O Ambiente GUIDE do MATLAB

(Graphical User Interface Development Environment)

Rafael Lopez Rangel

e

Luiz Fernando Martha

CIV2801 – Fundamentos da Computação Gráfica Aplicada

2019.2

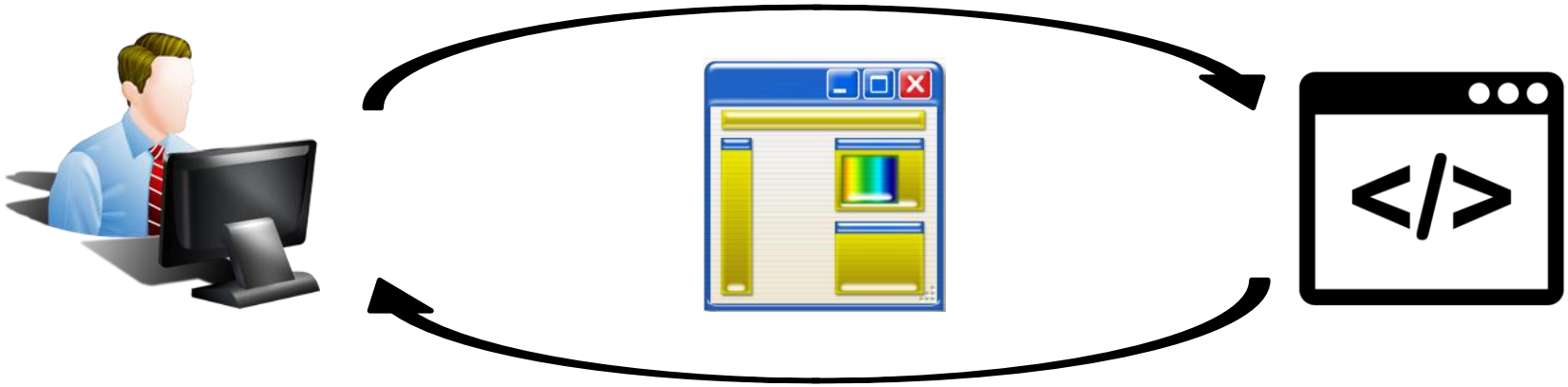
Interface Gráfica do Usuário (GUI)

- Uma interface gráfica permite a interação do usuário do computador com um programa por meio de elementos gráficos (botões, menus, etc.)

- O ambiente GUIDE permite a criação manual de uma interface gráfica do utilizador (GUI – Graphical User Interface) de forma rápida e fácil

- A comunicação entre as ações realizadas pelo usuário sobre os elementos gráficos e a resposta do programa se dá por meio do paradigma da Programação Orientada à Eventos

- Interfaces gráficas também podem ser criadas no MATLAB apenas com código de programação (criação programática)



Interface Gráfica do Usuário (GUI)

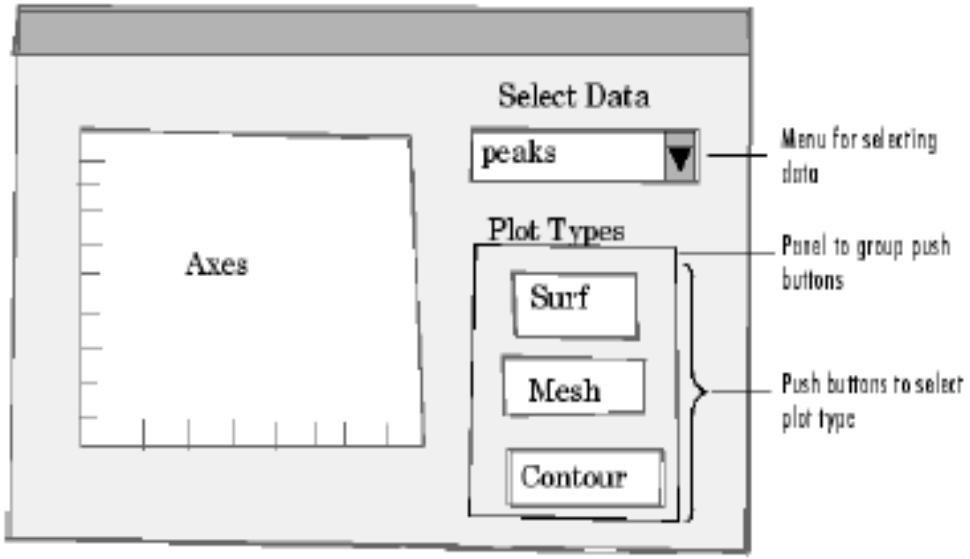
Antes de começar a montar uma interface gráfica é necessário projetá-la.
Para isso, deve-se pensar em como se dará a interação do usuário com o programa:

- Quais funcionalidades estarão disponíveis e como serão executadas
- Quais os componentes serão utilizados para permitir o usuário realizar essas tarefas
- Prever possíveis erros de uso do programa para cercar as ações do usuário
- Organização do layout

Para projetar interfaces com um grau de complexidade relativamente alto, recomenda-se esboçar o layout em uma folha de papel.

http://www.usabilitynet.org/management/b_overview.htm

http://asktog.com/atc/principles-of-interaction-design/



Interface Gráfica do Usuário (GUI)

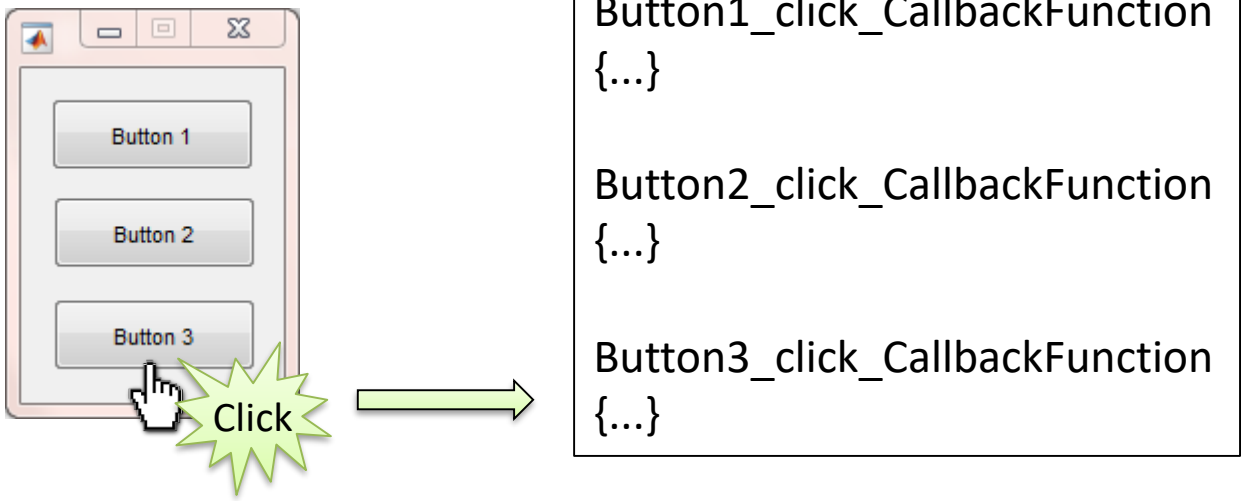
Após projetar a interface, deve-se decidir qual técnica será utilizada para criá-la.
No MATLAB é possível criar uma interface de duas formas: Utilizando o ambiente interativo GUIDE, ou fazendo programaticamente, apenas através de código de programação.

A utilização do GUIDE torna o trabalho muito mais simples e intuitivo, porém a técnica utilizada depende da experiência do programador, suas preferências e o tipo de GUI que se deseja. De acordo com a MathWorks, essa decisão pode ser baseada no grau de complexidade da GUI:

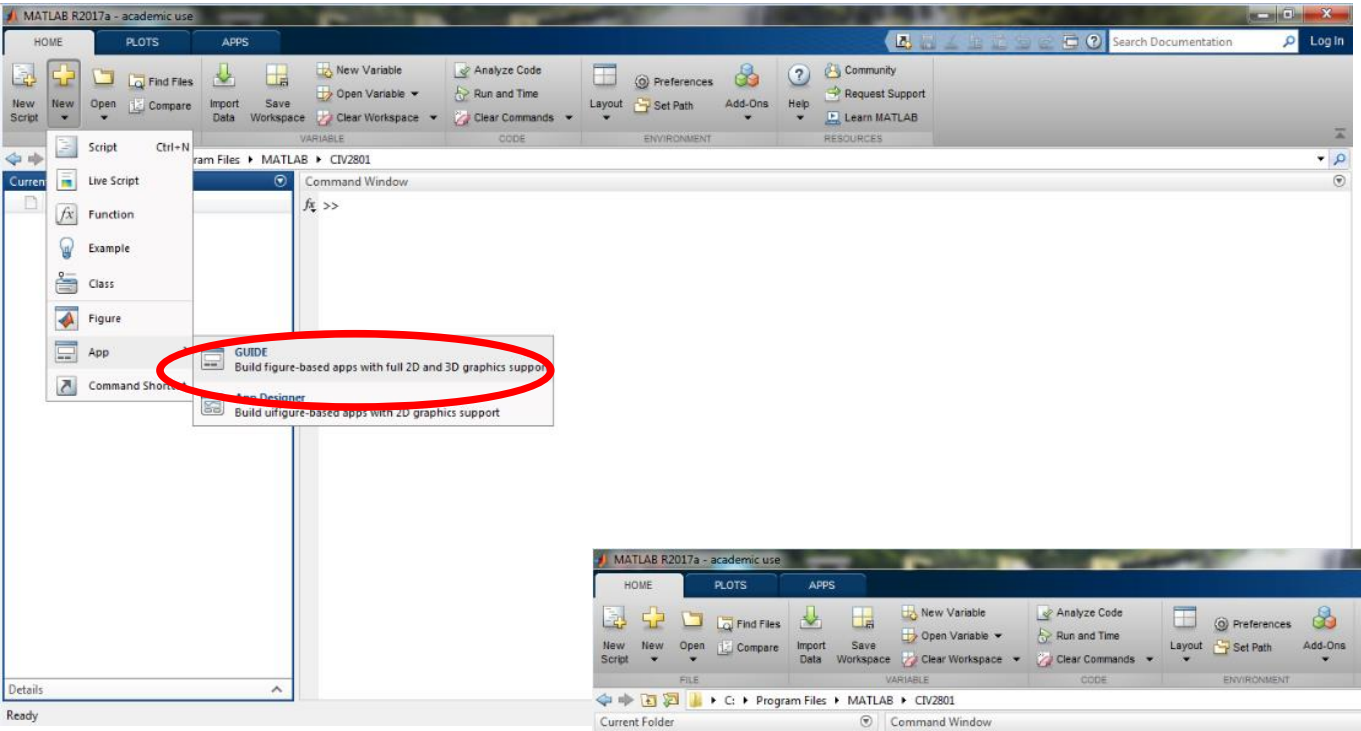
GUI	Technique
Dialog box	MATLAB provides a selection of standard dialog boxes that you can create with a single function call. For links to these functions, see "Predefined Dialog Boxes" in the MATLAB Function Reference documentation.
GUI containing just a few components	It is often simpler to create GUIs that contain only a few components programmatically. Each component can be fully defined with a single function call.
Moderately complex GUIs	GUIDE simplifies the creation of such GUIs.
Complex GUIs with many components, and GUIs that require interaction with other GUIs	Creating such GUIs programmatically lets you control exact placement of the components and provides reproducibility.

Programação Orientada à Eventos

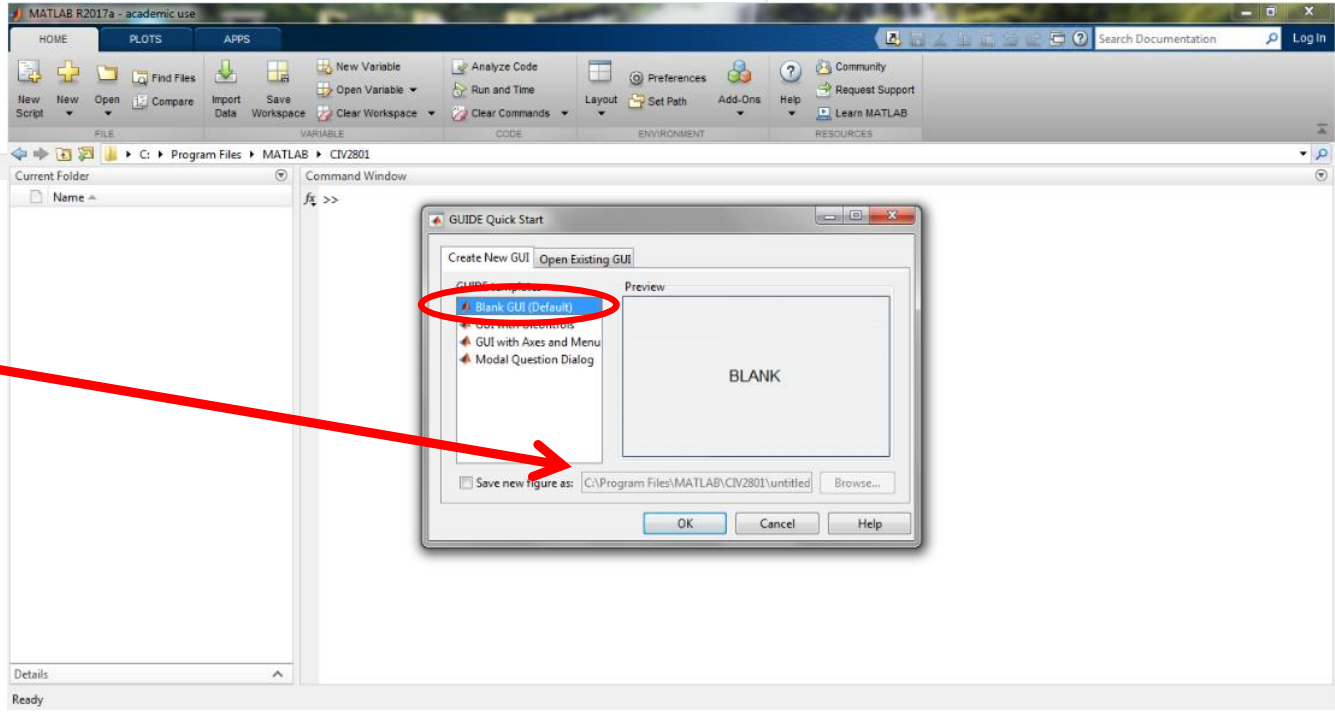
- O fluxo do código é guiado por indicações externas chamadas eventos.
- Eventos são as diferentes ações que usuários podem realizar sobre os componentes adicionados à interface.
- Cada evento está associado à uma função chamada *Callback*, disparada quando se verifica a ocorrência de tal evento, que define a reação do programa.



Inicializando uma Janela de Interface Gráfica Vazia



Também é possível acessar o GUIDE digitando guide na janela de comando e pressionando Enter

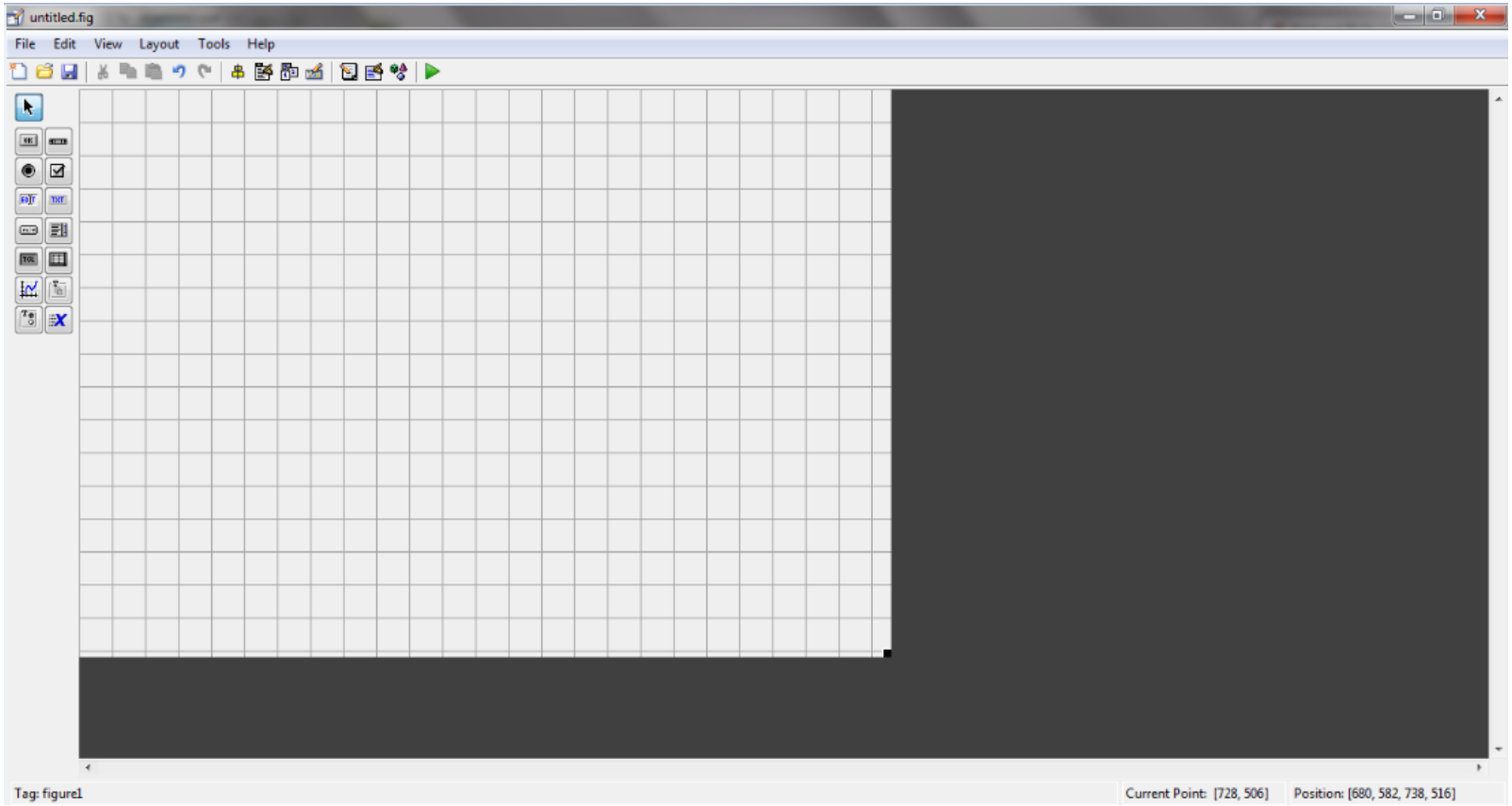


Sempre atentar ao diretório no qual se está trabalhando

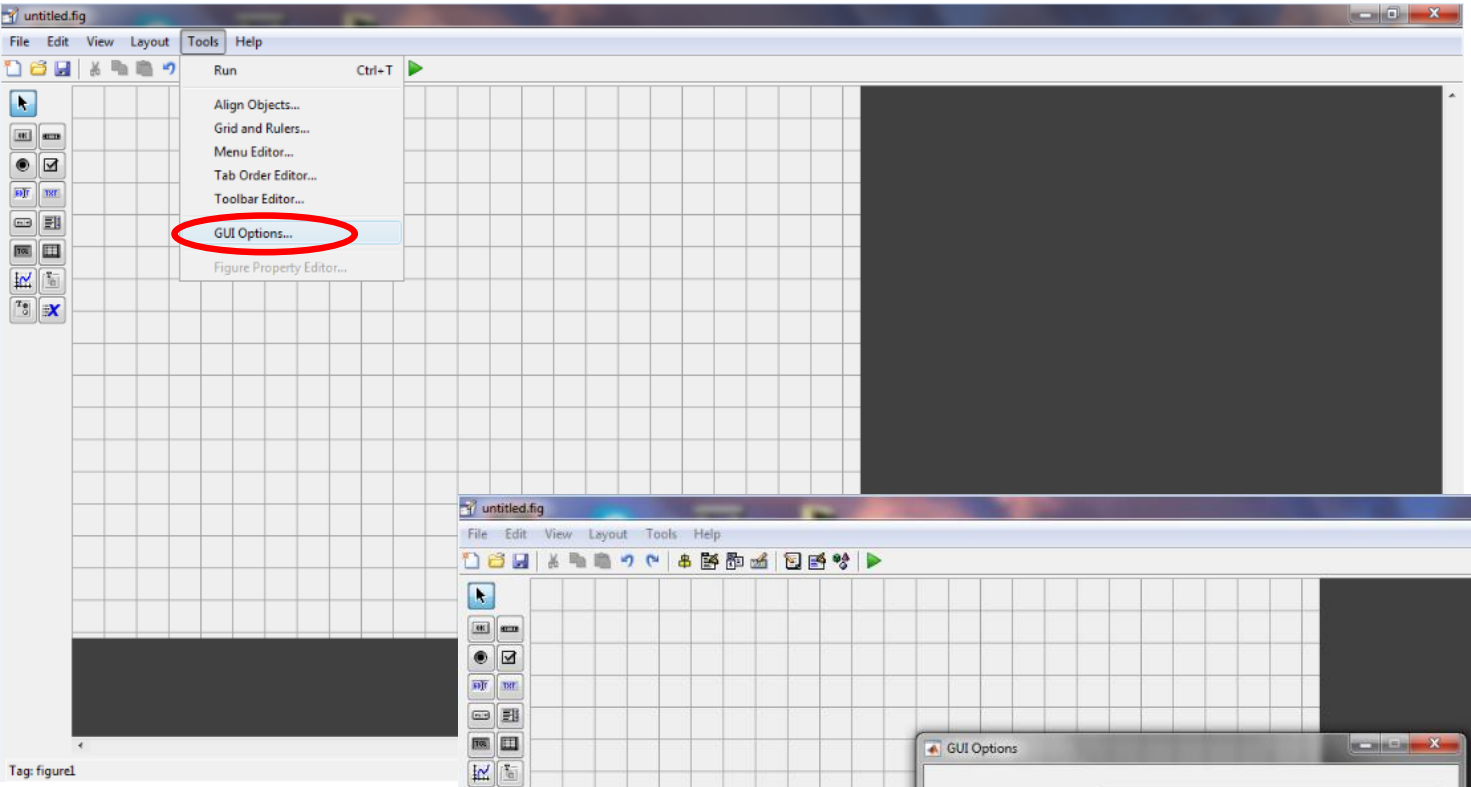
GUIDE: Ambiente de Criação de GUIs

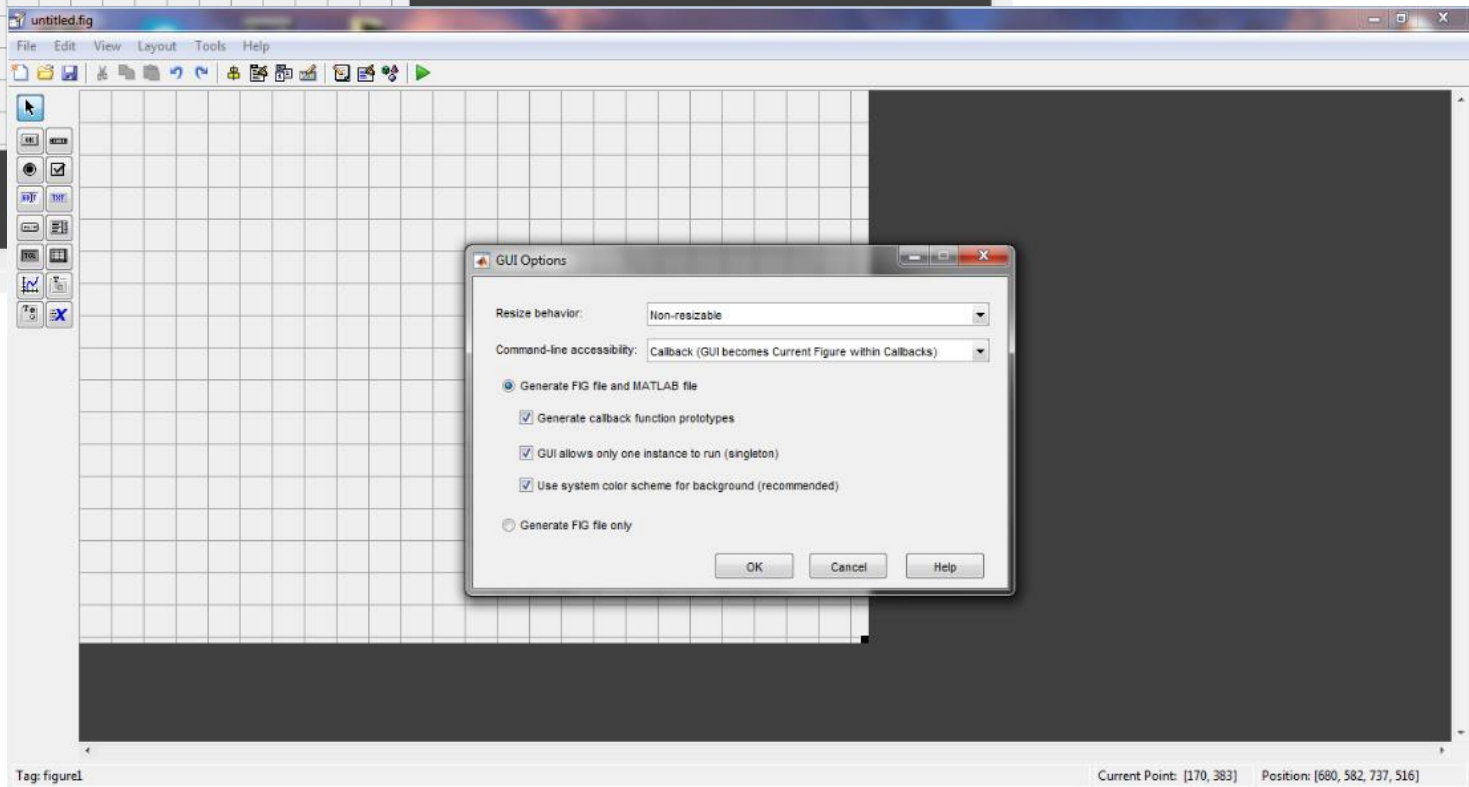
É possível elaborar o layout de uma GUI, atribuindo e modificando as propriedades dos componentes gráficos.

Um arquivo de código MATLAB pode ser automaticamente gerado com as funções *callback* para programar o comportamento destes componentes, de acordo com as ações do usuário.



Alterando as Opções da GUI





Alterando as Opções da GUI

- **Resize behavior**

Controla as opções de redimensionamento da janela.

- Non-resizable: A janela terá um tamanho fixo e usuários não podem redimensiona-la.

- Proportional: Permite usuários à alterarem o tamanho da janela, fazendo com que seus componentes sejam redimensionados na mesma escala.

- Use SizeChangedFcn: Permite escrever uma função que controla o comportamento da janela e seus componentes quando usuários alteram suas dimensões.

Alterando as Opções da GUI

- **Command-line accessibility**

Controla o acesso ao controle do comportamento e propriedades da janela e seus componentes.

- Callback: Modificações na GUI podem ser feitas apenas através de funções *callback*, não sendo permitido seu controle através da janela de comando ou scripts.

- Off: O comportamento e propriedades da GUI não podem ser modificados externamente ao ambiente GUIDE.

- On: Permite que o controle da GUI seja acessada através de funções *callback*, da janela de comando e de scripts.

Alterando as Opções da GUI

- **Generate FIG file and MATLAB file**

Quando selecionada, esta opção gera dois arquivos quando a GUI é salva, um arquivo *.fig* e um arquivo *.m*, que devem ter o mesmo nome e estar na mesma pasta.

O arquivo *.fig* guarda as informações sobre o layout e propriedades da janela e seus componentes, em um arquivo binário.

O arquivo MATLAB *.m* é onde o programador pode codificar o comportamento da GUI através de funções *callback* que seguem o paradigma da Programação Orientada à Eventos.

Se a opção "Generate FIG file only" for selecionada, apenas o arquivo *.fig* é gerado, sendo responsabilidade do programador criar por conta própria um arquivo *.m* para especificar o comportamento da janela e seus componentes.

myGUI.fig

myGUI.m

Alterando as Opções da GUI

- **Generate callback function prototype**

Esta opção faz com que templates das funções *callback* mais comuns de cada componente sejam adicionadas automaticamente ao arquivo *.m* da GUI.
São adicionadas apenas a declaração de cada função *callback* no arquivo, cabendo ao programador inserir o código que ditará a resposta do programa ao evento que dispara a respectiva função.

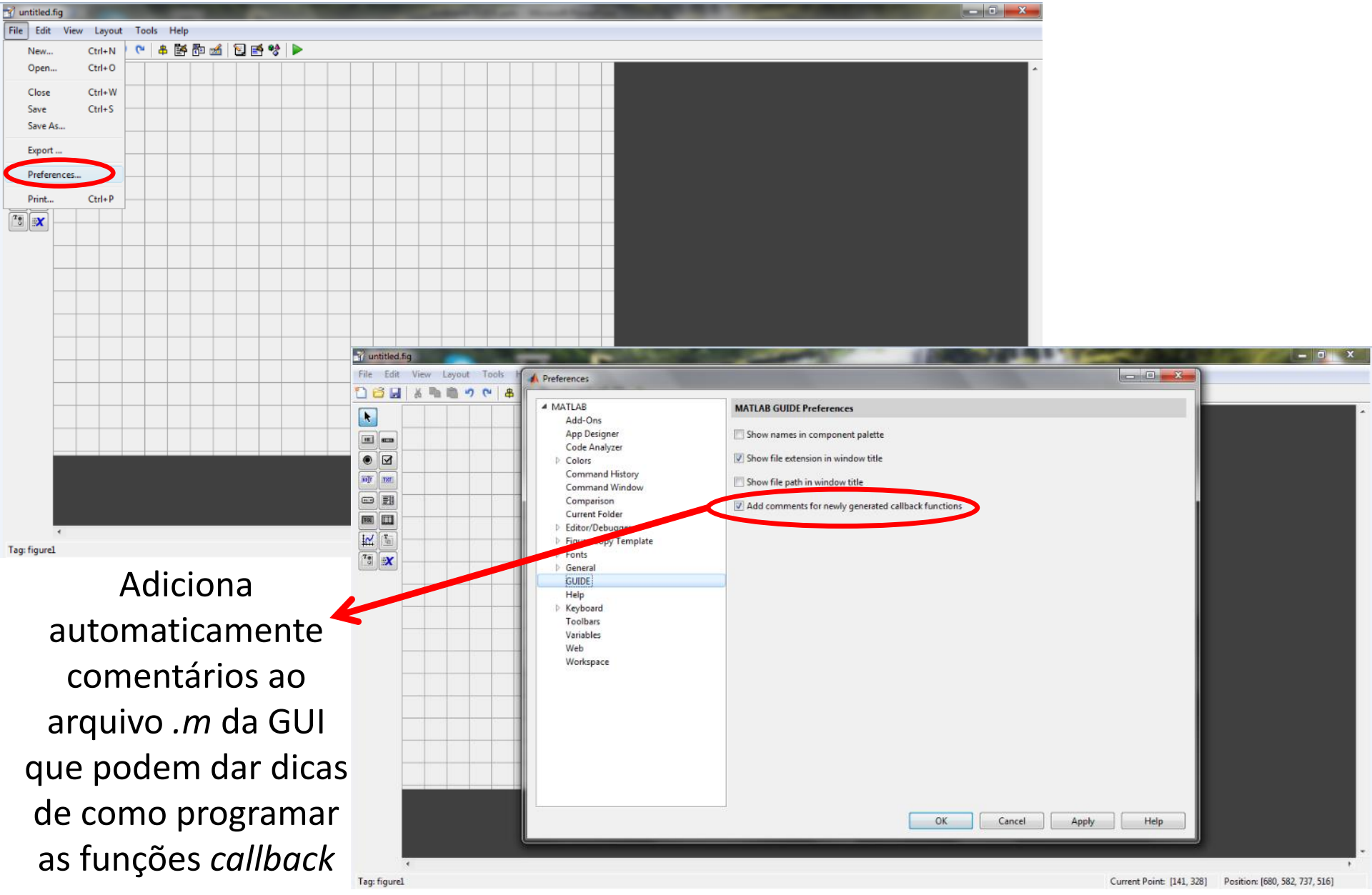
- **GUI allows only one instance to run (singleton)**

Quando selecionada, esta opção permite que apenas uma janela da interface criada seja aberta de cada vez. O que não permite que janelas desta mesma interface se sobreponham.

- **Use system color scheme for background**

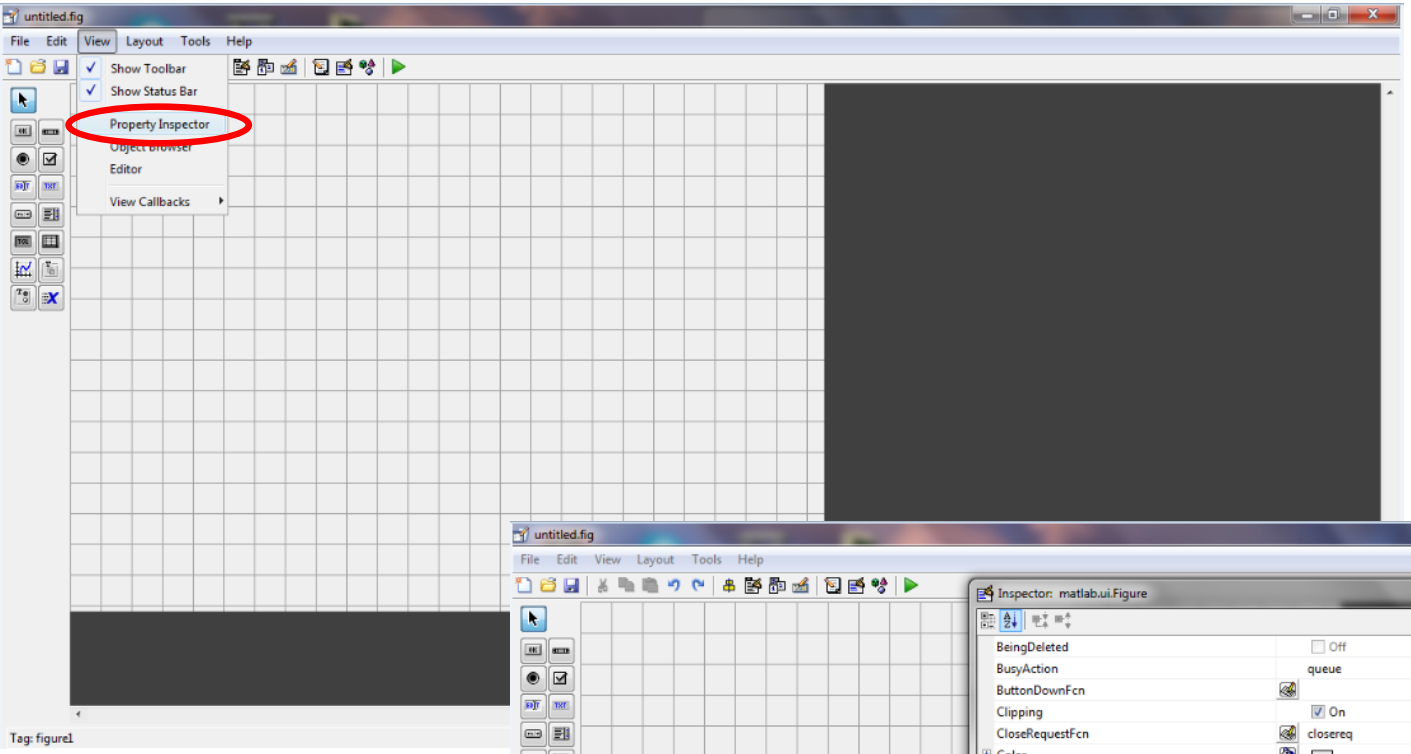
Faz com que a cor de fundo da janela seja a mesma cor dos componentes automaticamente.

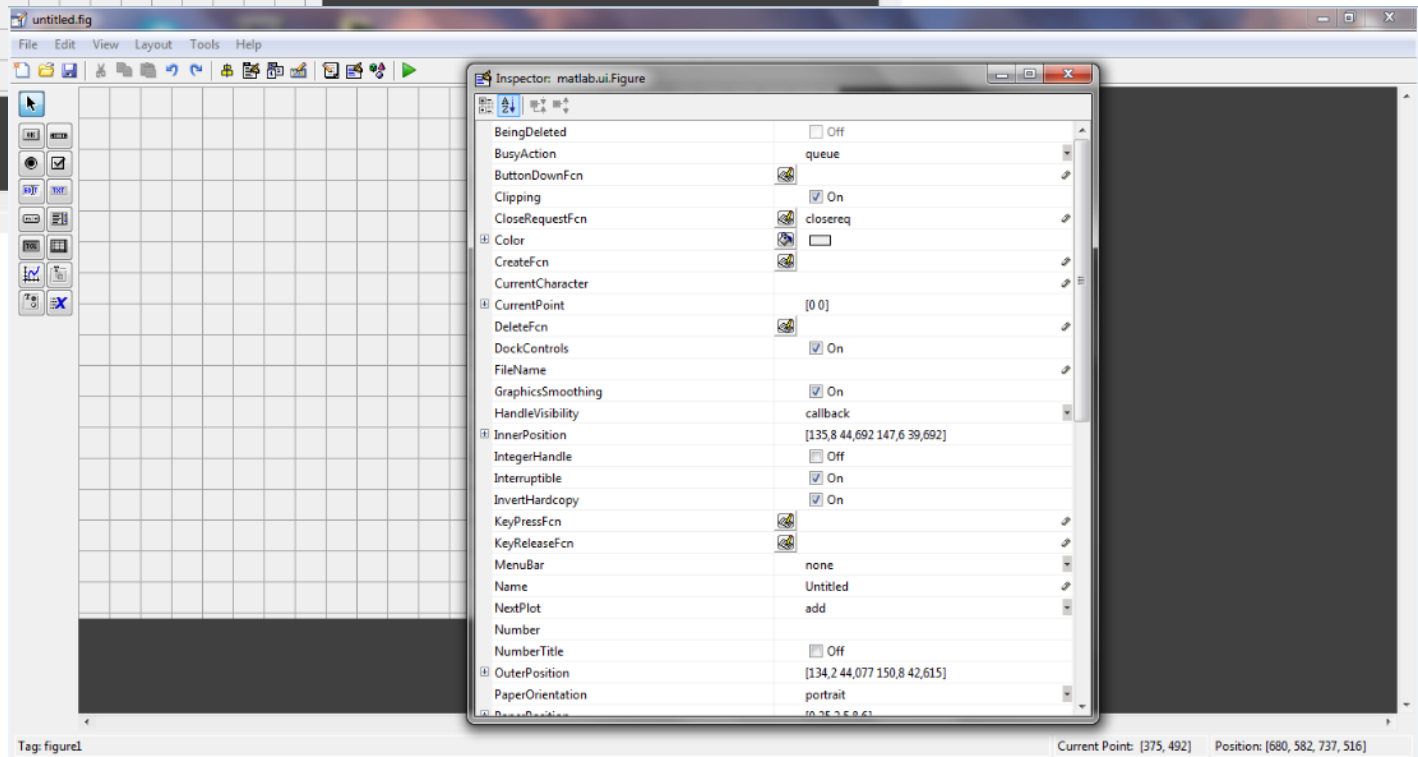
Alterando as Preferências da GUI



Adiciona automaticamente comentários ao arquivo *.m* da GUI que podem dar dicas de como programar as funções *callback*

Alterando as Propriedades da Janela



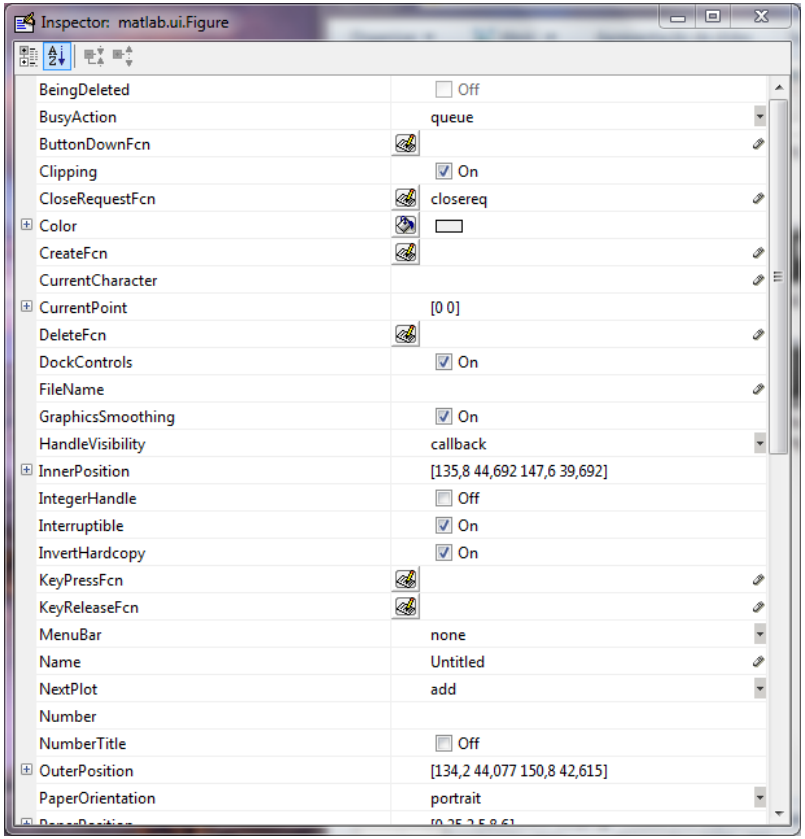


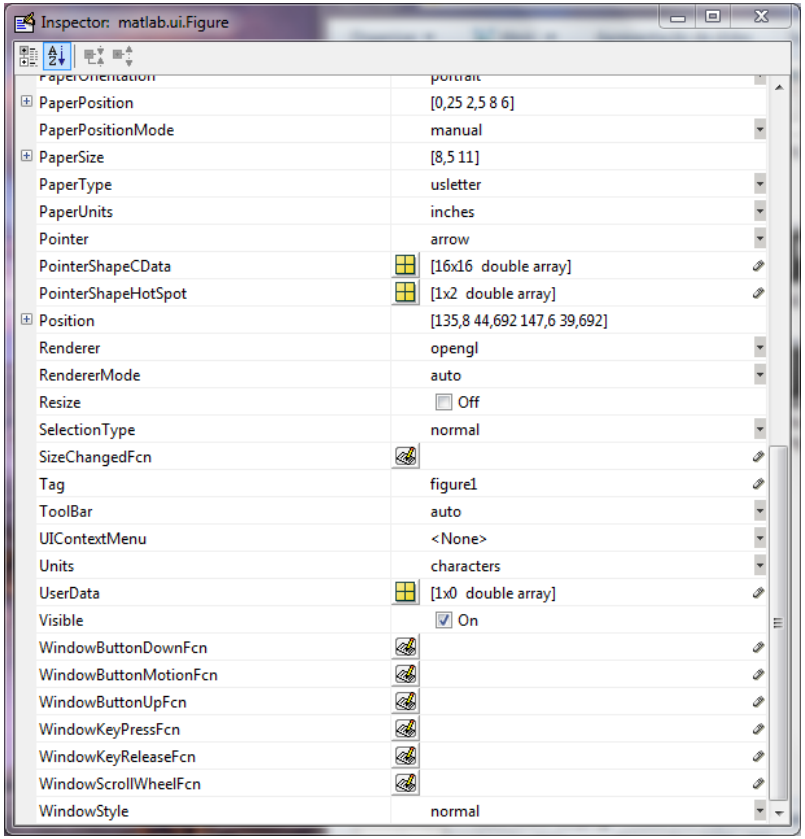
O *Property Inspector* da janela de interface ou qualquer outro componente também pode ser acessado com um duplo clique.

Alterando as Propriedades da Janela

O *Property Inspector* da acesso às propriedades específicas de um componente gráfico e permite modifica-las para estabelecer a configuração padrão de inicialização.

Essas propriedades também podem ser acessadas e modificadas nas funções *callback*, janela de comandos ou script, dependendo da opção selecionada em “Command-line accessibility”.





Alterando as Propriedades da Janela

Algumas propriedades básicas da janela de interface gráfica:

- **MenuBar/Toolbar**: Permite adicionar o menu padrão de figuras do MATLAB com a opção “figure”. Este menu não é adicionado à janelas com a propriedade “WindowStyle” selecionada como “Modal”.

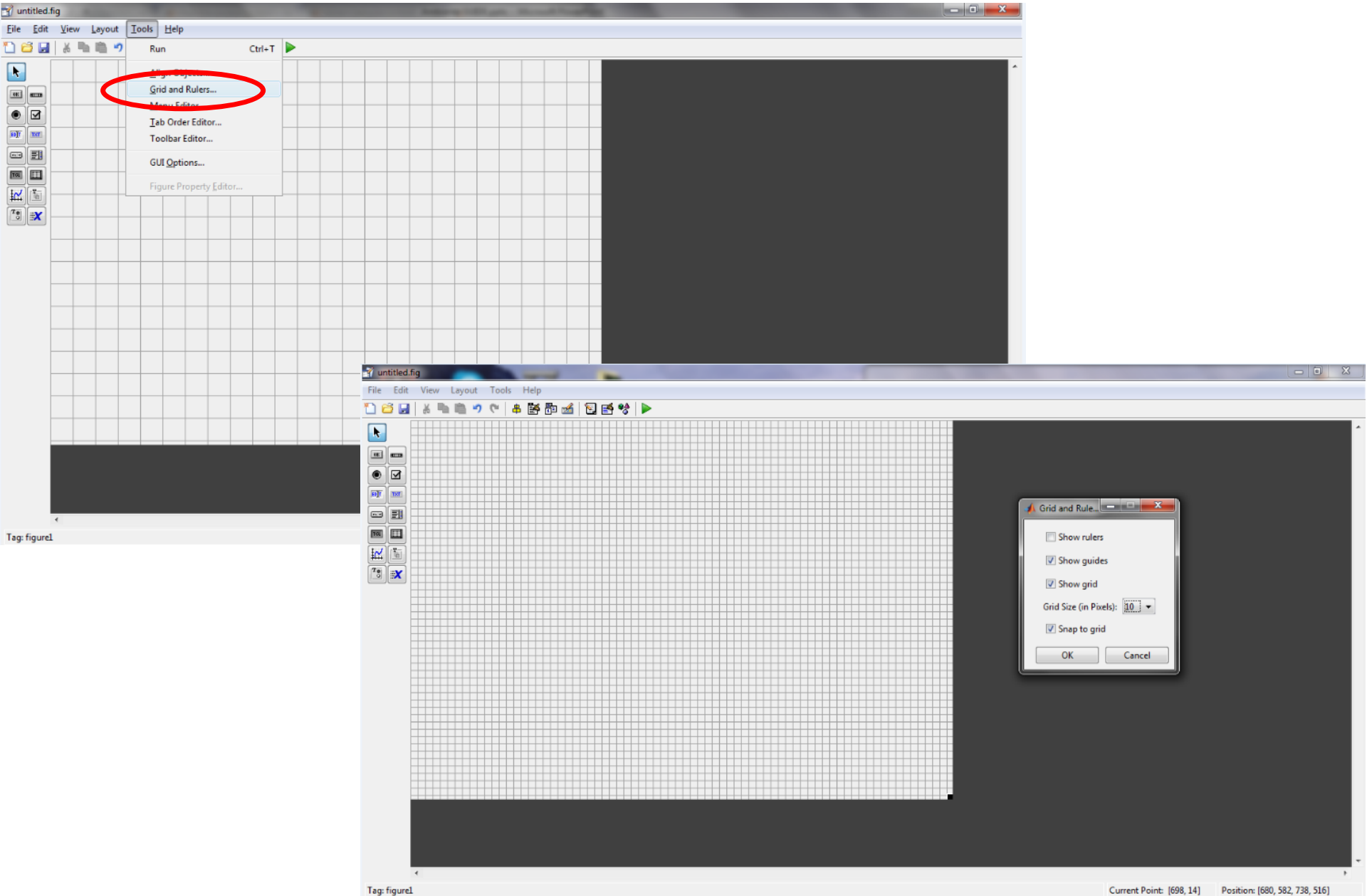
- **Name**: Define o nome que aparecerá na barra superior da janela.

- **Tag**: Propriedade presente em todos os componentes que define o nome pelo qual estes são identificados no código. É importante sempre modificar essa propriedade para cada componente adicionado para que o código não fique confuso, já que o nome padrão é a numeração na qual os componentes são adicionados.

- **WindowStyle**: Quando a opção “modal” é selecionada a janela permanece fixa na tela, impedindo que o usuário clique for a desta antes de fecha-la.

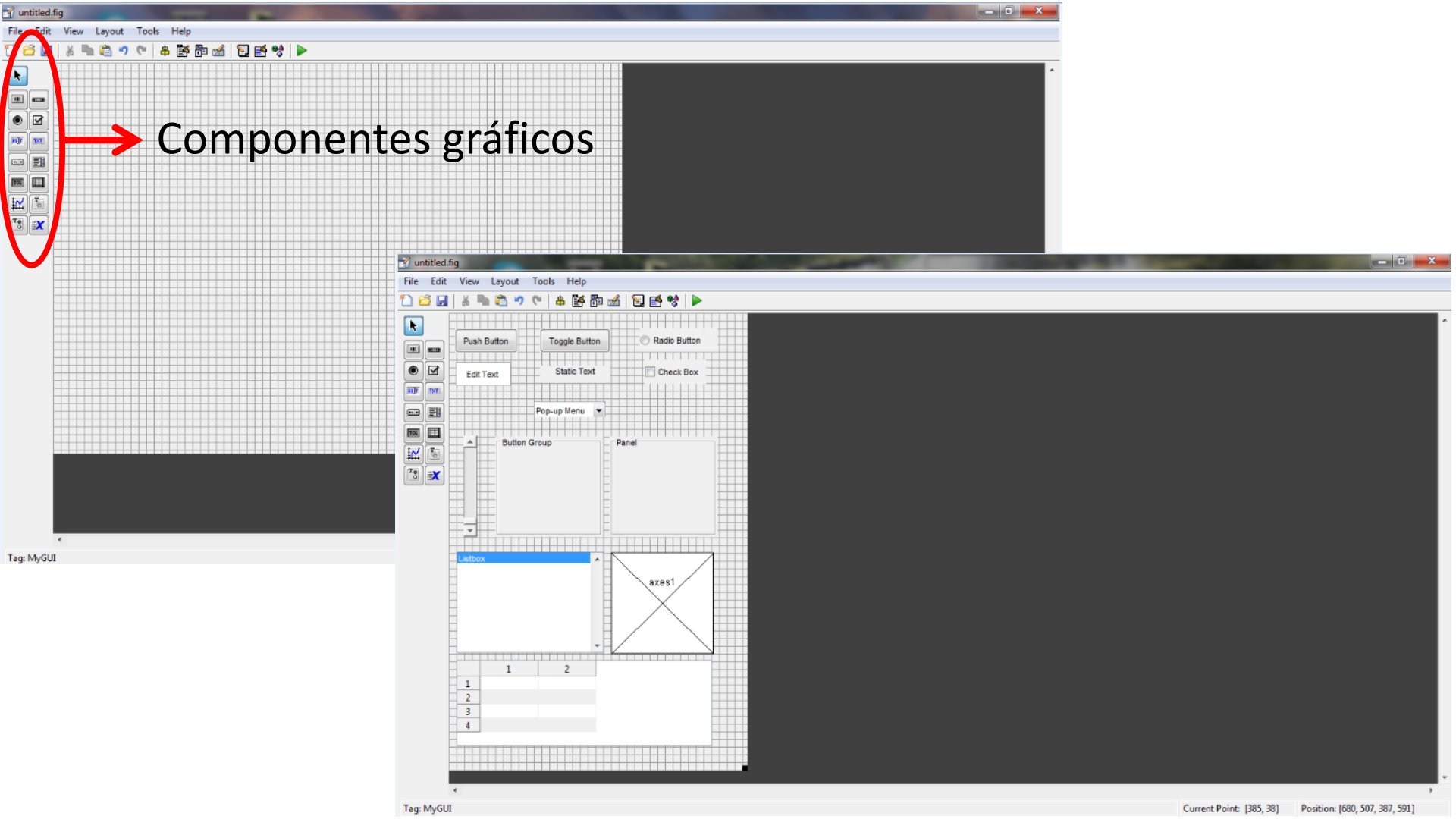
Descrição de todas as propriedades da janela de interface:
https://www.mathworks.com/help/matlab/ref/figure-properties.html

Adicionando Componentes Gráficos



Adicionando Componentes Gráficos

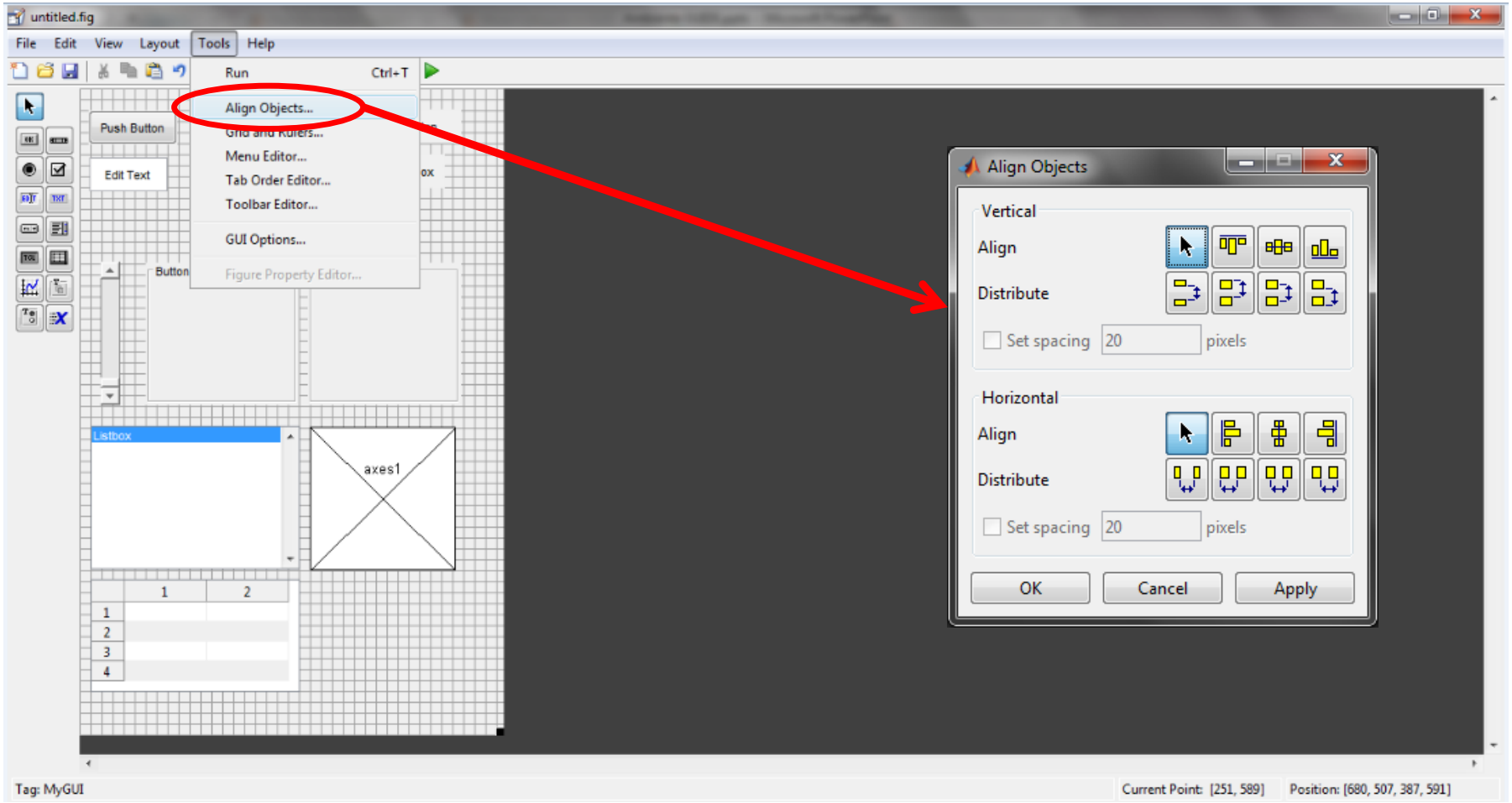
O GUIDE funciona como um sistema “drag and drop”, no qual os componentes gráficos podem ser arrastados e posicionados manualmente na janela de interface



Adicionando Componentes Gráficos

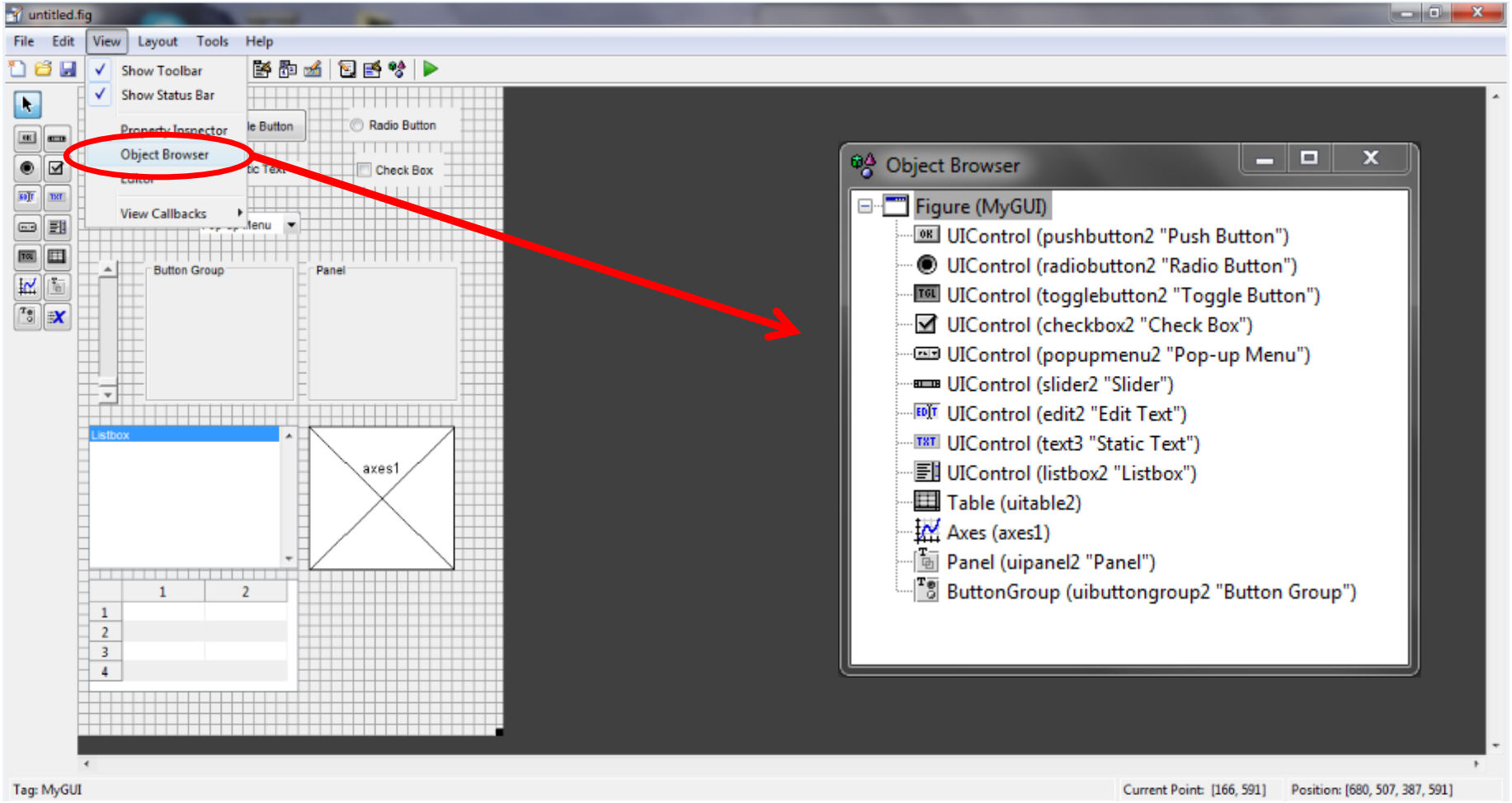
Recomenda-se utilizar a ferramenta “Align Objects” para organizar os componentes na janela.

Além disso, é possível modificar manualmente a propriedade “Position” no Property Inspector de cada componente.



Componentes Gráficos

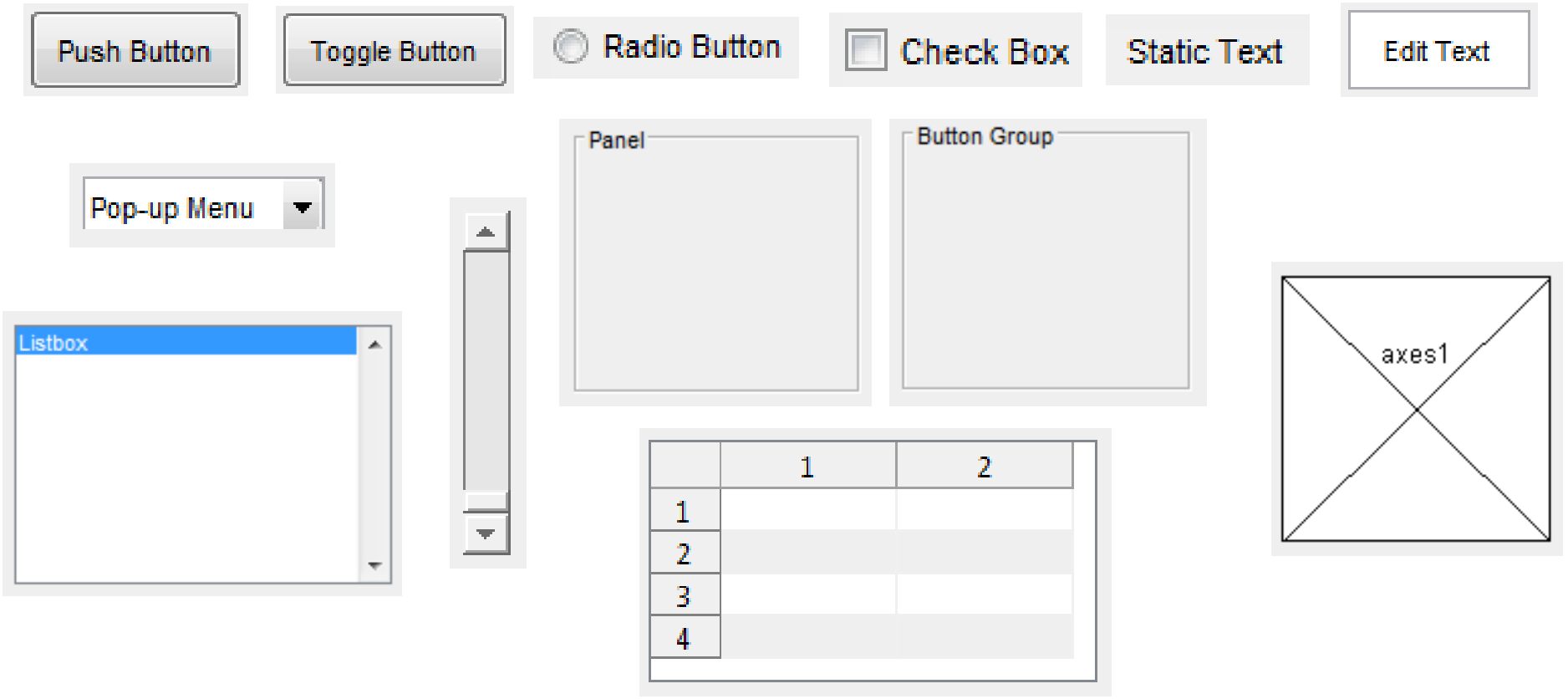
Object Browser: Mostra a organização hierárquica de todos os componentes inseridos



Componentes Gráficos

Cada tipo de componente gráfico possui propriedades particulares que podem ser acessadas e alteradas no *Property Inspector*, ou nas funções *callback*.

Por possuírem funcionalidades distintas, cada tipo de componente possui um grupo de ações que um usuário pode exercer sobre ele, sendo que para cada uma dessas ações é possível associa-la à uma função *callback*.



Componentes Gráficos

Algumas propriedades comuns à diversos tipos diferentes de componentes gráficos:

- **Enable**: Permite tornar um componente desabilitado ou inativo para o usuário. No primeiro caso, o componente fica com uma aparência sombreada, já no segundo continua com a mesma aparência de como se estivesse habilitado porém usuários não são capazes de utiliza-lo.

- **Estilos de Fonte**: Grupo de propriedades presente em qualquer componente que exiba algum texto e que permitem definir o tipo, tamanho, cor e orientação da fonte.

- **Position**: Permite definir as coordenadas X e Y do canto inferior direito do componente em relação à origem da janela de interface ou do painel no qual esteja inserido. Também inclui a possibilidade de alterar a largura e altura do componente.

Componentes Gráficos

- **String**: Propriedade também presente em componentes que exibam algum texto e que permite definir o texto exibido.

- **Tag**: Como já mencionado, essa propriedade é presente em todos os componentes e define o nome pelo qual estes são identificados no código. É importante sempre modificar essa propriedade para cada componente adicionado.

- **Visible**: Propriedade presente em todos os componentes que, se desativada, torna o componente invisível.

Descrição de todas as propriedades dos componentes gráficos:
https://www.mathworks.com/help/matlab/ref/uicontrol-properties.html

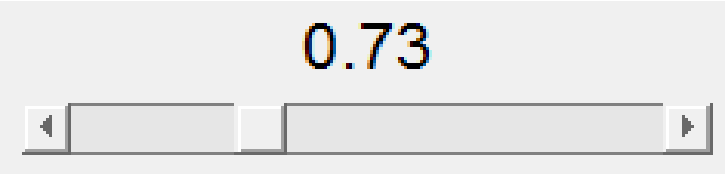
Componentes Gráficos: Static Text

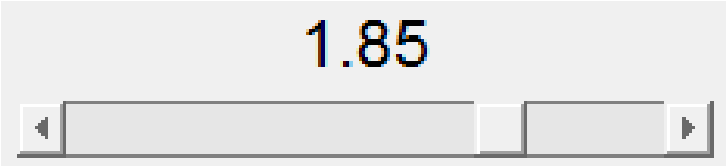
Caixa de texto para exibir strings que não pode ser modificada pelo usuário.

Geralmente este componente não está associado à nenhuma função *callback*, sendo utilizado apenas para rotular outros elementos e fornecer informações ao usuário.

Algumas vezes a informação exibida deve ser modificada, para isso a propriedade "String" deve ser acessada em um função *callback* e modificada.

Ex.: *Static Text* que indica o valor de uma barra de escala (componente do tipo *Slider*)





Componentes Gráficos: Edit Text

Caixa de texto que pode ser alterada pelo usuário para inserir ou editar uma string.

Para tornar a caixa de texto multilinha basta alterar as propriedades “Max” e “Min” para valores arbitrários em que a diferença do valor da primeira e da segunda seja maior do que 1, caso contrário a caixa permite apenas uma linha de texto.

Este componente também geralmente não está associado à nenhuma função *callback*, sendo muitas vezes utilizado para permitir que usuários forneçam valores numéricos ao programa.

Esses valores fornecidos pelos usuários nas caixas de texto podem ser obtidos acessando a propriedade “String” do respectivo componente.

Deve-se atentar ao fato de que, por mais que o usuário tenha inseiro um número, este é tido como uma cadeia de caracteres. Para realizar cálculos com este valor, deve-se fazer a conversão de string para um valor numérico (comando str2num ou str2double).

A propriedade “String” só é alterada quando se clica for a da caixa de texto ou a tecla “Enter” é pressionada (“Enter” + Ctrl para caixas multilinha).

Componentes Gráficos: Push Button

Botão simples para realizar alguma ação quando ativado por um clique do botão esquerdo do mouse.

Esta ação (resposta do programa ao evento de clique do mouse sobre o botão) deve ser programada em uma função callback que é disparada no instante em que se libera o clique no interior do botão.

Componentes Gráficos: Toggle Button

Semelhante ao *Push Button*, porém possui dois estados: ativado (pressionado) e desativado (não pressionado).

Quando ativado, a propriedade “Value” adquire o valor da propriedade “Max”, e quando desativado adquire o valor da propriedade “Min”. Ambas as propriedades “Max” e “Min” podem ter valores quaisquer, desde que sejam diferentes. (O padrão é que esses valores sejam 0 e 1)

Portanto, para verificar se um *Toggle Button* está ativado ou desativado, basta acessar e comparar o valor da propriedade “Value” com as propriedades “Max” e “Min”.

Uma função *callback* é disparada sempre que este componente é clicado.

Botão Desativado

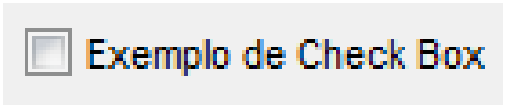
Botão Ativado

Componentes Gráficos: Check Box

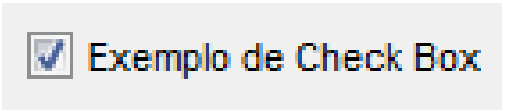
Assim como *Toggle Buttons*, possui o estado ativado e desativado. Quando ativado, a propriedade “Value” adquire o valor da propriedade “Max”, e quando desativado adquire o valor da propriedade “Min”.

Este componente é utilizado quando se deseja prover ao usuário opções independentes, que podem ter seleção múltipla.

Uma função *callback* é disparada sempre que este componente é clicado.



Caixa Desativada



Caixa Ativada

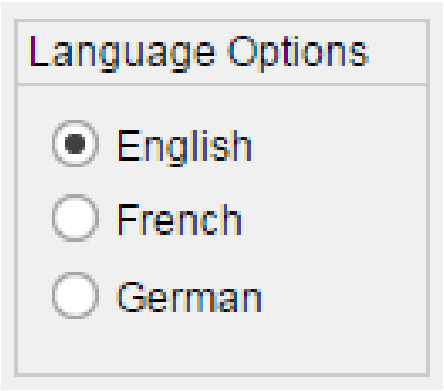
Componentes Gráficos: Radio Button e Button Group

Componente similar à *Check Box*, porém a diferença é que este pode ser utilizado para dar opções mutualmente exclusivas ao usuário, ou seja, apenas um *Radio Button* de um grupo pode ser selecionado.

A maneira mais simples de tornar um grupo de *Radio Buttons* (ou também *Toggle Buttons*) mutualmente exclusivos é inseri-los dentro de um *Button Group*, que nada mais é do que um painel que relaciona estes componentes.

O nome de um *Button Group* pode ser alterado através da propriedade “Title”.

Ex.: Seleção de idioma:



Componentes Gráficos: Pop-up Menu

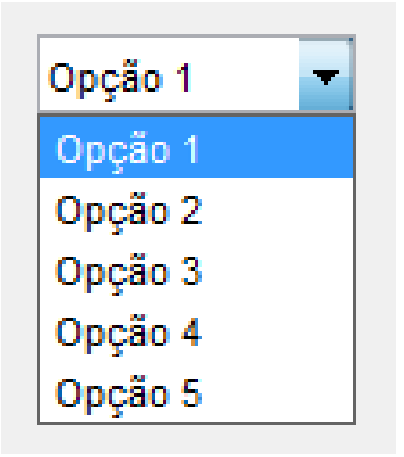
Componente que exibe uma lista de opções quando aberto, e quando fechado indica a opção corrente.

Geralmente é utilizado quando se deseja criar uma lista de opções mutualmente exclusivas, porém não se dispõe de espaço suficiente para apresentar tais opções em um grupo de *Radio Buttons* ou *Toggle Buttons.*

Para editar as opções deve-se modificar a propriedade “String”, em que cada linha corresponde a uma opção diferente.

Para determinar qual a opção que está selecionada, deve-se acessar a propriedade “Value”, cujo valor indica o número da opção, ou seja, se a terceira opção estiver selecionada, a propriedade “Value” terá o número 3 como valor.

Uma função *callback* é disparada sempre que a opção é trocada.



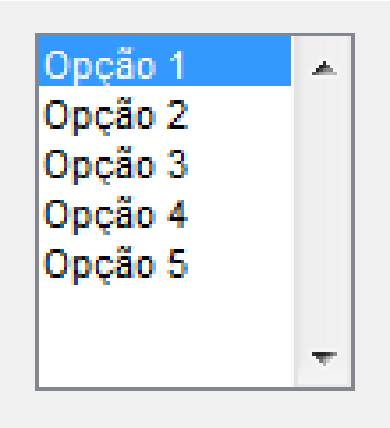
Componentes Gráficos: List Box

Uma *Listbox* exibe uma lista de itens na qual é possível selecionar vários ao mesmo tempo.

Para habilitar a seleção múltipla, a diferença entre os valores das propriedades “Max” e “Min” deve ser maior que 1, caso contrário, só será permitida seleção única de intens.

Para editar as opções deve-se modificar a propriedade “String”, em que cada linha corresponde a uma opção diferente.

A propriedade “Value” é um vetor que armazena os índices das linhas selecionadas.



Componentes Gráficos: Slider

Sliders são componentes geralmente utilizados para variar um valor numérico entre um valor mínimo e um máximo.

É atribuída à propriedade “Value” um valor que corresponde a posição do objeto deslizante que é proporcional aos valores mínimo e máximo atribuídos às propriedades “Min” e “Max”. Quando o objeto está no extremo esquerdo do *Slider*, a propriedade “Value” assume o valor da propriedade “Min”, e quando se encontra no extremo direito assume o valor da propriedade “Max”.

Na propriedade “SliderStep” é possível configurar o tamanho do movimento do objeto deslizante quando se clica em uma das setas que modificam a posição do objeto deslizante.

Uma função *callback* é disparada somente quando o objeto deslizante é solto, portanto se o objeto for arrastado com o mouse o valor da propriedade “Value” não é alterado dinamicamente.

Componentes Gráficos: Table

Componente para armazenar dados textuais ou numéricos em forma de tabela.

O número e formatação de linhas e colunas pode ser definido editando a propriedade “ColumnFormat”. Os dados podem ser acessados e editados através da propriedade “Data”, na qual é possível fornecer um vetor ou matriz de dados que serão exibidos na tabela.

Componentes Gráficos: Panel

Tem apenas o objetivo de organizar a interface do programa em painéis que agrupam diferentes tipos de componentes.

O nome de um painel pode ser alterado através da propriedade “Title”.

Componentes Gráficos: Axes

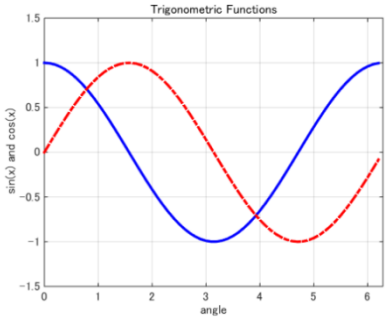
Eixos são onde os objetos gráficos são plotados e geralmente são utilizados como o canvas da interface gráfica.

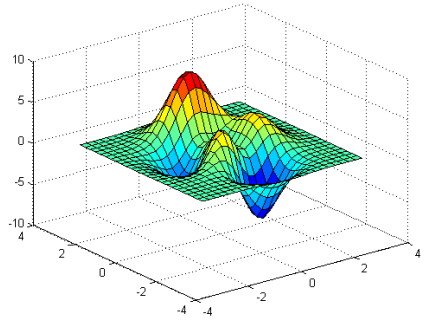
Quando um comando de plotagem é executado, este utiliza o eixo ativo que é o mais recentemente criado ou o último utilizado. Em interfaces com mais de um eixo, deve-se especificar qual eixo é o ativo antes de executar qualquer comando de plotagem para evitar que se utilize o eixo errado.

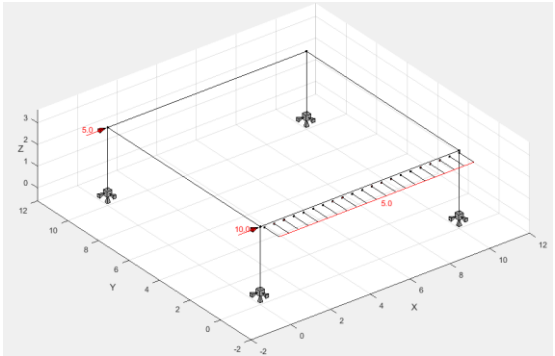
Imagens que se deseja exibir na interface devem ser carregadas em um eixo.

Os eixos são naturalmente tridimensionais e possuem diversas propriedades e comandos que permitem controlar a visualização dos objetos gráficos.

https://www.mathworks.com/help/matlab/ref/axes-properties.html

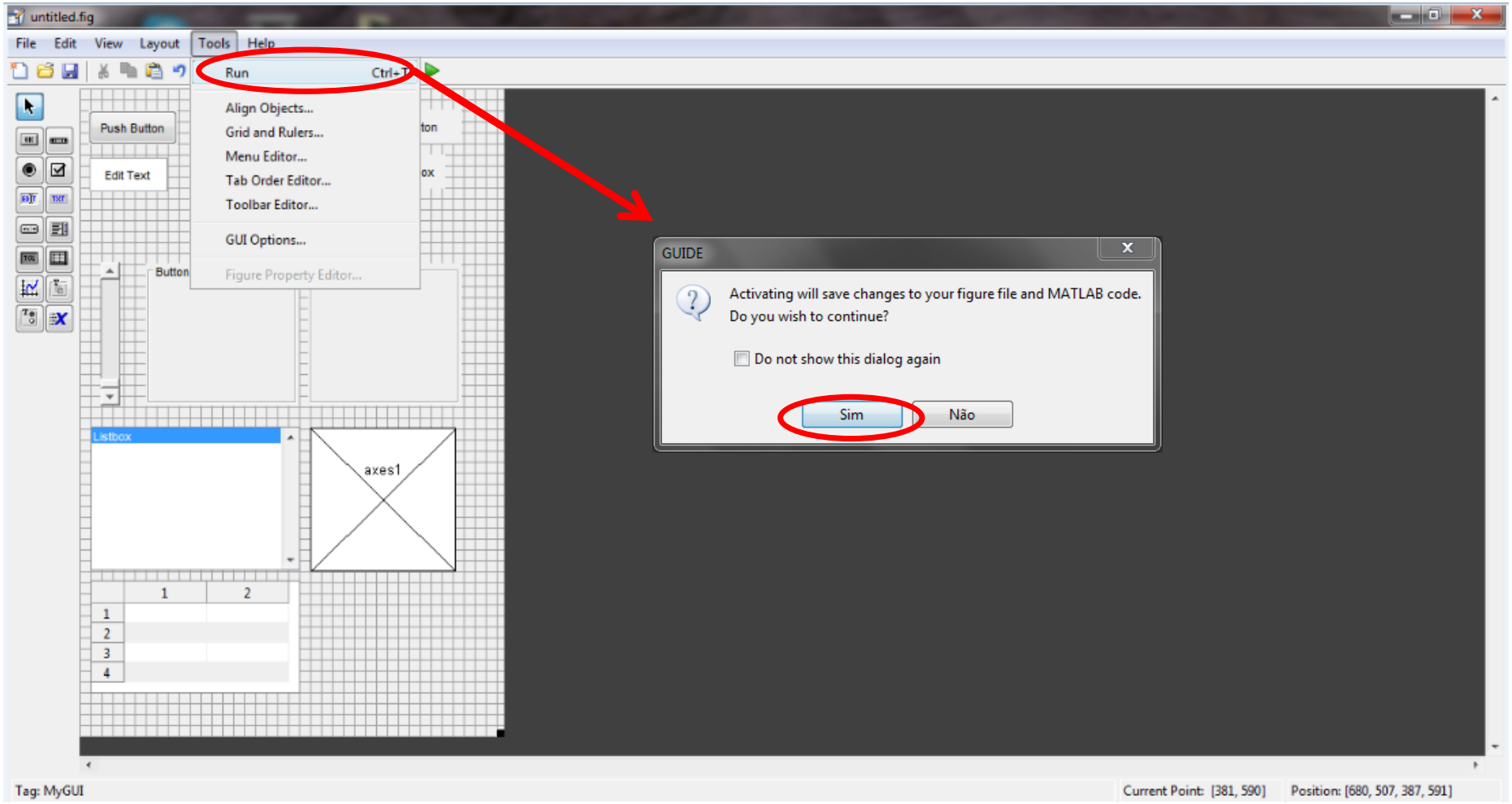






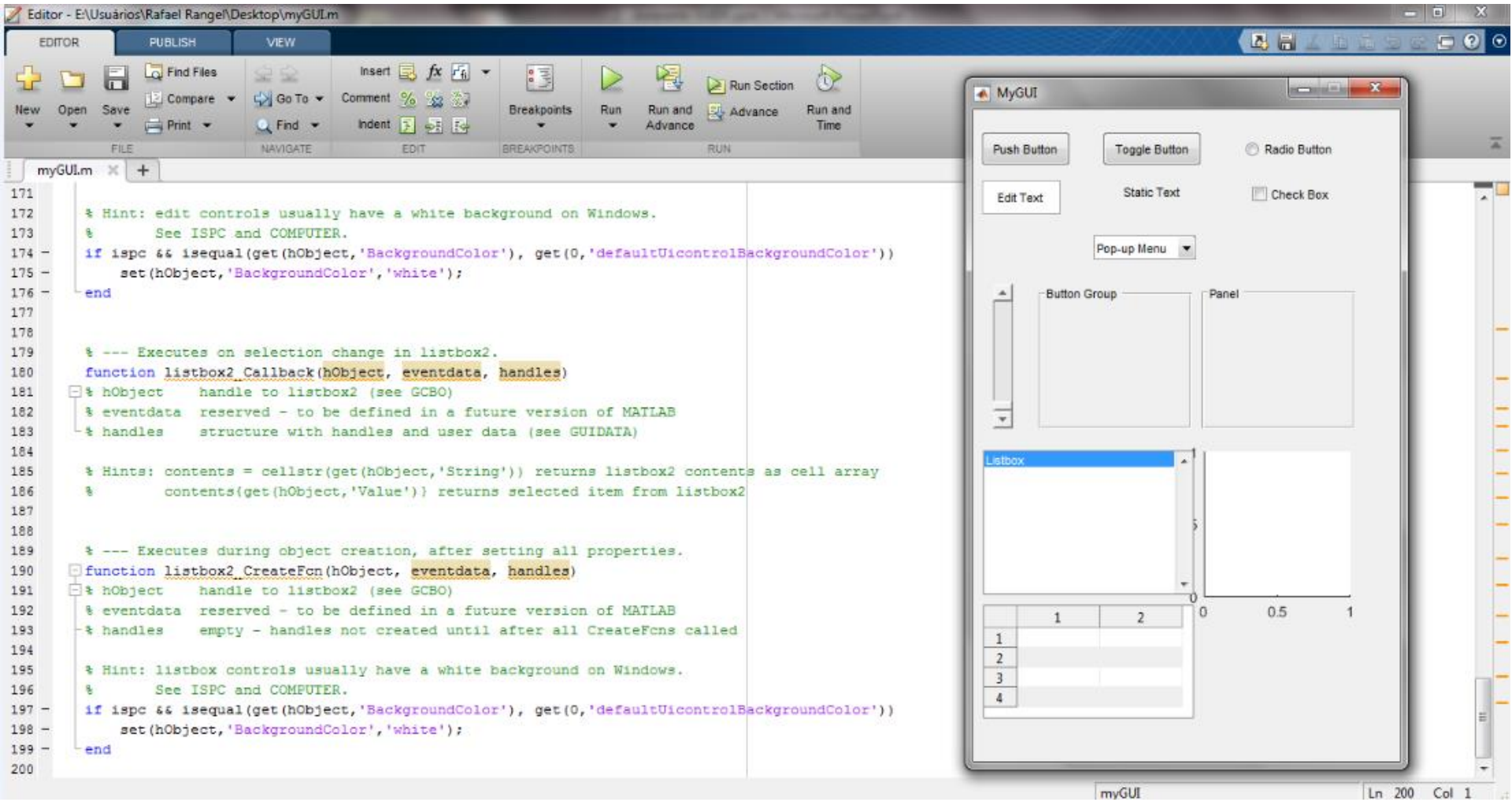
Salvando e Rodando a GUI

Ao rodar (Run) a GUI, será perguntado onde se deseja salvar os arquivos *.fig* e *.m*.



Salvando e Rodando a GUI

Após selecionar o diretório e salvar a GUI, a janela de interface será aberta junto com o arquivo MATLAB com as funções *callback* que deverão ser programadas.



Arquivo de Funções Callback

Neste arquivo são automaticamente geradas as funções *callback* relacionadas aos principais eventos que podem ocorrer em cada componente gráfico da interface.

Para aplicações simples, essas funções geradas automaticamente são suficientes, porém para aplicações com funcionalidades mais complexas pode ser necessário criar novas funções associadas aos eventos desejados.

Mesmo que algumas funções criadas automaticamente não sejam utilizadas, não se deve remove-las do arquivo.

Ao remover ou adicionar componentes da interface, o arquivo atualiza automaticamente as funções após salvar ou rodar a interface na GUIDE.

Comentários foram automaticamente gerados para explicar a utilidade de cada função e dar dicas de como usa-las.

As funções *callback* são um tipo de função chamado de *Nested Function*, que podem ser agrupadas todas em um mesmo arquivo, e não necessitam da declaração “end” para finalizá-las.

Arquivo de Funções Callback

- **Funções Callbacks Padrões Geradas Automaticamente:**

  - **myGUI** (Sempre deve ter o mesmo nome dado aos arquivos)
    Responsável por inicializar a interface. Não se deve alterá-la.

  - **myGUI_OpeningFcn**
    Função de abertura da interface. Nela pode-se programar as propriedades iniciais dos componentes e tudo que deve ocorrer no momento em que a GUI é aberta.

  - **myGUI_OutputFcn**
    Função cujo retorno é exibido na janela de comando.

  - **pushbutton1_Callback**
    Callback para programar a resposta do programa ao clique no *Push Button* cuja tag é “pushbutton1”.

  - **radiobutton1_Callback**
    Callback para programar a resposta do programa ao clique no *Radio Button* cuja tag é “radiobutton1”.

Arquivo de Funções Callback

- **togglebutton1_Callback**

Callback para programar a resposta do programa ao clique no *Toggle Button* cuja tag é “togglebutton1”.

- **checkbox1_Callback**

Callback para programar a resposta do programa ao clique no *Check Box* cuja tag é “checkbox1”.

- **popupmenu1_Callback**

Callback para programar a resposta do programa ao trocar a opção do *Pop-up menu* cuja tag é “popupmenu1”.

- **popupmenu1_CreateFcn**

Callback executada no momento em que o componente “popupmenu1” é criado.

- **slider1_Callback**

Callback para programar a resposta do programa ao movimentar o *Slider* cuja tag é “slider1”. Essa função não é disparada dinamicamente enquanto o slider esteja sendo movimentado.

Arquivo de Funções Callback

- **slider1_CreateFcn**

Callback executada no momento em que o componente “slider1” é criado.

- **edit1_Callback**

Callback para programar a resposta do programa ao editar o texto da Caixa de Texto cuja tag é “edit1”.

- **edit1_CreateFcn**

Callback executada no momento em que o componente “edit1” é criado.

- **listbox1_Callback**

Callback para programar a resposta do programa ao selecionar uma opção do *List Box* cuja tag é “listbox”.

- **listbox1_CreateFcn**

Callback executada no momento em que o componente “listbox1” é criado.

Arquivo de Funções Callback

- **Acessando Propriedades dos Componentes:**

Todos os componentes da interface ficam “pendurados” como campos da estrutura *handles*.

Para obter uma determinada propriedade de um componente, e salva-la em uma variável qualquer “var”, deve-se utilizar o seguinte comando:

var = get(handles.TagDoComponente, ’NomeDaPropriedade’);

Ou

var = handles.TagDoComponente.NomeDaPropriedade;

Para alterar uma determinada propriedade de um componente:

set(handles.TagDoComponente, ’NomeDaPropriedade’, ‘ValorDesejado’);

Ou

handles.TagDoComponente.NomeDaPropriedade = ValorDesejado;

Arquivo de Funções Callback

- **Passando Informações entre as Funções:**

As variáveis utilizadas em uma função *callback* tem um escopo local, ou seja, existem somente enquanto a função está sendo executada e são automaticamente deletadas da memória assim que a execução da função é encerrada.

Para ter acesso ao valor armazenado em uma variável à partir de qualquer função, é necessário salvar essa variável na estrutura *handles*, já que essa estrutura é passada como argumento de entrada para todas as funções:

```
handles.var = var;
```

Para obter o valor da variável “var” à partir de qualquer função:

```
var = handles.var;
```

Sempre que a estrutura *handles* for alterada, deve-se salvar essas alterações inserindo o seguinte comando ao final de cada função callback que tenha modificado a *handles*:

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

Arquivo de Funções Callback

Todas as funções *callback* também recebem um argumento de entrada chamado *hObject*, que nada mais é do que a handle para o componente que ativa a própria função.

Por exemplo, na função que é ativada pelo evento de clique no componente "pushbutton1", pode-se substituir *handles.pushbutton1* por *hObject*.

Arquivo de Funções Callback

- **Múltiplas Interfaces Gráficas:**

Em aplicações que necessitam de mais de uma janela de interface, deve-se criar cada GUI independentemente e salvar seus arquivos no mesmo diretório.

O comando para abrir uma janela de interface é simplesmente digitar o seu nome.

Quando se tem mais de uma interface, pode ser necessário transmitir informações entre os arquivos das diferentes GUIs, porém o escopo da estrutura *handles* é local aos arquivos de cada GUI.

Neste caso, uma solução possível é salvar a variável desejada na raiz, que pode ser acessada de qualquer arquivo, através do seguinte comando:

setappdata(0, 'var', var);

Para buscar a variável salva na raiz, utiliza-se o seguinte comando:

var = getappdata(0, 'var');

Funções Úteis

- Mover a Janela para o Centro da Tela: *movegui(gcf, 'center')*

- Caixas de Diálogo:
  Diferentes tipos de caixas de diálogo predefinidas podem ser exibidas com comandos simples, disponíveis em:

  https://www.mathworks.com/help/matlab/predefined-dialog-boxes.html

- Manipulação dos Eixos
  Especificar o eixo ativo: *axes(handles.TagDoEixo);*
  Variável que armazena o eixo ativo: *gcf*
  Limpar todos os objetos gráficos: *cla reset;*
  Objetos gráficos não sejam substituídos na próxima plotagem: *hold on;*
  Especificar os limites de um eixo: *xlim([a, b]);* ou *ylim([a, b]);* ou *zlim([a, b]);*
  Fazer com que os eixos X, Y e Z tenham a mesma escala: *axis equal;*
  Ativar a grade e/ou régua lateral: *grid on;* e *ruler on;*
  Definir as coordenadas do centro da câmera: *campos(x, y, z);*
  Obter as coordenadas do centro da câmera: *var = campos;*
  Mover a câmera para uma visão 2D ou 3D: *view(2);* ou *view(3);*

Funções Úteis

- Manipulação de Arquivos:
  Criar um arquivo txt para escrita: *txt = fopen(fullname, 'wt');*
  Criar um arquivo txt para leitura: *txt = fopen(fullname, 'rt');*
  Abrir um arquivo na tela: *winopen('NomeDoArquivo.extenção');*
  Imprimir dados: *fprintf(txt, 'texto ou número %f', valor);*
  Ler dados: *fscanf(txt, '%f', NúmeroDeDados);*