Subject: Re: Vector output of idlgrpolygon models
Posted by Karl[1] on Wed, 09 Nov 2011 17:27:49 GMT
View Forum Message <> Reply to Message

I was going to point out that Select could be called for each pixel, but you figured that out :-).

But I also didn't point that out because I still don't think it is a perfect solution.

Did you actually send the resulting polygon list to vector output and inspect the results?   I think that rendering the resulting polygon list to the display would look pretty good, since you're getting help from the depth buffer. But that won't be the case on a vector device.

If two of your surfaces (toroids) intersect, there are going to be some faces that intersect with each other.

```
   EYE
    |
    V

  \     /
   \   /
    \ /
^    \/
|    /\
Z   /  \
X - - >
```

Your algorithm would (correctly) select both surfaces and put them both in the list to send to the vector device. That device is going to render then both in the order you supply them and you'll see one surface or the other near the intersection. The part of the second surface to draw that is behind the first drawn surface will draw on top of the first surface, which is incorrect.

But perhaps your data does not do this and you may be fine.

The GL2PS approach is very interesting. It uses the same technique that the IDL Clipboard uses (OpenGL Feedback buffer) and sorts things in a similar way if you pick the simple sort option. I had also mentioned using BSP trees and I note that GL2PS offers a BSP tree sort option. This would address the problem I illustrated above with the intersecting surfaces. These two surfaces would each be split at the intersection line, resulting in four surfaces. The two "back" pieces would not be in the remaining list and would not draw, leaving the two "front" pieces to represent the correct rendering of the intersection. Might still be worth a look if you need this sort of precision.