Subject: Re: Copying objects
Posted by J.D. Smith on Thu, 02 Dec 1999 08:00:00 GMT
View Forum Message <> Reply to Message

James Tappin wrote:
>
> Is there a "clean" way to make a copy of an object. The best I could do for a
> nice easy case with no pointers or object references in the class was:
>
> pro Object1::set_all, tstr
> for i = 0, n_tags(tstr)-1 do self.(i) = tstr.(i)
> end
>
> function Object1::copy
> temp = {object1}
> for i = 0, n_tags(temp)-1 do temp.(i) = self.(i)
> newobj = obj_new('object1')
> newobj -> set_all, temp
> return, newobj
> end
>
> While it works, it seems to be a bit of a kludge. Is there a better way?
>

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).

Other interesting applications of this method:

Objects which replace themselves from file. Don't forget to kill the replaced object:

```
oldself=self
; Be sure to do some error-catching in here!
restore,file,/RELAXED_STRUCTURE_ASSIGNMENT  ;self is overwritten from file!
;; Do some error catching, but if it worked...
```

obj_destroy,oldself

and viola! a new self-identity! Instant "Revert from disk" command. I use it all the time.

Another fun and useful technique: "disconnect" irrelevant portions of objects before saving them, especially those which will implicitly define structures and classes that might be changing alot, and which you don't want to keep up with the method resolution issue. Simply use pointers, and do something like:

```
uselesssav=self.UselessStructPtr
self.UselessStructPtr=ptr_new() ; save a null pointer instead!
self->Save
self.UselessStructPtr=uselesssav
```

etc. You get the idea. Be sure to disconnect only those things for which a null-pointer value isn't disasterous!

Good Luck,

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 |*|