Subject: widget objects
Posted by mmiller3 on Tue, 30 Mar 2004 20:56:25 GMT
View Forum Message <> Reply to Message

I've got some questions regarding objects and widgets and am
hoping that someone here can help point me toward a robust
design. Here's the situation...

I have a collection of IDL objects that include GUI components.
I use these in two ways: by themselves in their own top level
windows and swallowed in widget_bases of other GUIS, like a
compound widget. This works pretty well for me, but has some
limitations. For example, I have a color map object that allows
me to select a color map and adjust the limits. It inherits from
a publisher class so other objects that might use it can
subscribe. When the color map is changed, it tells its
subscribers. Typically a subscriber will redraw something using
the color map when it is notified of a change.

Now I've decided that in some cases, it's is ok to have a color
map GUI on the screen (as part of a main window or in its own
window) all the time. In other cases, I'd really like to put the
color map in its own window which can be drawn and withdrawn at
will, like with tk's withdrawn window state. I don't believe
that this is possible with IDL, so instead I thought I'd separate
the color map object from the gui so that an instance can create
and activate the gui and destroy it at will (at the programmers
will, that is) without destroying the entire object.

This sounds fairly straight forward, but I'm confident that I can
make it complex enough to be cumbersome. I wonder if any of yous
idlers have given this sort of thing some thought and have ideas
that you'd be willing to share or just discuss in general. The
part in particular that I'm wondering about is the capability of
putting a gui component in it's own top level base or in another
window. I've implemented this with a widget object class which
is appended below. Using it just seems a bit cumbersome to me
and I can't quiet put my finger on why.

Mike

P.S. This uses the generic event handler from
http://www.rsinc.com/codebank/search.asp?FID=209

--
Michael A. Miller   mmiller3@iupui.edu
  Imaging Sciences, Department of Radiology, IU School of Medicine

```
 ;==============================================================
===============
function example1
;; Add a wobject to an existing top level base widget:

top = widget_base(title='existing top level')

wobj = obj_new('wobject', top=top, /frame, /row)
wobj->create_widgets
wobj->xmanager

widget_control, top, /realize

return, wobj

end

 ;==============================================================
===============
function example2
;; Make a wobject with it's own top level base widget:

wobj = obj_new('wobject', /frame, /row, title='xyz')
wobj->create_widgets
wobj->xmanager

widget_control, wobj->get_top(), /realize

return, wobj

end

 ;==============================================================
===============
function wobject::init, top=top, group=group, _extra=extra_keywords

self.top = -1L
self.base = -1L
self.top_level = 0

result = 0

;; If I've been passed a top level widget and a group, use them:
if ( n_elements(top) eq 1 ) $
  and ( n_elements(group) eq 1 ) then begin
    self.top = top
    self.base = widget_base(self.top,  group=group, _extra=extra_keywords)
    result = 1
endif

;; If I've been passed a top level widget and not a group, use just
;; the top:
if ( n_elements(top) eq 1 ) $
  and not ( n_elements(group) eq 1 ) then begin
    self.top = top
    self.base = widget_base(self.top, _extra=extra_keywords)
    result = 1
endif

;; If I've been passed a group widget, but not a top, create a top and
;; use the group:
if ( not ( n_elements(top) eq 1 ) ) $
  and ( n_elements(group) eq 1 ) then begin
    self.top = widget_base(group=group, _extra=extra_keywords)
    self.base = self.top
    self.top_level = 1
    result = 1
endif

;; If I haven't been passed anything, make a top without a group:
if ( not ( n_elements(top) eq 1 ) ) $
  and ( not ( n_elements(group) eq 1 ) ) then begin
    self.top = widget_base(_extra=extra_keywords)
    self.base = self.top
    self.top_level = 1
    result = 1
endif

return, result
end

 ;==============================================================
===============
pro wobject::create_widgets
;; example code stub - overload with your own complete method!

button = widget_button(self.base, value='ok', uname='ok')
button = widget_button(self.base, value='cancel', uname='cancel')
button = widget_button(self.base, value='close', uname='close')

end

 ;==============================================================
===============
pro wobject::xmanager, _extra=extra_keywords

widget_control, self.base, set_uvalue=self, event_pro='generic_class_event'
xmanager, 'generic_class', self.base, /no_block

end

 ;============================================================
===============
pro wobject::close
if self.top_level then begin
   widget_control, self.top, /destroy
endif
end

 ;============================================================
===============
pro wobject::cleanup
self->close
end

 ;============================================================
===============
function wobject::get_top
return, self.top
end

 ;============================================================
===============
function wobject::get_base
return, self.base
end

 ;============================================================
===============
pro wobject::event, event
;; example code stub - overload with your own complete method!

case event.id of

   widget_info(event.top, find_by_uname='close'): self->close

   else: begin
      print, 'event from: ', event.id, ' (', widget_info(event.id, /uname), ')'
      help, event, /structure
   end

endcase
```

```
end

 ;==============================================================
===============
pro wobject__define
struct = { wobject, $
        top: 0L, $          ; top level widget id
        base: 0L, $          ; top level base widget id.  Create
                        ; widgets in this base to have this
                        ; wobject's event handler handle their
                        ; events
        top_level: 0 $       ; flag telling if this is a top level
                        ; wobject with its own window or not
      }
end
 ;==============================================================
===============
```