Subject: Re: Comparison of IDL and PV-Wave (and OO)
Posted by David Ritscher on Fri, 11 Jul 1997 07:00:00 GMT
View Forum Message <> Reply to Message

...
> I understand that PvWave and IDL diverged some years ago; but the FAQ
> implies that they are still quite similar. I wonder if anyone could
> compare them, strengths and weaknesses. I'm particularly interested in:
...

Dear Steve and group,

I am a long-time IDL and PV-Wave user. The comparison between the two products is currently rather complicated. I try to program so that most of what I do is compatible with both packages (at least, except for the widget part of things). Up till the newest release of IDL (5.0) the basics of the languages were the same, with only trivial differences, except for:

1. A different widget interface
2. A different set of math extension routines. IDL uses Recipes in C, PV-Wave uses the IMSL routines. After the makers of IMSL and PV-Wave merged, IDL lost access to the IMSL routines and then chose to adopt the Recipes in C routines.

With IDL 5.0, it looks like one of the players is starting a much-needed new thrust, towards an interactive data and graphics language that can integrate current computer science developments. Although the new version is mostly unchanged, it looks a little bit more like C++. With luck, they will continue to bring the product closer to a 1997 state-of-the-art, while providing mechanisms for backwards compatibility.

IDL 5.0 introduces some elements of object-oriented programming. I have developed a software system for cardiology analysis that is in daily use in our hospital here. It contains some 35,000 lines of code. It is not object oriented, and I am reaching a point where I lose tremendous time and energy to repairing things in this code, fixing things that get broken when a new feature is added, etc. From this experience the need for an object-oriented approach has become clear to me. PV-Wave says that they have no current intention of changing the basis of the language so that it would support object-oriented programming. (They speak about object oriented, but this has to do with some pre-packaged tools, not with the basis of the language itself).

Whether object oriented is important for your application depends on what you'll be doing. If the chief goal is to use the software as a

hand calculator with good graphics display capabilities, it is probably irrelevant. If the goal is to design software components to be used and extended by a group of people, then I would consider it crucial.

But speaking of object oriented, the makers of PV-Wave are demonstrating some strides in this direction, namely JWave and JNL. It looks like they have given up on the idea of updating their own language and are instead jumping on the Java bandwagon, and providing tools for Java that provides advantages normally available within PV-Wave. There is a writeup on JWAVE on a Netscape DevEdge page: http://developer.netscape.com/guides/components/
JNL is an extension to Java that provides needed numerical capabilities.

Don't forget to look at MATLAB as another possible choice. They are also making strides now, and it looks like they are adding enough new language features that one could now program a real system within MATLAB (before, there were major deficiencies, such as no 2-D arrays, no integer type, etc.). Particularly interesting with MATLAB are the availability of 'compilers' that convert MATLAB code to C code. A major advantage is the rich collection of toolboxes, often writen by well-known people in the area of the toolbox.

As to the cost question, this is also somewhat hard to evaluate. Here in an academic setting in Germany, out state has obtained state-wide licensing. For my institute within the University an unlimited license for all possible platforms costs about $400 per year. Meanwhile, I see that the IDL folks have learned from the MATLAB folks, and are offering an IDL student edition for US$79, International Price: US$105. For commercial purchases, the packages aren't cheap. There it depends on which toolkits you buy, etc.

It is a hard time to make a choice between these two branches of the language, since both companies are working hard to try to differentiate themselves. There remain problems within the common core language, and I hope that this competition will lead to solutions to these problems. As someone here recently mentioned, when one write, in either language, "a = 3", one has just defined a TWO BYTE INTEGER variable (And this, in 1997!) This was probably a logical default when the language was first conceived. A mechanism for backwards compatibility plus future development needs to be reached; perhaps a system flag, where, for example, !LANGUAGE_LEVEL=1 inserted into routines would provide backwards compatibility to a specified language version.

I would be interested in hearing from current IDL users as to how sufficient the current object-oriented features are, and what is missing. Reading through the documentation I see, for example, that

polymorphism and overloading aren't there.

Regards,

David Ritscher

--
David Ritscher
Zentralinstitut fuer Biomedizinische Technik Tel: ++49 (731) 502 5313
Albert-Einstein-Allee 47 Fax: ++49 (731) 502 5315
Universitaet Ulm Internet:
D-89069 ULM
david.ritscher@zibmt.uni-ulm.de
Germany