Subject: Re: Object within an Object's Structure
Posted by Paul Van Delst[1] on Thu, 10 Jun 2010 22:00:59 GMT
View Forum Message <> Reply to Message

carl wrote:
> Hello,
>
> I am trying to make use of what OOP functionality IDL has, and I wish
> to have as an object variable another object of a different type.
>
> Is this possible? If so, how do I specify this?
>
> I have an object type orbitingBody, and want to make an object
> orbitingPair. Would this work? how would it be then coded in my init
> function?
>
> pro orbitingpair__define
> struct = { orbitingPair,
> planet: obj_new('orbitingbody', args)
> star: obj_new('orbitingbody', args)
> etc...}
> end

Hmm. Why restrict yourself to only planetary and star orbiting bodies? (e.g. asteroids? comets?) Or even just two?

Assuming your "orbitingbody" object definition contains property information about the type of orbiting body (i.e. planet, star, moon, asteroid, comet, etc), why not something like:

```
pro orbitingSet__define
  void = { orbitingSet,
           n_bodies = 0L, $
           ; Container for orbiting bodies
           INHERITS IDL_Container }
end
```

?

You can then add as many orbiting bodies as you like:

```
pro orbitingSet::Add, $
  obj, $ ; Object to add
  _REF_EXTRA = Extra   ; Keywords passed onto IDL_Container::Add

  ; Add object to the container
  self->IDL_Container::Add, obj, _EXTRA = Extra

  ; If the object added is an orbitingBody object, increment the counter
```

```
  IF ( OBJ_ISA(obj, 'orbitingBody') ) THEN self.n_bodies++
end
```

This way you can store more information in your "orbitingSet" container if the need ever arises... and the objects you add needn't be just orbitingBody objects either.

cheers,

paulv
