Subject: Re: Vector output of idlgrpolygon models
Posted by D D on Tue, 08 Nov 2011 19:58:04 GMT
View Forum Message <> Reply to Message

On Nov 8, 5:13 pm, Karl <Karl.W.Schu...@gmail.com> wrote:
> You might try looking at the REJECT property in IDLgrPolygon. If your surfaces are defined correctly so that the normals are correct, REJECT can be set to prevent drawing the polygons that face away from the viewer. This might reduce the number of unwanted polygons you are dealing with.
>
> Also might look at the VECT_SORTING keyword in IDLgrClipBoard::Draw(). Graphics hardware uses Z-buffers to take care of the hidden surface removal problem. The clipboard doesn't have such hardware and does coarse-grained sorting of the objects based on their depth instead. I can't remember if it uses the average Z of the entire IDLgrPolygon object, or the average Z of each triangle making up the polygon. If the latter is true, then that level of resolution may be good enough to sort out your surfaces. There will likely be problems where the toroids intersect, so look carefully there. The clipboard object won't split up intersecting triangles and draw just the visible pieces.
>
> A more robust solution would be to use a BSP tree to sort them out and take care of splitting intersecting faces, but that is a lot of work.

I activate hardware rendering by default on my draw widgets, IDL then
sets them to software if the machine doesn't have suitable hardware.
Indeed I would have thought that the hardware to file comparison would
be the most likely place for a difference to occur.

As most of my surfaces aren't closed setting the REJECT keyword to
anything other than 0 means I lose half of my surface. Additionally
many of the "hidden" polygons will have normals pointing towards (or
away from) the camera. It's a shame because at first I thought REJECT
was what I was after. It is an option if I enforce a fixed viewpoint
but this is not great.

VECT_SORTING again looks useful but only alters the order items are
drawn, not if they are drawn.

Currently I draw my surfaces as a single polygon object, I have tested
splitting each surface into the individual polygons and redrawing.
This makes the visualisation and manipulation a fair bit slower but
still doesn't allow automated hidden object removal with vect_sorting
or reject.

One (probably idiotic) thought i've just had is if it's possible for a
user to click in the window to select a polygon (i.e. IDL can take an
x-y position and tell you what you've clicked on) then you could
technically click every polygon you can see and mark it, then once
you've covered the screen toggle all the unmarked polygons HIDE

property. How you would go about this in practice i'm not sure, moreover I'd guess this is likely to take a considerable amount of time if one were to scan over the full window at the smallest resolution. I'm sure some optimisation could be made but this still sounds like it's probably not a good way to go (if it's even possible).

Any other suggestions would be much appreciated but I think it looks like there's no simple way to achieve this. It will either involve writing a routine, putting up with large files or invoking an external tool (i'm trying hidden object removal with CorelDraw at the moment but it doesn't like the number of objects I think).

Thanks,
David

(Going slightly off topic [well off IDL anyway] : Searching for a solution to this I came across the C library GL2PS which apparently does this hidden object removal and is a routine for creating Postscript files from opengl commands. As I have little knowledge of opengl and no knowledge of C i've not got anywhere with it but it would certainly could form a nice output object! There is optional support for it in Paraview [which can read vrml] so this is one possible route.)