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