Subject: Re: How to use IDLgrShader?
Posted by Jim Pendleton on Tue, 29 May 2007 00:01:39 GMT
View Forum Message <> Reply to Message

There are a number of examples of both vertex and fragment (pixel) shading in IDL64\examples\doc\shaders. Two built-in utility classes, idlgrshaderbytscl__define.pro and idlgrshaderconvol3__define.pro, can be found in the \lib subdirectory as well.

The IDL docs do not go into a great deal of detail about how to write a shader program in GLSL (OpenGL Shader Language). That is left as an exercise for the reader.

Here are a couple items to be aware of, though.

First, debugging a GLSL script is challenging. There is no "print" statement in GLSL to help you track your scripts' progress. There are a number of third-party packages that can assist you, however, such as RenderMonkey, available from ATI (ati.amd.com). NVIDIA also offers a debugging utility.

Second, you will notice that IDL provides for a software fallback mechanism if a graphics card doesn't support the shader extension. In this mode, OpenGL calls back into an IDL routine to get its vertex and/or pixel values rather than having the graphics card execute the operations. (See the documentation for the ::Filter method.) However, some tests have shown that it is often faster to simply perform the equivalent operation in IDL code without resorting to the IDLgrShader and a software fallback if the graphics card does not support shaders. For example an old-fashioned call to IDLgrImage->SetProperty, Data = BYTSCL(rawdata) will generally run faster than the software ::Filter callback, but the IDL code will often be significantly slower than the hardware acceleration (depending on the size of the image.)

Third, your graphics performance enhancement will be very dependent on the type of card you have, your CPU(s), and the type of operation you are attempting to accelerate. Not all cards are created equal and they are often optimized for one purpose over another. You may find that there are thresholds for image sizes below which performing calculations in IDL rather than on the card will result in improved display speed.

Some useful additional sources of information on shading are "The OpenGL Shading Language" by Randi J. Rost and the OpenGL Shading Language specification available at OpenGL.org.

Jim P.

<airy.jiang@gmail.com> wrote in message
news:1180338565.647666.123600@a26g2000pre.googlegroups.com.. .
> IDL 6.4 is available.A new IDLgrShader object provides a way to
> associate a shader program with the existing IDL graphic objects.That
> is seemed like a nice tool to increasing the speed of graphics
> rendering dramatically .But how to use it?anyone got the example
> source code,or the demo to show the effect of it?i'll very appreciate
> that someone can answer this question.
>