Subject: Re: base widgets growing uncontrollably.... ?
Posted by John-David T. Smith on Fri, 27 Jul 2001 20:59:05 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Paul van Delst writes:
>
>> Each separate compound widget function realises *and* registers the widget using
>> XMANAGER. I did this so that when I killed the top level base, the cleanup
>> routines for all the child compound widgets would be called. If I don't do a
>>
>> XMANAGER, id, 'widget_name', /JUST_REG, CLEANUP = 'widget_name_cleanup'
>>
>> in the compound widget creation functions, then I am left with a bunch of
>> dangling pointers - that used to be in the compound widget's top-level base user
>> value - hanging about afterwards. The XMANAGER call in *each* compound widget
>> creation function was the only way I could get stuff cleaned up in a
>> heirarchial-type of way. I want to keep the information structure for each
>> compound widget separate in it's own top-level base user value (rather than
>> shoving everything in the god-base uvalue) as I envisage these routines to be
>> usable on their own, not just as a component of a container GUI.
>>
>> If anyone has a better method of doing this please let me know. I couldn't
>> figure out how to make the child widget cleanup routines (for the compound
>> widgets) "visible" unless I put in separate XMANAGER calls.
>
> I only ever have a single XMANAGER command in a widget
> program. But I almost always have compound widgets
> (and almost all of these are compound widget objects
> these days). The way I clean these widget objects up
> is by using a KILL_NOTIFY on the compound widget's
> top-level base. This is allowed, because these are
> not really top-level bases, of course, and are not
> directly managed by XMANAGER. I always store the
> object reference in some easy-to-locate uvalue in
> the object widget, so it is easy to find the object
> reference and destroy it. This cleans everything up
> properly.
>

And to point out the obvious, there's no reason you can't make compound
widgets also objects, rather than having an all-in-one object widget
design. You might then have a larger object interface which "composits"
(i.e. includes) the sub-objects directly, perhaps creating them itself.

Then, cleanup a simple matter of putting in place the relevant "Cleanup"
methods, and cleaning up your composited objects in the master Cleanup

(i.e. "self.fancycompoundobj1->Cleanup"). You can also allow a single routine to do double duty as a master kill notify and the event<->object interface (for those like me who cringe at littering the otherwise pristine namespace with non-methods):

```
widget_control, base,set_uvalue=self,KILL_NOTIFY="class_event",/REALIZE
XManager,'class',base,/NO_BLOCK
```

Then the event callback looks like:

```
;; Pass on events *AND* serve as a kill notify (destroy the object)
pro class_event, ev_or_id
  if size(ev_or_id,/TYPE) ne 8 then begin
     widget_control, ev_or_id, get_uvalue=self
     obj_destroy,self
     return
  endif
  widget_control,ev_or_id.top,get_uvalue=self
  self->Event,ev_or_id
end
```

JD