Subject: Re: a plea for more reliable mathematical routines
Posted by FIT on Thu, 16 Sep 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Mirko Vukovic wrote:

> In article <37DE1600.5E99BDE4@zedat.fu-berlin.de>,
>   fit@functional-imaging.com wrote:
>
>> I definitely disagree. It is inferior to Java, Python, C/C++ (if
> You're able to
>> program a little bit of OpenGL and Motif yourself) to name only some,
> far too
>> expensive, introducing new bugs with every release (maybe a merger
> with Micro$
>> would be adequate), lacking hooks for any reasonable development
> environment (or
>> have You ever managed to get it to work with Rose or SNiFF+ to name
> only a few).
>
> I would agree if you compare them as general purpose languages. But for
> data analysis and writing imaging routines, I presume that IDL beats
> these, since it was designed (with flaws) for that purpose. You
> can accomplish the same with the languages you mentioned, but with
> how much effort.

Well, its nice to hear that there at least once was a design idea. Apart
from the fact that it allows convenient manipulation of higher dimensional
arrays (but of course indexing, storage etc. opposite to a mathematician's
habit), which was not unproblematic until some years ago in other languages
I definitely do not see anything more. Linking with numerous publicly
available libraries gives You better functionality and - as image processing
mostly is mathematics and IDL is especially poor there - more reliable
results.

>
>
> I restrict my comment for small and medium sized applications. For
> a huge application with millions of lines of code, it may be more
> worthwile to go to Java/C++/..., simply because of the ruggedgness
> and the development tools.
>

Everything above say 1000 LOC intended to be reused should definitely be
designed (!!) and implemented properly (meaning not IDL).

>
> Regarding the above issues I would prefer a comparison of IDL with

> PV-Wave, matlab, mathcad -- none of which I use.
>
>> Secondly, I definitely did not characterize objects as childish but
> the way
>> they're used and implemented in IDL (look folks, now we're object
> oriented !).
>> What has been done there to the object paradigm is pretty much the
> same as they
>> did to numerical mathematics (look folks, we've the numerical recipes
>> implemented, ok the results are shaky at best, but look we have them
>> implemented). To incorporate an object oriented paradigm
> (encompassing, yes
>> David, a development process as well) is a little different to
> providing a syntax
>> of o->x() form.
>>
> I agree that 5.2 is not up to C++ regarding oop, but with some
> programming conventions, can you achieve much of the same results?
> Like, you cannot define a private/public interface, but can
> you as a programmer label an interface as such and use it in
> a consistant way. I agree it is inferior to an explicit declaration,
> but better than nothing. (here I am threading a ``tiny bit'' beyond
> my expertise)
>

1.) That's exactly what OO is about. It's not just an syntactic
(in)convenience but design and programming for an interface and for reuse
(not code). Much of the result of OO efforts is the interface and thus IDL's
pseudo OO will not (not !!) achieve any of the results a moderately
experoenced designer will achieve with OO methodology.
2.) There are no two programmers on this globe who do the same thing
consistently the same way.

>
> Mirko
>
> Sent via Deja.com http://www.deja.com/
> Share what you know. Learn what you don't.

--
Functional Imaging Technologies GmbH
Siemensstr. 40/41
12247 Berlin
Germany

fon.: +49 (0)30 76 90 24 80
fax.: +49 (0)30 76 90 24 81

mailto:fit@functional-imaging.com
http://www.functional-imaging.com