Subject: Re: Double-precision plotting doesn't work in object graphics
Posted by karl_schultz on Fri, 01 Feb 2002 15:12:06 GMT
View Forum Message <> Reply to Message

"Mark Hadfield" <m.hadfield@niwa.co.nz> wrote in message
news:<a3d57g$pt1$1@newsreader.mailgate.org>...
> In IDL version 5.4 RSI introduced double precision support in its graphics
> routines and objects to allow (amongst other things) plotting of times
> expressed in Julian Days. I have just noticed that it doesn't work as
> advertised in Object Graphics.
>
> The routine below demonstrates this. It generates & plots two day's worth of
> data at 15-minute intervals. Option 0 gives you a Direct Graphics plot: it
> shows a nice clean ramp. Option 1 gives an Object graphics plot: it shows
> steps at ~0.3 days spacing.
>
> Here is a calculation of the granularity expected in single- and
> double-precision representations of Julian dates:
>
> IDL> m = machar() & feps = m.eps
> IDL> m = machar(/DOUBLE) & deps = m.eps
> IDL> print, 2452301*feps, 2452301*deps
> 0.292337 5.4452021e-010
>
> The step spacing in the Object Graphics plot is suspiciously similar to the
> single-precision granularity. Obviously some single-precision numbers are
> creeping into the Object Graphics calculations.
>
> Version tested: { x86 Win32 Windows Microsoft Windows 5.5 Aug 28 2001 32 64}
>
> Oh well, it looks like I'll have to translate the data and fudge the labels
> a little longer.
>
> ---
> Mark Hadfield
> m.hadfield@niwa.co.nz http://katipo.niwa.co.nz/~hadfield
> National Institute for Water and Atmospheric Research

< snip program >

The short answer is to add /DOUBLE to your call to XOBJVIEW. Then the
plot looks like the direct graphics plot.

Here's why:

Object Graphics uses OpenGL. OpenGL only guarantees floating point
precision that is about what you get with IEEE single precision
numbers (no accident, I suppose). The OpenGL folks figured that most

apps would already be OK with this or could easily make themselves OK with this. Also, a lot of graphics hardware is built with this sort of limitation on the same assumption.

Normally, IDL passes data to OpenGL in "user" data space. That is, if you are drawing things in the 1d+10 range, we pass these numbers to OpenGL and let it transform this data to the screen space, which is in the 0-2000 sort of range.

But OGL ends up storing your user data in single-precision floating point, and so you will lose precision as OGL stores it away in a single precision float. That is why you got the stair-step pattern.

To get around this, IDL can be configured to perform all the transforms from user to screen space in double precision ABOVE the OGL layer. OGL ends up getting coordinates in screen space and happily processes them without applying a general model transform. The way to activate this mode is to turn on the DOUBLE property on IDLgrView. The DOUBLE keyword on XOBJVIEW does exactly this.

Why isn't this the default behavior? For software rendering, it probably does not matter much, because we're just moving the vertex transform step out of OGL and into IDL. Many OpenGL libs are optimized to skip the transform step if the current transform matrix is the identity.

The problem is that people with devices that have hardware-accelerated transform and lighting would be very upset because IDL would always be doing the transform in software and not be using the nice hardware on the card. The T&L hardware essentially can do the transform for free, since it is running on the graphics CPU and is probably overlapping with the vertex submission code that IDL is running at the same time. For awhile, only workstation graphics hardware could do this. Now the commonly available GEForce2/3 class cards have this capability as well.

Karl
RSI