Subject: Re: destroying graphics objects
Posted by Brad Gom on Tue, 04 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Brad Gom wrote:

> As an aside, did IDL 5.0 have a bug with the
> IDLgrModel->get(isa=object_class_name) method? When I use this to return an
> object reference for something in a model, it returns the wrong object in IDL
> 5.0, but seems to work in version 5.3.
>

On closer examination of the Help file, I see that the ISA keyword is ignored if the ALL keyword is not also set -which means that the get method will just return the first object it finds. If it returned nothing, or gave an error, then I would have caught this sooner!

Subject: Re: destroying graphics objects
Posted by Brad Gom on Tue, 04 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> Brad Gom (b_gom@hotmail.com) writes:
>
>> I'm having a little conceptual difficulty with managing graphics
>> objects. I am making a plot object which can contain an arbitrary number
>> of IDLgrPlot objects. I need to be able to add and remove them from the
>> plot at will (and keep track of them in my code). Since this is going
>> to be an object-widget, the less things to keep track of in the code the
>> better.
>
> Sounds like a job for a LIST object. :-)
> You can find one on my web page, if you are interested.

I have a 'stack' object that would also do the trick, but I'd rather properly use the features of the IDL objects..

>> My question is this: Is is better to store all the IDLgrPlot objects in
>> an IDLContainer object for ease of access, or to just search for them in
>> the IDLgrModel that they are linked to?
>
> A model *is* a container object. That is, a model has inherited
> the CONTAINER object. So anything you add to a model is automatically
> destroyed when the model is destroyed. I'd search for the object in
> the model.

As an aside, did IDL 5.0 have a bug with the

IDLgrModel->get(isa=object_class_name)  method? When I use this to return an object reference for something in a model, it returns the wrong object in IDL 5.0, but seems to work in version 5.3.

>>  Should I remove an object from a model
>>  before I destroy the object, or does it matter?
>
>  I haven't tested this, but I can imagine that a model would have
>  a great deal of difficulty rendering something that was no longer
>  in existence. Knowing what I do about computer languages, I would
>  probably be willing to place a rather largish wager than it *would*
>  make a difference. I'd remove any object from the model before
>  I destroyed it.

I know this is better form, but it would simplify things for me if I just had a conainer for just the plot objects and nothing else, and delete the container when necessary without worrying about what else is in the model.

With attached code, it looks like the model is rendered properly and returns the right valid objects when I delete objects without previously removing them from the model.  So long as there are no hidden problems or memory leaks with this technique, then I'll use it.

Thanks,

Brad

## File Attachments

1) objtest.pro, downloaded 82 times

---

Subject: Re: destroying graphics objects
Posted by davidf on Tue, 04 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Brad Gom (b_gom@hotmail.com) writes:

>  I'm having a little conceptual difficulty with managing graphics
>  objects. I am making a plot object which can contain an arbitrary number
>  of IDLgrPlot objects. I need to be able to add and remove them from the
>  plot at will (and keep track of them in my code).  Since this is going
>  to be an object-widget, the less things to keep track of in the code the
>  better.

Sounds like a job for a LIST object. :-)
You can find one on my web page, if you are interested.

> My question is this:  Is is better to store all the IDLgrPlot objects in
> an IDLContainer object for ease of access, or to just search for them in
> the IDLgrModel that they are linked to?

A model *is* a container object. That is, a model has inherited
the CONTAINER object. So anything you add to a model is automatically
destroyed when the model is destroyed. I'd search for the object in
the model.

> If I keep them in a container,
> and then destroy the container, do I have to let the IDLgrModel know
> that I destroyed those objects? Should I remove an object from a model
> before I destroy the object, or does it matter?

I haven't tested this, but I can imagine that a model would have
a great deal of difficulty rendering something that was no longer
in existence. Knowing what I do about computer languages, I would
probably be willing to place a rather largish wager than it *would*
make a difference. I'd remove any object from the model before
I destroyed it.

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: destroying graphics objects
Posted by Struan Gray on Wed, 05 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Brad Gom, b_gom@hotmail.com writes:

> I know this is better form, but it would simplify
> things for me if I just had a conainer for just
> the plot objects and nothing else, and delete the
> container when necessary without worrying about
> what else is in the model.

There are always several ways of doing things, but if the combined
plot is not a useful object in its own right (i.e. if it doesn't
represent some kind of synthesis from the data) I would prefer to keep

control of the individual plots outside the pure display routines.

The simplest way to do this is to put all the individual plots into an IDLgrModel, which as someone said, is subclassed from the container object. This will let you adjust the scaling position and orientation of all the plots at once, which is useful when changing the display window or when printing.

The trick is to add this model to a second model using the /alias keyword. Then you can pass the second model to the graphics hierarchy. Everything will plot correctly but when the graphics heirarchy is destroyed only the second model gets auto-destroyed, leaving the first model and all your plots intact in memory.

A final advantage is that you can add models as aliases to any number of other models, which allows you to have multiple views of the same data on screen. If, say, you want to adjust one of the plots, you can display it in a seperate graphics window for editing, and see the updates live in both the single plot and the multi-plot window.

Eventually you end up with a proliferation of views, scenes and printer objects all displaying one of your plots in various ways. Despite the complexity, you know exactly who 'owns' the plot and when it will and won't be auto-destroyed.

Struan

---