Subject: Re: How to avoid texture-map overlapping?
Posted by Karl Schultz on Tue, 15 Nov 2005 17:24:36 GMT
View Forum Message <> Reply to Message

On Mon, 14 Nov 2005 16:46:26 -0800, uniqrat wrote:

> Hi,all
> Here is my program:
>
snip
>
> I want to make the beam transparent.So I put a texture on it. But it is
> overlapped.
> How can I map it uniformly?
> Thanks a lot.

Oooh, you are hitting the classic translucent polygon sorting problem.

Search the newsgroup archives for "pimento" (yes, really) and strings like "BSP", "transparency", and "polygon", etc...

The basic explanation is that when you are drawing translucent meshes like this, you have to draw them back to front in order to get the correct visual effect.

When you can rotate an object like yours with XOBJVIEW, there are going to be few orientations where it will look correct.

In the general case, you'd have to sort your mesh and reorder the polygons in it every time the view angle changes so that the polygons draw back to front. I may have posted an example of an experimental BSP polygon object that does this sort of thing.

In some special cases, like if the mesh is a sphere, you can create a small number (like 4 or 8) of IDLgrPolygon objects, each with the individual polygons sorted for one of 4 or 8 orientations, and then select with the HIDE property the one with the orientation appropriate for the current viewing angle. This isn't as hard as it sounds, and might work for your "ray", since it has topology similar to a sphere, if you can get rid of that dimple in the front.

Anyway, look through the newsgroup archives, as this has been covered quite a bit.

Karl

Subject: Re: How to avoid texture-map overlapping?
Posted by Rick Towler on Tue, 15 Nov 2005 18:50:00 GMT
View Forum Message <> Reply to Message

uniqrat@163.com wrote:
> Hi,all
> I want to make the beam transparent.So I put a texture on it. But it is
> overlapped.
> How can I map it uniformly?
> Thanks a lot.

In addition to what Karl has suggested, you can also break your beam up into lengthwise sections, then order the sections in your model based on your point of view. Depending on how many sections you use, you may still have a few rendering issues but they should be minimal.

The easiest approach would be to code your beam as an object. A child of IDLgrModel. The init method would create the beam sections and add them to the "self" model. The other key method would be the order or sort method that would accept a 3 element vector (the view vector) and would order the sections in the self model based on the view vector.

You would then need to create your own viewer which would calculate the view vector (vector from the viewer to the center of your beam object) and pass that to your beam object on each draw.

I have done a lot of work with transparency and rendering order. Out of the options that Karl has posted, this is one of the better ones. The ideal solution would be the BSP polygon object but it needs a lot of work/optimization and I have never found the time.

You may decide that all of this isn't worth it. Depending on the project, I certainly have :) But if you decide to go this route, I have a lot of code that can get you started, but there will be a bit of overhead learning my viewing system. Let me know if you're interested.

-Rick

Subject: Re: How to avoid texture-map overlapping?
Posted by uniqrat on Wed, 16 Nov 2005 00:07:53 GMT
View Forum Message <> Reply to Message

Thanks a lot for the advice from both of you.
Although I have learned programming with IDL for several years, I am a fresher in this problem. So if you can provide some example in this field, I believe I can learn more and discuss this problem in near future, ^_^.

It's still a nut for me to crack. Jim Pendleton sent me a letter in
which he told me add a keyword "reject=1" . It quite appears
overlapping can not be seen. But I am not sure if overlapping is
avoided.
Anyway, I get the expected effect :)
Thank you all again.

Subject: Re: How to avoid texture-map overlapping?
Posted by Rick Towler on Wed, 16 Nov 2005 01:00:32 GMT
View Forum Message <> Reply to Message

uniqrat@163.com wrote:
> Thanks a lot for the advice from both of you.
> Although I have learned programming with IDL for several years, I am a
> fresher in this problem. So if you can provide some example in this
> field, I believe I can learn more and discuss this problem in near
> future, ^_^.

I'll see what I can dig up.

> It's still a nut for me to crack. Jim Pendleton sent me a letter in
> which he told me add a keyword "reject=1" . It quite appears
> overlapping can not be seen. But I am not sure if overlapping is
> avoided.

This is a cheap fix ;)

If you have read some of the other posts regarding this topic what you will have gathered is that the line or "seam" you see is the boundary between the part of the beam where both the back and front parts of the beam have been rendered and where only the front has been rendered. If you have a black background and a white beam, the lighter section of the beam is actually the "correct" rendering as it is comprised of the color from the background + the color from the back of the beam + the color from the front of the beam. By setting REJECT=1 the polygons that comprise the back side of the beam are hidden. In effect you are only seeing the front half of the beam. The end result is a beam without the seam but also one that is darker than expected.

If you never look at the beam ends then this doesn't matter. But (as in my case) you regularly look at the beam end you expect the beam "face" to be darker than the beam "walls" and these color differences define the beam edges. Maybe a minor point but most people picked up on it quickly when playing with my application.

At the same time this may work perfectly well for you and you don't have to invest a lot of time taking the complicated approach. I couldn't

help it. I'm a stickler for details... :)

-Rick
