Построение GUI в MATLAB вручную

Некоторые функции GUI

Пример gca

```
x = linspace(0,10);
y = sin(4*x);
figure;
plot(x,y);
set(gca,'FontSize',12,...
      'TickDir','out',...
      'TickLength',[.02,.02],...
      'YLim',[-2,2])
```

Пример gcf

```
figure
surf(peaks);
set(gcf,'Color',[0,0.5,0.5],...
        'Toolbar','none')
```

Пример `allchild`

```
clc
clear
figure
x=1:0.01:3;
y1=sin(x);
y2=cos(x);
plot(x,y1,x,y2)
get(gca,'Children')
m=allchild(gca)
set(m,'Color','r')
```

Пример `findobj`

```
%добавить строки в предыдущий пример
h = findobj(gca,'Color','r')
set(h(1),'Color','b')
```

Пример `copyobj`

```
figure(9)
h = surf(peaks);
colormap hot
figure           % Create a new figure
axes             % Create an axes object in the figure
new_handle = copyobj(h,gca);
view(3)
grid on
```

Пример `delete`

```
% Добавить последнюю строку к предыдущему примеру
delete(9)
```

Пример `get`

```
clc
clear
figure
x=1:0.01:3;
y1=sin(x);
y2=cos(x);
h=plot(x,y1,x,y2)
get(h)
set(h,'LineWidth',4)
```

Пример `set`

```
clc
clear
figure(123)
plot(peaks)
set(findobj('Type','line'),'Color','k')
```


Простые объекты uicontrol:

errordlg, helpdlg, msgbox, warndlg, inputdlg, questdlg

Пример `warndlg`

```
% Open warning dialog box
warnfig = warndlg('Warning: Something''s not right!', 'Warning')
```

Пример `errordlg`

```
% Create and open error dialog box
errfig = errordlg('You have made an Error!','User Error',...
'replace')
```

Пример `helpdlg`

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

Пример msgbox

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

Пример(2) msgbox .

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

Пример. questdlg

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

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


Пример. inputdlg

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

Пример (2). inputdlg Multiple Input Dialogs

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

Пример . File/Directory Selection Dialog Boxes

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

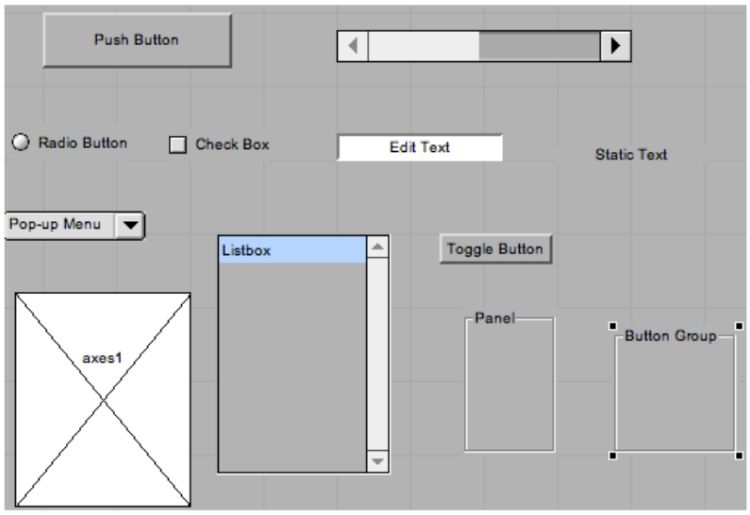
Пример . Setting Fonts and Colours

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

Uicontrol Elements

MATLAB предоставляет ряд базовых элементов графического интерфейса:

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



Чтобы создать элемент управления uicontrol, используйте команду MATLAB:

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

Пример `slider`

```
figure(77)
h_slider = uicontrol('Style','slider',...
'units','normalized',...
'position',[.3 .6 .15 .05]);
```

Пример(2) `slider`.

```
function slidergui(command_str)
% Slider
% Simple Example of creating slider GUIs.
if nargin < 1
command_str = 'initialize';
end
if strcmp(command_str,'initialize')
h_fig = figure(1);  clf;
h_slider = uicontrol(h_fig,...
'callback','slidergui(''Slider Moved'');',...
'style','slider',...
'min',-100,'max',100,...
'position',[25 20 150 20]);
set(h_fig,'userdata',h_slider);
else
```

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

Пример GUI с одной кнопкой

```
function myplotgui
% создание графического окна с заголовком myplotgui и без надписи Figure 1
hF = figure('Name', 'myplotgui', 'NumberTitle', 'off');
% создание осей
hA = axes('Position', [0.1 0.2 0.8 0.7]);
% создание кнопки и задание свойству Callback значения указателя на подфункцию
hBtnPlot = uicontrol('Style', 'pushbutton', ...
     'Position', [20 20 120 30],...
    'String', 'Plot',...
    'Callback', @BtnPlotCallback);

function BtnPlotCallback(src,evt)
% подфункция обработки события Callback кнопки Plot
surf(peaks(30))
```

Значение свойства Style	Что получается
pushbutton (по умолчанию)	кнопка
togglebutton	кнопка-переключатель
radiobutton	переключатель
checkbox	флаг
edit	область ввода текса
text	статический текст
slider	полоса скроллинга
listbox	список
popupmenu	раскрывающийся список

У всех объектов uicontrol многие остальные свойства имеют схожие значения, например: Position, Units, FontName, FontSize. Некоторые свойства имеют разный смысл. Если значение свойства String кнопки или флага - надпись на кнопке или флаге, то для списка это его содержимое. Кстати, событие Callback не единственно возможное. Назначение всех свойств и событий элементов интерфейса, создаваемых функцией uicontrol, описано в справочной системе:

```
>> doc uicontrol
```

Задание дополнительных параметров в функциях обработки событий

В предыдущем разделе мы рассмотрели самый простой способ задания функций обработки событий для объектов приложения с графическим интерфейсом, при котором предполагается, что функции имеют только два входных аргумента: src (указатель на объект, событие которого выполняется) и evt (структура, используемая в некоторых событиях). Тогда при создании объекта достаточно его свойству, связанному с событием, установить в качестве значения указатель на подфункцию, например:

```
h = uicontrol(...,'Callback', @ObjectCallback)
```

а сама функция ObjectCallback должна иметь заголовок

```
function ObjectCallback(src, evt)
```

Если требуется передать дополнительные аргументы в функцию обработки события некоторого объекта, то они указываются после аргумента evt в списке входных аргументов, а при связывании указателя на функцию с событием, задается массив ячеек. Первый его элемент - указатель на функцию обработки события, а остальные - дополнительные аргументы, например:

```
h = uicontrol(...,'Callback', {@ObjectCallback, par1, par2, ...})

function ObjectCallback(src, evt, par1, par2, ...)
```

Вот простой пример приложения myplotgui2, в котором есть две пары осей и две кнопки, нажатие на каждую кнопку приводит к построению графика на осях, расположенных над этой кнопкой.

Подфункция обработки события BtnPlotCallback одна для левой и правой кнопки. Третьим входным аргументом функции BtnPlotCallback является указатель на нужные оси, которые при помощи функции axes делаются текущими при нажатии на кнопку:

Пример GUI myplotgui2

```
function myplotgui2
% создаем графическое окно
hF = figure('Name', 'myplotgui2', 'NumberTitle', 'off',...
    'MenuBar', 'none', 'Units', 'characters');
set(hF, 'Position', [10 10 100 30]);
% создаем две пары осей
hA1 = axes('Position', [0.1 0.2 0.3 0.7]);
hA2 = axes('Position', [0.6 0.2 0.3 0.7]);
% создаем две кнопки, для каждой кнопки функция BtnPlotCallback
% вызывается со своим третьим входным аргументом
hBtnPlot1 = uicontrol('Style', 'pushbutton', 'Units', 'normalized',...
    'Position', [0.1 0.05 0.3 0.05],...
    'String', 'Plot', 'Callback', {@BtnPlotCallback, hA1});
hBtnPlot2 = uicontrol('Style', 'pushbutton','Units','normalized',...
    'Position', [0.6  0.05 0.3 0.05],...
    'String', 'Plot', 'Callback', {@BtnPlotCallback, hA2});

function BtnPlotCallback(src, evt, h)
% подфункция обработки события нажатия на кнопку
axes(h) % делаем нужную пару осей текущими
surf(peaks(30)) % строим график
```


Скрытие указателей объектов приложения с GUI

Для чего желательно скрывать указатели на объекты приложения с графическим интерфейсом? Если приложение работает и пользователь набирает в командной строке, например bar([1 2 3 1 2]), то столбцевая диаграмма построится именно в окне приложения, а вовсе не в отдельном графическом окне (проверьте для приведенных выше приложений myplotgui, myplotgui1 и myplotgui2). Графические функции MATLAB строят график в существующем окне или существующих осях, а новое графическое окно или новые оси создаются тогда, когда окна или осей нет, или MATLAB их не видит. То есть скрытие указателей нужно для предотвращения случайного изменения вида приложения пользователем.

Для скрытия указателей самый простой способ сделать их доступными только при обработке событий объектов приложения. В приведенных выше примерах для этого достаточно в конце основной функции установить свойство HandleVisibility графического окна в 'callback'.

```
function myplotgui
.........
set(hF, 'HandleVisibility', 'callback');
```

Теперь все графические команды, выполняемые из командной строки, создадут новое графическое окно со своими осями и выведут график на них.

Надо иметь ввиду, что у корневого объекта Root есть свойство ShowHiddenHandles, которое по умолчанию установлено в 'off' и видимость указателей определяется значением свойства HandleVisibility каждого графического окна. Если установить свойство ShowHiddenHandles корневого объекта в 'on', то все указатели будут видны вне зависимости от значения свойства HandleVisibility графического окна.

Получение указателей на объекты приложения в функциях обработки событий, функция guihandles.

В подфункциях обработки события элементов интерфейса и объектов приложения с графическим интерфейсом часто требуется получить указатели на другие элементы интерфейса и объекты. В примере myplotgui2 предыдущего раздела мы обошлись всего одной функцией обработки события Callback кнопок для демонстрации вызова функции обработки события с дополнительными параметрами. Для выполнения различных действий при нажатии на правую и левую кнопки приложения (см. рис. 2) придется написать две функции обработки событий Callback - каждую для своей кнопки. При этом возникает следующий вопрос: как в функции обработки события Callback кнопки узнать указатель на нужную пару осей окна приложения.

В MATLAB имеется функция guihandles, которая возвращает структуру с полями, содержащими указатели на все объекты графического окна приложения (поля структуры отделяются точкой от ее имени, например handles.axLeft - поле axLeft структуры handles). Поля этой структуры совпадают со значениями свойства Tag объектов. Поэтому, при создании объекта требуется задавать их свойству Tag некоторое уникальное значение. Причем это значение MATLAB должен воспринимать как имя переменной (чтобы оно могло быть полем структуры). Т.е. например, 'axLeft' или 'ax_Left' могут быть значениями свойства Tag, а '5axLeft' или 'ax-Left' нет. Входным аргументом функции guihandles может являться указатель на любого потомка графического окна приложения. Итак, если при создании осей их свойству Tag было присвоено значение axLeft:

```
axes(...,'Tag', 'axLeft',...)
```

то для получения указателя на них в некоторой функции обработки события другого графического объекта следует использовать guihandles, возвращающую структуру с указателями (имя handles для структуры используется в справочной системе MATLAB и в среде GUIDE, поэтому мы сохраним его):

```
        function ObjectCallback(src, evt)
...
handles = guihandles(src);
% теперь handles.axLeft содержит указатель на оси
...
```

При этом следует иметь ввиду, что при графическом выводе высокоуровневые функции MATLAB (plot, surf и др.) изменяют значения всех свойств осей, кроме Position. Значением свойства Tag становится

пустая строка (как по умолчанию). Так происходит потому, что свойство осей NextPlot по умолчанию установлено в 'replace'. Поэтому при создании осей его следует установить в 'replacechildren':

```
axes(...,'NextPlot', 'replacechildren', 'Tag', 'axLeft',...)
```

что при новом графическом выводе высокоуровневой графической функцией приводит к удалению всех потомков осей, но сохранению значений свойств осей.

В следующем примере приведено приложение, которое выглядит так же, как и предыдущее приложение myplotgui2, но для обработки события Callback каждой кнопки запрограммирована своя функция, в которой нужные оси делаются текущими.

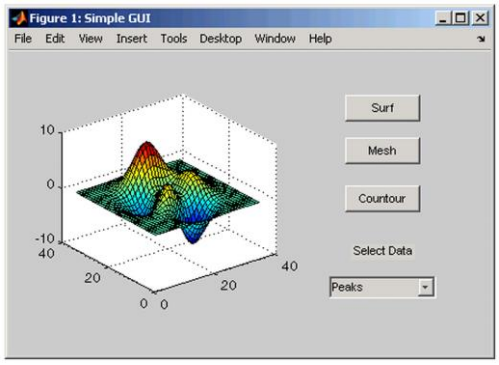
Пример GUI myplotgui3

```
function myplotgui3
% создаем графическое окно с тегом win
hF = figure('Name', 'myplotgui3', 'NumberTitle','off',...
    'MenuBar', 'none', 'Units', 'characters',...
    'Position', [10 10 100 30], 'Tag', 'win');
% создаем оси с тегом axLeft
axes('Position', [0.1 0.2 0.3 0.7], 'Tag', 'axLeft',...
    'NextPlot', 'replacechildren');
% создаем оси с тегом axRight
axes('Position', [0.6 0.2 0.3 0.7],'Tag', 'axRight',...
    'NextPlot', 'replacechildren');
% создаем кнопку с тегом btnLeft
uicontrol('Style', 'pushbutton', 'Units','normalized',...
    'Position', [0.1 0.05 0.3 0.05],...
    'String', 'Plot', 'Callback', @BtnLeftCallback,...
    'Tag', 'btnLeft');
% создаем кнопку с тегом btnRight
uicontrol('Style', 'pushbutton','Units', 'normalized',...
    'Position', [0.6  0.05 0.3 0.05],...
    'String', 'Plot', 'Callback', @BtnRightCallback,...
    'Tag', 'btnRight');
% скрываем указатель на окно приложения
set(hF, 'HandleVisibility', 'callback');

function BtnLeftCallback(src, evt)
% подфункция обработки события нажатия на левую кнопку
% записываем в структуру handles указатели на объекты приложения
handles = guihandles(src);
% сейчас в поле axLeft структуры handles находится указатель на левые оси
% делаем их текущими
axes(handles.axLeft)
barh(rand(5)) % строим график

function BtnRightCallback(src, evt)
% подфункция обработки события нажатия на правую кнопку
% записываем в структуру handles указатели на объекты приложения
handles = guihandles(src);
% сейчас в поле axRight структуры handles находится указатель на правые оси
% делаем их текущими
axes(handles.axRight)
bar(rand(5)) % строим график
```

Пример GUI. Две кнопки

```
function simpleGuiNested

h.fig = figure('position', [800 30 210 60]);

h.buttonOne = uicontrol('style', 'pushbutton',...
'position',[10 10 100 40], ...
'string' , 'Add button', ...
'callback', {@addButton});

function addButton(hObject, eventdata)

h.buttonTwo = uicontrol('style', 'pushbutton', ...
'position',[100 10 100 40], ...
'string' , 'Remove button', ...
'callback', {@removeButton});

set(h.buttonOne, 'enable', 'off');

end

function removeButton(hObject, eventdata)

delete(h.buttonTwo)
h = rmfield(h, 'buttonTwo');

set(h.buttonOne, 'enable', 'on')

end

end
```


Пример Simple_GUI




```
function simple_gui2
% SIMPLE_GUI2 Select a data set from the pop-up menu, then
% click one of the plot-type push buttons. Clicking the button
% plots the selected data in the axes.

%  Create and then hide the GUI as it is being constructed.
f = figure('Visible','off','Position',[360,500,450,285]);

% Construct the components.
hsurf    = uicontrol('Style','pushbutton',...
             'String','Surf','Position',[315,220,70,25],...
             'Callback',{@surfbutton_Callback});
hmesh    = uicontrol('Style','pushbutton',...
             'String','Mesh','Position',[315,180,70,25],...
             'Callback',{@meshbutton_Callback});
```

```
hcontour = uicontrol('Style','pushbutton',...
             'String','Countour','Position',[315,135,70,25],...
             'Callback',{@contourbutton_Callback});
htext  = uicontrol('Style','text','String','Select Data',...
           'Position',[325,90,60,15]);
hpopup = uicontrol('Style','popupmenu',...
           'String',{'Peaks','Membrane','Sinc'},...
           'Position',[300,50,100,25],...
           'Callback',{@popup_menu_Callback});
ha = axes('Units','pixels','Position',[50,60,200,185]);
align([hsurf,hmesh,hcontour,htext,hpopup],'Center','None');

% Initialize the GUI.
% Change units to normalized so components resize automatically.
set([f,hsurf,hmesh,hcontour,htext,hpopup],'Units','normalized');

% Generate the data to plot.
peaks_data = peaks(35);
membrane_data = membrane;
[x,y] = meshgrid(-8:.5:8);
r = sqrt(x.^2+y.^2) + eps;
sinc_data = sin(r)./r;

% Create a plot in the axes.
current_data = peaks_data;
surf(current_data);

% Assign the GUI a name to appear in the window title.
set(f,'Name','Simple GUI')

% Move the GUI to the center of the screen.
movegui(f,'center')

% Make the GUI visible.
set(f,'Visible','on');


%  Pop-up menu callback. Read the pop-up menu Value property to
%  determine which item is currently displayed and make it the
%  current data. This callback automatically has access to
%  current_data because this function is nested at a lower level.
   function popup_menu_Callback(source,eventdata)
      % Determine the selected data set.
      str = get(source, 'String');
      val = get(source,'Value');
      % Set current data to the selected data set.
      switch str{val};
      case 'Peaks' % User selects Peaks.
         current_data = peaks_data;
      case 'Membrane' % User selects Membrane.
         current_data = membrane_data;
      case 'Sinc' % User selects Sinc.
         current_data = sinc_data;
      end
   end

  % Push button callbacks. Each callback plots current_data in the
  % specified plot type.

  function surfbutton_Callback(source,eventdata)
  % Display surf plot of the currently selected data.
       surf(current_data);
  end

  function meshbutton_Callback(source,eventdata)
```

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

  function contourbutton_Callback(source,eventdata)
  % Display contour plot of the currently selected data.
       contour(current_data);
  end
end
```