第 11 章 图形用户界面（GUI）制作

用户界面（或接口）是指：人与机器（或程序）之间交互作用的工具和方法。如键盘、鼠标、跟踪球、话筒都可成为与计算机交换信息的接口。

图形用户界面（Graphical User Interfaces ，GUI）则是由窗口、光标、按键、菜单、文字说明等对象（Objects）构成的一个用户界面。用户通过一定的方法（如鼠标或键盘）选择、激活这些图形对象，使计算机产生某种动作或变化，比如实现计算、绘图等。

假如读者所从事的数据分析、解方程、计算结果可视工作比较单一，那么一般不会考虑 GUI 的制作。但是如果读者想向别人提供应用程序，想进行某种技术、方法的演示，想制作一个供反复使用且操作简单的专用工具，那么图形用户界面也许是最好的选择之一。

MATLAB 为表现其基本功能而设计的演示程序 demo 是使用图形界面的最好范例。MATLAB 的用户，在指令窗中运行 demo 打开那图形界面后，只要用鼠标进行选择和点击，就可浏览那丰富多彩的内容。

即便比较熟悉 MATLAB 的读者，在他初次编写 GUI 程序时，也会感到棘手。为使读者获得制作自己 GUI 的体验，本章“入门”节提供了一个简单的示例。读者只要输入所提供的程序，就可引出相应的界面。

本章第 2 节叙述图形用户界面的设计原则和一般制作步骤。第 3、4 节分别介绍用户菜单、用户控件的制作。出于“由浅入深”的考虑，前 4 节制作 GUI 是通过 M 脚本文件实现的。利用 M 函数文件制作 GUI，需要解决数据传递问题，为此专设第 5 节给予阐述和示例。MATLAB 5.x 版为方便用户制作图形界面，提供了一个交互式的设计工具 guide 。关于该工具的使用方法，被放在第 6 节中，以一个综合例题为设计目标逐步展开。

在此提醒读者，假如要比较准确的理解本章程序和掌握本章内容，请先阅读第 10 章关于图柄的内容。

11.1入门

【*例 11.1-1】对于传递函数为$G=\dfrac{1}{s^2+2\zeta s+1}$的归一化二阶系统，制作一个能绘制该系统单位阶跃响应的图形用户界面。本例演示：（A）图形界面的大致生成过程；（B）静态文本和编辑框的生成；（C）坐标方格控制键的形成；（D）如何使用该界面。

（1）产生图形窗和轴位框：

```
clf reset
H=axes('unit','normalized','position',[0,0,1,1],'visible','off');
set(gcf,'currentaxes',H);
str='\fontname{隶书}归一化二阶系统的阶跃响应曲线';
text(0.12,0.93,str,'fontsize',13);
h_fig=get(H,'parent');
set(h_fig,'unit','normalized','position',[0.1,0.2,0.7,0.4]);
h_axes=axes('parent',h_fig,...
   'unit','normalized','position',[0.1,0.15,0.55,0.7],...
   'xlim',[0 15],'ylim',[0 1.8],'fontsize',8);
```

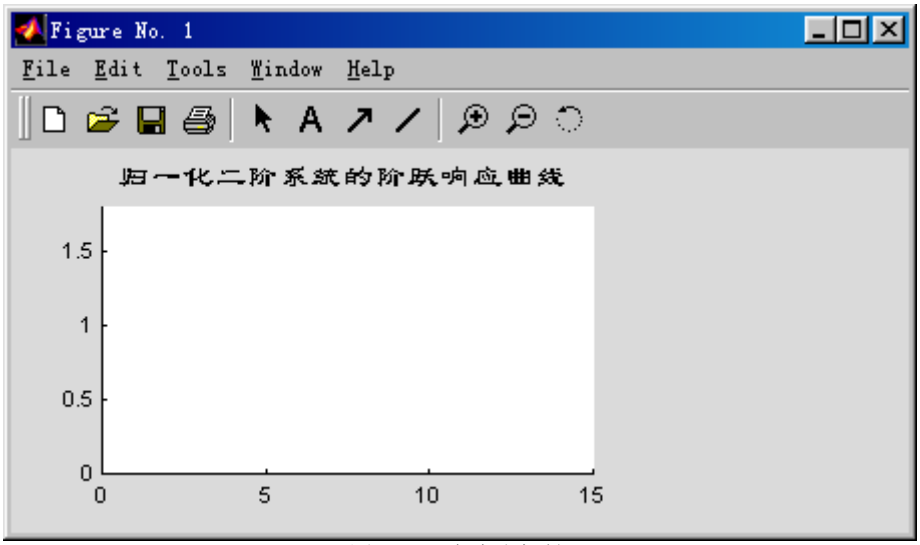

图 11.1-1 产生坐标轴

（2）在坐标框右侧生成作解释用的“静态文本”和可接受输入的“编辑框”：

```
h_text=uicontrol(h_fig,'style','text',...
   'unit','normalized','position',[0.67,0.73,0.25,0.14],...
   'horizontal','left','string',{'输入阻尼比系数','zeta ='});
h_edit=uicontrol(h_fig,'style','edit',...
   'unit','normalized','position',[0.67,0.59,0.25,0.14],...
   'horizontal','left',...
   'callback',[...
      'z=str2num(get(gcbo,''string''));',...
      't=0:0.1:15;',...
      'for k=1:length(z);',...
      's2=tf(1,[1 2*z(k) 1]); ',...
      'y(:,k)=step(s2,t);',...
      'plot(t,y(:,k));',...
      'if (length(z)>1) ,hold on,end,',...
      'end;',...
   'hold off,']);
```

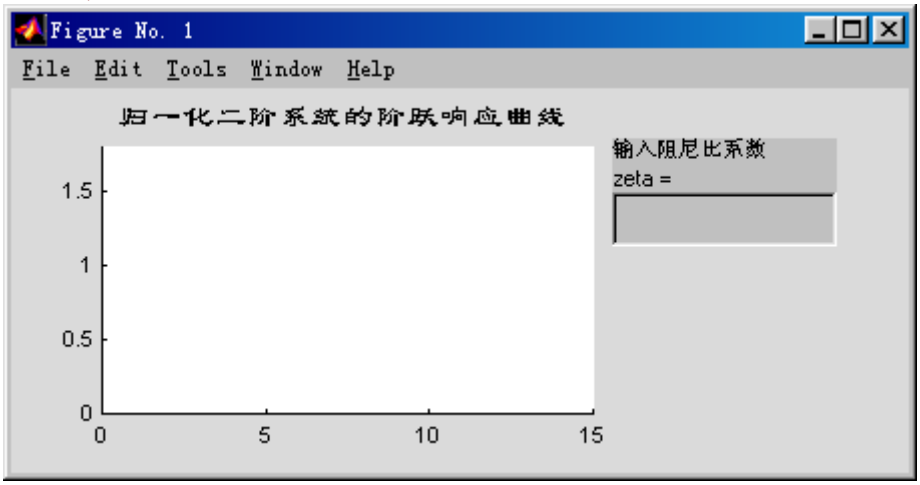

图 11.1-2 在图形界面中添加编辑框和文本框

（3）形成坐标方格控制按键：

```
h_push1=uicontrol(h_fig,'style','push',...
```

```
    'unit','normalized','position',[0.67,0.37,0.12,0.15],...
    'string','grid on','callback','grid on');
h_push2=uicontrol(h_fig,'style','push',...
    'unit','normalized','position',[0.67,0.15,0.12,0.15],...
    'string','grid off','callback','grid off');
```

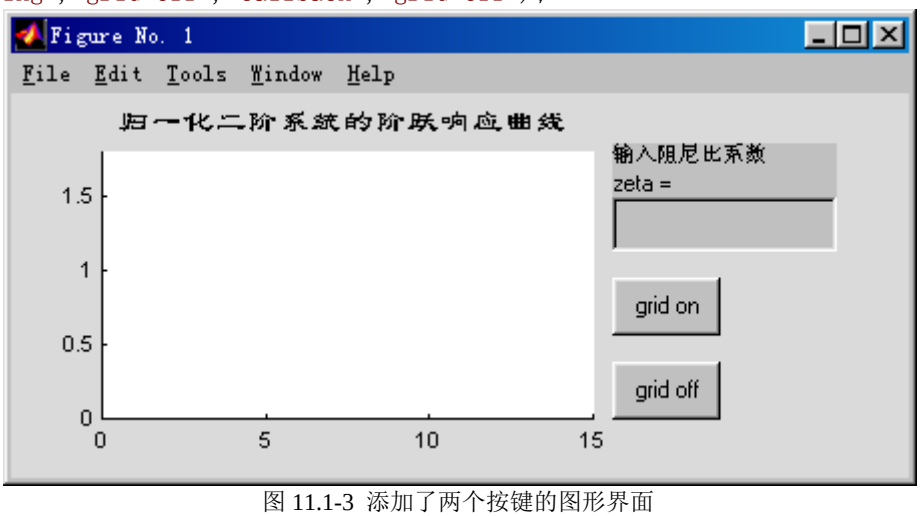

图 11.1-3 添加了两个按键的图形界面

（4）输入阻尼比系数，可得单位阶跃响应曲线：

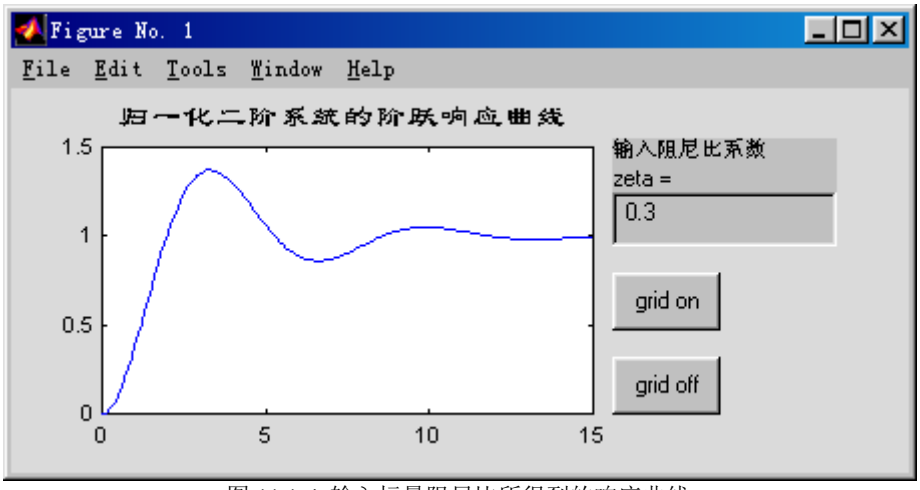

图 11.1-4 输入标量阻尼比所得到的响应曲线

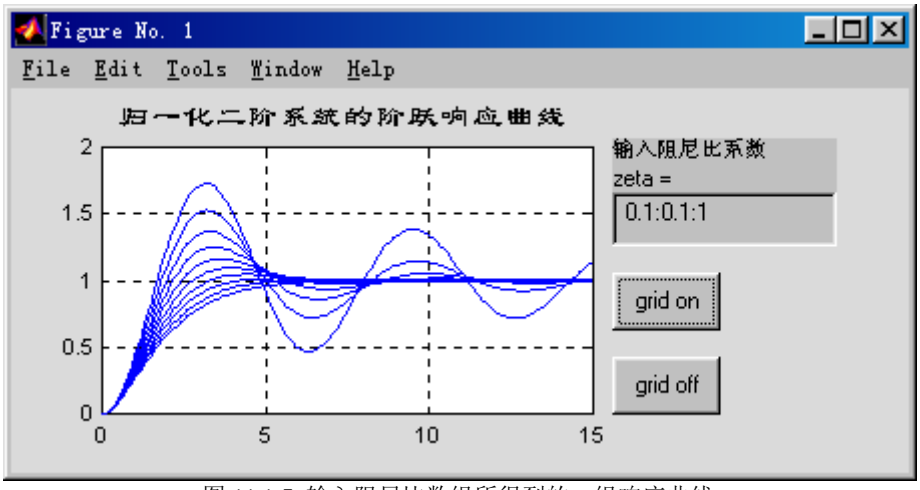

图 11.1-5 输入阻尼比数组所得到的一组响应曲线

11.2图形用户界面的设计原则和一般步骤

11.2.1设计原则

11.2.2 一般制作步骤

11.3界面菜单（uimenu）

11.3.1图形窗的标准菜单

【例 11.3.1-1】本例说明：如何隐藏和恢复标准菜单的显示。

（1）获得缺省设置的标准菜单

```
figure
```

（2）隐去标准菜单的两种方法

```
set(H_fig , 'MenuBar','none');
set(gcf,'menubar',menubar);
```

（3）恢复图形窗上标准菜单

```
set(gcf,'menubar','figure');
```

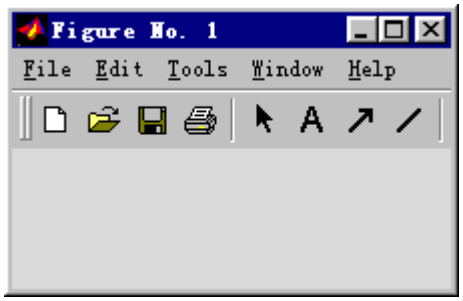

图 11.3.1-1 含有菜单条的图形窗

图 11.3.1-2 移去菜单条的图形窗

11.3.2自制的用户菜单

【*例 11.3.2-1】本例演示：如何自制一个带下拉菜单表的用户菜单（如图 11.3.2-1 所示）。该菜单能使图形窗背景颜色设置为兰色或红色。

```
figure                                          %创建一个图形窗
h_menu=uimenu(gcf,'label','Color');             %制作用户顶层菜单项Color      <2>
h_submenu1=uimenu(h_menu,'label','Blue',...     %制作下拉菜单项Blue           <3>
    'callback','set(gcf,''Color'',''blue'')');                              %<4>
h_submenu2=uimenu(h_menu,'label','Red',...      %制作下拉菜单Red              <5>
    'callback','set(gcf,''Color'',''red'')');                          %<6>
```

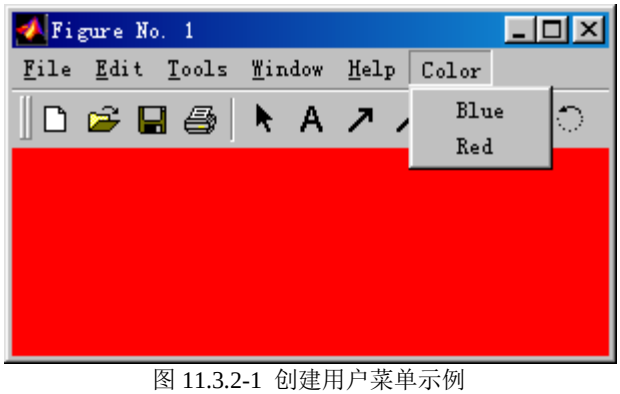

图 11.3.2-1 创建用户菜单示例

11.3.3 用户菜单的属性

11.3.3.1 回调属性和菜单名

【*例 11.3.3.1-1】本例的目标是：在图形窗上自制一个名为【Test】的“顶层菜单项”；当用鼠标点动该菜单项时，将产生一个带分格的封闭坐标轴。通过本例说明：（A）回调属性的运作机理；（B）用户顶层菜单项的制作（C）uimenu 属性的设置方法；（D）复杂字符串的构成方法和注意事项。

（1）在 MATLAB 指令窗中运行以下程序可产生带分格的封闭坐标轴（见图 11.3.3.1-1）

```
grid on,set(gca,'box','on')
```

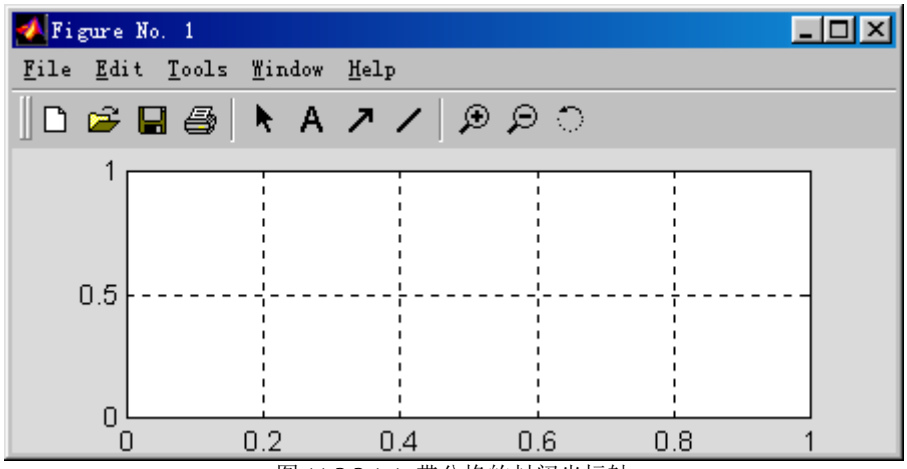

图 11.3.3.1-1 带分格的封闭坐标轴

（2）在 MATLAB 指令窗中用以下 eval 指令可产生与图 11.3.3.1-1 相同的界面

```
eval('grid on,set(gca,''box'',''on'')')
```

（3）产生图 11.3.3.1-2 界面的 uimenu 的书写格式一：直接连续表示法

```
uimenu('Label','Test','Callback','grid on,set(gca,''box'',''on''),')
```

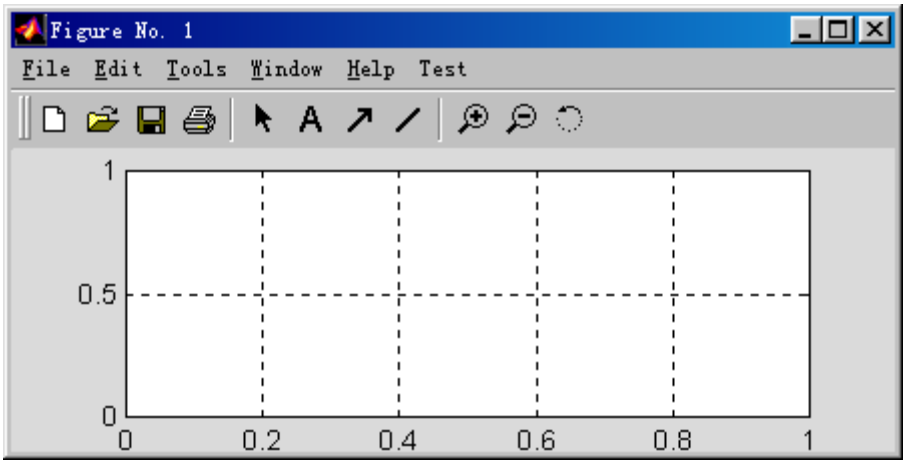

图 11.3.3.1-2 通过顶层菜单 Test 形成的带分格的封闭坐标轴

（4）产生图 11.3.3.1-2 界面的 uimenu 的书写格式二：方括号续行号表示法

```
uimenu('Label','Test', ...
       'Callback',['grid on,' , ...
                   'set(gca,''box'',''on'');'])
```

（5）产生图 11.3.3.1-2 界面的 uimenu 的书写格式三：串变量法

```
Lpv='Test';
Cpv=['grid on,','set(gca,''box'',''on''),'];
uimenu('Label', Lpv, 'Callback' , Cpv)
```

（6）产生图 11.3.3.1-2 界面的 uimenu 的书写格式四：构架表示法

```
PS.Label='Test';
PS.Callback=['grid on;','set(gca,''box'',''on'');'];
uimenu(PS)
```

11.3.3.2 设置简捷键或快捷键

【*例 11.3.3.2-1】本例目标：使图 11.3.2-1 所示菜单成为图 11.3.3.2-1 那样，Color 菜单项及其下拉的 Blue 菜单各带一个简捷键，而另一项下拉菜单 Red 带一个快捷键。

[EXM11332_1.M]

```
figure
h_menu=uimenu(gcf,'Label','&Color');          %带简捷键C的用户菜单Color   <2>
h_submenu1=uimenu(h_menu,'Label','&Blue',... %带简捷键B的的下拉菜单Blue  <3>
   'Callback','set(gcf,''color'',''blue'')');
h_submenu2=uimenu(h_menu,'label','Red',...    %制作另一个下拉菜单Red
   'Callback','set(gcf,''color'',''red'')',...
   'Accelerator','r');                        %为Red菜单设置快捷键R       <7>
```

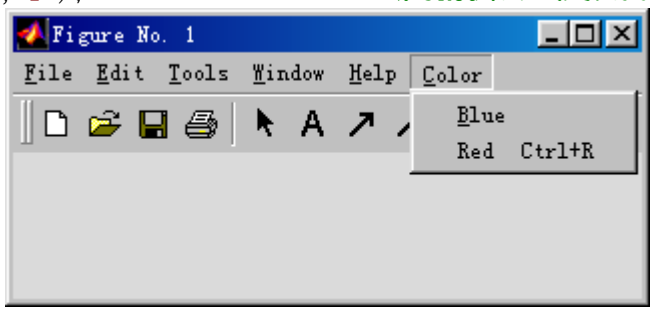

图 11.3.3.2-1 为用户菜单设置快捷键

11.3.3.3 用户菜单的外观设计

【*例 11.3.3.3-1】本例演示：（A）把用户菜单 'Option' 设置为顶层的第 3 菜单项；（B）下拉菜单被两条分隔线分为三个菜单区；（C）最下菜单项又有两个子菜单组成。

（1）编写程序，生成如图 11.3.3.3-1 所示界面

[EXM11333_1.M]

```
figure
h_menu=uimenu('label','Option','Position',3);
h_sub1=uimenu(h_menu,'label','grid on','callback','grid on');
h_sub2=uimenu(h_menu,'label','grid off','callback','grid on');
h_sub3=uimenu(h_menu,'label','box on','callback','box on',...
    'separator','on');                                          %<6>
h_sub4=uimenu(h_menu,'label','box off','callback','box off');
h_sub5=uimenu(h_menu,'label','Figure Color','Separator','on');  %<8>
h_subsub1=uimenu(h_sub5,'label','Red','ForeGroundColor','r',... %<9>
   'callback','set(gcf,''Color'',''r'')');
h_subsub2=uimenu(h_sub5,'label','Reset',...
   'callback','set(gcf,''Color'',''w'')');
```

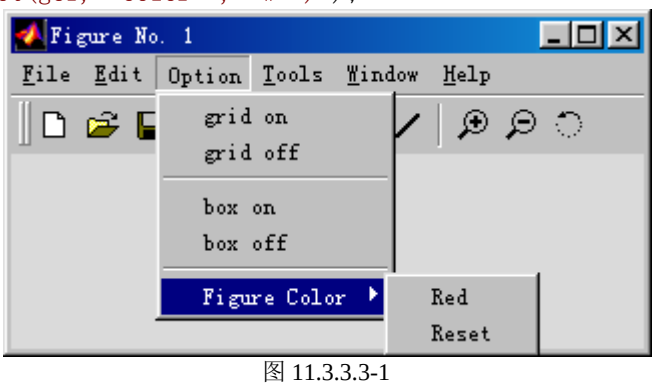

图 11.3.3.3-1

（2）位置属性的获取

```
Pos_O=get(h_menu,'position'),                  %查询 Option 菜单位置值
Pos_BoxOn=get(h_sub3,'position')               %查询 box ob 子菜单位置值
Pos_Red=get(h_subsub1,'position')          %查询 red 子菜单的位置值
Pos_O =
     3
Pos_BoxOn =
     3
Pos_Red =
     1
```

【*例 11.3.3.3-2】本例演示：当某菜单项选中后，如何使该菜单项贴上检录符“√”。

[exm11333_2.m]

```
figure
h_menu=uimenu('label','Option');
h_sub1=uimenu(h_menu,'label','Grid on',...                  %<3>
```

```
    'callback',[...
     'grid on,',...
     'set(h_sub1,''checked'',''on''),',...
     'set(h_sub2,''checked'',''off''),',...
    ]);
h_sub2=uimenu(h_menu,'label','Grid off',...                    %<4>
    'callback',[...
     'grid off,',...
     'set(h_sub2,''checked'',''on''),',...
     'set(h_sub1,''checked'',''off''),',...
     ]);
```

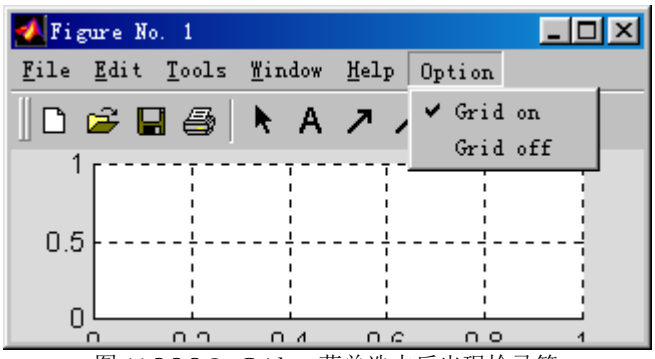

图 11.3.3.3-2　Grid on 菜单选中后出现检录符

11.3.3.4　使能（Enable）与可见性（Visible）属性

【*例 11.3.3.4-1】本例目标：制作一个带四个子菜单项的顶层菜单项；该下拉菜单分为两个功能区；每个功能区的两个菜单项是相互对立的，因此采用使能属性处理；当图形窗坐标轴消隐时，整个坐标分隔控制功能区不可见。

（1）编写如下脚本 M 文件 exm11334_1.m

[EXM11334_1.M]

```
clf
h_menu=uimenu('label','Option');          %产生顶层菜单项Option
h_sub1=uimenu(h_menu,'label','Axis on');  %产生Axis on菜单项，由缺省设置而使能
h_sub2=uimenu(h_menu,'label','Axis off',...
   'enable','off');                       %产生Axis off菜单项，但失能
h_sub3=uimenu(h_menu,'label','Grid on',...
   'separator','on','visible','off');     %产生与上分隔的Grid on菜单项，但不可见
h_sub4=uimenu(h_menu,'label','Grid off',...
   'visible','off');                      %产生Grid off菜单项，但不可见
set(h_sub1,'callback',[...                %选中Axis on菜单项后，产生回调操作
   'Axis on,',...                             %画坐标
   'set(h_sub1,''enable'',''off''),',...      %Axis on菜单项失能
   'set(h_sub2,''enable'',''on''),',...       %Axis off菜单项使能
   'set(h_sub3,''visible'',''on''),',...      %Grid on菜单项可见
   'set(h_sub4,''visible'',''on''),']);       %Grid off菜单项可见
set(h_sub2,'callback',[...                % %选中Axis off菜单项后，产生回调操作
   'axis off,',...                            %使坐标消失
   'set(h_sub1,''enable'',''on''),',...       %Axis on菜单项使能
   'set(h_sub2,''enable'',''off''),',...      %Axis off菜单项失能
```

```
    'set(h_sub3,''visible'',''off''),',...     %Grid on菜单项不可见
    'set(h_sub4,''visible'',''off''),']);      %Grid off菜单项不可见
set(h_sub3,'callback',[...                    %选中Grid on菜单项后，产生回调
    'grid on,',...                             %画坐标分格线
    'set(h_sub3,''enable'',''off''),',...      %Grid on菜单项失能
    'set(h_sub4,''enable'',''on''),']);        %Grid off菜单项使能
set(h_sub4,'callback',[...                    %选中Grid off菜单项，产生回调
    'grid off,',...                            %消除坐标分格线
    'set(h_sub3,''enable'',''on''),',...       %Grid on菜单项使能
    'set(h_sub4,''enable'',''off''),']);       %Grid off菜单项失能
```

（2）在 MATLAB 指令窗中运行 exm11334_1，得到图 11.3.3.4-1 所示的界面

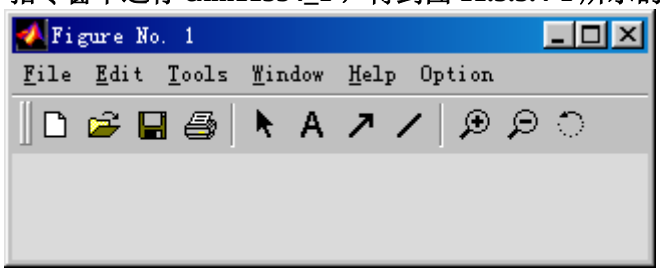

图 11.3.3.4-1

（3）选中【Option】菜单项，界面呈现如图 11.3.3.4-2 所示。

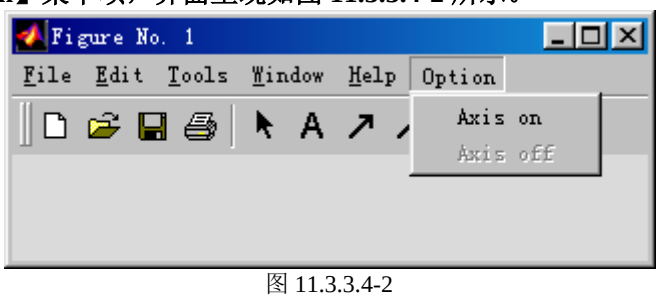

图 11.3.3.4-2

（4）选中【Option：Axis on】后，界面呈现如图 11.3.3.4-3 所示。

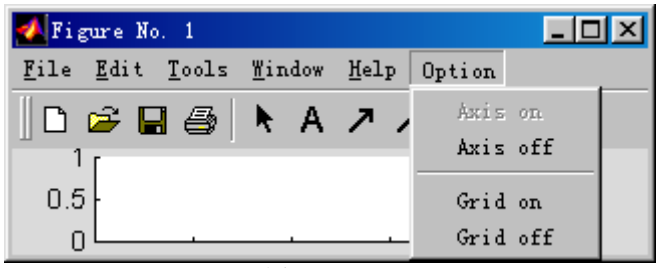

图 11.3.3.4-3

（5）选中【Option：Grid on】后，界面呈现如图 11.3.3.4-4 所示。

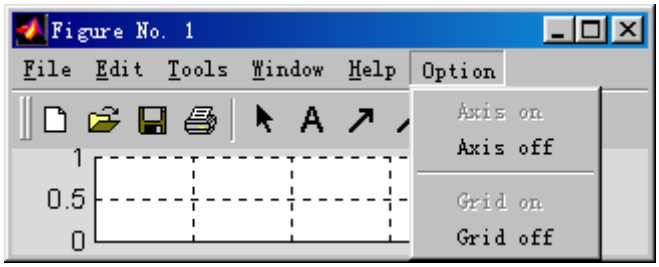

图 11.3.3.4-4

11.3.4 现场菜单的制作

【*例 11.3.4-1】目标：绘制一条 Sa 曲线，创建一个与之相联系的现场菜单，用以控制 Sa 曲线的颜色。

（1）编写脚本 M 文件 exm1134_1.m

[EXM1134_1.M]

```
t=(-3*pi:pi/50:3*pi)+eps;
y=sin(t)./t;
hline=plot(t,y);                                    %绘制Sa曲线
cm=uicontextmenu;                                   %创建现场菜单
%制作具体菜单项，定义相应的回调
uimenu(cm,'label','Red','callback','set(hline,''color'',''r''),')
uimenu(cm,'label','Blue','callback','set(hline,''color'',''b''),')
uimenu(cm,'label','Green','callback','set(hline,''color'',''g''),')
set(hline,'uicontextmenu',cm)                       %使cm现场菜单与Sa曲线相联系
```

（2）在指令窗中运行文件 exm1134_1.m，得到图 11.3.4-1 所示的（但为蓝色的）Sa 曲线。

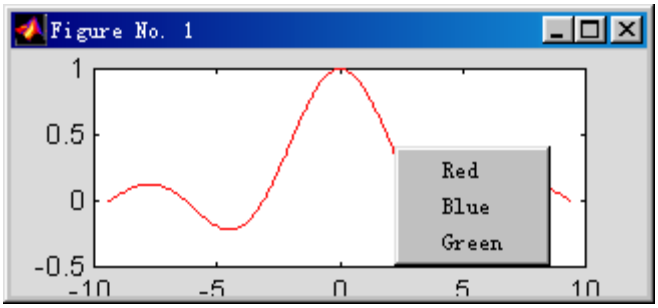

图 11.3.4-1 Context 菜单

（3）将鼠标指针指向线条，点击鼠标右键的同时弹出现场菜单，在选中某菜单项（如 Red）后，Sa 曲线就改变（为红）颜色（如图 11.3.4-1 所示）。

11.4用户控件（uicontrol）

11.4.1 控件制作函数

11.4.2 用户控件的种类

11.4.3 控件制作示例

11.4.3.1　双位按键、无线电按键、控件区域框示例

【*例 11.4.3.1-1】目标：创建一个界面包含 4 种控件：静态文本、“无线电”选择开关、双位按键、控件区域框。

[EXM11431_1.M]

```
clf reset
set(gcf,'menubar','none')
set(gcf,'unit','normalized','position',[0.2,0.2,0.64,0.32]);
set(gcf,'defaultuicontrolunits','normalized')    %设置用户缺省控件单位属性值
```

```
h_axes=axes('position',[0.05,0.2,0.6,0.6]);
t=0:pi/50:2*pi;y=sin(t);plot(t,y);
set(h_axes,'xlim',[0,2*pi]);
set(gcf,'defaultuicontrolhorizontal','left');
htitle=title('正弦曲线');
set(gcf,'defaultuicontrolfontsize',12);          %设置用户缺省控件字体属性值
uicontrol('style','frame',...                    %创建用户控件区          <11>
   'position',[0.67,0.55,0.25,0.25]);
uicontrol('style','text',...                     %创建静态文本框          <13>
   'string','正斜体图名:',...
   'position',[0.68,0.77,0.18,0.1],...
   'horizontal','left');
hr1=uicontrol(gcf,'style','radio',...            %创建“无线电”选择按键 <17>
   'string','正体',...                           %按键功能的文字标识'正体'
   'position',[0.7,0.69,0.15,0.08]);             %按键位置
set(hr1,'value',get(hr1,'Max'));%因图名缺省使用正体，所以小圆圈应被点黑 <20>
set(hr1,'callback',[...                                %                <21>
       'set(hr1,''value'',get(hr1,''max'')),',...      %选中将小圆圈点黑  <22>
       'set(hr2,''value'',get(hr2,''min'')),',...      %将“互斥”选项点白 <23>
       'set(htitle,''fontangle'',''normal''),',...     %使图名字体正体显示
    ]);
hr2=uicontrol(gcf,'style','radio',...            %创建“无线电”选择按键 <26>
    'string','斜体',...                          %按键功能的文字标识'斜体'
    'position',[0.7,0.58,0.15,0.08],...          %按键位置
    'callback',[...
       'set(hr1,''value'',get(hr1,''min'')),',...      %                <30>
       'set(hr2,''value'',get(hr2,''max'')),',...      %                <31>
       'set(htitle,''fontangle'',''italic'')',...      %使图名字体斜体显示
    ]);                                                %                <33>
  ht=uicontrol(gcf,'style','toggle',...          %制作双位按键            <34>
     'string','Grid',...
     'position',[0.67,0.40,0.15,0.12],...
     'callback','grid');
```

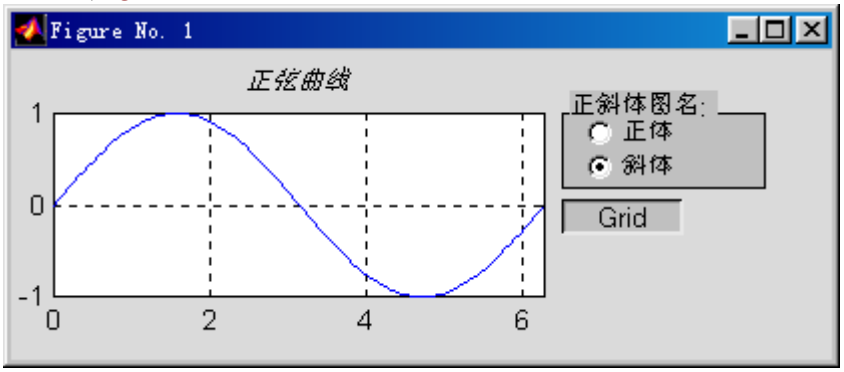

图 11.4.3.1-1 静态文本、选择开关、双位按键及控件区域框

11.4.3.2 静态文本框、滑动键、检录框示例

【*例 11.4.3.2-1】目标：制作演示“归一化二阶系统单位阶跃响应”的交互界面。在该界面中，阻尼比可在[0.02,2.02]中连续调节，标志当前阻尼比值；可标志峰值时间和大小；可标

志（响应从 0 到 0.95 所需的）上升时间。本例涉及以下主要内容：（A）静态文本的创建和实时改写。（B）滑动键的创建；'Max' 和 'Min' 的设置；'Value' 的设置和获取。（C）检录框的创建；'Value' 的获取。（D）受多个控件影响的回调操作。

[EXM11432_1.M]

```
clf reset
set(gcf,'unit','normalized','position',[0.1,0.2,0.64,0.35]);
set(gcf,'defaultuicontrolunits','normalized');
set(gcf,'defaultuicontrolfontsize',12);
set(gcf,'defaultuicontrolfontname','隶书');
set(gcf,'defaultuicontrolhorizontal','left');
str='归一化二阶系统阶跃响应曲线';
set(gcf,'name',str,'numbertitle','off');      %书写图形窗名
h_axes=axes('position',[0.05,0.2,0.6,0.7]);   %定义轴位框位置
set(h_axes,'xlim',[0,15]);                    %设置时间轴长度
str1='当前阻尼比=';
t=0:0.1:10;z=0.5;y=step(1,[1 2*z 1],t);
hline=plot(t,y);
htext=uicontrol(gcf,'style','text',...    %制作静态说明文本框              〈14〉
   'position',[0.67,0.8,0.33,0.1],...
   'string',[str1,sprintf('%1.4g\',z)]);
hslider=uicontrol(gcf,'style','slider',...    %创建滑动键                  〈17〉
   'position',[0.67,0.65,0.33,0.1],...
   'max',2.02,'min',0.02,...        %设最大阻尼比为2，最小阻尼比为0.02     〈19〉
   'sliderstep',[0.01,0.05],...%箭头操纵滑动步长1%，游标滑动步长5%         〈20〉
   'Value',0.5);                 %缺省取阻尼比等于0.5                    〈21〉
hcheck1=uicontrol(gcf,'style','checkbox',... %创建峰值检录框              〈22〉
   'string','最大峰值' ,...
   'position',[0.67,0.50,0.33,0.11]);
vchk1=get(hcheck1,'value');                   %获得峰值检录框的状态值     〈25〉
hcheck2=uicontrol(gcf,'style','checkbox',... %创建上升时间检录框          〈26〉
   'string','上升时间(0->0.95)',...
   'position',[0.67,0.35,0.33,0.11]);
vchk2=get(hcheck2,'value');                   %获得上升时间检录框的状态值〈29〉
set(hslider,'callback',[...                   %操作滑动键,引起回调        〈30〉
   'z=get(gcbo,''value'');',...               %获得滑动键状态值           〈31〉
   'callcheck(htext,str1,z,vchk1,vchk2)']);   %被回调的函数文件           〈32〉
set(hcheck1,'callback',[...                   %操作峰值检录框,引起回调    〈33〉
   'vchk1=get(gcbo,''value'');',...           %获得峰值检录框状态值       〈34〉
   'callcheck(htext,str1,z,vchk1,vchk2)']);   %被回调的函数文件           〈35〉
set(hcheck2,'callback',[...                   %操作峰值检录框,引起回调    〈36〉
   'vchk2=get(gcbo,''value'');',...           %获得峰值检录框状态值       〈37〉
   'callcheck(htext,str1,z,vchk1,vchk2)']);   %被回调的函数文件           〈38〉
```

[CALLCHECK.M]

```
function callcheck(htext,str1,z,vchk1,vchk2)
cla,set(htext,'string',[str1,sprintf('%1.4g\',z)]);%更新静态文本框内容 〈2〉
dt=0.1;t=0:dt:15;N=length(t);y=step(1,[1 2*z 1],t);plot(t,y);
if vchk1                                  %假如峰值框被选中              〈4〉
   [ym,km]=max(y);
   if km<(N-3)                            %假如在设定时间范围内能插值   〈6〉
      k1=km-3;k2=km+3;k12=k1:k2;tt=t(k12);
```

```
        yy=spline(t(k12),y(k12),tt);                    %局部样条插值          <8>
        [yym,kkm]=max(yy);                              %求更精确的峰值位置
        line(tt(kkm),yym,'marker','.',...               %画峰值点              <10>
           'markeredgecolor','r','markersize',20);
        ystr=['ymax = ',sprintf('%1.4g\',yym)];
        tstr=['tmax = ',sprintf('%1.4g\',tt(kkm))];
        text(tt(kkm),1.05*yym,{ystr;tstr})
    else                                        %假如在设定时间范围内不能插值      <15>
        text(10,0.4*y(end),{'ymax --> 1';'tmax --> inf'})
    end
end
if vchk2                                        %假如上升时间框被选中              <19>
    k95=min(find(y>0.95));k952=[(k95-1),k95];
    t95=interp1(y(k952),t(k952),0.95);          %线性插值              <21>
    line(t95,0.95,'marker','o','markeredgecolor','k','markersize',6);
    tstr95=['t95 = ',sprintf('%1.4g\',t95)];
    text(t95,0.65,tstr95)
end
```

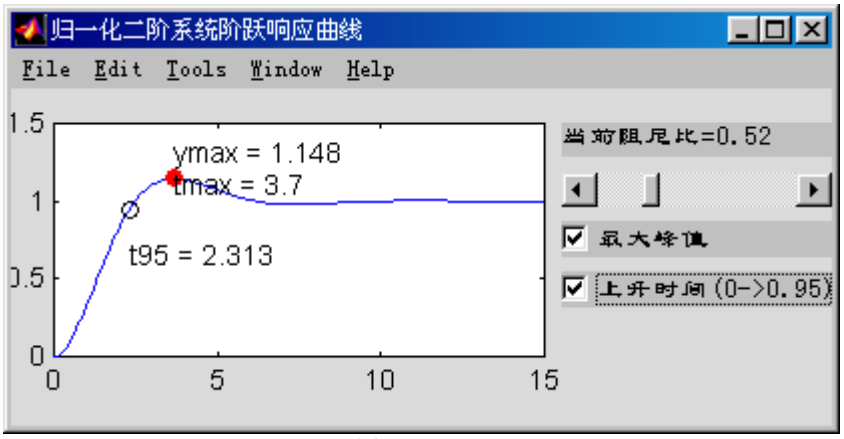

图 11.4.3.2-1

11.4.3.3 可编辑框、弹出框、列表框、按键示例

【*例 11.4.3.3-1】目标：制作一个能绘制任意图形的交互界面。它包括：可编辑文本框、弹出框、列表框。本例的关键内容是：如何使编辑框允许输入多行指令。

[EXM11433_1.M]

```
clf reset                                       %                      <1>
set(gcf,'unit','normalized','position',[0.1,0.4,0.85,0.35]);%设置图形窗大小
set(gcf,'defaultuicontrolunits','normalized');
set(gcf,'defaultuicontrolfontsize',11);
set(gcf,'defaultuicontrolfontname','隶书');
set(gcf,'defaultuicontrolhorizontal','left');
set(gcf,'menubar','none');                      %删除图形窗工具条
str='通过多行指令绘图的交互界面';
set(gcf,'name',str,'numbertitle','off');        %书写图形窗名
h_axes=axes('position',[0.05,0.15,0.45,0.70],'visible','off');%定义轴位框位置
uicontrol(gcf,'Style','text',...                %制作静态文本框
   'position',[0.52,0.87,0.26,0.1],...
```

```
    'String','绘图指令输入框');
hedit=uicontrol(gcf,'Style','edit',...                    %制作可编辑文本框      <14>
    'position',[0.52,0.05,0.26,0.8],...
    'Max',2);                          %取2，使Max-Min>1，而允许多行输入          <16>
hpop=uicontrol(gcf,'style','popup',...                    %制作弹出菜单          <17>
    'position',[0.8,0.73,0.18,0.12],...
    'string','spring|summer|autumn|winter');%设置弹出框中选项名                  <19>
hlist=uicontrol(gcf,'Style','list',...                    %制作列表框            <20>
    'position',[0.8,0.23,0.18,0.37],...
    'string','Grid on|Box on|Hidden off|Axis off',...%设置列表框中选项名        <22>
    'Max',2);                          %取2，使Max-Min>1，而允许多项选择          <23>
hpush=uicontrol(gcf,'Style','push',...    %制作与列表框配用的按键                 <24>
    'position',[0.8,0.05,0.18,0.15],'string','Apply');
set(hedit,'callback','calledit(hedit,hpop,hlist)');%编辑框输入引起回调           <26>
set(hpop,'callback','calledit(hedit,hpop,hlist)');  %弹出框选择引起回调          <27>
set(hpush,'callback','calledit(hedit,hpop,hlist)');%按键引起的回调               <28>
```

[CALLEDIT.M]

```
function calledit(hedit,hpop,hlist)
ct=get(hedit,'string');                %获得输入的字符串函数                    <2>
vpop=get(hpop,'value');                %获得选项的位置标识                      <3>
vlist=get(hlist,'value');              %获得选项位置向量                        <4>
if ~isempty(ct)                        %可编辑框输入非空时                      <5>
   eval(ct')                           %运行从编辑文本框送入的指令              <6>
   popstr={'spring','summer','autumn','winter'};        %弹出框色图矩阵      <7>
   liststr={'grid on','box on','hidden off','axis off'};%列表框选项内容        <8>
   invstr={'grid off','box off','hidden on','axis on'};%列表框的逆指令          <9>
   colormap(eval(popstr{vpop}))        %采用弹出框所选色图                      <10>
   vv=zeros(1,4);vv(vlist)=1;
   for k=1:4
      if vv(k);eval(liststr{k});else eval(invstr{k});end %按列表选项影响图形
   end
end
```

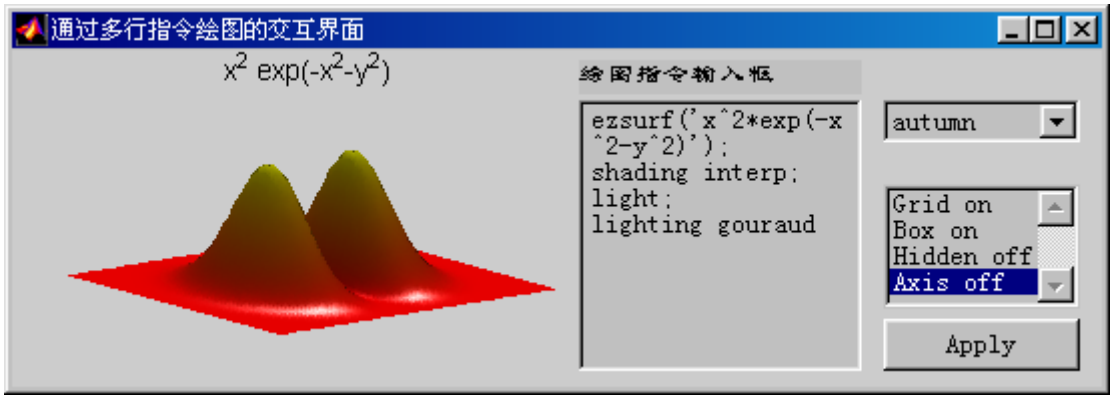

图 11.4.3.3-1

11.5 由 M 函数文件产生用户菜单和控件

11.5.1利用全局变量编写用户界面函数文件

【*例 11.5.1-1】目标：利用 M 函数文件创建与例 11.4.3.3-1 相同的用户界面。本例演示：如何依靠全局变量传递控件的图柄，从而保证回调动作正确执行。

（1）编写 M 函数文件 exm1151_1.m 和 calledit1.m

[exm1151_1.m]

```
function exm1151_1( )
global hedit hpop hlist
（这中间是：原 exm11433_1.m 第〈1〉行到第〈25〉行的全部指令）
set(hedit,'callback','calledit1');                %编辑框输入引起回调  〈26〉
set(hpop,'callback','calledit1');                 %弹出框选择引起回调  〈27〉
set(hpush,'callback','calledit1');                %按键引起的回调      〈28〉
```

[CALLEDIT1.M]

```
function calledit1( )
global hedit hpop hlist
（下面续接内容是：原 calledit.m 第〈2〉行以下的全部指令）
```

（2）在 MATLAB 指令窗中运行 exm1151_1 就可获得题目所要求的图形用户界面。

11.5.2利用 'UserData' 属性编写用户界面函数文件

【*例 11.5.2-1】目标：利用 M 函数文件创建与例 11.4.3.3-1 相同的用户界面。本例演示：如何依靠图形窗的'UserData' 属性传送用户控件的图柄，从而保证回调动作正确执行。

（1）编写 M 函数文件 exm1152_1.m 和 calledit2.m

[exm1152_1.m]

```
function exm1152_1( )
（这中间是：原 exm11433_1.m 第〈1〉行到第〈25〉行的全部指令）
set(hedit,'callback','calledit2');                %编辑框输入引起回调  〈26〉
set(hpop,'callback','calledit2');                 %弹出框选择引起回调  〈27〉
set(hpush,'callback','calledit2');                %按键引起的回调      〈28〉
set(gcf,'UserData',[hedit,hpop,hlist])
```

[calledit2.m]

```
function calledit2( )
H=get(gcf,'UserData');
ct=get(H(1),'string');          %获得输入的字符串函数          〈2〉
vpop=get(H(2),'value');         %获得选项的位置标识            〈3〉
vlist=get(H(3),'value');        %获得选项位置向量              〈4〉
（下面续接内容是：原 calledit.m 第〈5〉行以下的全部指令）
```

（2）在 MATLAB 指令窗中运行 exm1152_1 就可获得题目所要求的图形用户界面。

11.5.3利用递归法编写用户界面函数文件

【*例 11.5.3-1】目标：利用 M 函数文件创建与例 11.4.3.3-1 相同的用户界面。本例演示：如何依靠图形窗'UserData' 属性在递归调用中传送用户控件的图柄，保证回调动作正确执行。

（1）编写 M 函数文件 exm1153_1.m

[exm1153_1.m]

```
function exm1153_1(flag)
if nargin<1;flag='startup';end     %允许在无输入宗量形式下调用该函数      <2>
if ~ischar(flag);error('flag must be character ''startup''.');end
switch flag                        %切换控制                              <4>
case 'startup'                                  %                         <5>
clf reset                                       %                         <6>
set(gcf,'unit','normalized','position',[0.1,0.4,0.85,0.35]);
set(gcf,'defaultuicontrolunits','normalized');
set(gcf,'defaultuicontrolfontsize',11);
set(gcf,'defaultuicontrolfontname','隶书');
set(gcf,'defaultuicontrolhorizontal','left');
set(gcf,'menubar','none');                      %删除图形窗工具条
str='通过多行指令绘图的交互界面';
set(gcf,'name',str,'numbertitle','off');        %书写图形窗名
h_axes=axes('position',[0.05,0.15,0.45,0.70],'visible','off');
uicontrol(gcf,'Style','text',...                %制作静态文本框
   'position',[0.52,0.87,0.26,0.1],...
   'String','绘图指令输入框');
hedit=uicontrol(gcf,'Style','edit',...          %制作可编辑文本框      <19>
   'position',[0.52,0.05,0.26,0.8],...                    %             <20>
   'Max',2);                    %取2，使Max-Min>1，而允许多行输入      <21>
hpop=uicontrol(gcf,'style','popup',...          %制作弹出菜单          <22>
   'position',[0.8,0.73,0.18,0.12],...                    %             <23>
   'string','spring|summer|autumn|winter');%设置弹出框中选项名          <24>
hlist=uicontrol(gcf,'Style','list',...          %制作列表框            <25>
   'position',[0.8,0.23,0.18,0.37],...                    %             <26>
   'string','Grid on|Box on|Hidden off|Axis off',...%设置列表框中选项名 <27>
   'Max',2);                    %取2，使Max-Min>1，而允许多项选择      <28>
hpush=uicontrol(gcf,'Style','push',...    %制作与列表框配用的按键       <29>
   'position',[0.8,0.05,0.18,0.15],'string','Apply');
set(hedit,'callback','exm1153_1(''set'')');     %编辑框输入引起回调    <31>
set(hpop,'callback','exm1153_1(''set'')');      %弹出框选择引起回调    <32>
set(hpush,'callback','exm1153_1(''set'')');     %按键引起的回调        <33>
set(gcf,'UserData',[hedit,hpop,hlist]);  %向'UserData'存放图柄         <34>
case 'set'                         %以下是回调函数                     <35>
H=get(gcf,'UserData');             %从'UserData'获取图柄               <36>
ct=get(H(1),'string');             %获得输入的字符串函数               <37>
vpop=get(H(2),'value');            %获得选项的位置标识                 <38>
vlist=get(H(3),'value');           %获得选项位置向量                   <39>
if ~isempty(ct)
   eval(ct')                       %运行从编辑文本框送入的指令
   popstr={'spring','summer','autumn','winter'};       %弹出框色图矩阵
   liststr={'grid on','box on','hidden off','axis off'};%列表框选项内容
   invstr={'grid off','box off','hidden on','axis on'};%列表框的逆指令
   colormap(eval(popstr{vpop}))       %采用弹出框所选色图
   vv=zeros(1,4);vv(vlist)=1;
   for k=1:4
      if vv(k);eval(liststr{k});else eval(invstr{k});end %按列表选项影响图形
```

```
    end
end                                                          %  〈50〉
end
```

（2）在 MATLAB 指令窗中运行 **exm1153_1** 就可获得题目所要求的图形用户界面（即图 11.4.3.3-1 无图形时的初始界面）。

【*例 11.5.3-2】目标：利用 M 函数文件创建与例 11.4.3.3-1 相同的用户界面。本例演示：如何依靠 'Tag'属性 与 findobj 指令的配合使用获取回调操作所必须的控件图柄，保证回调动作正确执行。

本例的程序可由 exm1153_1.m 做如下修改而得：

（1）删去 exm1153_1.m 的指令<34><36>。

（2）在 exm1153_1.m 的<20>和<21>行之间增添一行

```
  'Tag','H_edit',...
```

（3）在 exm1153_1.m 的<23>和<24>行之间增添一行

```
  'Tag','H_popup',...
```

（4）在 exm1153_1.m 的<27>和<28>行之间增添一行

```
  'Tag','H_list',...
```

（5）把 exm1153_1.m 的<31><32><33>条指令中的 exm1153_1 改为 exm1153_2。

（6）在 exm1153_1.m 的<35>和<37>行之间增添以下三条指令。

```
H(1)=findobj(gcf,'Tag','H_edit');
H(2)=findobj(gcf,'Tag','H_popup');
H(3)=findobj(gcf,'Tag','H_list');
```

（7）把 exm1153_1.m 的函数头修改为

```
function exm1153_2(flag)
```

（8）把修改后的文件“另存为”exm1153_2.m，就完成了新文件的编写。

（9）在 MATLAB 指令窗中运行 **exm1153_2** 就可获得题目所要求的图形用户界面。

11.6图形用户界面设计工具

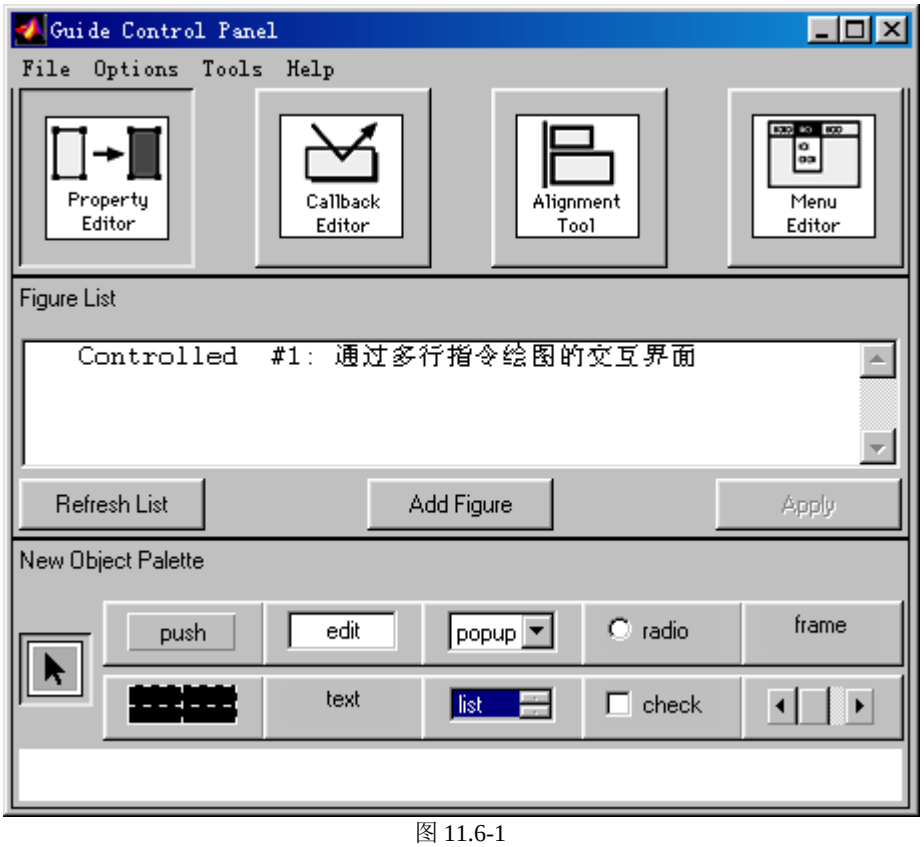

图 11.6-1

11.6.1 界面设计工具的结构和调用指令

11.6.1.1 界面设计工具的结构

11.6.1.2 图形窗的激活态和受控态

11.6.1.3 启动交互式编辑工具的指令

11.6.2交互式用户界面设计工具应用示例

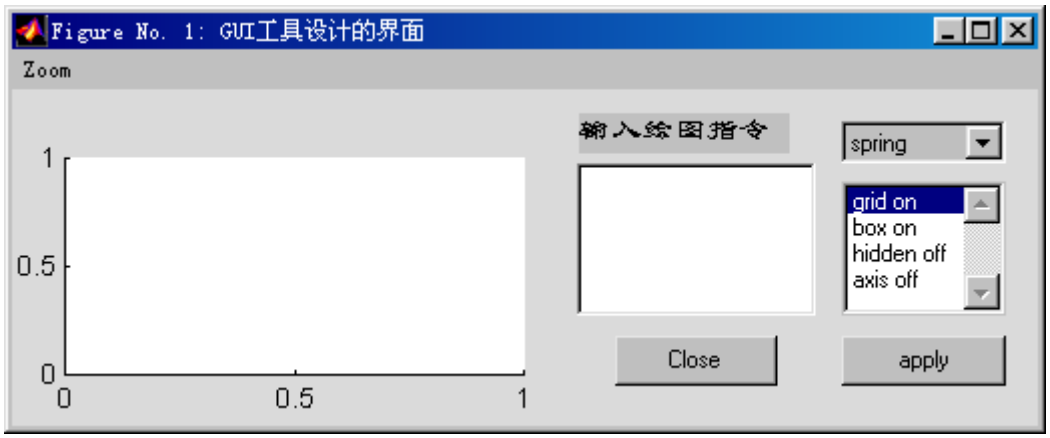

图 11.6.2-1 待制作的用户界面

11.6.2.1 工序一：窗口初始位置和大小设计

【例 11.6.2.1-1】本例演示：界面设计工具 guide 的启动和用户界面窗口初试几何制作。

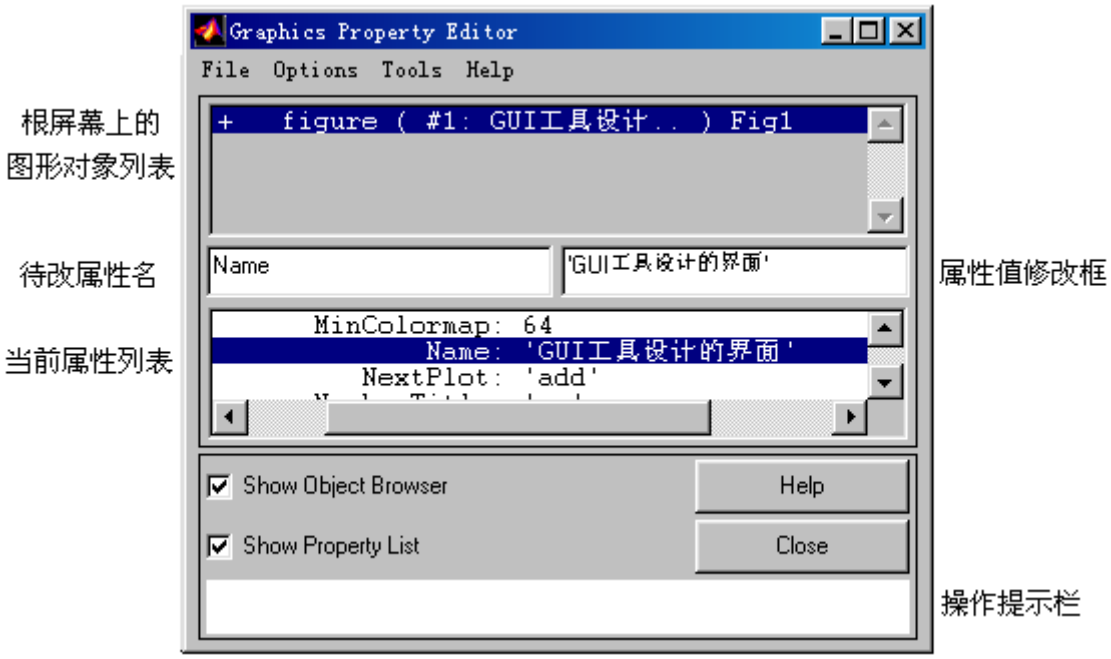

图 11.6.2.1-1 属性编辑工具界面

11.6.2.2 工序二：对象的几何布局

【例 11.6.2.2-1】整个用户界面的几何布局：“轴”、控件种类、相对位置及大致尺寸。本例演示：（A）设计工具控制面板上“新对象模块区”的图标的使用；（B）几何布局时不必太多考虑各对象的精细位置和大小。

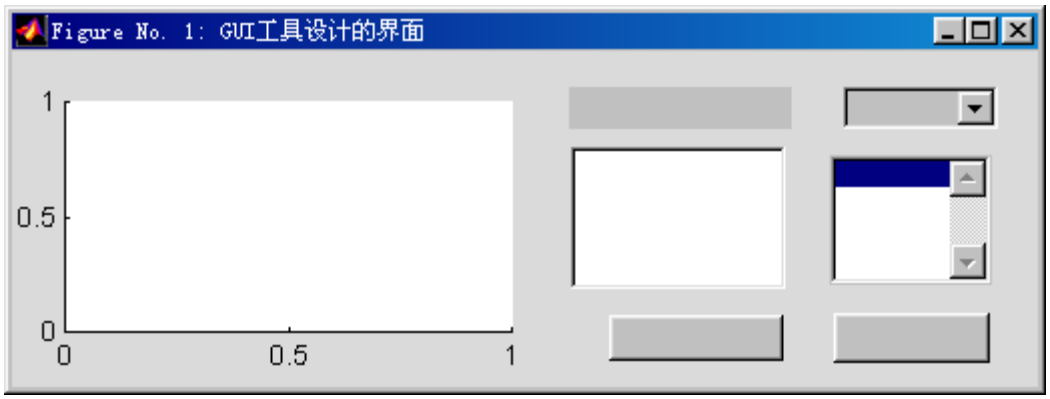

图 11.6.2.2-1 “第二道工序” 构成控件布局的界面

11.6.2.3 工序三：新建对象的属性设置

【例 11.6.2.3-1】控件关键属性的设置。本例演示：(A)属性编辑工具的使用。(B)当 'Callback' 属性值可用比较简单的 MATLAB 语句表达时，则直接填写；如果语句较多，表达复杂，那么就应采用一个待写的 M 函数名填写。本例中的回调都借助 M 函数文件实现。（C）当控件上有字符串标识时，应注意文字的对齐方式和注意字体大小，使外观上与对象大小协调。（D）控件的 'String' 属性字符串的输入格式。在这过程中，可能还要适当调整对象几何尺寸，使字符表现清晰醒目。（E） 'Units' 采用 'normalized'，使得所有新建对象随所在图形窗按比例缩放。

11.6.2.4 工序四：用户菜单的制作

用户菜单制作工序比较独立，因此该工序可前可后，也可以与“工序一”相合并。

【例 11.6.2.4-1】用户菜单的制作。本例演示：菜单编辑工具的使用。在本例中菜单引起的回调都是直接、简单的 MATLAB 语句。

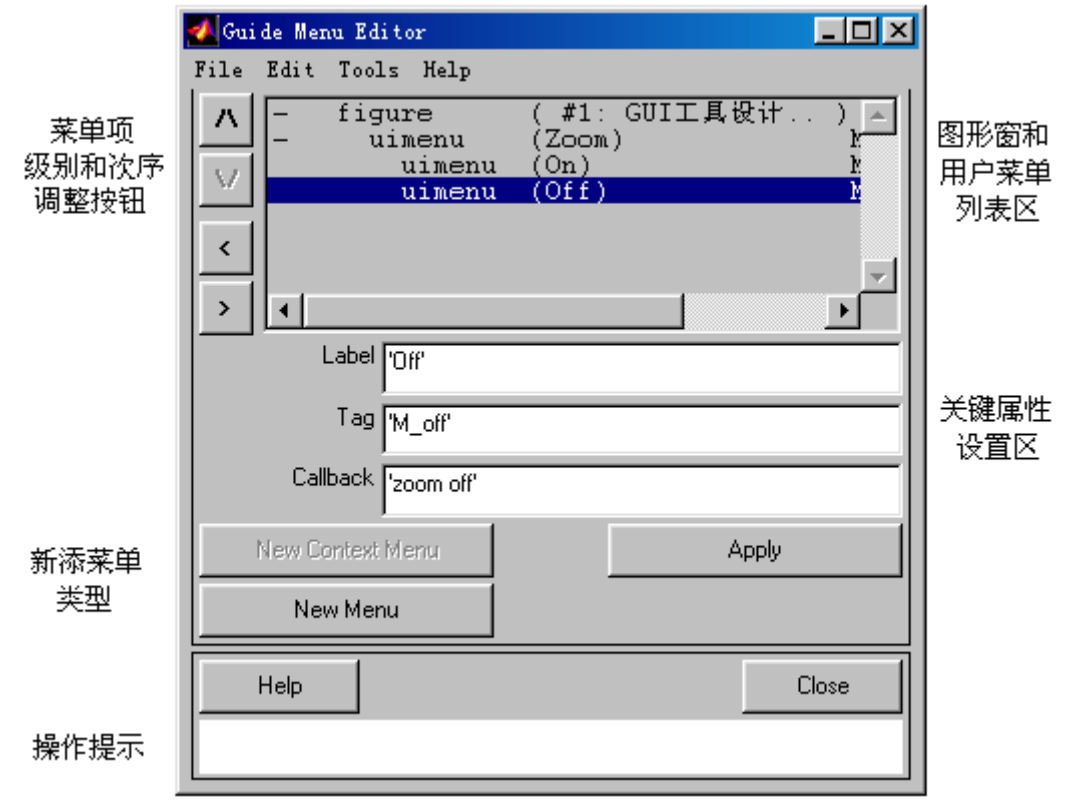

图 11.6.2.4-1 用户菜单编辑器界面

11.6.2.5 工序五：新建图形对象的齐整化

【例 11.6.2.5-1】控件的齐整化。本例演示：演示“对齐编辑工具”的使用。

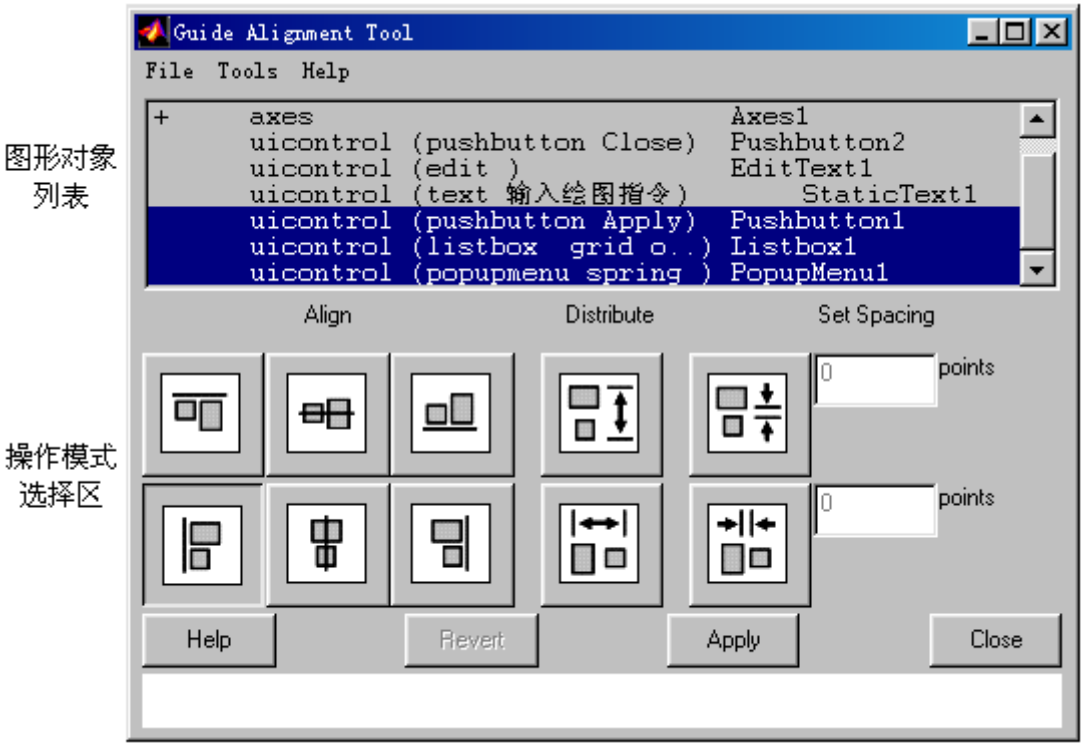

图 11.6.2.5-1

11.6.2.6　工序六：回调函数的编写

【例 11.6.2.6-1】回调函数的编写。本例演示：从处理方便出发编写回调函数。（关于回调函数的详细讨论，请看第 11.5 节的三个算例）

（1）弹出框的回调函数 Mycolormap.m

[Mycolormap.m]

```
function Mycolormap
popstr={'spring','summer','autumn','winter'};        %弹出框色图矩阵
vpop=get(findobj(gcf,'Tag','PopupMenu1'),'value');  %获得选项的位置标识
colormap(eval(popstr{vpop}))                          %采用弹出框所选色图
```

（2）列表框和“Apply”按键配合的回调函数 Myapply.m

[MYAPPLY.M]

```
function Myapply
vlist=get(findobj(gcf,'Tag','Listbox1'),'value');       %获得选项位置向量
liststr={'grid on','box on','hidden off','axis off'};  %列表框选项内容
invstr={'grid off','box off','hidden on','axis on'};    %列表框的逆指令
vv=zeros(1,4);vv(vlist)=1;
for k=1:4
   if vv(k);eval(liststr{k});else eval(invstr{k});end  %按列表选项影响图形
end
```

（3）动态编辑框的回调函数 Myedit.m

[MYEDIT.M]

```
function Myedit
ct=get(findobj(gcf,'Tag','EditText1'),'string');
eval(ct')
```

11.6.2.7　工序七：界面功能的全面测试

11.6.2.8　为读者提供的配套文件和数据

（1）机器自动生成的主控文件

[Mygui1.m]

```
function fig = Mygui1()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
% This problem is solved by saving the output as a FIG-file.%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.%
% NOTE: certain newer features in MATLAB may not have been saved in this
% M-file due to limitations of this format, which has been superseded by
% FIG-files.  Figures which have been annotated using the plot editor tools
% are incompatible with the M-file/MAT-file format, and should be saved as
% FIG-files.
load Mygui1
h0 = figure('Units','normalized', ...
   'Color',[0.8553876799929505 0.8553876799929505 0.8553876799929505], ...
```

```
    'Colormap',mat0, ...
    'FileName','F:\99\m5\Mygui1.m', ...
    'MenuBar','none', ...
    'Name','GUI工具设计的界面', ...
    'PaperPosition',[18 180 575.9999999999999 432], ...
    'PaperUnits','points', ...
    'Position',[0.1484375 0.5291666666666667 0.8 0.35], ...
    'Tag','Fig1', ...
    'ToolBar','none');
h1 = uimenu('Parent',h0, ...
    'Label','Zoom', ...
    'Tag','M_Z');
h2 = uimenu('Parent',h1, ...
    'Callback','zoom on', ...
    'Label','On', ...
    'Tag','M_Zon');
h2 = uimenu('Parent',h1, ...
    'Callback','zoom off', ...
    'Label','Off', ...
    'Tag','M_Zoff');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.7529411764706 0.7529411764706 0.7529411764706], ...
    'Callback','Mycolormap', ...
    'ListboxTop',0, ...
    'Position',[0.80859375 0.773809523809524 0.16 0.130952380952381], ...
    'String',['spring ';'summer ';'autumn ';'winter '], ...
    'Style','popupmenu', ...
    'Tag','PopupMenu1', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[1 1 1], ...
    'Max',2, ...
    'Position',[0.80859375 0.327380952380952 0.16 0.398809523809524], ...
    'String',['grid on   ';'box on    ';'hidden off';'axis off  '], ...
    'Style','listbox', ...
    'Tag','Listbox1', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[.752941176470588 .752941176470588 .752941176470588], ...
    'Callback','Myapply', ...
    'ListboxTop',0, ...
    'Position',[0.80859375 0.12 0.16 0.15], ...
    'String','apply', ...
    'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[1 1 1], ...
```

```
    'Callback','Myedit', ...
    'FontName','Times New Roman', ...
    'FontSize',10, ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Max',2, ...
    'Position',[.55078125 .3273809523809524 .232421875 .4523809523809523], ...
    'Style','edit', ...
    'Tag','EditText1');
h1 = axes('Parent',h0, ...
    'CameraUpVector',[0 1 0], ...
    'CameraUpVectorMode','manual', ...
    'Color',[1 1 1], ...
    'ColorOrder',mat1, ...
    'Position',[0.05 0.15 0.45 0.65], ...
    'Tag','Axes1', ...
    'XColor',[0 0 0], ...
    'YColor',[0 0 0], ...
    'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[0.4978165938864628 -0.2222222222222221 9.160254037844386], ...
    'Tag','Axes1Text4', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[-0.1353711790393013 0.4907407407407408 9.160254037844386], ...
    'Rotation',90, ...
    'Tag','Axes1Text3', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Position',mat2, ...
    'Tag','Axes1Text2', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[0.4978165938864628 1.064814814814815 9.160254037844386], ...
    'Tag','Axes1Text1', ...
```

```
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'FontName','隶书', ...
    'FontSize',13, ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[0.55 0.8 0.2 0.12], ...
    'String','输入绘图指令', ...
    'Style','text', ...
    'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[.752941176470588 .752941176470588 .752941176470588], ...
    'Callback','close(gcbf)', ...
    'ListboxTop',0, ...
    'Position',[0.5869921874999999 0.12 0.16 0.15], ...
    'String','Close', ...
    'Tag','Pushbutton2');
if nargout > 0, fig = h0; end
```

（2）配套数据文件

[MYGUIZZY.M]

```
function Myguizzy
%假如Mygui1.m所在目录不是d:\matbook5\mdisk ，那么第<10>条就应做相应的改变。
%一定要保证本函数生成的 Mygui1.mat 与 Mygui1.m 在同一目录。
mat0=jet(64);
mat1=[    0         0    1.0000
          0    0.5000         0
    1.0000         0         0
         0    0.7500    0.7500
    0.7500         0    0.7500
    0.7500    0.7500         0
    0.2500    0.2500    0.2500];
mat2=[-0.1179    1.3056    9.1603];
save d:\matbook5\mdisk\Mygui1                                    %<10>
```

（3）如何利用本节所提供的文件产生图 11.6.2-1 所示的界面

- 把本节提供的 Mygui1.m , Myguizzy.m , Mycolormap.m , Myapply.m , Myedit.m 五个文件放在 MATLAB 的搜索路径上。
- 先运行 Myguizzy.m ， 创建数据文件 Mygui1.mat 。
- 运行 Mygui1.m ， 就可得到符合要求的界面。

第21章　创建图形用户界面

用户界面是人，即用户与计算机或计算机程序的接触点或交互方式，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和文本，若有扬声器还可产生声音。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。

图形用户界面或GUI是包含图形对象，如：窗口、图标、菜单和文本的用户界面。以某种方式选择或激活这些对象，通常引起动作或发生变化。最常见的激活方法是用鼠标或其它点击设备去控制屏幕上的鼠标指针的运动。按下鼠标按钮，标志着对象的选择或其它动作。

与上一章讨论MATLAB句柄图形功能的相同方式，它让用户按规定设计MATLAB显示信息的方法，本章所描述的图形用户界面的功能，它让用户定制用户与MATLAB的交互方式。命令窗口不是唯一与MATLAB的交互方式。

本章将说明图形句柄**uicontrol** 和**uimenu**对象的使用，把图形界面加到MATLAB的函数和M文件。**uimenu**对象能在图形窗口中产生下拉式菜单和子菜单。**uicontrol**对象能建立如按钮，滚动条，弹出式菜单以及文本框等对象。

MATLAB在**demo**命令中包含了GUI功能的极好例子。

```
>> demo
```

研究该命令，以了解**uimenu**和**uicontrol**如何给MATLAB函数提供交互输入。

21.1　谁创建图形界面GUI?为什么?

在运行了demo例子后，很可能会问“为什么要在MATLAB中建立一个*GUI*？”这是一个很好的问题，简单的回答是可能并不需要．使用MATLAB来分析数据，求解问题，绘制结果的绝大多数的人，并不会发现GUI工具很有用。

但另一方面，GUI可以在MATLAB中生成非常有效的工具和应用程序，或是建立演示工作的交互式界面。

生成用户图形界面的最常见的理由：

编写一个需多次反复使用的实用函数，菜单、按钮、文本框作为输入方法具有意义；或
编写函数或开发应用程序供别人使用；或
创建一个过程、技术或分析方法的交互式示例；或
认为GUI的简洁，性能良好，并且想实践一下。

许多基于GUI的工具函数包含在精通MATLAB工具箱中，将在后续章节进行讨论。其它由MATLAB用户编制的工具和实用程序装入MATLAB的GUI函数。工具的大多数可在**Mathworks** 匿名**FTP**节点和其它资源中获得。

在我们开始讨论之前，记住对“句柄图形”的理解是设计和实现GUI的先决条件，如果你跳过了前一章，现在应重新回去阅读。

21.2　GUI对象层次结构

正如我们在上一章所展示的那样，由图形命令生成的每一事物是一个图形对象。图形对象不仅包括**uimenu**和**uicontrol**对象，而且还包括图形、坐标轴和他们的子对象。让我们从另一个角度来看这一层次结构。计算机的屏幕本身是根结点，图形是根对象的子对象，坐标轴，**uimenu** ，**uicontrol**是图形的子对象。

根可以包括多个图形，每个图形含有一组或多组坐标轴以及其子对象，每个图形也可以有一个或多个与坐标轴无关的**uimenu**和**uicontrol**。虽然**uicontrol**对象无子对象结点，但他们确实具有多种类型。**uimenu**对象常将其它的**uimenu**对象作为其子对象。

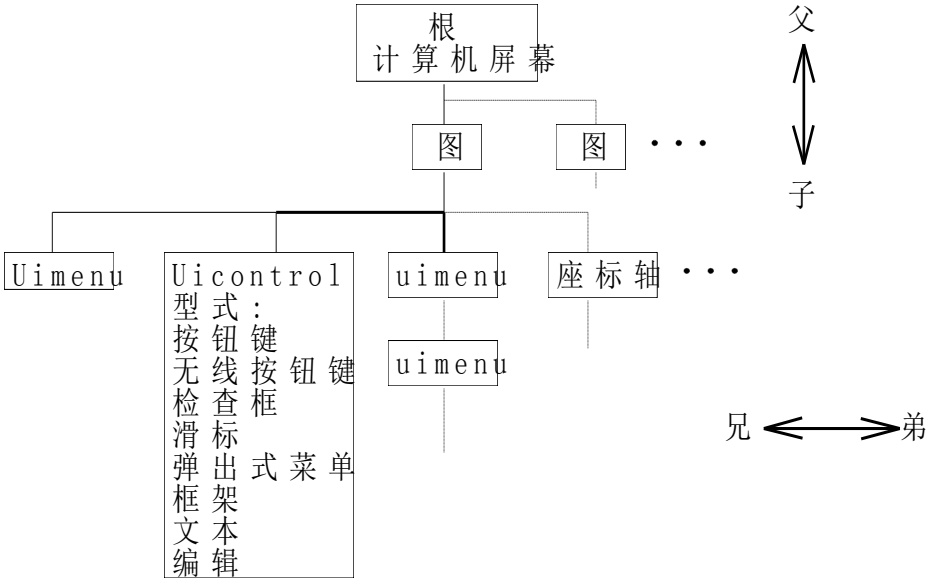

图21.1 GUI对象层次结构图

运行MATLAB的不同型号的计算机或平台上，产生不同的图形显示。Unix工作站使用不同的X Window系统，具有几个窗口程序，如**mwn**或**twm**以控制显示的布局。PC机靠Microsoft Windows或Windows NT进行窗口管理，Macintosh计算机用Macintosh工具箱程序作窗口。虽然在各种平台上，显示看起来有很大的不同，但在很多的情况下，句柄图形的编码是一致的。MATLAB在内部处理平台和窗口系统的差别。体现句柄图形例程的函数，包括应用**uimenu**和**uicontrol**对象的函数，通常运行在所有平台。存在已知差异的地方将在本章后面给出。

21.3 菜单

在每一个窗口系统中使用菜单让用户选择命令和选项。通常在显示屏或窗口的顶部有一菜单条。移动鼠标指针到菜单标志上按下鼠标按键，顶层菜单就被选中，一列菜单项就从菜单标志拉下来。这种款式菜单就叫下拉式菜单。按下鼠标将指针移动至菜单项并松开鼠标，则完成菜单项的选择。MS-Windows 和一些X Window系统平台还提供另一种选择菜单的方法。在顶层菜单上按下并松开鼠标，或称单击鼠标，则打开下拉菜单。然后，移动鼠标指针至下拉菜单项再次单击鼠标，就选择菜单项。在下拉菜单中选择一项就引起动作的发生。

一个菜单项还可用自己的菜单项列表而作为子菜单。子菜单项在子菜单的标志右边显示小三角或箭头以表示菜单还有更多子菜单项可供选择。如果子菜单的菜单项被选择，另一个具有更多菜单项的菜单显示在此菜单的右边的下拉菜单中。有时这种菜单称之行走菜单。选中其中一个菜单项也引起某些动作的产生。

子菜单可以嵌套，但层次的数目受到窗口系统及有用资源的限制。

菜单的布置

Macintosh在显示屏的顶部使用包含下拉菜单标题的菜单条。选择图形窗口时，菜单条变化以反映激活的图形窗中可利用的选项。标准的Macintosh菜单标题包括**Apple**、 **File**、

Edit、**Window** 和**Help** 或**Balloon Help**。由**uimenu**所加的菜单标题放在**Window**和**Help**之间。如果想从菜单条中删去**File**、 **Edit**、和 **Window**菜单标题，可以使用**Set**命令。

```
>> set(Hf_fig1，‘ Menubar ‘，‘ none ‘)
```

Apple和**Help**不可从菜单条中删除。同样，标准的菜单是用以下的命令恢复：

```
>> set(Hf_fig1，‘Menubar‘，‘figure‘)
```

在Microsoft窗口系统下，菜单条位于图形窗口的顶部。每个图形窗口有自己的菜单条，它包含**File**，**Edit**，**Window**和**Help**标题。由**uimenu**所加的菜单标题放在**Help**之后。可以使用与上面相同的**Set**命令从菜单条中删去或恢复所有的标准菜单。

在X Window系统工作站上没有MATLAB标准图形窗口菜单的标题。窗口管理器可将菜单条放置在屏幕上每一个窗口上端，但这些菜单条与MATLAB菜单无关。当建立第一个**uimenu**对象时，MATLAB 在图形窗口的顶部边缘生成自己的菜单条。

建立菜单和子菜单

我们采用函数**uimenu**建立菜单项。**uimenu**的句法与其他对象创建函数相似。如，

```
>> Hm_1=uimenu(Hx_parent，‘PropertyName‘，PropertyValue，...)
```

其中**Hm_1**是由**uimenu**生成的菜单项的句柄，通过设定**uimenu**对象的属性值‘**PropertyName**’，**PropertyValue**这对命令定义了菜单特性；**Hx_parent**是缺省的父辈对象的句柄，必须是图形和**uimenu**对象。

uimenu对象中最重要的属性是' Label ' 和 ' Callback ' 。 ' Label ' **属性值是菜单条和下拉菜单项上的文本字符串，以确认菜单项。**' Callback ' **属性值是MATLAB字符串，当选中菜单项时，它传给**eval**，用以执行。**

菜单举例

下面的例子用函数**uimenu**将简单菜单加到当前的图形窗口中。这里提出的例子说明如何只用几个MATLAB命令来建立工作菜单。后面的例子将详细地讨论**uimenu**的命令和属性。

下例用两个下拉菜单将菜单条加到当前窗口中。首先，建立名为Example的顶部菜单输入。

```
>> Hm_ex=uimenu(gcf，‘Label‘，‘Example‘);
```

在此菜单下有两个菜单项。第一项标志为Grid，切换坐标轴格栅的状态。

```
>> Hm_exgrid=uimenu(Hm_ex，‘Label‘，‘Grid‘，‘Callback‘，‘Grid‘);
```

注意，句柄Hm_ex是用于与上层菜单相关联的。这项**uimenu**按**eval**的要求出现在上层菜单之下。还需注意的是属性‘Callback‘的值，它是一个带引号的字符串。

Example下的第二项标志为**View**，并带有子菜单。

```
>> Hm_exview=uimenu(Hm_ex，‘Label‘，‘View‘);
```

View菜单有两项选择2-D和3-D视图。

```
>> Hm_ex2d=uimenu(Hm_exview，‘Label‘，‘2-D‘，‘Callback‘，‘view(2)‘);
```

```
>> Hm_ex3d=uimenu(Hm_exview，‘Label‘，‘3-D‘，‘Callback‘，‘View(3)‘);
```

注意以上这些是**View**的子菜单，因为它们指定**Hm_exview**作为其父对象。
现在，将第二个顶层菜单加到标题为**Close**的菜单条中。

```
>> Hm_ex=uimenu(gcf，‘Label‘，‘Close‘);
```

由该顶层菜单Close加入了两个菜单项。第一项关闭图形窗口，第二项使图形窗口打开，但去掉用户菜单。

```
>> Hm_clfig=uimenu(Hm_close，‘Label‘，‘Close Figure‘，‘Callback‘，‘Close‘);

>> Hm_clmenu=uimenu(Hm_close，‘Label‘，‘Remove Menu‘，...
                ‘Callback‘，‘delete(Hm_ex)； delete(Hm_ex)； drawnow‘);
```

在精通MATLAB工具箱的脚本文件**mmenu1.m**中含有上面的例子。所以你可以运行**mmenu1.m**来验证上例。

菜单属性

如上所示，同句柄图形函数一样，在建立图形对象时可定义**uimenu**属性，或用set改变属性。所有可设定的属性，包括标题、菜单颜色、甚至回调字符串都可以用set来改变。这种功能十分便于迅速地定制菜单和属性。

表21.1列出了MATLAB 4.2版本中的**uimenu**对象的属性及其属性值。带有*的属性是非文件式的，使用时需加小心。在括号{}内的属性值是缺省值。

表21.1

Uimenu 对象的属性	
Accelerator	指定菜单项等价的按键或快捷键。对于X-windows，按键顺序是**Control**- 字符；Macintosh系统，按键顺序是**Command** - 字符或# - 字符
BackgroundColor	**uimenu**背景色,是一个3元素的RGB向量或MATLAB预先定义的颜色名称。缺省的背景色是**亮灰**色
Callback	MATLAB回调字符串，选择菜单项时，回调串传给函数**eval**；初始值为空矩阵
Checked	被选项的校验标记
on:	校验标记出现在所选项的旁边
{**off**}:	校验标记不显示
Enable	菜单使能状态
{**on**}:	菜单项使能。选择菜单项能将**Callback**字符串传给**eval**
off:	菜单项不使能，菜单标志变灰。选择菜单项不起任何作用。
ForegroundColor	**uimenu**前景（文本）色,是一个三元素的RGB向量或MATLAB预先定义的颜色名称。缺省的前景色是**黑色**
Label	含有菜单项标志的文本串。在PC系统中，标记中前面有‘&‘，定义了快捷键，它由**Alt - 字符**激活

Position		**uimenu**对象的相对位置。顶层菜单从左到右编号，子菜单从上至下编号
Separator		分割符 - 线模式
	on:	分割线在菜单项之上
	{**off**}:	不画分割线
*Visible		**uimenu**对象的可视性
	{**on**}:	**uimenu**对象在屏幕上可见
	off:	**uimenu**对象不可见
ButtonDownFcn		当对象被选择时，MATLAB的回调串传给函数**eval**。初始值为空矩阵。
Children		其它**uimenu**对象的句柄。
Clipping		限幅模式
	{**on**}:	对uimenu对象无效果
	off:	对uimenu对象无效果
DestroyFcn		仅用于Macintosh 4.2 版本。没有文本说明。
Interrruptible		指明**ButtonDownFcn**和**CallBack**串可否中断
	{**no**}:	回调不可中断
	yes:	回调串可中断
Parent		父对象的句柄；如果**uimenu**对象是顶层菜单，则为图形对象；若**uimenu**是子菜单，则为父的**uimenu**对象句柄
*Select		值为**[on|off]**
*Tag		文本串
Type		只读对象辩识串，通常为**uimenu**
UserData		用户指定的数据。可以是矩阵，字符串等等
Visible		**uimenu**对象的可视性
	{**on**}:	**uimenu**对象在屏幕上可见
	off:	**uimenu**对象不可见

属性值仅仅定义了**uimenu**对象的性质并且控制菜单如何显示；它们也决定了选择菜单项所引起的动作。其中某些性质将在下面更详细地讨论。

菜单快捷键

' **Label** ' 属性义定了出现在菜单或菜单项中的标志。它也可以用来定义Microsoft Windows系统的快捷键：标志字符串中，在所需字符前加上&，例如：

```
>> Hm_top=uimenu( ' Label ' ,  ' Example ' );

>> uimenu(Hm_top,  ' Label ' ,  ' &Grid ' ,  ' CallBack ' ,  ' grid ' );
```

它定义了键盘上**G**为快捷键。菜单项标志将以Grid形式出现在菜单上。为激活快捷键，在选择图形窗口时按Alt键并按下**G**键。快捷键不一定是字符串的第一字符。下例中**R**为快捷键：

```
>> uimenu(Hm_top,  ' Label ' ,  ' G & rid ' ,  ' CallBack ' ,  ' grid ' );
```

则标志以**Grid**形式出现在菜单上。

Macintosh平台用 ' **Accelerator** ' 属性而不是 ' **Label** ' 来定义快捷键。在Macintosh

>> uimenu(Hm_top，‘Label‘，‘Grid‘，‘Accelerator‘，‘G‘，‘CallBack‘，‘grid‘)；

定义**G**为快捷键，菜单标志以**Grid#G**的形式出现。为激活快捷键，选择图形窗口时，按**Command**键或#键并按下**G**键。

在Macintosh上不可为顶层菜单定义快捷键。另外，已经定义在标准Macintosh上的快捷键，如**Command C**或**# C**不撤除标准Macintosh菜单，就不能复制。

在X-Window 系统中定义和使用快捷键与Macintosh相似，但仍存在某些差异。同Macintosh一样，用 **' Accelerator '** 属性而不是 **' Label '** 属性来定义快捷键。但快捷键在菜单上显示不同。例如，

>> uimenu(Hm_top，‘Label‘，‘Grid‘，‘Accelerator‘，‘G‘，‘CallBack‘，‘grid‘)；

定义字母**G**为快捷键。菜单标志常以Gr**id<Ctrl>-G**出现在菜单上。未使用快捷键，需按**Control**键并按下**G**键。同Macintosh一样，不可为顶层菜单定义快捷键。

虽然我们没有对此验证，但MATLAB用户手册指出当给定相同命令时，用X的某些工作站动作不一样。如果顶层菜单标志中有带下划线的字符，就可按**Meta**键并按下下划字符键来选择菜单。如果子菜单项标志中含有带下划线的字符，就按下该字符键来选择菜单项。请参阅键盘使用说明，以为系统确定合适的Meta键。

菜单的外观

影响菜单的布置和外观的三个属性为 **' Position '**，**' Checked '**，和 **' Separator '** .**uimenu**对象的 **' Position** 属性值是一个整数，它定义了相对于其它菜单和菜单项的位置。在生成菜单时，设定 **' Position '** 属性。菜单条的最左端的菜单条和下拉菜单中的上端菜单项处在位置1。

设置 ‘ **Position '** 属性可以重新排列菜单位置。考虑下面的例子：

```
>> Hm_1=uimenu(‘Label‘，‘first‘)；        % Create two menus

>>  Hm_2=uimenu(‘Label‘，‘Second‘)；

>> get(Hm_1，‘Position‘)                  % Check the locations
ans=
     1

>> get(Hm_2，‘Position‘)
ans=
     2

>> set(Hm_2，‘Position‘，1)                  % Change menu order

>> get(Hm_1，‘Position‘)                  % check the locations

ans=
     2

>> get(Hm_2，‘Position‘)

ans=
     1
```

注意，当一个**uimenu**的 **' Position '** 属性改变，就将移动其它**uimenu**以适应此变化，它们的 **' Position '** 属性均更新。子菜单中菜单选项的编号以同样的方式重新排列。

属性 ' **Checked** ' 的值使校验标记出现在菜单项标志的左边。缺省值为 ' **off** ' 。命令

```
>> set(Hm_item，‘Checked‘，‘on‘)
```

使校验标记出现在**Hm_item uimenu**标志的旁边。对于创建代表属性的菜单项，该命令十分有用。例如，

```
>>Hm_top=uimenu(‘Label‘，‘Example‘);
>>Hm_box=uimenu(Hm_top，‘Label‘，‘Axis Box‘，...
‘CallBack‘，[...
        ‘if strcmp(get(fca，‘‘Box‘‘)，‘‘on‘‘)，‘，...
        ‘seet(gca，‘‘Box‘‘，‘‘off‘‘)，‘，...
        set(Hm_box，‘‘Checked‘‘，...off‘‘)，‘，...
        ‘else，‘，...
        ‘set(gca，‘‘Box‘‘，‘‘on‘‘)，‘，...
        ‘set(Hm_box，‘‘Checked‘‘，‘‘on‘‘)，‘，...‘end
‘]);
```

建立了以**Axis Box**为标志的下拉菜单项。当选中该项时，就运行回调字符串所表示的命令。回调字符串确定了当前坐标轴的 ' **Box** ' 属性值，并适当地设定坐标轴的 ‘ **Box** ' 属性及**uimenu**的 ' **Checked** ' 属性。该例子可从精通MATLAB工具箱的M脚本文件**mmenu2.m**中得到。

可以通过改变**uimenu**的 ' **Label** ' 属性以反映菜单项当前的状态。下面的例子（在**mmenu3.m**中)改变了菜单项标志本身，而不是加一个校验标记：

```
>> Hm_top=uimenu(‘Label‘，‘Example‘);

>> Hm_box+uimenu(Hm_top，‘Label‘，‘Axis Box‘，...
        ‘CallBack‘，[...
        ‘if strcmp(get(gca，‘‘Box‘‘)，‘‘on‘‘)，‘，...
        ‘set(gca，‘‘Box‘‘，‘‘off‘‘)，‘，...
        set(Hm_box，‘‘Label‘‘，‘‘Set Box On‘‘)，‘，...
        ‘else，‘，...
        ‘set(gca，‘‘Box‘‘，‘‘on‘‘)，‘，...
        ‘set(Hm_box，‘‘Label‘‘，‘‘Set box Off‘‘)，‘，...
        ‘end‘]);
```

使用 ' **Separator** ' 属性可将下拉菜单分成局部组。如果一个**uimenu**项的 ' **Separator** ' 属性是 ' **on** '，则在下拉菜单中显示此项时，此项的上端有一条水平线将其与前面的菜单项隔开。缺省值是 ' **off** '，但在菜单生成时可以改变，

```
>> Hm_box=uimenu(Hm_top，‘Label‘，‘Box‘，‘Seperator‘，‘on‘);
```

或是在以后使用**set**命令

```
>> set(Hm_box，‘Separator‘，‘on‘);
```

顶层菜单忽略 ' **Seprator** '的属性值。

使用 **'Seperator'** 属性可将下拉菜单项分成若干逻辑组。如果对**uimenu**项 **'Seperator'** 属性设置为**'on'**，用在一条其上面的水平线来表现该项，将其与前面的菜单项形象地区分开。缺省值为 **'off'**，但在生成菜单时可以改变，

```
>> Hm_box=uimenu(Hm_top，‘Label‘，‘Box‘，‘Seperator‘.‘on‘)
```

或是在以后使用**set**命令：

```
>> set(Hm_box.‘Seperator‘，‘on‘)
```

顶层菜单忽略 **'Seperator'** 属性值。

颜色控制

uimenu对象可设置两个颜色属性。**'BackGroundColor'** 属性控制填充菜单背景的颜色。缺省值是浅灰。另一颜色属性为 **'ForeGroundColor'**，它确定菜单项文本的颜色，缺省值是黑色。

颜色属性同样能很好地用于顶层菜单条和下拉菜单。颜色可以用来表示状态信息或简单加上菜单的特 色。例如，挑选线段颜色。在子菜单中，各菜单项的背景可以填充合适的色彩。

```
>> Hm_green=uimenu(Hm_color，'Label'，'Green'，'BackgroundColor'，'g'，...
                   'Callback'，'set(Hl_line，'Color'，'g')');
```

菜单项去能

改变对象**uimenu**的 **'Enable'** 值或 **'Visible'** 属性可使菜单项暂时去能。**'Enable'** 属性通常设为 **'on'**。当 **'Enable'** 属性'设为 **'off'** 时，标志字符串变灰，菜单项去能。在这种状态下，菜单项保持可见但不能被选择。此属性可用来将不恰当的菜单选择去能。

下面的例子(**mmenu4.m**)说明了用两个菜单项和 **'Enable'** 属性来设定坐标轴的 **'Box'** 属性的另一种方法。

```
>> Hm_top = uimenu('Label'，Example');

>> Hm_boxon = uimenu(Hm_top，'Label'，'Set Box On'...'CallBack'，[...
                'set(gca，''Box''，''on'')，'，...
                'set(Hm_boxon，''Enable''，''off'')，'，...
                'set(Hm_boxoff，''Enable''，''Enable''，''on'')']);

>> Hm_boxoff = uimenu(Hm_top，'Label'，Set Box Off'，...'Enable'，'off'，...
'CallBack'，[...
                'set(gca，''Box''，''off'')，'，...
                'set(Hm_boxon，''Enable''，''on'')，'，...
                'set(Hm_boxoff，''Enable''，''off'')']);
```

设定**'Visible'** 属性为 **'off'，** 可将菜单项完全隐藏。菜单项象是从屏幕中消失，而其它菜单项改变了在显示器上的位置以填补由当前不可见菜单造成的空隙。然而，不可见的菜单仍然存在，而且 **uimenu** 对象的 **'Position'** 属性值也不改变。当属性 **'Visible'** 又重新设为 **'on'** 时，菜单项重新出现在正常的位置。

这个性质可以用来暂时地撤消一个菜单。下面的例子(**mmenu5.m**)建立了两个顶层菜单和两个菜单项。

```
>> Hm_control = uimenu('Label', 'Control');

>> Hm_extra = uimenu('Label', 'Extra');

>> Hm_limit = uimenu(Hm_control, 'Label', 'Limited Menus', ...
                    'CallBack', 'set(Hm_extra, ''Viseble'', ''off'')');

>> Hm_full = uimenu(Hm_control, 'Label', 'Full Menus', ...
                 'CallBack', 'set(Hm_extra, ''Visible'', ''on'')');
```

当选择了**Limited Menus**项时，**Extra**菜单就从菜单条中消失。当选择了**Full Menus**项时，**Extra**菜单又重新显示在原来的位置的菜单条上。

回调属性

'**Callback**'属性值是一个MATLAB字符串，MATLAB将它传给函数**eval**并在**命令窗口工作空间**执行。它对于函数M文件有重要的隐含意义，我们将在本章后面继续讨论这一属性。

因为'**Callback**'属性必须是字符串，所以在字符号内多重MATLAB命令、后续行以及字符串都会使必需的句法变得十分复杂。如果有不止一个命令要执行，命令间必须适当地分隔开来。例如，

```
>>uimenu('Label', 'Test', 'CallBack', 'grid on; set(gca, ''Box'', ''on'')');
```

把一个字符串传给**eval**，使命令

```
>> grid on; set(gca, 'Box', 'on')
```

在命令窗口工作空间中执行。这是合法的句法，因为命令用逗号或分号隔开，多重命令可输入到同一命令行中。在定义回调函数时，也遵循MATLAB规定，即在已引用的字符串内，用两个单引号来表示单引号。

字符串可以串接起来生成一个合法MATLAB字符串，只要把它们括在方括号中。

```
>>uimenu('Label', 'Test', 'CallBack', ['grid on, ', ' set(gca, ''Box'', ''on'')']);
```

注意字符串'**grid on**'含有所需的逗号以分隔两个命令。

如果使用了续行号，上述命令可写为

```
>>uimenu('Label', 'Test', ...
        'CallBack', [...
        'grid on, ', ...
        ' set(gca, ''Box'', ''on'')'...
        ]);
```

上例中命令行被分隔，每行的末尾加上了三个句号表示命令的继续。注意到上列单行的所有元素都被保留，包括字符串分隔命令的逗号。在'grid on, …'行中最后引号后的逗号是可选的；下一行开始的空格起相同的作用。欲了解详情，请参阅前面关于建立行向量的章节。

如果引号、逗号和括号不正确输入，MATLAB将给出警告；但在复杂回调字符串中很难寻找错误的。为了使错误最少，对包含MATLAB语句的回调字符串请记住以下的一些规则：

把整个回调字符串括在方括号中，不要忘记最后的右括号'）'。
把各语句括上单引号。
已引用的字符串内，要用双引号。如：**'quoted'：'a ' 'quoted' ' string'；'Quote ' 'a ' ' 'quoted' ' ' string' 'now'。**在引号后要用逗号或空格结尾。
除了最后一句，各语句在**引号内**要以逗号或分号结尾；在**引号后**要用逗号或空格结尾。
有后续行的各行要以三个句号(...)结尾。

前面的例子之一**mmenu4.m**是所涉及的回调字符串句法的很好说明。

```
>> Hm_top = unimenu(Label ' ， ' Example ' )；

>> HM_boxon = uimenu(Hm_top，...
                ' Label ' ， ' Set ' Box on ' ，...
                ' CallBack ' ，[...
                   ' set(gca，"Box"，"on")， ' ，...
                   set(Hm_boxon，"Enable"，"off")， ' ，...
                   set(Hm_boxoff，"Enable"，"on") ' ]；

>> Hm_boxoff = uimenu(Hm_top，...
                  ' Label ' ， ' Set Box off ' ，...
                  ' Enable ' ， ' off ' ' ， ' ，...
                  ' CallBack ' ，[...
                     ' set(gca，"Box"，"off"， ' ，...
                     ' set(Hm_boxon，"Enable"，"on")， ' ，...
                     ' set(Hm_boxoff"Enable"，"off") ' ]；
```

上例中还引出了关于回调函数另一个重点，在变量**Hm_boxoff**定义之前，在回调串中用**Hm_boxoff**替代**Hm_boxon**。因为回调串只是一个字符串，MATLAB 不会给出警告，而且仅在**uimenu**被激活并将字符串传给**eval**时才由MATLAB执行。它隐含有函数M文件的设计和测试，这将在本章后面讨论。

M文件的举例

下例将演示一组简单菜单的生成。该例子包含在精通MATLAB工具箱的函数M文件**mmenus**中。正如下面所示的那样，这个函数文件被分隔成了若干块，以便于讨论函数的各个方面。

首先，定义一个函数并在当前的图形中用顶层**Line**菜单建立菜单条，该菜单分别含有三个子菜单：**Line Style，Line Width，Line Color**。

```
function mmenus()
%  MMENUS Simple menu example.
%  MMENUS uses waitforbuttonpress and gco in callback strings
%  to let the user make a menu selection and then select an object
%  by clicking on it with the mouse. The callback strings then use
%  the set function to apply the property value to the selected
%  object.
%  Copyright (c) 1996 by Prentice-Hall，Inc.
```

```
Hm_line = uimenu(gcf，'label'.'Line')；
Hm_lstyle = uimenu(Hm_line，'label'，'Line Style')；
Hm_lwidth = uimenu(Hm_line，'label'，'Line width')；
Hm_lcolor = uimenu(Hm_'label，'Line Color')；
```

其次，使用**waitforbuttonpress**和**gco**得到当前对象的句柄，确认它为一个线对象，并采用适当的'**Linestyle**'值。注意这些菜单项句柄以后不再使用，所以它们不必保存。

```
uimenu(Hm_lstyle，'Label'，Solid'，...
    'CallBack'，('waitforbuttonpress；'，...
        'if get(gco，"type")=="line"，'...
          'set(gco，"LineStyle"，"-")，'...
        'end'])；

uimenu(Hm_lstyle，'Label'，Dotted'，...
    'Callback'，['waitforbuttonpress；'，...
        'if get(gco，"Type")=="line"，'，...
          'set(gco，"LineStyle"，"，"：")，'...
        'end'])；

uimenu(Hm_lstyle，'Label'，Dashed'，...
    'Callback'，['waitforbuttonpress；'，...
        'if get(gco，"Type")=="line"，'，...
          'set(gco，"LineStyle"，"-"：")，'...
        'end'])；

uimenu(Hm_lstyle，'Label'，DashDot'，...
    'Callback'，['waitforbuttonpress；'，...
        'if get(gco，"Type")=="line"，'，...
          'set(gco，"LineStyle"，"-"：")，'...
        'end'])；
```

现在，对**Line width**子菜单项也做同样的操作：

```
uimenu(Hm_lwitdth，'Label'，'Default'，...
    'CallBack'，['witforbuttonpress；'，...
        'if get(gco，"Type")=="line"，'，...
          'set(gco，"LineWidth"，0.5)，'，...
        'end'])；

uimenu(Hm_lwitdth，'Label'，'Thick'，...
    'CallBack'，['witforbuttonpress；'，...
        'if get(gco，"Type")=="line"，'，...
          'set(gco，"LineWidth"，2.0)，'，...
        'end'])；

uimenu(Hm_lwitdth，'Label'，'Thicker'，...
    'CallBack'，['witforbuttonpress；'，...
        'if get(gco，"Type")=="line"，'，...
          'set(gco，"LineWidth"，3.0)，'，...
```

```
        'end']);

    uimenu(Hm_lwitdth, 'Label', 'Thickest', ...
        'CallBack', ['witforbuttonpress;', ...
            'if get(gco, "Type")=="line", ', ...
              'set(gco , "LineWidth", 4.0), ', ...
            'end']);
```

对**Line Color**子菜单项也做同样的操作。将菜单项背景加色并在合适时改变文本颜色。

```
    uimenu(Hm_lcolor, 'Label', 'yellow', ...
        'BackgroundColor', 'y', ...
        'CallBack', ['witforbuttonpress;', ...
            'if get(gco, "Type")=="line", ', ...
              'set(gco , "color", '"y"), ', ...
            'end']);

    uimenu(Hm_lcolor, 'Label', 'Magenta', ...
        'BackgroundColor', 'm', 'ForegroundColor', 'w'...
        'CallBack', ['witforbuttonpress;', ...
            'if get(gco, "Type")=="line", ', ...
              'set(gco , "color", '"m"), ', ...
            'end']);

    uimenu(Hm_lcolor, 'Label', 'yan', ...
        'BackgroundColor', 'c', ...
        'CallBack', ['witforbuttonpress;', ...
            'if get(gco, "Type")=="line", ', ...
              'set(gco , "color", '"c"), ', ...
            'end']);

    uimenu(Hm_lcolor, 'Label', 'Red', ...
        'BackgroundColor', 'r', 'ForegroundColor', 'w', ...
        'CallBack', ['witforbuttonpress;', ...
            'if get(gco, "Type")=="line", ', ...
              'set(gco , "color", '"r"), ', ...
            'end']);

    uimenu(Hm_lcolor, 'Label', 'Green', ...
        'BackgroundColor', 'g', ...
        'CallBack', ['witforbuttonpress;', ...
            'if get(gco, "Type")=="line", ', ...
              'set(gco , "color", '"g"), ', ...
            'end']);

    uimenu(Hm_lcolor, 'Label', 'Blue', ...
        'BackgroundColor', 'b', 'ForegroundColor', 'w', ...
        'CallBack', ['witforbuttonpress;', ...
            'if get(gco, "Type")=="line", ', ...
              'set(gco , "color", '"b"), ', ...
            'end']);
```

```
uimenu(Hm_lcolor，'Label'，'White'，...
    'BackgroundColor'，'w'，...
    'CallBack'，['witforbuttonpress；'，...
        'if get(gco，"Type")=="line"，'，...
          'set(gco，"color"，'"w")，'，...
        'end']);

uimenu(Hm_lcolor，'Label'，'Black'，...
    'BackgroundColor'，'k'，'ForegroundColor'，'w'，...
    'CallBack'，['witforbuttonpress；'，...
```

```
    'if get(gco，"Type")=="line"， '，...
      'set(gco，"color"， '"k")， '，...
    'end']);
```

为使这一函数更完全，可用同样方法增加另外的菜单以改变坐标轴、曲面、补片和图形的属性。例如，下面加上了一个**Color**映象以改变图形颜色的映象。

```
Hm_cmap=uimenu(gcf， 'Label'， 'Color Map');

uimenu(Hm_cmap， 'Label'， 'Lighter'， 'CallBack'， 'brighten(.3)');
uimenu(Hm_cmap， 'Label'， 'Darker'， 'CallBack'， 'brighten(-.3)');
uimenu(Hm_cmap， 'Label'， 'Default'， 'CallBack'， 'colormap("default")');
uimenu(Hm_cmap， 'Label'， 'Gray'， 'CallBack'， 'colormap(gray)');
uimenu(Hm_cmap， 'Label'， 'Hot'， 'CallBack'， 'colormap(hot)');
uimenu(Hm_cmap， 'Label'， 'Cool'， 'CallBack'， 'colormap(cool)');
uimenu(Hm_cmap， 'Label'， 'Bone'， 'CallBack'， 'colormap(bone)');
uimenu(Hm_cmap， 'Label'， 'Copper'， 'CallBack'， 'colormap(copper)');
uimenu(Hm_cmap， 'Label'， 'Pink'， 'CallBack'， 'colormap(pink)');
uimenu(Hm_cmap， 'Label'， 'Prism '， 'CallBack'， 'colormap(prism)');
uimenu(Hm_cmap， 'Label'， 'Jet'， 'CallBack'， 'colormap(jet)');
uimenu(Hm_cmap， 'Label'， 'Flag'， 'CallBack'， 'colormap(flag)');
uimenu(Hm_cmap， 'Label'， 'HSV'， 'CallBack'， 'colormap(hsvflag)');
```

最后，加上含有两个菜单项的**Quit**菜单。其中，**Close Figure**关闭图形；**Remove Menus**让图形打开但删除用户的顶层菜单及所有它的子菜单。**drawnow**命令立即删除菜单。

```
Hm_quit=uimenu(GCF， 'lABEL'， 'quit');

uimenu(Hm_quit， 'Label'， 'Close Figure'， 'CallBack'， 'close；return');

uimenu(Hm_quit， 'Label'， 'Remove Menu'， ...
      'Callback'， [...
          'delete(findobj(gcf，"Type，"uimenu"，"parent"，gcf))， '，...
          'drawnow']);
```

精通MATLAB工具箱中有许多采用这种技术的函数和其它建立有用的基于对象工具的函数。其中许多函数将在本章后面详细讨论。

21.4 控制框

在各计算机平台上，窗口系统都采用控制框和菜单，让用户进行某些操作，或设置选项或属性。控制框是图形对象，如图标、文本框和滚动条，它和菜单一起使用以建立用户图形界面，称之为窗口系统和计算机窗口管理器。

MATLAB控制框，又称**uicontrol**，与窗口管理器所用的函数十分相似。它们是图形对象，可以放置在MATLAB的图形窗中的任何位置并用鼠标激活。MATLAB的 **uicontrol**包括按钮、滑标、文本框及 弹出式菜单。

由MATLAB生成的**uicontrol**对象在Macintosh、MS-Windows 和X Window系统平台上，有稍微不同的外观，因为窗口系统表达图形对象的方法是不同的。但是，功能本质是相同的，所以相同的MATLAB编码将生成同样的对象，它在不同平台完成同样的功能。

Uicontrol由函数**uicontrol**生成。常用句法与前面所讨论的**uimenu**相同。

>>Hc_1=uicontrol(Hf_fig，'PropertyName'，PropertyValue，...）

其中，**Hc_1**是由函数**uicontrol**生成**uicontrol**对象的句柄。通过设定**uicontrol**对象的属性值'**PropertyName**'，'**PropertyValue**'定义了**uicontrol**的属性；**Hf_fig**是父对象的句柄，它必须是图形。如果图形对象句柄省略，就用当前的图形。

建立不同类型的控制框

MATLAB共有八种不同类型或型式的控制框。它们均用函数**uicontrol**建立。属性'**Style**'决定了所建控制框的类型。'**Callback**'属性值是当控制框激活时，传给**eval**在命令窗口空间执行的MATLAB字符串。

下面将分别对八种**uicontrol**对象进行讨论，并用示例说明。**uicontrol**对象的属性更为透彻的讨论和应用中更为完整的例子将在以后给出。

按钮键

按钮键，又称命令按钮或只叫按钮，是小的长方形屏幕对象，常常在对象本身标有文本。将鼠标指针移动至对象，来选择按钮键**uicontrol**，单击鼠标按钮，执行由回调字符串所定义的动作。按钮键的'**Style**'属性值是'**pushbutton**'。

按钮键典型地用于执行一个动作而不是改变状态或设定属性。下面的例子（**mmctl1.m**）建立标志为**Close**的按钮键**uicontrol**。当激活该按钮时，**close**关闭当前的图形。以象素为单位的'**Position**'属性定义按钮键的大小和位置，这是缺省的'**Units**'属性值。属性'**String**'定义了按钮的标志。

```
>>Hc_close=uicontrol(gcff，'Style'，'push'，...
                'Position'，[10 10 100 25]，...
                'String'，Close'，...
                'CallBack'，'close);
```

无线按钮

无线按钮，又称选择按钮或切换按钮，它由一个标志并和标志文本的左端一个小圆圈或小菱形所形成。当选择时，圆圈或菱形被填充，且'**Value**'属性值设为1；若未被选择，指示符被清除，'**Value**'属性值设为0。无线按钮键'**style**'的属性值是'**radiobutton**'。

无线按钮典型地用在一组互斥的选项中选择一项。为了确保互斥性，各无线按钮**uicontrol**的回调字符串必须不选组中其它项，将它们各项的'**Value**'设为0。然而，这只是一个约定，如果需要，无线按钮可与检查框交换使用。

下面的例子（**mmctl2.m**）建立了两个互斥选项的无线按钮，它将坐标轴'**Box**'属性开或关闭。

```
>> Hc_boxon = uicontrol(gcf，'Style'，'radio'，...
                'Position'，[20 45 100 20]，...
                'String'，'Set box on
                'Value'，0，...'，...
                'CallBack'，[...
                  'set(Hc_boxon'，"Value"，1'...
                  'set(Hc_boxoff'，"Value"，0'...
                  'set(gca，"Box"，"on")']);

>> Hc_boxoff = uicontrol(gcf，'Style'，'radio'，...
```

```
'Position'，[20 20 100 20]，...
'String'， 'Set box off
'Value'， 1， ...'， ...
'CallBack
'CallBack'，[...
  'set(Hc_boxon'，"Value"，0'...
  'set(Hc_boxoff'，"Value"，1'...
  'set(gca，"Box"，"off")'])；
```

检查框

检查框，又称*切换按钮*，它由具有标志并在标志的左边的一个小方框所组成。激活时，**uicontrol**在检查和清除状态之间切换。在检查状态时，根据平台的不同，方框被填充，或在框内含**x**，'**Value**'属性值设为1。若为清除状态，则方框变空，'**Value**'属性值设为0。

检查框典型地用于表明选项的状态或属性。通常检查框是独立的对象，如果需要，检查框可与无线按钮交换使用。

下面的例子（**mmctl3.m**）建立了一个检查框**uicontrol**， 设置坐标轴'**Box**'属性，当此检查框被激活时，测试'**Value**'属性以确定检查框是否以往被检查或清除过，并适当设置'**Box**'属性。

```
>> Hc_box = uicontrol(gcf， 'Style'， 'check'， ...
          'Position'，[100 50 100 20]，...
          'String'， 'Axis Box'，...
          'CallBack'，[...
            'if get(Hc_box，"Value")==1， '...
              'set(gca，"Box"，"on")， '...
            'else， '...
              '(gca，"Box"，"off"， '...
            'end'])；
```

静态文本框

静态文本框是仅仅显示一个文本字符串的**uicontrol**，该字符串是由'**string**'属性所确定的。静态文本框的'**Style**'属性值是'**text**'。静态文本框典型地用于显示标志、用户信息及当前值。

静态文本框之所以称之为'静态'，是因为用户不能动态地修改所显示的文本。文本只能通过改变'**String**'属性来更改。

在X Window系统中，静态文本框可只含有一行文字；如文本框太短，不能容纳文本串，则只显示部分文字。然而在Macintosh和MS-Window平台，长于文本框的文本串将字串起来，即在可能的地方，用词间分割的虚线显示多行。

下面的例子（**mmct14.m**）建立了含有MATLAB版本号的文本框。

```
>> Hc_ver = uicontrol(gcf， 'Style'， 'text'， ...
          'Position'，[10 10 150 20]，...
          'String'，['MATLAB Version'， version])；
```

与其它的文本对象，如：坐标标题和坐标标志不同，函数的编著者对用在uicontrol文本串的字体不明显地控制。用在uicontrol文本串的字体和命令窗口的字体一致，可由用户设定。

X window系统的用户在激活MATLAB时，可以给出命令行的参量，如

matlab -fn 9x14bold

该命令在命令窗和**uicontrol**文本串中使用**9x14bold**字体。Mocintosh对命令窗和**uicontrol**文本串使用相同的字体，但字体可以由用户在命令窗的**Options**菜单中用**Text Style**项来设定。PC用户具有选项，可从命令窗口的**Options**菜单的**Command Window Font**项来设置命令窗口字体，并从同一菜单中的**Uicontrol Font**项，为**uincontrol**文本串设置不同的字体。我们希望未来的MATLAB版本会在**uimenu**和**uicontrol**文本串中增加控制字体的属性。

可编辑文本框

编辑文本框，象静态文本框一样，在屏幕上显示字符。但与静态文本框不同，可编辑文本框允许用户动态地编辑或重新安排文本串，就象使用文本编辑器或文字处理器一样。在'**String**'属性中有该信息。可编辑文本框**uicontrol**的'**Style**'属性值是'**edit**'。可编辑文本框典型地用在让用户输入文本串或特定值。

可编辑文本框可包含一行或多行文本。单行可编辑文本框只接受一行输入，而多行可编辑文本框可接受行以上的输入。单行可编辑文本框的输入以**Return**键结尾。在X window和MS-Window系统中，多行文本输入以**Control-Return**键结尾，而在Macintosh中用**Command-Return**键。

下面的例子（**mmct15.m**)建立了静态文本标志和一个单行可编辑文本框。用户可以在文本框中输入颜色映象名，而后回调字符串把它放到在图中。

```
>>Hc_label=uicontrol(gcf，'Style'，'text'，...
            'Position'[10 10 70 20]，...
            'String'，'Colormap：');

>>Hc_map =uicontrol(gcf，'Style'，'edit '，...
            'Position'[80 10 60 20]，...
            'String'，'hsv'，...
            'callback'，'colormap(eval(get(Hc_map，"String")))');
```

通过把'Max'属性及'Min'属性设置成数值，诸如Max-Min>1，建立多行可编辑文本框。Max属性不指定最大的行数。多行可编辑文本框可具有无限多行。

一个多行可编辑文本框表示如下：

```
>>Hc_multi=uicontrol(gcf，'Style'，'edit'，...
                    'Position'，[20 50 75 75]，...
                    'String'，'Line 1 |Line 2|Line 3'...
                    'Max'，2);
```

多行字符串被指定为单个引号的字符串，用垂直条字符'|'指明在何处分行。

滑标

滑标，或称滚动条，包括三个独立的部分，分别是滚动槽、或长方条区域，代表有效对象值范围；滚动槽内的指示器，代表滑标当前值；以及在槽的两端的箭头。滑标**uicontrol**的'**Style**'属性值是'**slider**'。

滑标典型地用于从几个值域范围中选定一个。滑标值有三种方式设定。方法一：鼠标指针指向指示器，移动指示器。拖动鼠标时，要按住鼠标按钮，当指示器位于期望位置后松开鼠标。方法二：当指针处于槽中但在指示器的一侧时，单击鼠标按钮，指示器按该侧方向移动距离约等于整个值域范围的10% ；方法三：在滑标不论哪端单击鼠标箭头；指示器沿着

箭头的方向移动大约为滑标范围的1% 。滑标通常与所用文本**uicontrol**对象一起显示标志、当前滑标值及值域范围。

下面的例子（**mmct16.m**）实现了一个滑标，可以用于设置视点方位角。用了三个文本框分别指示滑标的最大值，最小值和当前值。

```
>> vw = get(gca，'View')；

>> Hc_az = uicontrol(gcf，'Style'，'slider'，...
          'Position'[10 5 140 20]，...
          'Min'，-90，'Max'，90，'Value'，vw(1)'，...
          'CallBack'，[...
              'set(Hc_cur，"String"，num22str(get(Hc_az，"Value")))，'...
              'set(gca，"View"，[get(Hc_az，"Value") vw(2)])'])；

>> Hc_min = uicontrol(gcf，'Style'，'text'，...
          'Position'，[10 25 40 20]，...
          'String'，num2str(get(Hc_az，'Min')))；

num2str(get(Hc_az，'Min')))；

>> Hc_max = uicontrol(gcf，'Style'，'text'，...
          'Position'，[110 25 40 20]，...
          'String'，num2str(get(Hc_az，'Max))；

>> Hc_cur = uicontrol(gcf，'Style'，'text'，...
          'Position'，[60 25 40 20]，...
          'String'num2str(get(Hc_az，'Value')))；
```

滑标的'**Position**'属性包含熟悉的向量[**left bottom width height**]，其单位由'**Units**'属性设定。滑标的方向取决于宽与高之比。如果**width > height**就画水平方向的滑标，如果**width < height**就画垂直方向的滑标。仅在X-Window系统平台中，如果滑标的一个方向的大小比另一个方向小4倍，就不显示。其它操作平台上的滑标均有箭头。

弹出式菜单

弹出式菜单典型地用于向用户提出互斥的一系列选项清单，让用户可以选择。弹出式菜单，不同于前面论述过的下拉式菜单，不受菜单条的限制。弹出式菜单可位于图形窗口内的任何位置。弹出式菜单的**'Style'**属性值是'**popupmenu**'。

当关闭时，弹出式菜单以矩形或按钮的形式出现，按钮上含有当前选择的标志，在标志右侧有一个向下的箭头或凸起的小方块来表明**uicontrol**对象是一个弹出式菜单。当指针处在弹出式**uicontrol**之上并按下鼠标时，出现其它选项。移动指针到不同的选项，松开鼠标就关闭弹出式菜单，显示新的选项。MS-Windows 和某些X Window系统平台允许用户单击弹出式菜单，打开它，而后单击另一个选项来进行选择。

当选择一个弹出项时，'**Value**'属性值设置成选择向量所选元素的下标。选项的标志指定为一个字符串，用垂直条'|'分隔，与指定多行文本的方法一样。下面的例子（**mmct17.m**）建立了图形颜色的弹出式菜单。回调函数把图形的'**Color**'属性值设定为所选值。每种与颜色相关的RGB值存储在弹出控制框的'**UserDate**'属性中。所有句柄图形对象的'**UserData**'属性仅仅为单独矩阵提供孤立的存储。

```
>> Hc_fcolor = uicontrol(gcf，'Style'，'popumenu'，...
```

```
'Position'，[20 20 80 20]，...
'String'，'Black|Red|Yellow|Green|Cyan|Blue|Magenta|White'，...
'Value'，1，...
UserData'，[[0 0 0]；...
          [1 0 0]；...
          [1 1 0]；...
          [0 1 0]；...
          [0 1 1]；...
          [0 0 1]；...
          [1 0 1]；...
          [1 1 1]；...
CallBack'，[...
   'UD=get(Hc_fcolor，''UserData'')；'，...
   'set(gcf，''Color''，UD(get(Hc_fcolor，''Value'')，：))'])；
```

弹出式菜单的**'Position'**属性含有熟悉的向量[**left bottom width height**]，其中宽度与高度决定了弹出对象的大小。在X Window和Macintosh系统中，就是被关闭的弹出式菜单的大小。打开时，菜单展开适合显示屏幕大小所有的选项。在MS-Windows系统中，高度值基本上被忽略，这些平台建立高度足够的弹出式菜单，显示一行文本而不管**height**的值。

框架

框架**uicontrol**对象仅是带色彩的矩形区域。框架提供了视觉的分隔性。在这点上，框架与**uimenu**的**'Sepatator'**属性相似。框架典型地用于组成无线按钮或其它**uicontrol**对象。在其它对象放入框架之前，框架应事先定义。否则，框架可能覆盖控制框使它们不可见。
下面的例子（**mmct18.m**）建立了一个框架，把两个按钮和一个标志放入其中。

```
>> Hc_frame = uicontrol(gcf，'Style'，'frame'，'Position'，[250 200 95 65])；

>> Hc_pb1 = uicontrol(gcf，'Style'，'pudhbutton'，...
        'Position'，[255 205 40 40]，'String'，'OK')；

>> Hc_pb2 = uicontrol(gcf，'Style'，'pushbutton'，...
        'Position'，[300 205 40 40]，'String'，'NOT')；


>> Hc_1b1 = uicontrol(gcf，'Style'，'text'，...
        'Position'，[255 250 85 10]，'Str'，'Push Me')；
```

控制框属性

如句柄图形对象建立函数一样，**uicontrol**属性可在对象建立时定义，或如上所示，用**set**命令来改变。所有可设定的属性，包括字符串文本、回调串、甚至控制框函数类型都可以用**set**来改变。本章后面有若干例子。

表21.2列出了MATLAB 4.2版本中**uicontrol**对象的属性及其值。带有*的属性为非文件式的，使用时需加小心。由{}括起来的属性值是缺省值。

表21.2

Uicontrol 对象属性

BackgroundColor		**uicontrol**背景色。3元素的RGB向量或MATLAB一个预先定义的颜色名称。缺省的背景色是浅灰色。
Callback		MATLAB回调串，当**uicontrol**激活时，回调串传给函数**eval**；初始值为空矩阵。
ForegroundColor		**uicontrol**前景（文本）色。3元素的RGB向量或MATLAB一个预先定义的颜色名称。缺省的文本色是黑色。
HorizontalAlignment		标志串的水平排列
	left：	相对于**uicontrol**文本左对齐
	{**center**}：	相对于**uicontrol**文本居中
	right：	相对于**uicontrol**文本右对齐
Max		属性 ' **Value** ' 的最大许可值。最大值取决于**uicontrol**的 ' Type ' 当**uicontrol**处于**on**状态时，无线按钮及检查框将**Value**设定为**Max**；该值定义了弹出式菜单最小下标值或滑标的最大值。当**Max-Min>1**时，可编辑文本框是多行文本。缺省值为1
Min		属性 ' **Value** ' 的最小许可值。最小值取决于**uicontrol**的 ' **Type** ' **uicontrol**处于**off**状态时。无线按钮及检查框将**Value**设定为**Min**；该值定义了弹出式菜单最小下标值或滑标的最小值。当**Max-Min>1**时，可编辑文本框是多行文本。缺省值为0
Position		位置向量**[left bottom width height]**。其中，**[left height]**表示相对于图形对象左下角的**uicontrol**的左下角位置。**[widht height]**表示**uicontrol**的尺寸大小，其单位由属性**Units**确定。
Enable*		控制框使能状态
	{**on**}：	**uicontrol**使能。激活**uicontrol**，将**Callback**字符串传给**eval**
	off：	**unicntrol**不使能，标志串模糊不清。激活**unicontrol**不起作用
String		文本字符串，在按钮键，无线按钮，检查框和弹出式菜单上指定**uicontrol**的标志。对于可编辑文本框，该属性设置成由用户输入的字符串。对弹出式菜单或可编辑文本框中多个选项或，每一项用垂直条(|)分隔，整个字符串用引号括起来。框架和滑标，不用引号
Style		定义**uicontrol**对象的类型
	{**pushbutton**}：	按钮键：选择时执行一个动作。
	radiobutton：	无线按钮键：单独使用时，在两个状态之间切换；成组使用时，让用户选择一个选项
	checkbox：	检查框：单独使用时，在两个状态之间切换；成组使用时，让用户选择一个选项
	edit：	可编辑框：显示一个字符串并可让用户改变
	text：	静态文本框：显示一个字符串
	slider：	滑标：让用户在值域范围内选择一个值。
	frame：	框架：显示包围一个或几个**uicontrol**的框架，使其形成一个逻辑群。
	popumenu：	弹出式菜单：含有许多互斥的选择的弹出式菜单

Units		位置属性值的单位
	inches:	英寸
	centimeters:	厘米
	normalized:	归一化的坐标值，图形的左下角映射为[0 0]而右上角
	points:	的映射为为[1 1]
	{**pixels**}:	打印设置点，等于1/72 英寸
		屏幕的象素。计算机屏幕分辨率的最小单位。
Value		**uicontrol**的当前值。无线按钮和检查框在 ' **on** ' 状态时，**value**设为**Max**，当是 ' **off** ' 状态时，**value**设为**Min**。由滑标将滑标的**value**设置为数值（**Min≤Value≤Max**），弹出式菜单把**value**值设置所选择选项的下标（**1≤Value≤Max**）。文本对象和按钮不设置该属性。
ButtonDownFcn		当**uicontrol**被选择时，MATLAB回调串传给函数**eval**。初始值为空矩阵
Children		**Uicontrol**对象一般无子对象，通常返回空矩阵
Clipping		限幅模式
	{**on**}:	对**uicontrol**对象无作用效果
	off:	对**uicontrol**对象无作用效果
DestroyFcn		只对Macintosh 4.2 版本。没有文件说明
Interrruptible		指定**ButtonDownFcn**和**CallBack**串是否可中断
	{**on**}:	回调不能由其它回调中断
	off:	回调串可被中断
Parent		包含**uicontrol**对象的图形句柄
*Select		值为**[on|off]**
*Tag		文本串
Type		只读对象辩识串，通常为**uicontrol**
UserData		用户指定的数据。可以是矩阵，字符串等等
Visible		**uicontrol**对象的可视性
	{**on**}:	**uicontrol**对象在屏幕上可见
	off:	**uicontrol**对象不可见，但仍然存在

控制框布置的考虑

uicontrol的属性 ' **Position** ' 和 ' **Units** ' 用于分配图形窗口中对象的位置。**uicontrol**的缺省 ' **Position** ' 向量为**[20 20 60 20]**，以象素表示。该值是缺省 ' **Units** ' 值。这是一个**60×20**象素**uicontrol**的象素矩形，**uicontrol**左下角位于父图形左下角的靠右边20个象素点，靠上边20 个象素。缺省的图形尺寸大约为**560×420**个象素，位于显示屏的中上部。利用这些信息，**uicontrol**的布置就变成了一个2维几何布局问题。

要加若干限制。比如，MS-Window忽略位置向量高度值，仅有足够高度以显示一行文本。在所有其它情况下，必须保证控制框足够大以容纳控制框标志字符串。因为显示控制框 ' **String** ' 属性值所用的字体与用在命令窗口的字体一致，所以，用户对于标志每个控制框的字体属性无法控制，不同平台用不同字体，整体上可由用户改变。

通常确定控制框的大小和位置是一个尝试的过程。即使结果很满意，图形在另一个平台上的外观也许会很不一样，还需调整。通常希望把控制框制作得比所需要的更大一些，以便在所有平台上其标志都可读。

正因为图形有缺省尺寸，不能保证图形都具有缺省大小。若在现有的图形窗口内加上**uicontrol**和**uimenu**，则图形尺寸会比缺省值大或小。另外，用户可以在任何时候，对任何图形重调尺寸，除非把图形的 ' **Resize** ' 属性值置 ' **off** ' ，才可避免这样做。

当把**uicontrol**加到尺寸可能重新调整的图形时，有两点需要考虑：属性'**Units**'和由固定字体标志字符串所加的约束。若**uicontrol**的位置是以绝对单位指定如：**象素、英寸、厘米或点**，则窗口尺寸的重调不会影响**uicontrol**的大小和位置。**uicontrol**相对于图形左下角的位置不变；若图形变小，则某些**uicontrol**可能不可见。

若**uicontrol**的位置以归一化单位指定，则当图形尺寸调整时，**uicontrol**相互之间及与图形本身之间的关系保持不变。但是，使用归一化的单位有一个缺点，即如果图形变小，**uicontrol**对象大小也要调整，则**uicontrol**的标志可能看不见，因为字体大小是固定的。任何超出调整后的**uicontrol**的部分就被删去。希望以后MATLAB版本会给程序员对**uicontrol**和**uimenu** 的标记字符串的字体属性有更多的控制框。

M 文件举例

下面例子说明了本章所讨论的某些控制框的用法。精通MATLAB工具箱中的函数**mmclock**在屏幕上生成一个数字钟，它用任选参数设定钟的位置。它使用了框架、文本、无线按钮、检查框和按钮键**uicontrol**。

为了在PC上运行这个例子，在命令窗的**Option**菜单中选择**Enable Background Process**菜单项。这样将引起MATLAB进入无限循环和死锁。为了在Macintosh上得到较好的结果，关掉**Option**菜单的**Background Operation**检查标记。X Windows系统版本不需任何调整。

首先，建立函数的语句和帮助文本。

```
function T=mmclock(X，Y)
%MMCLOCK Place a digital clock on the screen.
% MMCLOCK places a digital clock at the upper-right corner
% of the display screen.
% MMCLOCK(X，Y) places a digital clock at position X pixels
% to the right and Y pixels above the bottom of the screen.
% T=MMCLOCK returns the current date and time as a string
% when 'Close' is pressed.
```

设定初始值，分析输入参量。

```
fsize=[200 150]； sec=1； mil=0；
mstr=['Jan'； 'Feb'； 'Mar'； 'Apr'； 'May'； 'Jun'
      'Jul'； 'Aug'； 'Sep'； 'Oct'； 'Nov'； 'Dec'];
scr=get(0，'ScreenSize')；

if nargin= =0
   figpos=[scr(3)-fsize(1)-20 scr(4)-fsize(2)-5 fsize(1：2)]；
elseif nargin= =2
   figpos={X Y fsize(1：2)]；
else
    error('Invalid Arguments')；
end
```

建立图形，在图中设置**uicontrol**为某些缺省值。

```
Hf_clock=figure('Position'，figpos'，...
          'Color'，[.7 .7 .7]，...
          'NumberTitle'，'off'，...
          'Name'，'Digital Clock')；
set(Hf_clock，'DefaultUicontrolUnits'，'normalized'，...
         'DefaultUicontrolBackgroundColor'，get(Hf_clock，'Color'))；
```

对秒建立**Close** 按钮键、无线按钮；对24小时建立检查框。

```
Hc_close=uicontrol( ' Style ' ， ' push ' ， ...
        ' Position ' ，[.65 .05 .30 .30]，...
        ' BackgroundColor ' ，[.8 .8 .9]，...
        ' String ' ， ' Close ' ，...
        ' CallBack ' ， ' close(gcf) ' )；

Hc_sec=uicontrol( ' Style ' ， ' radiobutton ' ，...
        ' Position ' ，[.05 .05 .50 .13]，...
        ' Value ' ，sec，...
        ' String ' ， ' Seconds ' )；

Hc_mil=uicontrol( ' Style ' ， ' checkbox ' ，...
        ' Position ' ，[.05 .22 .50 .13]，...
        ' Value ' ，mil，...
        ' String ' ， ' 24-Hour ' )；
```

对日期和时间显示创建框架和文本串。

```
Hc_dframe=uicontrol( ' Style ' ， ' frame ' ， ' Position ' ，[.04 .71 .92 .24])；
Hc_date   =uicontrol( ' Style ' ， ' text ' ， ' Position ' ，[.05 .72 .90 .22])；
Hc_tframe=uicontrol( ' Style ' ， ' frame ' ， ' Position ' ，[.04 .41 .92 .24])；
Hc_time=uicontrol( ' Style ' ， ' text ' ， ' Position ' ，[.05 .42 .90 .22])；
```

循环，直到按钮键回调函数关闭图形，每秒更新显示一次。

```
while find(get(0， ' Children ' ) = = Hf_clock % Looop while clock exists
    sec = get(Hc_sec， ' Value ' )；
    mil = get(Hc_mil， ' Value ' )；
    now=fix(clock)；
    datestr=sprintf( ' %s %2d %4d ' ， mstr(now(2)，： ，now(3)，now(1))；
    if mil
       suffx= ' ' ；
    else
       if now(4)>12
          suffix= ' pm ' ；
          now(4)=rem(now(4)，12)；
       else
          suffix= ' am ' ；
       end
    end
    timestr=[num2str9now(4)) ' ： ' sprintf( ' %02d ' ， now(5))]；
    if sec
      timestr=[timestr ' ： sprintf( ' %02d，now(6))]；
    end
timestr=[timestr suffix]；
    set(Hc_date. ' String ' ， datestr)；
    set(Hc_time， ' String ' ， timestr)；
    pause(1)
end
```

最后，如果有需，返回日期和时间字符串。

```
if nargout
    T=[datestr '-' timestr];
end
```

虽然这个例子说明了如何用**uicontrol**建立一个完整的个GUI函数，但并不实用。因为该函数保持循环直至**Close**按钮按下，非等到时钟图形关闭，命令窗口才能使用。

注意，上例只含有一个回调函数字符串，即在按钮键回调串中的**'close'**语句。其它按钮值是在While循环中得到，而不是按钮本身用回调函数激发一个动作。更复杂的**M文件**需要复杂的回调。

21.5 编程和回调考虑

用户句柄图形界面的基础部分已经阐述过了，现在是应用它的时候了。正如所见，在命令行通过输入**uimenu**和**uicontrol**来建立**效率不高**。脚本或函数M文件使用更为简便。假定想实现一个M文件，首先确定是否要编写脚本文件或函数文件。

脚本与函数

脚本文件似乎是当然的选择。在脚本中，所有的命令都在命令工作窗口执行，因此随时可以使用所有的MATLAB函数和对象句柄。将信息传给回调函数无任何困难。然而这里有几种权衡。首先，当所有的变量都是可利用时，工作空间内充斥了变量名和变量值，即使它们已经不再有用。其次，如果用户使用**clear**命令，重要的对象句柄就可能丢失。另一个缺点是：用脚本文件定义回调字符串可能变得十分复杂。例如，以下有一个滑标的文件片段，节选自精通MATLAB工具箱中脚本文件**mmsetclr**

```
Hc_rsli=uicontrol(Hf_fig，'Style'，'slider'，...
   'Positon'，[.10 .55 .35 .05]，...
   'min'，0，'max'，1，'Value'，initrgb(1)，...
   'CallBack'，[...
     'set(Hc_nfr，"BackgroundColor"，'，...
      '[get(Hc_rsli，"Val")，get(Hc_gsli，"Val")，get(Hc_bsli，"Val")])，'，...
     'set(Hc_ncur，"String"，'，...
     'sprintf("[%.2f %.2f %.2f]"，get(Hc_nfr，"BakgroundColor")))，'，...
     'hv=rgb2hsv(get(Hc_nfr，"BakgroundColor"))；'，...
     'set(Hc_hsli，"Val"，hv(1))，'，...
     'set(Hc_hcur，"String"，sprintf("%.2f"，hv(1)))，'，...
     'set(Hc_ssli，"Val"，hv(2))，'，...
     'set(Hc_scur，"String"，sprintf("%.2f"，hv(2)))，'，...
     'set(Hc_vsli，"Val"，hv(3))，'，...
     'set(Hc_vcur，"String"，sprintf("%.2f"，hv(3)))，'，...
     'set(Hc_rcur，"String"，sprintf("%.2f"，get(Hc_rsli，"Val")))'])；
```

另一个问题是，脚本文件比函数文件运行得慢，脚本在第一次运行时要编译。最后一点，脚本文件没有函数灵活。函数可接受输入参量并返回值。因此，函数可用作其它函数的参变量。

函数不会使命令窗工作空间拥挤；当反复调用时运行快速；接受输入参量并返回值；回调字符串书写不复杂。因此，在许多场合，函数M文件是最佳的选择。

考虑前面脚本文件中滑标定义的例子。以下是等价的文件片段，取自精通MATLAB工具箱中函数文件**mmsetc**。

```
Hc_rsli=uicontrol(Hf_fig，'Style'，'slider'，...
    'Position'，[.10 .55 .35 .05]，...
    'Min'，0，'Max'，1，'Value'，initrgb(1)，...
    'CallBack'，'mmsetc(0，"rgb2new")')；
```

注意回调字符串以不同的参量调用**mmsetc**。这是一个在回调中使用递归函数调用的例子。然而函数本身有它自己的问题。主要的困难源于要将回调字符串传给**eval**并在**命令窗口工作空间中运行**，而其余的程序码在函数工作空间内执行。这里用前面章节所讨论的变量和函数的大量规则。在函数内定义的变量在命令窗口工作空间不存在，因此不在回调串中使用。同时在命令窗工作空间的变量在函数本身内部也不存在。

有若干既能解决该问题又能利用函数优点的方法。全局变量、'**UserData**'属性、仅用于回调的特殊函数M文件和递归函数调用均是创建GUI函数十分有用的工具。

独立的回调函数

建立GUI函数的一个有效方法是编写独立的回调函数，专门执行一个或多个回调。函数使用的对象句柄和其它变量可以作为参量传递，必要时回调函数可返回值。

考虑先前的一个例子，建立一个方位角的滑标，以脚本文件来实现。

```
% setview.m script file

vw=get(gca，'View')；

Hc_az=uicontrol(gcf，'Style'，'slider'，...
        'Position'，[10  5  140  20]，...
        'Min'，-90，'Max'，90，'Value'，vw(1)，...
        'Callback'，[...
            'set(Hc_cur，'String'，num2str(get(Hc_az，'Value')))，'...
            'set(gca，'View'，[get(Hc_az，'Value') vw(2)])']);
Hc_min=uicontrol(gcf，'style'，'text'，...
         'Position'，[10  25  40  20]，...
         'String'，num2str(get(Hc_az，'Min')))；

Hc_max=uicontrol(gcf，'Style'，'text'，...
         'Position'，[110  25  40  20]，...
         'String'，num2str(get(Hc_az，'Max')))；

Hc_cur=uicontrol(gcf，'Style'，'text'，...
         'Position'，[60  25  40  20]，...
         'String'，num2str(get(Hc_az，'Value')))；
```

下面是同样的例子。作为一个函数，采用'**Tag**'属性来辨别控制框，并使用独立的M文件来执行回调。

```
funtion setview( )

vw=get(gca，'View')；

Hc_az=uicontrol(gcf，'Style'，'Slider'，...
        'Position'，[10  5  140  20]，...
        'Min'，-90，'Max'，90，'Value'，vw(1)，...
```

```
        'Tag', 'Azslider', ...
        'Callback', 'svcback');

Hc_min=uicontrol(gcf, 'style', 'text', ...
        'Position', [10 25 40 20], ...
        'String', num2str(get(Hc_az, 'Min')));

Hc_max=uicontrol(gcf, 'Style', 'text', ...
        'Position', [110 25 40 20], ...
        'String', num2str(get(Hc_az, 'Max')));

Hc_cur=uicontrol(gcf, 'Style', 'text', ...
        'Position', [60 25 40 20], ...
        'Tag', 'Azcur', ...
        'String', num2str(get(Hc_az, 'Value')));
```

回调函数本身如下：

```
function svcback( )

vw = get(gca, 'View');

Hc_az = findobj(gcf, 'Tag', 'AZslider');
Hc_cur = findobj(gcf, 'Tag', 'AZcur');

str = num2str(get(Hc_az, 'Value'));
newview =[get(Hc_az, 'Value') vw(2)];
set(Hc_cur, 'String', str)
set(gca, 'View', newview)
```

上面的例子并不节省很多代码，但却得到了用函数而不用脚本文件的优点：回调函数可以利用临时变量，而不使命令窗口工作空间拥挤；不需要**eval**所需的引号和字符串；在回调函数中命令的句法变得十分简单。使用独立回调函数技术，越复杂的回调(函数)越简单。

独立回调函数的缺点是：需要很大数目的M文件以实现一个含有若干控制框和菜单项的GUI函数，所有这些M文件必须在MATLAB路径中可得，且每一个文件又必须要有一个不同的文件名。在对文件名大小有限制且对大小写不敏感的平台上，如MS-windows，文件冲突的机会就增加了。而且回调函数只能被GUI函数调用而不能被用户调用。

递归函数调用

利用单独的M文件并递归地调用该文件，既可以避免多个M文件的复杂性，又可以利用函数的优点。使用开关 **switches**或**if elseif**语句，可将回调函数装入调用函数内。通常这样一种函数调用的结构为

```
function guifunc(switch)。
```

其中**switch**确定执行哪一个函数开关的参量，它可以是字符串'**startup**'，'**close**'，'**sectolor**' 等等，也可以是代码或数字。如**switch**是字符串，则可如下面所示的M文件片段那样将开关编程。

```
if nargin < 1, switch = 'startup'; end;
if ~isstr(switch), error('Invalid argument'), end;
```

```
if strcmp(switch， ' startup ' )，
   <statement to create controls or menus>
   <statements to implement the GUI function>
elseif strcmp(switch， ' setcolor ' )，
   <statements to perform the Callback associated with setcolor>
elseif strcmp(switch， ' close ' )，
   <statements to perform the Callback associated with close>
end
```

如果是代码或字符串，开关也可以相同方式编程。

```
if nargin < 1， switch = 0； end；
if isstr(switch)， error( ' Invalid argument ' )， end；
if switch = = 0，
    <statements to create controls or menus>
    <statements to implement the GUI function>
elseif switch = =1，
    <statements to perform the Callback associated with setcolor>
elseif switch ==2，
    <statements to perform the Callback associated with close>
end
```

下面的例子说明了方位角滑标如何可作为单独的函数M文件来实现：

```
function setview(switch)

if nargin < 1， switch = ' startup ' ； end；
if ~isstr(switch)， error( ' Invalid argument. ' )； end；

vw = get(gca， ' view ' )； % This information is needed in both sections

if strcmp(switch， ' startup ' ) % Define the controls and tag them

   Hc_az = uicontrol(gcf， ' Style ' ， ' slider ' ， ...
         ' Position ' ， [10 5 140 20]， ...
         ' Min ' ， -90， ' Max ' ， 90， ' Value ' vw(1)， ...
         ' Tag ' ， ' AZslider ' ， ...
         ' Callback ' ， ' setview( ' set ' ) ' )；

   Hc_min=uicontrol(gcf， ' Style ' ， ' text ' ， ...
          ' Position ' ， [10 25 40 20]， ...
          ' String ' ， num2str(get(Hc_az， ' Min ' )))；
   Hc_max = uicontrol(gcf， ' Style ' ， ' text ' ， ...
          ' Position ' ， [110 25 40 20]， ...
          ' String ' ， num2str(get(Hc_az， ' Max ' )))；

   Hc_cur =uicontrol(gcf， ' Style ' ， ' text ' ， ...
         ' Position ' ， [60 25 40 20]， ...
         ' Tag ' ， ' AZcur ' ， ...
         ' string ' ， num2str(get(Hc_az， ' Value ' )))；

elseif strcmp(switch， ' set ' ) % Execute the Callback
```

```
    Hc_az=findobj(gcf， ' Tag ' ， ' AZslider ' )；
    Hc_cur-findobj(gcf， ' Tag ' ， ' AZcur ' )；

    str = num2str(get(Hc_az， ' Value ' ))；
    newview - [get(Hc_az， ' Value ' ) vw(2)]；

    set(Hc_cur， ' String ' ， str)
    set(gca， ' View ' ， newview)
end
```

上述的两个例子均设置了**' tag '**属性，利用该属性和函数**findobj**寻找回调函数所需对象的句柄。另外两种方法将在下章描述。

全局变量

全局变量可用在函数中，使某些变量对GUI函数的所有部分都可用，全局变量是在函数的公共区说明，因此整个函数以及所有对函数的递归调用都可以利用全局变量，下面的例子说明如何利用全局变量将方位角滑标编程。

```
function setview(switch)

global HC_AZ HC_CUR % Create global variables

if nargin < 1， switch = ' startup ' ； end；
if ~isstr(switch， error( ' Invalid argument. ' )； end；

vw = get (gca， ' View ' )； % This information is needed in both sections
if strcmp(switch， ' startup ' ) % Define the controls

  Hc_AZ=uicontrol(gcf， ' style ' ， ' slider ' ， ...
        ' Position ' ， [10 5 140 20]， ...
        ' Min ' ， -90， ' Max ' ， 90， ' Value ' ， vw(1)， ...
        ' Callback ' ， ' setview( ' set ' ) ' )；

  Hc_min=uicontrol(gcf， ' Style ' ， ' text ' ， ...
        ' Position ' ， [10 25 40 20]， ...
        ' String ' ， num2str(get(Hc-AZ， ' Min ' ， )))；

  HcMax=uicontrol(gcf， ' Style ' ， ' text ' ， ...
        ' Position ' ， [110 25 40 20]， ...
        ' String ' ， num2str(get(Hc_AZ， ' Max ' )))；

  Hc_cur=uicontrol(gcf， ' style ' ， ' text ' ， ...
        ' Position ' ， [60 25 40 20]， ...
        ' String ' ， num2str(get(HC_AZ， ' Value ' )))；

elseif strcmp(switch， ' set ' ) % Execute the Callback

    str=num2str(get(HC_AZ， ' Value ' ))；
    newview= [get(HC_AZ， ' Value ' ) vw(2)]；

    set(HC_CUR， ' String ' ， str)
```

```
    set(gca，'View'，newview)

end
```

全局变量遵循MATLAB的规定，变量名要大写。不需要'**tag**'属性，且不使用它，另外因为对象句柄存在的，不需要用函数**findobj**去获取，故回调代码比较简单，全局变量通常使一个函数更有效。

不过有一点要注意，尽管一个变量在函数内说明为**全局**的，变量并不能自动地在命令窗口工作空间中利用，也不能在回调字符串内使用。但是，如果用户发命令：**>> clear global**，则所有全局变量则都被破坏，包括在函数内定义的那些变量。

当单独的一个图形或有限个变量要被所有的回调(函数)利用时，全局变量使用和递归性函数调用都是有效的技术。对于包含多个图形的更复杂的函数，或用独立对象回调函数实现的情况，'**UserData**'属性更合适。另外，只要可获得对象句柄，对象'**UserData**'**的**属性值在命令窗口工作空间中是存在的。

用户数据属性

同属性'**Tag**'一样，'**UserData**'属性可在函数之间或递归函数的不同部分之间传递信息。如果需要多个变量，这些变量可以在一个容易辨识的对象的属性'**UsetData**'中传递。**如前面所述，对与句柄图形对象在一起的单个数据矩阵**'UserData'**提供了存储**。下面的程序利用了当前图形的'**UserData**'属性来实现方位角滑标。

```
function setview(switch)

if nargin < 1， switch = 'startup'； end；

vw = get(gca，'View')； % This information is needed in both sections

if strcmp(switch，'startup') % Define the controls

   Hc_az = uicontrol(gcf，'Style'，'slider'，...
         'Position'，[10 5 140 20]，...
         'Min'，-90，'Max'，90，'Value'，vw(1)，...
         'Callback'，'setview("set")')；

   Hc_min = uicontrol(gcf，'Style'，'text'，...
         'Position'，[10 25 40 20]，...
         'String'，num2str(get(Hc_az，'Min')))；

  Hc_max = uicontrol(gcf，'Style'，'text'，...
         'Position'，[110 25 40 20]，...
         'String'，num2str(get(Hc_az，'Max')))；

  Hc_cur = uicontrol(gcf，'Style'，'text'，...
         'Position'，[60 25 40 20]，...
         'String'，num2str(get(Hc_az，'Value')))；

  set(gcf，'UserData'，[Hc_az Hc_cur])； % Store the object handles

elseif strcmp(switch，'set') % Execute the Callback

  Hc_all = get(gcf，'UserData')； % retrieve the object handles
```

```
    Hc_az = Hc_all(1);
    Hc_cur = Hc_all(2);

    str = num2str(get(Hc_az， ' Value ' ));
    newview = [get(Hc_az， ' Value ' ) vw(2)];
    set(Hc_cur， ' String ' ， str)
    set(gca， ' View ' ， newview)
end
```

句柄存储于 **' startup '** 末端，图形 属性**' UserData '** 中，在回调被执行前对此进行检索。如果有许多回调，如下面的程序片断所示，**' UserData '** 只需检索一次。

```
if strcmp(switch， ' startup ' ) % Define the controls and tag them

    % <The ' startup ' code is here>

    set(gcf， ' UserData ' ， [Hc_az Hc_cur]); % Store the object handles

else  % This must be a Callback

    Hc_all=get(gcf， ' UserData ' ); % Retrieve the object handles
    Hc_az=Hc_all(1);
    Hc_cur=Hc_all(2);

    if strcmp(switch， ' set ' )

        % <The ' set ' Callback code is here>

    elseif strcmp (switch， ' close ' )

        %<The ' close ' Callback code is here>
    % <Any other Callback code uses additional elseif clauses>
    end
end
```

调试GUI M文件

回调字符串在命令窗口工作空间中计算并执行的，这个情况对编写和调试GUI函数和脚本文件有某种隐含意义。回调字符串可以很复杂，尤其是在脚本文件中，这为句法错提供了许多机会，记录单引号、逗号、括号是令人头痛的事。如果出现了句法错误，MATLAB给出提示；只要对象的**' Callback '** 属性值是一个真正的文本串，MATLAB就认可了。只有当对象被激活并将回调字符串传给**eval**时，才检查回调字符串内部的句法错误。

这样让用户定义回调字符串，它涉及还未曾定义过的对象句柄和变量，这使编写相互参照的程序变得更容易，但是每个回调函数必须分别测试，保证回调字符串是合法的MATLAB命令，并且回调字符中涉及的所有变量可在命令窗口工作空间中是可利用的。

将回调象函数M文件一样编程或象GUI函数本身内的开关一样编程，就可以不运行整个GUI函数而对各个回调进行改变或测试。

因为回调字符串是在命令窗工作空间中而不在函数本身内计算，在函数与各回调之间传递数据就变得十分复杂。例如，函数test包含如下程序：

```
function test()

tpos1=[20  20  50  20];
```

```
tpos2=[20  80  50  20];

Hc_text=uicontrol('Style', 'text', 'String', 'Hello', 'Position', tposl);

Hc-push=uicontrol('Style', 'push', 'String', 'Move Text', ...
 'Position', [15  50  100  25], ...
 'Callback', 'set(Hc-text, "Position", tpos2)');
```

所有语句都是有效的MATLAB命令，且回调字符串也对有效的MATLAB语句估值。文本对象和按钮出现在图形上，但当激活按钮键时，MATLAB就出示错误。

```
>> test
>>
??? Undefined functiion or variable Hc_text.

??? Error while evaluating Callback string.
```

如果**test**是个脚本文件，就不会出现这样问题，因为所有变量可在命令窗口工作空间中使用，因为**test**是个函数，**Hc_text**和**tpos2**在命令窗口工作空间中均未定义，回调字符串执行失败。

一种解决方法是使用各个字符串元素来建立回调，该字符串元素由数值而非变量建立，例如，改变回调字符串如下：

```
'Callback', ['set9', ...
      sprintf('&.15g', Hc_text), ...
      ', ''Position'', ', ...
      sprintf('[%.15g  %.15g  %.15g  %.15g]', tpos2), ...
      ')']);
```

建立了包括**Hc_text**对象句柄值的一个字符串，该值变换成具有15位精度的字符串，而**tpos2**变量转换成矩阵表示的字符串。在函数内计算**sprinf**语句，然后将所得的字符串用在回调中。在命令窗工作空间执行的实际命令如下所示

```
eval('set(87.000244140625, ''Position'', [20  80  50  100])')
```

将一个对象句柄转换为字符串，必须保持全精度。上例中的变换，使用了小数点后15位的数字的精度。在MATLAB中句柄对象转换应使用这样的精度。

要记住，变量随后的变换不会改变回调字符串。在前面的例子中，在控制框定义之后改变**tpos2**的值，就无效果。例如，在函数结尾处加命令

```
tpos2=[20  200  50  20 ]
```

就无效果，因为在**tpos2**重新定义之前，通过计算**tpos2**，回调字符串已经建立。

21.6 指针和鼠标按钮事件

GUI函数利用鼠标箭头的位置和鼠标按钮的状态来控制MATLAB行动。本节讨论指针、对象位置和鼠标按钮动作之间的交互，以及MATLAB如何响应变化或事件，诸如：揿下按钮、松开按钮或箭头移动等。

回调属性，选择区域和堆积顺序

所有句柄图形对象具有一个至今还未阐述过的 **'ButtonDownFcn'** 属性，本节就进行讨论。**uimenu**和**uicontrol**均有 **'CallBack'** 属性，这个属性是菜单和控制框的应用核心。另外，图形有 **'KeyPressFcn'** 和 **'ResizeFcn'** 属性以及 **'WindowButtonUpFcn'** 和 **'WindowButtonMotionFcn'** 属性。与这些属性相关的值是回调字符串，即MATLAB字符串，当属性激活时，它传给**eval**。指针的位置确定事件涉及到哪些回调以及当事件发生时它们被激励的顺序。

前一章讨论过堆积顺序和与讨论相关的对象选择区域。根据图形中3个区域，MATLAB决定哪一个回调将被激励。如果指针是在句柄图形对象内，如同对象的 **'Position'** 属性所确定的那样，则指针就是在对象**上**。如果指针不在一个对象上，而在对象的选择区域内，则指针**靠近**对象。如果指针在图形内但既不靠近也不在另一对象之上，则可以认为指针关掉其它对象。如果若干对象及其选择区域相重叠，重叠顺序就决定了选择顺序。

句柄图形的线、曲面、补片、文本和坐标轴对象的选择区域已在上一章讨论过了。**uimenu**对象没有外部的选择区域，指针要么在**uimenu上**，要么不在其上。**uicontrol**对象越过图基位置，在各方向延伸大约5个象素有一个选择区域，指针可以**靠近**或在控制框**上**。记住这里讨论的堆积顺序和选择区域是针对4.2c版本的，在以后的MATLAB有某种程度的变化。

按钮点击

按钮点击可以定义为当鼠标指针在同一对象上时，揿下并随后松开鼠标按钮。如果鼠标指针在**uicontrol**或**uimenu**项上，按钮点击触发对象 **'Callback'** 属性字符串的执行，按钮揿下使控制框作好准备，并常常在视觉上改变**uicontrol**或**uimenu**，松开按钮触发回调。当鼠标指针不在**uicontrol**或**uimeun**上，**揿下按钮**和**松开按钮**事件，将在后面讨论。

揿下按钮

当鼠标指针位于一个图形窗口内，揿下鼠标按钮，根据指针位置和对句柄图形对象靠近程度将会发生不同的动作。如果一个对象被选择，它就变成了当前的对象，并上升到堆积顺序的最高端。如果没有对象被选择，图形本身就是当前对象，图形的 **'CurrentPoint'** 和 **'SelectionType'** 属性都被更新，然后激发适当的回调。表格21.3列出了指针位置与按钮揿下所激发的回调：

表21.3

指针位置	所激发的属性
在**uimenu** 或**uicontro**项**上面**	准备释放事件
在句柄图形**上**，或**接近**控制框	图形的**WindowButtonDownFcn**属性，然后是对象的**ButtonDownFcn**属性
在图形内，但不**在**或**接近**任何对象	图形的**WindowButtonDownFcn**属性，然后是图形的**ButtonDownFcn**属性

注意：按钮揿下事件总是在所选对象的 **'ButtonDownFcn'** 回调之前，引起图形的 **'WindowButtonDownFcn'** 回调，除非指针是在**uicontrol**或**uimenu**对象上。如果指针靠近控制框，则在图形的 **'WindouButtonDownFcn'** 回调完毕后，引起控制框的 **'ButtonDownFcn'** 回调而不是 **'Callback'** 属性回调。

按钮松开

当松开鼠标按钮时，图形的 **'CurrentPoint'** 属性就更新。如果没有定义 **'WindowButtonUpFcn'** 回调，则鼠标按钮松开时，属性 **'CurrentPoint'** 不更新。

指针的移动

当指针在图形内移动时，图形的 ' **CurrentPoint** ' 属性被更新，引起图形的 ' **WindowButtonMotionFcn** ' 回调；如果没有定义图形的 ' **WindowButtonMotionFcn** ' 回调，则指针移动时，属性 ' **CurrentPoint** ' 不更新。

回调的复合能产生许多有趣的效应。试一下包含在MATLAB于**demo**目录中的函数**sigdemo1**，**sigdemo2**，就可解其中的一些效果。另外将要讨论的一个例子是 精通MATLAB工具箱的函数**mmdraw**。

21.7 中断回调的规则

一旦回调开始执行，通常都在下一个回调事件处理之前运行完毕。将 ' **Interruptible** ' 属性设置为 ' **yes** ' 可改变这种缺省行为，从而当正在执行的回调遇到**drawnow**，**figure**，**getframe**或**pause**命令时，允许处理的回调事件悬挂起来。事件队列执行计算操作或设置对象属性的命令一经发出，MATLAB便进行处理；而涉及图形窗口输入或输出的命令则生存事件。事件包括产生回调的指针移动和鼠标按钮动作，以及重新绘制图形的命令。

回调处理

回调在达到**drawnow**，**getframe**，**pause**或**figure**命令之前一直执行。注意**gcf**和**gca**引起**figure**命令，而精通MATLAB工具箱中的函数**mmgcf**和**mmgca**不引发**figure**命令。不含有这些特殊命令的回调不会被中断，一旦达到这些特殊命令之一，停止执行回调，将其悬挂起来；并检查事件队列中每一个悬挂的事件。如果产生悬挂回调的对象的**Interruptible**属性设为 ' **yes** ' ，则在被悬挂的回调恢复之前按序处理所有悬挂。如果**Interruptible**属性设为 ' **no** ' ，即缺省值，则只处理悬挂的重画事件，放弃回调事件。

防止中断

即使正在执行回调是不能被中断的，当回调达到**drawnow**，**figure**，**gefframe**或**pause**命令时，仍然处理悬挂的重画事件。通过避免在回调中使用所有这些特殊命令，消除此类事情。如果回调中需要这些特殊命令，但又不要任何悬挂事件，甚至是刷新事件，来中断回调，则可以使用如下所讨论的特殊形式的**drawnow**。

Drawnow

Drawnow命令迫使MATLAB更新屏幕，只要MATLAB回到命令提示，或执行**drawnow**，**figure**，**getframe**或**pause**命令，屏幕就更新。**drawnow**的特殊形式**draunow(' discard ')**使事件队列中所有事件的放弃。在回调中将**drawnow(' discard ')**包含在一个特殊命令之前，就含有清除事件队列的效果，防止刷新事件，以及回调事件中断回调。

21.8 M文件举例

精通MATLAB工具箱中一些函数阐明了上面所讨论的若干技术。第一个例子**mmview3d**应用全局变量和递归函数调用，把方位角和仰角滑标加到图形中。函数中有大量的对象，但函数很直观。因为**mmview3d**文件相当得长，故分段表示。第一段定义了函数标号，帮助文本和全局变量。

```
function mmview3d(cmd)
%  MMVIE3D GUI controllled Azimuth and Elevation adustment.
%  for adjusting azimuth and elevation using the mouse.
%
%  The  ' Revert '  pushbutton reverts to the original view.
```

```
% The 'cmd' argument executes the callbacks.

% Copyright (c) 1996 by Prentice-Hall， Inc.

global Hc_acur Hc_esli Hc_ecur CVIEW
```

第二段处理初始用户的调用，建立必要的**uicontrol**对象并把回调定义为递归函数调用。

```
if nargin==0

%-----------------------------------------------------------------
% Assign a handle to the current figure window.
% Get the current view for slider initial values.
% Use normalized uicontrol units rather than the default 'pixels'.
%-----------------------------------------------------------------

Hf_fig=gcf；
CVIEW=get(gca， 'View')；
if abs(CVIEW(1))>180， CVIEW(1)=CVIEW(1)-(360*sign(CVIEW(1)))；end
set(Hf_fig， 'DefaultUicontrolUnits'， 'normalized')；

%-----------------------------------------------------------------
% Define azimuth and elevation sliders.
% The position is in normalized units (0-1).
% Maximum， minimum， and initial values are set.
%-----------------------------------------------------------------

Hc_asli=uicontrol(Hf_fig， 'style'， 'slider'， ...
'position'，[.09 .02 .3 .05]， ...
'min'，-180， 'max'，180， 'value'， CVIEW(1)， ...
'callback'， 'mmview3d(991)')；

Hc_esli=uicontrol(Hf_fig， 'style'， 'slider'， ...
'position；，[.92 .5 .04 .42]， ...
'min'，-90， 'max'，90， 'val'，CVIEW(2)， ...
'callback'， 'mmview3d(992)')；

%-----------------------------------------------------------------
% Place the text boxes showing the minimum and maximum values at the
% ends of each slider， These are text displays， and cannot be edited.
%-----------------------------------------------------------------

uicontrol(Hf_fig， 'style'， 'text'， ...
'pos'，[.02 .02 .07 .05]， ...
'string'，num2str(get(Hc_asli， 'min')))；

uicontrol(Hf_fig， 'style'， 'text'， ...
'pos'，[.39 .02 .07 .05]， ...
'string'，num2str(get(Hc_esli， 'min')))；

uicontrol(Hf_fig， 'style'， 'text'， ...
'pos'，[.915 .92 .05 .05]， ...
'string'，num2str(get(Hc_esli， 'max')))；
```

```
%--------------------------------------------------------------------
% Place labels for each slider
%--------------------------------------------------------------------

uicontrol(Hf_fig，'style'，'text'，...
'pos'，[9.095 .08 .15 .05]，...
'string'，'Azimuth')；

uicontrol(Hf_fig，'style'，'text'，...
'pos'，[.885 .39 .11 .05]，...
'string'，'Elevation')；

%--------------------------------------------------------------------
% Define the current value text displays for each slider，
%--------------------------------------------------------------------
% These are editable text display the current value
% of the slider and at the same time allow the user to enter
% a value using the keyboard.
%
% Note that the text is centered on XWindow Systems，but is
% left-justified on MS-Windows and Macintosh machines.
%
% The initial value is found from the value of the sliders.
% When text is entered into the text area and the return key is
% pressed， the callback string is evalueated.
%--------------------------------------------------------------------

Hc_acur=uicontrol(Hf_fig，'style'，'edit'，...
'pos'，[.25 .08 .13 .053]，...
'string'，num2str(get(Hc_asli，'val'))，...
'callback'，'mmview3d(993)')；

Hc_ecur=uicontrol(Hf_fig，'style'，'edit'，...
'pos'，[.885 .333 .11 .053]，...
'string'，num2str(get(Hc_esli，'val'))，...
'callback'，'mmview3d(994)')；

%--------------------------------------------------------------------
% Place a 'Done' button is the lower right corner.
% When clicked， all of the uicontrols eill be erased.
%--------------------------------------------------------------------

uicontrol(Hf_fig，'style'，'push'，...
'pos'，[.88 .02 .10 .08]，...
'backgrouncolor'，[.7 .7 .8]，...
'string'，'Done'，...
'callback'，'delete(finobj(gcf，''Type''，''uicontrol''))')；

%--------------------------------------------------------------------
% Place a 'Revert' button next to the 'Done' button.
% When clicked， the view reverts to the original view.
%--------------------------------------------------------------------
```

```
Hc_ecur=uicontrol(Hf_fig，'style'，'edit'，...
'pos'，[.77 .02 .10 .08]，...
'backgroundcolor'，[.7 .7 .8]，...
'string'，'Revert'，...
'callback'，'mmview3d(995)')；
```

现在已建立了控制框并定义了回调。

```
else

%--------------------------------------------------------------------
%  The callbacks for the azimuth and elevation sliders；
%--------------------------------------------------------------------
%  1)  get the value of the slider and display it in the text windows
%  2)  set the 'View' property to the current values of the azimuth
%      and elevation sliders.
%--------------------------------------------------------------------

if  cmd==991
   set(Hc_acur，'string'，num2str(get(Hc_asli，'val')))；
   set(gca，'View'，[get(Hc_asli，'val')，get(Hc_esli，'val')])；

elseif  cmd==992
   set(Hc_ecur，'string'，num2str(get(Hc_esli，'val')))；
   set(gca，'View'，[get(Hc_asli，'val')，get(Hc_esli，'val')])；

%--------------------------------------------------------------------
%  The 'slider current value' text display callbacks；
%--------------------------------------------------------------------
%  When text is entered into the text area and the return key is
%  pressed，  the entered value is compared to the limits.
%
%  If the limits have been exceeded，  the display is reset to the
%  value of the slider and an error message is displayed.
%
%  If the value is within the limits，  the slider is set to the
%  new value，  and the view is updated.
%--------------------------------------------------------------------

elseif cmd==993
   if str2num(get(Hc_acur，'string')) < get(Hc_asli，'min)...
    | str2num(get(Hc_acur，'string')) >get(Hc_asli，'max')
      set(Hc_acur，'string'，num2str(get(Hc_asli，'val')))；
      disp('ERROR - Value out of range')；
   else
     set(Hc_asli，'val'，str2num(get(Hc_acur，'string')))；
     set(gca，'View'，[get(Hc_asli，'val')，get(Hc_esli，'val')])；
   end
elseif cmd==994
   if str2num(get(Hc_ecur，'string')) < get(Hc_esli，'min)...
    | str2num(get(Hc_ecur，'string')) >get(Hc_esli，'max')
      set(Hc_ecur，'string'，num2str(get(Hc_esli，'val')))；
      disp('ERROR - Value out of range')；
```

```
    else
      set(Hc_esli，'val'，str2num(get(Hc_ecur，'string')))；
      set(gca，'View'，[get(Hc_asli，'val')，get(Hc_esli，'val')])；
    end

%-----------------------------------------------------------------
%  Revert to the original view.
%-----------------------------------------------------------------

elseif cmd==995
    set(Hc_asli，'val'，CVIEW(1))；
    set(Hc_esli，'val'，CVIEW(2))；
    set(Hc_acur，'sting'，num2str(get(Hc_asli，'val')))；
    set(Hc_ecur，'sting'，num2str(get(Hc_esli，'val')))；
    set(Hc_acur，'View'，[get(Hc_asli，'val')，get(Hc_esli，'val')])；

%-----------------------------------------------------------------
%  Must be bad arguments.
%-----------------------------------------------------------------

  else
    disp('mmview3d：  Illegal argument.')
  end
end
```

第二个例子**mmcxy**，在图形左下角建立一个小文本框，当指针处在图形中时，显示图形内鼠标指针的坐标**[x，y]**。点击鼠标清除坐标的显示。

虽然**mmcxy**是一个短小的函数，它仍然利用许多有效的GUI函数的元素，包括递归函数调用、全局变量和'**UserData**'属性。此例还说明图形的'**WindowButtonDownFcn**'和'**WindowButtonMotionFcn**'属性起动回调的用法。

```
function out=mmcxy(arg)
%  MMCXY Show x-y Coordinates of the mouse in the
%  lower left hand corner of the current 2-D figure window.
%  When the mouse is clicked，  the coordinates are erased.
%  XY=MMCXY returns XY=[x，y] coordinates where mouse was clicked.
%  XY=MMCXY returns XY=[] if a key press was used.

%  Copyright (c) by Prentice-Hall，  Inc.

global MMCXY_OUT

if  -nargin
    Hf=mmgcf；
    if  isempty(Hf)，  error('No Figure Available.')，  end
    Ha=findobj(Hf，'Type'，'axes')；
    if  isempty(Ha)，  error('No Axes in Current Figure，')，end.

    Hu=uicontrol(Hf，'Style'，'text'，...
                'units'，'pixels'，...
                'Position'，[1  1  140  15]，...
                'HorizontalAlignment'，'left')；
    set(Hf，'Pointer'，'crossh'，...
```

```
        'WindowButtonMotionFcn', 'mmcxy(''move'')', ...
        'WindowButtonDownFc', 'mmcxy(''end'')', ...
        'Userdata', Hu)
    figure(Hf)  %bring figure forward
    if nargout  %must return x-y data
       key=waitforbuttonpress;  %  pause until mouse is pressed
       if  key
          out=[];          %return empty if aborted
          mmcxy('end')   %clean thins up
       else
          out=MMCXY_OUT;  %now that move is complete return point
       end
       return
    end

elseif strcmp(arg, 'move')     %  mouse is moving in figure window
    cp=get(gca, 'CurrentPoint');  %  get current mouse position
    MMCXY_OUT=cp(1, 1: 2);
    xystr=sprintf('[%.3g]', MMCXY_OUT);
    Hu=get(gcf, 'Userdata');
    set(Hu, 'String' xystr)  %  put x-y coordinaates in text box

elseif strcmp(arg,  'end')  %  mouse click occurred,  clean things up
    Hu=get(gcf,  'Userdata');
    set(Hu, 'visible', 'off')  %  make sure text box disappears
    delete(Hu)
    set(gcf, 'Pointer', 'arrow', ...
         'WindowButtonMotionFcn', '', ...
         'WindowButtonDownFcn', '', ...
         'Userdata', [])
end
```

第一次被调用时，**mmcxy**建立文本**uicontrol**，改变指针形状，设定'**WindowButtonDownFcn**'和'**WindowButtonMotionFcn**'的回调，然后等待按键或先撤按钮。若有键撤下，就调用清除（cleanup）程序，清除文本框，恢复鼠标指针，清除图形回调及'**UserData**'属性。若点击鼠标按钮，'**WindowButtonDownFcn**'回调就处理清除任务。在等待时，图形中鼠标指针的移动会触发'**WindowButtonMotionFcn**'回调，更新uicontrol中文本串。

精通MATLAB工具箱中的另一个函数是**mmtext**，它利用'**WindowButtonDownFcn**'，'**WindowButtonUpFcn**'和'**WindowButtonMotionFcn**'回调的另一个短小函数以安置和拖曳文本。

```
function mmtext(arg)
%  MMTEXT place and drag text with mouse
%  MMTEXT waits for a mouse click on a text object
%  in the current figure then allows it to be dragged
%  while the mouse button remains down.
%  MMTEXT('whatever') places the string 'whatever' on
%  the current axes and allows it to be dragged.
%
%  MMTEXT becomes inactive after the move is complete or
%  no text object is selected.

%  Copyright (c) 1996 by Prentice-Hall,  Inc.
```

```
if -nargin， arg=0；end

if isstr(arg) % user entered text to be placed
   Ht=text('units'， 'normalized'， ...
      'Position'，[.05 .05]，...
      'String'，arg，...
      'HorizontalAlignment'， 'left'，...
      'VerticalAlignment'， 'baseline')；
   mmtext(0) % call mmtext again to drag it

elseif arg==0 % initial call， select text for dragging
   Hf=mmgcf；
   if isempty(Hf)， error('No Figure Available.')， end
   set(Hf， 'BackingStore'， 'off'， ...
      'WindowButtonDownFcn'， 'mmtext(1)')
   figure(Hf) % bring figure forward

elseif arg==1 & strcmp(get(gco， 'Type')， 'text') % text object selected

set(gco， 'Units'， 'data'，...
        'HorizontalAlignment'， 'left'，...
```

mmdraw是精通MATLAB工具箱中另一个有用GUI函数，它与**mmtext**十分相似，但更为复杂，此函数允许用户用鼠标在当前的坐标轴上画线并设置线的属性。

```
function mmdraw(arg1，arg2，arg3，arg4，arg5，arg6，arg7)
%MMDRAW Draw a Line and Set Properties Using Mouse.
%MMDRAW Draw a Line in the current axes using to the mose.
%Click at the starting point and drag to the end point
%In addition，properties can be given to the line.
%Properties are given in pairs，e.g.，MMDRW name vale...
%Properties：
%NAME        VALUE      (defult)
%color       [y m c r g b {w} k] or an rgb in quotes：  '[r g b]
%style       [-----{：  }--.]
%mark        [0 + . * x]
%width       points for linewidth {0.5}
%size        points for marker size {6}
%Examples：
%MMDRAW color r width 2 sets color to red and width to 2 points%MMDRAW mark +
size 8  set marker type to +and size to 8 points
%MMDRAW color  '[1 5 0]'  set color to orange

%Copyright (c) 1996 by Prentice-Hall，Inc.

global MMDRAW_Hl MMDRAW_EVAL

if nargin==0
     arg1='color'； arg2='w'； arg3='style'； arg4='：  '； nargin=4；
end

if isstr(arg1) % initial call，set thing up
```

```
    if isempty(Hf)，error('No Figure Available.')，end
    Ha=findobj(Hf，'Type，'axes')；
    if isempty(Ha)，error('No Axes in Current Figure.')，end
    set(Hf，'Pointer'，'Crossh'，...%set up callback for line star
        'BackingStore'，'off'，...
        'WindowButtonDownFcn'，'mmdraw(1)')
    figure(Hf)
    MMDRAW_EVAL='mmdrw(99)；%set up string to set attributes
    for i=1：nargin
      argi=eval(sprintf('arg%.of'，i))；
      MMDARW_EVAL=[MMDARW_EVAL '，''arg''''']；
    end
    MMDARW_EVAL=[MMDARW_EVAL，)']；

elseif arg1==1 % callback is line start point
    fp=get(gca，'CurrentPoint')；    %start point
    set(gca，'Userdata'，fp(1，1：2))  %store in axes userdata
    set(gcf，'WindowButtonMotionFcn'，'mmdraw(2)'，...
        '    'WindowButtonUpFcn'，'mmdraw(3)')

elseif arg1==2 % callback is mouse motion
    cp=get(gca，'CurrentPoint')；cp=cp(1，1：2)；
    fp=get(gca，'Userdata')；
    H1=line('Xdata'，[fp(1)；cp(1)]，'Ydata'，[fp(2)；cp(2)]，...
        'EraseMode'，'Xor，...
        'Color'，'w'，'LineStyle'，'：'，...
        "Clipping'，'off'；
    if ~isempty(MMDRAW_HL) % delete prior line if it exsts
      delete(MMDRAW_HL)
    end
    MMDRAW_HL=H1；% store current line handle

elseif arg1==3 % callback is line end point，finish up
    set(gcf，'Point'，'arrow，...
        'BackingStore'，'on'，...
        'WindowButtonDownFcn'，''，...
        'WindowButtonMotionFcn'，'mmtext(2)'，...
        'WindowButtonUpFcn'，'')
    set(gca，'Userdata'，[])
    set(MMDRAW_HL，'EraseMode'，'normal') % render line better
    eval(MMDRAW_EVAL)
    MMDRAW_EVAL=[]；

elseif arg1==99 % process line properties
    for i=2：2：nargin-1
      name=eval(sprintf('arg%.of'，i)，[])；% get name argument
      vale=eval(sprintf('arg%.of'，i+1)，[])；% get value argument
      if strcmp(name，'color')
        if value(1)=='['，value=eval(value)；end
        set(MMDRAW_HL，'color'，value)
      elseif strcmp(name，'style')
        set(MMDRAW_HL，'Lineatyle'，value)
```

```
            elseif strcmp(name， 'mark')
               set(MMDRAW_HL， 'Linestyle'， value)
            elseif strcmp(name， 'width')
               value=abs(eval(value))；
               set(MMDRAW_HL， 'LineWidth'， value)
            elseif strcmp(name， 'size')
               value=abs(eval(value))；
               set(MMDRAW_HL， 'MarkerSize' '， value)
            end
        end
        MMDRAW_HL=[]；
    end
```

虽然这里说明太长，但精通MATLAB工具箱中的函数**mmsetc**和**mmsetf**都是使用递归、全局变量和'**UserData**'属性的GUI函数的直观例子。也许愿意看一下**mmsetclr M**文件，它是函数**mmsetc**文件的脚本M文件。比较这两个文件，可以了解到，为实现各种GUI M文件要做出一些折衷。

21.9 对话框和请求程序

MATLAB具有建立对话框和'请求'的几个有用工具。对话框是弹出显示的单独窗口，它显示信息字符串。对话框含有一个或多个按钮键以供用户输入。请求框是在弹出显示的单独窗口，利用鼠标或键盘获得输入，并返回信息给调用函数。

对话框

所有MATLAB的对话框都是基于函数**dialog**，它的帮助文本如下

```
>>help diolog
DIALOG displays a dialog box.

    FIG = DIALOG (pl， vl....)displays a dialog box.
    valid param/value pairs include
    Style               error | warning | help | question
    Name                string
    Replace       on | off
    Resize        on | off
    BackgroundColor ColorSpec
    ButtonString        'Button1String | Button2String | ... '
    ButtonCalls         'ButtonCallback | Button2Callback |... '
    ForegroundColor ColorSpec
    Position      [x y width height]
                  [x y] - centers around screen point
    TextString          string
    Units               pixels | normal | cent | inches | points
    UserData            matrix

    Note： Until dialog becomes built-in， set and get
    are not valid for dialong objects.
    At most three buttons are allowed.
    The callbacks are ignored for"question"dialogs.
    If ButtonStrings /ButtonCalls are unspecified then it
    defaults to a single "ok" button which removes the figure.
    There ' s still problems with making the question dialog modal.
```

The entire parameter name must be passed in.
(i.e. no automatic completion).
Nothing beeps yet.
See also ERRORDLG，HELPDLG，QUESTDLG

注意：对话窗本身不是句柄图形对象，而是由一系列句柄图形对象构成的M文件。对话窗口是图形，包括与框架、编辑和按钮**uicontrol**对象共存的坐标轴。在将来的版本中，**dialog**可能成为具有更多功能的内置函数。缺省的对话框是一个帮助对话框，它是由 ' **Default help string** ' 字符串和标有 ' OK ' 的按钮键组成的编辑文本框。作为例子，键入>> **dialog**。

预先定义的对话框是由函数**helpdlg**，**enordlg**，**warndlg**和**gucstdlg**建立。**helpdlg和warndlg**接受文本字符串和窗口标题字符串作为输入参量。**errordlg**接收 ' **Replace** ' 变量作为输入。除了**questdlg**，所有上述函数都产生类似的对话框，它有各自缺省的标题和文本字符串。标有 ' **OK** ' 的单个按钮，则关闭对话框。**helpdlg**帮助文本是：

>> help helpdlg

HELPDLG： Displays a help dialog box.
HANDLE=HELPDLG(HELPSTRING，DLGNAME) displays the
message HelpString in a dialog box with title DLGNAME.
if a help dialog with that name is aldeady on the screen，
it is brought to the front.Otherwise it is created.

See also：DIALOG

帮助信息
Helpdlg：显示一个帮助文本框
HANDLE = HELPDLG(HELPSTRING，DLGNAME)在对话框中显示标题为dlgname的帮助信息helpstring。如果名为dlgname的帮助对话框已在屏幕上显示，则引到屏幕正面，否则就建立该帮助对话框。

参阅：dialog

---

warndlg的帮助文本是：

>>help warndlg
WARNDLG Creates a warning dialog box.
HANDLE=WARNDLG(WARNSTR，DLGNAME) creates a warning
diolog box which displays WARNSTR in a window named
DLGNAME.A pushbutton labeled OK must be pressed to
make the warning box disappear.

See also：DIALOG

errordlg的帮助文本是

>>help errordlg

ERRORDLG Creates an error dialog box
HANDLE = ERRORDLG(ERRORSTR，DLGNAME，Replace) creates an error dialog box which diplays ERRORSTR in a window nnnamed DLGNAME.A pushbutton labeled OK must be pressed to make the error box disappear.If REPLACE= '

on ' and an error dialog with Name DLGNAME already exits，it is simply brought to the front (no new dialog is created).

See also ： DIALOG

帮助信息
ERRORDLG：建立出错对话框
HANDLE=ERRORDLG(ERRORSTR，DLGNAME，REPLACE)建立显示出错信息errorstr、名为dlgname的出错对话框，要消除出错信息对话框，必须揿下标记为OK的按钮，如果replace= ' on ' 并且名为digname的出错对话框已经存在，则就引到屏幕正面，不再建立新的对话框。

参阅：dialog

提问的对话框稍有不同，前三种函数只显示一个按钮键并返回对话框图形对象的句柄。函数**questdlg**显示两个或三个按钮键，返回由用户所选的按钮的标志字符串。**questdlg**的帮助文本如下：

```
>>help qeustdlg

QEUSTDLG Creates a question dialog box.
    CLICK=QUESTDLG(Q，YES，O，CANCEL，DEFAULT) creates a question
    dialog box which displays Q.Up to three pushbuttons，
    with strings given by YES，NO，and CANCEL，will appear
    along with Q in the dialog.The dialog wil be destroyed
    returning the string CLICK depending on which button is
    clicked.DEFAULT is the default button number.
```

帮助信息
QUESTDLG：建立问题对话框
CLICK=QVESTDLG(Q，YES，NO，CANCEL，DEFAULT)建立显示信息Q的问题对话框。至多有三个按钮具有由YES，NO和CANCEL给定的字符串，按钮与Q一起，显示在对话框中。根据所击的按钮返回字符串CLICK对话框消失。DEFAULT是缺省的按钮数。

下面是函数的片断，说明了函数中提问对话框的应用：

```
question1= ' Change color map to copper? ' ;
response1=questdlg(question1， ' Sure ' ， ' Nope ' ， ' Maybe ' ， 2);
if stromp(response1， ' Sure ' )
  colormap(copper);
elseif stromp(response1， ' Maybe ' )
  wandlg( ' That response does not compute! ' );
  response2=questdly([ ' Please make up your mind.｜ ' question1]， ' Yes ' ， ' No ' );
  if stromp (response2， ' Yes ' )
     colormap(copper);
  end
end
```

请求程序

请求程序通过对话框获取用户的输入。请求程序是内置式GUI函数，它使用平台原有的窗口系统建立外观熟悉的请求程序。

函数**uigecfile**和**uiputfile**是所有平台上都有的内置式函数，用于交互地获得文件名，从而调用函数用它读取文件中数据或将数据存于文件中。**uigetfile**的帮助文本是：

```
>>help uigetfile

UIGETFILE Interactively retrieve a filename by displaying a dialog box.
    [FILENAME，PATHNAME]=UIGETFILE(' filterSpec '， ' dialogTitle '， X，Y)
    displays a dialog box for the user to fill in，and returns the
    filename and path strings.A successful return occurs only if
    the file exists.If the user selects a file that does not exist，
    an error message is displayed，and control returns to the dialog box.
    The user may then enter another filename，or press the Cancel button.

    All parameters are optional，but if one is used，all previous
    parameters must also be used.

    The felterSpec parameter determines the initial display of files in
    the dialog box.For example ' *.m ' lists all the MATLAB M-files.

    Parameter ' dialogTitle ' is a string containing the title of the dialog
    box.

    The X and Y parameters define the initial position of the dialog
    box in units of pixels. Some systems may not support this option.

    The output variable FILENAME is a string containing the mane of the
    file selected in the dialog box.If the user presses the Cancel button
    or if any error occurs，it is set to 0.

    The output parameter PATHNAME is a string containing the path of
    the file selected in the dialog box. If the user presses the Cancel
    button or if any error occurs，it is set to 0.

    See also UIPUTFILE.
```

```
帮助信息
UIGETFILE：通过显示对话框交互式地检索文件名
    [FILENAME，ATHNAME]=UIGETFILE(' filterspec '， ' dialogtitle '，x，y)显示一
    个对话框，让用户输入，并返回路径和文件名字符串。仅当文件存在时，才成功地
    返回。如果用户选择了一个并不存在的文件，就显示出错信息，控制框返回到对话
    框。用户可以输入另一个文件名或掀下Cancel按钮。

    所有输入参数都是可任选的，如果用其中之一个，也必须使用所有先前参数。
    参数filteospec决定对话框中文件的初始显示。例如 ' *.m ' 列出的所有M文件。
    参数 ' dialogtitle ' 是对话框标题字符串。
    以象素为单位参数x，y定义对话框的初始位置，有些系统可能不支持这个选项。
    输出变量filename是对话框内所选文件的名称字符串。如果用户掀了取消按钮或有
    错误发生，filename的值设置为0。
```

输出参数pathname是对话框内所选文件的路径名字符串。如果用户撤了取消按钮或有错误发生，pathname的值设置为0。
参阅 uiputfile

下面的例子说明了在函数中如何利用**uigetfile**，交互式地检索ASCII码数据文件，并绘制正弦数据

```
% Ask the user for a file name.

[datafile datapath]=uigetfile('.dat', 'Choose a data file');

% If the user selected an existing file，read the data.
% (The extra quotes avoid problems with spaces in file or path names
% on the Macintosh platform.)
% Then determine the variable name from the file name，
% copy the data to a variable，and plot the data.
if datafile
    eval(['load(' '' datapath datafile '' ')']);
    x=eval(strtok(datafile.'.'));
    plot (x，sin(x));
end
```

请求程序不接受并不存在的文件名。这种情况可能发生的原因是用户在请求程序中把文件名输入到文件框中。Macintosh版本无文本框的，用户必须选择一个存在的文件或撤下**Cancel**按钮以退出请求程序。

函数**uiputfile**与函数**uigetfie**十分相似：输入参量相似，而且两者均返回文件和路径字符串，**uiputfile**的帮助文件是

```
>> help uiputfile

UIPUTFILE Interactively retrieve a filename by displaying a dialog box.
[FILENAME，PATHNAME]=UIPUTFILE('inifFile', 'dialogTitle')
displays a dialog box and returns the filename and path strings.

The initFile paramete determines the initial display of files
in the dialog box.Full file name specifications as well as wildcards
are allowed.For example，'newfile.m' initializes the display to
that particular file and lists all othe existing .m files.This may
be used to provide a default file mane. A wildcard specification such
as '*.m' lists all the existing MATLAB M-files.

Parameter 'dialogTitle' is a string containing the title of the dialog
box.

The output variable FILENAME is a string containing the name of the
file selected in the dialog box. If the user presses the Cancel butto
or if any error occurs，it is set to 0.

The output variable FILENAME is a string containing the name of the
file selected in the dialog box. If the user presses the Concel button
or if any error occurs，it is set to 0.

[FILENAME，PATHNAME]=UIPUTFILE ('initFile', 'dialogTitle'，X，Y)
```

places the dialog box at screen position [X， Y] in pixel units.
Not all systems support this option.

Example：
[newmatfile， newpath]=uiputfile (' *.mat ' . ' Save As ')；

See also UIGETFILE.

如果用户选择了已经存在的文件，则出现一个对话窗，询问用户是否要删除存在的文件。如果回答**no**，则返回原来的请求程序等待另一次尝试；如果回答**yes**，则关闭请求程序和对话框并把文件名返回，文件并未被请求程序删除。如果需要，调用函数必须删除或覆盖文件。

值得牢记的是，这些函数中不论哪一个都未真正地读或写任何文件，调用函数必须做这些工作。这些函数仅仅是将文件名和路径返回给调用函数。

在MS-Windows和Macintosh平台上，**uisetcolor**让用户交互式地选择颜色并选择性地将此颜色施加到一个对象上。如同MATLAB4.2C版本，X Window系统平台不支持**uisetcolor**。

uisetcolor的帮助文本是：

>>help uisetclor

UISETCOLOR Interactively set a ColorSpec by displaying a dialog box.
C=UISETCOLOR(ARG， ' dialogTitle ') displays dialog box for the
user to fill in， and applies the selected color to the input
graphics object.

The parmeters are optional and may be specified in any order.

ARG may be either a handle to a graphics object or an RGB triple.
If a handle is used， it must specify a graphics object that supports
color.If RGB is used， it must be a valid RGB triple (e.g.， [1 0 0]
for red). In both cases， the color specified is used to initialize
initializes the color to black.

If parameter ' dialogTitle ' is used， it is a string containing the
title of the dialog box.

The output value C is the selected GB triple. If the input parmeter
is a handle， the graphics object ' s color is set to the RGB color
selected.

If the user presses Cancel from the dialog box， or if any error
occurs， the output value is set to input RGB triple， if provided；
otherwise， it is set to 0.

Example：
C=uisetcolor(hText， ' Set Text Color ')

NOTE：This function is only available in MS-Windows and Macintosh
versions of MATLAB.

帮助信息
UISETCOLOR ： 显示对话框，交互式地设置ColorSpace。

C=UISETCOLOR(ARG，'dialogtitle')显示一个对话框，让用户输入，并将所选颜色用于输入的图形对象。
参数是可任选，并可以以任何次序指定。

ARG可以是一个图形对象的句柄或是RGB3元组。如果使用句柄，必须指定支持颜色的图形对象；如果使用RGB，必须是有效的RGB3元组(如：[1 0 0]是红色)。在这两种情况下，所指定的颜色用于对话框的初始化。如果没有指定初始的RGB，将对话框初始化为黑色。

如果使用参数'dialogTitle'，该参数是对话框名称的字符串。
输出值C是所选的RGB3元组。如果输入参数是句柄，则图形对象的颜色就设定为所选的颜色。

如果用户撤下对话框中的Cancel按钮或有错误发生，输出值就设定为输入的RGB3元组；如果没有输入RGB3元组，则输出值设置为0。

例如：c=uisetcoior(htext，'settextcoior')

注意：该函数仅在的MS-Windous和Macintosh版本中可用。

精通MATLAB工具箱具有前面所提及的函数**mmsetc**。该函数可在所有的平台上工作，功能与**uisetcolor**十分相似。**mmsetc**甚至可允许用户用鼠标来选择颜色并将所选的颜色加到所选择的对象上。以下就是**mmsetc**的帮助文本：

```
>> help mmsetc

MMSETC Obtain an RGB triple interactively form a color sample.
    MMSETC displays a box for the user to select
    a color nteractively and displays the result.

    X=MMSETC returns the selected color in X.

    MMSETC ([r g b]) uses the RGB triple as the initial
    RGB value for modification.

    MMSETC C -or-
    MMSETC ('C') where C is a color spec (y，m，c，r，g，b，w，k)，uses
    the specified color as the initial value.

    MMSETC(H) where the input argument H is the handle of
    a valid graphics object that supports color，uses the color
    progerty of the object as the initial RGB value.

    MSETC select -or-
    MMSTEC('select') waits for the user to click on a valid
    graphics object that supports color，and uses the color
    property of the object as the initial RGB value.

    If the initial RGB value was obtained from an object or
    object handle，the 'Done' pushbutton will apply the
    resulting color property to the selected object.

    If no initial color is specified，black will be used.
```

Examples：

```
mmsetc
mycolor=mmsetc
mmsetc([.25 .62 .54])
mmsetc(H)
mmsetc g
mmsetc red
mmsetc select
mycolor=mmsetc( ' select ' )
```

下面的例子是利用精通MATLAB工具箱中函数**mmap**和**mmsetc**，交互式地使用单色映象：

```
>>mesh(peaks)

>>coiormap(mmap(mmsetc))
```

在MS-Windows和Machitosh平台上，有最后的请求程序**uisetfont**，它让用户可交互式选择字型属性并将其加于对象上。如同MATLAB4.2c版本，X Window系统平台不支持**uisetfont**。**uisetfont**帮助文件是：

```
>> help uisetfont

UISETFONT  Interactively set a font by displaying a dialog box.
    H=UISETFONT(HIN， ' dialogTitle ' ) displays a dialog box for the
    user to fill in，and applies the selected font to the input
    graphics object.

    The parmeters are optional and may be specified in any order.

    If parameter HIN is used，it must specity a handle tp a text or
    axis graphics object. The font properties currently assigned to
    this object are used to initialize the font dialog box.

    If parameter ' dialogTitle ' is used，it is a string containing the
    title of the dialog box.

    The output H is a handle to graphics object. If HIN is specified，
    H is identical to HIN.If HIN is not specified，a nwe text object
    is created with the selected font properties， and its handle is
    returned.

    If the user presses Cancel from the dialog box，or if any error
    occurs，the output value is set to the input handle，if provided；
    otherwise ，it is set to 0.

    Example：
        uisetfont(hText， ' Update Font ' )

    NOTE：This function is only available in MS-Windows and Macintosh
        versions of MATLAB.
```

精通MATLAB工具箱有一个称为函数**mmsetf**，功能类似于**uisetfont**。但可工作在所有平台上。**mmsetf**甚至可允许用户用鼠标来选择文本对象，然后将所选的字体属性加在所选的对象上。以下就是**mmsetf**的帮助文本：

```
>>help mmsetf

MMSETF Choose font characteristics interactively.
    MMSETF displays a dialog box for the user to select
    fornt characteristics.

    X=MMSETF returns the handle of the text object or 0
    if an error occurs of ' Cancel ' is pressed.

    MMSETF(H) where the input argument H is the handle of
    a valid text or axes object，uses the font characteristics
    of the object as the initial values.

    MMSETF select -or-
    MMSETF ( ' select ' )waits for the user to click on a valid
    graphics object，and uses the font characteristics
    of the object as the initial values.

    If the initial values were obtained from an object or
    object handle，the ' Done ' pushbutton will apply the
    resulting text properties to the selected object.

    If no initial object handle is specified，a zero is returned in X.

    Examples：
    mmsetf
    mmsetf(H)
    mmsetf select
    Hx_obj=mmsetf( ' select ' )
```

然而**mmsetf**中用于字体的选择是受限制的。不同的平台类型，甚至同一类型的不同计算机都可能安装了不同的字体。MATLAB的函数**uisetfont**是置于内部的，并利用计算机平台的操作系统列出可用的字体。因为**mmsetf**是的GUI 函数，不能确定哪种字体是可用的，所以用了通常可获得的有限选择。字体的大小也因同样原因受到限制。

mmsetf请求程序中的样本文本字符串指明了每一个属性改变的作用。选择后，如果出现的文本字符串并不如所希望的那样，则说明所选择的字体属性(如字体名称、大小等等)在所用计算机上是不存在的。见到的即所得到的。

21.10 用户自制的GUI M文件

许多MATLAB用户充分利用的MATLAB所提供的GUI工具，编写了一些有趣GUI函数。这中间大部分可在MATLAB的 匿名**FTP** 地址**/pub/contrib/graphics**路径中得到。Internet资源一章将详细地讨论如何从这一资源库中获得文件的详细指令。

一些M文件和M文件集含有大量的句柄图形GUI对象，并说明了**uimenu**和**uicontrol**的使用。其中让人印象最深刻的是科斯·荣格(Keith Roger)编写的**matdraw**集合。该文件集合可在**/pub/contrib/graphics/matdraw2.0**目录下获得。该文件集将工具条和画图工具调色板加到图形窗口，效果同集成到MATLAB的绘图程序一样。

另一个值得研究的是派瑞克·马查德(Patrick Marchard)编写的**guimaker**集合。这是交互式工具的集合，让用户应用GUI来建立GUI函数。可以用鼠标建立GUI对象，放置GUI对象

和指定GUI对象的大小。版本1.0以freeware形式发布，即它可以无偿使用。2.0版本以及以后的版本以shareware形式发布的，即你可以无偿地使用30天，然后你需注册并付不多的费用。两个版本均可在的 匿名ftp地址**/pub/contrib/graphics**路径中得到。

这里只是现有GUI函数很多例子中的一小部分。核实一下。也许会对别人要用但已消失的GUI应用程序有想法。第23章有详细指导信息帮助用户访问现有的MATLAB资源，甚至帮助用户成为M文件的贡献者。

21.11 小结

图形用户界面设计不是为每一个人的。但如果需要，MATLAB使它有可能仅由M文件来建立令人印象深刻的GUI函数。需用**mex**文件、高级语言或数据库调用对有用的MATLAB函数来建立一个赏心悦目的界面。

对GUI对象的字体控制局限性，在下一版本MATLAB的发行中应有所强调。对话框也有可能在未来成为内置函数。所有这些改进工作将使编写GUI函数更加容易，使它们既与平台无关并且更加悦目。

本章所讨论的函数总结如下

表21.4

句柄图形GUI函数	
uimenu(handle，'PropertyName'，value)	创建或改变图形的菜单
uicontrol(handle，'PropertyName'，value)	创建或改变对象的性质
dialog('PropertyName'，value)	显示对话框
helpdlg('HelpString'，'DlgName')	显示'帮助'对话框
warndlg('WarnString'，'DlgName')	显示'警告'对话框
errordlg('ErrString'，'DlgName'，Replace)	显示'出错对话框
questdlg('Qstring'，S1，s2，s3，Default)	显示'提问'对话框
uigetfile(Filter，Dlgname，X，Y)	交互式地检索文件名
uiputfile(InitFile，Dlgname，X，Y)	交互式地检索文件名
uisetcolor(Handle，Dlgname)	交互式地选择颜色
uisetcolor([r g b]，Dlgname)	交互式地选择颜色
uisetfont(Handle，Dlgname)	交互式地选择字体属性

以下是本章所提到的精通MATLAB工具箱中的函数：

表21.5

精通MATLAB工具箱中的句柄图形GUI函数	
mmenu	**uimenu**函数示例
mmclock(X，Y)	**uicontrol**函数示例
mmsetclr	函数**mmsetc**的有限脚本型式
mmview3d	把方位角和仰角的滑标加到图形上
mmcxy	用鼠标显示的x-y坐标
mmtext('String')	用鼠标放置和拖曳文本
mmdraw('Name'，Value)	用鼠标画线并设置属性
mmsetc(Handle)	用鼠标设置颜色属性
mmsetc(ColorSpec)	
mmsetc('select')	
mmsetf(Handle)	用鼠标设置字体属性
mmsetf('select')	