Subject: Re: VISualize 2009 summary
Posted by T.H. on Wed, 22 Apr 2009 10:17:34 GMT
View Forum Message <> Reply to Message

On Apr 21, 4:07 pm, Bob <bobnnamt...@gmail.com> wrote:
> On Apr 20, 12:01 pm, mgalloy <mgal...@gmail.com> wrote:
>
>> I'm going to write a few posts about the contents of last week's
>> VISualize 2009 seminar in Washington, DC, starting off with ITT VIS'
>> presentation about the roadmap for IDL 7.1 and beyond:
>
>> http://michaelgalloy.com/2009/04/20/idl-roadmap.html
>
>> Hopefully these will be useful for those unable to attend. I know I had
>> a great time, especially meeting face to face those of you I had only
>> communicated with online previously.
>
>> Mike
>> --www.michaelgalloy.com
>> Associate Research Scientist
>> Tech-X Corporation
>
> Out of the list of 13 items, I see 3 that interest me (to say that the
> development of IDL is going in the wrong direction is an
> understatement):
>
> 24-bit color PostScript in direct graphics
> modern language features (associative arrays, lists, operator
> overloading)
> modern UI toolkit
>
> The hashes and lists are especially promising, but where is the
> garbage collector? Certainly most people would take a garbage
> collector before most items on the list. Since a garbage collector
> seems unthinkable for ITT (and RSI before them), they should consider
> a reference type. This would be like a pointer to heap variable
> except that it would be guaranteed that only one variable would be
> pointing to it (sort of like an allocatable in Fortran). Since only
> one variable would be pointing to it, the heap variable could be
> garbage collected whenever the reference variable went out of scope or
> a new variable was assigned to it (this is exactly the same as a
> "normal" IDL variable so all the code is there to do it). As a
> further benefit the need to de-reference the variable to get at what
> it was pointing to would not be necessary which would clean up code
> using reference variables in structures or arrays. Here is an example
> how it could be used:
>
> ; define a structure with two reference variables (one defined and one

> not)
> struct = {a:ref_new(fltarr(10)), b:ref_new()}
>
> ; assign the 2nd one (note that ref_new is need to assign a new
> reference)
> struct.b = ref_new(fltarr(20))
>
> ; re-assign tag a of the structure
> ; (the heap variable containing fltarr(10) is garbage collected)
> struct.a = ref_new(fltarr(15))
>
> ; note that no "*" is need to de-reference struct.a
> print, struct.a[2:5]
>
> ; if struct.a was a pointer than this would be print, (*struct.a)
> [2:5], yuk
>
> For me at least, 95% of my usage of pointers in IDL is for use in
> structures or ptrarr's and reference variables could eliminate this
> need. These could be easily garbage collected and provide the added
> benefit of eliminating the (*ptr)[] syntax which is hard to read and
> error prone.
>
> What do you think?

I think that IDL has a lot of needs but a garbage collector is pretty low on the list since with proper programming all of your pointers will be tracked and handled anyway. I've very excited to see plans that they have for modernizing the language and interactive graphics.

Mike, Thanks for the post!

---