Subject: Re: mesh clipping
Posted by Karl Schultz on Wed, 28 Aug 2002 23:09:16 GMT
View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:akitis$20o6$1@nntp6.u.washington.edu...
>
> "lyubo" <lzagorch@cs.wright.edu> wrote
>
>> Rick, you were right. I really want to slice the mesh up interactively
>> and that's why I was trying to clip it to a plane.

One question to ask is if you want to actually clip your model - the data,
or just provide a visual clip.

You can easily clip the model with MESH_CLIP, but I think the OP said in the
first posting that merging them was too slow. Would it be possible to avoid
the merge and just display the clipped pieces? Is it important to merge the
pieces for some reason??? I don't know your data, but I can imagine many
circumstances where you can just display each part in its own IDLgrPolygon
and end up with something that looks the same as a single merged mesh.
Hopefully your data is small enough so that MESH_CLIP is still fast enough
to be interactive.

Visually, there is very little difference between displaying a (wire) mesh
with one or with several IDLgrPolygon objects.

For example, if you had one vertex list and one connectivity list with just
triangles in it:

```
verts = FLTARR(3,100)  ; 100 verts
conn = LONARR(4 * 50)  ; 50 triangles

; fill in arrays
...
; create objects

oPoly1 = OBJ_NEW('IDLgrPolygon', verts, conn[0:99])
oPoly2 = OBJ_NEW('IDLgrPolygon', verts, conn[100:*])
```

The visual appearance of these two meshes should be pretty indistinguishable
from a single mesh formed from the entire 'conn' list with a couple of
exceptions. You'll probably see a seam if you are doing filled polygons
with smooth shading. The seam would be easier to notice if the normals of
the polygons on either side of the seam are very different from each other.
But if you are doing wire frame, you should be alright. And if you used
alpha blending, the order makes a difference, as Rick is pointing out.

And yes, you can use the viewport and Z clip planes to do some visual clipping, but that would be pretty limited.

>> I guess alpha blending
>> will be faster but the question that I have here is how can I use alpha
>> blending with a mesh? I thought that I can apply alpha blending only to
>> texture mapped polygons, by using an alpha image as texture. With
>> the mesh I don't have any texture. I will try to find examples on the net,
>> I just wanted to thank you for your reply.
>
> Ahh, you have a wire mesh....
>
> You are *mostly* correct in thinking that you need to work with texture
> mapped solid polygons to use alpha blending. In IDL 5.5 there is a bug that
> allows you to texture wireframe models. But, before we go there, you need
> to texture your polygon first...
>
> For now, work with a solid polygon. Let's assume you want to draw your
> polygon in grey. Create a instance of IDLgrImage with this texture data:
>
> imagedat = [[180,180,180,255],[180,180,180,0]]
>
> Use this image object to texture your polygon.
>
> The trick will be setting up the texture coordinates. Your texcoords array
> will be a 2xn array where n is the number of verticies in your mesh and each
> coordinate pair maps a pixel in your image to a vertex in your mesh. So,
> for verticies you want "on" you will give it a texcoord of [0,0] and for
> verts you want off, [0,1] (or is it [1,0]? Well, you get the idea).
>
> There are a few things to watch out for. One is that if I remember
> correctly, I don't actually use texcoords of 0 or 1 to assign pixels at the
> edge of my texture. I ended up using 0.001 and 0.999. Unfortunatly I
> can't remember why...
>
> A second issue will be that you will not have a cleanly defined edge along
> your slices. IDL will blend from opaque to transparent giving you a "soft"
> edge. This may be a result of the type of shading used though..
>
> And then there is the order in which the polygon is drawn. It has to be
> drawn back to front. And if you rotate it 180 degrees you draw it back to

> front, which turns out to be front to back. I usually end up slicing my
> mesh into a +z portion and -z portion and then keep track of where the
> camera is and flip the two objects in my model when the camera crosses the
> xy plane.

I think that this is going to be a real show-stopper if we are talking about general meshes. In the most general sense, you'd have to sort your vertices by VIEWPORT Z (not model Z) if the orientation of the model changes for a frame. (By "sort", I mean arrange your connectivity list so that the polygons that are most distant from the viewer are drawn first.) Unless the data is constrained to be something more simple, like a sphere or being convex, this is a very difficult problem to solve.

We got away with this in the pimento case because it was a simple sphere.

I've seen some apps chop models into 8 "octants" and change the order they are drawn based on orientation to the viewer, which I guess is a pretty decent approximation.

But in general, the alpha approach is going to be a pretty hard way to do this. I'd try to use the MESH_CLIP approach. Perhaps you can create a mesh with very few polygons in it (with MESH_DECIMATE) to use while your user is sliding a clip plane interactively. When they "let go", display the final clipped mesh with the original model. There may be other techniques.

>
>
> Ahh, the wire mesh... Like I said, IDL 5.5 has a bug where wire mesh
> polygons can be textured. It just doesn't work as expected. But you
should
> be able to get it to work. Start with the solid and get that working...

This problem is fixed in 5.6. IDL 5.5 did not support texturing point or line polygons, but if you tried it, the texture coordinates were being ignored which made this "feature" hard to use. You can do some really cool things with this in IDL 5.6, if you like modulating colors along a line.

>> As far as my graphics adapter, I use Nvidia GeForce3 on a P4 2.0GHz
>> dual processor with 512Mb Ram platform. Which graphic adapters
>> support rendering of volumes?
>
> That I can't answer. We don't do volumes so I haven't ever investigated
> this. I can tell you that the high end consumer cards like your GF3 are
> optimized for gaming. They concentrate on fill rate first, then polygon
> count. If there is any support for volumes it is WAY down the list.

It is to the point where some companies sell dedicated volume-rendering graphics adapters that use special hardware for volumetric rendering.

Volumetric rendering is a completely different approach to rendering as compared to polygonal rendering, in the same way ray-tracing is also different from polygonal rendering. The volume renderer built into IDLgrVolume uses a software ray-casting approach to create the image, which is pretty compute-intensive. OpenGL acceleration has no impact on rendering IDL volumes, except when blitting the (2D) result to the screen.

---