Subject: Re: Object Widgets
Posted by J.D. Smith on Wed, 10 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Struan Gray wrote:
>
> Mark Hadfield, m.hadfield@niwa.cri.nz writes:
>
>>> http://www.sljus.lu.se/stm/IDL/Obj_Widgets/
>>
>> Interesting stuff, Struan! I note that your widget
>> template object, SLFoWid, does not create a top-level base.
>> This seems to me to be an obvious thing to do, since all
>> widget hierarchies require a TLB. What were the reasons for
>> your choice? Do you expect SLFoWid to be used for widgets
>> that aren't at the top level?
>
> That's the main idea. I want SLFoWid to define behaviour, not
> layout, and I would like to use it for compound widgets. This is why
> I have a seperate SLFOWID::XMANAGE method. I also found that putting
> all the widget creation statements into one INIT method makes it
> easier to maintain the code, and avoids having to pass around all the
> possible keywords to WIDGET_BASE. The web pages will (eventually)
> contain some discussion of my design decisions.
>
> I think my news server is behind the times a bit, as I got an
> email from you containing the following good points (I've changed
> their order). If you didn't post them as well, sorry for the breach
> of netiquette.
>
```
>>   2. Similarly it is a good idea in the object cleanup
>> routine to check if the widget hierarchy is still valid and,
>> if it is, destroy it, i.e.
>>
>> pro SLFoWid::Cleanup
>>
>>   print, 'SLFoWid::Cleanup'
>>   print, '  widget ID: ', self.myWidID
>>   print, '  object ID: ', self
>>
>>   if widget_info(self.myWidID, /VALID_ID) then $
>>      widget_control, self.myWidID, /DESTROY
>>
>> end   ; pro SLFoWid::Cleanup
```
>
> Strangely enough, despite promoting widgets-as-objects I had
> assumed that the widget would always be killed with WIDGET_CONTROL -
> which is daft I admit. I wanted to avoid getting stuck in an infinite

```
>  loop, with the widget and object cleanup routines calling each other
>  over and over.  One way to avoid that is for the object cleanup to use
>  WIDGET_INFO to see if self.myWidID is being managed, but that assumes
>  the Xmanager is being used.  Another is for the object and widget
>  cleanup routines to use keywords when they call each other, but that
>  seems inelegant.
>
>     After a little digging it seems that both WIDGET_CONTROL, /DESTROY
>  and OBJ_DESTROY take care to avoid recursion - ie, they seem to know
>  that things are in the process of being destroyed, despite the fact
>  that both the widget and the object IDs are still valid in their own
>  cleanup routines.  (note that your 'if widget_info...' is redundent,
>  since self.myWidID is valid until all the cleanup is done).   Since
>  WIDGET_CONTROL and OBJ_DESTROY seem to ignore demands to kill
>  widgets/objects that they're already in the process of killing, I've
>  just added a WIDGET_CONTROL, /DESTROY line to the object cleanup
>  method.
>
>     If RSI allows us to specify an object method as the cleanup
>  routine for a widget this problem will disappear.
>
>>    1. In your widget cleanup routine it is a good idea to
>>  check that the object is valid before trying to destroy it,
>>  i.e.
>>
>>  pro SLFoWid_Cleanup, myID
>>
>>    widget_control, myID, get_uvalue=myObjRef
>>    if obj_valid(myObjRef) then begin
>>      myObjRef -> GetProperty, no_block=no_block
>>      if no_block eq 1 then obj_destroy, myObjRef $
>>        else myObjRef -> cleanup
>>    endif
>>
>>  end   ; pro SLFoWid_Cleanup
>>
>>    This is advisable because it is possible for the object
>>  to have been destroyed (by OBJ_DESTROY, or HEAP_GC) behind
>>  XMANAGER's back.
>
>     As I said above, I was assuming that the widget would always be
>  destroyed as a widget, not as an object.  That said, even after the
>  change to the object cleanup routine above, there is no need to check
>  the object reference for validity since the only way you can end up in
>  an object or widget cleanup routine is when it dies, and at that point
>  the widget ID and object ID must still be valid.  I've left your check
>  in all the same: an inattentive programmer may have stored something
>  else in the uvalue and lost the object reference.  The check would
> still leave a dangling object reference, but at least the program
> won't crash.
>
>> 3. There is a problem in heap cleanup for blocking widgets.
>> If I create anew widget with
>>
>>     o = obj_new('SLFow_minimal')
>>
>> then hit the quit button, the heap is cleaned up. But if
>> I do the same with
>>
>>     o = obj_new('SLFow_minimal', /BLOCK)
>>
>> then the 'SLFow_minimal' object is left on the heap
>
>     This one I did catch myself (but hadn't updated the web version).
> I saw your later post, which uses the same solution I have, though I
> use this line:
>
>     return, self.no_block
>
>     at the end of all the INIT methods of subclassed widgets.  In
> general, I wanted to minimise the amount of stuff a subclass writer
> would have to remember to include in their methods, but this seems
> essential (as is getting the _ref_extra and _extra keywords right).
>
>     Note that obj_destroy called on a blocking widget (from another
> widget, or from a 'quit' button) will always produce an error, so it
> is safest to use widget_control, /destroy if you don't know beforehand
> whether a widget will be used in blocking or non-blocking mode.
> Actually, I can see few reasons to use blocking widgets, but since I
> thought I could handle them it seemed churlish to force widgets to be
> non-blocking.
```

I have a similar though less tightly coupled object widget superclass.  One suggestion I had would be to register the widget under the name:

obj_class(self)+self.title

or some such, in order to allow different instances of the same class to run at once. Another suggestion involves blocking vs. non-blocking widgets.  I find it convenient to make a separate "Start" method in addition to Init.  Init is used to set up data, etc., and Start is used to actually run XManager, etc.  Though this forces the user to remember on more thing, it's actually useful in other contexts as well... especially if several widgets programs which will be interacting are being created.  It has a similar flavor to XManager's JUST_REGISTER keyword.

JD

--

```
J.D. Smith                          |*|      WORK: (607) 255-5842
Cornell University Dept. of Astronomy  |*|           (607) 255-6263
304 Space Sciences Bldg.            |*|       FAX: (607) 255-5875
Ithaca, NY 14853                    |*|
```