Subject: Re: object newbie
Posted by promashkin on Mon, 14 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

I have a feeling that Chip's question is not quite answered by the dump
truck full of technicalities the group readily poured out. It looks to
me that the question is, "my object is just a structure but it is hell
of a lot harder for me to use it than a regular structure".
I would say that if all you need is the data and there is one instance
of it, you do not need an object. The whole idea is that object is a *reuseable*.
For example, you have one data processing task. You have a vector map
and an overlayed image. You want to be able to access the data for that
image only. Might as well write plain code for it.
Now, you have the same map, but 50 images that will be processed
similarly. In this case, if you write an object, then you can write the
"25 methods" only once, and very elegantly call those methods in your
widget event processing code. This way, you have intact data and a set
of actions (methods) you can perform on that data. And, if you happened
to aquire more bands of the image(s), you do not need to alter the code.
You just add more objects that use the same methods. On the other hand,
if you create a structure (or array of such), then you can't as easily
add another elements to it. Actions (code that modifies the data) tend
to become less organized (because they are not linked to data), and
event handlers become very complex.
This is not to mention inheritance, which simplifies dramatically
operations on data that are derivative or daughter to the main object class.
Cheers,
Pavel

Chip Sample wrote:
>
> I must admit that based on comments from this list, I have experimented with
> the object features of IDL for the past week or so, and have implemented
> them in a few places in a fairly big widget code I have written.
>
> My initial observations are that there seems to be less of a temptation to
> use common blocks when objects are used. On the other hand my first
> impression was that an object is a structure whose fields can not be
> accessed until you write additional "methods" to get at each and every damn
> one of them. So my object was littered with about 25 "methods" just so I
> could pry the data out of the object.
>
> I eventually came on a work around to write a "proto_object" with a method
> allowing you to pass a string containing a tag name which returns the
> contents of the field with that tag name. This "proto_object" is inherited
> by all other objects I create just so I can use this method. Along the way
> I found that the TAG_NAMES function in IDL doesn't work for objects so I had
> to create one. It basically copies the object structure into a regular

> structure so the TAG_NAMES can be used.
>
> Am I making this too hard?
>
> Chip