Subject: Re: Why are objects global?
Posted by Mark Hadfield on Wed, 19 Jul 2000 07:00:00 GMT
View Forum Message <> Reply to Message

<bjackel@phys.ucalgary.ca> wrote in message
news:3974A29E.78BE054C@phys.ucalgary.ca...
> Subject line says it all.
> ...
> Previously, variables in routines would be local unless
> 1) linked to a parameter or keyword
> 2) placed in a common block
> 3) attached to a pointer
>
> Why must objects be any different?
>
> I would argue that they should be treated the same as
> any other variables: cleaned up at the end of a procedure
> or function, unless one of the three conditions mentioned
> above is met. The current behaviour adds complexity to
> IDL, with no obvious advantages.

Well, I'd add another couple of items to your list:
4) linked to a widget hierarchy or graphics window
5) linked to another object that satisfies 1-4

But, anyway, what you're suggesting is known as garbage collection. It's used in a number of OO languages, e.g. Java, Python, Matlab. It's not without performance cost, because following all the paths by which an object might be linked to something that shouldn't be destroyed is a complex business. Simple garbage collectors tend to freeze processing of the rest of application from time to time. But it certainly can be done. RSI in their wisdom chose not to do it.

So far I haven't found it too onerous to handle object destruction manually. One thing I usually do is attach a "disposal" container to any long-lived object (e.g. a widget application) and put all the resources needed by the application in there. The disposal container is destroyed in the object's cleanup method and in turn destroys its contents.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield/
National Institute for Water and Atmospheric Research
PO Box 14-901, Wellington, New Zealand