Subject: Re: IDLgrLegend broken
Posted by Mark Hadfield on Thu, 07 Dec 2000 01:34:54 GMT
View Forum Message <> Reply to Message

A couple of corrrections/clarifications to my previous post:

> The simplest workaround is to call IDLgrLegend__Define *before* restoring
> your IDLgrLegend object (that is, of course, if you know you are about to
> restore one).

And the problem is that generally you don't know the class of all the
objects you are about to restore. But actually, it's not quite that bad. You
just have to call the __Define method *before calling any of the restored
object's methods*. And I think this can be done programmatically, as
follows:

1. When restoring a .sav file that might contain some objects, use the
RESTORE procedure's RESTORED_OBJECTS keyword to get a list of references to
the objects you have just restored. (In addition to the IDLgrLegend object
you want to restore, there will likely be IDLgrText, IDLgrPlot, etc objects
embedded in it.)

2. Immediately go through that list retrieving the class name for each
object and for each one call that class's __DEFINE method using
CALL_PROCEDURE. Surround this code with a catch block so that you don't trip
up on classes (like IDL built-in ones) that don't have a __DEFINE method.

3. Call methods on your objects to your heart's content.

At least I think that will usually work--I haven't tried it. there are
further problems that arise when saving & restoring objects, e.g.
parent-child relationships in graphics trees get broken.

And in regard to this one:

> It's a fundamental problem of IDL objects, deriving from the way methods
are
> resolved. It occurs because IDLgrLegend has a superclass (IDLgrModel)

The problem that Pavel reported *is* a fundamental problem of IDL objects,
but it would occur with classes that do not inherit from any superclass.
Let's say we restore an object of such a class (MyClass) and then call a
method (SomeMethod). Let's also assume that all the methods of MyClass are
in myclass__define.pro (as usual), and that MyClass__Define has never been
executed, so myclass__define.pro has never been compiled. What will IDL do?
It will look for SomeMethod amongst the functions that have been compiled
into memory, first for MyClass (doesn't find it) then for its superclasses
(there aren't any). Then it will look on disk for myclass__somemethod.pro.

But that doesn't exist. I don't think IDL is smart enough to look in
myclass__define.pro.

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