Subject: Re: Interactive Objects, Was: Simple GUI question
Posted by Richard French on Wed, 23 Apr 2003 13:38:21 GMT
View Forum Message <> Reply to Message

On 4/21/03 10:47 AM, in article MPG.190db260977ddbea989b5c@news.frii.com,
"David Fanning" <david@dfanning.com> wrote:

> I'm not exactly sure why people are afraid of objects.
> Certainly the basics of object programming can be taught
> in half a day. I make sure I include at least this much time
> in any IDL programming class I teach.
>

I have used IDL for a long time and I have not yet taken the time to learn how to use objects in any language. I've gotten as far as using an occasional widget or two, and in some cases a rather complex set of compound widgets, but not objects. Nearly all of my programming involves either scientific image processing or computations, and the sequence of things is typically as follows:

1) Get new data or 'clever' idea and use IDL in the old-fashioned interactive mode, with a journal file turned on. I might use several low-level IDL routines that I've developed over the years to help make this part a bit quicker.

2) Once I have something that looks like a good start, I turn the journal file into the skeleton of a program, and I think a bit about how important it might be to generalize the program, or modularize it into procedures or functions that I might want to use again some day. I try to find a balance between solving today's problem efficiently and being ready for next year's problem.

3) A KEY POINT for me is that I want absolutely reproducible results without having to do any interaction - that is, my final products are usually images, or PostScript figures, or tables that I format into TeX (using IDL to construct the TeX file), or a text file of results. I want to be able to run IDL two years from now, using the same program, the same data and the same input specifications, and get exactly the same final image, or plot, or table, or text file. In other words, I want to get rid of the 'I' in IDL as my final step, and be able to run a batch job that gives me the final result.

4) This is very different from, say, Adobe Photoshop, where someone might be able to produce a beautiful rendition of a blurred, overexposed original image, but not be able to tell me succinctly and reproducibly how they did it.

This is one of the reasons that I don't do a lot of GUI programming

involving user choices along the way. In instances in which I have done this, I end up saving a structure that contains all of the choices that the user has made, and then running the program in a non-interactive mode, making those same choices. But the design here gets tricky, since widget programs don't usually have a 'start' and a 'finish', compared to the programs I have in which I am basically doing a well-defined series of operations in a given order.

Now, how about objects? I know that a lot of these are really cool! I like being able to display images in a resizable window and having the image follow along. I like being able to click on an axis of a plot and reformat the style. I like using objects that are self-contained and do a well-defined task. My problem, when thinking about actually writing one of these beasts, is that there seem to be too many choices! How many methods should I specify? What about classes? In standard Idl programming, one needs to decide whether a procedure or function should do one small thing or several things, and it is not hard to develop an efficient style there. In widget programming, the problem gets trickier - how do you put together all of the widgets you want to use, and make sure that they interact with each other properly? I admit that, in my applications, I enjoy programming widgets but it takes longer than I can really justify, in terms of the actual use of the program in the end. As someone just beginning with objects, it is hard to get a good sense for how to break up a problem into manageable pieces, and to construct individual pieces that make sense.

To follow David's analogy, I sometimes think that my widget programming style ends up being a design for an all-in-one blender, buzz saw, and CD-ROM burner, without doing any of them well. Object programming leaves me at a loss in terms of these fundamental design principles. The usual thing that I see is an example that says: "Start with an object that does one little thing, such as squaring a number or an array." But then I end up with about 100 lines of code which, in the end, just do what I can already do in one line of IDL - square a number or an array. What I need to see is an example of simple objects that can do something useful in IDL that I can't already do in a few lines of code. That might help me get over the hump so that I could join David's bandwagon.

I should be working on my NASA proposal....

Dick French