Subject: Re: Copying objects
Posted by J.D. Smith on Fri, 03 Dec 1999 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> J.D. Smith (jdsmith@astro.cornell.edu) writes:
>
>> The easiest way to copy objects in bulk, composited objects and all, is to save
>> and restore, using:
>>
>> save,obj,FILENAME=savefile
>> restore,savefile,RESTORED_OBJECTS=newobj,/RELAXED_STRUCTURE_ ASSIGNMENT
>> newobj=newobj[0]
>>
>> Don't forget that the object reference variable is also restored, and will
>> overwrite a variable with the same name in that level (but this can be used to
>> advantage -- see below). Also beware of later restoring objects, since their
>> methods (and those of their superclasses) will be unavailable, and will not be
>> located automatically. You can prevent this... see a posting from last year on
>> restoring objects (I can re-post my resolve_obj if necessary).
>
> The lightly edited conversation JD refers to on the
> newsgroup about restoring objects can be found here:
>
> http://www.dfanning.com/tips/saved_objects.html
>

Actually, David, your website posting is incomplete, because it does not mention the problem of superclass method resolution. I emailed you about this last year, but it must have slipped through the cracks. The basic problem is that superclass methods are not automatically searched and compiled by the above procedure, since a restored object contains implicitly in its definition the class definitions of all its superclasses. Basically it's the same problem cropping up again. The solution is to recursively work your way up the inheritance tree by hand. *But*, one more wrinkle: if you're often updating your class definition, you need to have it defined *before* restoring the object, to avoid having the older definition (as saved with the object) shadow your new one. A new resolve_obj (see attached), and a method which addresses all of these concerns is therefore:

```
resolve_obj,CLASS='class'
restore,file, RESTORED_OBJECTS=obj,/RELAXED_STRUCTURE_ASSIGNMENT
```

Notice that now we are burdened with knowing *which* class to pre-compile. However, if you are willing to live with the potential of older class definitions shadowing newer ones, or don't mind ensuring the most recent class version is defined before any object restoration occurs (though in practice that

would eliminate the problem), you can still use:

restore,file, RESTORED_OBJECTS=obj,/RELAXED_STRUCTURE_ASSIGNMENT
resolve_obj,obj[0]

without worrying about which class is being restored. The only difference here is that putting the restore first opens up the potential of class definition shadowing. Sometimes this isn't a concern, or the convenience and generality of a generic restoration scheme outweighs the precautions which must be taken when using it.

The updated routine is attached. Notice the use of routine_info() to avoid compiling any class already compiled. Also note that the original routine will work well enough for classes which do not INHERIT... but most of the good one do ;).

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|

```
pro resolve_obj,obj,CLASS=class,ROUTINE_INFO=ri
  if n_params() ne 0 then begin
    if NOT obj_valid(obj) then begin
      message,'Object is not valid.'
    endif
    class=obj_class(obj)
  endif

  if n_elements(ri) eq 0 then ri=routine_info()

  for i=0,n_elements(class)-1 do begin
    defpro=class[i]+'__DEFINE'
    if (where(ri eq defpro))[0] eq -1 then begin
      ;; Compile and define the class.
      call_procedure,defpro
    endif
    supers=obj_class(class[i],/SUPERCLASS,COUNT=cnt)
    if cnt gt 0 then resolve_obj,CLASS=supers,ROUTINE_INFO=ri
  endfor
end
```

File Attachments

1) resolve_obj.pro, downloaded 142 times