Subject: Re: taking the widget plunge. help
Posted by John-David T. Smith on Mon, 11 Sep 2000 23:25:11 GMT
View Forum Message <> Reply to Message

> P.S. Let's just say I spent a couple of days writing
> documentation for FSC_PSConfig, *the* best program I've
> ever written, and no one downloads it or uses it. You just
> figure after a while, what's the point?

I took this as a friendly reminder to check out your new stuff, and I was frankly amazed at how much work you'd put into the documentation. Really an astounding effort. I will try to integrate FSC_PSConfig into some of my programs around here.

One note which I think is instructive. You include a section on customization where you outline how to directly modify your source to add personal or company-wide set-up lists. This is a very useful feature, but I think you're going to cause yourself and potential users grief here. It's a *perfect* place to flex our object oriented muscles. The problem will be that in a year you'll think of a great way to redesign it, or maybe RSI makes some changes to device which prompt a rewrite. Then, either all the users who have made their own modifications will be out of luck, or you'll be contrained in what kind of updates you can do. It is exactly these types of situations that scream out for some sort of object relationship. If, rather than giving direction on how to change your code, you gave a simple example of INHERIT'ing your class, and chaining to its setup code, you could fully preserve "forward compatibility" -- i.e. drop-in replacement of your updated code.

Or, since in this case the local setup changes are data-only (no fundamental method changes), you could simply provide access to an internally growable list of setups. Inheritance is not even really required.

I haven't looked closely, but a method which allows you to add setup lists (e.g. self->AddSetup,"Company Viewgraph",/EUROPEAN, FontNameSet="Helvetica"), would seem to do the trick. This might be called automatically in Init with all the built-in defaults. A user could INHERIT it, override and chain to AddSetup for a fully internal solution, or they could use a compound relationship and add the setups "from the outside" in whatever wrapper routine (or object) they use.

The details of how set-ups are stored, manipulated, etc., would be hidden, only the published interface of AddSetup need remain the same (or backwards compatible anyway... nothing to stop you adding new keywords as new features become available).

Anyway, it's just a thought. Perhaps you're afraid of scaring off potential users with objects. You shouldn't be. It's good for them.

JD

--

```
 J.D. Smith                             /*\    WORK: (607) 255-6263
 Cornell University Dept. of Astronomy  \*/         (607) 255-5842
 304 Space Sciences Bldg.               /*\     FAX: (607) 255-5875
 Ithaca, NY 14853                       \*/
```