Subject: Re: IDL objects (not object graphics) tutorial?
Posted by David Fanning on Thu, 24 Nov 2005 04:18:31 GMT
View Forum Message <> Reply to Message

Dick French writes:

> I'd like to learn how to make use of IDL objects. I'm not ready for object
> graphics yet, because I'd like to understand INITS and SELF and classes and
> methods before worrying about viewports and plots disappearing because I am
> not using the correct projection scheme.

> Someone must be out there just waiting to get rich writing a book on this
> topic.

Getting rich is the easy part. Watching your marriage fall apart and your kids become alienated while you labor away writing the darn thing days, nights, and weekends is the hard part.

And then, of course, you realize you are NOT going to get rich unless you put a lot more sex into the thing than it has now, so you wonder if maybe fame will make the cost worth it. In the end, you would have been further ahead to have just purchased the IDL T-shirt and have been done with it. :-)

The last chapter of my book has a decent introduction to objects, although I can see that the example I used might have a "why bother" feel to an astronomer. It is hard to find the right example. They have to be easy enough to be explained, but sophisticated enough to suggest real world problems.

In the Hawaii classes I resorted to writing slightly simplified versions of my Catalyst objects, and maybe the combination of me spilling all my secrets and the astronomers asking for certain functionality created some kind of synergy. All I know is that all of a sudden we were in a grove and writing programs that became significantly easier to write with objects than it would have been otherwise.

I have a feeling our Thanksgiving celebration will be abbreviated a bit tomorrow, so maybe I'll have time to add some annotations to the programs we wrote and make them available. (We have been celebrating Thanksgiving for 25+ years with some college friends, but this year they suddenly decided they are vegetarians and there would be no turkey! We tried this experiment 10 years ago and it was a disaster, but I guess they don't remember. Sigh...) Anyway, if my famous butternut squash dish doesn't take too long in the

morning, I'll see what I can do. :-)

Happy Thanksgiving!

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: IDL objects (not object graphics) tutorial?
Posted by Benjamin Hornberger on Thu, 24 Nov 2005 04:40:19 GMT
View Forum Message <> Reply to Message

Richard G. French wrote:
> I'd like to learn how to make use of IDL objects.

Not being an expert, I can give a few examples when I found objects useful. If you don't know anything about objects, it might be difficult to grasp in the beginning, but I hope you can get the idea.

Often, you'll read that objects remove the separation between data and methods (meaning analysis algorithms, procedures etc.). For instance, I have a couple data analysis operations which involve reading a raw data file, applying a some operations on the data (which include some external parameters) and writing or plotting out the result.

In classical procedural programming, you would have a routine that reads the data file into a variable. Then you would call one or several functions / procedures on the data, also passing the external parameters, and each of them would return some intermediate result into more variables. Finally, you would have some routines which write the different kinds of output (image, plot, binary file, ...). You have a lot of variables to keep track of. If you analyze several data files at once, they become hard to manage. Also, you have to type a lot since you pass data in and out of routines permanently.

After writing an object for that analysis, the process for the analysis technique 'analysisX' might look similar to this:

```
obj = obj_new('analysisX')
obj -> read_datafile, '/path/to/file'
obj -> set_param, paramA=x
obj -> analyze
obj -> show_image ;; might pop up image
obj -> write_binary, '/path/to/file'
```

obj_destroy, obj ;; or keep the object if you want to reuse it later

At first sight, this might not look that revolutionary, but for complex procedures, this can simplify things a lot since everything is contained in one "object". For instance, if you analyze two files at the same time, you only have to keep track of one more object reference instead of all the variables (which are stored in the object and can be extracted if necessary).

If you write the object accordingly, you can change a parameter and update everything up to the final result with one command, like

obj -> set_param, paramA=y, /update

Also, it becomes quite easy to write a GUI for this analysis procedure. In simple cases, the GUI doesn't need any real analysis code. Each button click or whatever event just has to be translated into calling an object method.

Another example where objects are useful is compound widgets. While simple compound widgets can be written with standard widget techniques, you run into limitations soon. The reason is that you can't extend the WIDGET_CONTROL procedure for the specifics of you compound widget -- the only thing you can do is get or set a value. If you write the compound widget as object, you can do anything you want by calling methods on the object reference. David Fanning's FSC_FIELD or FSC_DROPLIST are good examples for that.

For complex widget programs, it even makes sense to write the whole program as an object. One advantage is that you avoid passing around your "state" or "info" structure all the time, because every event handler has direct access to all internal variables in the SELF structure (if you know how to redirect the event to an object method). The second advantage is that it is easier to communicate between separate widget programs -- if they are objects, you just call methods on each other. If the are not, you usually have to send events, which is much more cumbersome.

Since I wrote a lot already and it's getting late, I won't dwell on the syntax of object writing. David's book as well as Ronn Kling's "Application Development with IDL" have an introduction into objects. Also, the IDL help files are not that bad. And I'm sure others can explain much better than me what a class, an instance of an object, a method and SELF is (I might give a try tomorrow).

Happy Thanksgiving,
Benjamin

Subject: Re: IDL objects (not object graphics) tutorial?
Posted by Antonio Santiago on Thu, 24 Nov 2005 07:55:27 GMT
View Forum Message <> Reply to Message

Richard G. French wrote:
> I'd like to learn how to make use of IDL objects. I'm not ready for object
> graphics yet, because I'd like to understand INITS and SELF and classes and
> methods before worrying about viewports and plots disappearing because I am
> not using the correct projection scheme. I've scoured the web in vain
> looking for a simple tutorial on how and when to use objects in IDL. I've
> found a few generic tutorials praising the virtues of object-oriented
> programming, but almost none of the examples give me any sense of why one
> would go to the trouble. For example, one tutorial describes an object that
> can return constants such as the speed of light or Planck's constant, but it
> isn't obvious to me why this is superior to a simple function that returns
> clight() or PlancksConst().
>

It can seems stupid but remeber the things can be done by multiple ways.
That is, all you can do with object can be done without it (but perhaps
is will be more difficult). All you can do with IDL, can be done with C
or Fortran, and can be done in assembler (o nooooo !!!).

Befere learn "object oriented programming", take a look at the basic
concepts related with the "object oriented", then you can understand how
this concepts are applied in IDL. I like to note that OO in IDL is a
bit... special. You can get examples in Java that is pretty clear and
understans the "self" (or this), calls to superclasses and so on.

The OO way isn't always the best way for all problems, but it is really
a more good aporach to some problems because the concepts of OO are more
closely (or not :)) to the real things than a functional aproach.

The first problem in OO is: Can you model/represent that you want? If
you can model it with classes, association and generalizations and it
helps you, then it is the right way, else it isn't.

For example, you can understand an image as an ObjectImage. This is
composed by ObjectPixel that it composed by three/four ObjectByte.

ObjectImage (1) ----------> (*) ObjectPixel (1) ---------> (4) ObjectByte

Then you can program this classes to implement the methods needed to
work with images, but I think this is a very bad use of OO.

> What I am looking for is something with a simple application or two in which
> it is both clear why using objects is superior AND which explains what is
> meant by self and methods and classes. Without some specific examples to

> look at, I am having a hard time making sense of the nomenclature or of the
> value of the approach.
>

A basic, little or simple application in IDL (or any language) not needs strictly the aproach to OOP. It depends on your needs. Well, here I would define the terms "little" and "simple" but I think you understand me.

I can gibve you an example of OO aplication I am working on (with the rest of my job companion :)). We are developed an application that can read some datas from radar, metheorological satelite, lightnings information and some gis information. Every data is "tranformed" to a so called class "frame", and every frame is placed in a "layer". Layers can be visible/invisible, animated/stoped/freezed, ... and all the layers are visualized together in the same style of photoshop :D (this sound very good).

Here is a good example of using OOP, because the concepts: layer and frame are easily represented and handled with object.

> This is prompted in part by David's nifty little pixmap object that I've
> already made use of in a new program - thanks, David.
>
> Someone must be out there just waiting to get rich writing a book on this
> topic. The second volume can be about object graphics - I'd settle for the
> first volume for now - a gentle introduction to objects in IDL. Any
> suggestions? Thanks!
>
> Dick French
>

--

Antonio Santiago Pï¿½rez
(email: santiago<<at>>grahi.upc.edu)
(www: http://www.grahi.upc.edu/santiago)
(www: http://asantiago.blogsite.org)

GRAHI - Grup de Recerca Aplicada en Hidrometeorologia
Universitat Politï¿½cnica de Catalunya
