Subject: Re: iSurface lighting questions
Posted by Karl Schultz on Wed, 22 Jun 2005 21:19:16 GMT
View Forum Message <> Reply to Message

On Wed, 22 Jun 2005 13:26:10 -0500, Kenneth Bowman wrote:

> In article <pan.2005.06.22.17.55.59.969000@rsinc.com>,
> Karl Schultz <k____schultz@rsinc.com> wrote:
>
>> On Wed, 22 Jun 2005 09:40:56 -0500, Kenneth Bowman wrote:
>>
>>> Why does the default lighting for iSurface illuminate the *bottom* of
>>> the surface?
>>>
>>> And if I want to add a light manually to illuminate the top, why do I
>>> have to
>>> drag it to the *bottom* of the window?
>>>
>>>
>> Exactly what command are you using?
>>
>> What happens if you jsut do something simple like
>>
>> isurface, dist(40)
>>
>> Does it look different if you force software rendering?
>
> The problem appears to be that my z-coordinate decreases upward. Here
> is a simple example.
>
> x = 10*FINDGEN(37)
> y = -90 + 10*FINDGEN(19)
> z = REPLICATE(500.0, 37, 19) + 100.0*RANDOMN(seed, 37*19) ISURFACE, z,
> x, y, /NO_SAVEPROMPT, $
> /SHADING, $
> XTITLE = 'Longitude', $
> XRANGE = [0.0, 360.0], $
> XMAJOR = 5, $
> XMINOR = 2, $
> YTITLE = 'Latitude', $
> YRANGE = [-90.0, 90.0], $
> YMAJOR = 7, $
> YMINOR = 2, $
> ZTITLE = 'Pressure', $
> ZRANGE = [1000.0, 0.0], $
> ZMAJOR = 6, $
> ZMINOR = 1
>

> I would consider this to be a bug, but perhaps it is a feature. :-(
> The way an additional light behaves is, at least, unintuitive.

OK, thanks for the example - it helps a lot.

I'm fairly sure that this effect is related to the normals that IDL computes for the surface.

When the graphics system (OpenGL) computes the color of each vertex in your surface, it does so by doing a calculation using the light source direction vector(s), the surface normal at the vertex, and your eye position. This varies the color of the surface at each vertex according to this illumination model and then each surface facet is drawn with color interpolation using these computed vertex colors.

It sort of looks like the normals for the surface are pointing "down" in this example, away from the light, which would tend to make the top of the surface look dark, which is what you are seeing.

These surface normals are computed by IDL in the IDLgrSurface object. When you give IDLgrSurface a bunch of Z values, it generates a surface mesh which actually ends up in something a lot like an IDLgrPolygon. IDL computes the vertex normals from the mesh it generated from your Z data.

IDLgrSurface (and by extension, iSurface) is a sort of convenience object that makes some assumptions about what direction is "up" for your surface. It generates the mesh with vertex ordering and normal direction in concert with these assumptions. These steps and assumptions work well for most cases.

One could gain more control over all this by using IDLgrPolygon and generating your own mesh, with your own vertex ordering (clockwise or counter-clockwise) and normal direction.

I'm not yet certain if this is a bug. With respect to the surface normals, if you accept the condition that the surface normals are always going to be oriented so that the "top" of the surface faces in a positive Z (world) direction, then you are getting exactly what you asked for. IDLgrSurface could support a property that controls what side of the surface is the "top", from a lighting perspective.

Another way to attack the problem is to realize that you are flipping your scene, effectively, in iSurface and need to move the position of the lights to adapt to the change. I'm pretty sure that they are ways to locate the visualization objects programmatically. You could find all the lights and move them to where you want them.

Karl