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

bjackel@phys.ucalgary.ca wrote:
>
> Subject line says it all.
>
> I've been playing with objects (not object graphics) for a
> while now, and can see how they might be useful. However,
> having to call "obj_destroy" manually at the end of functions
> and routines has been the source of many headaches.
>
> For example, consider the procedure below. It creates
> a "vector3" object which is used for various useful
> calculations, but is of no further use outside "example".
> Once we exit "example", variables "a" and "b" are automatically
> cleaned up. However, "c" gets leaked, unless we explicitly
> destroy it.
>
> 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.
>
> Brian
>
>

ok. Consider the following: You create a container object somwhere
within a program,
then you call a subroutine to create some object which shall be stored
in the
container. Although that object is never passed into or out of the
routine, you
certainly want to keep it "alive":

; pseudo example

pro make_contents, theContainer

```
  ; theContainer is the object reference of the container object
  rectangle = obj_new('grSymbol',bounds=,[0.,0.,1.,1.])

  theContainer->Add(rectangle)
end

; main
theContainer = obj_new('IDL_Container')
make_contents,theContainer
help,theContainer->Get(index=0)
end
```

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   [[
[[[[[[[[[

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