Subject: Re: When should objects be used?
Posted by John Persing on Thu, 24 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

I have used objects for over a year and spurred some of the recent discussion. I have used two objects, in four different contexts. But before I start with that, let me make two observations on your programming: 1) it works, 2) and you use reusable programming units. Those are the two more important features of any scientific programming effort.

Use widgets if you want to interact with the data in a random manner to explore different ideas. In this context, your widgets would only "front-end" the codes that you have already written. I agree that you don't want to produce your publication figures with a widget. There are other options, like storing all your variables created by the widget in a structure that can be saved or come up with a mechanism that can store your button clicks, but those are probably not as robust as handwriting a procedure does that things that you would have done on the widget.

The first time I used objects was to create an "editor" in a text widget, but the "editor" was to be a restrictive program development environment. The skinny is that the user would have a lot of liberty to edit the lines of the "editor" in some ways, but with very great restrictions in others. What I wanted to do was put strict curtain around the data edited by the "editor", which is one advantage of objects. Because all changes in the object must be routed through a limited number of methods, you can control strictly how the object is changed. But if you are not coding for a "dumb user" (a programmer should be arrogant enough to claim it is all the other people who are "dumb") then this shouldn't motivate you.

The second object I made was a little more ambitious. I made an object to store any kind of data and I have used this object with three datasets with entirely different structures. The first two datasets were just to test the concept, while I really had the third problem in mind. The first dataset was to read a text file containing the radii and masses of each of the planets and of each of the satellites around each planets, so basically a structure array with an additional mechanism for attributing a satellite to a parent planet. I discovered that the routines I want to attach to the object are those that are well-used and are intimately related to the data itself, such as the utility that attributes satellites to parents. The second dataset was to store the x and y positions of N point vortices over T time steps, so basically it was a multidimensional array. An object is certainly not the best way to deal with this problem either, but it was useful for testing concepts. A side benefit though was that I was able to create a single, master widget interface that could interface with all three datasets by reusing this object. (How I did that would be a topic of another long post.)

But the third problem is where the object is ideally suited. I ingest a bunch of large 4-D arrays (yet small enough for all to stick in memory) of temps, winds, moisture, etc. that is output from an atmospheric forecast model. Then from these fields, I can compute derived fields, such as vorticity, horizontal gradients, etc. But I don't always. What I wanted to do was to create variable structure that would make it easy for me to write code with. I want to be able to perform gradients or averages on any field, or even any list of fields. I want to do so at will. So I stuff all the fields into an object called "dat". Any new field get added to dat. All the fields are referred to by a string name. The data is stored as a simple structure array, where the first element is the "name" as a string and the second element is the "dat" as a pointer, which can point to anything. The primary methods to the object are "query", "assign", and "delete".

dat->query("u") asks for the u-winds.
dat->assign, "speed", SQRT((dat->query("u"))^2.0 + (dat->query("v"))^2.0)
creates a new variable with the wind speeds
dat->delete, "speed" gets rid of the new field.

Also, there is a mechanism for querying for a slice of a data field. Other methods check to see if a variable exists and to print out a summary list of the names of the fields stored. As you may see, this allows for a free-flowing mechanism for interacting with the data, liberally creating derived data at whim. Most of the mathematics are performed in separate procedures. One example is a code to produce radial profiles (of the cylindrical dataset) of an arbitrary list of data

compute_radial_means, dat, ['vort', 'u', 'v']

where the output will be stored in dat as the new fields "rad_prof_vort", "rad_prof_u", and "rad_prof_v". Because the datafields are referred to by string name and all the data fields are stuffed in dat, this free-flowing approach works. Keywords allow the additional retrieval of asymmetric portions and settings for dealing with masked data points. And again, I can use the same tested widget metaphor used above. This allows me to do one-liners at the IDL prompt with ease and write code quickly.

I have seen the short-comings of this approach, though. If a field isn't in the object, I either have to do alot of robustness coding, or just rely on the errors generated. I generally do the later because, of course, I am not a "dumm user". Also, I wish I would have thought of a mechanism for storing metadata for each field, specifically how to map each of the dimensions of each field to independent variables. I will definetly come up with an additional mechanism for handling bad datapoint masks, which are now not so satisfactorily done by setting a large negative number.

Some advice: Only use an object to solve a specific shortcoming of your present approach or better yet that you anticipate from your future

programming effort. Also, implement objects from the start of the programming project. The choice of data structure should always be your first decision. Then you start coding.

--

}3 John Persing }3
http://www.frii.com/~persing persing@frii.com
Half of all Americans earn less than the median income!!!!!!
