Subject: Re: Interface, Widget, Motif questions
Posted by idl on Mon, 21 Mar 1994 08:27:14 GMT
View Forum Message <> Reply to Message

In article 2m3nsjINNh2@i32.sma.ch, stl@sma.ch (Stephen Strebel) writes:

stuff deleted

(simple things like coloring a button; dropdown list boxes;
> spin buttons; etc..) I would love to heat the creative (even if ugly
>) things people are doing. Or does this problem really not exist and
> there is some really amazing little tool out there that i on't know
> about. :-)
>
Since IDL 3.0 it is possible to write compound widgets (see IDL User's Guide 3.5, chapter 20) . I'm sure there will be soon a lot of such compound widgets on the net from the idl user community. This will solve most of your needs. As a first step in this direction I wrote a such a compound widget
for you. It creates colored buttons, one of your wishes above ...

Have fun, Thomas

```
;
;+
; NAME:
;    CW_CLBTN
;
; PURPOSE:
;    Creates a color button
;
; CATEGORY:
;    Compound Widgets
;
; CALLING SEQUENCE:
;    widget = CW_CLBTN(parent)
;
; INPUTS:
;       PARENT - The ID of the parent widget.
;
; KEYWORD PARAMETERS:
;       Note:    There is no VALUE keyword to set the button label
;                allowed on widget creation because the button is based
;                small draw widget, the button label cannot be set
;                before the widget is realized, so you have to use
;                widget_control, <id>, SET_VALUE after your main base
;                has been realized. See example below.
;
; UVALUE - Supplies the user value for the widget.
```

```
;
;      XSIZE    Button x size in device coordinates
;
;      YSIZE    Button y size in device coordinates
;
;      FONT  Font for button label
;
;      COLORS   intarr(4) 4 element vector with color indices for:
;
;               element 1 ( index 0):  light side ( left, top )
;               element 2 ( index 1):  button front
;               element 3 ( index 2):  shadow side ( right, bottom )
;   element 4 ( index 3):  button label color
;
;
;     colors(0) - - - - - - +
;        |      +-----------|---------------+
;               |\          V              /|
;        |      | \                      / |
;               |  +---------------------+  |
;        |      |  |                     |  |
;               |  |    colors(1)        |  |
;        + - - -|> |                     |  |
;               |  |                     |  |
;               |  |   <Button label>    |  |
;               |  |          ^          | <- - - - +
;               |  |       colors(3)     |  |       |
;               |  |                     |  |
;               |  |                     |  |       |
;               |  +---------------------+  |
;               | /                     \ |       |
;               |/              ^       \|
;               +-----------------|---------+       |
;
;                                 + - - - - - - colors(2)
;
;
;
; OUTPUTS:
;      The ID of the created widget is returned.
;
; COMMON BLOCKS:
; None.
;
; EXAMPLE:
; Extract all inside the '*' border and save to 't_clbtn.pro'. Call then
; t_clbtn from the IDL prompt.
;  ********************************************************** ************
;
```

```
; * PRO T_CLBTN_EVENT, ev
; * COMMON t_clbtn_com, s
; *
; *   CASE ev.id OF
; *           s.red_btn_w: begin
; *                          CASE ev.type OF
; *                            0: print, 'red button pressed'
; *                            1: print, 'red button rleased'
; *                          ENDCASE
; *                        end
; *           s.blue_btn_w: begin
; *                          CASE ev.type OF
; *                            0: print, 'blue button pressed'
; *                            1: print, 'blue button rleased'
; *                          ENDCASE
; *                       end
; *           s.green_btn_w: begin
; *                          CASE ev.type OF
; *                            0: print, 'green button pressed'
; *                            1: print, 'green button rleased'
; *                          ENDCASE
; *                        end
; *      s.quit_btn_w: begin
; *                          widget_control, ev.top, /DESTROY
; *                        end
; *   ENDCASE
; *
; * END
; *
; * PRO T_CLBTN
; * COMMON t_clbtn_com, s
; * Test CW_CLBTN, compound widget for color buttons
; *
; * CASE !d.name OF
; *
; *   'WIN': fn = 'TIMES*ITALIC*BOLD*18'
; *
; *   'X':   fn = '-adobe-times-bold-i-normal--18-180-75-75-p-98-iso8859-1'
; *
; * ELSE: message, 'Add a valable font name for '+!d.name
; *
; * ENDCASE
; *
; * ; Define simple color table for color button demo
; *
; * ;   0     1    2    3      4      5      6       7     8     9
; * ; black white red dk_red, blue, dk_blue, green, dk_green grey  dk_grey
; *
```

```
; * r=[ 0,   255, 255, 160,     0,      0,      0,       0,   200,   80   ]
; * g=[ 0,   255,   0,   0,     0,      0,    255,     160,   200,   80   ]
; * b=[ 0,   255,   0,   0,   255,    160,      0,       0,   200,   80   ]
; *
; * ; Create dummy pixmap to reduce the number of colors used by
; * ; this idl session to 10 ( to  avoid color flashing )
; * window, XSIZE=10, YSIZE=10, COLORS=10, /PIXMAP, /FREE
; * wdelete, !d.window
; * tvlct, r, g, b
; *
; *  base_w = widget_base(TITLE='Test for cw_clbtn')
; *  cntl_w = widget_base(base_w, /ROW)
; *  red_btn_w   = cw_clbtn(cntl_w, COLORS=[1,2,3,1], XSIZE=54, YSIZE=40, $
; *               FONT=fn)
; *  blue_btn_w  = cw_clbtn(cntl_w, COLORS=[1,4,5,1], XSIZE=54, YSIZE=40, $
; *              FONT=fn)
; *  green_btn_w = cw_clbtn(cntl_w, COLORS=[1,6,7,0], XSIZE=54, YSIZE=40, $
; *               FONT=fn)
; *
; *  quit_btn_w = cw_clbtn(cntl_w, COLORS=[1,8,9,0], XSIZE=54, YSIZE=40, $
; *               FONT=fn)
; *
; *
; *  s = { red_btn_w:  red_btn_w,   $
; *        blue_btn_w: blue_btn_w,  $
; *        green_btn_w:green_btn_w, $
; *        quit_btn_w:quit_btn_w }
; *
; *  widget_control, base_w, /REALIZE
; *
; *  widget_control, s.red_btn_w, SET_VALUE='red'
; *  widget_control, s.blue_btn_w, SET_VALUE='blue'
; *  widget_control, s.green_btn_w, SET_VALUE='green'
; *  widget_control, s.quit_btn_w, SET_VALUE='quit'
; *
; *  xmanager, 'T_CLBTN', base_w, GROUP = GROUP
; *
; * END
;  ************************************************************ *************
;
; SIDE EFFETCS:
;
; RESTRICTIONS:
;      Requires IDL 3.5.1 or above, only tested on SunOS 4.1.3, Solaris 2.3,
;      MS-Windows 3.1
;      Color table(s) must be well orgnized to avoid unexpected button
;      color changes.
;
```

```
; PROCEDURE:
; WIDGET_CONTROL, id, SET_VALUE=value can be used to change the
;  current value displayed by the widget.
;
; WIDGET_CONTROL, id, GET_VALUE=var can be used to obtain the current
;  value displayed by the widget.
;
; MODIFICATION HISTORY:
;   Written, 15-March-1994, Thomas.Oettli@sma.ch,
;                           Swiss Meteorlogical Institute
;
;-
;

PRO DSP_BTTN, state

 old_w_id=!d.window
 widget_control, state.drw_w, GET_VALUE=curr_w_id
 wset, curr_w_id

 xun=!d.x_size / 13.999
 yun=!d.y_size / 7.999


 polyfill, [0, 14, 14, 0]*xun, [0, 0, 8, 8]*yun, $
           /DEVICE, COLOR=state.colors(1)

 polyfill, [0, 1, 1, 13, 14, 0]*xun, $
           [0, 1, 7, 7, 8, 8]*yun, $
           /DEVICE, COLOR=state.colors(0)

 polyfill, [0, 14, 14, 13, 13, 1]*xun, [0, 0, 8, 7, 1, 1]*yun, $
           /DEVICE, COLOR=state.colors(2)

 if ( state.value NE "" ) then begin

  if ( state.font NE "" ) then device, FONT=state.font

    x = ( !d.x_size / 2 ) - ( state.text_width / 2 )
    y = ( !d.y_size / 2 ) - ( !d.y_ch_size / 2 )
    xyouts, x, y, state.value, FONT=0, $
          /DEVICE, COLOR=state.colors(3), T3d=0

    if ( state.old_font NE "" ) then device, FONT=state.old_font

   endif

   empty
```

```
   if ( old_w_id NE -1 ) then wset, old_w_id

END


FUNCTION CW_CLBTN_EVENT, ev

parent = ev.handler

stash = widget_info(parent, /CHILD)
widget_control, stash, GET_UVALUE = state, /NO_COPY

 CASE ev.id OF

   state.drw_w: begin
        if (  ev.press gt 0 ) then begin

          old_colors=state.colors
          state.colors=[old_colors(1),old_colors(1),$
                       old_colors(1),old_colors(3)]
          dsp_bttn, state
          state.colors=old_colors

        endif else begin
            if  ( ev.release gt 0 ) then begin
             dsp_bttn, state

            endif
           endelse
         end

 ENDCASE

widget_control, stash, SET_UVALUE = state, /NO_COPY

return, { id:parent, top:ev.top, handler:0L, type:ev.type }

END

PRO CW_CLBTN_SET_VAL, id, value

stash = widget_info(id, /CHILD)
widget_control, stash, GET_UVALUE=state, /NO_COPY

state.value=value

; Get text width
```

```
window, XSIZE=200, YSIZE=100, /PIXMAP, /FREE    ; dummy pixmap to draw text
if ( state.font NE "" ) then device, FONT=state.font
xyouts, 0,0, value, FONT=0, width=text_width, /DEVICE
text_width = text_width * !d.x_size
if ( state.old_font NE "" ) then device, FONT=state.old_font
wdelete, !d.window

state.text_width=text_width

dsp_bttn, state

widget_control, stash, SET_UVALUE=state, /NO_COPY

END

FUNCTION CW_CLBTN_GET_VAL, id

 stash = widget_info(id, /CHILD)
 widget_control, stash, GET_UVALUE=state, /NO_COPY

 ret = state.value

 widget_control, stash, SET_UVALUE=state, /NO_COPY
 return, ret

END

FUNCTION CW_CLBTN, parent, XSIZE=XSIZE, YSIZE=YSIZE, $
            UVALUE = uval, COLORS=COLORS, FONT=FONT

if ( n_params() EQ 0 ) then $
  message, 'Yous must specify a parent for cw_clbtn!'
on_error, 2

if not ( keyword_set(uval) ) then uval = 0

if not ( keyword_set(FONT) ) then font = ""

if not ( keyword_set(COLORS) ) then begin
     tmp = !d.table_size / 3
     colors = [3,2,1,0]*tmp
endif

device, GET_CURRENT_FONT=old_font

if ( keyword_set(VALUE) ) then begin
  value=value
  text_width=get_text_width(value, font, old_font)
```

```
endif else begin
  value=''
  text_width=0.0
endelse

if not ( keyword_set(XSIZE) ) then xsize= 20 + text_width

if not ( keyword_set(YSIZE) ) then ysize = 20


state = { drw_w:0,           $
          colors:colors,     $
          value:value,       $
          text_width:text_width, $
          old_font:old_font, $
          font:font }

mainbase = widget_base(parent, UVALUE=uval,$
                 EVENT_FUNC='CW_CLBTN_EVENT', $
                 FUNC_GET_VALUE='CW_CLBTN_GET_VAL', $
                 PRO_SET_VALUE='CW_CLBTN_SET_VAL' )

state.drw_w = widget_draw(mainbase, XSIZE=xsize, YSIZE=ysize, /BUTTON_EVENTS)


widget_control, widget_info(mainbase, /CHILD), SET_UVALUE=state, /NO_COPY

return, mainbase


END
```