**Desarrollo de GUIs en MATLAB**

**Jorge De Los Santos**



Versión 1.0 (Febrero 2014)

El autor agradece cualquier sugerencia, observación o comentario

delossantosmfq@gmail.com || matlabtyp@gmail.com

Obtenga la versión más reciente en*:* [*http://matlab-typ.blogspot.mx*](http://matlab-typ.blogspot.mx/)

Este texto se distribuye bajo licencia:



Para consultar detalles acerca de la licencia visite: https://creativecommons.org/licenses/by-nc-nd/2.5/mx

Contenido

[Capítulo I. Introducción a MATLAB 7](#_Toc394694658)

[1.1 ¿Qué es MATLAB? 7](#_Toc394694659)

[1.2 Tipos de datos y operadores 7](#_Toc394694660)

[1.3 MATLAB como una calculadora 8](#_Toc394694661)

[1.4 Ficheros de comandos 8](#_Toc394694662)

[1.5 Funciones y subfunciones 9](#_Toc394694663)

[1.6 Sentencias de control y bucles 10](#_Toc394694664)

[1.8.1 Sentencia if-elseif-else 10](#_Toc394694665)

[1.8.2 Sentencia switch 11](#_Toc394694666)

[1.8.3 Bucle for 12](#_Toc394694667)

[1.8.4 Bucle while 13](#_Toc394694668)

[1.7 Vectores y matrices 15](#_Toc394694669)

[1.8 Cell arrays y estructuras 15](#_Toc394694670)

[Capítulo II. GUIs, una primera aproximación 17](#_Toc394694671)

[2.1 ¿Qué es una GUI? 17](#_Toc394694672)

[2.2 GUIs en MATLAB 17](#_Toc394694673)

[Capítulo III. Figure y Axes 20](#_Toc394694674)

[Capítulo IV. Controles gráficos 23](#_Toc394694675)

[Capítulo V. Cuadros de diálogos 25](#_Toc394694676)

[1.1 Elemento figure 28](#_Toc394694677)

[1.2 Controles gráficos (uicontrol) 29](#_Toc394694678)

[1.3 Axes 30](#_Toc394694679)

[1.4 Tablas 31](#_Toc394694680)

[1.5 Paneles y Button Groups 31](#_Toc394694681)

[1.6 Menús 32](#_Toc394694682)

[1.7 Cuadros de diálogos predefinidos 32](#_Toc394694683)

[1.8 Funciones get y set 34](#_Toc394694684)

[1.9 Mi primer GUI 34](#_Toc394694685)

[2.1 Conversor de temperatura 36](#_Toc394694686)

[2.2 Modifica tamaño y color de fuente 39](#_Toc394694687)

## Acerca de este libro…

Este texto tiene como objetivo proporcionar una referencia para iniciarse en el mundo de las interfaces gráficas de usuario creadas en MATLAB. Es necesario que el lector esté familiarizado con el entorno de programación y conozca los principios básicos como las instrucciones de control, la definición de funciones, y el manejo de cell arrays y estructuras.

En la mayoría de los cursos de GUIs MATLAB que existen en la red se utiliza el entorno de desarrollo (GUIDE) para agilizar el proceso de construcción. Sin embargo en este curso se prescindirá del mismo, y se utilizarán solamente bloques de código para generar las GUI (Programmatic GUI).

El texto se divide en dos partes, la primera es una introducción breve a los elementos gráficos que componen una interfaz gráfica, haciéndose referencia a las propiedades más usuales de cada uno; en la segunda parte se exponen ejemplos de GUIs desarrolladas en MATLAB, cada una pretende mostrar cuestiones significativas que resulten útiles e interesantes.

Bienvenido.

**Jorge De Los Santos**

**CAPÍTULO I**

Introducción a MATLAB

# Capítulo I. Introducción a MATLAB

## 1.1 ¿Qué es MATLAB?

MATLAB es un lenguaje de programación de alto nivel y entorno de desarrollo interactivo, utilizado para numerosas aplicaciones de carácter técnico y científicas. MATLAB permite realizar adquisición y análisis de datos, desarrollo de algoritmos computacionales, creación y simulación de modelos físicos y la visualización gráfica de procesos determinados. Entre los campos de uso de MATLAB se incluyen el procesamiento digital de audio, imágenes y vídeo, sistemas de control, finanzas computacionales, biología computacional, redes neuronales, etc.

## 1.2 Tipos de datos y operadores

En la tabla siguiente se resumen los tipos de datos nativos de MATLAB:

|  |  |
| --- | --- |
| **TIPO DE DATO** | **DESCRIPCIÓN** |
| Numérico | Dentro de este tipo de dato se incluyen enteros y números de coma flotante. Por defecto MATLAB maneja el tipo double para todos los valores numéricos. |
| Char | Cadenas de texto o caracteres. Se especifican utilizando comillas simples para que MATLAB pueda interpretarlos como tal. |
| Lógico | Los tipos de datos lógicos o booleanos sólo admiten dos valores (falso o verdadero). Su utilidad estriba en las comparaciones lógicas y/o toma de decisiones en las estructuras de control. |
| Cell array | Un cell array es un arreglo de datos que permite almacenar diversos tipos de elementos. |
| Estructura | Una estructura de datos, al igual que un cell array permite almacenar diversos tipos de datos, pero en este caso utilizando campos. |

## 1.3 MATLAB como una calculadora

## 1.4 Ficheros de comandos

Los ficheros de comandos, conocidos también como “scripts”, son archivos de texto sin formato (ASCII) con la extensión característica de los archivos de MATLAB (\*.m), se utilizan para almacenar una serie de comandos o instrucciones que se ejecutan sucesivamente y que habrán de realizar una tarea específica. Los scripts de MATLAB pueden editarse utilizando cualquier editor de texto sin formato (Bloc de Notas, Notepad++, Sublime Text, etc…), aunque es más recomendable utilizar el editor de MATLAB, puesto que proporciona herramientas que facilitan la corrección de errores, el control sobre la ejecución del código y la capacidad de autocompletado y sugerencias cuando se utilizan funciones nativas de MATLAB.

Para crear un nuevo script puede pulsar la combinación **Ctrl + N** (bajo SO Windows), o buscar en la interfaz de MATLAB la opción New y enseguida seleccionar Script; si prefiere hacerlo desde la ventana de comandos puede introducir el comando edit que le abrirá un nuevo script.

Para guardar un fichero de comandos utilice la opción Save de la barra de herramientas o bien mediante la combinación de teclas **Ctrl + S** en Windows. Debe tomarse en cuenta que al guardar un script se le proporcione un nombre que no entre en conflicto con las funciones nativas de MATLAB o las palabras reservadas del lenguaje. Algunas recomendaciones que deben seguirse para nombrar un script son:

* El nombre deberá contener sólo letras, números o guiones bajos.
* No deberá comenzar con un carácter diferente a una letra (Por ejemplo: 102metodo.m, es un nombre inválido dado que comienza con un número).
* Evite utilizar nombres de funciones nativas de MATLAB o palabras reservadas del lenguaje que podrían ocasionar conflictos.

## 1.5 Funciones y subfunciones

Las funciones son porciones de código que por lo general aceptan argumentos o valores de entrada y devuelven un valor de salida. Una función es una herramienta muy útil en la programación, dado que permite la reutilización de código para procedimientos que así lo requieran, lo cual implica una mayor productividad. MATLAB está compuesto por una multitud de funciones agrupadas en toolboxs, cada una de ellas pensada para resolver una situación concreta.

La estructura básica de una función contiene los siguientes elementos:

* La palabra reservada function
* Los valores de salida
* El nombre de la función
* Los argumentos de entrada
* Cuerpo de la función

Para una mejor comprensión de cada uno de esos elementos, refiérase a las siguientes líneas de código:

function res = suma(a,b)

res = a+b;

end

La función anterior llamada suma, recibe como argumentos de entrada dos valores numéricos a y b, y devuelve un resultado guardado en res que equivale a la suma aritmética de las variables de entrada. Si ejecutamos la función en la ventana de comandos obtenemos algo similar a esto:

>> s=suma(3,2)

s =

5

Si no hace una asignación el resultado devuelto se guarda en la variable ans.

## 1.6 Sentencias de control y bucles

### 1.8.1 Sentencia if-elseif-else

La sentencia if se utiliza como bifurcación simple por sí sola, es decir, en aquellas situaciones en las cuales se requiera evaluar solamente una condición, por ejemplo, suponga que tiene dos números a y b y necesita comprobar si son iguales y ejecutar una acción, para ello bastaría con una sentenciaif simple:

if a==b

disp('a es igual a b');

end

A diferencia del caso anterior hay situaciones que requieren la ejecución de una acción cuando la condición se cumpla y de otra en caso contrario, entonces puede utilizarse una bifurcación doble formada por las sentencias if-else. Retomando el ejemplo para la bifurcación if simple, podríamos modificarlo de tal manera que envíe también un mensaje (ejecute una acción) para cuando la condición no se cumple:

if a==b

disp('a es igual a b');

else

disp('a es diferente de b');

end

Ahora imagine que para los ejemplos anteriores se necesita especificar si a=b, si a>b o bien si a<b, lo cual implicaría tener una sentencia de selección múltiple if-elseif-else que permite escoger entre varias opciones, evaluándose en orden descendente, por ejemplo refiérase a la siguiente estructura:

if cond1

% Instrucciones

elseif cond2

% Instrucciones

elseif cond3

% Instrucciones

.

.

.

elseif condN

% Instrucciones

else

% Instrucciones

end

MATLAB evalúa primeramente la condición 1 contenida en la sentencia if (cond1) y en el caso de no cumplirse evalúa la siguiente condición de forma sucesiva (cond2, cond3, …); finalmente y en el caso de que ninguna de las opciones evaluadas se cumpla, se ejecuta la instrucción contenida en la sentencia else. A continuación se muestra el ejemplo de una bifurcación múltiple para la situación descrita al principio:

if a==b

disp('a es igual que b');

elseif a>b

disp('a es mayor que b');

elseif a<b

disp('a es menor que b');

end

### 1.8.2 Sentencia switch

La sentencia switch es una bifurcación múltiple que permite seleccionar entre varias opciones o casos la acción a ejecutar. La sintaxis general es:

switch var

case opc1

% Instrucciones

case opc2

% Instrucciones

.

.

.

otherwise

% Intrucciones

end

Siendo **var** la variable que servirá como criterio de selección. Después de la palabra reservada case, se coloca el valor de **var** para el cual se ejecutarán esas instrucciones, y en otherwise se insertan las instrucciones que MATLAB deberá ejecutar por defecto en caso de no cumplirse ninguno de los casos especificados.

Enseguida se muestran dos ejemplos correspondientes a la sentencia de selección switch:

X=input('Inserte 0 o 1: ');

switch X

case 0

disp('Insertó cero');

case 1

disp('Insertó uno');

otherwise

warning('Valor incorrecto, verifique');

end

letra=input('Inserte una letra: ','s');

switch letra

case {'a','e','i','o','u'}

disp('Es una vocal');

otherwise

disp('Es una consonante');

end

### 1.8.3 Bucle for

La sintaxis general de un bucle for se muestra enseguida:

for i=inicio:incremento:fin

% Instrucciones...

end

El valor **inicio** es a partir del cual se ejecutará el ciclo, el **incremento** es la cantidad que varía en cada paso de ejecución, y el valor de **final** establece el último valor que tomará el ciclo.

El siguiente código muestra un ciclo for muy básico, el cual simplemente muestra en consola el valor actual adquirido por la variable.

for i=1:10

fprintf('Valor actual: %g \n',i);

end

Cuando no se especifica el incremento, como el caso anterior, MATLAB asume que es unitario.

Es posible utilizar ciclos for anidados, por ejemplo para cuando se requiere recorrer una matriz en sus dos dimensiones y ejecutar operaciones elemento por elemento. Véase el siguiente ejemplo:

A=round(rand(5)\*10);

for i=1:5

for j=1:5

disp(A(i,j));

end

end

### 1.8.4 Bucle while

El bucle while se utiliza, por lo general, cuando no se tiene un rango definido sobre el cual se realice la ejecución del ciclo o bien cuando la terminación del mismo viene dada por una condición. La sintaxis más común es:

while cond

% Instrucciones

% ...

% ...

% ...

end

Donde **cond** es la condición que determina la finalización de ejecución.

Enseguida se muestra un ejemplo muy básico que muestra en pantalla el valor de una variable utilizada como referencia:

k=1;

while k<10

disp(k);

k=k+1;

end

Lo anterior muestra en consola el valor de **k** mientras esta sea menor a 10, es decir muestra todos los valores enteros en el intervalo [1 9], es importante notar que la variable **k** debe incrementarse en cada ciclo para que en un momento determinado la condición de finalización se cumpla, de lo contrario se convertiría en un bucle infinito.

Ahora, veamos un ejemplo más práctico. La aproximación de una raíz cuadrada por el método babilónico implica realizar n iteraciones mediante la siguiente expresión:



Donde **x** es el número del cual se calcula la raíz cuadrada. A continuación se muestra el código implementado en MATLAB utilizando un bucle while:

x=input('Introduzca un número positivo: ');

r=x;

ra=0;

while ra~=r

ra=r;

r=(1/2)\*(x/r+r);

end

fprintf('\nRaíz cuadrada de %g = %g\n\n',x,r);

Como se observa, en la variable **ra** se guarda la raíz aproximada calculada en una iteración anterior, de manera que esta sirva como comparación respecto a la nueva raíz calculada, el bucle termina cuando la diferencia entre el valor actual y el anterior es inferior a la tolerancia numérica (eps) soportada por MATLAB y por ende pasan a considerarse como valores iguales.

## 1.7 Vectores y matrices

## 1.8 Cell arrays y estructuras

**CAPÍTULO II**

GUIs, una primera aproximación

# Capítulo II. GUIs, una primera aproximación

## 2.1 ¿Qué es una GUI?

Una interfaz gráfica de usuario o GUI, por sus siglas en inglés (Graphical User Interface) es un elemento gráfico que contiene uno o más controles que están disponibles para interactuar con el usuario mediante un entorno visual sencillo, el cual permite la comunicación entre el usuario y el computador. Las GUIs se desarrollan con la finalidad de proporcionar a los usuarios, poco ambientados con el entorno de la programación, herramientas de tipo gráfico que permitan una interacción más intuitiva y además abstraerles de la codificación que hace funcionar al programa.

Sin ir más allá, el propio entorno de MATLAB es una interfaz gráfica que permita al usuario manipular con gran facilidad cada uno de sus elementos.

## 2.2 GUIs en MATLAB

Algunos de los componentes más comunes de una GUI creada en MATLAB son menús, barras de herramientas, botones, menús desplegables, cajas de texto, entre otros.

En las interfaces gráficas creadas en MATLAB pueden aprovecharse todas las herramientas matemáticas y de ingeniería que proporciona MATLAB, permiten además la manipulación de archivos de datos, así como la interacción con otras GUIs y mostrar datos mediante tablas y gráficas de gran calidad.

Generalmente las GUIs son programadas para que respondan a la manipulación del usuario con una acción específica. Los controles gráficos que componen una GUI están relacionados con una rutina de programación, llamados *callbacks* en el entorno MATLAB, que se ejecuta cuando sucede un determinado evento, que puede ser la entrada de caracteres mediante el teclado, el clic de un botón del mouse, o situarse sobre un objeto.

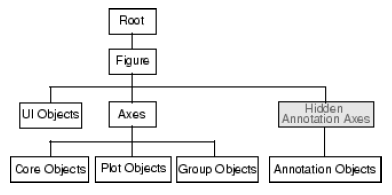
**CAPÍTULO III**

Figure y Axes

# Capítulo III. Figure y Axes

## 3.1 Los objetos gráficos en MATLAB

Los objetos gráficos en MATLAB son los elementos básicos utilizados para mostrar información en pantalla. Cada elemento es identificado mediante una referencia comúnmente llamada *handle*, mediante la cual pueden manipularse todas las propiedades o atributos del objeto. Además, es necesario hacer mención que los objetos gráficos tienen una estructura jerarquizada que permite definir qué objeto puede contener a otro, enseguida se muestra un esquema ejemplificando lo dicho:



Note que Root es el objeto de mayor jerarquía y que hace referencia al propio entorno gráfico de MATLAB. Las propiedades del objeto "raíz" puede obtenerlas utilizando la función get como sigue:

>> get(0, 'Propiedad')

Por ejemplo, con 'ScreenSize' puede obtener el tamaño de la pantalla:

>> get(0,'ScreenSize')

ans =

1 1 1366 768

O simplemente puede hacer lo siguiente para visualizar todas las características:

>> get(0)

CallbackObject = []

CommandWindowSize = [105 28]

CurrentFigure = []

Diary = off

DiaryFile = diary

Echo = off

FixedWidthFontName = Courier New

Format = short

FormatSpacing = loose

Language = es\_mx

MonitorPositions = [1 1 1366 768]

More = off

PointerLocation = [891 326]

RecursionLimit = [500]

ScreenDepth = [32]

ScreenPixelsPerInch = [96]

ScreenSize = [1 1 1366 768]

ShowHiddenHandles = off

Units = pixels

BeingDeleted = off

ButtonDownFcn =

Children = []

Clipping = on

CreateFcn =

DeleteFcn =

BusyAction = queue

HandleVisibility = on

HitTest = on

Interruptible = on

Parent = []

Selected = off

SelectionHighlight = on

Tag =

Type = root

UIContextMenu = []

UserData = []

Visible = on

**CAPÍTULO IV**

Controles gráficos

# Capítulo IV. Controles gráficos

**CAPÍTULO V**

Cuadros de diálogos

# Capítulo V. Cuadros de diálogos

Una interfaz gráfica de usuario (GUI) es un elemento gráfico que contiene uno o más controles que están disponibles para interactuar con el usuario mediante un entorno visual sencillo, el cual permite la comunicación entre el usuario y el computador. Algunos de los componentes más comunes de una GUI creada en MATLAB son menús, barras de herramientas, botones, menús desplegables, cajas de texto, entre otros.

En las interfaces gráficas creadas en MATLAB pueden aprovecharse todas las herramientas matemáticas y de ingeniería que proporciona MATLAB, permiten además la manipulación de archivos de datos, así como la interacción con otras GUIs y mostrar datos mediante tablas y gráficas de gran calidad.

Generalmente las GUIs son programadas para que respondan a la manipulación del usuario con una acción específica. Los controles gráficos que componen una GUI están relacionados con una rutina de programación, llamada *callbacks* en el entorno MATLAB, que se ejecuta cuando sucede un determinado evento, que puede ser la entrada de caracteres mediante el teclado, el clic de un botón del mouse, o situarse sobre un objeto.

## 1.1 Elemento figure

En MATLAB cada interfaz gráfica está creada sobre un objeto figure, en este elemento se añaden todos los controles gráficos que componen la GUI. La forma más simple de definir un objeto figure se ejemplifica enseguida:

hFig = figure;

Donde hFig es el handle del elemento figure.

Es muy común que al momento de definir o crear un objeto figure, se especifiquen algunas de sus propiedades con la sintaxis siguiente:

hFig = figure('Propiedad ', 'Valor ',...);

A continuación se muestra un ejemplo característico:

hFig = figure('NumberTitle','off',...

'MenuBar','None',...

'Name','Figure Ejemplo',...

'Position',[200 200 300 300]);

Las propiedades más comunes de un elemento figure se muestran en la Tabla 1.1.

|  |  |
| --- | --- |
| **PROPIEDAD** | **DESCRIPCIÓN** |
| Color | Establece el color de figure. El valor puede especificarse mediante un vector de tres elementos en formato RGB (Valores entre 0 y 1). |
| MenuBar | Oculta o muestra la barra de menús estándar de MATLAB. Por defecto es visible, si quiere ocultarse debe especificar un valor none. |
| Name | Título mostrado en la ventana de la figura. El valor especificado es una cadena de caracteres. |
| NumberTitle | Determina si la numeración de los elementos figure creada automáticamente por MATLAB será visible. El valor por defecto es on, para ocultar deberá especificarse off. |
| Position | Especifica el tamaño de la GUI y la posición relativa a la esquina inferior izquierda de la pantalla. El valor se establece mediante un vector de cuatro elementos cuya estructura es la siguiente: [Distancia de la izquierda, Distancia de la parte inferior, Ancho, Alto]; |
| Resize | Determina si puede modificarse el tamaño de la GUI utilizando el mouse. Los valores aceptados son: off y on, siendo este último el valor por defecto. |
| Toolbar | Muestra o borra el menú de herramientas del objeto figure. |
| Units | Unidad de medida que se utilizará para interpretar el vector de la propiedad position. Los valores disponibles son: centimeters, characters, inches, normalized, point y pixels. Siendo este último el valor por defecto. |
| Visible | Establece si la GUI es visible. Valores: on y off. |

Tabla 1.1 Propiedades comunes de figure

|  |
| --- |
| ||TIP DE PROGRAMACIÓN||¿Cómo centrar la GUI? |
| Una de las cuestiones más comunes cuando se inicia en el desarrollo de interfaces gráficas es, ¿cómo centrar la GUI en la pantalla?; si se está utilizando el entorno de desarrollo GUIDE basta con incluir la instrucción movegui('center') en la inicialización (OpeningFcn).  Para centrar una programmatic GUI puede utilizar la función centerfig, cuya estructura es:  centerfig(hFig)  Siendo hFig el handle del elemento figure que define la GUI. Puede utilizar gcf para hacer referencia al figure en cuestión sin referenciar el handle específico. |

## 1.2 Controles gráficos (uicontrol)

Los controles gráficos son creados mediante la función uicontrol, cuya estructura general se muestra:

hcont = uicontrol('Style', 'tipo de control',...

'Propiedad', 'Valor');

La propiedad style define el tipo de control gráfico (Ver Tabla 1.2)

|  |  |  |
| --- | --- | --- |
| **ELEMENTO** | **PROPIEDAD STYLE** | **DESCRIPCIÓN** |
| Edit Text | 'edit' | Caja o campo de texto en donde el usuario puede introducir o editar el string que contiene. Permite la inserción de todo tipo de caracteres. Generalmente se utiliza para posibilitar la introducción de datos a las rutinas de programación establecidas. |
| Static Text | 'text' | Campo de texto no editable. Suele utilizarse para mostrar resultados de un procedimiento o variables de salida que no habrán de modificarse. |
| Push Button | 'pushbutton' | Botón que puede ejecutar una acción al hacer clic sobre él. |
| Slider | 'slider' | Barra deslizable que permite seleccionar valores dentro de un rango definido, mediante la interacción con el usuario. |
| Pop up menú | 'popupmenu' | Menú desplegable que permite escoger entre diversas opciones. |
| List Box | 'listbox' | Similar a un menú desplegable en funcionamiento, con la diferencia que muestra en pantalla todos los elementos que le conforman. |
| Check Box | 'checkbox' | Puede estar seleccionado o no (activado o desactivado). Tiene utilidad en decisiones lógicas que “disparan” un evento. |
| Radio Button | 'radio' | Muy similar a un check box en funcionamiento, aunque generalmente suele utilizarse de manera agrupada en un panel o ButtonGroup. |
| Toggle Button | 'togglebutton' | Botón que simula un interruptor, puede estar activado o desactivado. |

## 1.3 Axes

Los axes son elementos que posibilitan a la GUI el mostrar gráficas generadas con los comandos usuales de MATLAB o bien imágenes que han sido previamente leídas.

La sintaxis general para crear un axes es como sigue:

hAx = axes('Propiedad','Valor',...)

Usualmente la primera propiedad que se modifica es el Parent, la cual especifica el objeto figure al cual pertenece el axes. Si no se proporciona valor alguno, MATLAB le asigna el Parent devuelto por gcf.

## 1.4 Tablas

Las tablas en una GUI MATLAB se crean utilizando la función uitable, teniendo como argumentos de entrada los modificadores de propiedades como los demás elementos. Algunas de las propiedades más comunes se enlistan en la tabla siguiente.

|  |  |
| --- | --- |
| **PROPIEDAD** | **DESCRIPCIÓN** |
| BackgroundColor | Color de fondo de las celdas. Puede especificarse mediante un vector de 3 elementos (todas las filas de un mismo color) o con una matriz de 2x3 (filas con colores intercalados). |
| ColumnEditable | Determina si una columna de datos puede ser editada por el usuario. Los valores se establecen utilizando un vector con tantos elementos como columnas, siendo true y false estos valores. |
| ColumnFormat | Establece el tipo de dato que contiene una columna (char, logical, numeric o bank). |
| ColumnName | Nombre de las columnas de la tabla. Por defecto se muestra el número de columna. |
| ColumnWidth | Ancho de las columnas. El valor preestablecido es auto, que ajusta el ancho al contenido de las columnas. Para establecer valores específicos debe insertarse un cell array con valores numéricos. |
| Data | Matriz que contiene los datos o valores de la tabla |
| Position | Vector de cuatro elementos, que determina la posición y el tamaño de la tabla respecto a la GUI a la cual pertenece. |
| RowName | Nombre de las filas de la tabla. Por defecto se muestra la numeración progresiva. |
| Units | Unidades de longitud en las cuales se mide el vector de posición y tamaño. |
| Visible | Determina si la tabla es visible. |

El número de filas y columnas de la tabla depende de la dimensión del arreglo introducido en la propiedad Data. Si no se hacen las modificaciones correspondientes, el ancho de cada columna y el formato se establecen de forma automática.

## 1.5 Paneles y Button Groups

Los paneles y button groups son elementos que sirven para contener y organizar de mejor forma los controles gráficos de una GUI. Los objetos pertenecientes a un panel definen su vector posición de manera relativa este.

La estructura general para definir un panel es:

hP = uipanel(hFig,'Propiedad','Valor',...);

Y la sintaxis para definir un button group:

hBg = uibuttongroup('Propiedad','Valor',...);

Para ambos casos, si no se proporciona un elemento figure en la propiedad Parent, se toma como valor el devuelto por gcf.

|  |
| --- |
| ||TIP DE PROGRAMACIÓN|| Selección de opciones en un Button Group |
| Un Button Group es utilizado para agrupar Radio Buttons en un panel, de tal forma que sólo puede seleccionarse una opción a la vez.  La función en la cual se programan las acciones ejecutadas se asocia al Button Group mediante la propiedad SelectionChangeFcn |

## 1.6 Menús

Por defecto un elemento figure o GUI MATLAB incluye una barra de menú y una barra de herramientas, ubicadas en la parte superior de la misma; pueden ocultarse o modificarse mediante las propiedades MenuBar y ToolBar.

Para crear un menú personalizado puede utilizarse la función uimenu, que permite una gran flexibilidad para crear menús. Un ejemplo característico se muestra a continuación:

hMenu = uimenu(hParent,'Label','Menu 1',...

'Callback',@menuFcn);

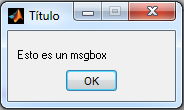
El handles hParent hace referencia al objeto gráfico al cual pertenece el menú, que puede ser el elemento figure o bien otro menú. La propiedad Label determina la cadena de texto mostrada al usuario en la interfaz gráfica y en el callback se especifica la acción que se ejecutará al seleccionar el menú.

## 1.7 Cuadros de diálogos predefinidos

Los cuadros de diálogo predefinidos son elementos gráficos que se asemejan a una pequeña ventana y cuya utilidad está ligada al envío de mensajes en pantalla al usuario.

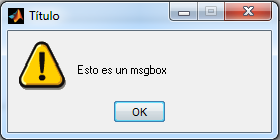
El cuadro de diálogo más común es el msgbox, la manera más simple de definirlo es utilizando dos argumentos de entrada, el string o mensaje que se muestra y el título mostrado en la parte superior:

hmb = msgbox('Esto es un msgbox','Título');



Puede utilizarse un tercer argumento de entrada en el que habrá de especificarse el tipo de ícono a mostrar en el msgbox, el cual puede ser del tipo error, help, warn o custom. Enseguida se muestra un ejemplo del tipo warn (advertencia):

hmb = msgbox('Esto es un msgbox','Título','warn');



Los msgbox de tipos warn, help y error contienen los íconos característicos que se utilizan para un mensaje de advertencia, de ayuda y de error, respectivamente. Se obtienen resultados similares si se utilizan los cuadros de diálogos predefinidos creados con las funciones warndlg, helpdlg y errordlg.

La especificación custom se utiliza para crear msgbox con íconos personalizados, los cuales se introducen mediante un cuarto argumento de entrada que debe ser una matriz con la información de la imagen, para ello puede utilizarse el comando imread y especificar la ruta y nombre de la imagen, por ejemplo:

% Ejemplo de msgbox con ícono personalizado

hmb = msgbox('Mensaje mostrado al usuario','Título',...

'custom',imread('imagen.png'));

## 1.8 Funciones get y set

*Función get*

La función get se utiliza para obtener el valor o configuración de una determinada propiedad de un objeto. Por ejemplo, suponiendo que nos interesa verificar el tamaño de una GUI determinada, para ello podrían ejecutarse las siguientes líneas de código:

A = get(hFig,'Position');

En la variable A se guarda el vector de cuatro elementos obtenido del valor de la propiedad Position (posición respecto a la pantalla y tamaño de la GUI). Como puede observarse en el ejemplo anterior el primer argumento es el handles del objeto gráfico y el otro, la propiedad que interesa obtenerse.

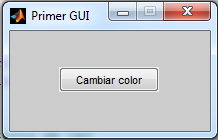
*Función set*

La función set configura o establece una propiedad determinada utilizando una estructura como la siguiente:

set(hG,'Propiedad','Valor');

Donde hG es el handle del objeto gráfico, el segundo argumento es la característica que habrá de modificarse y el último el valor establecido de la propiedad.

## 1.9 Mi primer GUI



## 2.1 Conversor de temperatura

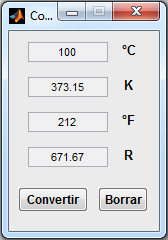
*Objetivo*

Comprender cómo insertar datos de manera interactiva utilizando campos de textos editables (Edit Text) y asociarlos a una determinada acción ejecutada mediante el callback de un Push Button.

*Descripción*

Esta interfaz gráfica permite obtener la conversión entre cuatro de las escalas de temperatura más usuales (Celsius, Kelvin, Farenheit, Rankine). Para ello debe introducir un valor en uno de los campos de texto y pulsar el botón Convertir. Si se introducen datos en más de un campo el programa establece un orden preferencial dado por la sentencia de control *if-elseif* que comienza en la línea 46.

Hasta la línea 38 se crean todos los elementos gráficos que componen la GUI, estableciendo las características y ubicación en la ventana para cada uno. La función convertir está asociada al callback del Push Button Convertir; en las primeras cuatro líneas de código de la función anterior (41-44) se lee mediante la propiedad String, el valor que se ha introducido en cada uno de los Edit Text, nótese que se hace referencia a los campos editables utilizando el handle obtenido previamente. La sentencia de control if-elseif tiene como función verificar el campo de texto en el cual se ha introducido un dato, comenzando de arriba hacia abajo, es decir que si todos los campos están rellenos el programa tomará el valor introducido en los grados centígrados.



|  |  |
| --- | --- |
| **1** | function converTemperatura |
| **2** | % Conversor de temperatura |
| **3** | % $ Autor: Jorge De Los Santos $ |
| **4** | % $ Email: delossantosmfq@gmail.com && matlabtyp@gmail.com $ |
| **5** | % $ Web: http://matlab-typ.blogspot.com $ |
| **6** |  |
| **7** |  |
| **8** | fcolor=[0.92 0.92 0.92]; |
| **9** | figure('NumberTitle','off','MenuBar','none','color',fcolor,... |
| **10** | 'Name','Conversor de Temperatura','position',[200 200 150 200],... |
| **11** | 'resize','off'); |
| **12** | centerfig(gcf); |
| **13** |  |
| **14** | % Grados Celsius |
| **15** | uicontrol('Style','text','String','°C','Position',[110 170 20 20],... |
| **16** | 'BackgroundColor',fcolor,'FontSize',10,'FontWeight','b'); |
| **17** | gC=uicontrol('Style','edit','String','','Position',[20 170 80 20]); |
| **18** |  |
| **19** | % Grados Kelvin |
| **20** | uicontrol('Style','text','String','K','Position',[110 135 20 20],... |
| **21** | 'BackgroundColor',fcolor,'FontSize',10,'FontWeight','b'); |
| **22** | gK=uicontrol('Style','edit','String','','Position',[20 135 80 20]); |
| **23** |  |
| **24** | % Grados Fahrenheit |
| **25** | uicontrol('Style','text','String','°F','Position',[110 100 20 20],... |
| **26** | 'BackgroundColor',fcolor,'FontSize',10,'FontWeight','b'); |
| **27** | gF=uicontrol('Style','edit','String','','Position',[20 100 80 20]); |
| **28** |  |
| **29** | % Grados Rankine |
| **30** | uicontrol('Style','text','String','R','Position',[110 65 20 20],... |
| **31** | 'BackgroundColor',fcolor,'FontSize',10,'FontWeight','b'); |
| **32** | gR=uicontrol('Style','edit','String','','Position',[20 65 80 20]); |
| **33** |  |
| **34** | uicontrol('Style','Pushbutton','String','Convertir','FontSize',9,... |
| **35** | 'FontWeight','b','Position',[10 20 70 25],'callback',@convertir); |
| **36** |  |
| **37** | uicontrol('Style','Pushbutton','String','Borrar','FontSize',9,... |
| **38** | 'FontWeight','b','Position',[90 20 50 25],'callback',@borrar); |
| **39** |  |
| **40** | function convertir(src,event) |
| **41** | C=get(gC,'String'); |
| **42** | K=get(gK,'String'); |
| **43** | F=get(gF,'String'); |
| **44** | R=get(gR,'String'); |
| **45** |  |
| **46** | if ~strcmp(C,'') |
| **47** | C=str2double(C); |
| **48** | K=C+273.15; |
| **49** | F=(9/5)\*C+32; |
| **50** | R=(C+273.15)\*(9/5); |
| **51** | elseif ~strcmp(K,'') |
| **52** | K=str2double(K); |
| **53** | C=K-273.15; |
| **54** | F=(9/5)\*K-459.67; |
| **55** | R=(9/5)\*K; |
| **56** | elseif ~strcmp(F,'') |
| **57** | F=str2double(F); |
| **58** | C=(F-32)\*(5/9); |
| **59** | K=(F+459.67)\*(5/9); |
| **60** | R=F+459.67; |
| **61** | elseif ~strcmp(R,'') |
| **62** | R=str2double(R); |
| **63** | C=(R-491.67)\*(5/9); |
| **64** | K=(5/9)\*R; |
| **65** | F=R-459.67; |
| **66** | else |
| **67** | warndlg('Inserte un valor en al menos un campo','Error'); |
| **68** | end |
| **69** |  |
| **70** | set(gC,'String',num2str(C)); |
| **71** | set(gK,'String',num2str(K)); |
| **72** | set(gF,'String',num2str(F)); |
| **73** | set(gR,'String',num2str(R)); |
| **74** | end |
| **75** |  |
| **76** | function borrar(src,event) |
| **77** | set(findobj('style','edit'),'str',''); |
| **78** | end |
| **79** | end |

## 2.2 Modifica tamaño y color de fuente

|  |  |
| --- | --- |
| **1** | function modifLetra |
| **2** | % Modifica el tamaño y color de letra de un Static Text |
| **3** | % utilizando menús desplegables (pop-up menu) |
| **4** |  |
| **5** | figure('MenuBar','none','NumberTitle','off','resize','off',... |
| **6** | 'Position',[100 100 200 150],'Name','Color/Tamaño de letra'); |
| **7** | centerfig(gcf); |
| **8** |  |
| **9** | txt=uicontrol('style','text','string','MATLAB GUI',... |
| **10** | 'position',[30 100 140 40]); |
| **11** |  |
| **12** | uicontrol('style','popupmenu','String',{'Seleccione un tamaño','8','9',... |
| **13** | '10','11','12','13','14','15','16'},'position',[20 60 160 20],... |
| **14** | 'callback',@dimFcn); |
| **15** |  |
| **16** | uicontrol('style','popupmenu','String',{'Seleccione un color','Rojo',... |
| **17** | 'Verde','Azul','Negro','Magenta','Amarillo'},... |
| **18** | 'position',[20 20 160 20],'callback',@colorFcn); |
| **19** |  |
| **20** | function dimFcn(src,event) |
| **21** | k=get(src,'Value'); |
| **22** | A=get(src,'String'); |
| **23** | n=A{k}; |
| **24** | if k~=1 |
| **25** | set(txt,'FontSize',str2double(n)); |
| **26** | end |
| **27** | end |
| **28** |  |
| **29** | function colorFcn(src,event) |
| **30** | k=get(src,'Value'); |
| **31** | A=get(src,'String'); |
| **32** | switch A{k} |
| **33** | case 'Rojo' |
| **34** | set(txt,'ForegroundColor','r'); |
| **35** | case 'Verde' |
| **36** | set(txt,'ForegroundColor','g'); |
| **37** | case 'Azul' |
| **38** | set(txt,'ForegroundColor','b'); |
| **39** | case 'Negro' |
| **40** | set(txt,'ForegroundColor','k'); |
| **41** | case 'Magenta' |
| **42** | set(txt,'ForegroundColor','m'); |
| **43** | case 'Amarillo' |
| **44** | set(txt,'ForegroundColor','y'); |
| **45** | otherwise |
| **46** | errordlg('Color no válido'); |
| **47** | end |
| **48** | end |
| **49** | end |

**Ejercicios**

1. Desarrolle una GUI que contenga tres pushbuttons, y prográmelos de manera que al activarlos muestren cuadros de diálogos predefinidos (errordlg, warndlg y helpdlg).

2. Agregue funcionalidades a la calculadora desarrollada en los ejemplos, de tal modo que se asemeje a una calculadora científica estándar.

3. Desarrolle una agenda personal en la cual pueda agregar nuevos contactos y guardarlos en un archivo de datos (utilizando cell arrays o estructuras), y que permita además visualizarlos y editarlos posteriormente.

4