Subject: Re: Interactive Objects, Was: Simple GUI question
Posted by rmoss4 on Sat, 19 Apr 2003 15:49:08 GMT
View Forum Message <> Reply to Message

When is the all-singing, all-dancing Catalyst library going to be
available to the public?

Robert Moss, PhD

Subject: Re: Interactive Objects, Was: Simple GUI question
Posted by David Fanning on Sat, 19 Apr 2003 16:34:50 GMT
View Forum Message <> Reply to Message

Robert Moss (rmoss4@houston.rr.com) writes:

> When is the all-singing, all-dancing Catalyst library going to be
> available to the public?

Sigh... Real soon now.

Cheers,

David

P.S. Let's just say that when I talk about the resurrection
I'm talking about my income, not about the story of Easter. :-(

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Interactive Objects, Was: Simple GUI question
Posted by R.Bauer on Sun, 20 Apr 2003 21:04:24 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> Reimar Bauer (R.Bauer@fz-juelich.de) writes:
>
>> I like very much the I of idl it stands for interactive.
>> For my opinion objects can't be interactive they are always well planned.
>>

>> The people starting with idl are almost very different in learning and
>> working with idl. In most cases my feeling is if they can theireselfs
>> work a bit on a solution without reading many books or asking a lot of
>> people for help they like idl more.
>
> I'm not sure how this discussion got turned in the
> direction of objects. I'm pretty sure it started out
> as a discussion of how to write a modal dialog widget.
> But the fact that it did encourages me. I've learned
> in my programming therapy sessions that talking about
> something is the first step in the process of accepting it. :-)

Dear David,

you are right! I was inspired myself several times by the first big library
I found from Ray Sterner especially the idea of a common timeaxis.

It's absolut correct thats there is a gap between the existing objects and a
catalyst library. And it takes always more time to write library routines
useful for others too against a "closed application".

If you have a more detailed look at our library you find some routines which
are based on more than hundred library routines. For me and some others it
is extremly easy to build a very complex program dependend on the existing
libraries. During the programming there are always new simple
library-routines are created to solve the main problem. Some are small some
are large. But after you have done this once you save a lot of time and all
the others which understands what's the library did saves time too.
Sometimes I wonder how much routines really are depended to an easy or
simple program. (A while ago I have solved one of the live tools I don't
know at the moment which one it was but it has dependcies to more than two
thousands routines)

What is your estimation in teaching people never have programmed before.
At the moment my feeling is if we don't start with objects by beginners it
is more difficult to get them later to work on objects too.
But on the other hand if someone has only small time to solve his problems
by idl without a "catalyst object library" it would better for him to show
him the "object free idl".

I think teaching objects will be always nearly the same as teaching of
writing library routines. In objects there is quite no difference between
this. But normally we try to teach people to use idl dependent on the idl
commands because they work in very different places with several of
different problems. We don't have libraries to solve all their problems. So
I believe it is better to show them how to build routines to solve their
own problems. (There is another yearly seminar for our people about the
increasing icg library)

But I believe it is very hard to teach a beginner the clever usage of
objects in idl with the momentanly existing objects.

I hope this discussion helps to form an opinion how we should proceed
teaching the next classes.

best regards

Reimar

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
========== ======
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

---

Subject: Re: Interactive Objects, Was: Simple GUI question
Posted by David Fanning on Mon, 21 Apr 2003 14:47:21 GMT
View Forum Message <> Reply to Message

Reimar Bauer (R.Bauer@fz-juelich.de) writes:

> What is your estimation in teaching people never have programmed before.
> At the moment my feeling is if we don't start with objects by beginners it
> is more difficult to get them later to work on objects too.
> But on the other hand if someone has only small time to solve his problems
> by idl without a "catalyst object library" it would better for him to show
> him the "object free idl".

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.

And while it is true that a library of objects makes application
development MUCH easier, I know for a fact that half a day is
enough to get a number of people hooked for good! Objects can
be that powerful. Even simple objects. (Rob Dimeo got turned on
over dinner one night, for goodness sake! I didn't even have to
draw any diagrams.)

> I think teaching objects will be always nearly the same as teaching of
> writing library routines. In objects there is quite no difference between
> this. But normally we try to teach people to use idl dependent on the idl
> commands because they work in very different places with several of
> different problems.

IDL courses are probably the wrong place to try to solve
everyone's problems. I'm happy if I can get people excited
enough about IDL that they want to go out and learn more
about it on their own. It is not unusual, though, for people
to complain to me after a class about the work they have ahead
of them. Sometimes they want to go back home and re-write *all*
their IDL programs!

I don't usually recommend that. Programming evolves like everything
else. I'm satisfied if the programs I write tomorrow are better than
the ones I wrote yesterday. If you know about objects and are not
completely frightened of the words "object programming", then objects
will naturally find a way into your programs. You can't keep them
out. :-)

> But I believe it is very hard to teach a beginner the clever usage of
> objects in idl with the momentanly existing objects.

Oh, I guess "clever" is in the eye of the beholder. Objects
seduce you (at least if you have a bit of imagination) into being
too clever. I think you write better objects if you are a bit
of a dullard. Probably why my library works so well. :-)

> I hope this discussion helps to form an opinion how we should proceed
> teaching the next classes.

Not learning about objects in an IDL programming course is
like taking a woodworking course and skipping the bit about
the dovetail saw. Yes, you can make boxes. They just won't
be as beautiful.

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

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