Subject: Re: Object Graphics fonts
Posted by Karl Schultz on Wed, 09 Jun 2004 21:11:05 GMT
View Forum Message <> Reply to Message

"Michael Wallace" <mwallace.no.spam@no.spam.swri.edu.invalid> wrote in message news:10ceq7qsnl25ub3@corp.supernews.com...
>> Is it possible that using IDLgrWindow is giving you better results only
>> because the dimensions are larger? The default IDLgrBuffer size is 640x480.
>>
>> Aside from the frame buffer size difference, I don't see how hardware
>> rendering would produce a better display, where text is concerned. I could
>> see how turning on line or polygon or even full-screen AA might help lines
>> and polygons, but text is now drawn with texture maps and I'm not sure that
>> graphics cards should be messing with texture contents other than performing
>> the usual interpolations.
>
> Well, maybe I spoke too quickly. Using an IDLgrWindow gives me
> different results than using and IDLgrBuffer. The fonts look good in
> both, but the fonts are larger in the IDLgrWindow than they are in the
> IDLgrBuffer. I did make sure to have the DIMENSIONS keyword set to the
> same value for both of my tests. If I look at the differences between
> the two images, there are very subtle changes in pixel position (usually
> just one pixel off) for an axis or other lines, but there is a definite
> difference in the font size. This is why I made my previous comment
> about the fonts looking "better" in the IDLgrWindow... the fonts were
> slightly bigger and so they appeared cleaner. I didn't realize that the
> size of the fonts was just slightly larger.
>
> It does seem odd that only changing one line of code to use an
> IDLgrBuffer instead of an IDLgrWindow would cause these little one pixel
> differences and a slight font size difference. Is there an explanation
> for this or does this get chalked up to the mystery that is IDL?

The explanation is that the devices have different resolutions:

```
IDL> oWin = OBJ_NEW('IDLgrWindow')
IDL> oWin->GetProperty, Resolution=winres

IDL> print, winres

0.026437500 0.026416666

IDL> oBuff = OBJ_NEW('IDLgrBuffer')
```

IDL> oBuff->GetProperty, Resolution=buffres

IDL> print, buffres

0.035277778 0.035277778

IDL does its best to draw the glyphs at the requested size. A common size is 12 points and that does not mean 12 pixels, although on most displays the size of a pixel is pretty close to a point (1/72nd of an inch). IDL uses the device resolution (size of a pixel) to decide how many pixels to use to draw the glyphs. If the resolutions are different, then the number of pixels used will be different, in order to get the same physical size.

If your window pixel size is smaller than the buffer pixel size, as is the case above, then it would take more pixels to draw the same size glyph. That's why the text looked better in the window. Bitmap glyph rendering systems like FreeType do a much better job with more pixels in the glyph box. When you get down to really small boxes like 4 or 5 pixels high, the glyph quality really goes down.

You can set the RESOLUTION property of IDLgrBuffer to match that of the window.

Also, IDL does its best to get an accurate value for the resolution of the display, but some display driver interfaces don't return very accurate values and there's a lot of potential for inexact values. But they are pretty close. Printers tend to be more accurate in this area. And people don't tend to put their rulers up to the screen.
>> IDL 6.0 was the first release with the FreeType-based texture mapping
>> rendering for IDLgrText. We've made a few bug fixes and a lot of
>> improvements in this area for IDL 6.1. Without an exact testcase I can use
>> to compare 6.0 and 6.1 text rendering, I can't tell if Michael's specific
>> concerns are addressed or not. I looked through the bug database and found
>> that I fixed a problem that sounds very much what Michael is describing.
>> The problem would occur when the model space was scaled unequally and a
>> non-default baseline and updir was used. In any event, 6.1 may address the
>> problem.
>
> Well, since the fonts I was concerned with were Y axis fonts, they would
> have non-standard UPDIR and BASELINE. And in my code I would usually
> create the axis in one command and then use SetProperty to set the text
> of the axis. I don't know enough to know if anything I've done has

> caused the model space to scale unequally or not.
>
> Speaking of 6.1, when will be able to get our grubby little hands on it?
> ;-)

The beta program is almost over, so real soon now. But I'm not sure all the bug fixes mentioned above are in the beta code.

Karl
