Subject: Re: fast for loop
Posted by JD Smith on Mon, 11 Jun 2007 19:00:36 GMT
View Forum Message <> Reply to Message

On Mon, 11 Jun 2007 07:01:36 -0700, Conor wrote:

> On Jun 10, 10:55 pm, David Fanning <n...@dfanning.com> wrote:
>> airy.ji...@gmail.com writes:
>>> Sometimes we could use some special function to avoid them.Sometimes
>>> we could use more lines of code to avoid them.
>>> There are many skills to make the program more efficient and fast.But
>>> in fact loops are ineluctable,the key is how
>>> to use it.
>>> Anyway,I agree with you ,Mark.IDL indeed need to improve its
>>> efficiency on the loops and some arithmatic computing.IDL6.4
>>> shows a lot of features which can be proved thier hard working for
>>> it.At least 50% elevation of the drawing speed and new
>>> OpenGL Object indicates an nice future of the IDL.It's wothy for us to
>>> waiting new edtion of the IDL which can give us some
>>> fast loops,^_^.
>>
>> Yeah, and life would be more interesting if pigs could fly.
>>
>> If fast loops are what you were after, I'd guess you would
>> design a language that looked very much like C or FORTRAN.
>> IDL is something different and I don't see much point wishing
>> it wasn't.

> Granted, it would be nice to have fast for loops (for those times
> where you really just have to use one). However, I do also enjoy the
> challenge of having to come up with fun new ways to avoid them.
> There's nothing more satisfying than taking a couple lines of code
> wrapped inside a for loop and turning it into one line of convoluted
> array operations. Normally I have no artistic talen what-so-ever, so
> coming up with confusing idl code in order to avoid for loops is my
> way of expressing my creative talents :)

It's funny because it's true. Some of the tricks we resort to to get good
performance out of IDL fall in the category of elegant. Most do not.
I've long advocated a specially compiled for loop which drops essentially
all the features of the IDL interpreter, which no doubt are what make a
single round trip through the for loop so slow (warning: this is informed
speculation). This "optimized side loop", which might get enabled with a
compiler flag, would have some inherent inflexibility, but should offer
much better performance.

JD