Subject: Re: Object Models for Data?
Posted by Mirko Vukovic on Tue, 30 Sep 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Struan Gray wrote:
>
> I'm about to start converting my main application to an
> object-oriented version and intend to start by objectifying my data
> models. Naturally I have lots of lovely low-level ideas, but because
> I eventually want to write widgets which can browse, display and
> manipulate any of my data sets, no matter where or on what machine
> they were aquired, I want an overall data model that is robust and
> flexible enough to cope with most things that I might want to throw at
> it in future.
>
> The obvious route is to objectify the HDF routines, since they
> provide a standard way of describing data sets with lots of different
> components, and it is easy to see how an HDF file on disk can be
> converted into some sort of container object in memory. A basic set
> of methods for describing, adding and extracting the object's data can
> also be stolen from the HDF file routines.
>
> My only problem (at least, in this respect ;-) is that I have a
> strong suspicion that I am re-inventing the wheel, and that more
> intelligent lifeforms have probably already developed a suitable
> model. I have had a cursory delve into the NCSA HDF documentation and
> website(s) but although the whole thing reeks of object orientation,
> the end result for the user of the HDF libraries seems to be boring
> old arrays and variables: ie the data stops being an object once it
> leaves the disk.
>
> Hence an appeal to the newsgroup: does anyone know of any freely
> available projects or standards for the description and manipulation
> of scientific data *in memory*? Would anyone be interested in trying
> to put together a general object model for data in IDL applications
> that could be used as a basis for shared code? My own feeling is that
> a standardised data object would make it much, much simpler for IDL
> (and PV-WAVE) users to share widget applications with each other, as
> well as making it easier to do hip things like interface applications
> to the web.
>
> Ideas? All bouquets and brickbats gratefully received.
>
>
> Struan
> (struan.gray@sljus.lu.se)
I am not into widgets, but I have been thinking along these lines for
the past several weeks and came up with my own model object.

Now, I am not certain that this is exactly what Struan had in mind but my model object serves the following purpose:

- provides a unifying environment for storing variables and parameters pertaining to a set of equations -- a physical model.
- provides a standard interface to the physical model.
- provides a standard interface for plotting routines.

(Frankly, the thing is not up to specs right now, but first and last point are almost there)

One would start with specifying the name of the model and a label (a shorthand for identifying purposes). Next (but not in temporal or any other order) would come the variables and parameters, again each with a name and label. If two of the variables are vectors, they can be optionally transformed into matrices. The vector or matrices are noted since they are the changing variables and this info is used in the plot method later on.

Finally the model can contain other objects.

Plotting is done by a standard interface, with the model method determining the details of the plot (axis and labels), based on the variables provided. If an object is to be plotted, it's plotting method is called.

This project is still under way, as I am only getting into OOP (and using the OO design heuristics book along the way), and the design of the routines is still not fully satisfying.

One interesting side effect of the project is an approach for modeling and plotting results that I came up with.

The problem was that I needed flexibility in specifying the plotting output, which was hard to accomplish considering the structure of OOP and how hard it is to access data from an object. Specifically, I needed to plot x vs. y not y vs x.

I decided to break up the task into a) modeling and b) plotting. Modeling was done largely using the framework above. The results of the model would be in one object or another, each with it's plotting method.

The plotting task was done with MPPM (multi position plot manager), a routine that remembers the plot sysvars for each open window (either separate window or separate region within a window). Each plot window can have it's own set of interface methods to commands like PLOT,

CONTOUR, SURFACE. These determine what is the x axis, y axis and things like that.

Now,the object plotting method can do a test to see if the interface method exists, and if it does, it passes the data to it.

This way, the shape and form of the plot is done separately from the modeling process, during the plot window initialization.

Is any of this clear? Anyway, what all this points to is the need for IDL++, where we can override the meaning of PLOT,and things like that to mean whatever the user wants at some particular instance. (this I say without even knowing what C++ is all about other than operator overloading -- and I am probably even wrong there)

Well so much for now.

--
Mirko Vukovic, Ph.D 3075 Hansen Way M/S K-109
Novellus Systems Palo Alto, CA, 94304
415/424-4969 mirko.vukovic@grc.varian.com
fax:415/424-6988
