CAPÍTULO 11: APÉNDICE

ANEXO I: Almacenamiento en Archivo

Matlab permite almacenar en el disco las variables del espacio de trabajo. De este modo es posible parar una sesión de trabajo y continuar en otro momento sin tener que repetir los cálculos. La orden más común para salvar los datos es **save,** que puede usarse de varias maneras. En la siguiente tabla mostramos un resumen:

Orden	Operación realizada
save	Crea el archivo de nombre matlab.mat en la carpeta actual. Dicho archivo contiene todas las variables que existen en ese momento en el entorno de Matlab.
save *nombre_archivo*	Crea el archivo de nombre *nombre_archivo.mat* en la carpeta actual. Dicho archivo contiene todas las variables que existen en ese momento en el entorno de Matlab.
save *nombre_archivo x, y, z*	Crea el archivo de nombre *nombre_archivo.mat* en la carpeta actual. Dicho archivo contiene solamente las variables *x*, *y* y *z*.

Tabla A1.1: Tabla resumen del comando save

Para recuperar las variables almacenadas en un fichero previamente creado, emplearemos principalmente la orden **load**. La siguiente tabla ilustra tres operaciones típicas de recuperación de datos

Orden	Operación realizada
load	Lee todas las variables del archivo Matlab.mat de la carpeta actual. Si alguna de las variables del disco tiene nombre coincidente con otra que previamente existe en Matlab se producirá la destrucción de la variable existente para dejar su sitio a la variable del disco
load *nombre_archivo*	Igual que en el caso anterior pero leyendo del archivo nombre_archivo.mat de la carpeta actual
load *nombre_archivo x, y ,z*	Igual que en el caso anterior pero leyendo únicamente las variables *x, y, z*

Tabla A1.2: Tabla resumen del comando load

ANEXO II: Barra de Progreso

Código Matlab de la función '*progressbar*' empleado para crear la barra de progreso que indica el tiempo y porcentaje restante de la simulación que realizamos.

```
function [stopBar] =  progressbar(fractiondone, position)

if(~exist('fractiondone'))
    return
end
% Description:
% progressbar(fractiondone,position) provides an indication of the
% progress of some task using graphics and text. Calling progressbar
% repeatedly will update
% the figure and automatically estimate the amount of time remaining.
% This implementation of progressbar is intended to be extremely
% simple to use while providing a high quality user experience.
%
% Features:
% - Can add progressbar to existing m-files with a single line of
% code.
% - The figure closes automatically when the task is complete.
% - Only one progressbar can exist so old figures don't clutter the
% desktop.
% - Remaining time estimate is accurate even if the figure gets
% closed.
% - Minimal execution time. Won't slow down code.
% - Random color and position options.
%
% Usage:
% fractiondone specifies what fraction (0.0 - 1.0) of the task is
% complete.
% Typically, the figure will be updated according to that value.
% However, if
% fractiondone == 0.0, a new figure is created (an existing figure
% would be closed first). If fractiondone == 1.0, the progressbar
% figure will close.
% position determines where the progressbar figure appears on screen.
% This argument only has an effect when a progress bar is first
% created or is reset by calling with fractiondone = 0. The progress
% bar's position can be specified as follows:
%       [x, y] - Position of lower left corner in normalized units
% (0.0 - 1.0)
%           0   - Centered (Default)
%           1   - Upper right
%           2   - Upper left
%           3   - Lower left
%           4   - Lower right
%           5   - Random [x, y] position
% The color of the progressbar is choosen randomly when it is created
% or reset. Clicking inside the figure will cause a random color
% change.
% For best results, call progressbar(0) (or just progressbar) before
% starting a task. This sets the proper starting time to calculate
% time remaining.
```

```
%
% Example Function Calls:
%   progressbar(fractiondone,position)
%   progressbar                % Initialize/reset
%   progressbar(0)             % Initialize/reset
%   progressbar(0,4)           % Initialize/reset and specify position
%   progressbar(0,[0.2 0.7])   % Initialize/reset and specify position
%   progressbar(0.5)           % Update
%   progressbar(1)             % Close
%
% Demo:
%   n = 1000;
%   progressbar % Create figure and set starting time
%   for i = 1:n
%       pause(0.01) % Do something important
%       progressbar(i/n) % Update figure
%   end

stopBar = 0;
persistent progfig progpatch starttime lastupdate firstIteration

% Set defaults for variables not passed in
if nargin < 1
    fractiondone = 0;
end
if nargin < 2
    position = 0;
end

try
    % Access progfig to see if it exists ('try' will fail if it
% doesn't)
    dummy = get(progfig,'UserData');
    % If progress bar needs to be reset, close figure and set handle
% to empty
    if fractiondone == 0
        delete(progfig) % Close progress bar
        progfig = []; % Set to empty so a new progress bar is created
    end
catch
    progfig = []; % Set to empty so a new progress bar is created
end


percentdone = floor(100*fractiondone);

% Create new progress bar if needed

if (isempty(progfig) && (isempty(firstIteration)))
    firstIteration = 1;
    % Calculate position of progress bar in normalized units
    scrsz = [0 0 1 1];
    width = scrsz(3)/4;
    height = scrsz(4)/50;
    if (length(position) == 1)
       hpad = scrsz(3)/64; % Padding from left or right edge of screen
       vpad = scrsz(4)/24; % Padding from top or bottom edge of screen
       left   = scrsz(3)/2 - width/2;  % Default
       bottom = scrsz(4)/2 - height/2; % Default
        switch position
```

```
            case 0 % Center
            % Do nothing (default)
            case 1 % Top-right
                left   = scrsz(3) - width  - hpad;
                bottom = scrsz(4) - height - vpad;
            case 2 % Top-left
                left   = hpad;
                bottom = scrsz(4) - height - vpad;
            case 3 % Bottom-left
                left   = hpad;
                bottom = vpad;
            case 4 % Bottom-right
                left   = scrsz(3) - width  - hpad;
                bottom = vpad;
            case 5 % Random
                left   = rand * (scrsz(3)-width);
                bottom = rand * (scrsz(4)-height);
            otherwise
                warning('position must be (0-5). Reset to 0.')
        end
        position = [left bottom];
    elseif length(position) == 2
        % Error checking on position
        if (position(1) < 0) | (scrsz(3)-width < position(1))
            position(1) = max(min(position(1),scrsz(3)-width),0);
            warning('Horizontal position adjusted to fit on screen.')
        end
        if (position(2) < 0) | (scrsz(4)-height < position(2))
            position(2) = max(min(position(2),scrsz(4)-height),0);
            warning('Vertical position adjusted to fit on screen.')
        end
    else
        error('position is not formatted correctly')
    end

    % Initialize progress bar
    progfig = figure(...
        'Units',            'normalized',...
        'Position',         [position width height+0.1],...
        'NumberTitle',      'off',...
        'Resize',           'off',...
        'MenuBar',          'none',...
        'BackingStore',     'off' );
    progaxes = axes(...
        'Position',         [0.02 0.15 0.96 0.25],...
        'XLim',             [0 1],...
        'YLim',             [0 1],...
        'Box',              'on',...
        'ytick',            [],...
        'xtick',            [] );
    protext = text(0.03,2.25,'El cálculo puede tardar varios
minutos.','Fontname','Calibri','Fontsize',10);
    progpatch = patch(...
        'XData',            [0 0 0 0],...
        'YData',            [0 0 1 1],...
        'EraseMode',        'none' );


% enable this code if you want the bar to change colors when the
% user clicks on the progress bar
%     set(progfig,  'ButtonDownFcn',{@changecolor,progpatch});
```

```
%     set(progaxes, 'ButtonDownFcn',{@changecolor,progpatch});
%     set(progpatch,'ButtonDownFcn',{@changecolor,progpatch});
%     changecolor(0,0,progpatch)

    set(progpatch,'FaceColor',[.1 1 .1]);

    % Set time of last update to ensure a redraw
    lastupdate = clock - 1;

    % Task starting time reference
    if isempty(starttime) | (fractiondone == 0)
        starttime = clock;
    end

    set(progfig,'CloseRequestFcn',@closeBar);

end

% if the user closes the progress bar during the data processing
% then this will erase all the variables are return 1 to the output
if (isempty(progfig) && ~(fractiondone==0))
    delete(progfig) % Close progress bar

    % Clear persistent vars
    clear progfig progpatch starttime lastupdate firstIteration
    stopBar = 1;
    return

end



% Enforce a minimum time interval between updates
% but allows for the case when the bar reaches 100% so that the user
% can see it
if (etime(clock,lastupdate) < 0.01 && ~(percentdone == 100))
    return
end

% Update progress patch
set(progpatch,'XData',[0 fractiondone fractiondone 0])

% Update progress figure title bar
if (fractiondone == 0)
    titlebarstr = ' 0%';
else
    runtime = etime(clock,starttime);
    timeleft = runtime/fractiondone - runtime;
    timeleftstr = sec2timestr(timeleft);
    titlebarstr = sprintf('%2d%%    %s
remaining',percentdone,timeleftstr);
end
set(progfig,'Name',titlebarstr)

% Force redraw to show changes
drawnow

% If task completed, close figure and clear vars, then exit
```

```
if percentdone == 100 % Task completed
     delete(progfig) % Close progress bar

    %change the close request function back to normal
    %set(progfig,'CloseRequestFcn','closereq');
    % Clear persistent vars
      clear progfig progpatch starttime lastupdate firstIteration
    return
end



% Record time of this update
lastupdate = clock;

%%
% ------------------------------------------------------------------------
function changecolor(h,e,progpatch)
Change the color of the progress bar patch

colorlim = 2.8; % Must be <= 3.0 - This keeps the color from being too
light
thiscolor = rand(1,3);
while sum(thiscolor) > colorlim
    thiscolor = rand(1,3);
end
set(progpatch,'FaceColor',thiscolor);

%%
% ------------------------------------------------------------------------
function timestr = sec2timestr(sec)
% Convert a time measurement from seconds into a human readable
string.

% Convert seconds to other units
d = floor(sec/86400); % Days
sec = sec - d*86400;
h = floor(sec/3600); % Hours
sec = sec - h*3600;
m = floor(sec/60); % Minutes
sec = sec - m*60;
s = floor(sec); % Seconds

% Create time string
if d > 0
    if d > 9
        timestr = sprintf('%d day',d);
    else
        timestr = sprintf('%d day, %d hr',d,h);
    end
elseif h > 0
    if h > 9
        timestr = sprintf('%d hr',h);
    else
        timestr = sprintf('%d hr, %d min',h,m);
    end
elseif m > 0
    if m > 9
        timestr = sprintf('%d min',m);
    else
        timestr = sprintf('%d min, %d sec',m,s);
```

```
    end
else
    timestr = sprintf('%d sec',s);
end

function closeBar(src,evnt)

selection = questdlg('¿Quiere parar este proceso?',...
                     'Parar Proceso',...
                     'Si','No','Si');
switch selection,
   case 'Si',
    delete(gcf)
   case 'No'
     return
end
```

ANEXO III: UICONTROLS GUI

BackgroundColor

El color usado para rellenar el rectángulo de uicontrol. Específica un color usando un vector de tres elementos RGB (rojo, verde y azul) o uno de los nombres ya predefinidos en Matlab. El color por "default" es determinado por la configuración del sistema.

BusyAction

Interrupción de la rutina de llamada (callback). Si una es ejecutada y el usuario activa un evento en un objeto para el cual una llamada está definida, esa llamada trata de interrumpir la primera llamada. La primera llamada puede ser interrumpida solamente por uno de los siguientes comandos: *drawnow*, *figure*, *getframe*, *pause* o *waitfor*; si la llamada no contiene ninguno de estos comandos no puede ser interrumpida.

Si la propiedad *Interruptible* del objeto que se está ejecutando la llamada esta desactivada (off), la llamada no puede ser interrumpida (excepto por algunas llamadas). La propiedad *BusyAction* del objeto que su llamada está esperando para ejecutarse determina lo que le pasa a la llamada:

Si el valor es queue, la llamada es agregada al evento queue y se ejecuta después de que la primera llamada termine de ejecutarse.

Si el valor es Cancel, el evento es descartado y la llamada no se ejecuta.

Nota: Si la llamada interrumpida es una llamada de *DeleteFcn* o *CreateFcn* o una de una figura de *CloseRequest* or *ResizeFcn*, se interrumpe y ejecuta sin importar el valor de la propiedad Interrumpible del objeto

ButtonDownFcn

Una rutina de llamada que se ejecuta cuando presionas un botón del ratón mientras el cursor esta en un unicontrol. Cuando la propiedad enable del unicontrol esta desactivada, el *ButtonDownFcn* se ejecuta cuando haces click en el unicontrol. Esto es útil para implementar acciones para modificar interactivamente las propiedades de control del objeto, como el tamaño y la posición.

Esta rutina se define como una cadena (string) que es una expresión valida en Matlab o el nombre de un archivo M (M-file). La expresión se ejecuta en el espacio de trabajo de Matlab.

La propiedad de llamada define la rutina de llamada que se ejecuta cuando das click en el botón.

Callback

Controla la acción. Una rutina que se ejecuta cuando se activa un objeto de la clase uicontrol.

Define esta rutina como una cadena. La expresión se ejecuta en el espacio de trabajo de Matlab.

Para ejecutarla rutina para un control de texto editable, escribe el texto deseado y después sigue uno de los siguientes pasos:

-Mueve la selección del objeto (da click en cualquier otra parte)
-Para un texto editable de una sola línea, presiona Return
-Para una caja de texto (text box), presiona Ctrl-Return.

Esta rutina definida para los componentes frame y static text no se ejecuta porque ninguna acción está asociada con estos objetos.

Cdata

Imagen de color verdadero mostrada en un control. Una matriz tridimensional de valores RGB que definen una imagen de color verdadero que es mostrada ya sea en un push button o un toggle button. Cada valor debe tener un rango entre cero y uno.

CreateFcn

Rutina de llamada ejecutada cuando se crea un objeto. Esta propiedad define una rutina de llamada que es ejecutada cuando Matlab crea un objeto de la clase uicontrol. Se debe definir esta propiedad como un valor por default para los uicontrols.

DeleteFcn

Una rutina de llamada que se ejecuta cuando borras un objeto uicontrol. Matlab ejecuta la rutina antes de destruir las propiedades del objeto, así sus valores están disponibles para la rutina de llamada.

Enable

Activa o desactiva el uicontrol. Esta propiedad controla como los uicontrols responden a un click del mouse, incluyendo que rutina de llamada se ejecuta.

on - El uicontrol es operacional
inactive - no es operacional pero se ve como si estuviera activado
off - No es operacional y su etiqueta se vuelve gris

Cuando se da click izquierdo a un objeto uicontrol que tiene su propiedad enable activada (en on), Matlab ejecuta estas en este orden:

- Asigna la propiedad de la figura SelectionType

- Ejecuta la rutina de llamada del control.

- No asigna la propiedad de la figura CurrentPoint y tampoco ejecuta ni la propiedad de control ButtonDownFcn ni la rutina de llamada de la figura WindowButtonDownFcn

Cuando se da click izquierdo en un uicontrol en el cual su propiedad Enable está inactiva, o cuando das click derecho en uno en el que Enable tiene cualquier valor, Matlab ejecuta estas acciones in este orden:

- Asigna la propiedad de la figura SelectionType.

- Asigna la propiedad de la figura CurrentPoint.

- Ejecuta la rutina de llamada WindowButtonDownFcn de la figura.

- En un click derecho, si el uicontrol está asociado con un context menú, registra el context menú

- Ejecuta la llamada ButtonDownFcn del control

- Ejecuta la llamada del elemento seleccionado del context menú

- No ejecuta la rutina de llamada del control

Poniendo esta propiedad inactiva te capacita para implementar arrastre o cambio de tamaño de objetos usando la rutina de llamada ButtonDownFcn.

Extent

Tamaño de un carácter cadena uicontrol. Un vector de cuatro elementos que define el tamaño y la posición de un carácter de tipo cadena usado para etiquetar el uicontrol.

Tiene la forma:

[0, 0, width, height]

Los dos primeros elementos siempre son cero. width (ancho) y height (alto) son las dimensiones del rectángulo. Todas las medidas son unidades especificadas por la propiedad Units.

Ya que la propiedad Extent está definida en las mismas unidades que el uicontrol mismo, puedes usar esta propiedad para determinar el tamaño adecuado del ancho del uicontrol con respecto a su etiqueta. Haciendo lo siguiente:

- Definiendo la propiedad String y seleccionando la fuente usando las propiedades relevantes.

- Tomando el valor de la propiedad Extend

- Definiendo width y height de la propiedad Position (posición) propiamente a ser de alguna manera más grandes que width y height de Extend.

FontAngle

Inclinación de un carácter. Poniendo esta propiedad en Italic (italica) u oblique (oblicua) selecciona una versión inclinada de la fuente, cuando está disponible en tu sistema.

FontName

El nombre de la fuente que mostrara la cadena. Para mostrar e imprimir correctamente, debe ser un tipo de fuente que tu sistema soporte.

Para usar un ancho ajustado que se vea bien en cualquier exterior (y que se muestre correctamente en Japón, donde usan caracteres "multibyte"), Pon al FontName la cadena FixedWidth (esta cadena es sensible a las mayúsculas):

set (uicontrol_handle, 'FontName', 'FixedWidth')

FontSize

Tamaño de la fuente. Un número que especifica el tamaño de la fuente que va a ser mostrado en la cadena, en unidades determinadas por la propiedad FontUnits. El tamaño por default es dependiente del sistema.

FontUnits

Unidades del tamaño de la fuente. Esta propiedad determina las unidades usadas por la propiedad FontSize. Las unidades normalizadas interpretan el FontSize como una fracción de la altura del uicontrol. Cuando tú cambias el tamaño del uicontrol, Matlab

modifica la pantalla FontSize. pixels (pixeles), inches (pulgadas), centimeters (centímetros) y points (puntos) son unidades absolutas (1 punto = 1/72 pulgada).

FontWeight

Formato de un carácter. Poniendo esta propiedad en Bold hace que Matlab use una versión "negrita" de la fuente, cuando está disponible en tu sistema.

ForegroundColor

Color de texto. Esta propiedad determina el color del texto definido por la propiedad String. Especifica un color usando un vector de tres elementos RGB o un nombre predefinido en Matlab.

HandleVisibility

Controla el acceso al manejador (handle) de un objeto por usuarios de la línea de comando y GUI's. Esta propiedad determina cuando un manejador de un objeto es visible en la lista de los objetos hijos de su clase padre.

HandleVisibility es útil para prevenir que los usuarios puedan accidentalmente borrar o dibujar en una figura que contiene solo dispositivos de interfaz de usuarios.

Los identificadores son siempre visibles cuando HandleVisibilty esta activada (en on).

Asignando HandleVisibility a una llamada hace que el manejador sea visible para rutinas de llamada o funciones invocadas por rutinas de llamada, pero no para las que son invocadas desde la línea de comando. Esto es para proteger los GUI's de los usuarios de la línea de comando, y también permite a las rutinas de llamada tener el completo acceso a los identificadores de los objetos.

Poniendo HandleVisibility en off hace al manejador siempre invisible. Esto puede ser necesario cuando una rutina de llamada invoca a una función que puede dañar al GUI, porque así temporalmente esconde sus propios identificadores mientras se ejecuta dicha función.

Cuando un manejador no es visible en la lista de hijos de su clase padre, no puede ser regresado por funciones que obtengan identificadores buscando la jerarquía del objeto o "preguntando" las propiedades del manejador. Esto incluye get, findobj, gca, gcf, gco, newplot, cla, clf y close.

Cuando la visibilidad del manejador (HandleVisibility) es restringida usando llamadas o poniéndola en off, el manejador del objeto no aparece en la propiedad children de su

padre, las figuras no aparecen en la propiedad CurrentFigure de Root (raiz), los objetos no aparecen en la propiedad de Root CallbackObjet o en la propiedad de la figura CurrentObjet, y los Axes (ejes) no aparecen en la propiedad CurrentAxes de sus padres.

Puedes poner la propiedad de Root ShowHiddenHandles en on para hacer visibles a todos los identificadores, sin importar los ajustes de su propiedad HandleVisibility (esto no afecta sus valores).

Los identificadores que están escondidos siguen siendo validos, si se conoce el manejador de un objeto tu puedes asignar (set) y obtener (get) sus propiedades, y pasárselas a cualquier función que opere identificadores

HorizontalAlignment

Alineación Horizontal de una cadena de una etiqueta. Esta propiedad determina la justificación del texto definido por la propiedad String (la etiqueta uicontrol):

- left - El texto se justifica a la izquierda con respecto al uicontrol
- center - El texto se centra con respecto al uicontrol
- right - El texto se justifica a la derecha con respecto al uicontrol

En el sistema operativo Windows de Microsoft, esta propiedad afecta solo a los uicontrols edit y text.

Interruptible

Rutina de llamada modo de interrupción. Si una llamada se está ejecutando y el usuario dispara un evento (como un click del mouse) en un objeto para el cual está definida una llamada, la llamada intenta interrumpir la primera llamada. Matlab procesa las llamadas conforme a estos factores:

La propiedad Interruptible del objeto que se está ejecutando la llamada

- Si la llamada que se está ejecutando contiene las declaraciones drawnow, figure, getframe, pause o waitfor
- La propiedad BusyAction de el objeto que su llamada está esperando para ejecutarse

Si la propiedad Interruptible del objeto que su llamada se está ejecutando está en on, la llamada puede ser interrumpida. La llamada interrumpe ejecución a la siguiente declaración drawnow, figure, getframe, pause o waitfor, y procesa los eventos en el evento de la cola, en el cual incluye la llamada que está esperando.

Si la propiedad Interruptible del objeto que su llamada se está ejecutando esta en off, la llamada no puede ser interrumpida (a excepción de ciertas llamadas). La propiedad BusyAction del objeto de el cual su llamada está esperando para ejecutarse determina que pasa con la llamada.

Nota: Si la llamada interrumpida es una llamada de *DeleteFcn* o *CreateFcn* o una llamada de *CloseRequest* o ResizeFcn de una figura, interrumpe la llamada que se está ejecutando sin importar el valor de la propiedad Interruptible de ese objeto. La llamada interruptora empieza la ejecución a la siguiente declaración de drawnow, figure, getframe, pause o waitfor. La rutina de llamada WindowButtonDownFcn de una figura, o las propiedades ButtonDownFcn o Callback de un objeto son procesadas de acuerdo a las reglas descritas arriba.

ListboxTop

Esta propiedad se aplica solo al estilo de uicontrol listbox. Especifica que cadena aparece en la posición más alta de un Listbox (caja de lista) que no es lo suficientemente largo para mostrar todos los elementos de la lista. ListboxTop es un índice en el arreglo de cadenas definido por la propiedad String y debe tener un valor entre cero y el número de cadenas. Valores no enteros son redondeados al próximo valor entero más pequeño.

Max

Valor máximo. Esta propiedad especifica el valor más grande aceptado por la propiedad Value.

Diferentes estilos de uicontrols interpretan la propiedad max de diferente manera:

- Check boxes - Max es el valor de la propiedad Value mientras el check box está seleccionado.

- Edit text - Si Max - Min > 1, entonces el edit text (texto editable) acepta varias líneas de entrada. Si Max - Min <= 1, entonces el editable text solo aceptan una línea de entrada.

- List boxes - Si Max - Min > 1, entonces el listbox acepta selección múltiple de elementos. Si Max - Min <= 1, entonces no aceptan selección múltiple

- Radiobuttons - Max es el valor de la propiedad Value mientras el radiobutton está seleccionado.

- Sliders - Max es el valor máximo del slider (deslizador) y tiene que ser más grande que la propiedad min. El valor por defecto es 1.

- Toggle buttons - Max es el valor de la propiedad Value mientras el radio button está seleccionado. El valor por defecto es 1.

- Frames, pop-up menús, pushbuttons, y statictext no usan la propiedad Max

Min

Valor mínimo. Esta propiedad especifica el valor más pequeño aceptado por la propiedad Value.

Diferentes estilos de uicontrols interpretan la propiedad min de diferente manera:

- Check boxes - Min es el valor de la propiedad Value mientras el checkbox no está seleccionado.

- Edit text - Si Max - Min > 1, entonces el edit text (texto editable) acepta varias líneas de entrada. Si Max - Min <= 1, entonces el edit text solo aceptan una línea de entrada.

- List boxes - Si Max - Min > 1, entonces el list box acepta selección múltiple de elementos. Si Max - Min <= 1, entonces no aceptan selección múltiple

- Radio buttons - Min es el valor de la propiedad Value mientras el radio button no está seleccionado.

- Sliders - Min es el valor mínimo del slider (deslizador) y tiene que ser menos que la propiedad max. El valor por defecto es 0.

- Toggle buttons - Min es el valor de la propiedad Value mientras el radio button no está seleccionado. El valor por defecto es 0.

- Frames, pop-up menus, push buttons, y static text no usan la propiedad Min

Parent

Padre de un unicontrol. El manejador del objeto padre del uicontrol. El padre de un objeto uicontrol es la figura (figure) en el que aparece. Puedes mover un objeto uicontrol a otra figura asignando esta propiedad al manejador del nuevo padre.

Position

Tamaño y posición de un uicontrol. El rectángulo definido por esta propiedad especifica el tamaño y la posición del control en la ventana de la figura. Especifica la posición como:

[left bottom width height]

left (izquierda) y bottom (abajo) son la distancia de la esquina izquierda de abajo de la figura a la esquina izquierda de abajo del objeto. width (ancho) y height (alto) son las dimensiones del rectángulo uicontrol. Todas las medidas están en unidades especificadas por la propiedad Units.

En el sistema operativo Windows de Microsoft, la altura de los pop-up menus es automáticamente determinada por el tamaño de la fuente. El valor que tú especifiques para height de la propiedad Position no tiene ningún efecto.

Los valores de width y height determinan la orientación de los sliders (deslizadores). Si width es más grande que height, entonces el slider es orientado horizontalmente, de lo contrario es orientado de manera vertical.

Selected

Objeto seleccionado. Cuando esta propiedad esta activada (en on), Matlab muestra identificadores de selección si la propiedad SelectionHighlight está también en on. Tú puedes, por ejemplo, definir ButtonDownFcn para que ajuste a esta propiedad, permitiendo al usuario que seleccione el objeto con el mouse.

SelectionHighlight

Objeto resaltado cuando esta seleccionado. Cuando la Propiedad está en on, Matlab indica el estado de seleccionado dibujando cuatro identificadores de filo y cuatro de horilla. Cuando la propiedad esta en off, Matlab no dibuja los identificadores. ****

SliderStep

Paso del slider (deslizador). Esta propiedad controla la cantidad que la propiedad del slider Value cambia cuando le das click en el botón de la flecha (min_step) o en la barra del slider (max_step). Especifica SliderStep como un vector de dos elementos; cada valor debe estar en un rango de cero y uno. El tamaño del paso es una función del especificado SliderStep y al rango total del slider (Max - Min). El valor por defecto, [0 .01 0 .10], proporciona 1 por ciento oportunidades de clicks en el botón de la flecha y 10 por ciento en la barra.

Por ejemplo, si creas el siguiente slider,

uicontrol ('style', 'slider', 'min', 1, 'max', 7,... 'SliderStep', [0 .1 0 .6]), dando click en el botón de la flecha mueve el indicador por, 0.1*(7-1)

ans= 0.6

y dando click en la barra mueve el indicador 0.6*(7-1)

ans = 3.6

Nota: que si el tamaño del paso especificado mueve el slider a un valor fuera del rango, el indicador se mueve solo al valor max o min.

String

Para los check boxes, edit text, push buttons, radio buttons, static text, and toggle buttons, el texto que se muestra en el objeto. Para los list boxes y pop-up menú, el conjunto de elementos o artículos del objeto.

Para objetos uicontrol que muestran solo una línea de texto, si el valor de String es especificado como arreglo de celdas tipo cadena (los elementos en las celdas son cadenas) o una matriz rellena, solo la primer cadena se muestra; los demás son ignorados, el carácter guion vertical ('|') no es interpretado como un cambio de línea sino que se muestra en el texto del uicontrol. Para texto editable o estático (edit text o static text), cambios de línea ocurren entre cada columna de la matriz, cada celda del arreglo tipo cadena y después de cada carácter \n.

Para múltiples elementos en un list box o un pop-up menú, tu puedes especificar los elementos como un arreglo de celdas tipo cadenas, una matriz rellena de cadenas, o en un vector separados por guiones verticales ('|').

Para edit text, el valor de esta propiedad es asignado a la cadena capturada por el usuario.

Style

Estilo del objeto uicontrol a crear. La propiedad Style (estilo) especifica el tipo de uicontrol a crear.

Tag

Etiqueta del objeto especificada por el usuario. La propiedad tag proporciona una manera de identificar objetos gráficos con una etiqueta especificada por el usuario. Esta es particularmente útil cuando construyes programas gráficos interactivos que de otra manera necesitarían definir identificadores de objetos como variables globales o pasarlas como argumentos entre rutinas de llamada. Puedes definir la propiedad tag como cualquier cadena.

TooltipString

Contenido de la "pista" (el letrero que aparece si dejas el mouse sobre un objeto sin dar click). La propiedad TooltipString especifica el texto de la "pista" asociado con el uicontrol. Cuando el usuario mueve el puntero mouse sobre un control y lo deja ahí, la pista aparece.

Type

Clase de objeto gráfico. Para los objetos uicontrol, la propiedad Type siempre es la cadena 'uicontrol'.

UIContextMenu

Asocia un context menú con uicontrol. Asigna a esta propiedad el manejador de un objeto uicontextmenu. Matlab muestra el context menú cuando das click derecho en el uicontrol. Usa la función uicontextmenu para crear el context menú.

Units

Unidades de medida. Las unidades que Matlab usa para interpretar las propiedades Extend y Position. Todas las unidades son medidas de la esquina inferior izquierda de la figura ventana.

UserData

Datos especificados por el usuario. Cualquier dato que quieras asociar con un objeto uicontrol. Matlab no usa este dato pero tú puedes acceder a él utilizando los comandos set (asigna) y get (recupera).

Value

Valor actual del uicontrol. La clase (o estilo) uicontrol determina los posibles valores que esta propiedad puede tener:

- Check boxes - ponen su propiedad Value (valor) en Max (el máximo) cuando están seleccionados y en Min (el mínimo) cuando no están seleccionados.

- List boxes - ajustan su valor a un vector correspondiente a los elementos de la lista

- Pop-up menú - ajustan su valor a un índice de artículos seleccionados, donde 1 corresponde al primer elemento del menú.

- Radio buttons - ponen su propiedad Value (valor) en Max (el máximo) cuando están seleccionados y en Min (el mínimo) cuando no están seleccionados.

- Sliders - ajustan su valor al número indicado por la posición del slider (deslizador)

- Toggle buttons - ponen su propiedad Value en Max cuando están presionados (seleccionados) y en Min cuando no están seleccionados.

- Edit text, Frames, Push buttons, y Static text no usan esta propiedad.

Puedes ajustar este valor ya sea interactivamente con el mouse o a través de la función set.

Visible

Visibilidad del uicontrol. Por defecto, todos los uicontrols son visibles. Cuando esta propiedad está desactivada (en off), el control no es visible, pero sigue existiendo y puedes buscar y modificar su propiedades.

REFERENCIAS

[1] Proyecto Fin de Carrera: “Estudio y despliegue de redes WI-FI”

[2] Proyecto Fin de Carrera: “Modelo de Cobertura para Redes Inalámbricas de Interiores”

[3] J.G. Proakis. Digital communications. McGraw-Hill Series in Electrical Engineering. McGraw-Hill Book Company, second edition, 1989.

[4] William C.Y. Lee. Estimate of local average power of a mobile radio signal. IEEE Transactions on Vehicular Technology, 34(2), 1985.

[5] J.D. Parsons and A.S. Bajwa. Wideband characterization of fading mobile radiochannels. IEE Proceedings-F, 129(2):95{101, Abril 1982.

[6] J.D. Parsons. The mobile radio propagation channel. Pentech House, London,UK, 1992.

[7] Philip A. Bello. Characterization of randomly time-variant linear channels. IEEETransactions on Communication Systems, 4(11):360{393, Diciembre 1963.

[8] Santiago J. Flores Asenjo. Caracterización del Canal Radio Móvil en el Interior de Edificios con Múltiples plantas mediante Técnicas de Lanzado de Rayos.

[9] Homayoun Hashemi. The indoor radio propagation channel. Proceedings of the IEEE, 81(7):943-967, Julio 1993.

[10] H. Hashemi and D.Tholl. Statistical modeling and simulation of the RMS delayspread of indoor radio propagation channels. IEEE Transactions on VehicularTechnology, 43(1):110-120, Febrero 1994.

[11] R. Agustí, F.J. Pérez, and S. Ruiz. Obtención del perfil de retardo de potencias en un entorno “indoor” mediante el uso de un analizador de redes. In VI Symposium Nacional de la URSI, Cáceres, Septiembre 1991. URSI.

[12] A.S.Bajwa and J.D.Parsons. Small-area characterization of UHF urban andsuburban mobile radio propagation. IEE Proceedings-F, 129(2):102-109, Abril1982.

[13] U. Dersch and E. Zollinger. Propagation mechanisms in microcell and indoor environments. In IEEE Proceedings, Joint COST 227 Workshop on Mobil Communications, Septiembre 1993.

[14] V.M. Pérez and J. Jiménez. Medidas de propagación a 1.8GHz en ambientes interiores. In VII Symposium Nacional de la URSI, volume 2, pages 925-929, Valencia, Septiembre 1993. URSI.

[15] D. Molkdar. Review on radio propagation into and within buildings. IEE Proceedings-H, 138(1):61-73, Febrero 1991.

[16] D.C. Cox, R.R. Murray, and A.W. Norris. Measurements of 800MHz radio transmission into buildings with metallic walls. The Bell System Technical Journal, 62(9):2695-2717, Noviembre 1983.

[17] S.E. Alexander. Characterising buildings for propagation at 900MHz. Electronic Letters, 19(20):860, Septiembre 1983.

[18] D.M.J. Devarsirvatham, C. Banerjee, M.J. Krain, and D.A. Rappaport. Multi frequency radiowave propagation measurements in the portable radio environment. In Conference Record of the IEEE International Conference on Communications'90, volume 4. IEEE, Abril 1990.

[19] J.M. Keenan and A.J. Motley. Radio coverage in buildings. British Telecom Technology Journal, 8(1):19-24, Enero 1990.

[20] R. Herradón, F. Jiménez, "Caracterización Estadística de la propagación para Comunicaciones Inalámbricas en el interior de fábricas". EUIT de Telecomunicaciones. Universidad Politécnica de Madrid.

[21] C. Tornevik, J.E. Berg, and F. Lotse. 900MHz propagation measurements andpath loss models for diferent indoor environments. In Proceedings of VTC'93, New Jersey, USA, Mayo 1993. IEEE.

[22] Scott Y. Seidel and T.S. Rappaport. 914MHz path loss prediction models for indoor wireless communications in multifloored buildings. IEEE Transactions on Antennas and Propagation, 40(2):207-217, Febrero 1992.

[23] F.C. Owen and C.D. Pudney. In-building propagation at 900MHz and 1650MHz for Digital Cordless Telephones. In Sixth International Conference on Antennas and Propagation, ICAP'89, volume 2: Propagation, University of Warwick, Coventry, UK, Abril 1989. IEEE.

[24] Emmanuele Trucco and Alessandro Verri. Introductory Techniques for 3-D Computer Vision. Prentice Hall, 1998

[25] J.W. McKown and R.L. Hamilton, Jr. Ray tracing as a design tool for radionetworks. IEEE Network Magazine, 5(6):27-30, Noviembre 1991.

[26] F. Escolano, O. Colomina, M.A.Cazorla. Visión Artificial: Extracción de Características I, 2006.

[27] Hill Green. Canny Edge Detection Tutorial. 2002.

[28] Marchand, P. Holland T, Graphics and GUIs with Matlab, Charlan & Hall/CRC, 198-207.

[29] Matlab Central disponible en:

http://www.mathworks.com/Matlabcentral/link_exchange/Matlab/Tutorials/

[30] Matlab disponible en http://www.math.utah.edu/lab/ms/Matlab/Matlab.html

[31] Manual de Matlab versión 7.0