Subject: Re: how do I create an image file from an object graphics window?
Posted by JD Smith on Fri, 23 Jul 2004 18:30:47 GMT
View Forum Message <> Reply to Message

On Fri, 23 Jul 2004 10:28:10 -0600, Karl Schultz wrote:

>
> "Haje Korth" <haje.korth@jhuapl.edu> wrote in message
> news:cdqth5qro1@aplcore.jhuapl.edu...
>> Holger,
>> I have used 'idlgrclipboard' object in the past, which can create
> Postscript
>> file. However, the implementation in 6.0 is still buggy. 6.1 will be
> better,
>> there may still be some issues with alpha blending. Any way, if your
>> view/scene is not too complicated it may work for you. A note on the
>> side: It amazes me that RSI worked out so many details on making object
>> graphics look pretty and totoally forgot to spend the time working on
>> creating descent quality output of the graphics. In order to get what I
>> want, I
> have
>> to write every code twice, once in object graphics for the screen and
>> then use direct graphics techniques to create the PS file. Not very
>> efficient.....
>
> Remember that the clipboard has both bitmap and vector modes. The bitmap
> mode captures the contents of the scene exactly as you would see it on the
> screen. You can also do pretty much the same thing by getting the data
> out of the grBuffer and grWindow objects.
>
> Yes, vector output in 6.1 is quite a bit better, but we still need to
> understand that vector output cannot possibly recreate all the graphical
> features that you might use on the display. Vector output systems (e.g.,
> PostScript, Windows metafiles) are not really "3D" in any way, while
> Object Gaphics obviously is a 3D system. It's difficult to map a system
> with high capabilities onto ones with lesser capabilities. For example,
> vector systems do not have depth buffers. IDL does a crude depth sort in
> vector output to approximate the effect of a depth buffer, but it won't
> sort things out completely. Similar restrictions apply for things like
> alpha blending.
>
> One of the main motivations for Object Graphics vector output was to
> reduce the size of the graphics output. In bitmap mode, even a simple
> plot with a few dozen lines and some text would require several MB of
> space, depending on the dimensions of the drawable, which seems silly when
> there is so little data actually in the plot. With vector output, the
> same data can be represented with a few dozen line plot commands and some
> text strings, which adds up to a PostScript file of 1K or so in length So

> vector output can be a big win when working with plots, charts, and other
> visualizations that are more "2D" than "3D" and don't use a lot of
> advanced rendering features. Bitmap output is better when you need to
> preserve all those "3D" qualities.

Speaking entirely without having used Object Graphics, but the problem with bitmap output is always the tradeoff between absurdly large sizes, and too coarse resolution. When object scenes contains lines, text, and other glyphs, the output will not be good if copied at screen resolution and printed or sent to a journal. Ideally, an output system could do a combination: take a bitmap "background" of everything which is too complicated for, e.g., Postscript to handle, and overlay then lines, text, glyphs, etc. as vector entities. It may not be trivial to divide items into non-handleable vs. handleable, but it would surely produce better results at smaller output sizes.

JD
