Subject: Transparency in object_graphics
Posted by dcleon@gmail.com on Tue, 13 Jan 2009 17:35:50 GMT
View Forum Message <> Reply to Message

All,
Hopefully I'm missing something obvious. IDL's help pages indicate that its possible to get a 4-channel image using the IMAGE_DATA keyword to IDLgrWindow:GetProperty and IDLgrBuffer::GetProperty methods. Yet, whenever I try I end up with a 3-channel (RGB) image despite the fact that I'm successfully using transparency in the images I'm producing.

Does anyone know if its possible to get 4-channel output through the IMAGE_DATA property ? What causes transparency to be disabled in IDLgrWindow or IDLgrBuffer when it was being used in the graphics tree being drawn ?

My goal is to output plots with transparent backgrounds using WRITE_PNG (I'm aware that this will require conversion to an indexed color model). Since the background color gets blended with the some of the graphics elements it doesn't work to simply include the background color in the TRANSPARENT array on the call to WRITE_PNG.

Thanks.
dave

Subject: Re: Transparency in object_graphics
Posted by Karl[1] on Wed, 14 Jan 2009 22:40:51 GMT
View Forum Message <> Reply to Message

Some devices support "destination alpha" which is where the alpha channel actually contains alpha data as dictated by the blend equation. These devices might return RGBA. And use of destination alpha is pretty rare. Others don't have destination alpha and so would return RGB. Having said that, I don't know if the IDL implementation would return the fourth channel, if it existed in the hardware. But, as you will see, it does not really matter.

Lack of destination alpha, when IMAGE_DATA returns RGB, does NOT disable transparency during rendering. Transparency is NOT getting "disabled" when you observe RGB data getting returned.

Most commonly used blend functions don't need or use destination alpha. What happens is that the SOURCE alpha is used to compute the colors in your blended/translucent/transparent output, resulting in simple RGB colors, according to the current blend function. That is, the alpha gets absorbed into the RGB and not pushed through to the

frame buffer in a separate alpha channel.

To solve your problem, if you have a consistent background color, you could use color keying to specify that all pixels of that color are transparent. I think PNG supports that, but I know you can specify a transparent pixel in GIF images.

Karl

On Jan 13, 11:35 am, "dcl...@gmail.com" <dcl...@gmail.com> wrote:
> All,
> Hopefully I'm missing something obvious. IDL's help pages indicate
> that its possible to get a 4-channel image using the IMAGE_DATA
> keyword to IDLgrWindow:GetProperty and IDLgrBuffer::GetProperty
> methods. Yet, whenever I try I end up with a 3-channel (RGB) image
> despite the fact that I'm successfully using transparency in the
> images I'm producing.
>
> Does anyone know if its possible to get 4-channel output through the
> IMAGE_DATA property ? What causes transparency to be disabled in
> IDLgrWindow or IDLgrBuffer when it was being used in the graphics tree
> being drawn ?
>
> My goal is to output plots with transparent backgrounds using
> WRITE_PNG (I'm aware that this will require conversion to an indexed
> color model). Since the background color gets blended with the some
> of the graphics elements it doesn't work to simply include the
> background color in the TRANSPARENT array on the call to WRITE_PNG.
>
> Thanks.
> dave