Subject: Tip for using Compound Widgets
Posted by MKatz843 on Sat, 03 Aug 2002 19:03:19 GMT
View Forum Message <> Reply to Message

I do a lot of widget programming in IDL, and I've recently come across
an elegant solution I'd like to share. Others have probably already
thought of this, or may have a better idea, so I hope I'll start a
dialog.

Suppose you have a complex compound widget with lots of differnt
functions. You want to tell it to do all kinds of things, but you
don't want to lose that "black box" aspect that makes cw_widgets so
powerful.

So for each differnt thing I want the widget to do, I encode the
command in the name of a structure and then pass that as a SET_VALUE
in a WIDGET_CONTROL as follows:

WIDGET_CONTROL, cw_ID, SET_VALUE={COMMAND_TYPE, arg1:val1, arg2:val2,
. . . }

so it might look like one of these:

WIDGET_CONTROL, cw_ID, SET_VALUE={NEW_IMAGE, img_ptr:ptr_new(image)}
WIDGET_CONTROL, cw_ID, SET_VALUE={SET_RANGE, range:[0, 16383]}
WIDGET_CONTROL, cw_ID, SET_VALUE={REFRESH_DISPLAY,
VIEWPLANE_RECT=[0,0,10,20]}

In order to make this work, a few things are required. Here are some
details.

1) The compound widget function is defined with a SET_VALUE routine.
That is, the first base you declare has a SET_VALUE explicitly set to
a routine that you're using to interpret these commands.

2) I use the IDL-recommended trick of storing a state variable
(generally a big structure with widget IDs and object pointers to
everything that needs to be changed) in the UVALUE of the first child
widget of the main base. I created a function and a procedure to get
and set the first child's UVALUE in one command (see below).
IMPORTANT NOTE: If you use this method, don't forget that every time
you update the state variable, you have to *save it* in the UVALUE of
the base's first child. Otherwise changes could be lost. So every call
of
"state = first_child_uvalue(id)" should be followed by a
"first_child_set_uvalue, base, state".

3) The compound widget's set_value procedure may look like this

```
;-----------
; Procedure to execute commands sent by SET_VALUE keyword to
WIDGET_CONTROL
;
pro cw_test_set_value, base, arg
  state = first_child_uvalue(base, /NO_COPY)  ;--- retrieve state
variable. Use NO_COPY for speed.

  if size(arg, /type) NE 8 then return  ;--- verify that it's a
structure
  case struct_name(arg) of
   'NEW_IMAGE': begin
      . . .
     end
   'SET_RANGE': begin
      . . .
     end
   'REFRESH_DISPLAY': begin
      . . .
     end
   else: print, 'Unknown command in cw_test_set_value'
  endcase

  first_child_set_uvalue, base, state, /NO_COPY ;--- store state
variable. Very Important!

return
end
```

4) The following functions are convenient to use for the above.

```
;-----------
; Return the name of a structure variable
;
function struct_name, a
return, (size(a, /type) EQ 8) ? tag_names(a, /STRUCTURE_NAME) : ''
end

;-----------
; Return the contents of the UVALUE of the first child of base.
;
function first_child_uvalue, base, NO_COPY=NO_COPY
  if not widget_info(base, /VALID_ID) then return, 0
  first_child = widget_info(base, /CHILD)
  if not widget_info(first_child, /VALID_ID) then return, 0
  widget_control, first_child, GET_UVALUE=uval, NO_COPY=NO_COPY
return, uval
```

```
end

;-----------
; Set the contents of the UVALUE of the first child of base.
;
pro first_child_set_uvalue, base, uval, NO_COPY=NO_COPY
  if not widget_info(base, /VALID_ID) then return
  first_child = widget_info(base, /CHILD)
  if not widget_info(first_child, /VALID_ID) then return
  widget_control, first_child, SET_UVALUE=uval, NO_COPY=NO_COPY
end
```

I'd be interested to know if anyone else uses tricks like this, or has
a better way,

M. Katz