Volume Rendering (3)

Hardware Texture Mapping Methods

Texture Mapping

Texture Mapping (2)

Each texel has 2D coordinates assigned to it.

assign the texture coordinates to each polygon to establish the mapping

Texture based volume rendering

- Render each xz slice in the volume as a texture-mapped polygon
- The texture contains RGBA (color and opacity)
- The polygons are drawn from back to front

Tex. Mapping for Volume Rendering

Remember ray casting...

Turn off the z-buffer; Enable blending
For (each slice from back to front) {
  - Load the 2D slice of data into texture memory
  - Create a polygon corresponding to the slice
  - Assign texture coordinates to four corners of the polygon
  - Render and composite the polygon (use OpenGL alpha blending)
Some Considerations... (2)

What if we change the viewing position?

That is okay, we just change the eye position (or rotate the polygons and re-render).

Until ...

Some Considerations... (3)

You are not going to see anything this way...

This is because the view direction now is Parallel to the slice planes

What do we do?

Some Considerations... (4)

What do we do?

• Change the orientation of slicing planes
• Now the slice polygons are parallel to XZ plane in the object space
• We need to reorganize the input textures
• Reorganize the textures on the fly is too time consuming. We might want to prepare this texture sets beforehand

Some Considerations... (5)

When do we need to change the slicing planes?

When the major component of view vector changes from z to y

Problem

Object-space axis-aligned method can create artifact: Poping Effect

There is a sudden change of viewing slicing (and thus the sampling rate) then the view vector transits from one major direction to another. The change in the result image intensity can be quite visible

Some Considerations... (6)

Major component of view vector?

Normalized view vector (x,y,z) -> get the maximum one

If x: then the slicing planes are parallel to yz plane
If y: then the slicing planes are parallel to xz plane
If z: then the slicing planes are parallel to xy plane

-> This is called (object-space) axis-aligned method.

Therefore, we need to keep three copies of data around all the time!!
Solution

Use Image-space axis-aligned slicing plane:
- the slicing planes are always parallel to the view plane

Image-Space Axis-Aligned

Arbitrary slicing through the volume and texture mapping capabilities are needed:
- Arbitrary slicing: this can be computed using software in real time

This is basically polygon-volume clipping

Image-Space Axis-Aligned (2)

Texture mapping to the arbitrary slices

This requires 3D Solid texture mapping

Input texture: A bunch of slices (volume)
Depending on the position of the polygon, appropriate textures are mapped to the polygon.

Solid (3D) Texture Mapping

Now the input texture space is 3D
Texture coordinates \((r,s) \rightarrow (r,s,t)\)

Pros and Cons

Advantages:
- Higher quality
- No popping effect

Disadvantages:
- Need to compute the slicing planes for every view angle
- 3D texture mapping is not supported by most of graphics hardware except high end ones (although people are working on it right now: ATI Radeon card, $250, starts to support it)

Both 2D or 3D hardware texture mapping methods can not compute Shading on the fly. The input textures have to be pre-shaded.