CAPÍTULO 5: INTERFAZ DE USUARIO (GUI)

5.1.- ¿Por qué usar Matlab?

Existen varios lenguajes de programación en los cuales es posible crear una interfaz para el control de diferentes aparatos electrónicos, como pueden ser: lenguaje *C++*, *Visual Basic*, *LabWindows*, *Matlab*, *LabView*, por mencionar algunos. Sin embargo, *LabWindows* ha sido el preferido de científicos, ingenieros, y técnicos para crear diversas soluciones, para una variedad de industrias alrededor del mundo.

Esto se debe a que es una poderosa herramienta para escribir programas de adquisición de datos, a su manera amistosa de solucionar problemas de automatización y medición, además de que está compuesto de un lenguaje de programación gráfico basado en C, que es uno de los lenguajes de programación más conocidos y poderosos. Otro factor que influye es el hecho de que es un software creado por *National Instruments*, y por lo tanto contiene un conjunto completo de las librerías integradas de instrumentación, con lo que facilita la programación de aplicaciones para el control de instrumentos y adquisición de datos. Sin embargo, a pesar de todas estas ventajas que representa *LabWindows* y en busca de nuevas alternativas que pudieran ser más eficientes para este tipo de aplicaciones se optó por trabajar con Matlab debido a que, no es sólo un paquete de computación y graficación, sino una herramienta versátil y flexible, que permite a usuarios que cuentan con conocimientos de programación básicos producir gráficas e interfaces gráficas de usuario (*GUIs*) sofisticadas, y para programadores con más experiencia tiene la versatilidad de poder interactuar con otros lenguajes como C.

Matlab es uno de los lenguajes de programación más utilizados en el ámbito de la investigación debido a su gran capacidad para el procesamiento de cálculos matemáticos. Además de que cuenta con *ToolBoxes* (Cajas de herramientas) que contienen controles que facilitan aún más la programación de aplicaciones específicas en diferentes áreas del conocimiento como pueden ser: comunicaciones, control, procesamiento digital de señales, etc.

Un programa en Matlab se puede hacer en tres formas:

- Desde la Línea de Comandos
- Desde un archivo
- Desde una Interfaz Gráfica de Usuario (*GUI*)

Por tanto, se eligió Matlab v7.5.0 para el desarrollo de la herramienta software debido a la gran facilidad que éste tiene en cuanto al diseño de interfaz gráfica (*Guide*), para el procesamiento gráfico, y su gran versatilidad en cuanto a operaciones matriciales. Este también posee una gran facilidad de programación.

5.2.- Presentación del GUIDE

El *GUIDE* (*Graphical User Interface Development Environment*: Entorno de desarrollo de Interfaz Gráfica de Usuario) es una serie de herramientas que proporciona Matlab para la creación de *GUIs* (*Graphical User Interface*: Interfaz Gráfica de Usuario).

Un *GUI* es un conjunto de uno o varios paneles que están provistos de algunos de los siguientes elementos:

• Controles (botones, menús desplegables, etc.), que permitirán al usuario interactuar con el *GUI* y establecer el flujo de ejecución.

• Menús.

• Ejes de coordenadas, que permiten dibujar gráficos e imágenes.

En esta herramienta vamos a usar el *GUIDE* para:

• Diseñar el aspecto del *GUI*.

• Programar el *GUI*: El *GUIDE* genera de forma automática un fichero .m que controla el funcionamiento del *GUI*. Este fichero .m inicializa el *GUI* y contiene un marco para todos los callbacks del *GUI* (las órdenes que se ejecutan cuando el usuario interactúa con un elemento del *GUI*). Usando el editor de Matlab podremos añadir código a los callbacks para realizar las funciones que queramos asignarles.

El beneficio que proporciona el uso de *GUIs* es evidente, ya que permiten al usuario ejecutar cómodamente código desarrollado en Matlab sin necesidad de cumplir la incómoda sintaxis funcional necesaria cuando se trabaja desde la línea de órdenes.

A diferencia de la ejecución de funciones o scripts de Matlab, la ejecución de *GUIs* no predetermina el flujo de ejecución del código. Es el usuario, a través de su interacción con el *GUI*, el que determina el orden en que se ejecutan las diferentes órdenes y funciones desarrolladas. Otra diferencia importante es que la ejecución no termina cuando finaliza la ejecución del script o función, sino que el *GUI* permanece abierto, permitiendo al usuario invocar la ejecución de ese u otro código desarrollado.

El desarrollo de *GUIs* se realiza en dos etapas:

• Diseño de los componentes (controles, menús y ejes) que formarán el *GUI*.

• Codificación de la respuesta de cada uno de los componentes ante la interacción del usuario.

Más adelante se verá cómo invocar y comenzar a ejecutar el *GUIDE*, pero ahora nos adelantaremos y echaremos un vistazo a la ventana principal de *GUIDE*, lo que nos permitirá conocer las herramientas que componen el entorno:

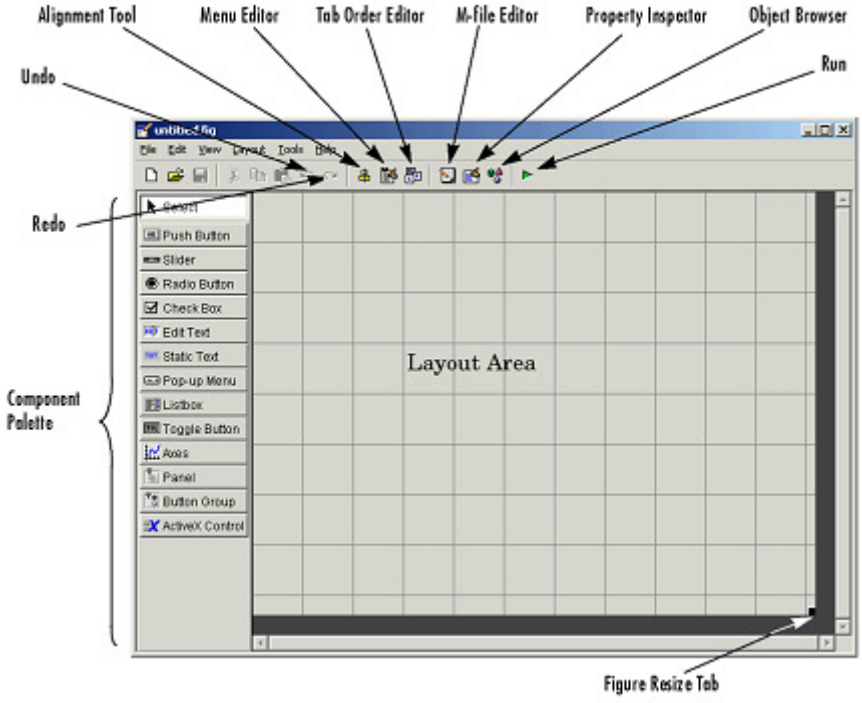

Figura 5.1: Ventana principal de *GUIDE* con sus componentes

• **Barra de menús**, en la que se pueden encontrar las funciones de edición de *GUIs*.

• **Paleta de componentes** (*Component Palette*), que contiene los componentes que se pueden añadir a un *GUI*:

- *Push Button*
- *Toggle Button*
- *Check Box*
- *Radio Button*
- *Edit Text*
- *Static Text*
- *Slider*
- *List Box*

- *Pop-Up Menu*
- *Axes*
- *Button Group*
- *ActiveX Component*

La siguiente tabla muestra una descripción de los componentes:

Control	*Valor de estilo*	*Descripción*
Check box	'checkbox'	Indica el estado de una opción o atributo
Editable Text	'edit'	Caja para editar texto
Pop-up menu	'popupmenu'	Provee una lista de opciones
List Box	'listbox'	Muestra una lista deslizable
Push Button	'pushbutton'	Invoca un evento inmediatamente
Radio Button	'radio'	Indica una opción que puede ser seleccionada
Toggle Button	'togglebutton'	Solo dos estados, "on" o "off"
Slider	'slider'	Usado para representar un rango de valores
Static Text	'text'	Muestra un string de texto en una caja
Panel button		Agrupa botones como un grupo
Button Group		Permite exclusividad de selección con los radio button

Tabla 5.1: Descripción de los componentes del *GUIDE*

• **Barra de herramientas**, en la que se pueden encontrar botones para activar las herramientas más importantes del *GUIDE*:

- Herramienta de alineación (*Alignment Tool*), que permitirá establecer opciones de alineación entre los diferentes componentes del *GUI*.
- Editor de menús (*Menu Editor*), que permitirá definir las entradas que compondrán el menú del *GUI*.
- Editor de orden de tabulación (*Tab Order Editor*), que permite establecer el orden en que se accede a los componentes al pulsar el tabulador (en lugar de usar el ratón).
- Editor de ficheros .m (*M-file Editor*), con el que se podrá añadir código a las funciones *callback* correspondientes a cada componente.
- Inspector de propiedades (*Property Inspector*), que ofrece la posibilidad de examinar y cambiar las propiedades de los componentes.
- Navegador de objetos (*Object Brow*ser), especialmente útil cuando se tiene un número elevado de componentes, permite examinar la lista de objetos del *GUI*, organizándolos según su jerarquía.
- Botón de ejecución (*Run Button*).

5.3.- Organización de los objetos gráficos en Matlab

Matlab ofrece una serie de funciones de dibujo y visualización de datos. Algunas de estas funciones son *imread*, *imshow*, *imagesc* o *plot*, entre otras. Cuando se llama a una función gráfica, Matlab crea el gráfico requerido usando objetos como ventanas, ejes de coordenadas, líneas, texto, etc.

Se puede trabajar con gráficos en Matlab a tres niveles diferentes:

• Realizando llamadas a funciones de dibujo y visualización. Matlab muestra todas las gráficas en un tipo de ventanas especiales conocidas como figures, en las que se sitúan unos ejes de coordenadas. Estos ejes son los que proporcionan el sistema de coordenadas necesario para realizar la visualización de los datos.

• Trabajando a "bajo nivel", haciendo las llamadas necesarias a funciones de Matlab para ir creando los objetos gráficos que sean necesarios para hacer la visualización. A este nivel se realizarán múltiples llamadas a la función set y otras similares para establecer las propiedades de los distintos objetos gráficos que se vayan creando o para modificarlas durante la ejecución del programa. Este modo de trabajar es muy similar al uso tradicional de una biblioteca gráfica que se puede hacer en un lenguaje de programación convencional.

• Empleando el *GUIDE*, el entorno de desarrollo de *GUIs*, que nos permitirá definir todos los componentes gráficos que deseemos, establecer sus propiedades e incorporar código de respuesta a cada una de las acciones del usuario a través de una herramienta gráfica de cómodo manejo. Tiene la ventaja de que en cualquier momento podremos elegir si deseamos trabajar a alto o bajo nivel, pudiendo acceder al código asociado a través del editor de Matlab.

Como se puede ver, en cualquiera de estos tres casos se está haciendo uso del sistema de objetos gráficos de Matlab.

Los objetos gráficos son los elementos básicos empleados por Matlab para visualizar datos, y están organizados en una jerarquía como la de la siguiente figura, en la que se muestran los tipos de objetos gráficos empleados más frecuentemente:

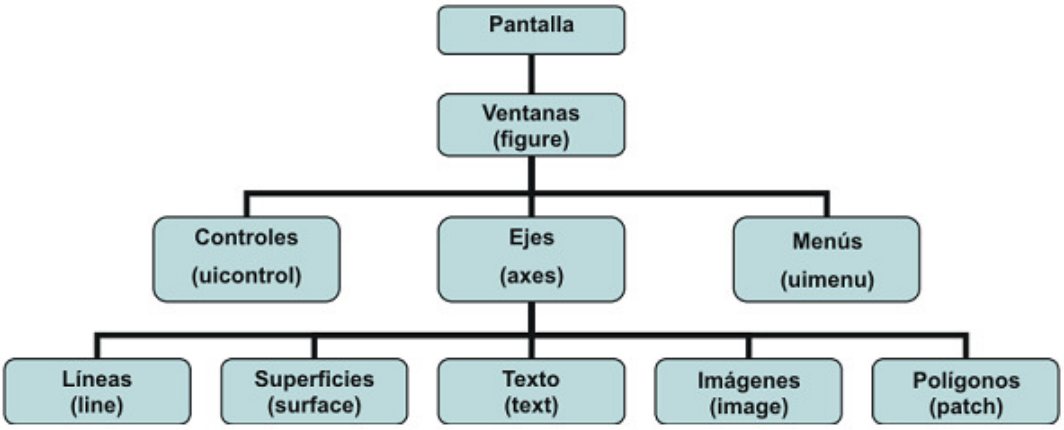

Figura 5.2: Objetos gráficos empleados en Matlab

Como se ve en la figura, el objeto más general es la pantalla. Es la raíz de la jerarquía. Una pantalla puede contener una o varias ventanas (*figure*). A su vez, cada una de las ventanas podrá contener controles (*uicontrol*), como son los botones, menús desplegables, etc., menús (*uimenu*) y uno o más ejes de coordenadas (*axes*) en los que se podrán representar objetos de nivel inferior. Los ejes pueden incluir cinco tipos de elementos gráficos: líneas (*line*), polígonos (*patch*), texto (*text*), superficies (*surface*) e imágenes de mapa de bit (*image*).

Al establecerse relaciones padre-hijo entre los objetos gráficos se facilita su gestión, ya que, por ejemplo, cuando se borra un objeto, se borrarán automáticamente todos los descendientes de éste.

Cada objeto está asociado a un identificador (*handle*) único desde el momento de su creación. A través de este identificador podremos modificar las características (llamadas propiedades del objeto) de un objeto gráfico. Naturalmente, también se puede establecer las propiedades de un objeto en el momento de su creación (cambiarlas con respecto a los valores por omisión). El identificador del objeto raíz, la pantalla, es siempre cero. El identificador de las distintas ventanas (*figure*) es un entero que aparecerá en la barra de título de la ventana. Los identificadores de los demás objetos gráficos son números reales.

Cualquier identificador se puede obtener como valor de retorno de una función y almacenarse en una variable.

Puede haber varias ventanas abiertas, pero sólo una de ellas es la ventana activa en cada momento. De la misma forma, una ventana puede contener varios ejes de coordenadas, pero sólo unos son los ejes activos. Por último, el objeto activo es el último objeto creado o sobre el que se ha hecho clic con el ratón. Se puede obtener los identificadores de la ventana, los ejes y el objeto activos con las órdenes:

- *gcf*: devuelve un entero, el identificador de la ventana activa.

• *gca*: devuelve el identificador de los ejes activos.

• *gco*: devuelve el identificador del objeto activo.

La principal utilidad que tiene conocer los identificadores de los objetos gráficos es que a través de ellos se podrán modificar las propiedades de los objetos o incluso borrarlos:

• *set* (id): muestra en pantalla todas las propiedades del objeto al que corresponde el identificador id.

• *get* (id): produce un listado de las propiedades y de sus valores.

• *set* (id, 'propiedad', 'valor'): establece un nuevo valor para la propiedad del objeto con identificador id. Se pueden establecer varias propiedades en la misma llamada a get incluyendo una lista de parejas 'propiedad', 'valor' en la llamada.

• *get* (id, 'propiedad'): obtiene el valor de la propiedad especificada.

• *delete* (id): borra el objeto cuyo identificador es id y todos sus hijos.

5.4.- Primeros pasos con el GUIDE

En este apartado se va introducir muy básicamente la realización de una interfaz gráfica con el *GUIDE*.

Al llamar al *GUIDE* con la orden *guide* en la línea de comando de Matlab aparecerá el siguiente cuadro de diálogo *GUIDE Quick Start*.

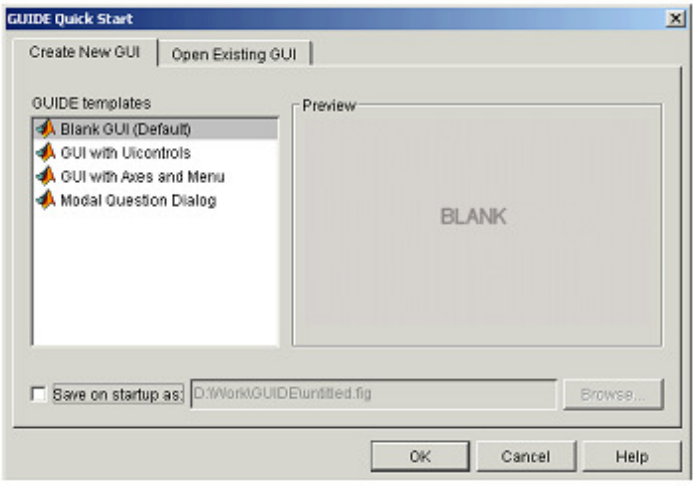

Figura 5.3: Cuadro de diálogo de inicio del *GUIDE*

donde se presentan las siguientes opciones:

a) *Blank GUI (Default)*

La opción de interfaz gráfica de usuario en blanco (viene predeterminada), presenta un formulario nuevo, en el cual se puede diseñar un programa.

b) *GUI with Uicontrols*

Esta opción presenta un ejemplo en el cual se calcula la masa, dada la densidad y el volumen, en alguno de los dos sistemas de unidades. Podemos ejecutar este ejemplo y obtener resultados.

c) *GUI with Axes and Menu*

Esta opción es otro ejemplo el cual contiene el menú *File* con las opciones *Open*, *Print* y *Close*. En el formulario tiene un *Popup menu*, un *Push button* y un objeto *Axes*. Se podrá ejecutar el programa eligiendo alguna de las seis opciones que se encuentran en el menú despegable y haciendo clic en el botón de comando.

d) *Modal Question Dialog*

Con esta opción se muestra en la pantalla un cuadro de diálogo común, el cual consta de una pequeña imagen, una etiqueta y dos botones *Yes* y *No*, dependiendo del botón que se presione, el *GUI* retorna el texto seleccionado (la cadena de caracteres *'Yes'* o *'No'*).

Si, por ejemplo, se elige la primera opción, *Blank GUI*, se muestra el Editor de diseño (*Layout Editor*), que es el panel de control de todas las herramientas del *GUIDE*.

Se podrá diseñar un *GUI* arrastrando elementos (botones, menús desplegables, ejes, etc.) desde la paleta de componentes, que se encuentra en el lado izquierdo del editor. Por ejemplo, si arrastramos un botón al área de diseño tendremos algo como la figura 5.4:

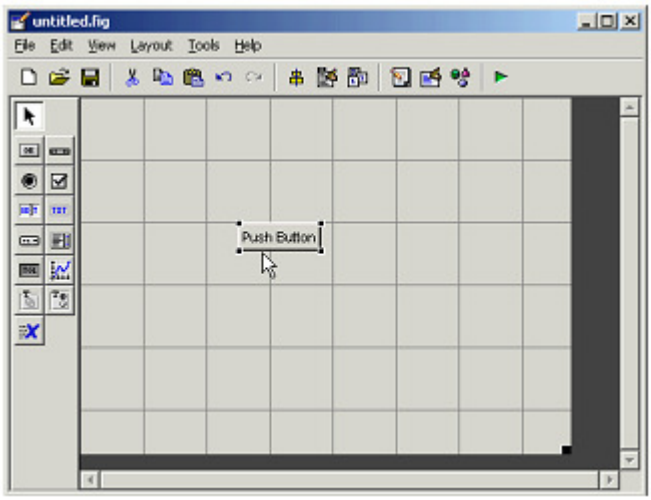

Figura 5.4: Añadir un *Push Button* a la interfaz gráfica

La barra de herramientas del *GUIDE* cuenta con los siguientes botones (figura 5.5):

	Alinear objetos.
	Editor de menú.
	Editor de orden de etiqueta.
	Editor del M-file.
	Propiedades de objetos.
	Navegador de objetos.
	Grabar y ejecutar (ctrl. + T).

Figura 5.5: Herramientas de la interfaz gráfica

También se puede usar el editor para establecer las propiedades de los componentes del *GUI*.

5.5.- Propiedades de los componentes

Cada uno de los elementos de *GUI*, tiene un conjunto de opciones a las que se puede acceder con clic derecho.

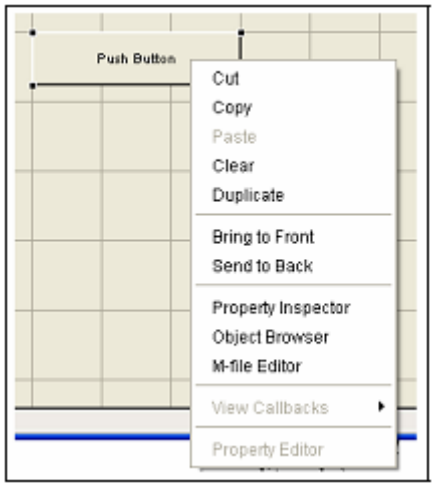

Figura 5.6: Opciones de los componentes del *GUIDE*

La opción *Property Inspector* permite personalizar cada elemento.

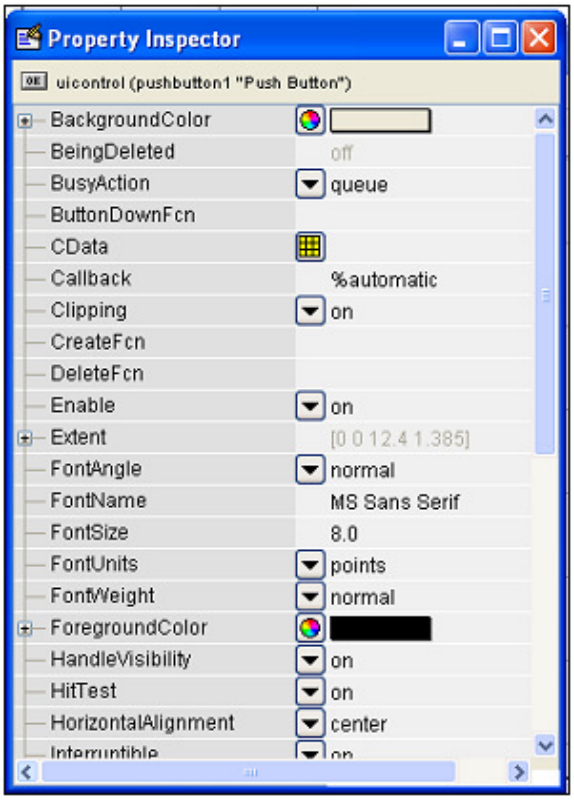

Figura 5.7: Entorno *Property Inspector*. Permite ver y editar las propiedades de un objeto

Al hacer clic derecho en el elemento ubicado en el área de diseño, una de las opciones más importantes es *'View Callbacks'*, la cual, al ejecutarla, abre el archivo .m asociado al diseño que se está empleando y se posiciona en la parte del programa que corresponde a la subrutina que se ejecutará cuando se realice una determinada acción sobre el elemento que estamos editando.

Por ejemplo, al ejecutar *View Callbacks>>Callbacks* en el *Push Button*, se mostrará el archivo .m ubicándose en esta parte del programa:

```
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)
```

5.6.- Funcionamiento de una aplicación GUI

Una aplicación *GUIDE* consta de dos archivos: .m y .fig. El archivo .m es el que contiene el código con las correspondencias de los botones de control de la interfaz y el archivo .fig contiene los elementos gráficos.

Estos dos ficheros se corresponden con las dos etapas de creación de un *GUI*: la de diseño y la de programación.

Cada vez que se adicione un nuevo elemento en la interfaz gráfica, se genera automáticamente código en el archivo .m.

El fichero .m del *GUI* se encarga de las siguientes tareas:

- Inicializa el *GUI*.
- Contiene código para realizar las tareas necesarias antes de que aparezca el *GUI* en pantalla, como la creación de datos o gráficos.
- Contiene las funciones *callback* que se ejecutan cada vez que el usuario hace clic en un componente.

Inicialmente, cada *callback* contiene sólo una línea de definición de la función. Podemos añadir código que haga que el componente funcione como deseemos.

Para abrir el fichero .m del *GUI* podemos hacer clic en el icono de la barra de herramientas del Editor de diseño. El fichero 'proyecto.m' corresponde a *GUI* creado a

partir de la plantilla *GUI 'with Axes and Menu'*. Si se abre este fichero en el editor de Matlab, se puede acceder directamente a la función *callback* de cualquier elemento del *GUI* a través del botón  , que mostrará un menú como el de la siguiente figura:

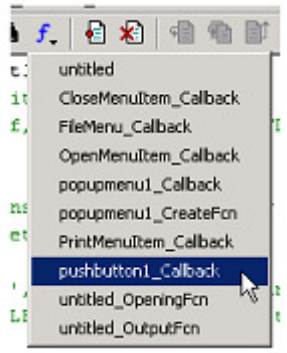

Figura 5.8: Menú para accede a la función *callback* de cualquier elemento

Para ejecutar una Interfaz Gráfica, si se ha etiquetado con el nombre proyecto.fig, simplemente se escribirá en la ventana de comandos:

```
>>  proyecto.
```

O haciendo clic derecho en el m-file y seleccionando la opción *RUN*.

5.6.1.- Manejo de datos entre los elementos de la aplicación y el archivo .m

Todos los valores de las propiedades de los elementos (color, valor, posición,…) y los valores de las variables transitorias del programa se almacenan en una estructura, los cuales son accedidos mediante un único y mismo identificador para todos éstos. Tomando el programa listado anteriormente, el identificador se asigna en:

```
>> handles.output = hObject;
```

handles, es el identificador de los datos de la aplicación. Esta definición de identificador es salvada con la siguiente instrucción:

```
>> guidata(hObject, handles);
```

guidata, es la sentencia para salvar los datos de la aplicación.

Aviso: *guidata* es la función que guarda las variables y propiedades de los elementos en la estructura de datos de la aplicación, por lo tanto, como regla general, en cada subrutina se debe escribir en la última línea lo siguiente:

>> guidata(hObject,handles);

Esta sentencia garantiza que cualquier cambio o asignación de propiedades o variables queda almacenado.

Por ejemplo, si dentro de una subrutina una operación dio como resultado una variable "materiales" para poder utilizarla desde el programa principal u otra subrutina se debe salvar de la siguiente manera:

```
handles.materiales=materiales;
guidata(hObject,handles);
```

La primera línea crea la variable "materiales" a la estructura de datos de la aplicación apuntada por *handles* y la segunda graba el valor.

5.6.2.- Sentencias GET y SET

La asignación u obtención de valores de los componentes se realiza mediante las sentencias *get* y *set*. Por ejemplo, si se quiere que la variable utpl tenga el valor de un *Slider* etiquetado con el nombre slider, se escribe:

>> utpl= get (handles.slider1,'Value');

Notar que siempre se obtienen los datos a través de los identificadores *handles*.

Para asignar el valor a la variable utpl al *Static Text* etiquetado como *text1* se escribe:

>> set (handles.text1,'String',utpl); %Escribe el valor del *Slider* en *static-text*