Subject: Re: WMENU Routine
Posted by Matthew Larkum on Thu, 13 Jun 1996 07:00:00 GMT
View Forum Message <> Reply to Message

mike harwood wrote:
>
> I have recieved some IDL code from another organization with a WMENU
> routine. I have looked in the IDL documentation but cannot find any
> references to this. Can anyone tell me what this routine does?
>
> ThanksI used to use the WMENU function a lot with PV-Wave utill we switched to IDL for Windows where it wasn't supported. The manual for PV-Wave reads thus:

WMENU Function

Displays a menu inside the current window whose choices are given by the elements of a string array and which returns the index of the user's response.

Usage

result = WMENU(strings)

Returned Value

result = Returns -1 if the user did not select a menu item. Otherwise, the returned value ranges from 0 to the number of elements in strings minus one.

Input Keywords

Initial_Selection - The index of the initial selection.

Title - The index of the strings element that is the title.

Xpos - The position on the X axis for menu

Ypos - The position on the Y axis for menu

I solved the problem by making a replace all of WMENU with WMENU2 as given below. It calls XWMENU which is also given below. Please don't write and tell me my widget programming is stuffed. This was my first attempt with IDL.

Hope it helps,

Matthew.

Larkum@pyl.unibe.ch

---------- Cut Here ----------

```
;
; $Id: template,v 1.1 91/03/11 20:30:47 jeffry Exp $
;
;+
; NAME: WMENU2
; WRITTEN BY: Matthew Larkum, University of Bern, Switzerland.
; DATE: Tuesday, 30th August, 1994.
; PURPOSE: Displays menu and returns choice - works like WMENU for IDL
; for Windows as well.
; CATEGORY: Input/Output
; CALLING SEQUENCE: choice = WMENU2(list)
; INPUTS: list = string array containing menu choices.
; OPTIONAL INPUT PARAMETERS: None.
; KEYWORD PARAMETERS: init = integer number for the initial choice highlighted.
; title = integer number if the first element of list is the title
; OUTPUTS: None.
; OPTIONAL OUTPUT PARAMETERS: None.
; COMMON BLOCKS: None.
; SIDE EFFECTS: None.
; RESTRICTIONS: None.
; PROCEDURE:
; EXAMPLE: choice = WMENU2(['Choose a color:','Red','White','Blue'], $
;   init=1,title=0)
; MODIFICATION HISTORY:
;-
function wmenu2, list, init=init, title=title


if !d.name eq "WIN" then begin
 if n_elements(title) ne 0 then begin
  newlist = list(1:n_elements(list)-1)
  title = list(0)
  add = 1
  endif $
 else begin
  newlist = list
  title = "Menu"
  add = 0
  endelse

 return, xwmenu(newlist,title=title)+add
 endif $ ; using IDL for windows
else begin
 if n_elements(init) eq 0 then init = 0
```

```
 if keyword_set(title) then $
  return, wmenu(list,init=init,title=title) $
 else $
  return, wmenu(list,init=init)
 endelse

end

---------- Cut Here ----------

;
; $Id: template,v 1.1 91/03/11 20:30:47 jeffry Exp $
;
;+
; NAME: XWMENU
; WRITTEN BY: Matthew Larkum, University of Bern, Switzerland.
; DATE: Fri, 17th October, 1995.
; PURPOSE: Widget version of old WMENU routine with yummy extras
; CATEGORY: widgets
; CALLING SEQUENCE: value = XWMENU(list)
; INPUTS: list = array of values for the menu list
; OPTIONAL INPUT PARAMETERS: None.
; KEYWORD PARAMETERS: title = title for menu
; cancel = adds a cancel button (return value always -1)
; values = list of return values corresponding to list values
;  defaults to the number as in old WMENU function
; OUTPUTS: value of button chosen
; OPTIONAL OUTPUT PARAMETERS: None.
; COMMON BLOCKS: None.
; SIDE EFFECTS: None.
; RESTRICTIONS: None.
; NOTE: Works like the old WMENU procedure.  This procedure had a
; keyword, "init" which is now redundant, but can be given to
; XWMENU anyway - it just doesn't do anything (for backward
; compatibility)
; PROCEDURE: To return the values 20,40 & 60 from a menu with cancel
; button (value = -1) do the following:
;
; value = XWMENU(indgen(3)*20+20,/cancel,values=indgen(3)*20+20,$
;  title='What you see is what you get!')
; MODIFICATION HISTORY:
;-


PRO xwmenu_event, event
; This is the event handler for a "xwmenu" widget.

; The COMMON block is used because both xwmenu and xwmenu_event must
```

```
; know about the group leader.
COMMON xwmenugroup, xwbase, rvalue

; Use WIDGET_CONTROL to get the user value of any widget touched and put
; that value into 'eventval':

WIDGET_CONTROL, event.id, GET_UVALUE = eventval

; Perform actions based on the user value of the button which was pressed.

rvalue = eventval
widget_control, event.top, /destroy

END



PRO xwmenupro, options, GROUP=GROUP, title=title, cancel=cancel, values=values
; This is the procedure that creates a widget application

; The COMMON block is used because both xwmenu and xwmenu_event must
; know the group leader.
COMMON xwmenugroup, xwbase, rvalue

; A top-level base widget with the title "Pop-Up Widget Example" will
; hold the exclusive buttons:
xwbase = WIDGET_BASE(TITLE = title, $
 /COLUMN, $
 XSIZE=8*max([(size(byte(strtrim(options,2))))(1), 5*(cancel+1), strlen(title)]))

textframe = WIDGET_TEXT(xwbase,value=title)
but = intarr(n_elements(options))
for i=0,n_elements(options)-1 do $
 but(i) = WIDGET_BUTTON(xwbase, $
  UVALUE = values(i), $
  VALUE = strtrim(options(i),2))
if keyword_set(cancel) then $
 cancelbut = widget_button(xwbase, $
  uvalue = -1, $
  value = 'CANCEL')

; Realize the widgets:
WIDGET_CONTROL, xwbase, /REALIZE

; Hand off control of the widget to the XMANAGER:
XMANAGER, "xwmenu", xwbase, GROUP_LEADER=GROUP, event_handler="xwmenu_event",
$
 /modal
```

```
END

function xwmenu, options, title=title, cancel=cancel, values=values, init=init
COMMON xwmenugroup, xwbase, rvalue

if not keyword_set(title) then title = "Menu"
if not keyword_set(cancel) then cancel = 0 $
else cancel=1
if not keyword_set(values) then values = indgen(n_elements(options))

xwmenupro, options, title=title, cancel=cancel, values=values
if n_elements(rvalue) eq 0 then rvalue = -1 ;*** default cancel
return, rvalue
end
```