Subject: Copying objects
Posted by James Tappin on Tue, 30 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

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?

James
--

```
 +----------------------+-----------------------------+-- -------+
| James Tappin          | School of Physics & Astronomy |  O__    |
| sjt@star.sr.bham.ac.uk | University of Birmingham     | --  \/` |
| Ph: 0121-414-6462. Fax: 0121-414-3722                |         |
 +-----------------------------------------------------+-- -------+
```


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

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

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

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 |*|

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

J.D. Smith (jdsmith@astro.cornell.edu) writes:

> 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.

EEEeeiiiigh! I've tried to fix this about 50 times, JD.
Sorry. I'm going to work on it right now before I get
distracted yet again. :-(

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

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 117 times

Subject: Re: copying objects
Posted by Robbie on Thu, 26 Jul 2007 23:58:26 GMT
View Forum Message <> Reply to Message

I thought that the definition of cloning means allocating a new object
and copying all fields.
If you don't want to create a complete copy the object then you should
consider creating a 'proxy' object which
looks up the fields of another object.

Cloning an object:
http://www.dfanning.com/tips/copy_objects.html

Something similar to a proxy object:
http://www.ittvis.com/codebank/search.asp?FID=508

NB.
I have had some difficulties using STRUCT_ASSIGN on objects, so I
usually just use N_TAGS

Robbie

Subject: Re: copying objects
Posted by Ingo von Borstel on Fri, 27 Jul 2007 07:20:25 GMT
View Forum Message <> Reply to Message

Autsch...
should have found this page on David's site... obviously I'm too
unobservant.

Thanks. I guess a 'deep cloning' is in order as my objects contain
pointer references...

Best regards,
Ingo
--
Ingo von Borstel <newsgroups@planetmaker.de>
Public Key: http://www.planetmaker.de/ingo.asc

If you need an urgent reply, replace newsgroups by vgap.