Subject: base widgets growing uncontrollably.... ?
Posted by Paul van Delst on Wed, 25 Jul 2001 15:10:46 GMT
View Forum Message <> Reply to Message

Hi,

I'm seeing a weird effect with an IDL widget app I put together and I'm hoping someone here might recognise the symptoms and inform me of some widget_base keyword I have forgotten.

I create a base within which I create several compound widgets (that I wrote) each, of course, within it's own base. Currently I have three compund widgets in the base (all of which contains exclusive button bases); two with column=1 and the third with grid_layout=1 and column=3. Now, when I add more buttons to the third compound widget (currently I have three columns of five buttons), the size of the other compound widgets grow! It's at the point now where half of the space in the widget display is just empty, dead space. If I remove items from the third compund widget (which is always of a size such there is no empty space) the dead space in the other two shrinks.

Anyone have an idea what I'm doing wrong and how to fix it? I would rather not have to use XSIZE/YSIZE/etc keywords.

Thanks for any info/insights.

paulv

p.s. :
IDL> print, !version
{ x86 linux unix 5.4.1 Jan 16 2001 32 32}

and my linux is RH 6.1

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
Alexander Pope.

Subject: Re: base widgets growing uncontrollably.... ?
Posted by Paul van Delst on Fri, 27 Jul 2001 14:48:32 GMT
View Forum Message <> Reply to Message

Marc Schellens wrote:
>
>> WOOHOO! That did it! Inserting the UPDATE in the following code fixed the problem.
>>

>> You're a star! This problem was annoying the crap outta me to the point of tossing and
>> turning at night.
>>
>> Onya!
>>
>> paulv
>
> Thanks,
> I have to pass the compliment to Stein Vidar Hagfors Haugan,
> actually I think I got it long time ago from Davids page Alex
> mentioned below (but didn't remeber from where I had it).
>
> Anyway, good that it helped you!
>
> But was astonishes me is, that you use the update BEFORE
> the base is realized.
> I didn't know that update makes a difference then.
>
> What I do not understand in your program is,
> why you first unmap the base during creation
> and then map it when realizing?
> Until a base is realized (= brough to screen) mapping didn't matter
> (at least for the tlb).

I unmap the top level base right at the start so I don't see the final GUI being created widget by widget in front of me - it was interesting to watch but didn't look very nice. Unmapping then mapping at the end makes the complete widget just pop up when it has finished being created/assembled.

> OR: Might it be that in the real program the realizing is done before
> and
> so the realizing there is unnecessary (which would also explain the
> astonishing effect of update to non realized widgets)?

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.

Maybe I should just stick everything in the god-base id..... I gotta sit down
and think about this a bit more....

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
Alexander Pope.

Subject: Re: base widgets growing uncontrollably.... ?
Posted by david[2] on Fri, 27 Jul 2001 19:24:36 GMT
View Forum Message <> Reply to Message

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.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: base widgets growing uncontrollably.... ?
Posted by Paul van Delst on Fri, 27 Jul 2001 19:41:17 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> 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.

Cool bananas! That exactly the keyword thingo I was looking for. I took out the realisation and xmanager registration from the compound widget routines and everything went correctly into the dustbin at end of program. Thanks.

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;

Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
Alexander Pope.

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

Subject: Re: base widgets growing uncontrollably.... ?
Posted by Paul van Delst on Fri, 27 Jul 2001 21:21:06 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
>
> 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

I will have to trust your much greater expertise on these matters since I'm still in the group (probably the minority nowadays) that has no idea how to even start using objects in IDL (I did give it a shot when it first came out in IDL 5.0(?)). When I see stuff like "self->Event,ev_or_id" the eyes glaze over and I take a break from work and check stock quotes or a news site. :o)

Procedurally yours,

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
Alexander Pope.

Subject: Re: base widgets growing uncontrollably.... ?
Posted by david[2] on Fri, 27 Jul 2001 22:04:46 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:
>
> I will have to trust your much greater expertise on these matters since I'm
> still in the group (probably the minority nowadays) that has no idea how to even
> start using objects in IDL (I did give it a shot when it first came out in IDL
> 5.0(?)). When I see stuff like "self->Event,ev_or_id" the eyes glaze over and I
> take a break from work and check stock quotes or a news site. :o)

Thank goodness! I thought I was the only one!

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: base widgets growing uncontrollably.... ?
Posted by John-David T. Smith on Fri, 27 Jul 2001 23:49:29 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
>
> JD Smith wrote:
>>
>> 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
>
> I will have to trust your much greater expertise on these matters since I'm
> still in the group (probably the minority nowadays) that has no idea how to even

> start using objects in IDL (I did give it a shot when it first came out in IDL
> 5.0(?)). When I see stuff like "self->Event,ev_or_id" the eyes glaze over and I
> take a break from work and check stock quotes or a news site. :o)
>
> Procedurally yours,

Just think how little restful break-time you'd get in a day if not for
overly compact programmatic mechanisms.

Objectingly(ively?) yours,

JD

P.S. My one line object tutorial:

objects->DoThings, withstuff, LIKE_STRUCTURES=only_smarter

Subject: Re: base widgets growing uncontrollably.... ?
Posted by Jim Pendleton on Sat, 28 Jul 2001 02:57:25 GMT
View Forum Message <> Reply to Message

"JD Smith" <jdsmith@astro.cornell.edu> wrote in message
news:3B61FE09.765EE529@astro.cornell.edu...

[...stuff]

> Paul van Delst wrote:

[...more stuff]
>>
>> JD Smith wrote:
>>>
>>> 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.
>>>
>
>
> Just think how little restful break-time you'd get in a day if not for
> overly compact programmatic mechanisms.
>
> Objectingly(ively?) yours,

>
> JD
>
> P.S. My one line object tutorial:
>
> objects->DoThings, withstuff, LIKE_STRUCTURES=only_smarter

If you're writing applications of more than a couple hundred lines and you're not using an object-based event handler described by JD earlier, you're missing the revolution.

And if you're writing class functionality that *relies on* the IDL widget system, you're missing the next revolution. A GUI is no more than an accessor to class data and methods. With sufficient forethought in design, IDL classes should be able to interact with Ion, VB, C++, perl, python or any other front-end.

Think outside the widgets.

Jim P.

Subject: Re: base widgets growing uncontrollably.... ?
Posted by Paul van Delst on Mon, 30 Jul 2001 13:44:50 GMT
View Forum Message <> Reply to Message

Jim Pendleton wrote:
>
> "JD Smith" <jdsmith@astro.cornell.edu> wrote in message
> news:3B61FE09.765EE529@astro.cornell.edu...
>
> [...stuff]
>
>> Paul van Delst wrote:
>
> [...more stuff]
>>>
>>> JD Smith wrote:
>>>>
>>>> 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.
>>>>
>>

>>
>> Just think how little restful break-time you'd get in a day if not for
>> overly compact programmatic mechanisms.
>>
>> Objectingly(ively?) yours,
>>
>> JD
>>
>> P.S. My one line object tutorial:
>>
>> objects->DoThings, withstuff, LIKE_STRUCTURES=only_smarter

I liked the one line tutorial - it actually made things clearer. Seriously. :o)

> If you're writing applications of more than a couple hundred lines and
> you're not using an object-based event handler described by JD earlier,
> you're missing the revolution.

Uh-oh. Who will be the first against the wall when the revolution comes? :o)

> And if you're writing class functionality that *relies on* the
> IDL widget system, you're missing the next revolution. A GUI is no
> more than an accessor to class data and methods. With sufficient
> forethought in design, IDL classes should be able to interact with Ion, VB,
> C++, perl, python or any other front-end.

I have no doubt all this stuff is very important. But in context. My interest is
to look at my data. That's pretty much it. I slapped together my widget program
(which was significantly less than a couple hundred lines) solely because I was
tired of backspacing and changing an structure array index on the IDL command
line to look at a different variable; that is to say, my class functionality
consists of PLOT and SHADE_SURF. For what I do, IDL is simply an easy to use
(most of the time!), relatively non-specialised tool to let me visualise my data
quickly, and in a variety of ways, so I can glean as much info/insight from it
as I can.

> Think outside the widgets.

If it's any consolation, I apply object based design principles as much as I can
to my software written in my workhorse language, Fortran 90. :o)

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.

Alexander Pope.

Subject: Re: base widgets growing uncontrollably.... ?
Posted by Stein Vidar Hagfors H[1] on Mon, 30 Jul 2001 20:42:06 GMT
View Forum Message <> Reply to Message

Paul van Delst <paul.vandelst@noaa.gov> writes:

[..]
> When I see stuff like
> "self->Event,ev_or_id" the eyes glaze over and I take a break from
> work and check stock quotes or a news site. :o)

Whereas I kick myself for not having time to pursue the beauty of it
all :-\

--
---------- ----------
Stein Vidar Hagfors Haugan
ESA SOHO SOC/European Space Agency Science Operations Coordinator for SOHO

NASA Goddard Space Flight Center, Email: shaugan@esa.nascom.nasa.gov
Mail Code 682.3, Bld. 26, Room G-1, Tel.: 1-301-286-9028/240-354-6066
Greenbelt, Maryland 20771, USA. Fax: 1-301-286-0264
---------- ----------

Subject: Re: base widgets growing uncontrollably.... ?
Posted by Martin Schultz on Wed, 01 Aug 2001 12:38:30 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@astro.cornell.edu> writes:

> 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.

That's exactly the line I am pursuing with my base GUI object and its children.
Unfortunately, progress on this is extremely slow these days, but if you are
interested to get a snapshot of what is available, I will put these routines
on our ftp server...

Wrt procedures vs. objects: I think that objects are very "natural" in the
sense that you can almost "speak" object oriented statements. A line like

    container->Add, myobject
can be worded as "Add myobject to the container (object)",
    graph->SetProperty, color='blue', thick=2
can be communicated as "Set the following properties of the graph (object): color, thick(ness)"

So, for some lisp guru, it may even be possible to write a speech interface that will produce object programs out of everyday conversations ;-) But I realized myself that it is very hard for a person who has been trained for years to forget about the sloppiness of human languages and strictly adhere to procedural thinking to get over this and "relax" and think more in everyday terms such as tasks that need to be fulfilled (and then you need to be strict too ;-( ).

Cheers,

Martin

--
[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[