Subject: Compound widget problem
Posted by sjt on Tue, 01 Aug 1995 07:00:00 GMT
View Forum Message <> Reply to Message

I have a simple compound widget definition (attached below) which creates
a toggle button which cycles through 2 (or more) states when clicked on.
However, there is a problem if the first menu created and managed
contains such a compound, then sending an event from ANY button (not just
the the coumpound) crashes the XMANAGER with the error:

```
% Array dimensions must be greater than 0.
% Execution halted at XMANAGER </datana/idl/lib/prog/xmanager.pro( 457)>
  (WIDGET_EVENT).
%       Called from GRAFFER_PRO </base/sjt/idl_etc/oddments/graffer.pro( 344)>.
%       Called from $MAIN$ .
```

This error also recurs for any subsequent button event from menus with or
without TBUTTON compounds after the XMANAGER has been cleared & restarted.

But if a previous menu (which may be destroyed before creating the menu
with the TBUTTON compund(s)) has been managed then everything behaves as
it should.

Other type of widgets (e.g. editable text widgets) are not affected at all!

Any ideas???

-------[BEGIN CW_TBUTTON.PRO]----------

```
;+
; CW_TBUTTON
; Compound widget for toggle (state cycling) buttons.
;
; Usage:
; bid = cw_tbutton(base, < keys >)
;
; Return Value:
; bid long The widget id of the button (actually a
;   containing base)
;
; Argument:
; base long input The id of the base widget containing
;    the button.
;
; Keywords (all input):
; frame ? If set, then draw a frame round the button.
; font string The font to use for the button text.
; group_leader long The ID of the group leader for the button.
; uvalue ??? The UVALUE to associate with the compound widget.
```

```
; value string List of values to associate with the button
; state int Initial state of the button.
;
; History:
; Original: 25/7/95; SJT
;-

function Cw_tb_getv, id, value

; Extract the state structure and return the state setting.

basid = widget_info(id, /child)
widget_control, basid, get_uvalue = uvs

return, uvs.state

end

pro Cw_tb_setv, id, value

; Extract the state structure

basid = widget_info(id, /child)
widget_control, basid, get_uvalue = uvs, /no_copy

; Unmap all sub bases

for j = 0, n_elements(uvs.basids)-1 do  $
  widget_control, uvs.basids(j), map = 0

; Reset state and map the relevant subbase.

uvs.state = value mod n_elements(uvs.bids)
widget_control, uvs.basids(uvs.state), map = 1

widget_control, basid, set_uvalue = uvs, /no_copy

end

function Cw_tb_evf, event

; Extract the state structure.

basid = widget_info(event.handler, /child)
widget_control, basid, get_uvalue = uvs, /no_copy

; update the state and map the appropriate button.
```

```
widget_control, uvs.basids(uvs.state), map = 0
uvs.state = (uvs.state + 1) mod n_elements(uvs.bids)
widget_control, uvs.basids(uvs.state), map = 1

; Pass on a WIDGET_BUTTON structure with the state in the SELECT
; field.

rev = {widget_button, event.handler, event.top, event.handler, uvs.state}

widget_control, basid, set_uvalue = uvs, /no_copy

return, rev

end


function Cw_tbutton, parent, $
                     frame=frame, $
                     font=font, $
                     group_leader=group_leader, $
                     uvalue=uvalue, $
                     value=value, $
                     state=state

; Define undefined keyword values.

if (n_params() eq 0) then parent = widget_base()
if (n_elements(state) eq 0) then state = 0l
if (n_elements(group_leader) eq 0) then group_leader = 0l
if (n_elements(frame) eq 0) then frame = 0
if (n_elements(font) eq 0) then font = ''

; This is the base that will be seen by the outside user as the
; CW id.

base = widget_base(parent, frame = frame)

; One subbase and one button per possible state. Need the bases
; because mapping buttons doesn't work properly.

bid = lonarr(n_elements(value))
basid = lonarr(n_elements(value))

; define the bsaes and buttons

for j = 0, n_elements(value)-1 do begin
   basid(j) = widget_base(base)
   bid(j) = widget_button(basid(j), $
```

```
                    value = value(j), $
                    font = font)
endfor

; Structure to store the state of the compound. Just need the
; IDs of the buttons and their sub-bases (strictly only need one
; of these as WIDGET_INFO could get one from the other but it's
; easier this way). Will be made into the UVALUE of the subbase
; holding the first state.

uvs = { bids: bid, $
      Basids: basid, $
      State:  0l}

widget_control, basid(0), set_uvalue = uvs

; Set attribute of the user-visible base.

widget_control, base, set_uvalue = uvalue, group_leader = group_leader, $
  event_func = 'cw_tb_evf', func_get_value = 'cw_tb_getv', $
  pro_set_value = 'cw_tb_setv'

; Finally set the state of the compound widget.

widget_control, base, set_value = state

return, base

end

-------[END CW_TBUTTON.PRO]-------------
```


```
--
 +-----------------------+------------------------------- --+---------+
| James Tappin,          | School of Physics & Space Research |  O__    |
| sjt@star.sr.bham.ac.uk | University of Birmingham       | --  \/` |
| Ph: 0121-414-6462. Fax: 0121-414-3722         |  |
 +-------------------------------------------------------- --+---------+
```

Subject: Re: Compound widget problem
Posted by sjt on Thu, 24 Aug 1995 07:00:00 GMT

View Forum Message <> Reply to Message

I'll post this reply to my own question in case it's gotten anyone worried.

Problem solved, by changing the WIDGET_BUTTON structure to a
WIDGET_TBUTTON (with exactly the same tags!) goodness only knows why that

works but it does.

--

+----------------------+------------------------------- --+---------+
James Tappin,	School of Physics & Space Research	O__
sjt@star.sr.bham.ac.uk	University of Birmingham	-- \/`
Ph: 0121-414-6462. Fax: 0121-414-3722	
+--- --+---------+