

Universidade Estadual Paulista

Faculdade de Ciências e Tecnologia

Campus de Presidente Prudente

# CONSTRUINDO INTERFACES GRÁFICAS COM O MATLAB<sup>®</sup>

Alexandre Fioravante de Siqueira – DFQB – FCT – UNESP

Messias Meneguette Junior – DMEC – FCT – UNESP

Presidente Prudente, São Paulo

Agosto de 2010

# Sumário

## Prefácio

<b>1. Interfaces Gráficas com o Usuário no Matlab</b>	<b>1</b>
1.1. O que é uma GUI?	
1.2. Os modos de criação	
1.3. Projetando sua GUI	
<b>2. Guide – O caminho mais fácil até as GUIs</b>	<b>3</b>
2.1. O que é o Guide?	
2.2. Programando a GUI	
<b>3. GUIs e seus componentes</b>	<b>5</b>
3.1. Os componentes básicos de uma GUI	
3.2. O inspetor de propriedades	
<b>4. Criando e modificando menus</b>	<b>11</b>
4.1. A criação de menus e submenus no Guide	
4.2. Propriedades dos menus	
<b>5. Exemplos, exemplos, exemplos</b>	<b>14</b>
<b>6. Para onde ir agora?</b>	<b>21</b>
<b>Referências Bibliográficas</b>	

## Prefácio

O Matlab, criado no fim dos anos 1970 por Cleve Moler (um dos fundadores da *MathWorks, Inc.*), é destinado a cálculos com matrizes. No entanto, é reconhecido mundialmente como uma das melhores ferramentas para o processamento matemático.

Voltado às áreas de engenharia, física, estatística, economia, entre muitas outras, o Matlab não é apenas um pacote de computação e plotagem; este software conta com diversas extensões (as chamadas *toolboxes*) que o tornam muito versátil e poderoso. Algumas dessas extensões, por exemplo, são a *signal processing toolbox*, *image processing toolbox*, *data acquisition toolbox*, *neural network toolbox*.

Com tantas opções disponíveis, o Matlab se tornou uma vantajosa ferramenta que permite, mesmo aos usuários com capacidades elementares de programação, produzir gráficos complexos e resolver problemas de elevado grau de dificuldade.

Softwares profissionais em sua maioria contam com interfaces gráficas intuitivas, de fácil acesso. Linguagens de programação bem conhecidas, como Pascal, Basic, Clipper e COBOL possuem ferramentas voltadas para a criação de interfaces gráficas com o usuário (soluções disponíveis respectivamente, para cada linguagem: Delphi, Visual Basic, FlagShip e AcuCobol). Embora o foco dos programadores Matlab seja a resolução de problemas matemáticos avançados, o Matlab possibilita também a construção de sofisticadas interfaces gráficas.

Este trabalho tem como objetivo fornecer um curso introdutório sobre a criação de interfaces gráficas com o usuário (GUIs), assunto por vezes não conhecido entre os programadores Matlab. Após o estudo do material, espera-se que o programador desenvolva interfaces gráficas básicas, e também consiga se situar na busca de informações mais avançadas sobre a criação de GUIs.

# 1. Interfaces Gráficas com o Usuário no Matlab

## 1.1. O que é uma GUI?

Uma interface gráfica com o usuário (*graphical user interface*, a famosa GUI) é uma interface “pictórica” para um programa. Uma GUI provê um ambiente familiar para o trabalho do usuário, fornecendo recursos como janelas, botões, menus, entre outros.

Pode-se criar uma interface gráfica se o aplicativo que você está desenvolvendo vai ser utilizado por outras pessoas, ou se a função que você está escrevendo vai ser usada várias vezes. Nesses casos, menus, botões e caixas de texto podem ser usados como métodos de inserção de dados.

Tenha em mente que a GUI deve se comportar de maneira inteligível, de forma que o usuário presuma qual a ação que o programa tomará quando este interagir com um objeto pertencente à interface. Por exemplo, quando um botão for clicado, é natural que a ação adotada pelo programa seja descrita pelo nome etiquetado no botão.

O Matlab contém exemplos interessantes de suas habilidades com as GUIs. Utilize, na janela de comandos (*command window*), o comando

```
>> demo
```

e veja o vídeo associado, ou consulte a ajuda do programa.

## 1.2. Os modos de criação

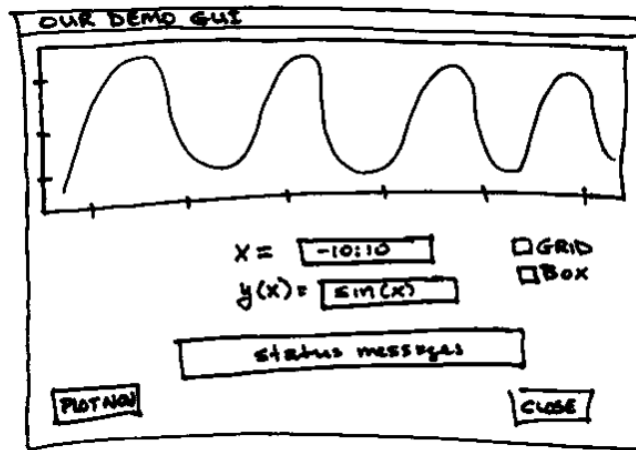
Basicamente, há dois meios de se criar uma GUI:

- Criando funções diretamente na linha de comando (o que pode ser dolorido, assombroso e desumano);
- Simplificando drasticamente a tarefa, com o uso da função Guide, existente no Matlab.

Neste curso as GUIs serão criadas com o auxílio do Guide, e depois programadas de acordo com a necessidade do usuário.

## 1.3. Projetando sua GUI

Antes de por a “mão na massa”, é interessante que haja um projeto das GUIs que você deseja construir. Obviamente, este projeto é feito depois das outras funções da interface terem sido planejadas. Um projeto de GUI pode ser feito com papel e caneta, como um rascunho da interface que você deseja que o programa tenha. A ideia de um esboço é exemplificada na Figura 1 [3].



**Figura 1.** Um esboço de interface gráfica com o usuário.

O rascunho em um papel é simples, mas independente da complexidade da GUI, é sempre um bom meio de começar.

Depois de projetada, é hora de estruturar sua GUI no Guide.

## 2. Guide – O caminho mais fácil até as GUIs

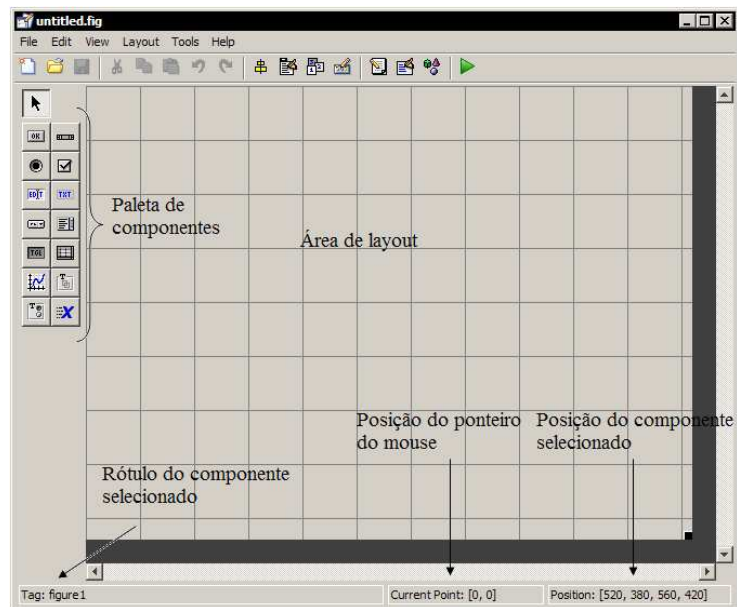
### 2.1. O que é o Guide?

O Guide (*graphical user interface development environment*, ou ambiente de criação de interfaces gráficas com o usuário) fornece um conjunto de ferramentas para a criação de GUIs. Estas ferramentas simplificam o processo de diagramação e programação das interfaces gráficas.

Para iniciar o Guide, digitamos na janela de comandos do Matlab:

```
>> guide
```

Na Figura 2, temos o Guide com uma GUI em branco.



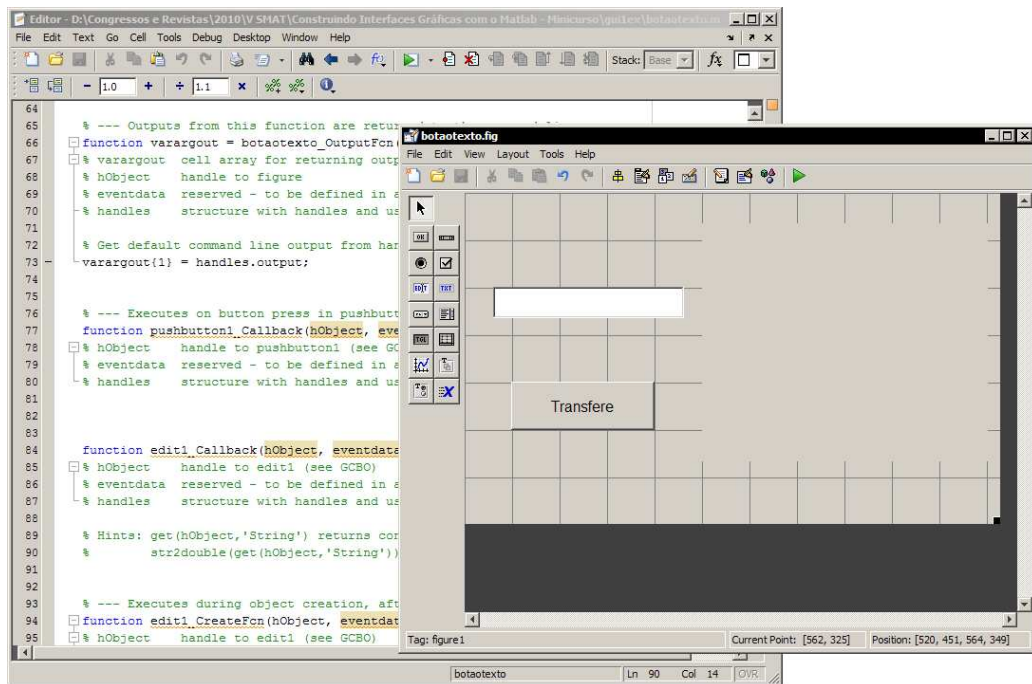
**Figura 2.** A tela inicial do Guide, exibindo uma GUI em branco.

Quando uma GUI é criada no Guide, ele se encarrega de criar dois arquivos: o arquivo do corpo da interface gráfica (que é salvo como uma figura de extensão `.fig`, do Matlab) e outro arquivo contendo funções que controlam como a GUI trabalha (salvo como um arquivo de função `.m`, do Matlab). Este arquivo fornece códigos para iniciar a GUI e contém a estrutura para os *callbacks* da interface (as rotinas que são executadas quando o usuário interage com um componente da GUI).

### 2.2. Programando a GUI

O Guide é um programa simples de ser utilizado. Podemos trabalhar facilmente editando a GUI, colocando objetos como botões e caixas de texto diretamente na área de layout. Porém, para que a GUI interaja com o usuário de maneira satisfatória, é preciso programar as ações de seus objetos. Fazemos isso alterando a função `.m` da interface gráfica que foi criada.

Como exemplo, na Figura 3, temos a GUI `botaotexto.fig`, e o arquivo `botaotexto.m` no editor do Matlab.



**Figura 3.** A GUI `botaotexto.fig`, e seu respectivo arquivo `.m` no Matlab Editor.

Esta interface gráfica exibe um botão, uma caixa de edição e um rótulo de texto. O Matlab chama esses objetos de *pushbutton*, *edit text* e *static text*, respectivamente.





Observe, no arquivo `botaotexto.m`, os *callbacks* já citados anteriormente. Quando clicarmos o *pushbutton*, ativaremos seu *callback*, que determinará a ação do botão. Da mesma forma, quando digitarmos alguns caracteres na *edit text* e pressionarmos “Enter”, ativaremos seu *callback*.

Note que *static text* não tem sua função *callback*. Tenha em mente que nem sempre todos os *callbacks* são utilizados; veremos isso em detalhes adiante.

### 3. GUIs e seus componentes

#### 3.1. Os componentes básicos de uma GUI

Observe, na Figura 2, a paleta de componentes do Guide. A Tabela 1, abaixo, descreve os componentes mais usados.

Objeto	Nome ( <i>tag</i> no Matlab)	Descrição
<b>Controles gráficos</b>		
	Botão simples ( <i>push button</i> )	Por vezes chamado de botão de controle, é um componente gráfico que normalmente contém um rótulo de texto. Clicando-o, o Matlab executa a ação definida pelo <i>callback</i> do objeto.
	Controle deslizante ( <i>slider</i> )	<i>Sliders</i> costumam ser utilizados para selecionar um valor em um intervalo de valores. Cada mudança de valor executa a ação definida no <i>callback</i> do objeto.
	Botão de rádio ( <i>radio button</i> )	O botão de rádio é um tipo de botão de chave, que contém um rótulo e um círculo ao lado deste rótulo. Quando selecionado, o círculo é preenchido. Botões de rádio costumam ser utilizados para selecionar uma entre um grupo mutuamente exclusivo de opções. Cada clique em um botão de rádio executa o <i>callback</i> definido para o objeto.
	Caixa de controle ( <i>check box</i> )	Estas caixas consistem em botões com um rótulo e uma caixa quadrada ao lado deste rótulo. Quando ativado, o controle alterna entre preenchido e limpo. As caixas de controle costumam ser usadas para indicar o estado de uma opção, ou selecionar várias opções não-exclusivas. Cada clique em um botão de rádio executa o <i>callback</i> definido.



---

## Controles gráficos

---



Caixa de edição (*edit text*)

Exibe texto em uma caixa, de forma que este texto possa ser editado ou substituído. Quando o usuário pressiona o botão “Enter”, o *callback* definido é executado.



Menu popup (*pop-up menu*)

São úteis para apresentar uma lista de escolhas mutuamente exclusivas ao usuário. Estes menus podem ser colocados em qualquer lugar da janela principal.



Caixa de listagem (*listbox*)

Parecem caixas de texto de múltiplas linhas que permitem que os usuários selecionem itens individuais ou múltiplos de uma lista com um clique do mouse. O *callback* definido é executado quando um valor da caixa é selecionado.



Botão de chave (*toggle button*)

Os botões de chave são idênticos aos botões simples, exceto pelo fato de alternarem entre dois estados (pressionado e não pressionado). Como no *pushbutton*, cada clique do mouse executa o *callback* definido.

---

## Elementos estáticos

---



Eixos (*axes*)

Os eixos criam um conjunto para exibição de dados. Para os eixos NUNCA são definidos *callbacks*.



Rótulo de texto (*static text*)

O rótulo de texto cria uma área para exibição de um conjunto de caracteres. Nestes rótulos podem ser exibidos dados coletados, por exemplo, de uma *edit text*, ou resultados de operações. Contudo, como para os *axes*, não são definidos *callbacks* para os rótulos de texto.

---

## Elementos estáticos

---



Painel (*panel*)

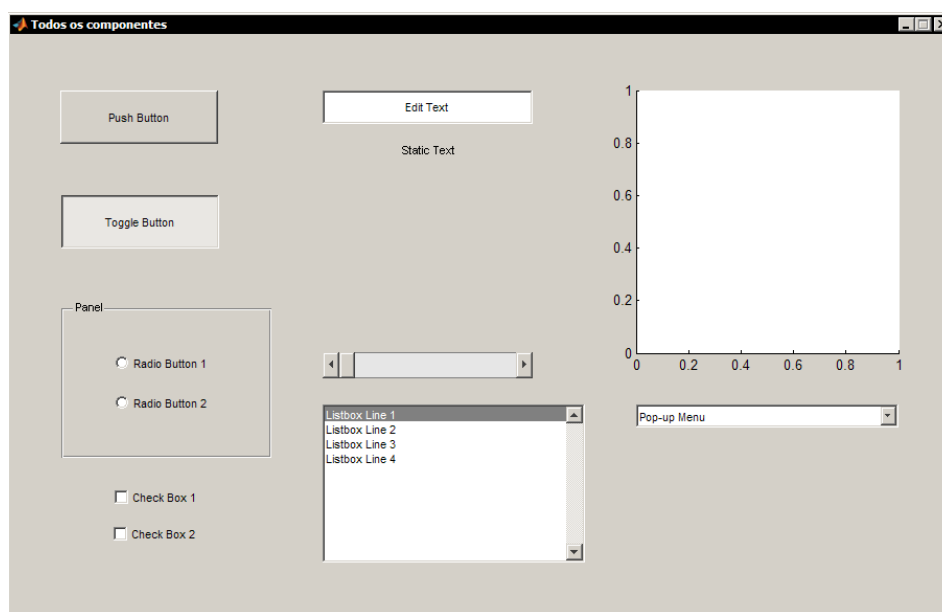
Painéis, chamados de *frames* em algumas versões do Matlab, são áreas destinadas a agrupar conjuntos de componentes. Também não geram *callbacks*.

---

**Tabela 1.** Descrição dos componentes mais usados na criação de uma GUI.

---

Na Figura 4, como exemplo, temos uma GUI com todos esses componentes.



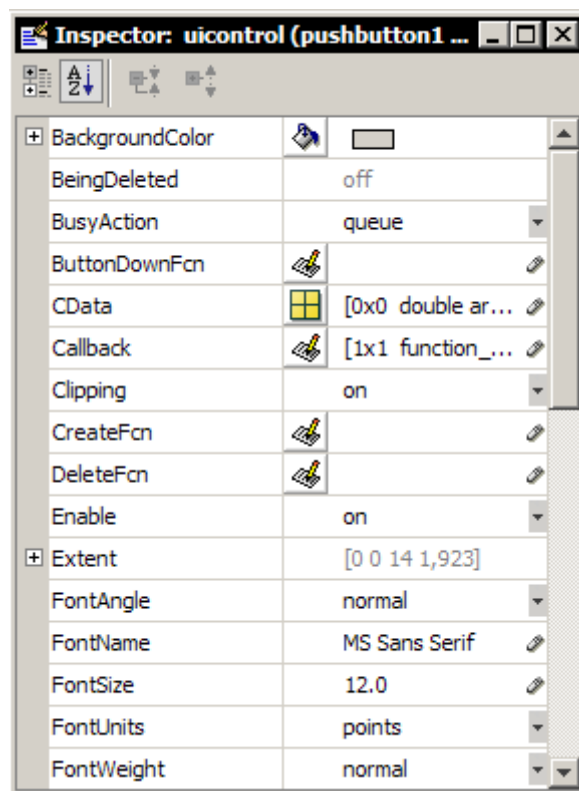
**Figura 4.** Interface gráfica com todos os componentes descritos na Tabela 1.

Conhecendo estes componentes e utilizando-os convenientemente, estaremos a um passo da criação de GUIs úteis e funcionais. Depois de projetar a GUI, basta programar as ações de cada componente.

Entretanto, algumas vezes é interessante mudar a cor de fundo da janela, ou o tipo de fonte de um rótulo de texto, entre outros aspectos da interface. Podemos fazer isso no Guide utilizando o inspetor de propriedades dos componentes.

### 3.2. O inspetor de propriedades

Se executarmos um clique duplo em qualquer um dos componentes no momento de criação da sua interface gráfica no Guide, o inspetor de propriedades (*Inspector*; *Property Inspector* nas primeiras versões) é acionado. Pode-se também clicar no componente em questão com o botão direito do mouse, e selecionar a opção *Property Inspector*, ou escolher *View* → *Property Inspector* no menu do Guide. Na Figura 5, temos o inspetor de propriedades da interface gráfica `botaotexto`, exibida na Figura 3. As propriedades apresentadas são referentes ao *pushbutton* daquela GUI.



**Figura 5.** O inspetor de propriedades do Guide. As propriedades exibidas referem-se a um *pushbutton*.

O inspetor de propriedades exibe as propriedades do objeto, e se alterará conforme outros objetos são clicados. Podemos alterar os valores das propriedades como necessário.

Na Tabela 2, dada abaixo, temos algumas propriedades compartilhadas pela maioria dos componentes descritos na Tabela 1.

Propriedade do componente	Descrição
<i>BackgroundColor</i>	Especifica a cor do plano de fundo do objeto. Existe um valor predefinido, e outro valor pode ser escolhido diretamente em uma paleta de cores ou especificando-se valores numéricos (HSB ou RGB).
<i>Callback</i>	Especifica o nome e os parâmetros da função a ser chamada quando o objeto é ativado.
<i>Enable</i>	Especifica se o objeto pode ser selecionado ou não. Se o objeto em questão não está habilitado, ele não responderá ao clique do mouse ou a entrada de dados do teclado. Possíveis valores são ‘on’, ‘off’ e (em versões mais recentes) ‘inactive’.

Propriedade do componente	Descrição
<i>FontAngle</i>	Um conjunto de caracteres contendo o ângulo da fonte do texto exibido no objeto. Valores possíveis são <i>'normal'</i> , <i>'italic'</i> e <i>'oblique'</i> .
<i>FontName</i>	Um conjunto de caracteres contendo o nome da fonte do texto exibido no objeto.
<i>FontSize</i>	Um número que especifica o tamanho da fonte exibida no objeto. O tamanho da fonte é especificado em pontos, por padrão.
<i>FontWeight</i>	Um conjunto de caracteres contendo a espessura da fonte exibida no objeto. Valores possíveis são <i>'light'</i> , <i>'normal'</i> , <i>'demi'</i> , e <i>'bold'</i> .
<i>ForegroundColor</i>	Especifica a cor do plano da frente do objeto. Da mesma forma que para <i>BackgroundColor</i> , outro valor pode ser escolhido por uma paleta de cores ou com valores numéricos.
<i>HorizontalAlignment</i>	Especifica o alinhamento horizontal do texto contido no objeto. Possíveis valores são <i>'left'</i> , <i>'center'</i> e <i>'right'</i> .
<i>Max</i>	O valor máximo da propriedade <i>Value</i> do objeto.
<i>Min</i>	O valor mínimo da propriedade <i>Value</i> do objeto.
<i>Position</i>	Especifica a posição do objeto na tela, nas unidades especificadas pela propriedade <i>Units</i> . Esta propriedade constitui de um vetor de 4 elementos, no qual os dois primeiros elementos são as posições <i>x</i> e <i>y</i> da borda inferior esquerda do componente (relativos à janela que contém o objeto), e os últimos são a largura ( <i>'width'</i> ) e a altura ( <i>'height'</i> ) do componente.
<i>Tag</i>	O 'nome' do objeto. Em qualquer função pertencente à GUI, quando quisermos nos referir a um componente qualquer, usaremos o nome dado em <i>Tag</i> para este objeto.

Propriedade do componente	Descrição
<i>TooltipString</i>	Especifica o texto de ajuda que será exibido quando o usuário deixa o ponteiro do mouse por determinado tempo sobre o objeto.
<i>Units</i>	As unidades usadas para descrever a posição da janela e de seus componentes. Possíveis valores são <i>'inches'</i> , <i>'centimeters'</i> , <i>'normalized'</i> , <i>'points'</i> , <i>'pixels'</i> ou <i>'characters'</i> . O valor padrão em versões mais recentes é <i>'characters'</i> ; em versões anteriores este valor é <i>'pixels'</i> .
<i>Value</i>	O valor atual do componente. Para botões de chave, caixas de controle e botões de rádio, o valor é <i>'max'</i> quando o botão está ativado e <i>'min'</i> se o botão está desativado. Outros controles possuem significados diferentes para este termo.
<i>Visible</i>	Especifica se o objeto está ou não visível. Os possíveis valores são <i>'on'</i> e <i>'off'</i> .

**Tabela 2.** Propriedades comuns à maioria dos componentes dados pelo Guide.

Conhecendo estas propriedades, podemos começar a customizar a aparência da interface gráfica.

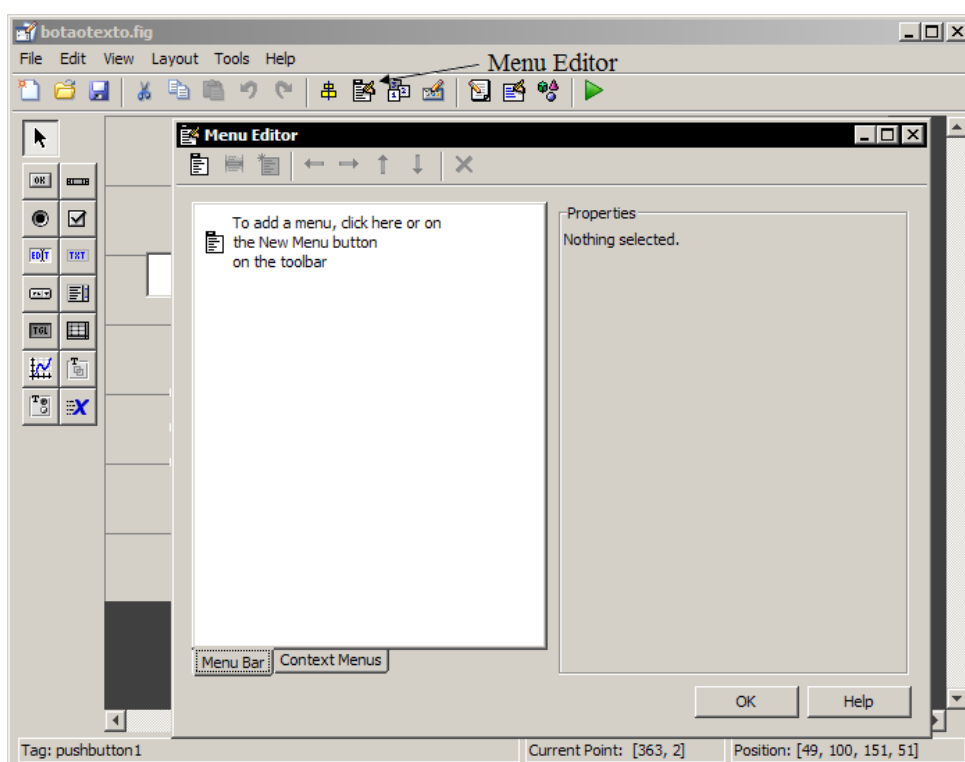
## 4. Criando e modificando menus

### 4.1. A criação de menus e submenus no Guide

Menus podem também ser adicionados às interfaces gráficas criadas no Guide. Eles são úteis quando queremos disponibilizar opções que não são necessárias em todo o momento de execução, ou adicionar funções que são utilizadas com pouca frequência.

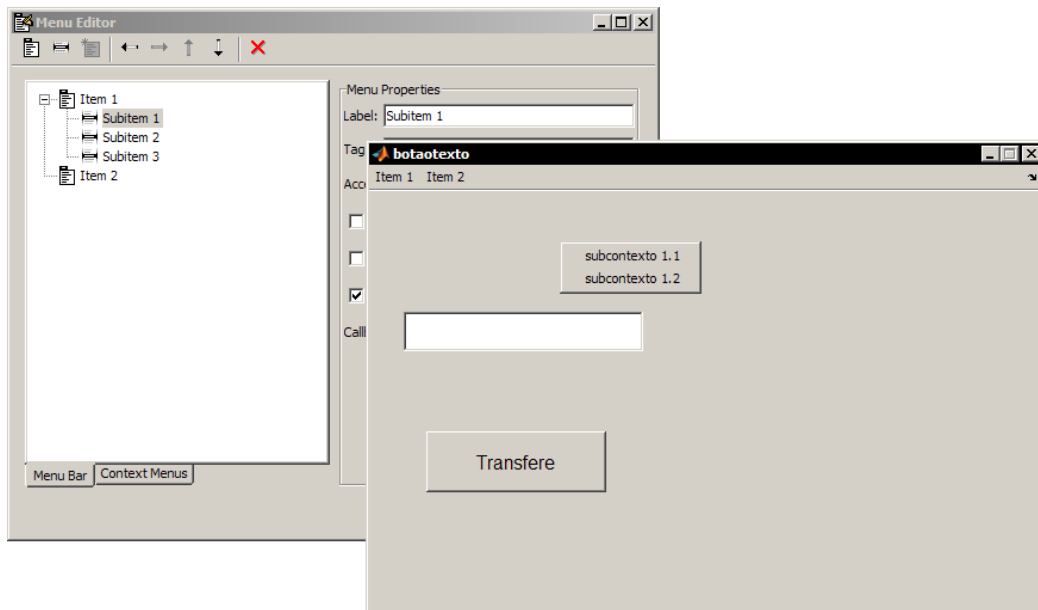
Há dois tipos de menus disponíveis no Guide: os menus padrão (*standard*), que se situam na barra de menus, no topo de uma janela, e os menus de contexto, que aparecem quando o usuário clica com o botão direito sobre um determinado objeto.

O Guide possui uma ferramenta para edição de menus, o *Menu Editor*. Para acioná-lo, podemos ir para *Tools* → *Menu Editor*, ou simplesmente clicar no ícone que exibe um menu. Este ícone é mostrado na Figura 6, juntamente com o *Menu Editor*.



**Figura 6.** O manipulador de menus do Guide, *Menu Editor*.

A Figura 7 mostra exemplos de menus padrão no *Menu Editor*, e a interface gráfica com um menu de contexto ativado. É importante saber que, para os menus de contexto trabalharem da maneira esperada, devemos alterar a propriedade *UIContextMenu* do componente ao qual queremos associar tal menu.

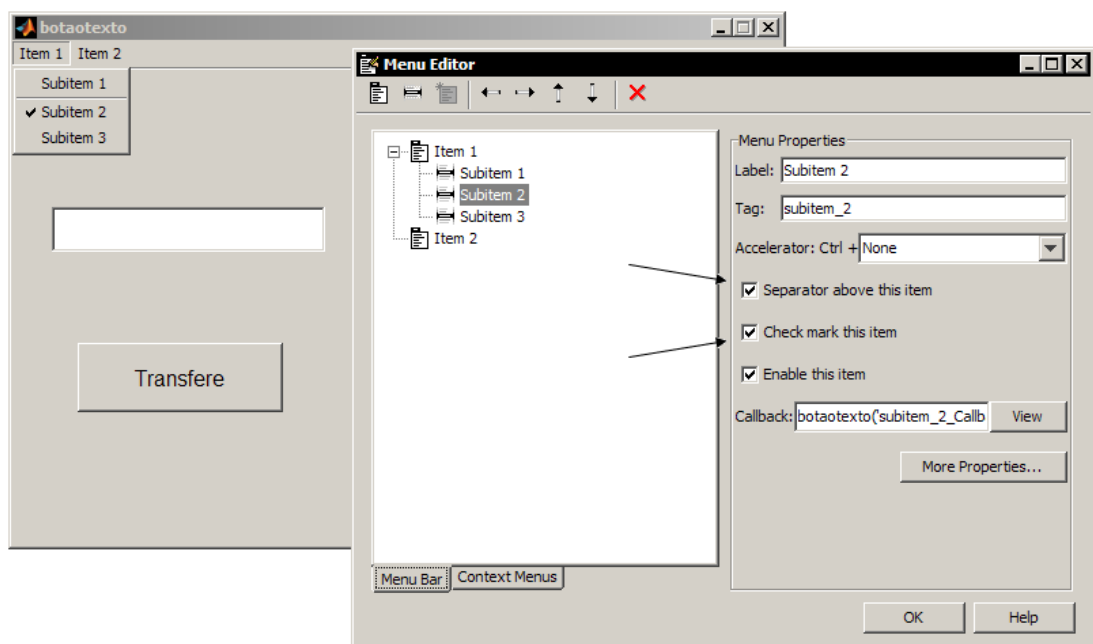


**Figura 7.** Menus padrão e de contexto criados no *Menu Editor*.

Depois de construídos, podemos trabalhar com algumas propriedades dos menus, a fim de deixar a interface ainda mais intuitiva.

## 4.2. Propriedades dos menus

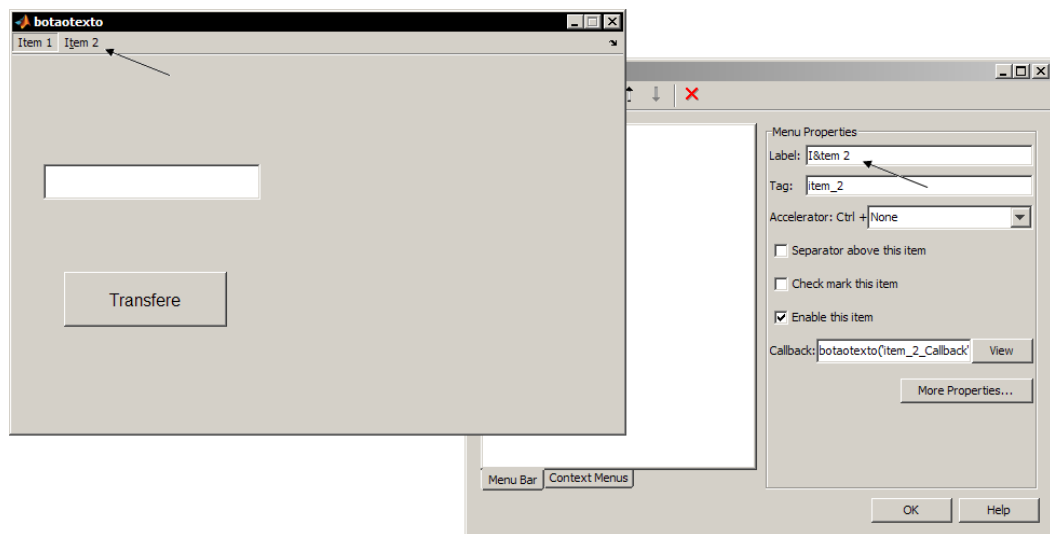
Alguns recursos de interesse na personalização de menus são as caixas “*separator above this item*”, que quando selecionada mostra uma barra que separa os subitens do menu acima do submenu relacionado, e “*check mark this item*”, que quando marcada mostra o subitem correspondente selecionado com uma marcação. Na Figura 8, exemplos das duas opções na GUI *botatexto* e onde encontrá-las no *Menu Editor*.



**Figura 8.** As opções “*separator above this item*” e “*check mark this item*” selecionadas para o subitem 2, e a interface gráfica resultante.

Quando lidamos com menus, é comum observarmos a presença de atalhos que aparecem na forma de letras sublinhadas. As “letras mnemônicas”, como são chamadas, servem para agilizar o acesso ao menu. Para utilizá-las, pressionamos ALT e a letra correspondente ao item de menu que desejamos acionar.

Para implementar a funcionalidade das letras mnemônicas no Menu Editor do Guide, simplesmente colocamos o caractere “&” imediatamente antes da letra desejada. Por exemplo, na interface exibida na Figura 8, se quisermos que o menu “Item 2” tenha como atalho a combinação ALT + T, escrevemos “I&tem 2” na propriedade “Label” do *Menu Editor*. Note que, em tempo de execução, o atalho apenas aparecerá quando pressionarmos ALT. Na Figura 9, temos o exemplo descrito acima na GUI `botaotexto`.



**Figura 9.** Exemplo de interface gráfica com uma letra mnemônica e a correspondente entrada no *Menu Editor*.

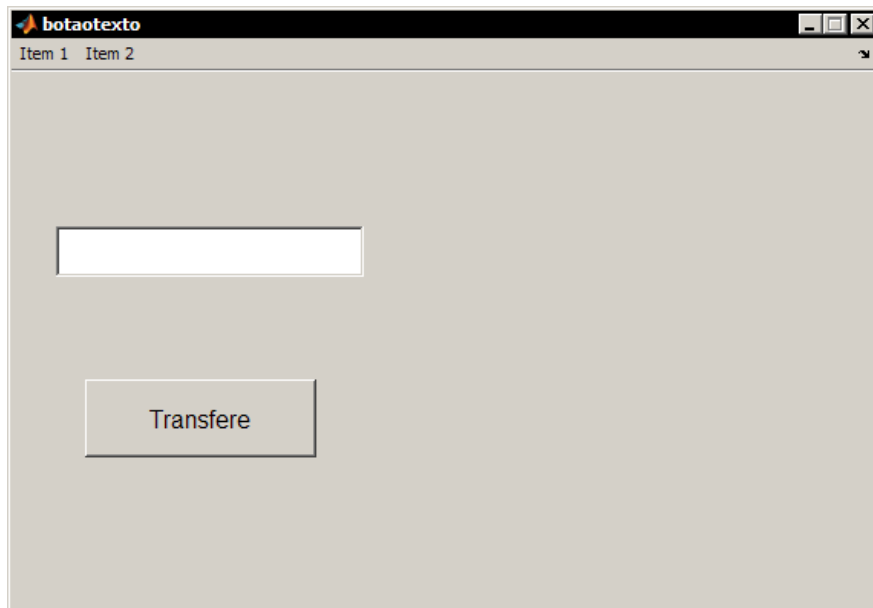


## 5. Exemplos, exemplos, exemplos

Com todas as ferramentas adquiridas nas seções anteriores, chegamos na parte mais interessante: programar!

### **Exemplo 1:** *Trabalhando com a GUI* botaotexto

Vamos relembrar a interface gráfica de nome botaotexto, vista em execução na Figura 10.



**Figura 10.** A GUI botaotexto, a ser utilizada no Exemplo 1.

Neste exemplo, quando o botão ‘Transfere’ for pressionado, o texto digitado na caixa de edição *edit1* será copiado para o rótulo de texto *text1*.

Para isso, o *callback* do botão *pushbutton1* deverá conter os seguintes comandos:

```
% --- Executes on button press in pushbutton1.  
function pushbutton1_Callback(hObject, eventdata, handles)  
% hObject      handle to pushbutton1 (see GCBO)  
% eventdata    reserved - to be defined in a future version  
of MATLAB  
% handles      structure with handles and user data (see  
GUIDATA)  
  
texto = get(handles.edit1, 'String');  
set(handles.text1, 'String', texto);
```

### **Exemplo 2: Todos os botões de rádio estão selecionados?**

Os botões de rádio geralmente são exclusivos; ou seja, quando um é clicado, os outros relacionados àquele são desativados. Isso não é implementado automaticamente, o que requer esforço extra de nossa parte. Neste exemplo, veremos qual a programação necessária a fim de que este problema não aconteça na nossa interface.

Na Figura 11, a interface gráfica ilustra o problema que gostaríamos de evitar. A propriedade “*String*” de *radiobutton1*, *radiobutton2* e *radiobutton3* foi alterada para “Sim”, “Não” e “Talvez”, respectivamente.



**Figura 11.** A GUI *botaoradio*, a ser utilizada no Exemplo 2.

Editaremos os *callbacks* de *radiobutton1* desta forma:

```
% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject      handle to radiobutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of
radiobutton1

set(handles.radiobutton1,'Value',1);
set(handles.radiobutton2,'Value',0);
set(handles.radiobutton3,'Value',0);
```

Por sua vez, os *callbacks* de *radiobutton2* serão:

```
% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject      handle to radiobutton2 (see GCBO)
% eventdata    reserved - to be defined in a future version
of MATLAB
```

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

```
% Hint: get(hObject,'Value') returns toggle state of
radiobutton2
```

```
set(handles.radiobutton1,'Value',0);
set(handles.radiobutton2,'Value',1);
set(handles.radiobutton3,'Value',0);
```

E finalmente, os *callbacks* de *radiobutton3*:

```
% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject      handle to radiobutton3 (see GCBO)
% eventdata    reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)
```

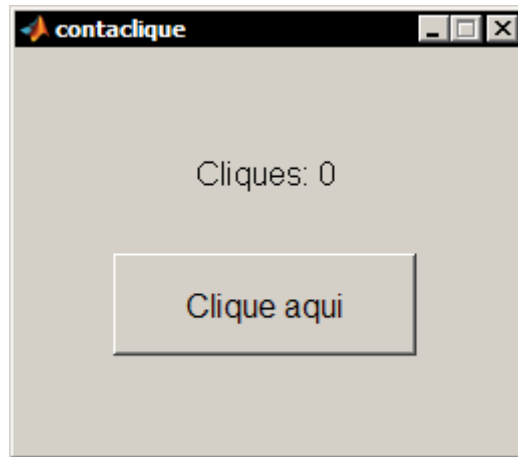
```
% Hint: get(hObject,'Value') returns toggle state of
radiobutton3
```

```
set(handles.radiobutton1,'Value',0);
set(handles.radiobutton2,'Value',0);
set(handles.radiobutton3,'Value',1);
```

Com isso, os botões de rádio serão selecionados um por vez.

### **Exemplo 3:** *Um simples contador de cliques*

Continuando a interação de botões com outros objetos, agora teremos um contador de cliques. A interface gráfica deste contador é dada abaixo, na Figura 12. A propriedade “*String*” de *text1* e de *pushbutton1* foi alterada para “Cliques: 0” e “Clique aqui”, respectivamente, e a propriedade “*FontSize*” dos dois objetos foi alterada para “12”.



**Figura 12.** A GUI *contaclique*, a ser utilizada no Exemplo 3.

Os *callbacks* de *pushbutton1* serão:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)

% Declara e inicializa a variável para armazenar a contagem
persistent contador
if isempty(contador)
    contador = 0;
end

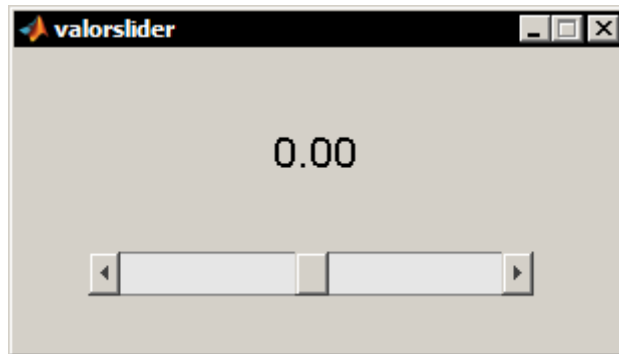
% Atualiza a contagem
contador = contador + 1;

% Cria o novo texto
str = sprintf('Cliques: %d',contador);

% Atualiza o rótulo de texto
set(handles.text1, 'String', str);
```

#### **Exemplo 4:** *Um slider ditando valores*

Neste exemplo, veremos como fazer para que um *slider* mostre valores entre um intervalo de números em um rótulo de texto. A interface gráfica a ser utilizada é dada na Figura 13. A propriedade “*String*” de *text1* foi modificada para “0.00”, e a propriedade “*FontSize*”, para “16”. As propriedades “*Min*” e “*Max*” de *slider1* foram alteradas, respectivamente, para “-10” e “10”.



**Figura 12.** A GUI *valorslider*, a ser utilizada no Exemplo 4.

Para que a interface retorne os valores, os *callbacks* de *slider1* serão:

```
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject      handle to slider1 (see GCBO)
% eventdata    reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%         get(hObject,'Min') and get(hObject,'Max') to
determine range of slider

% Encontra o valor do slider
valor = get(handles.slider1,'Value');

% Copia o valor no rótulo de texto
str = sprintf('%.2f',valor);
set(handles.text1,'String',str);
```

**Exemplo 5:** *Reinventando a roda com as caixas de diálogo*

O último exemplo mostra as caixas de diálogo predefinidas pelo Matlab, que são práticas e facilmente utilizáveis. A Tabela 3 exibe um resumo das caixas predefinidas mais comuns.

Função	Descrição
<i>dialog</i>	Cria uma figura para caixa de diálogo ou GUI.
<i>errordlg</i>	Indica erro na caixa de diálogo.
<i>helpdlg</i>	Fornece ajuda pela caixa de diálogo.
<i>inputdlg</i>	Permite entrada de dados pela caixa de diálogo.
<i>listdlg</i>	Seleciona itens de uma lista pela caixa de diálogo.
<i>uigetfile</i>	Abre um arquivo em uma caixa de diálogo.
<i>uiputfile</i>	Armazena um arquivo em uma caixa de diálogo.
<i>uisetcolor</i>	Aplica a cor selecionada ao objeto gráfico da caixa de diálogo.
<i>uisetfont</i>	Aplica a fonte selecionada ao objeto gráfico da caixa de diálogo.
<i>warndlg</i>	Caixa de advertência.

**Tabela 2.** Propriedades comuns à maioria dos componentes dados pelo Guide.

Alguns exemplos da utilização dos comandos para incorporar essas caixas de diálogo à sua GUI:

*dialog:*

```
>> dialog
```

*errordlg:*

```
>> errordlg('String de erro', 'Nome da janela')
```

*helpdlg*

```
>> helpdlg('String de ajuda', 'Nome da janela')
```

*inputdlg:*

```
>> prompt = {'O tamanho da matriz para x^2:', 'Nome do  
mapa de cores:'};  
>> nome = 'Entrada para função de picos';  
>> numlinhas = 1;  
>> resp = {'20', 'hsv'};  
>> resposta = inputdlg(prompt, nome, numlinhas, resp);
```

*listdlg*

```
>> d = dir;  
>> str = {d.name};  
>> listdlg('PromptString', 'Selecione um  
arquivo:', 'SelectionMode', 'single', 'ListString', str)
```

*uigetfile*

```
>> uigetfile({'*.m;*.fig;*.mat;*.mdl', 'Todos os  
arquivos MATLAB (*.m, *.fig, *.mat, *.mdl)'; '*.m',  
'Arquivos M (*.m)'; '*.fig', 'Figuras (*.fig)';  
'*.mat', 'Arquivos MAT (*.mat)'; '*.mdl', 'Modelos (*.mdl)';  
'*.*', 'Todos os arquivos (*.*)'}, 'Escolha um arquivo');
```

*uiputfile*

```
>> uiputfile({'*.m;*.fig;*.mat;*.mdl', ' Todos os  
arquivos MATLAB (*.m, *.fig, *.mat, *.mdl)'; '*.*', 'Todos  
os arquivos (*.*)'}, 'Salvar como');
```

*uisetcolor*

```
>> hText = text(.5, .5, 'Sou muito feliz com o  
Matlab!');  
>> uisetcolor(hText, 'Cor do texto')
```

*uisetfont*

```
>> textol = uicontrol('style', 'text', 'string',  
'XxYyZz');  
>> uisetfont(textol, 'Alterar Fonte');
```

*warndlg*

```
>> warndlg('String de aviso', 'Nome da janela')
```

## 6. Para onde ir agora?

Depois de aprender o básico sobre interfaces gráficas no Matlab, o que fazer? Sentar e chorar? Não, não. Ficar tranquilo e se acomodar com o que sabe? Menos ainda! Mas quando você começar a procurar e sentir a “escassez” de material pode surgir um desânimo. Onde começamos a procurar? Aqui, algumas dicas.

- A referência mais natural pra procurarmos ajuda é o site da *MathWorks, Inc*, a empresa criadora do Matlab. O site é [www.mathworks.com](http://www.mathworks.com), e lá, procure por *Academia*. Destaque para o manual (que pode ser encontrado em PDF no site) *Creating Graphical User Interfaces*, de autoria da própria *MathWorks*.

- O verdadeiro “pai dos burros” da internet, talvez o site mais fantástico da rede, [www.google.com.br](http://www.google.com.br). Procure por “*GUI Matlab*”, e gaste algum tempo consultando os melhores resultados, como livros, apresentações e apostilas.

- O site *Sharp Programmer* possui, além das seções voltadas para outras linguagens de programação, também uma seção voltada para as interfaces gráficas no Matlab. Esta seção está em [www.sharpprogrammer.com/matlab/matlab-gui-tutorials/](http://www.sharpprogrammer.com/matlab/matlab-gui-tutorials/).

- Fora da internet existem também ótimos materiais, que tratam da questão das GUIs de maneira mais aprofundada. As referências bibliográficas deste material são exemplos de bons livros.



## **Referências bibliográficas**

- [1] Ashi, R. Y. A., Ameri, A. A. “Introduction to Graphical User Interface (GUI)”, UAE University.
- [2] Hanselman, D., Littlefield, B. R. “Mastering Matlab 6”, Prentice Hall, 2000.
- [3] Marchand, P., Holland, O. T. “Graphics and GUIs with Matlab”, Chapman & Hall, 2003.
- [4] Vaz Junior, C. A. “Desenvolvimento de Interface Gráfica em Ambiente Matlab”, 2005.