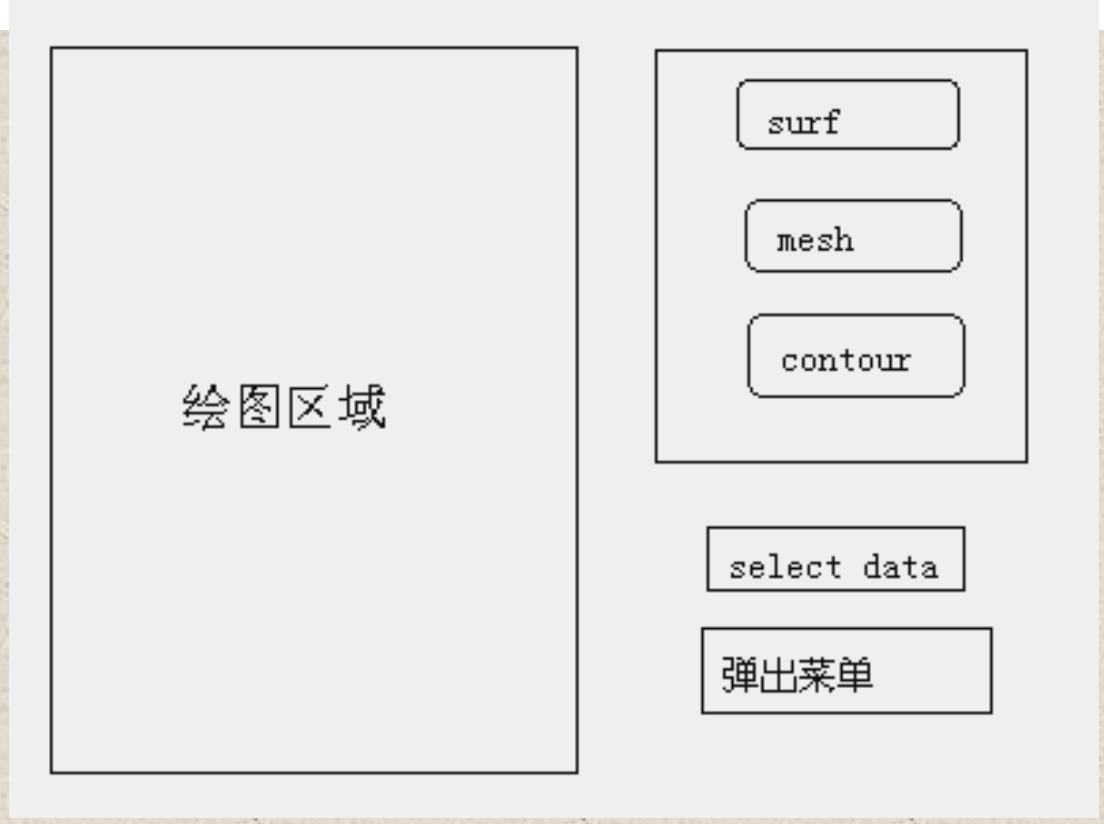
```
hf=figure('Color',[0,1,1],'Position',[100,200,400,200],...
  'Name','数制转换','NumberTitle','off','MenuBar','none');
 uicontrol(hf,'Style','Text', 'Units','normalized',...
  'Position',[0.05,0.8,0.45,0.1],'Horizontal','center',...
  'String','输 入 框','Back',[0,1,1]);
 uicontrol(hf,'Style','Text','Position',[0.5,0.8,0.45,0.1],...
  'Units','normalized','Horizontal','center',...
  'String','输 出 框','Back',[0,1,1]);
 uicontrol(hf,'Style','Frame','Position',[0.04,0.33,0.45,0.45],...
  'Units','normalized','Back',[1,1,0]);
 uicontrol(hf,'Style','Text','Position',[0.05,0.6,0.25,0.1],...
  'Units','normalized','Horizontal','center',...
  'String','十进制数','Back',[1,1,0]);
 uicontrol(hf,'Style','Text','Position',[0.05,0.4,0.25,0.1],...
  'Units','normalized','Horizontal','center',...
  'String','2～16进制','Back',[1,1,0]);
 he1=uicontrol(hf,'Style','Edit','Position',[0.25,0.6,0.2,0.1],...
  'Units','normalized','Back',[0,1,0]);
 he2=uicontrol(hf,'Style','Edit','Position',[0.25,0.4,0.2,0.1],...
  'Units','normalized','Back',[0,1,0]);
 uicontrol(hf,'Style','Frame','Position',[0.52,0.33,0.45,0.45],...
  'Units','normalized','Back',[1,1,0]);
 ht=uicontrol(hf,'Style','Text','Position',[0.6,0.5,0.3,0.1],...
  'Units','normalized','Horizontal','center','Back',[0,1,0]);
 COMM=['n=str2num(get(he1,''String''));','b=str2num(get(he2,''String''));',...
  'dec=trdec(n,b);','set(ht,''string'',dec);'];
 uicontrol(hf,'Style','Push','Position',[0.18,0.1,0.2,0.12],...
   'String','转 换','Units','normalized','Call',COMM);
 uicontrol(hf,'Style','Push','Position',[0.65,0.1,0.2,0.12],...
  'String','退 出','Units','normalized','Call','close(hf)');
```

程序调用了trdec.m函数文件，该函数的作用是将任意十进制整数转换为2~16进制字符串。trdec.m函数文件如下：

```
function dec=trdec(n,b)
   ch1='0123456789ABCDEF';   %十六进制的16个符号
   k=1;
   while n~=0          %不断除某进制基数取余直到商为0
      p(k)=rem(n,b);
      n=fix(n/b);
      k=k+1;
   end
   k=k-1;
   strdec='';
   while k>=1                  %形成某进制数的字符串
      kb=p(k);
      strdec=strcat(strdec,ch1(kb+1:kb+1));
      k=k-1;
   end
   dec=strdec;
```

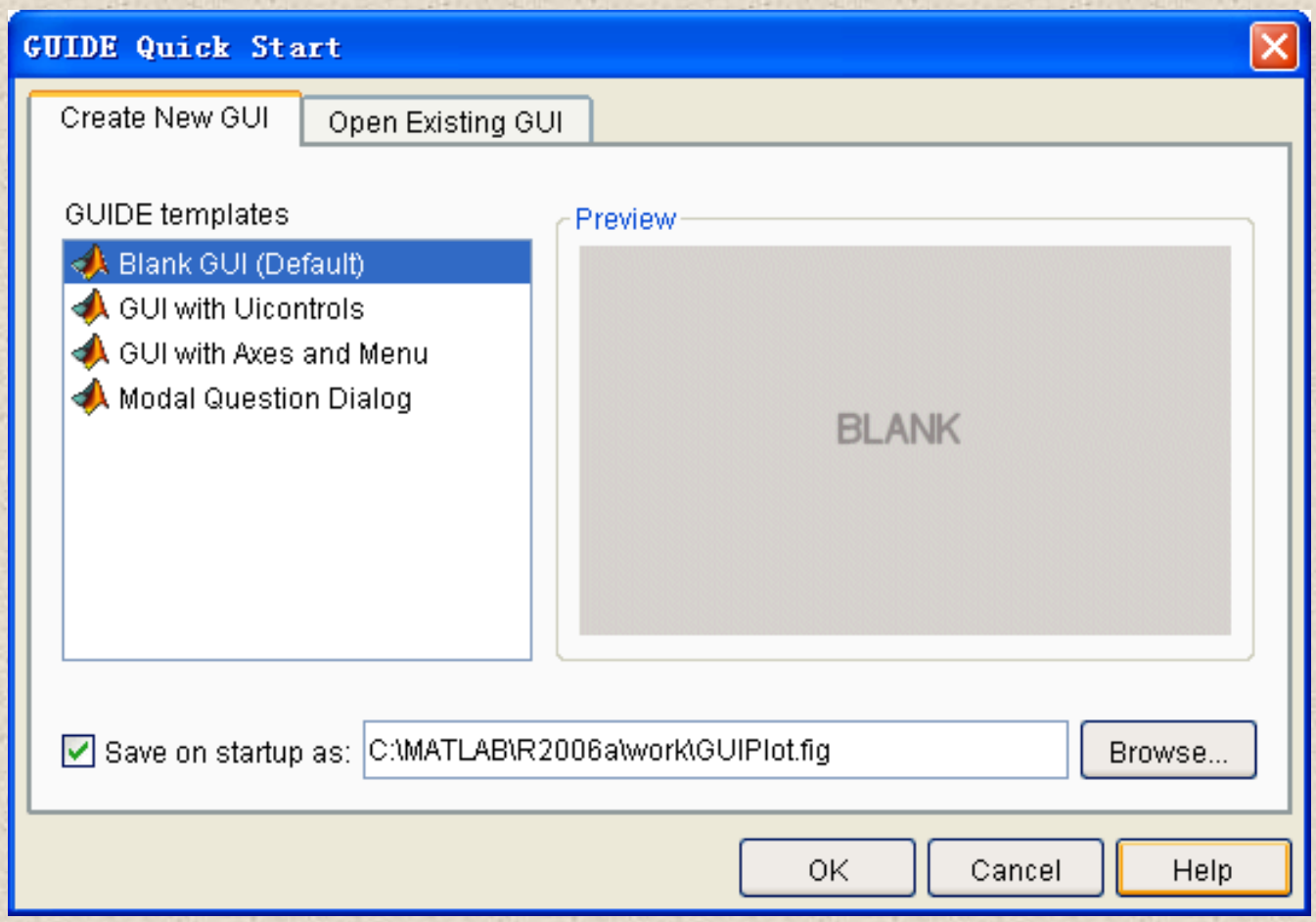
Guide创建GUI 实例

创建一个 GUI，该 GUI 实现三维图形的绘制。预创建界面中应包含一个绘图区域；一个面板，其中包含三个绘图按钮，分别实现表面图、网格图和等值线的绘制；一个弹出菜单，用以选择数据类型，并且用静态文本进行说明。其草图如图所示。

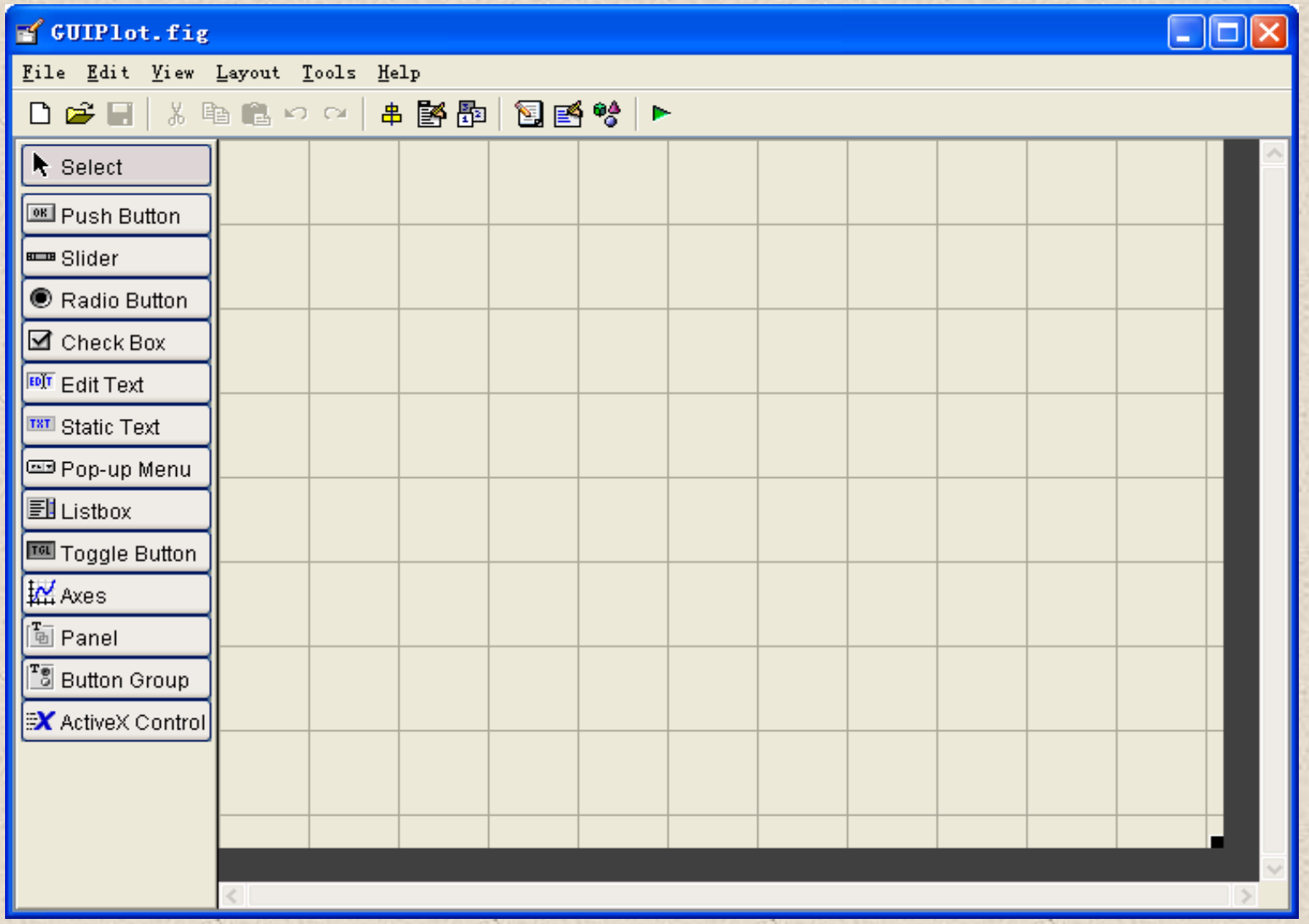


点击工具栏中的 GUIDE 图标，启动 GUIDE，系统打开界面如图所示。

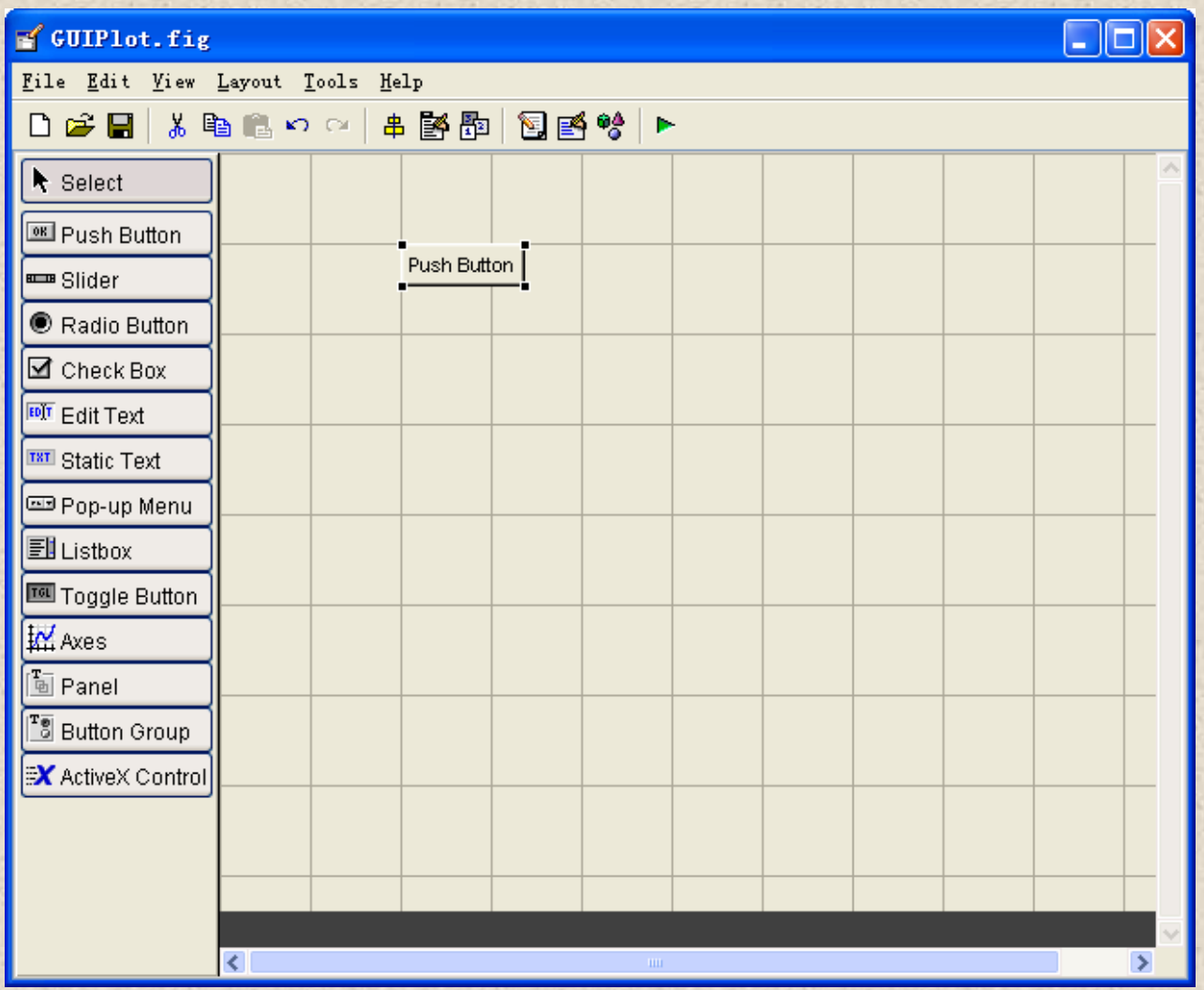
选择新建 GUI 标签，并选择新建空的 GUI，选中下面的保存选项，输入文件名，得到结果如图所示。



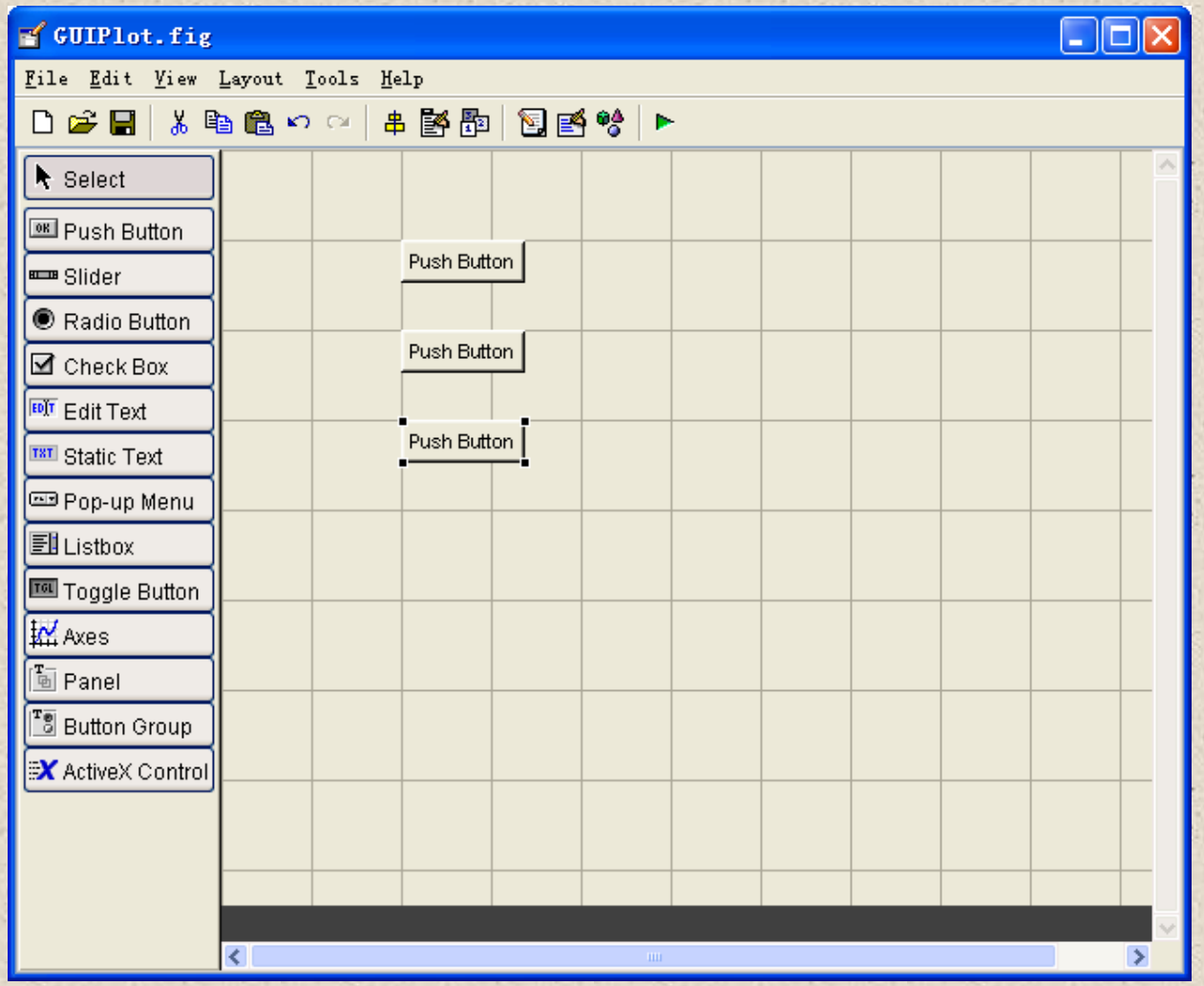
该窗口中包括菜单栏、控制工具栏、GUI 控件面板、GUI 编辑区域等，在 GUI 编辑区域右下脚，可以通过鼠标拖曳的方式改变 GUI 界面的大小。



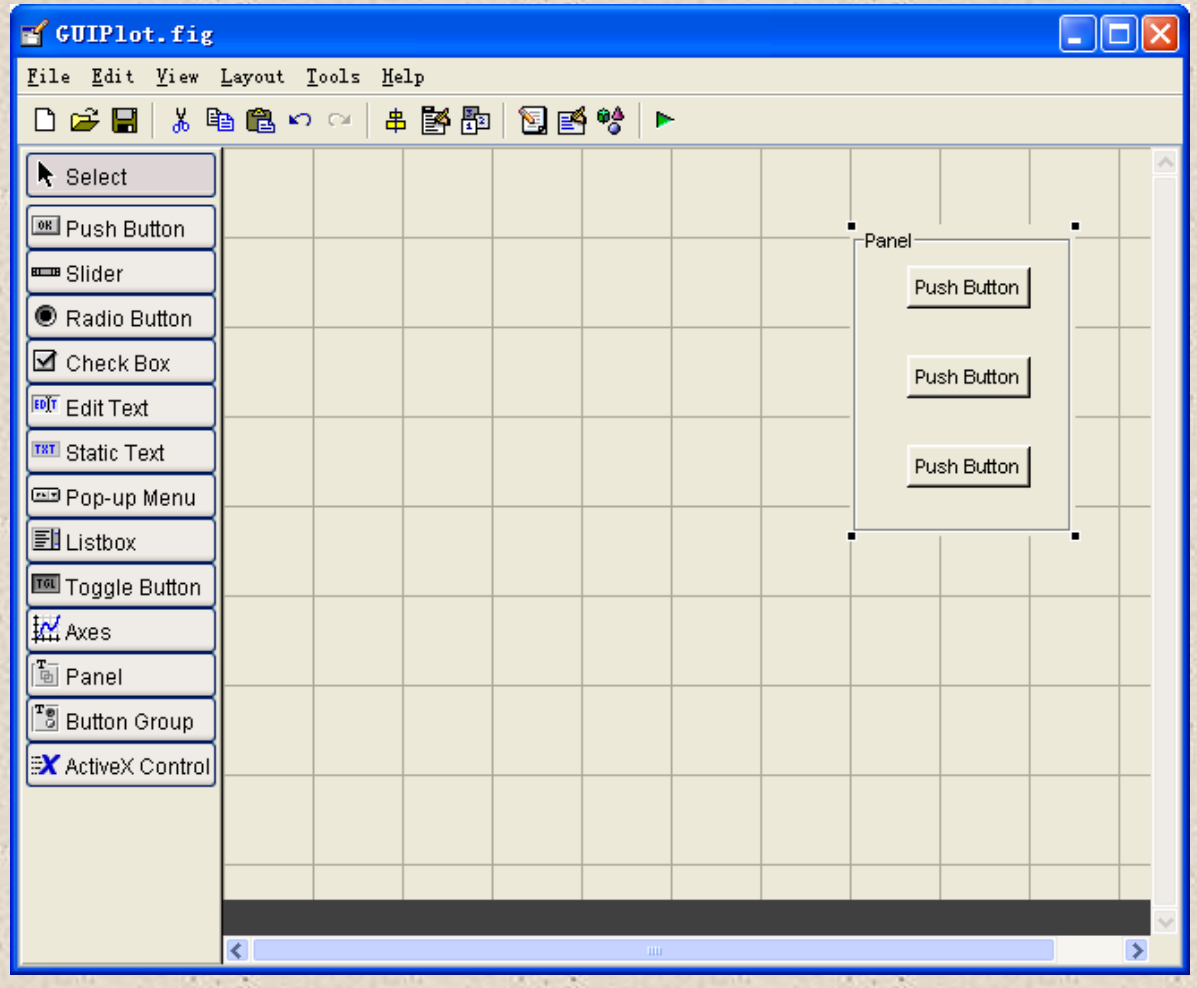
首先向界面中添加按钮。用鼠标点击 **Push Button**，并拖曳至 **GUI** 编辑区。



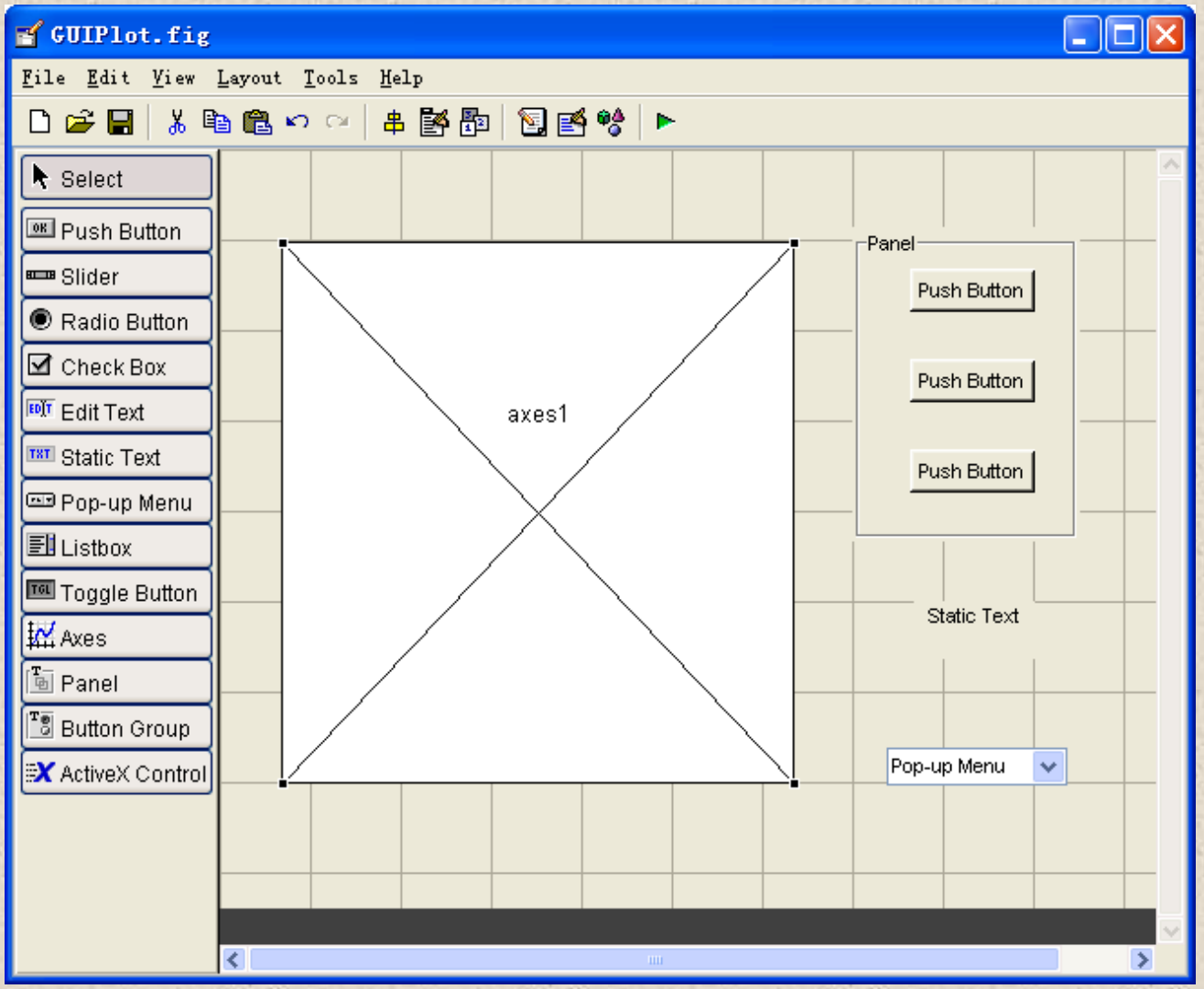
在该按钮上点击右键，选择 **Duplicate**，将该按钮复制两次，并移动到合适的位置。



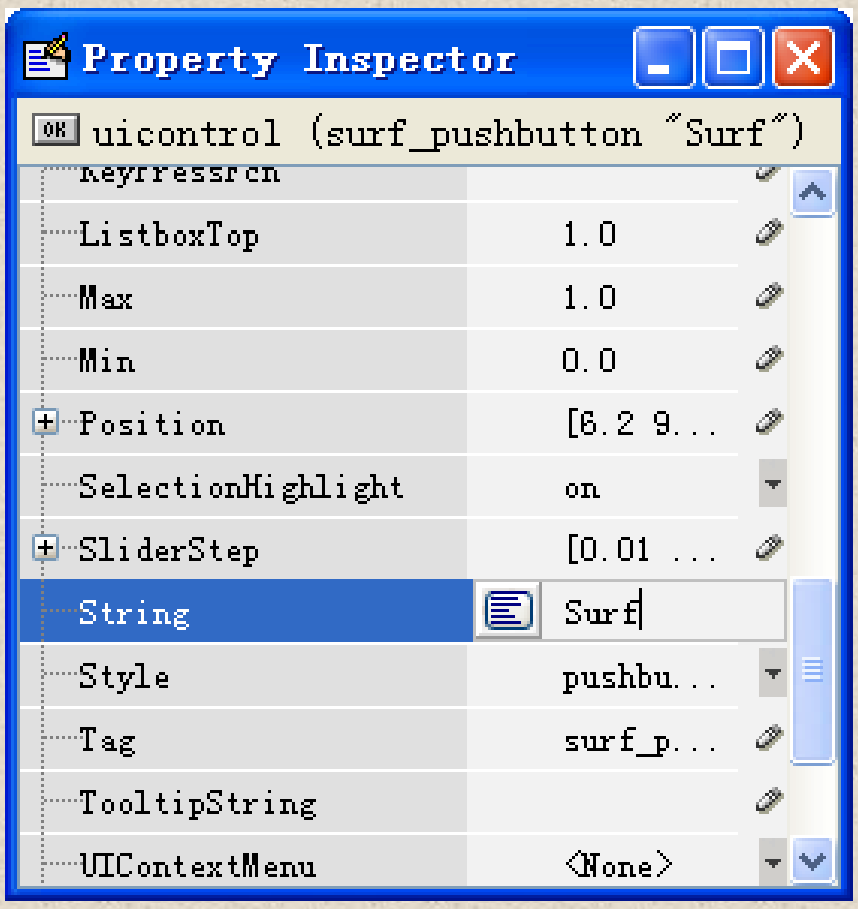
然后将这三个按钮添加到面板中。在编辑区的右侧添加面板，并将三个按钮移动到面板中。



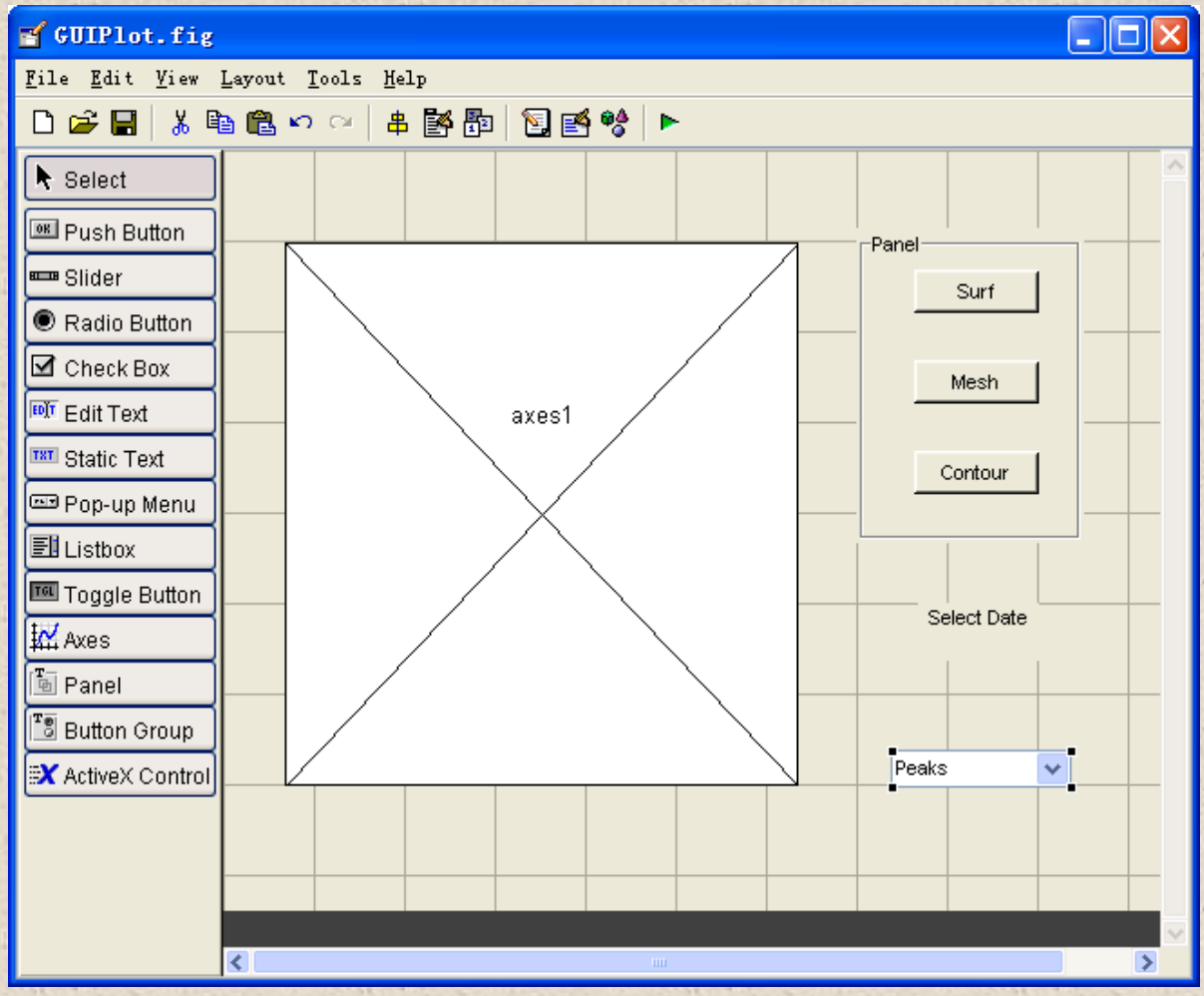
下面继续向其中添加静态文本、弹出菜单和绘图区 。



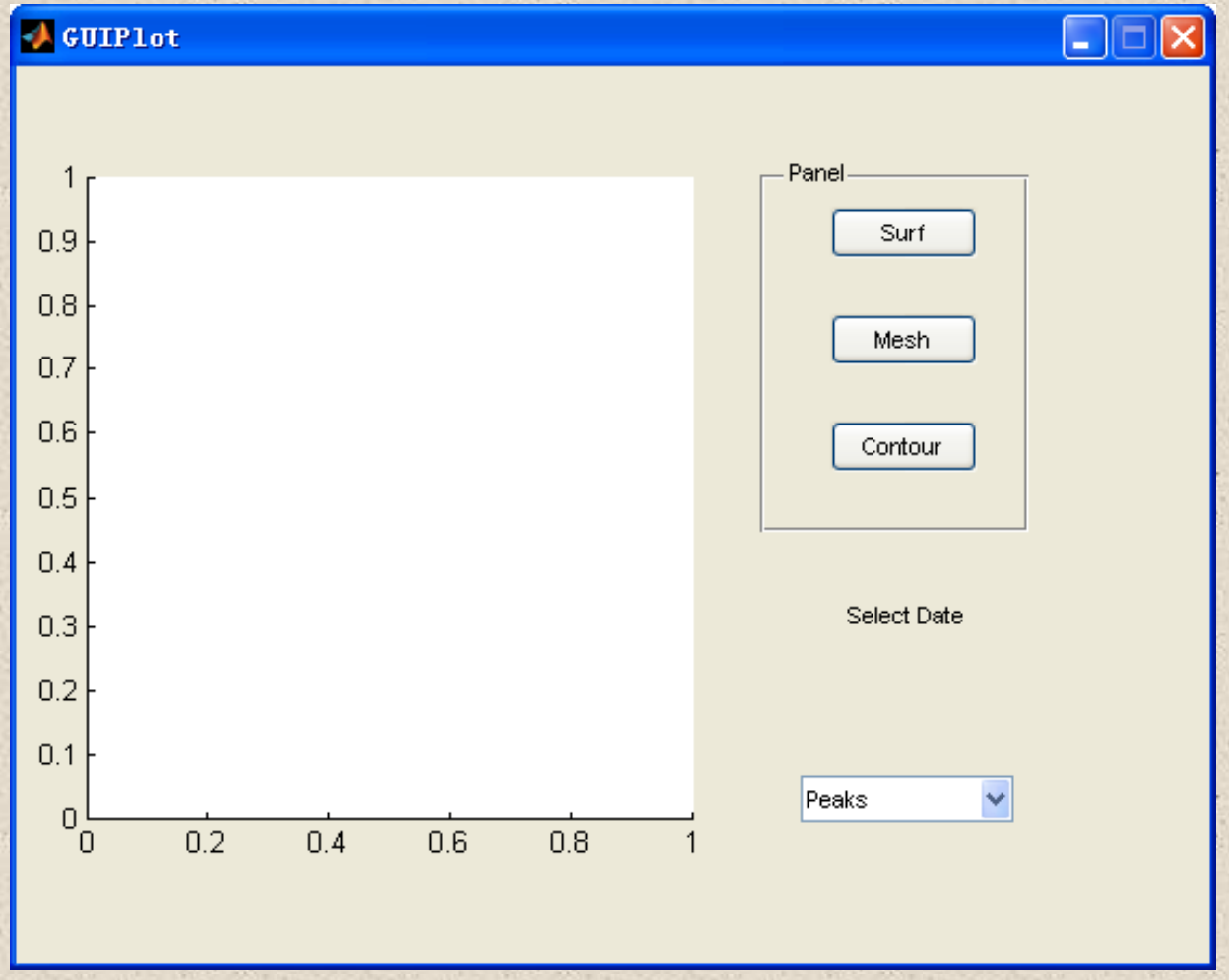
点击工具栏中 **Property Inspector**，打开属性编辑器。设置各个控件的属性，如设置按钮的属性，设置第一个按钮的显示文字为 **Surf**，标签名为 **surf_pushbutton**。



设置其他控件的属性。


GUIPlot.fig
File Edit View Layout Tools Help
Select
Push Button
Slider
Radio Button
Check Box
Edit Text
Static Text
Pop-up Menu
Listbox
Toggle Button
Axes
Panel
Button Group
ActiveX Control
axes1
Panel
Surf
Mesh
Contour
Select Date
Peaks

点击工具栏中的绿色箭头，运行该 GUI。

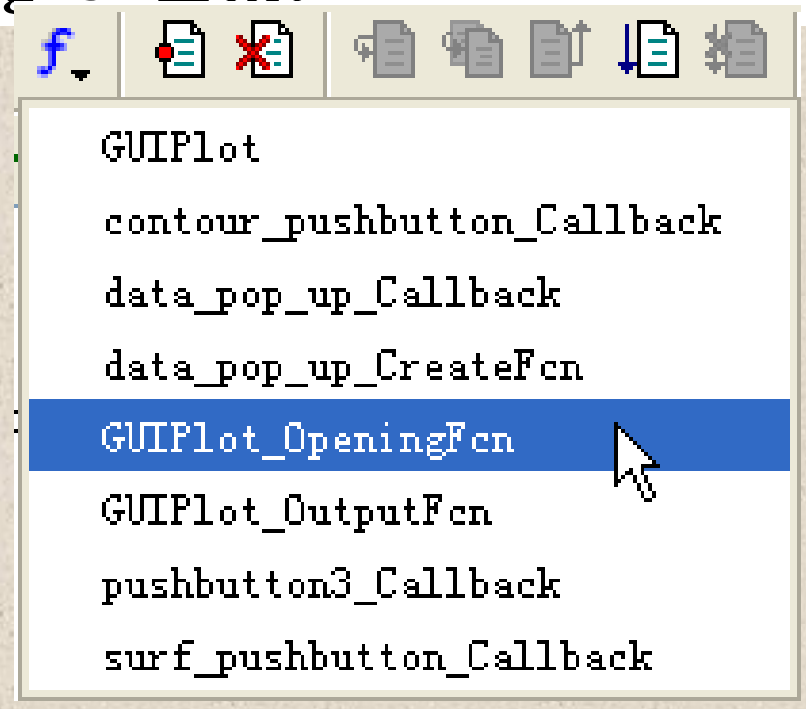


编写响应函数

在创建 GUI 时系统已经为其自动生成了 M 文件，该文件中包含 GUI 中控件对应的响应函数，及系统函数等。

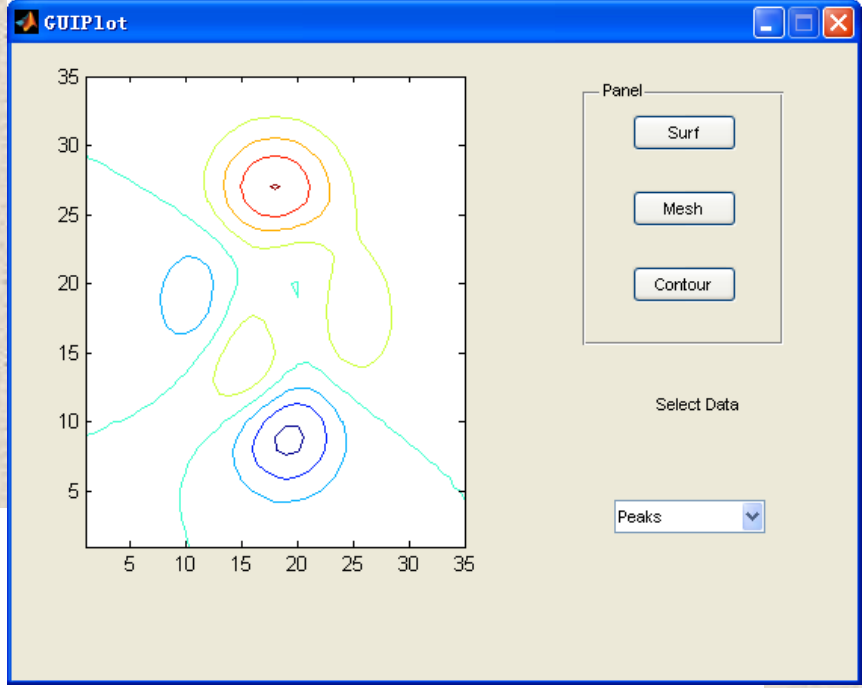
首先编写数据生成函数。

在 GUI 向导中点击 M-file Editor，打开 M 文件编辑器。打开的编辑器中为该 GUI 对应的 M 文件。点击编辑器中的函数查看工具，显示其中包含的函数，选择 GUIPlot_OpeningFcn 函数。



该函数中已有部分内容，现在其中添加数据生成函数。添加后该函数的内容为：

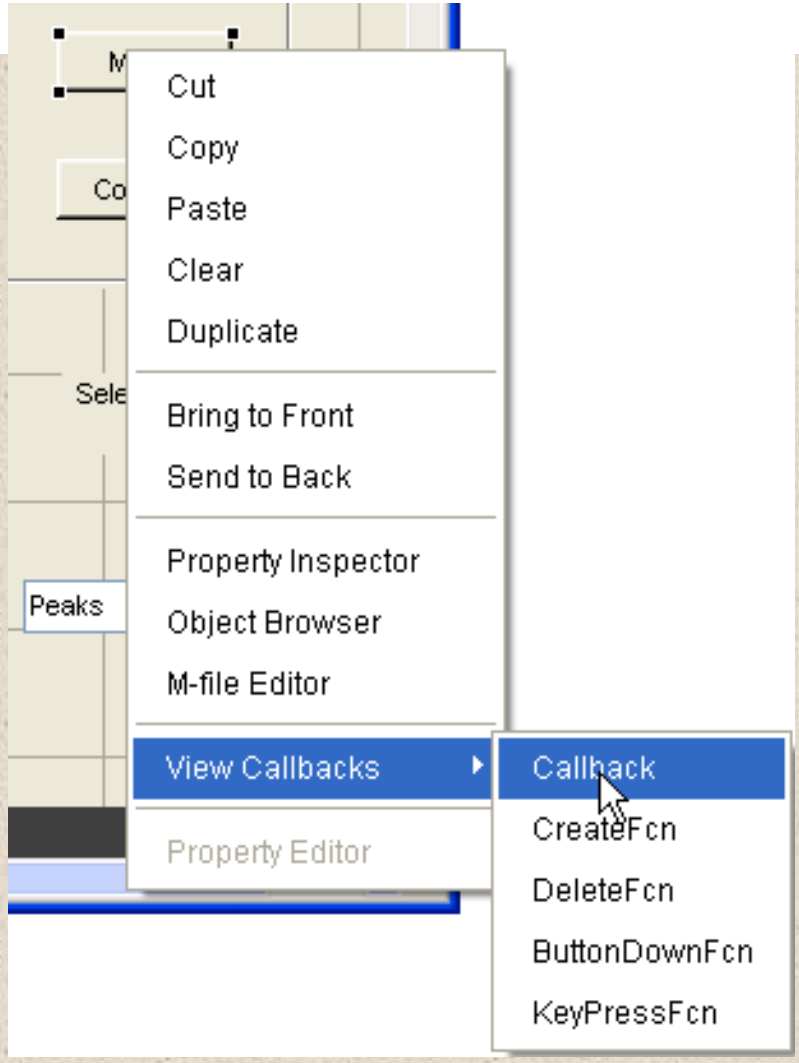
```
% --- Executes just before GUIPlot is made visible.
function GUIPlot_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUIPlot (see 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;
contour(handles.current_data)
% Choose default command line output for GUIPlot
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes GUIPlot wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

该函数首先生成三组数据，并设置初始数据为 peaks 数据，且初始图形为等值线。修改该函数后再次运行 GUI，得到结果如图 。

继续修改按钮及弹出菜单的响应函数。用户可以通过 M 文件编辑器中的函数查看工具查找相应函数，或者在 GUI 编辑器中右键点击相应控件，选择 View Callbacks 中的 Callback，系统自动打开 M 文件编辑器，并且光标位于相应的函数处。



弹出菜单的响应函数：

```
% --- Executes on selection change in data_pop_up.
function data_pop_up_Callback(hObject, eventdata, handles)
% hObject    handle to data_pop_up (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% 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)
% Hints: contents = get(hObject,'String') returns data_pop_up contents as cell array
%        contents{get(hObject,'Value')} returns selected item from data_pop_up
```

该函数首先取得弹出菜单的 String 属性和 Value 属性，后通过分支语句选择数据

。

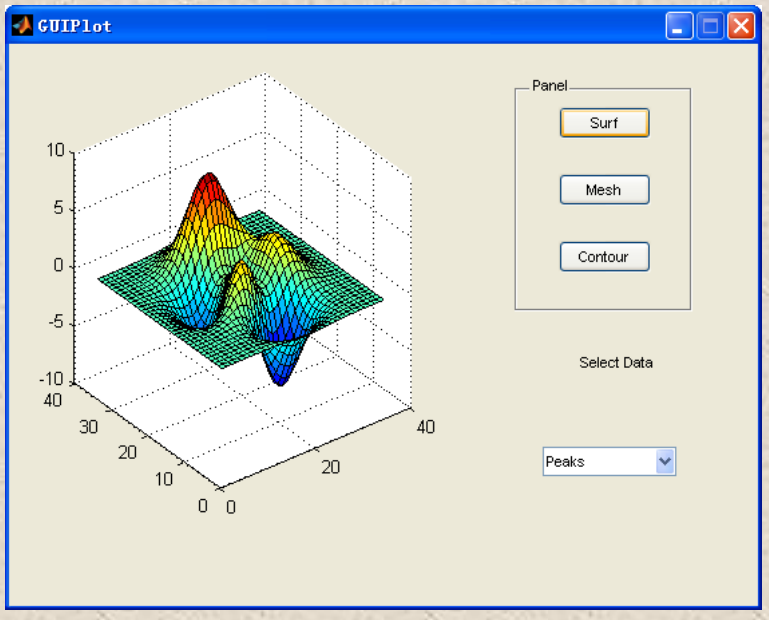
三个按钮的响应函数分别为：

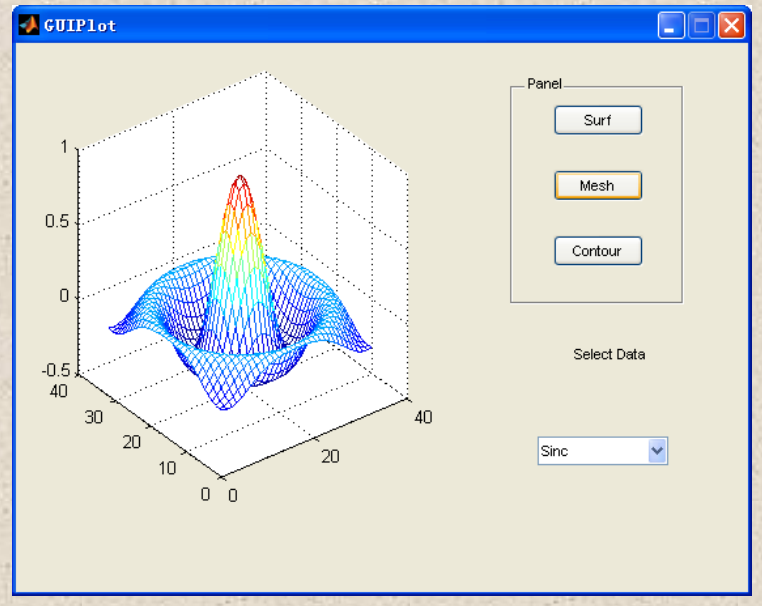
```
% --- Executes on button press in surfpushbutton.
function surfpushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to surfpushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Display surf plot of the currently selected data.
surf(handles.current_data);
```

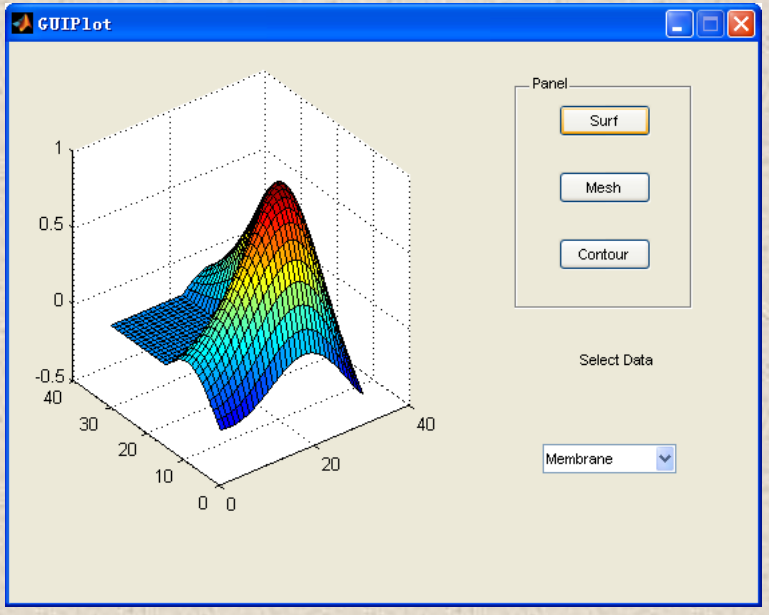
```
% --- Executes on button press in meshpushbutton.
function meshpushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to meshpushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Display mesh plot of the currently selected data.
mesh(handles.current_data);
```

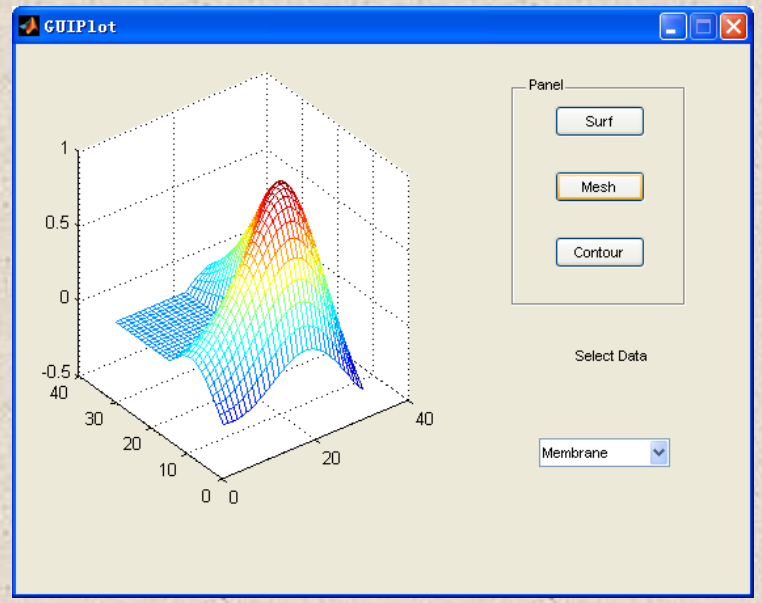
```
% --- Executes on button press in contourpushbutton.
function contourpushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to contourpushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Display contour plot of the currently selected data.
contour(handles.current_data);
```

再次运行该 GUI，得到最后的结果。









三、编写 GUI 代码

在创建 GUI 界面后，需要为界面中的控件编写响应函数，这些函数决定当事件发生时的具体操作。

- 通常情况下，一个 GUI 包含两个文件，一个 FIG 文件和一个 M 文件。
- FIG 文件的扩展名为 .fig，是一种 MATLAB 文件，其中包含 GUI 的布局及其中包含的所有控件的相关信息。FIG 文件为二进制文件，只能通过 GUI 向导进行修改。
- M 文件扩展名为 .m，其中包含 GUI 的初始代码及相关响应函数的模板。用户需要在该文件中添加响应函数的具体内容。
- M 文件通常包含一个与文件同名的主函数，各个控件对应的响应函数，这些响应函数为主函数的子函数。

内容	描述
注释	程序注释。当在命令行调用 **help** 时显示
初始化代码	**GUI** 向导的初始任务。
Opening 函数	在用户访问 **GUI** 之前进行初始化任务
Output 函数	在控制权由 **Opening** 函数向命令行转移过程中向命令行返回输出结果
响应函数	这些函数决定控件的操作的结果。**GUI** 为事件驱动的程序，当事件发生时，系统调用相应的函数进行执行

通常情况下，在保存 GUI 时，向导会自动向 M 文件中添加响应函数。另外，用户也可以向 M 文件中添加其他的响应函数。通过向导，用户可以以下面两种方式向 M 文件中添加响应函数。

- 点击右键，在右键菜单的 View callbacks 中选择需要添加的响应函数类型，向导自动将其添加到 M 文件中，并在文本编辑器中打开该函数，用户可以对其进行编辑。如果该函数已经存在，则打开该函数。
- 在 View 菜单中，选择 View callbacks 中需要添加的响应函数类型。

响应函数

1．响应函数的定义及类型

响应函数与特定的 GUI 对象关联，或与 GUI 图形关联。当事件发生时，MATLAB 调用该事件所激发的响应函数。

GUI 图形及各种类型的控件有不同的响应函数类型。每个控件可以拥有的响应函数定义为该控件的属性，例如，一个按钮可以拥有五种响应函数属性：ButtonDownFcn、Callback、CreateFcn、DeleteFcn 和 KeyPressFcn。用户可以同时为每个属性创建响应函数。GUI 图形本身也可以拥有特定类型的响应函数。

每一种类型的响应函数都有其触发机制或者事件，MATLAB 中的响应函数属性、对应的触发事件及可以应用的控件如表 所示。

响应函数属性	触发事件	可用控件
ButtonDownFcn	用户在其对应控件 5 个象素范围内按下鼠标	坐标系、图形、按钮组、面板、用户接口控件
Callback	控制操作，用户按下按钮或选中一个菜单项	右键菜单、菜单、用户接口控件
CloseRequestFcn	关闭图形时执行	图形
CreateFcn	创建控件时初始化控件，初始化后显示该控件	坐标系、图形、按钮组、右键菜单、菜单、面板、用户接口控件
DeleteFcn	在控件图形关闭前清除该对象	坐标系、图形、按钮组、右键菜单、菜单、面板、用户接口控件

响应函数属性	触发事件	可用控件
KeyPressFcn	用户按下控件或图形对应的键盘	图形、用户接口控件
ResizeFcn	用户改变面板、按钮组或图形的大小，这些控件的 **Resize** 属性需处于 **On** 状态	按钮组、面板、图形
SelectionChangeFcn	用户在一个按钮组内部选择不同的按钮，或改变开关按钮的状态	按钮组
WindowButtonDownFcn	在图形窗口内部按下鼠标	图形
WindowButtonMotionFcn	在图形窗口内部移动鼠标	图形
WindowButtonUpFcn	松开鼠标按钮	图形

一个 GUI 中包含多个控件，GUIDE 中提供了一种方法，用于指定每个控件所对应的响应函数。

GUIDE 通过每个控件的响应属性将控件与对应的响应函数相关联。在默认情况下，GUIDE 将每个控件的最常用的响应属性设置为 %automatic。如每个按钮有五个响应属性，ButtonDownFcn、Callback、CreateFcn、DeleteFcn 和 KeyPressFcn，GUIDE 将其 Callback 属性设置为 %automatic。用户可以通过属性编辑器将其他响应属性设置为 %automatic。

当再次保存 GUI 时，GUIDE 将 %automatic 替换为响应函数的名称，该函数的名称由该控件 Tag 属性及响应函数的名称组成

3．响应函数的语法与参数

MATLAB 中对响应函数的语法和参数有一些约定，在 GUI 向导创建响应函数并写入 M 文件时便遵守这些约定。如下面为按钮的响应函数模板：

```
% --- Executes on button press in pushbutton1.
```

(说明该函数的触发事件)

```
function pushbutton1_Callback(hObject, eventdata, handles) %#ok
```

(函数定义行)

```
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

(对输入参数进行说明)

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

用户可以在这里输入函数的其他内容

GUI 向导创建函数模板时，函数的名称为：

控件标签（Tag属性）＋下划线＋函数属性。

如上面的模板中，控件标签为 **pushbutton1**，响应函数的属性为 **Callback**，因此函数名为 **pushbutton1_Callback**。

在添加控件后第一次保存 GUI 时，向导向 M 文件中添加相应的响应函数，函数名由当前 Tag 属性的当前值确定。因此，如果需要改变 Tag 属性的默认值，请在保存 GUI 前进行。

响应函数包含如下参数：

- hObject，对象句柄，如触发该函数的控件的句柄；
- eventdata，保留参数；
- handles，为一个结构体，包含图形中所有对象的句柄，如：

```
handles =
        figure1: 160.0011
          edit1: 9.0020
        uipanel1: 8.0017
      popupmenu1: 7.0018
     pushbutton1: 161.0011
         output: 160.0011
```

其中包含了文本编辑框、面板、弹出菜单和按钮。

GUI 向导创建 handles 结构体，并且在整个程序运行中保持其值不变。所有的响应函数使用该结构体作为输入参数。

4．初始化响应函数

GUI 的初始化函数包括 Opening 函数和Output 函数。

在每个 GUI M 文件中 opening 函数是第一个调用的函数。该函数在所有控件创建完成后，GUI 显示之前运行。用户可以通过 opening 函数设置程序的初始任务，如创建数据、读入数据等。

通常 opening 函数的名称为“M 文件名＋_ OpeningFcn”，如下面的初始模板：

```
% --- Executes just before mygui is made visible.
function mygui_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 mygui (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mygui wait for user response (see UIRESUME)
% uiwait(handles.mygui);
```

其中文件名为 **mygui**，函数名为 **mygui_OpeningFcn**。该函数包含四个参数，第四个参数 **varargin** 允许用户通过命令行向 **opening** 函数传递参数。**opening** 函数将这些参数添加到结构体 **handles** 中，供响应函数调用。

该函数中包含三行语句，如下。

handles.output = hObject ，向结构体 **handles** 中添加新元素 **output**，并将其值赋为输入参数 **hObject**，即 **GUI** 的句柄。该句柄供 **output** 函数调用。

guidata(hObject,handles)，保存**handles**。用户必须通过 **guidata** 保存结构体 **handles** 的任何改变。

uiwait(handles.mygui)，在初始情况下，该语句并不执行。该语句用于中断 **GUI** 执行等待用户反应或 **GUI** 被删除。如果需要该语句运行，删除前面的“**%**”即可。

output 函数用于向命令行返回 **GUI** 运行过程中产生的输出结果。该函数在 **opening** 函数返回控制权和控制权返回至命令行之间运行。因此，输出参数必须在 **opening** 函数中生成，或者在 **opening** 函数中调用 **uiwait** 函数中断 **output** 的执行，等待其他响应函数生成输出参数。

output 函数的函数名为“M 文件名＋_OutputFcn”，如下面的初始模板：

```
% --- Outputs from this function are returned to the command line.
function varargout = mygui_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;
```

该函数的函数名为 mygui_OutputFcn。output 函数有一个输出参数 varargout。在默认情况下，output 函数将 handles.output 的值赋予 varargout，因此 output 的默认输出为 GUI 的句柄。用户可以通过改变 handles.output 的值改变函数输出结果。

$$x = \sin\left(2\pi f_1 t\right) + \sin\left(2\pi f_2 t\right)$$

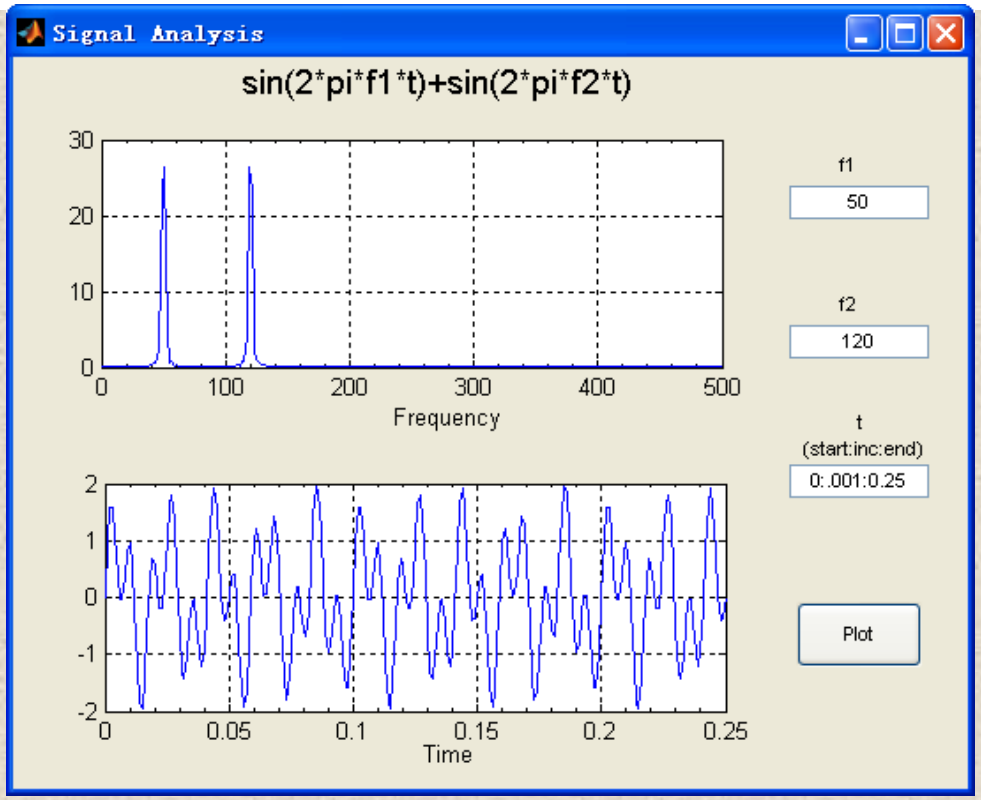
本例 GUI 的功能为在一个界面中绘制两个图形，为 上述函数的图像及其快速傅立叶（FFT）的图像。其中参数、和的值由界面输入。

该 GUI 的界面图形如图：

该 GUI 中需要解决的问题有：

（1）控制绘图命令的目标坐标系

（2）通过文本编辑器输入 MATLAB 表达式的参数

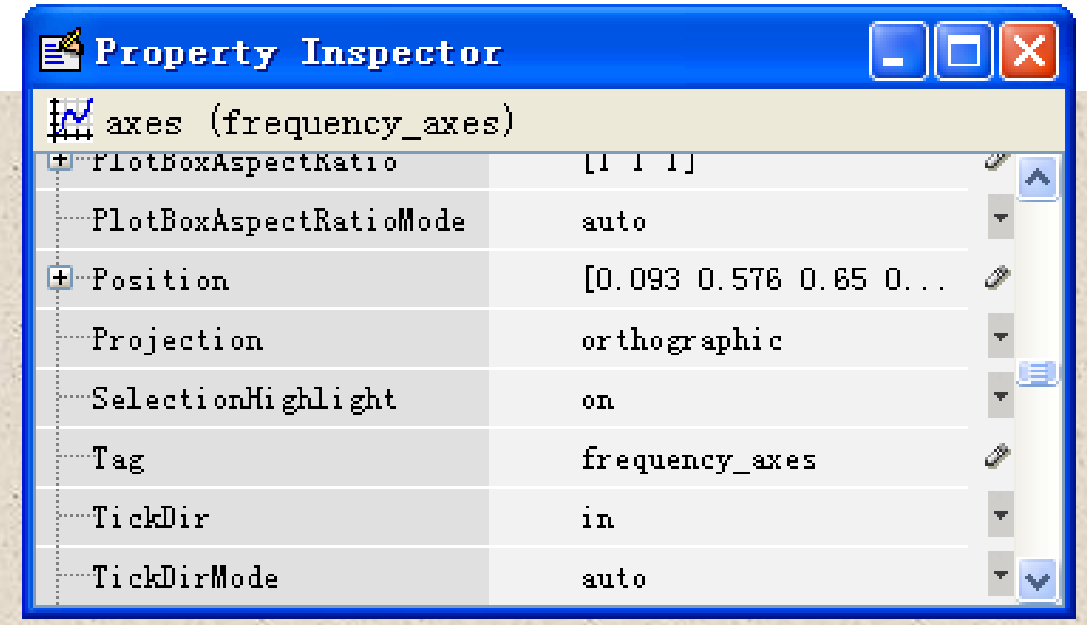


1．创建 GUI 界面

打开 GUIDE，新建 GUI，保存为 two_axes。向其中添加控件并设置这些控件的属性。

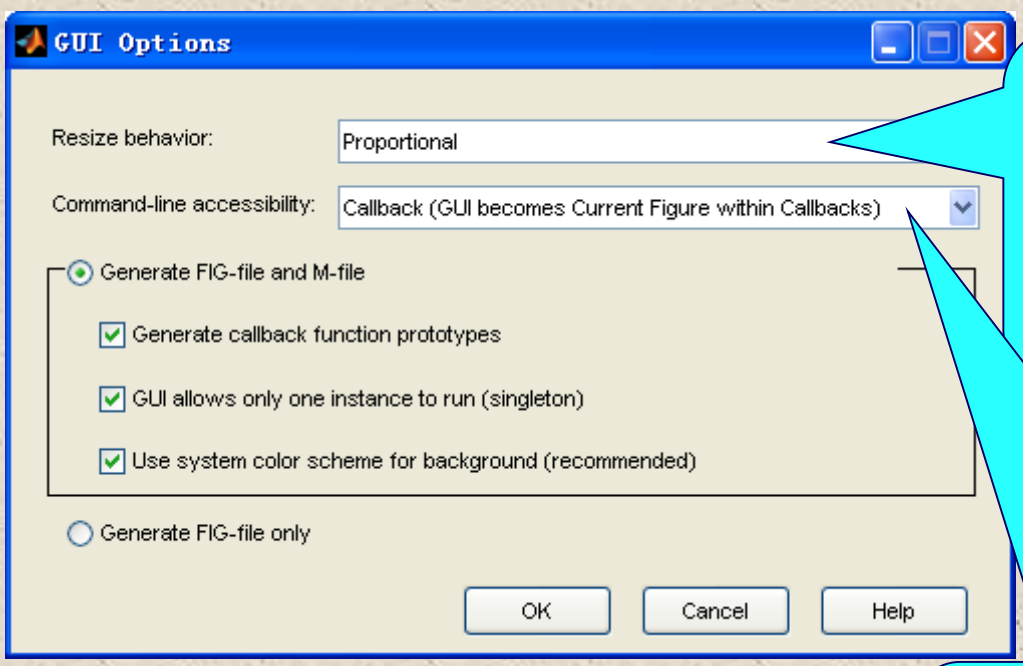
设置 f1 的 Tag 属性为 f1_input，初始值为 50，f2的 Tag 属性为 f2_input，的初始值为 120，t的 Tag 属性为 t_input，的初始值为 0:.001:0.25。这些初始值为打开该 GUI 时的默认值。

由于该 GUI 中包含两个图形，在绘制图形时必须指定坐标系。为实现这一功能，可以使用 handles 结构体，该结构体中包含 GUI 中所有控件的句柄。该结构体中的域名为控件的 Tag 属性值。在本 GUI 中，我们设置绘制函数时域的坐标系的句柄为 time_axes，绘制频域图形的坐标系为 frequency_axes，如图所示。



设置后，在响应函数中，可以通过 handles.frequency_axes 实现对该坐标系的调用。

设置控件完成后，设置 GUI 的属性。在 Tools 菜单中选择 GUI options…，弹出窗口如图所示。



设置 Resize behavior 为 Proportional，允许用户改变该 GUI 的大小，并且改变窗口大小时，GUI 中的控件大小按照比例同时改变。

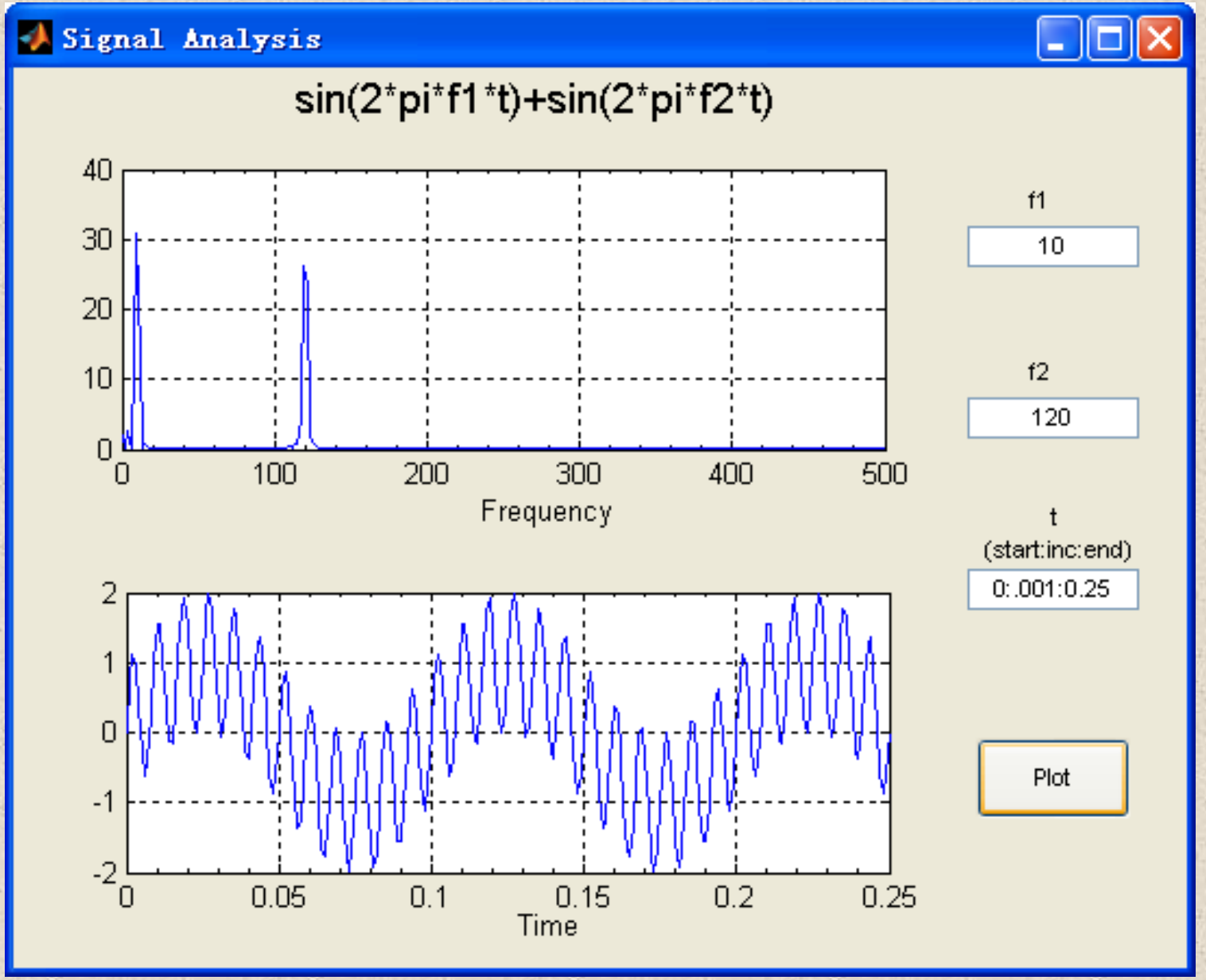
设置 Command-line accessibility 为 Callback 允许响应函数调用句柄，因此可以在响应函数中向坐标系中绘制图形。

该 GUI 需要从界面中读入参数，利用读入的参数计算函数的快速傅立叶变换，之后绘制图形。需要的响应函数只有一个，即按钮的响应函数。该函数的内容为：

```
function plot_button_Callback(hObject, eventdata, handles)
% hObject    handle to plot_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get user input from GUI
f1 = str2double(get(handles.f1_input,'String'));
f2 = str2double(get(handles.f2_input,'String'));
t = eval(get(handles.t_input,'String'));
```

```
% Calculate data
x = sin(2*pi*f1*t) + sin(2*pi*f2*t);
y = fft(x,512);
m = y.*conj(y)/512;
f = 1000*(0:256)/512;
% Create frequency plot
axes(handles.frequency_axes) % Select the proper axes
plot(f,m(1:257))
set(handles.frequency_axes,'XMinorTick','on')
grid on
% Create time plot
axes(handles.time_axes) % Select the proper axes
plot(t,x)
set(handles.time_axes,'XMinorTick','on')
grid on
```

代码完成后保存，运行该 GUI，得到结果如图：


Signal Analysis
sin(2*pi*f1*t)+sin(2*pi*f2*t)
40
30
20
10
0
0
100
200
300
400
500
Frequency
2
1
0
-1
-2
0
0.05
0.1
0.15
0.2
0.25
Time
f1
10
f2
120
t
(start:inc:end)
0:.001:0.25
Plot

四、程序菜单设计

1. 建立用户菜单

建立自定义的用户菜单的函数为**uimenu**，格式为：

Hm=uimenu(Hp，属性名1，属性值1，属性名2，属性值2，…)

功能：创建句柄值为Hm的自定义的用户菜单。其中Hp为其父对象的句柄，属性名和属性值构成属性二元对，定义用户菜单的属性。

因其调用方法不同，该函数可以用于建立一级菜单项和子菜单项。

- 建立一级菜单项的函数调用格式为：

一级菜单项句柄=uimenu(图形窗口句柄，属性名1，属性值1，属性名2，属性值2，…)

- 建立子菜单项的函数调用格式为：

子菜单项句柄=uimenu(一级菜单项句柄，属性名1，属性值1，属性名2，属性值2，…)

2．菜单对象常用属性

菜单对象除具有Children（子对象），Parent（父对象），Tag（标签），Type（类型），UserData（用户数据），Enable（使能）和 Visible（可见性）等公共属性，还有一些常用的特殊属性，如回调（callback）属性和菜单名(label)。另外，用户菜单的外观有四个属性：Position（位置），Separator（分隔线），checked（检录符）和ForeGroundColor（前景颜色）。

① Tag属性

Tag属性的取值是字符串，它定义了该菜单对象的一个标识值。定义了Tag属性后，在任何程序中都可以通过这个标识值找出该菜单对象。

② Type属性

Type属性的取值总是uimenu，这个属性值标明图形对象的类型。对菜单对象，其类型就是uimenu，用户不能改写这个属性。

③ UserData属性

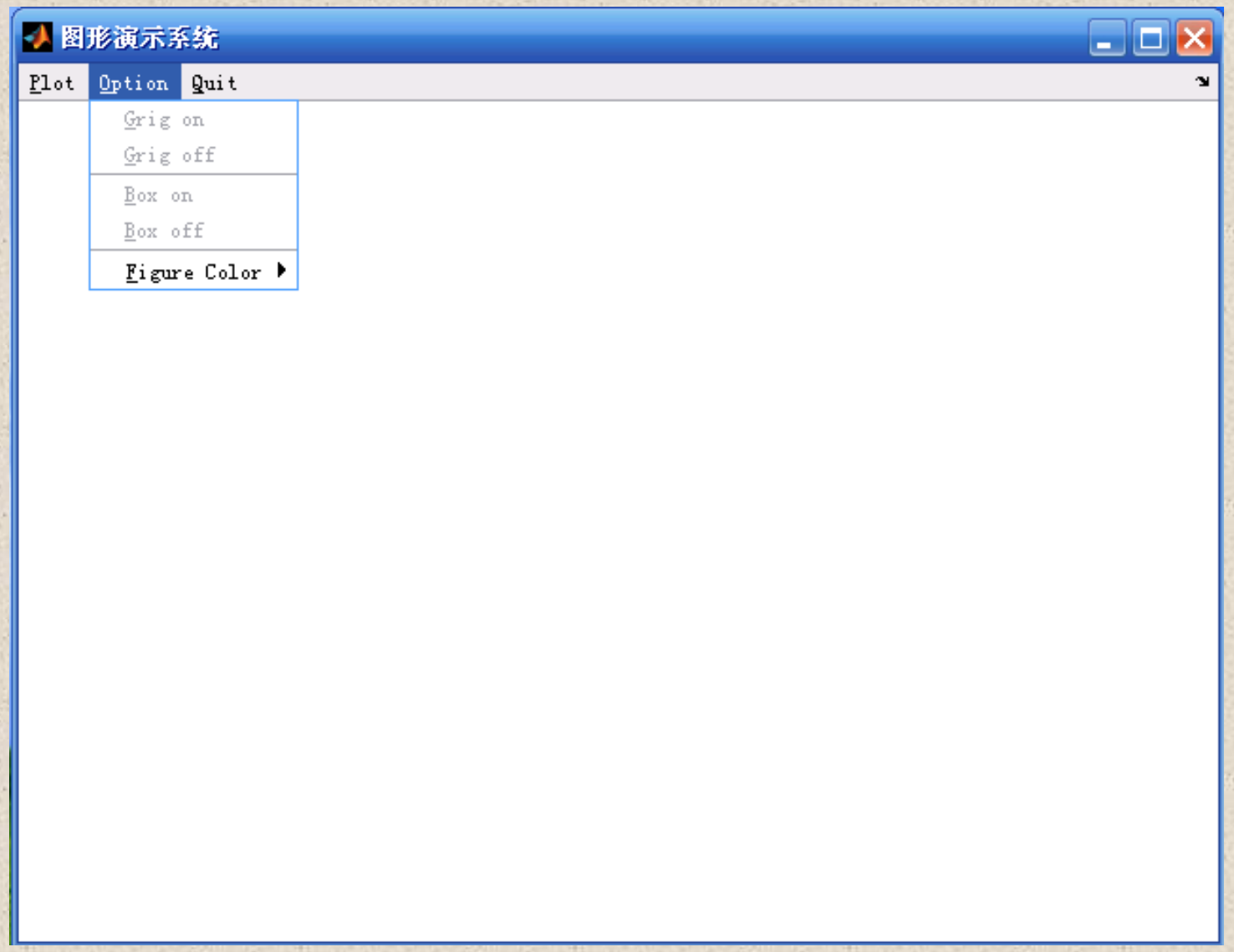
UserData属性的取值是一个矩阵，缺省值为空矩阵，用户可以在这个属性中保存与该菜单对象相关的重要数据或信息，借此可以达到传递数据或信息的目的。可以用set和get函数访问该属性。

例 **建立“图形演示系统”菜单。菜单条中含有3个菜单项：Plot、Option和Quit。Plot中有Sine Wave和Cosine Wave两个子菜单项，分别控制在本图形窗口画出正弦和余弦曲线。Option菜单项的内容为：Grid on和Grid off控制给坐标轴加网格线，Box on和Box off控制给坐标轴加边框，而且这4项只有在画有曲线时才是可选的。Figure Color控制图形窗口背景颜色。Quit控制是否退出系统。**

程序如下：

```
screen=get(0,'ScreenSize');
W=screen(3);H=screen(4);
figure('Color',[1,1,1],'Position',[0.2*H,0.2*H,0.6*W,0.4*H],...
      'Name','图形演示系统','NumberTitle','off','MenuBar','none');
%定义Plot菜单项
   hplot=uimenu(gcf,'Label','&Plot');
   uimenu(hplot,'Label','Sine Wave','Call',['t=-pi:pi/20:pi;','plot(t,sin(t));',...
      'set(hgon,''Enable'',''on'');','set(hgoff,''Enable'',''on'');',...
      'set(hbon,''Enable'',''on'');','set(hboff,''Enable'',''on'');']);
   uimenu(hplot,'Label','Cosine Wave','Call',['t=-pi:pi/20:pi;','plot(t,cos(t));',...
      'set(hgon,''Enable'',''on'');','set(hgoff,''Enable'',''on'');',...
      'set(hbon,''Enable'',''on'');','set(hboff,''Enable'',''on'');']);
%定义Option菜单项
   hoption=uimenu(gcf,'Label','&Option');
   hgon=uimenu(hoption,'Label','&Grig on','Call','grid on','Enable','off');
   hgoff=uimenu(hoption,'Label','&Grig off','Call','grid off','Enable','off');
   hbon=uimenu(hoption,'Label','&Box on','separator','on','Call','box
on','Enable','off');
   hboff=uimenu(hoption,'Label','&Box off','Call','box off','Enable','off');
   hfigcor=uimenu(hoption,'Label','&Figure Color','Separator','on');
   uimenu(hfigcor,'Label','&Red','Accelerator','r','Call','set(gcf,''Color'',''r'');');
   uimenu(hfigcor,'Label','&Blue','Accelerator','b','Call','set(gcf,''Color'',''b'');');
   uimenu(hfigcor,'Label','&Yellow','Call','set(gcf,''Color'',''y'');');
   uimenu(hfigcor,'Label','&White','Call','set(gcf,''Color'',''w'');');
%定义Quit菜单项
   uimenu(gcf,'Label','&Quit','Call','close(gcf)');
```

所建立的“图形演示系统”菜单如下：



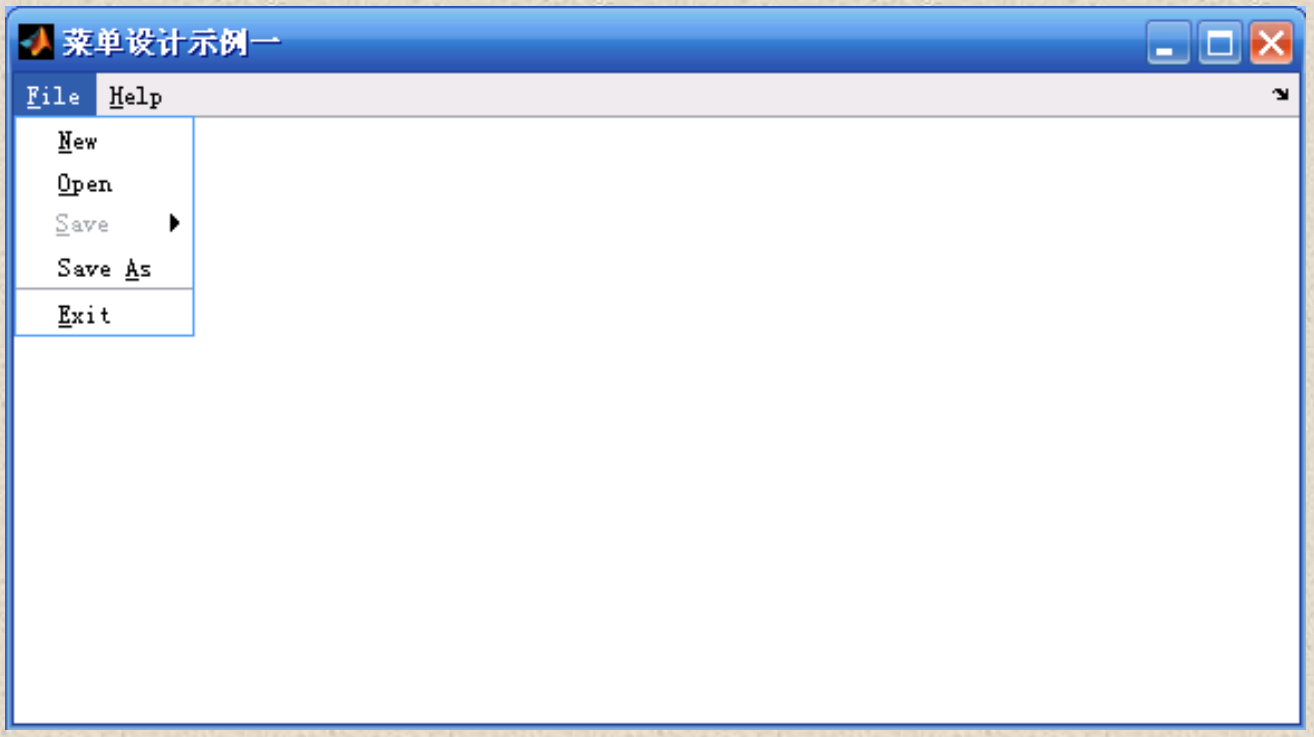
例 建立一个菜单系统。

- 菜单条中含有File和Help两个菜单项。
- 如果选择File中的New选项，则将显示New Item字样，
- 如果选择File中的Open选项，则将显示出Open Item字样。File中的Save菜单项初始时处于禁选状态，在选择Help选项之后将此菜单项恢复成可选状态，
- 如果选择File中的Save选项，则将出现一个新的菜单(三级菜单)，其中共有两个子菜单项Text file和Graphics file，如果选择第1项，则将变量k1和k2分别赋为0和1，然后调用file01.m文件来进行相应的处理(该文件需要另行编写)，如果选择第2项，则将变量k1和k2分别赋为1和0，然后调用file10.m文件来进行相应的处理(该文件也需要另行编写)。
- 如果选择File中的Save As选项，则将显示Save As Item字样。如果选择File中的Exit选项，则将关闭当前窗口。
- 如果选择Help中About …选项，则将显示Help Item字样，并将Save•菜单设置成可选状态。

程序如下：

```
screen=get(0,'ScreenSize');
      W=screen(3);H=screen(4);
      hf=figure('Color',[1,1,1],'Position',[1,1,0.4*W,0.3*H],...
        'Name','菜单设计示例一
','NumberTitle','off','MenuBar','none');
       hfile=uimenu(hf,'label','&File');
       hhelp=uimenu(hf,'label','&Help');
        uimenu(hfile,'label','&New','call','disp("New Item")');
        uimenu(hfile,'label','&Open','call','disp("Open Item")');
        hsave=uimenu(hfile,'label','&Save','Enable','off');
        uimenu(hsave,'label','Text file','call','k1=0;k2=1;file01;');
        uimenu(hsave,'label','Graphics file','call','k1=1;k2=0;file10;');
        uimenu(hfile,'label','Save &As','call','disp("Save As Item")');
        uimenu(hfile,'label','&Exit','separator','on','call','close(hf)');
        uimenu(hhelp,'label','About ...','call',...
          ['disp("Help Item");','set(hsave,"Enable","on")']);
```

所建立的菜单系统如下：



3. 快捷菜单

快捷菜单是用鼠标右键单击某对象时在屏幕上弹出的菜单。这种菜单出现的位置是不固定的，而且总是和某个图形对象相联系。

在Matlab中，可以使用uicontextmenu函数和图形对象的UIContextMenu属性来建立快捷菜单，具体步骤为:

① 利用uicontextmenu函数建立快捷菜单，格式为：

```
hc=uicontextmenu
```

功能：建立快捷菜单，并将句柄值赋给变量hc。

② 利用uimenu函数为快捷菜单建立菜单项，格式为：

```
uimenu('快捷菜单名'，属性名，属性值，...)
```

功能：为创建的快捷菜单赋值，其中属性名和属性值构成属性二元对。

③ 利用set函数将该快捷菜单和某图形对象联系起来。

例 绘制曲线 y=2e-0.5xsin(2πx)，并建立一个与之相联系的快捷菜单，用以控制曲线的线型和曲线宽度。

程序如下：

```
x=0:pi/100:2*pi;
y=2*exp(-0.5*x).*sin(2*pi*x);
hl=plot(x,y);
hc=uicontextmenu;          %建立快捷菜单
hls=uimenu(hc,'Label','线型');   %建立菜单项
hlw=uimenu(hc,'Label','线宽');
uimenu(hls,'Label','虚线','Call','set(hl,''LineStyle'','':'');');
uimenu(hls,'Label','实线','Call','set(hl,''LineStyle'',''-'');');
uimenu(hlw,'Label','加宽','Call','set(hl,''LineWidth'',2);');
uimenu(hlw,'Label','变细','Call','set(hl,''LineWidth'',0.5);');
set(hl,'UIContextMenu',hc);    %将该快捷菜单和曲线对象联系起来
```

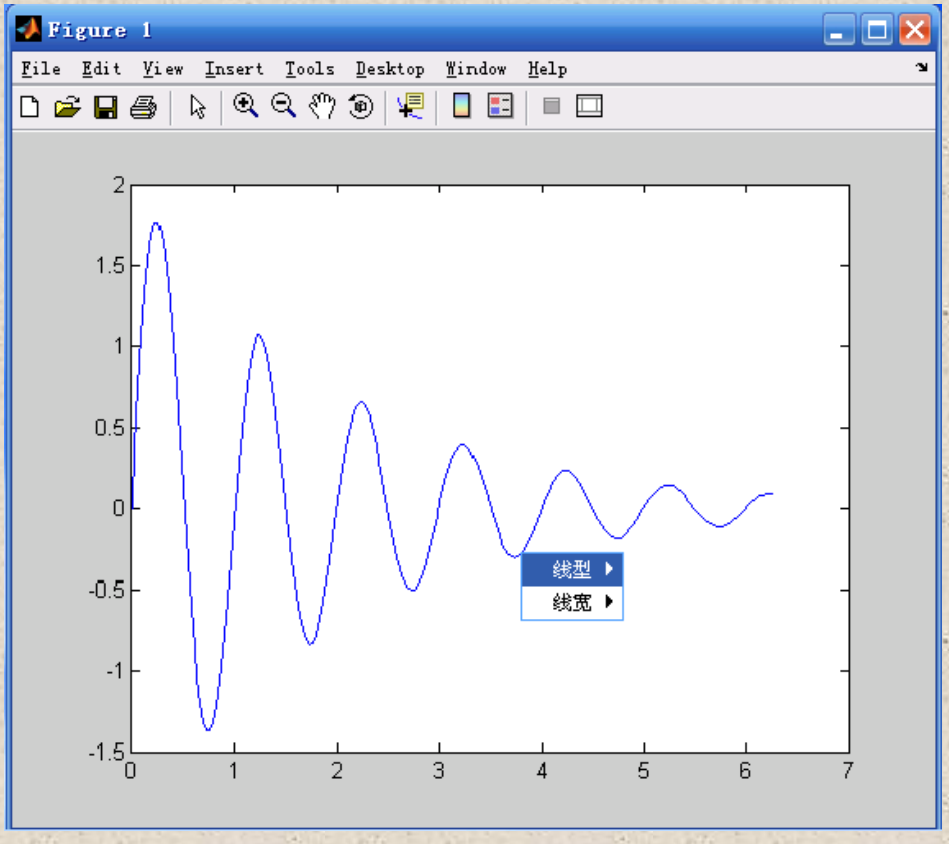


Figure 1
File Edit View Insert Tools Desktop Window Help
2
1.5
1
0.5
0
-0.5
-1
-1.5
0 1 2 3 4 5 6 7
线型
线宽

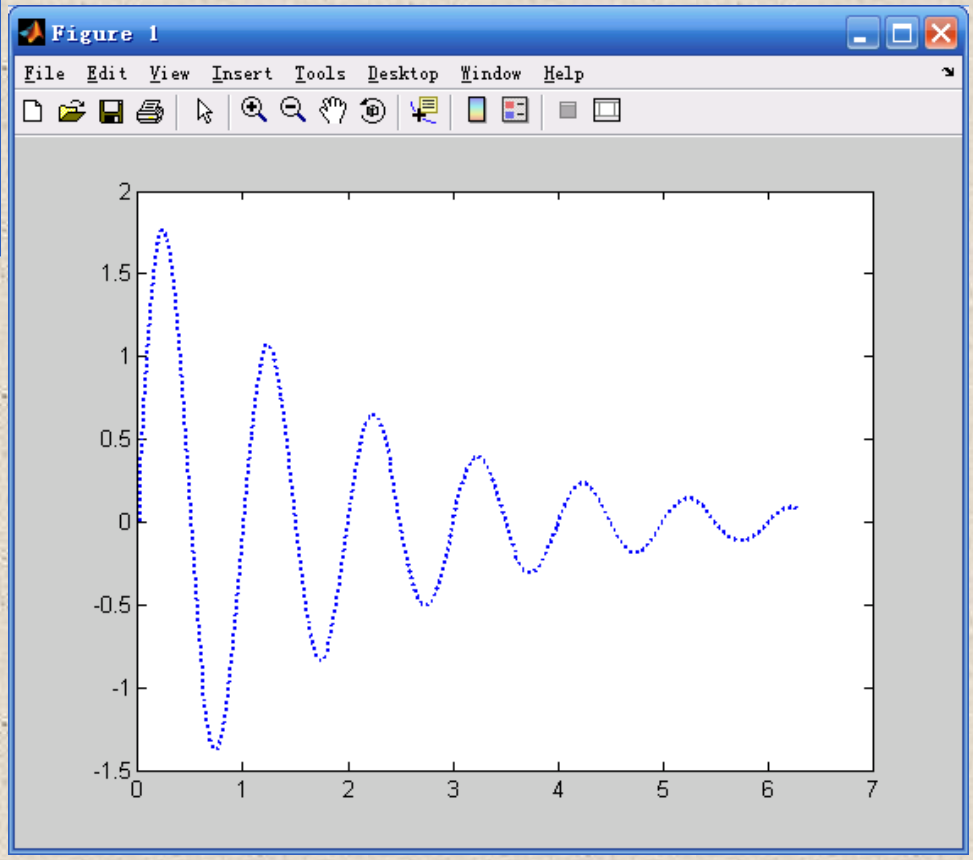


Figure 1
File Edit View Insert Tools Desktop Window Help
2
1.5
1
0.5
0
-0.5
-1
-1.5
0 1 2 3 4 5 6 7

五、程序对话框设计

在图形用户界面程序设计中，对话框是重要的信息显示和获取输入数据的用户界面对象。使用对话框，可以使应用程序的界面更加友好，使用更加方便。Matlab 提供了两类对话框，一类为 Windows 的公共对话框，另一类为 Matlab 风格的专用对话框。

1. 公共对话框

公共对话框是利用windows资源的对话框，包括文件打开、文件保存、颜色设置、字体设置、打印设置、打印预览、打印等。

① 文件打开对话框

用于打开文件，函数为 **uigetfile**，其调用格式为：

◆**uigetfile**：弹出文件打开对话框，列出当前目录下的所有Matlab文件；

◆**uigetfile('FilterSpec')**：弹出文件打开对话框，列出当前目录下的所有由‘FilterSpec’指定类型的文件；

◆**uigetfile('FilterSpec', 'DialogTitle')**：…同时设置文件打开对话框的标题为’DialogTitle’；

◆**uigetfile('FilterSpec', 'DialogTitle',x,y)**：…x,y参数用于确定文件打开对话框的位置；

◆**[fname,pname]=uigetfile(…)**：返回打开文件的文件名和路径。

② 文件保存对话框

用于保存文件，函数为uiputfile，其调用格式为：

◆**uiputfile：** 弹出文件保存对话框，列出当前目录下的所有Matlab文件；

◆**uiputfile('InitFile')：** 弹出文件保存对话框，列出当前目录下的所有由‘InitFile’指定类型的文件；

◆**uiputfile('InitFile', 'DialogTitle')：** …同时设置文件保存对话框的标题为’ DialogTitle’；

◆**uiputfile('InitFile', 'DialogTitle',x,y)：** …x,y参数用于确定文件保存对话框的位置；

◆**[fname, pname]=uiputfile(…)：** 返回保存文件的文件名和路径。

③ 颜色设置对话框

用于图形对象颜色的交互式设置，函数为 uisetcolor，其调用格式为：

c=uisetcolor(‘h_or_c，’DialogTitle’)

输入参数 h_or_c 可以是一个图形对象的句柄，也可以是一个三色 RGB 矢量，’ DialogTitle’为颜色设置对话框的标题。

④ 字体设置对话框

用于字体属性的交互式设置，函数为uisetfont，格式为：

◆**uisetfont：** 打开字体设置对话框，返回所选择字体的属性；

◆**uisetfont(h)：** h 为图形对象句柄，使用字体设置对话框重新设置该对象的字体属性；

◆**uisetfont(S)：** S为字体属性结构变量，S中包含的属性有FontName、FontUnits、FontSize、FontWeight、FontAngle，返回重新设置的属性值；

◆**uisetfont(h, 'DialogTitle')：** h 为图形对象句柄，使用字体设置对话框重新设置该对象的字体属性，'DialogTitle'设置对话框的标题；

◆ **uisetfont(S, 'DialogTitle')：** S为字体属性结构变量，S中包含的属性有FontName、FontUnits、FontSize、FontWeight、FontAngle，返回重新设置的属性值，'DialogTitle' 设置对话框的标题；

◆**S=uisetfont(…)：** 返回字体属性值，保存在结构变量S中。

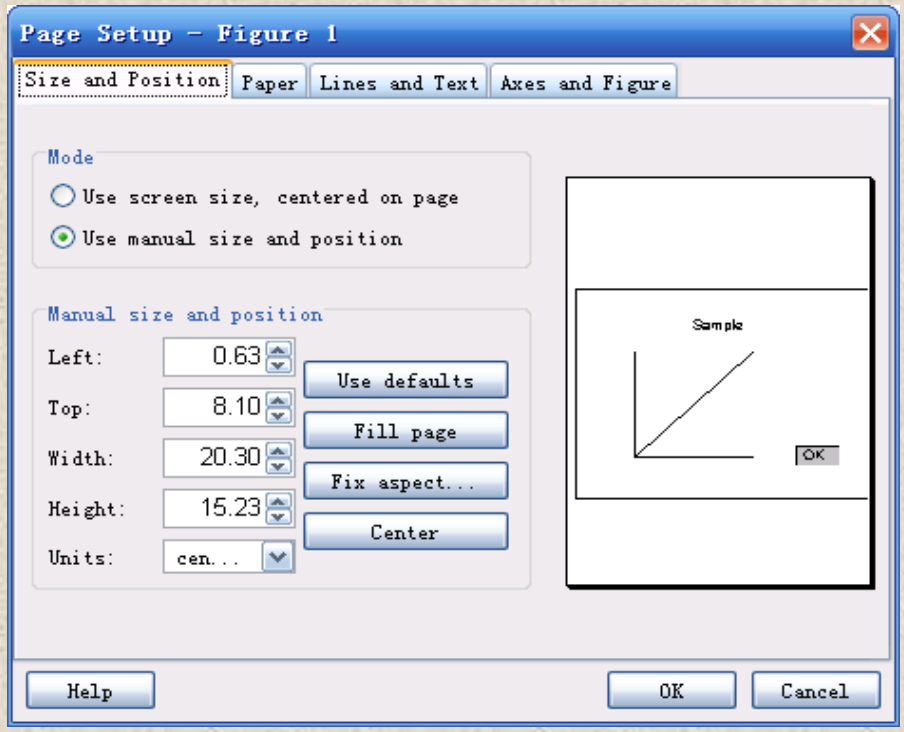
⑤ 打印设置对话框

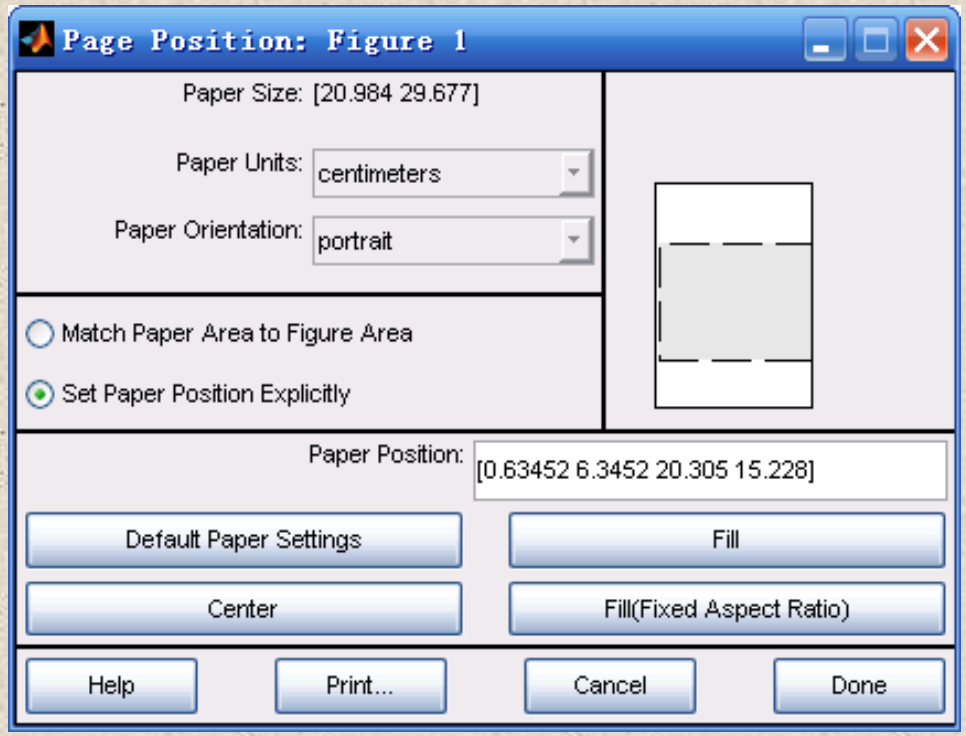
用于打印页面的交互式设置，有两个函数：**pagesetupdlg** 和 **pagedlg**（老版本，**Matlab6**中仍可用）。调用格式为:

◆**dlg=pagesetupdlg(fig)**：**fig**为图形窗口的句柄，省略时为当前图形窗口；

◆**pagedlg**：设置当前图形窗口；

◆**pagedlg(fig)**：设置以**fig**为句柄的图形窗口。



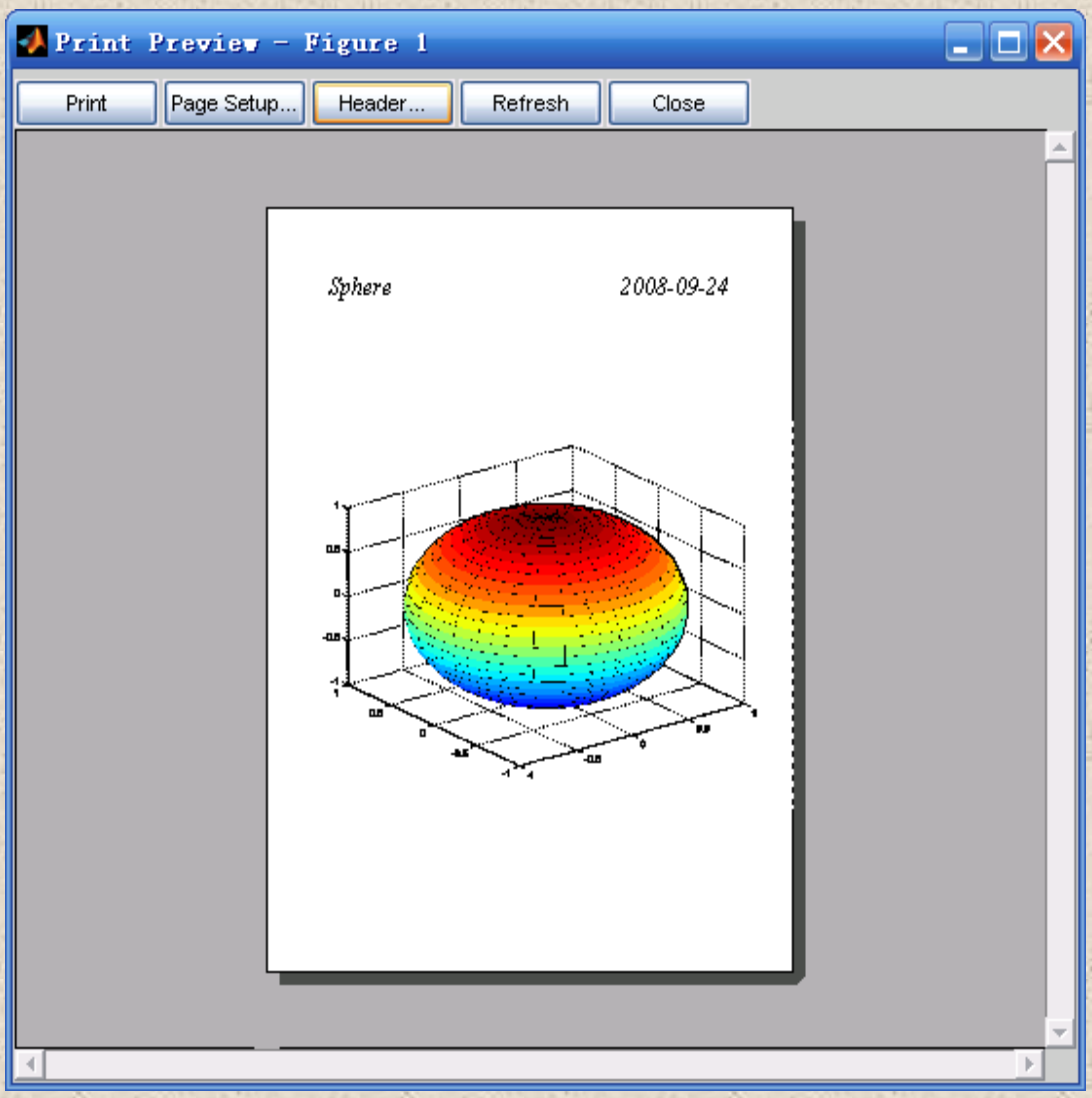


⑥ 打印预览对话框

用于对打印页面进行预览，函数为printpreview，格式为：

◆**printpreview：** 对当前图形窗口进行打印预览；

◆**printpreview(f)：** 对以f为句柄的图形窗口进行打印预览。



如右图所示，打印预览对话框上有5个按钮。

⑦ 打印对话框

为Windows的标准对话框，函数为 printdlg ，格式为：

◆**printdlg**：对当前图形窗口打开Windows打印对话框；

◆**printdlg(fig)**：对以fig为句柄的图形窗口打开Windows打印对话框；

◆**printdlg('-crossplatform', fig)**：打开crossplatform 模式的Matlab打印对话框；

◆**printdlg(-'setup', fig)**：在打印设置模式下，强制打开打印对话框。

2. Matlab专用对话框

Matlab除了使用公共对话框外，还提供了一些专用对话框，包括帮助、错误信息、信息提示、警告信息等。

① 错误信息对话框

用于提示错误信息，函数为errordlg，其调用格式为：

◆**errordlg：** 打开默认的错误信息对话框；

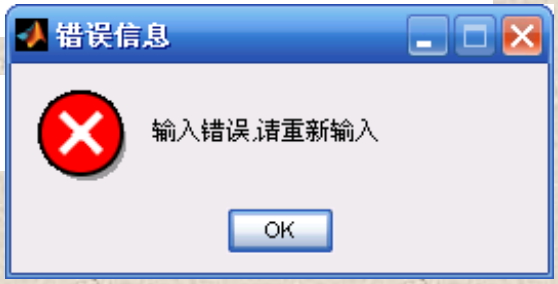
◆**errordlg(‘errorstring’)：** 打开显示‘errorstring’信息的错误信息对话框；

◆**errordlg(‘errorstring’, ’dlgname’)：** 打开显示‘errorstring’信息的错误信息对话框，对话框的标题由 ‘dlgname’指定；

◆**erordlg(‘errorstring’, ’dlgname’, ’on’)：** 打开显示‘errorstring’信息的错误信息对话框，对话框的标题由 ‘dlgname’指定。如果对话框已存在，‘on’参数将对话框显示在最前端；

◆**h=errodlg(…)：** 返回对话框句柄。

例 errordlg('输入错误,请重新输入','错误信息')



② 帮助对话框

用于帮助提示信息，函数为helpdlg，其调用格式为：

◆**helpdlg**：打开默认的帮助对话框；

◆**helpdlg('helpstring')**：打开显示‘errorstring’信息的帮助对话框；

◆**helpdlg('helpstring', 'dlgname')**：打开显示‘errorstring’信息的帮助对话框, 对话框的标题由‘dlgname’指定；

◆**h=helpdlg(…)**：返回对话框句柄。

例 helpdlg('矩阵尺寸必须相等','在线帮助')

③ 输入对话框

用于输入信息，函数为inputdlg，其调用格式为：

◆answer=inputdlg(prompt)：打开输入对话框，prompt为单元数组，用于定义输入数据窗口的个数和显示提示信息，answer为用于存储输入数据的单元数组；

◆answer=inputdlg(prompt, title)：与上者相同，title确定对话框的标题；

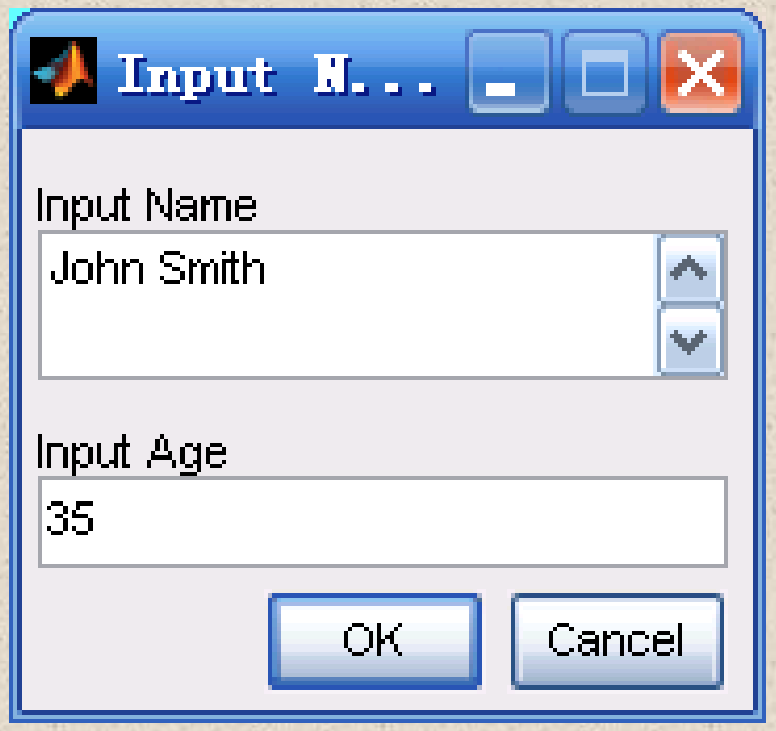
◆answer=inputdlg(prompt, title, lineNo)：参数lineNo可以是标量、列矢量或m×2阶矩阵，若为标量，表示每个输入窗口的行数均为lineNo；若为列矢量，则每个输入窗口的行数由列矢量lineNo的每个元素确定；若为矩阵，每个元素对应一个输入窗口，每行的第一列为输入窗口的行数，第二列为输入窗口的宽度；

◆answer=inputdlg(prompt, title, lineNo, defAns)：参数defans为一个单元数组，存储每个输入数据的默认值，元素个数必须与prompt 所定义的输入窗口数相同，所有元素必须是字符串；

◆answer=inputdlg(prompt, title, lineNo, defAns, Resize)：参数resize决定输入对话框的大小能否被调整，可选值为on或off.

例 创建两个输入窗口的输入对话框。

```
prompt={'Input Name', 'Input Age'};
title='Input Name and Age';
lines=[2 1]';
def={'John Smith', '35'};
answer=inputdlg(prompt, title, lines, def);
```



④ 列表选择对话框

用于在多个选项中选择需要的值，函数为listdlg，其调用格式为：

[selection, ok]=listdlg('Liststring',S,…)

输出参数selection为一个矢量，存储所选择的列表项的索引号，输入参数为可选项’Liststring’（字符单元数组），’SelectionMode’（’single’或’multiple（缺省值）’），’ListSize’([wight, height]), ’Name’ (对话框标题）等。

⑤ 信息提示对话框

用于显示提示信息，函数为msgbox，其调用格式为：

◆**msgbox(message)**：打开信息提示对话框，显示message信息；

◆**msgbox(message, title)**：…title确定对话框标题；

◆**msgbox(message, title, 'icon')**：…'icon'用于显示图标，可选图标包括：none(无图标，缺省值)、error、help、warn或custom（用户定义）；

◆**msgbox(message, title, 'custom', icondata, iconcmap)**：当使用用户定义图标时，iconData为定义图标的图像数据，iconCmap为图像的色彩图；

◆**msgbox(…, 'creatmode')**：选择模式creatMode，选项为：modal, non-modal 和replace；

◆**h=msgbox(…)**：返回对话框句柄。

⑥ 问题提示对话框

用于回答问题的多种选择，函数为questdlg，格式为：

◆button=questdlg(‘qstring’)：打开问题提示对话框，有三个按钮，分别为：Yes，No和Cancel，’ questdlg’确定提示信息;

◆button=questdlg(‘qstring’, ’title’)： … title确定对话框标题;

◆button=questdlg(‘qstring’, ’title’, ’default’)：当按回车键时，返回‘default’的值，‘default’必须是Yes，No或Cancel 之一;

◆button=questdlg(‘qstring’, ’title’, ’str1’, ’str2’, ’default’)：打开问题提示对话框，有两个按钮，分别由str1和str2确定，’ qstdlg’确定提示信息，‘title’确定对话框标题，‘default’必须是str1或str2之一;

◆button=questdlg(‘qstring’, ’title’, ’str1’, ’str2’, ’str3’, ’default’)：打开问题提示对话框，有三个按钮，分别由str1， str2和str3确定，’ qstdlg’确定提示信息，‘title’确定对话框标题，‘default’必须是str1, str2或str3之一。

⑦ 进程条

以图形方式显示运算或处理的进程，函数为waitbar，其调用格式为：

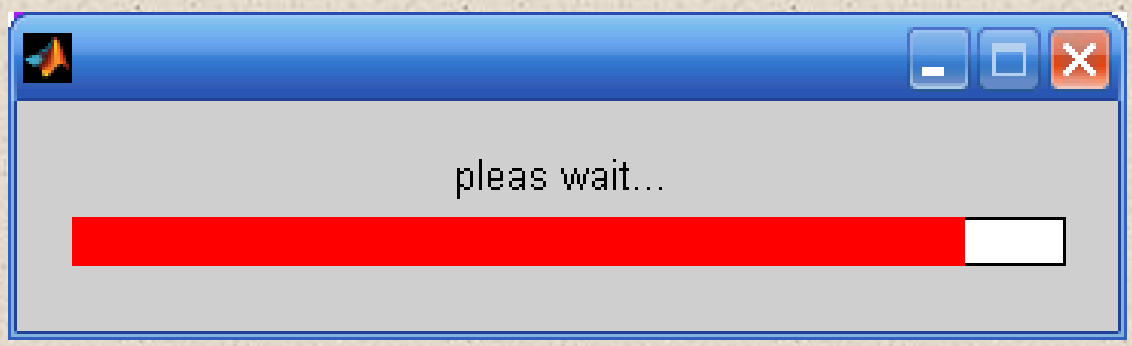
◆**h=waitbar(x, 'title')：** 显示以title为标题的进程条，x为进程条的比例长度，其值必须在0到1之间，h为返回的进程条对象的句柄；

◆**waitbar(x, 'title', 'creatcancelbtn', 'button_callback')：** 在进程条上使用CreatCancelBtn参数创建一个撤销按钮，在进程中按下撤销按钮将调用button_callback函数；

◆**waitbar(…, property_name, property_value, …)：** 选择其它由property_name定义的参数，参数值由property_value指定。

例　创建并使用进程条。

```
h=waitbar(0, 'pleas wait...');
for i=1:10000
  waitbar(i/10000,h)
end
  close(h)
```



⑧ 警告信息对话框

用于提示警告信息，函数为warndlg，其调用格式为：

h=warndlg(‘warningstring’, ’dlgname’)

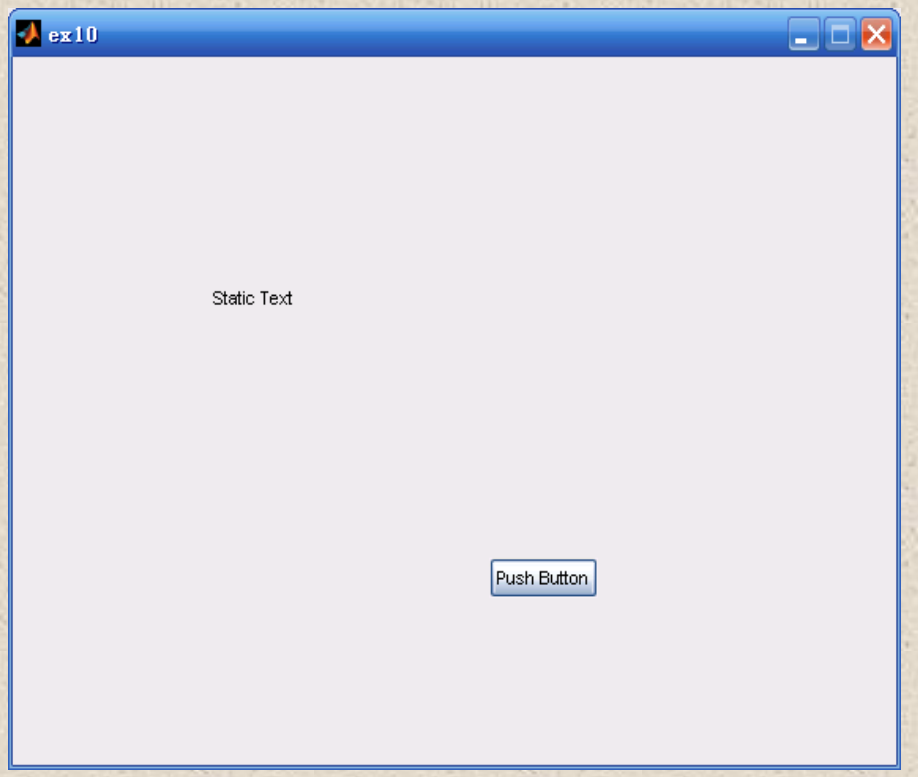
打开警告信息对话框，显示‘warningstring’信息，‘dlgname’确定对话框标题，h为返回的对话框句柄。

六、GUI程序设计实例

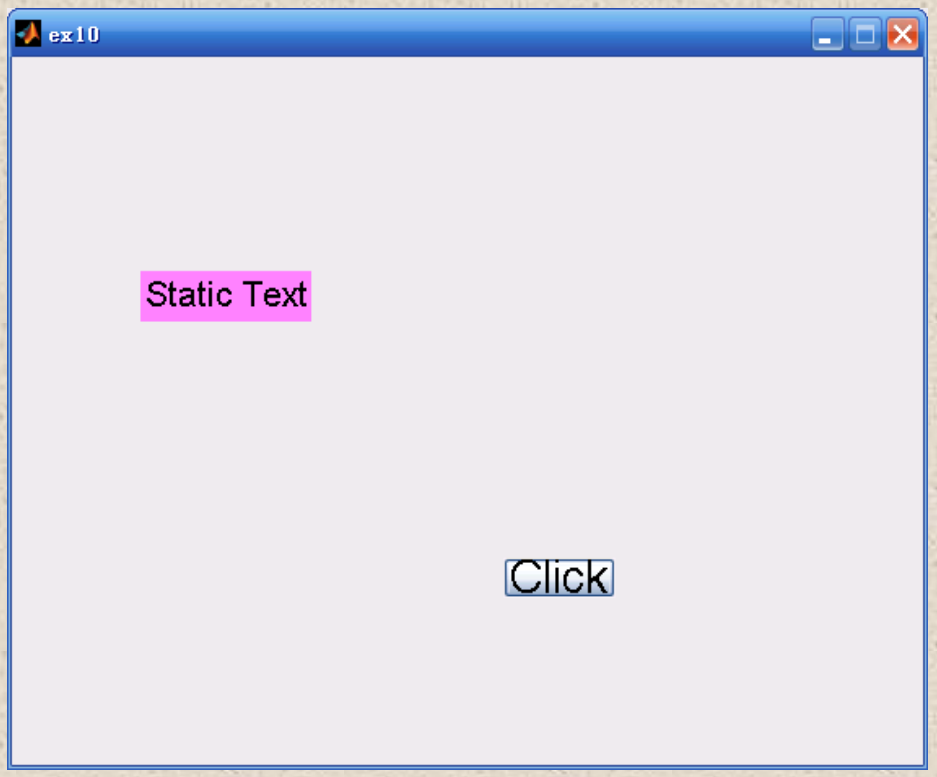
包括图形用户界面的设计和功能设计两个方面。

例　使用Push Button按钮与静态文本框设计GUI，在窗口中显示单击按钮次数。

(1)　在界面上安装一个命令按钮和一个静态文本框



(2)　使用对象的属性窗口设置控件的属性



(3) 打开该GUI的 m 文件，该文件中已经自动生成了许多代码。找到函数

function pushbutton1_Callback(hObject, eventdata, handles)

在这个函数名称下面写入如下程序段：

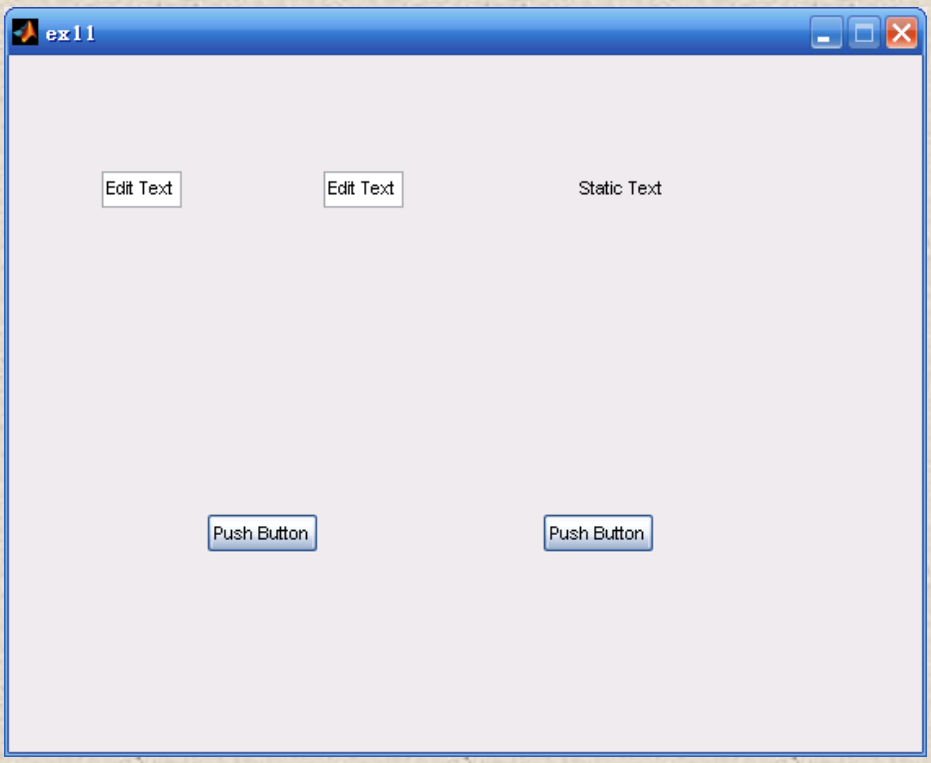
```
persistent c
if isempty(c)
    c=0
end
c=c+1;
str=sprintf('Total Clicks:  %d', c);
set(handles.text1, 'String', str);
```

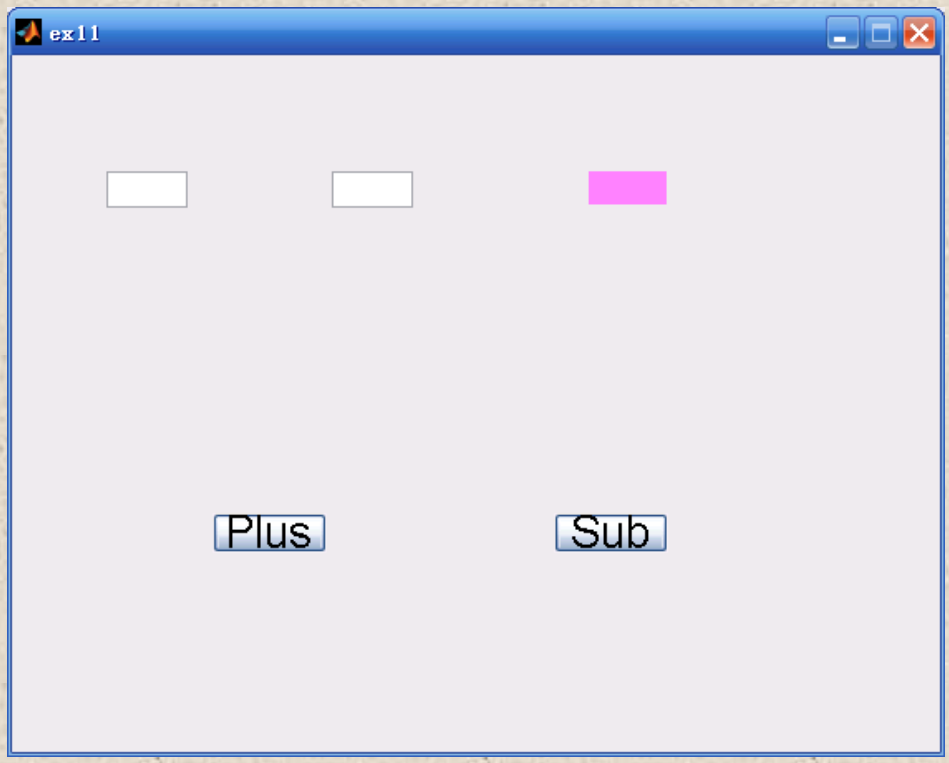
保存程序后，单击Click按钮，则在什么的文本框中显示单击次数。

例　制作一个简易的加减法计算器。

(1)　在界面上安装两个编辑文本框、一个静态文本框与两个命令按钮

(2)　使用对象的属性窗口设置控件的属性



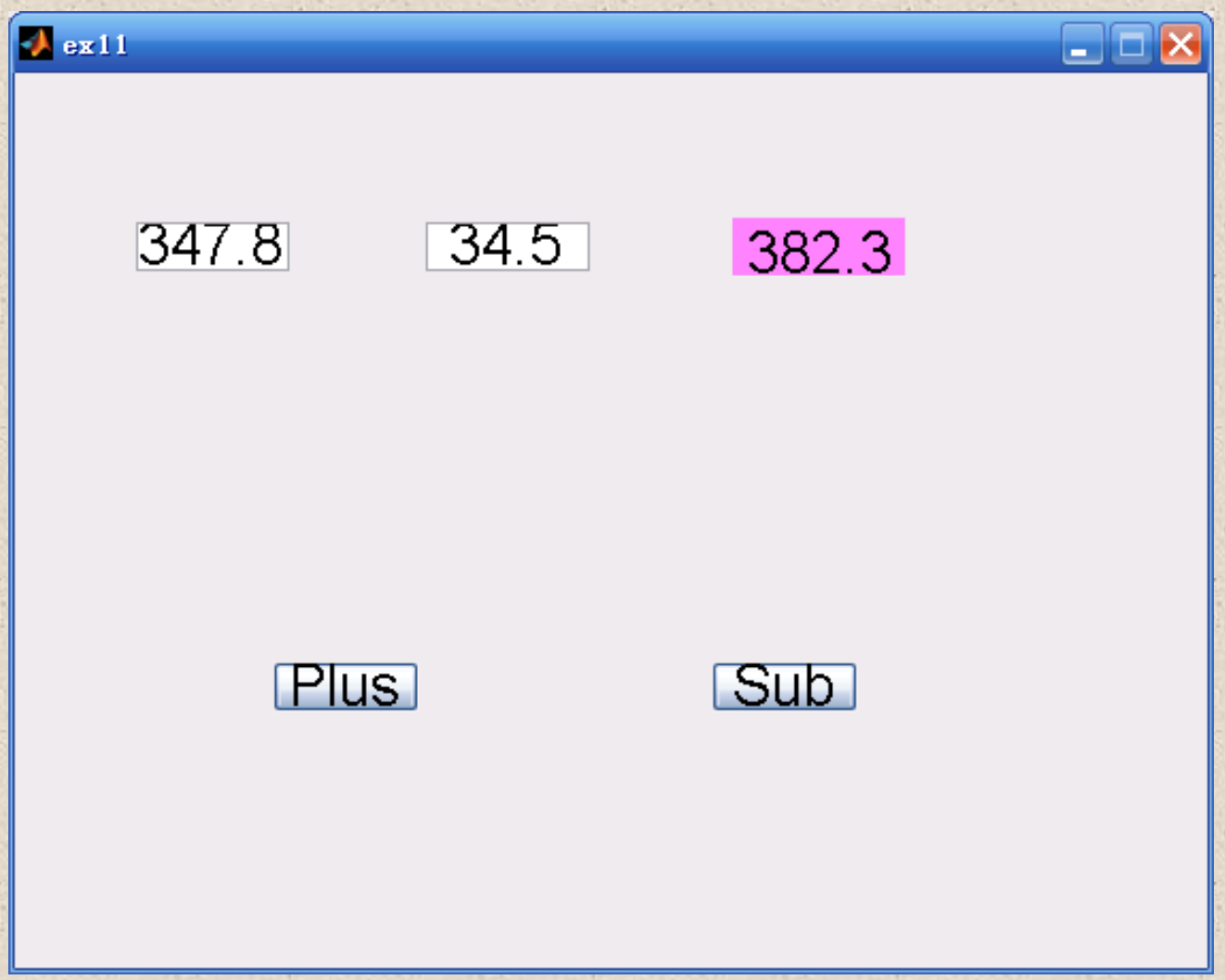


(3) 打开该GUI的 m 文件，在函数pushbutton1_Callback 与 pushbutton2_Callback

中加入代码，如下所示：

```
function pushbutton1_Callback(hObject, eventdata, handles)
s1=str2double(get(handles.edit1, 'String'))
s2=str2double(get(handles.edit2, 'String'))
set(handles.text1, 'String', s1+s2);
```

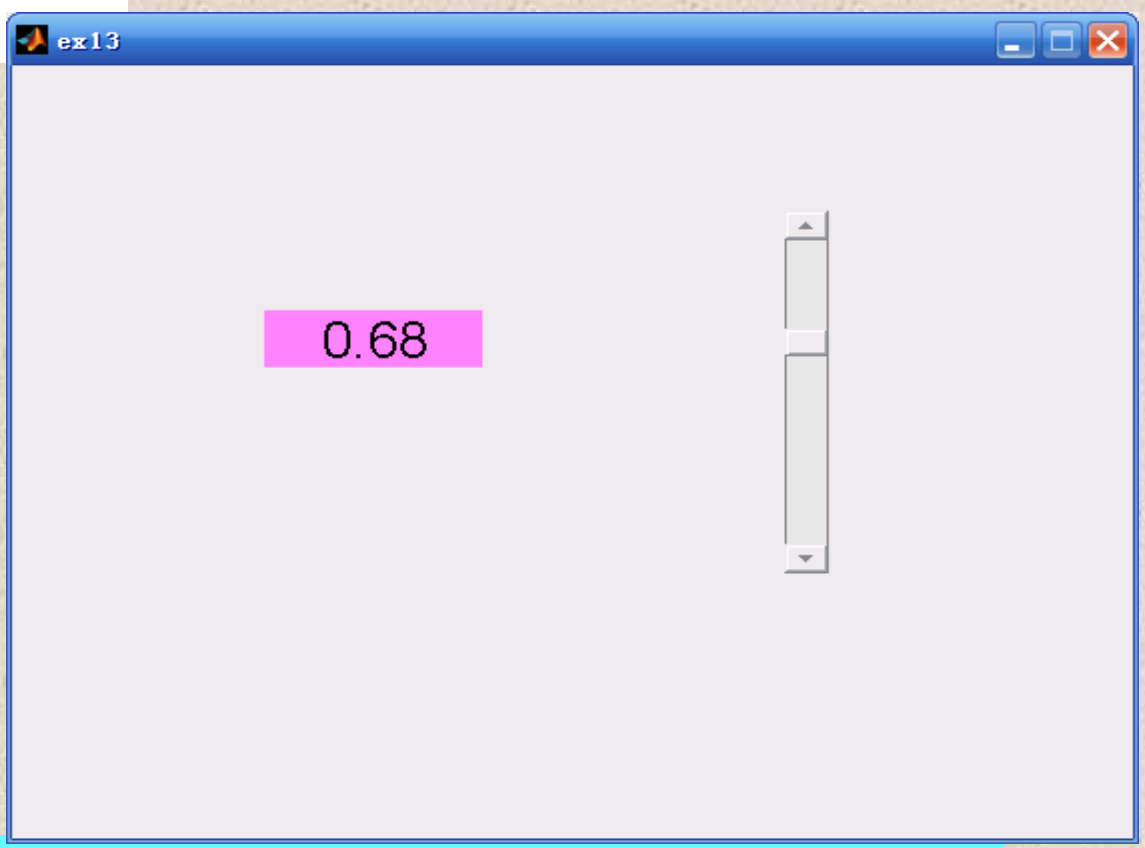
```
function pushbutton2_Callback(hObject, eventdata, handles)
s1=str2double(get(handles.edit1, 'String'))
s2=str2double(get(handles.edit2, 'String'))
set(handles.text1, ‘String’, s1-s2);
```


ex11
347.8
34.5
382.3
Plus
Sub

例　使用3个单选钮控制静态文本框的背景颜色。


```
function radiobutton1_Callback(hObject, eventdata, handles)
set(handles.text1, 'BackGroundColor', 'r')
function radiobutton2_Callback(hObject, eventdata, handles)
set(handles.text1, 'BackGroundColor', 'g')
function radiobutton3_Callback(hObject, eventdata, handles)
set(handles.text1, 'BackGroundColor', 'b')
```

例　使用滚动条。

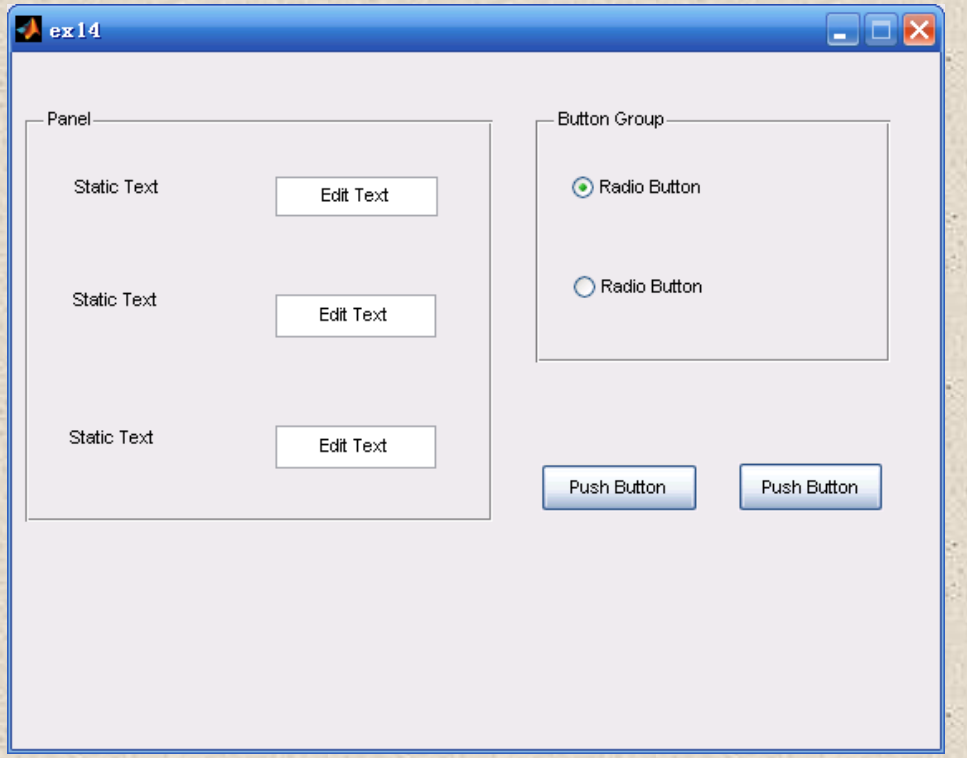


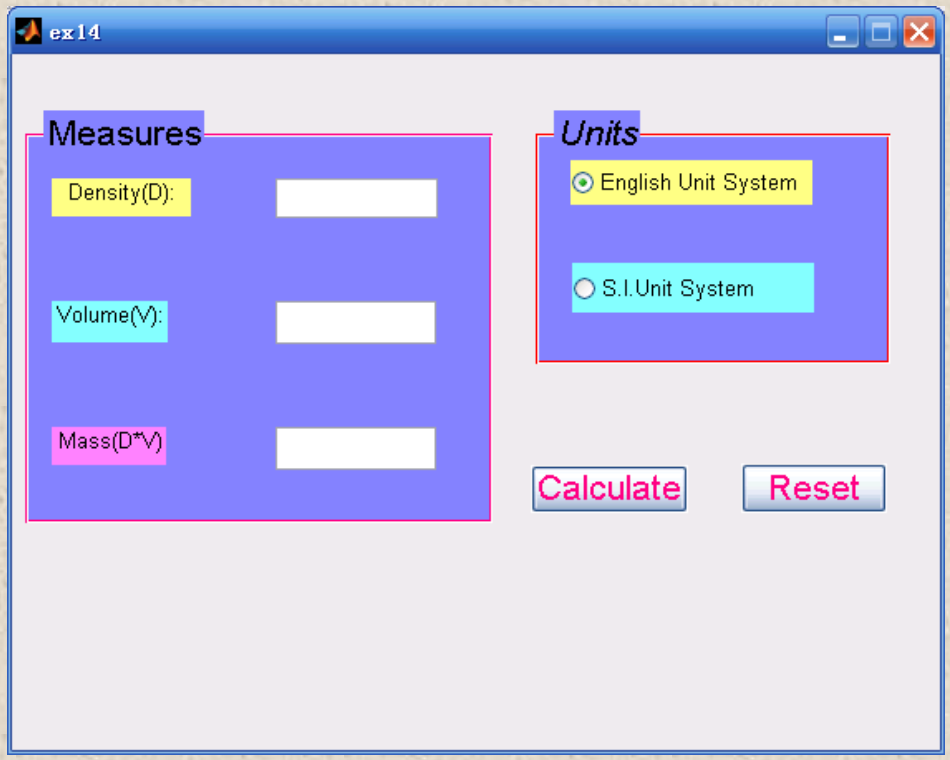

```
function slider1_Callback(hObject, eventdata, handles)
v=get(handles.slider1, 'Value');
str=sprintf('%.2f', v);
set(handles.text1, 'String', str);
```

例　制作一个根据密度与体积计算质量的简易计算器。

(1)　在界面上布置控件对象。

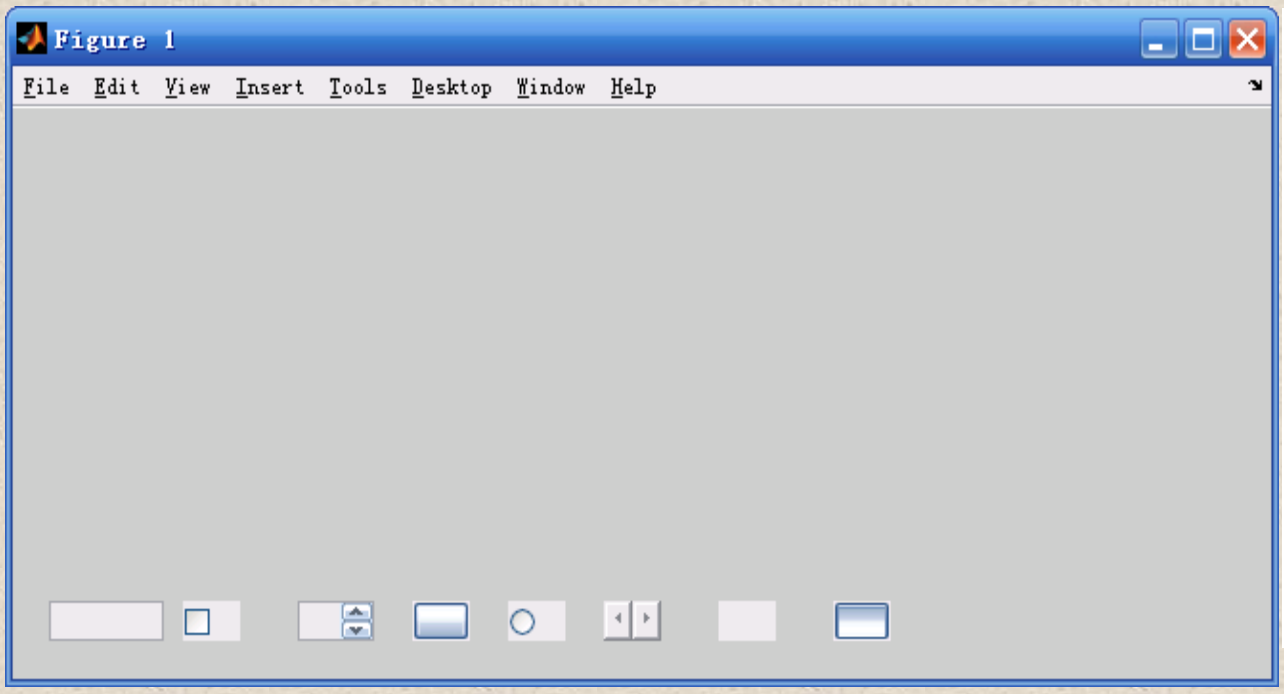
(2)　使用对象的属性窗口设置控件的属性。





例　使用程序把基本控件安装在图形窗口上。

```
h1=uicontrol('style','edit','TooltipString','Edit Text')
h2=uicontrol('style','checkbox','Position',[90,20,30,20],'TooltipString','Checkbox')
h3=uicontrol('style','listbox','Position',[150,20,40,20],'TooltipString','Listbox')
h4=uicontrol('style','pushbutton','Position',[210,20,30,20],'TooltipString','Pushbutton')
h5=uicontrol('style','radiobutton','Position',[260,20,30,20],'TooltipString','Radiobutton')
h6=uicontrol('style','slider','Position',[310,20,30,20],'TooltipString','Slider')
h7=uicontrol('style','text','Position',[370,20,30,20],'TooltipString','Static Text')
h8=uicontrol('style','toggle','Position',[430,20,30,20],'TooltipString','Togglebutton')
```



上面程序虽然实现了控件的安装，但是，还不能完成具体的功能。如果要完成特定的功能，需要加入其他语句。

例　在图形窗口底部安装一个命令按钮、一个可编辑文本框、一个静态文本框。针对命令按钮（pushbutton）编写程序，使程序运行后，点击该命令按钮，便随机绘制出一些折线；同时可编辑文本框背景色变为蓝色，静态文本框背景色变为红色。

主程序设计如下：

```
h1=uicontrol('style','pushbutton','Position',[280,0,50,20])
set(h1,'String','PUSH','Callback','fun')
```

主程序只创建安装了一个Pushbutton按钮，然后，在set语句中使用Callback属性调用函数fun。

函数fun代码如下：

```
function fun
  plot(rand(3,5))
  h2=uicontrol('style','edit','TooltipString',
          'Edit Text','Position',[80,0,30,20])
  h3=uicontrol('style','text','Position',
          [440,0,30,20],'TooltipString','Static Text')
  set(h2,'BackGroundColor',[0  0  1])
  set(h3,'BackGroundColor',[1  0  0])
```

在函数fun中除了绘图之外，还制作了一个Edit Text、一个Static Text，并且把这两个控件的背景色设置为蓝色[0 0 1]与红色[1 0 0]。

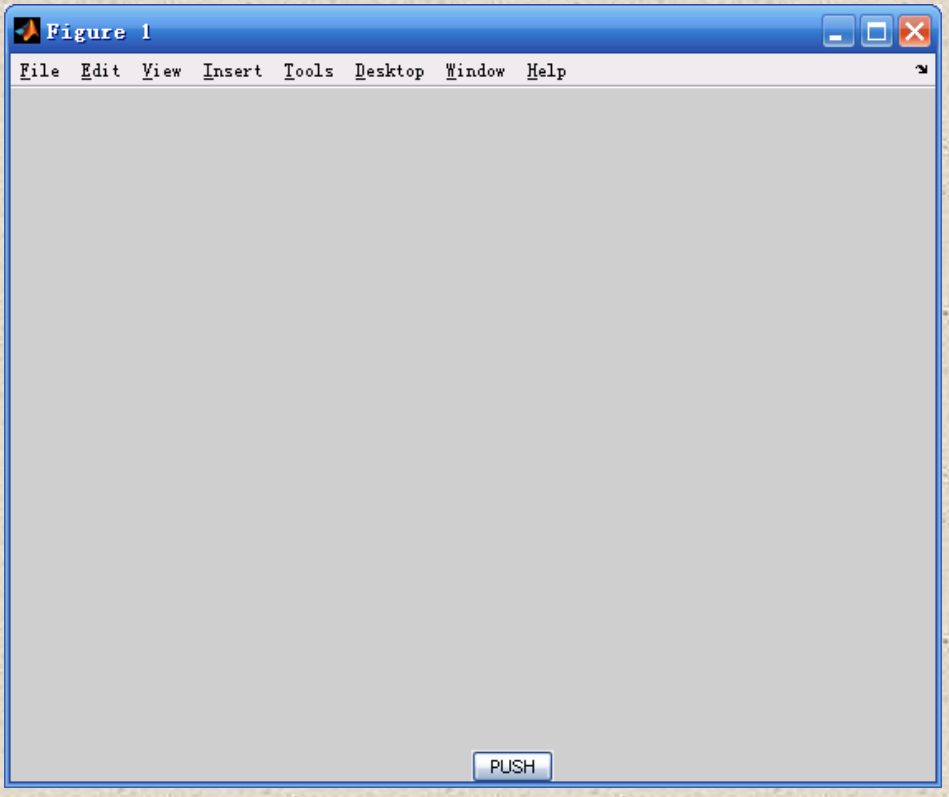


Figure 1
File Edit View Insert Tools Desktop Window Help
PUSH

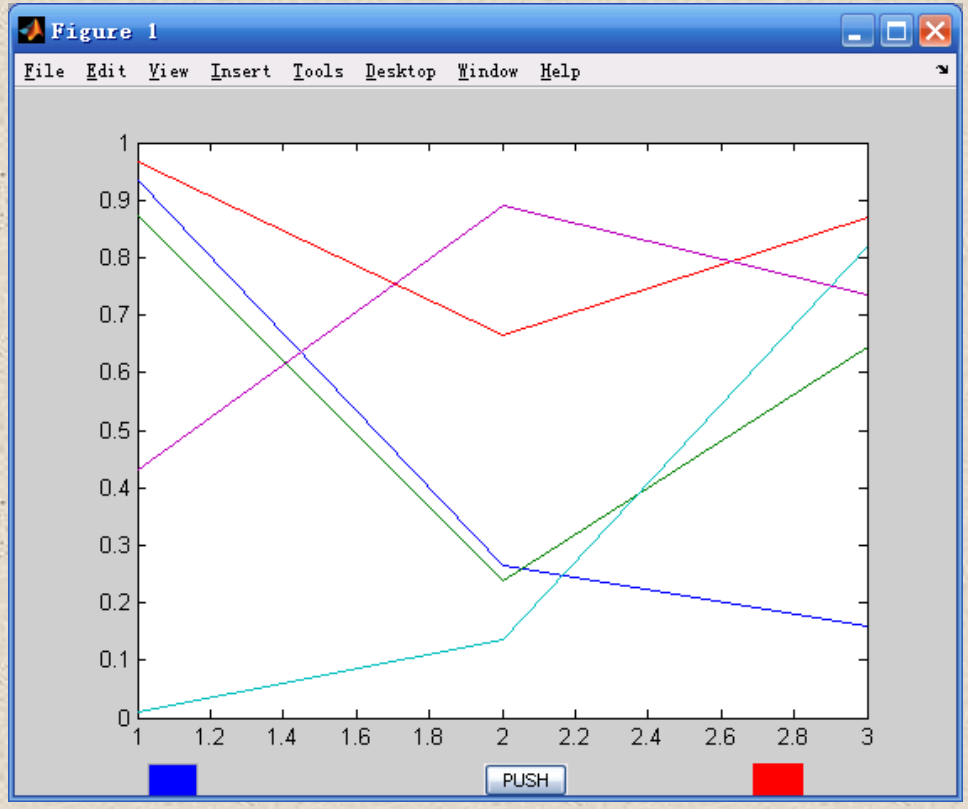


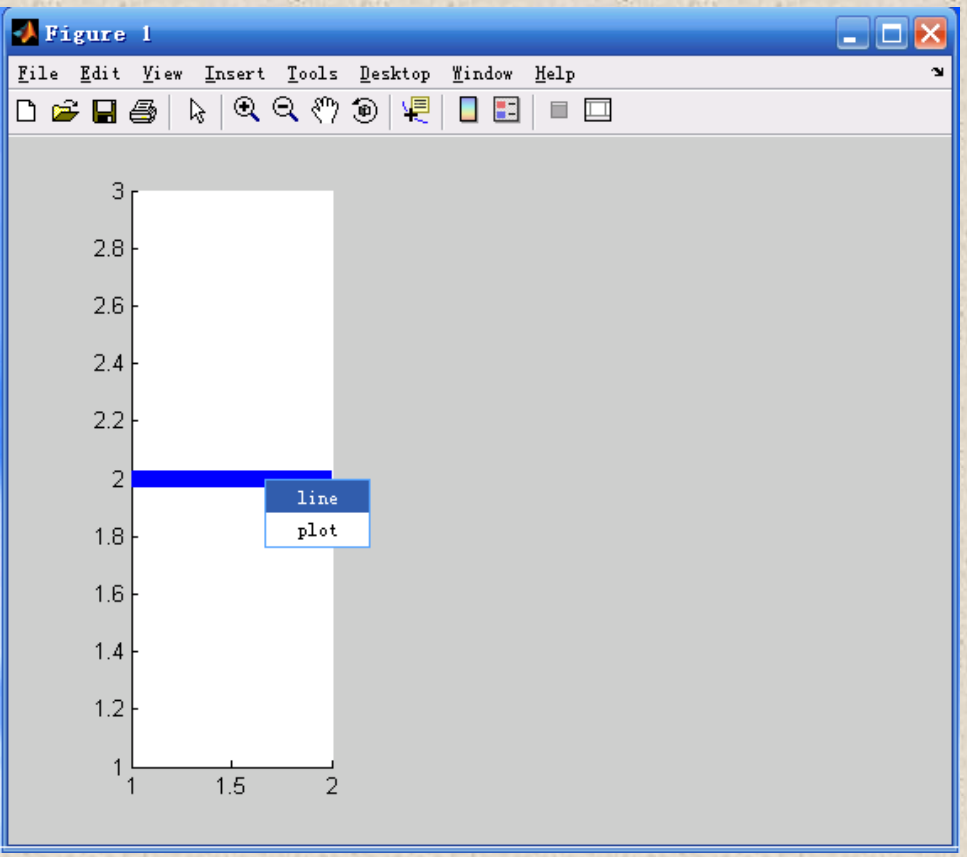
Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
PUSH

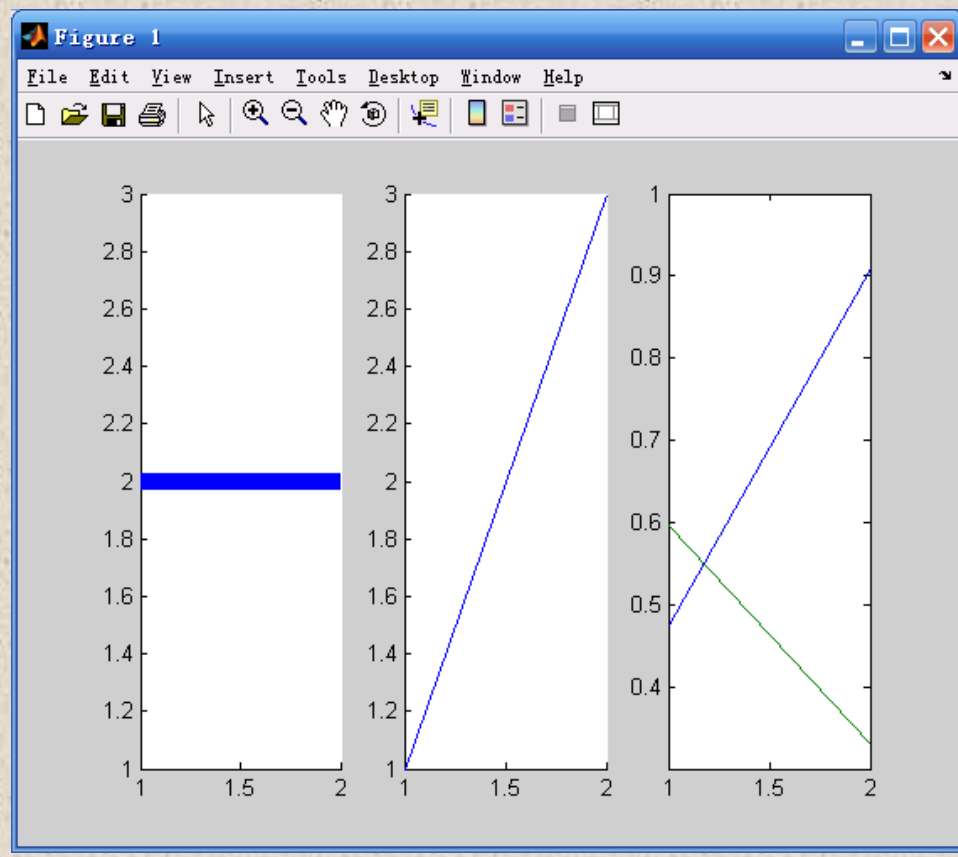
例 制作依附于某对象的弹出式菜单。

编写程序如下：

```
m=uicontextmenu;
subplot(1,3,1)
h1=line([1,2],[2,2],'LineWidth',8,'UIContextMenu',m)
c1=['subplot(1,3,2);line([1 2],[1 3])'];
c2=['subplot(1,3,3);plot(rand(2))'];
uimenu(m,'Label','line','Callback',c1);
uimenu(m,'Label','plot','Callback',c2);
```

程序运行后，先绘制出右图第一个图所示图形，在蓝色宽条上单击鼠标右键，出现菜单，菜单上有两个选项line与plot，选择line绘制出右图第二个图所示线段；选择plot绘制出右图第三个图所示两条（随机）线段。





例　用于绘图和图形旋转的GUI。

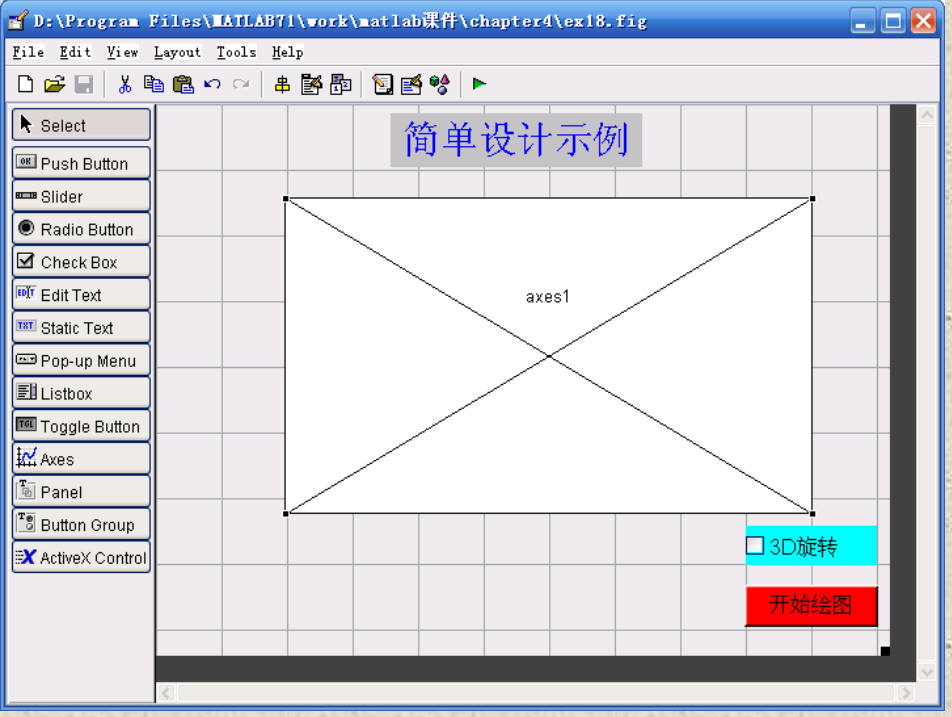
① 布置控件：一个坐标系、一个文本框、一个复选框，一个按钮；

② 定义文本框的属性：String---简单设计示例，FontName---隶书，FontSize—22；

③ 定义坐标系：Visible—off；

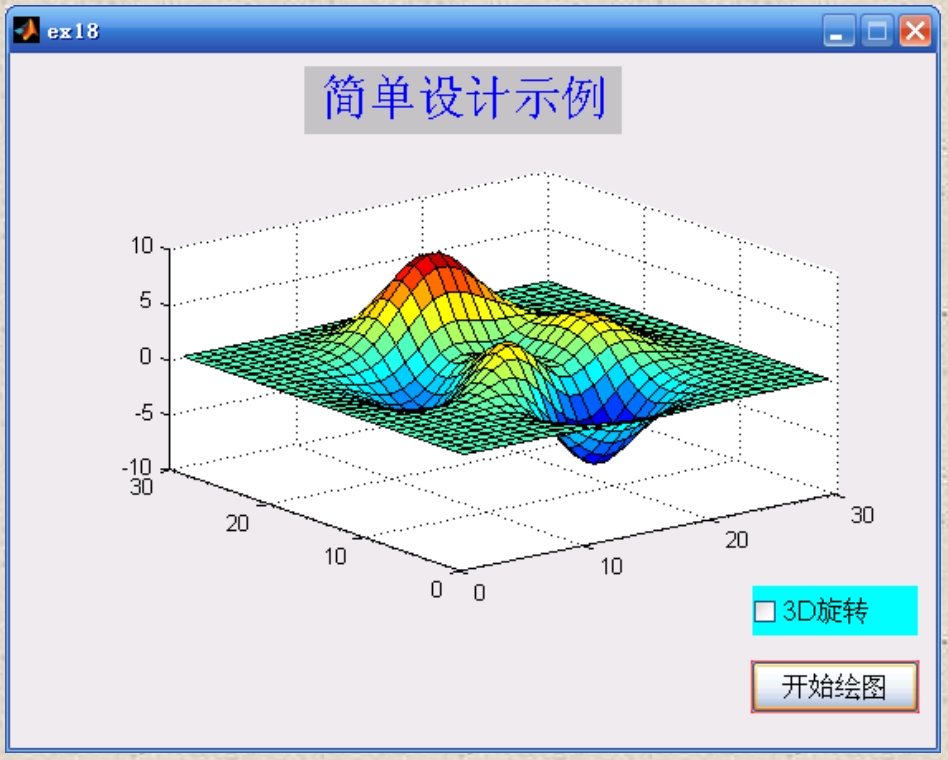
④ 定义按钮属性：String—开始绘图，FontName, ForegroundColor, FontSize, BackgroundColor,Callback---surf(peaks(30))；

⑤ 定义复选框：String—3D旋转,Callback—rotate3d。



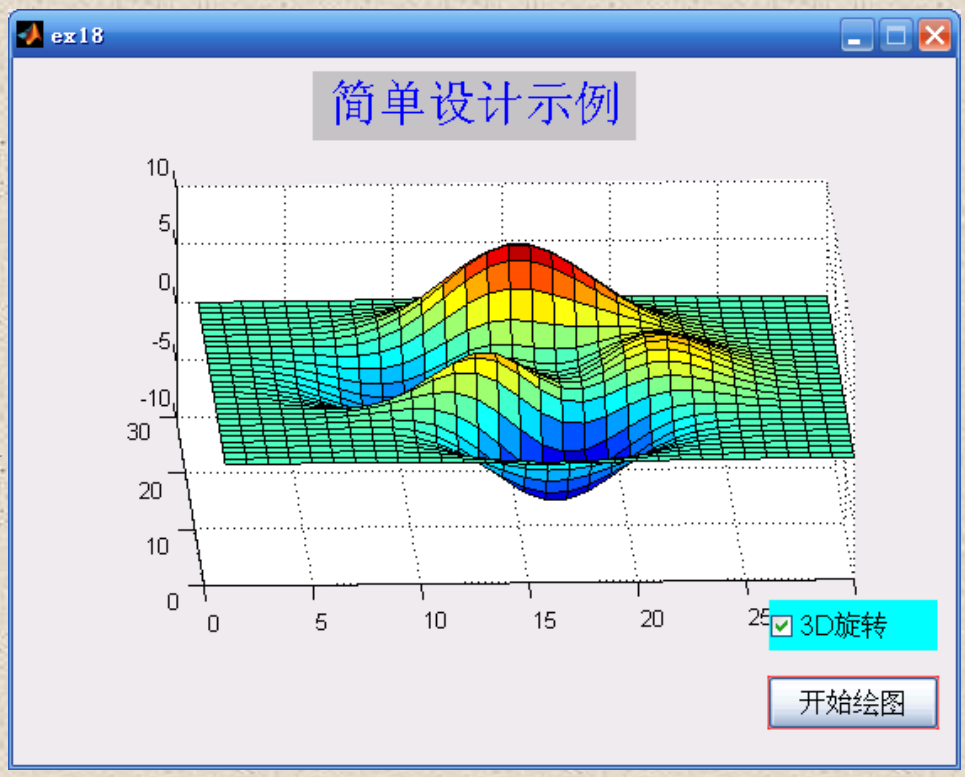
布局编辑器中编辑完成的图形用户界面

运行所创建的图形用户界面程序



点击按钮的结果

选择三维旋转功能后对图形进行的旋转操作



例　设计一个简单信号分析仪的程序，要求根据输入的两个频率和时间间隔，计算函数 $x=\sin(2\pi f_1 t)+\sin(2\pi f_2 t)$ 的值，并对函数进行快速傅立叶变换，最后分别绘制时域和频域的曲线。

（一）设计图形界面

设计步骤：

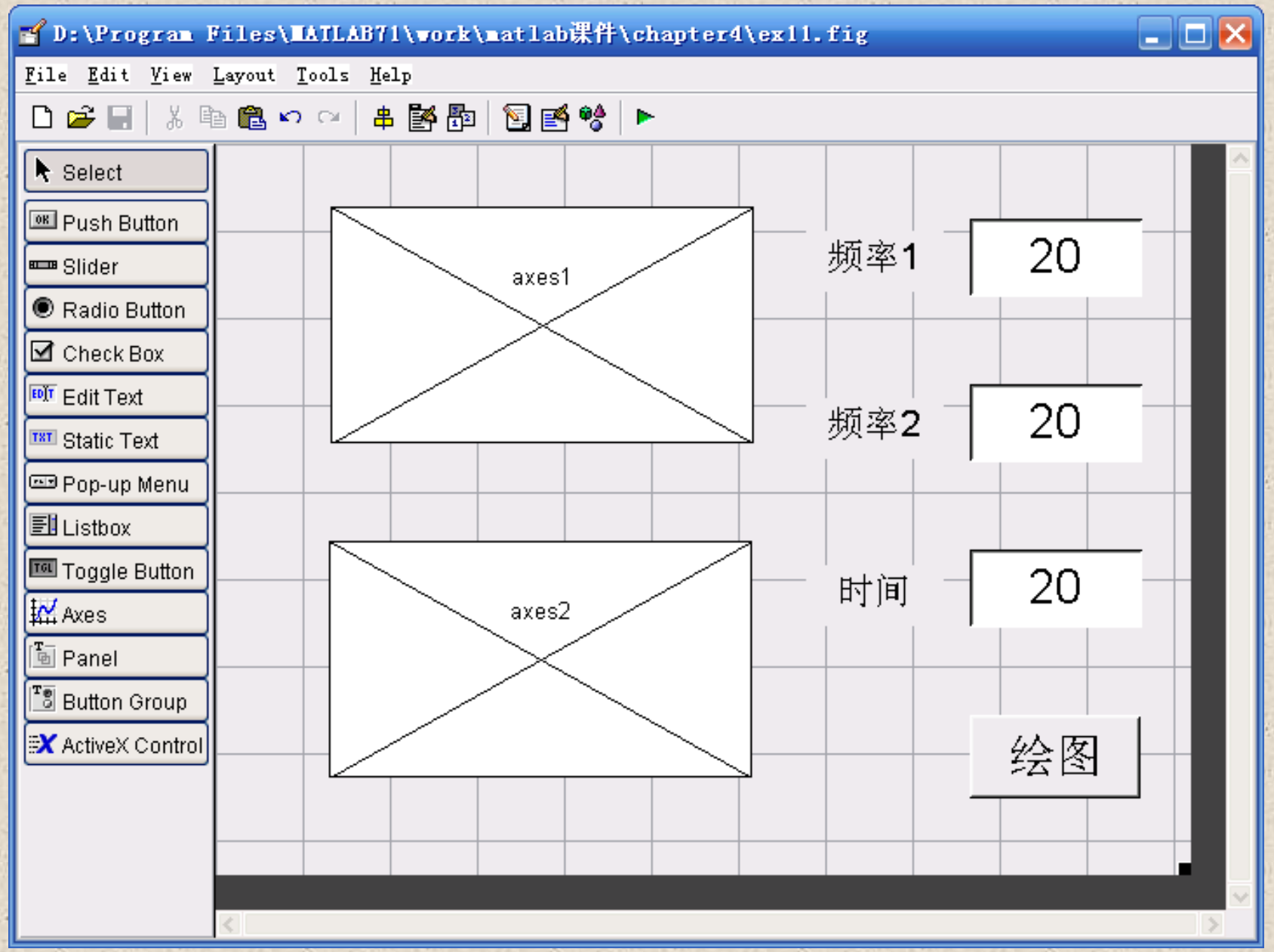
◆在布局编辑器中布置控件：本例中使用了2个坐标系、3个文本编辑框、1个按钮和3个静态文本框；

◆使用几何位置排列工具对控件的位置进行调整：

◆设计控件的属性：为显示美观，首先将文本编辑框和静态文本框的字号分别设置为20和16，将3个静态文本框的标题分别改为“频率1”、“频率2”和“时间”，将按钮的标题改为“绘图”。

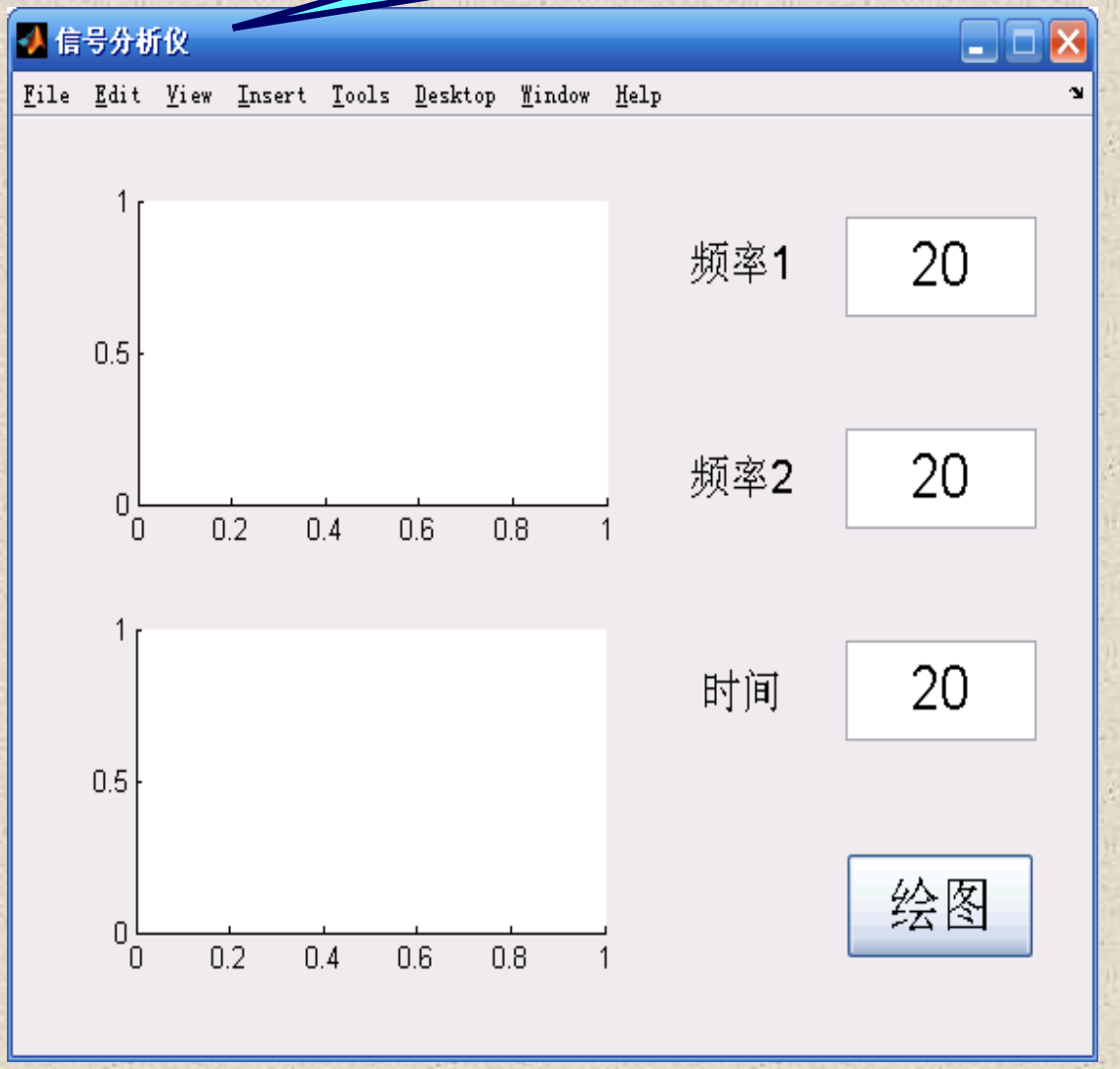
◆设置其他绘图属性。如设置主窗口的标题为“信号分析仪”。

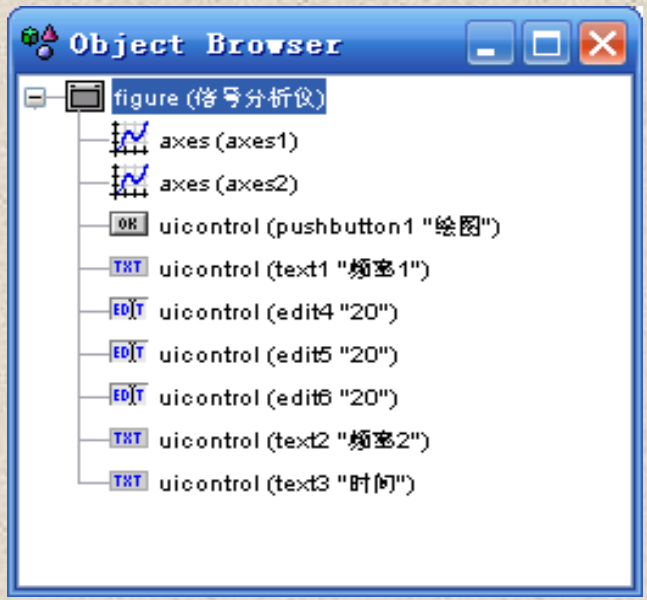
上述步骤基本完成了图形界面的设计，如下图所示：

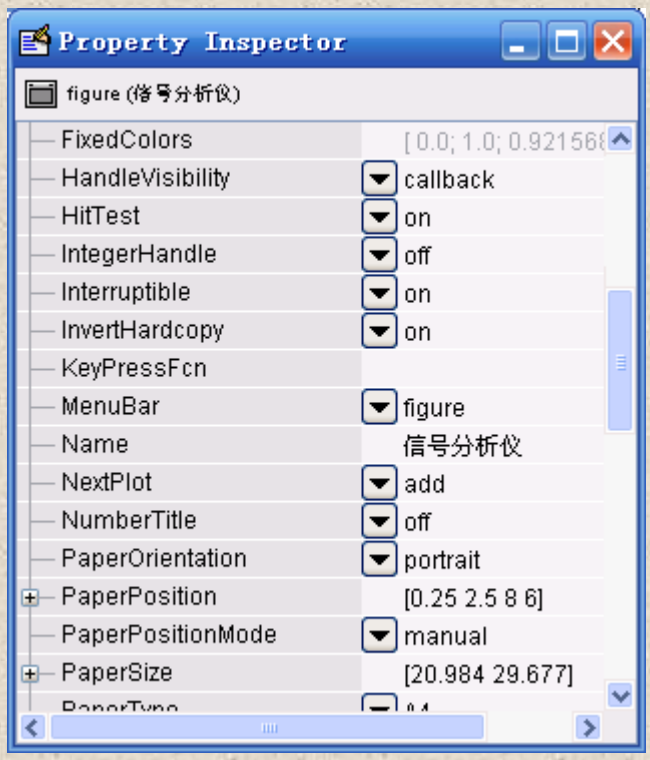


上述图形界面设计运行后显示的图形如下：

其设置如右图所示







（二）设置控件的标识

控件的标识(Tag)用于对各控件的识别。每个控件在创建时都会由开发环境自动产生一个标识，在程序设计中，为了编辑、记忆和维护的方便，一般为控件设置一个新的标识。

本例设置第一个坐标轴的标识为：frequency_axes, 用于显示频域图形; 第二个坐标轴的标识为：time_axes, 用于显示时域图形。三个文本编辑框的标识为f1_input, f2_input, t_input,分别用于输入两个频率和自变量时间的间隔。由于不需要返回3个静态文本框和按钮的值，这些控件的标识可以使用缺省值。

（三）编写代码

GUI图形界面的功能，还是要通过一定的设计思路和计算方法，由特定的程序来实现。为了实现程序的功能，还需要在运行程序前编写一些代码，完成程序中变量的赋值、输入输出、计算及绘图等工作。

1. 设置对象的初始值

分别设置三个文本编辑框的初始值为：

f1_input=20

f2_input=50

t_input=0:0.001:0.5

2. 编写代码

为按钮的调用函数编写代码，这段代码放在按钮的调用函数**pushbutton1_Callback()**中，代码包括以下部分：

（1） 从**GUI**获得用户输入的数据。本例中输入的3个数据分别为频率1、频率2和时间间隔。

```
f1=str2double(get(handles.f1_input,'String'));
f2=str2double(get(handles.f2_input,'String'));
t=eval(get(handles.t_input,'String'));
```

（2）计算数据。计算函数值，按指定点进行快速傅立叶变换，并计算频域的幅值和频域分辨率。

```
x=sin(2*pi*f1*t)+sin(2*pi*f2*t);
y=fft(x,512);
m=y.*conj(y)/512;
f=1000*(0:256)/512;
```

（3）在第一个坐标轴中绘制频域曲线。

```
axes(handles.frequency_axes)
plot(f,m(1:257))
set(handles.frequency_axes,'XminorTick','on')
grid on
```

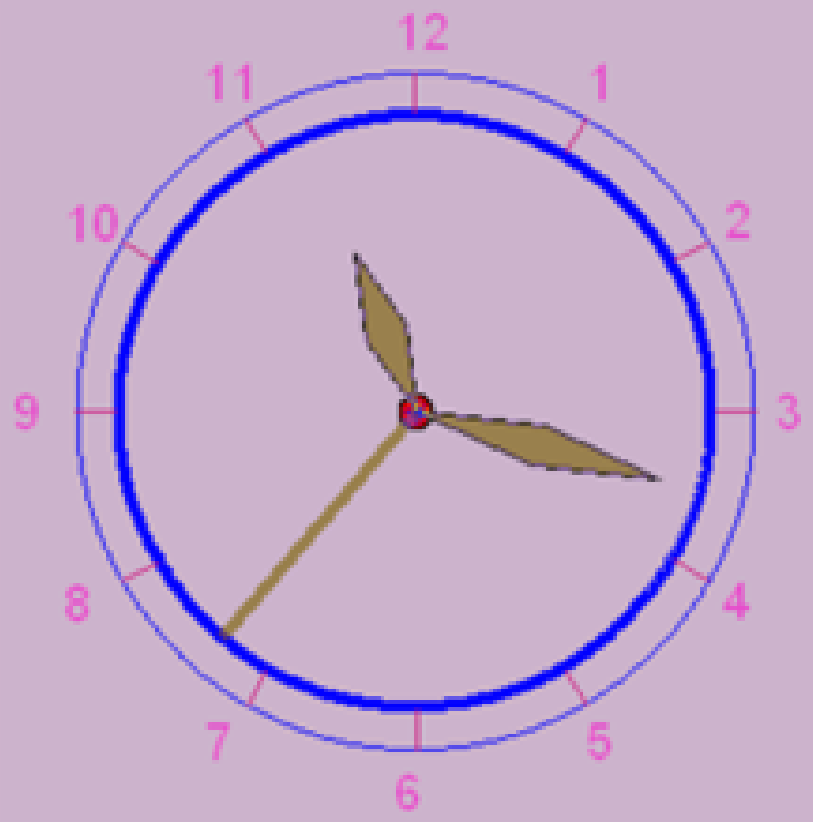
（4）在第二个坐标轴中绘制时域曲线。

```
axes(handles.time_axes)  %选择适当的坐标轴
plot(t,x)
set(handles.time_axes,'XminorTick','on')
grid on
```

3. 运行程序

时钟

更改日期　恢复当前日期　更改时间　恢复当前时间　退出



2008 年　6　月　9　日

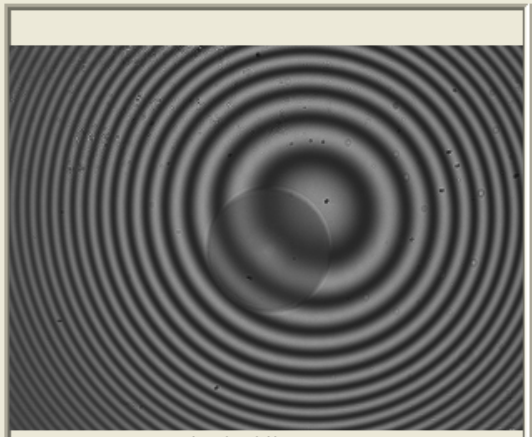
日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

制作By涂一云

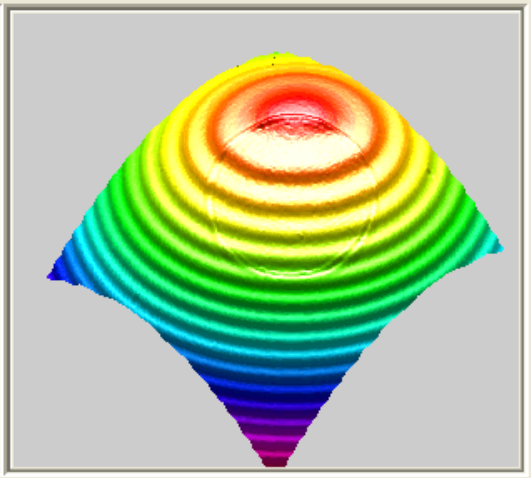

信号系统实验平台
信号系统实验平台
信号实验
连续信号的时域分析
离散信号的时域分析
连续信号的傅里叶分析
离散信号的傅里叶分析
傅里叶变换的基本性质
系统实验
连续系统的时域分析
离散系统的时域分析
系统的稳定性分析
LTI系统的频域分析
退出实验

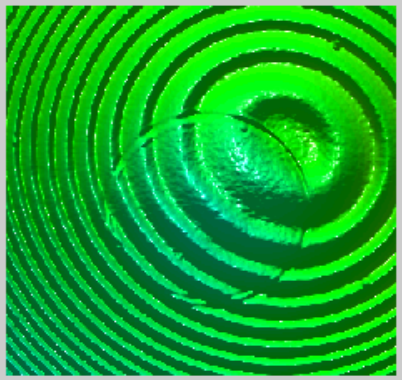
维度科技 Sana v4.3.3

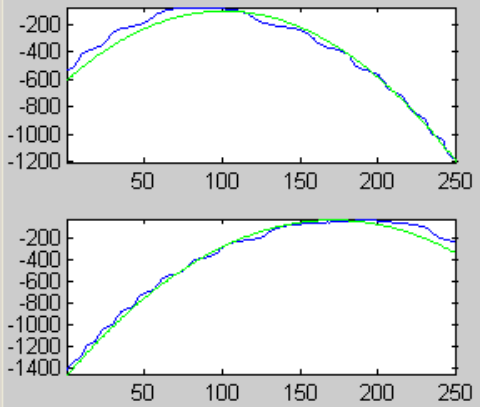
文件 系统设置 测量设置 校准 图像设置 窗口设置 帮助



实时图像







设置信息

连接器种类:	2.5mm/PC
连接器ID:	☑ 自动增加 ☐ 自定义
数据保存方式:	不保存
报告保存方式:	不保存
文件名:	test
连接器描述:	

标准 ☑ IEC ☐ Telcordia ☐ 自定义

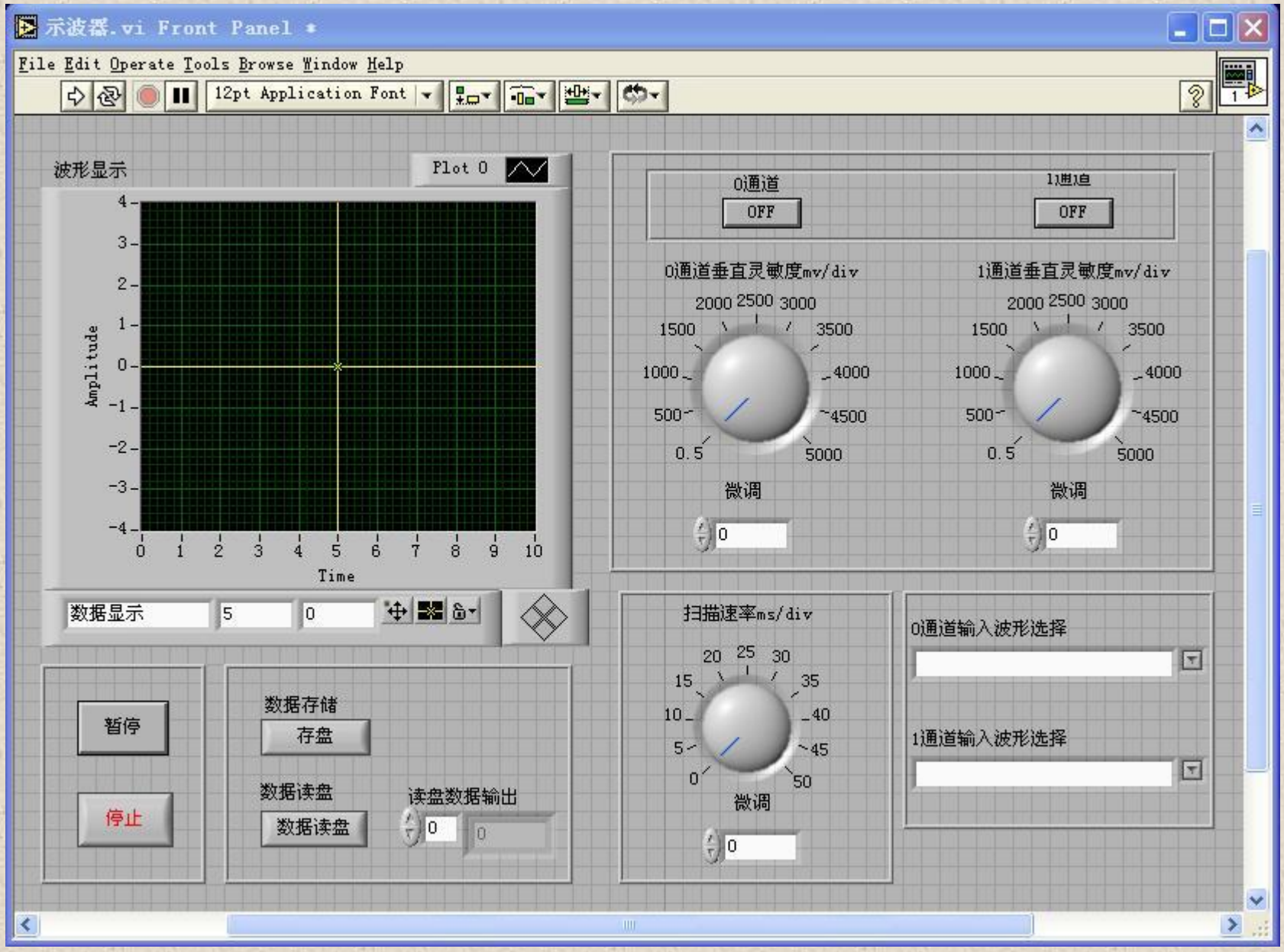
曲率半径(mm):	10	~	25
顶点偏移(μm):	0	~	50
光纤高度(nm):	-123	~	50
APC角度(deg)	\	~	\
APC键度误差(deg):	\	~	\

测量结果

连接器ID:	1
曲率半径(mm):	10.14
顶点偏移(μm):	53.84
光纤高度(nm):	-58
APC角度(deg):	\
APC键度误差(deg):	\

FAIL

Setup Calibrate Report Data Measure


示波器.vi Front Panel *
File Edit Operate Tools Browse Window Help
12pt Application Font
波形显示
Plot 0
Amplitude
Time
数据显示
0通道
OFF
1通道
OFF
0通道垂直灵敏度mv/div
1通道垂直灵敏度mv/div
微调
微调
扫描速率ms/div
微调
0通道输入波形选择
1通道输入波形选择
暂停
停止
数据存储
存盘
数据读盘
数据读盘
读盘数据输出