REYES

REYES

- You might be surprised to know that most frames of all Pixar's films and shorts do not use a global illumination model for rendering!
- Instead, they use REYES

REYES

- Renders Everything You Ever Saw
- Developed by Pixar and still(?) used as primary architecture for Pixar's Renderman implementation, prman
- Example of a "practical" rendering system.

Goals of REYES

- Complex models (in an era of balls and planes!)
- Model diversity (fractals, graftals, particle systems)
- Shading Complexity
- Minimal Ray Tracing (Use textures instead, focus on geometry)
- Fast…needed for animations (feature length film '87, took 1 year, 3 min/frame)
- Image quality (no jaggies, aliasing, Moiré patterns)
- Build for flexibility

REYES Design Principles

- Use natural coordinates
  - Texturing in object space
  - Visibility in image space
- Exploit hardware capabilities (parallelism)
- Common representation for geometry
- Locality
  - Geometric - one object at a time
  - Texture - read texture once and only when needed
  - Eliminates ray tracing and radiosity
- Linearity - time f(model size)
- Support (unlimitedly) large models
- Back door to allow use alternatives to render some
- Efficient access for texture maps

REYES

- REYES uses a basic Z-buffer
- Z-buffer algorithm
  - In addition to pixel values, array of depths at each pixel is maintained
  - Image space but object based algorithm
  - Only intensity of closest object is maintained.

REYES

- Z-buffer

```
Z-Buffer Algorithm
    Given
        List of polygons {P1, P2, ..., Pn }
        An array z-buffer[x,y] initialized to -∞
        An array Intensity[x,y]
    begin
        for each polygon P in the polygon list do {
            for each pixel (x,y) that intersects P do {
                calculate z-depth of P at (x,y)
                if z-depth > z-buffer[x,y] then {
                    Intensity[x,y] = intensity of P at (x,y)
                    z-buffer[x,y] = z-depth
                    }
                }
            }
        Display Intensity array
    end
```

Figure 2.1: The Z-Buffer Algorithm

REYES – Major Components

- Reliance on texture mapping
- Jitter supersampling
- Micropolygons

REYES – Reliance on Texture Mapping

- All means are taken to avoid ray tracing/radiosity
- Texture maps used for
  - Environment mapping
  - Reflections
  - Bump/displacement mappings – normal, coordinate modifications
  - Shadows – depth information from light source
- Especially efficient when considering rendering of multiple frames.

REYES - Texturing

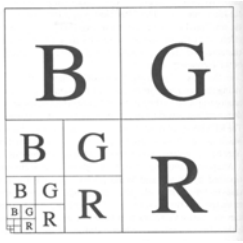
- Texture mapping efficiencies
  - Prefiltering of texture maps
  - Have texture resolution match that of patch resolution.
- Requires lots of work up front, which eliminates the need to do it at runtime.

REYES - Texturing

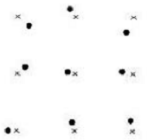
- Prefiltered textures
  - Textures stored as “pyramid” of images at various resolutions
  - Resolutions between levels of pyramid are done via interpolation
  - MIPMaps / FlashPix (Kodak)

Texture Mapping

- Mipmaps



REYES- Jittered Super-sampling



- Same idea as in distributed ray tracing
- Each pixel is subdivided into 16 subpixels
- Exact location of each subpixel sample determined by jittering.
- Z-buffer is kept at subpixel resolution
- Pixel value determined by averaging of subpixels comprising it.

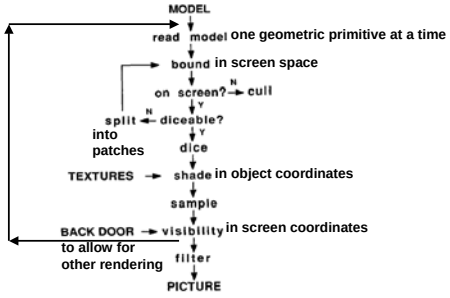
REYES - Micropolygons

- Shading values are calculated on a single geometric entity, the micropolygon
- Flat shaded quadrilaterals, half of a pixel on each side
  - Why half of a pixel?
- Each micropolygon is represented by a single color.

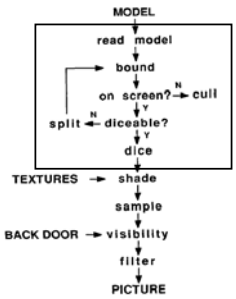
REYES - Micropolygons

- ***Dicing*** - Geometric primitives and patches must be converted to micropolygons
- Dice along boundaries in natural coordinate system of primitive
- Done in eye space although it uses an estimate of size on screen
- Primitives may need to be converted to patches before dicing

The REYES Algorithm

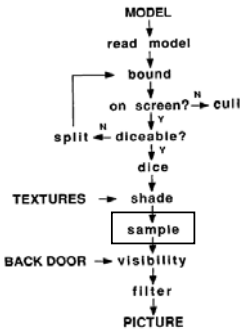


REYES - Reading in Object



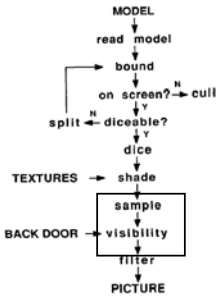
- Object computes its bounds in screen space
- Can be culled if not on screen
- Is split into patches, if necessary
- Diced, if on screen

REYES Shading



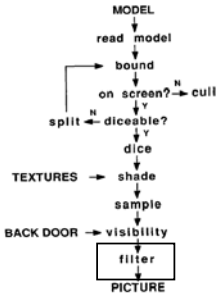
- Each micropolygon (stored in a grid) is shaded and textured.
- Note that shading is done before visibility testing -> extra work
- Object-based algorithm, i.e.., done for whole object without regard for other objects
- Enables use of vector machines/vertex shaders(?)
- Avoids reloading textures
- Controls subdivision coherency
- No clipping
- No need to deal with perspective issues
- Can use displacement maps

REYES Visibility/Sampling

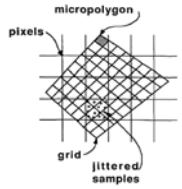


- Jitter sampling performed
- Visibility stored in z-buffer.
- Compositing is done, if required, at this step

REYES Picture Generation



- Construct pixel values from subsamples
- Additional filtering as required.



REYES Example

- Rendered at 1024x614
- 6.8 million micropolygons

Figure 6. 1986 Pixar Christmas Card by John Lasseter and Eben Ostby.

REYES

- Quick and effective rendering using classical CG techniques
  - No ray tracing
  - No radiosity
- Designed for efficiency

Renderman and Rendering

- Renderman is a rendering interface standard
  - Does not define how rendering is to be performed.
- ***prman*** and ***BMRT*** are both Renderman Compliant Renders:
  - ***prman*** uses REYES
  - (Not clear what the latest ***prman*** uses)
  - ***BMRT*** supports Ray Tracing and Radiosity

Renderman and Rendering

- What Renderman does define:
  - C-based API for describing a scene
  - Associate file format (RIB)
  - Shader language for procedural lighting, shading, modeling
- Complexity and generality is a result of the shader language.

Renderman and Rendering

- Lighting Constructs in RSL
  - Illuminance (point, axis, angle)
    - Computes all light arriving at a point within a given cone (axis and angle define the cone)
    - Could be from light sources or other objects
  - Implementation is up to the renderer thus
    - Could be determined using radiosity
    - Could be determined using ray tracing
    - Could be determined by indexing into a texture map.

Renderman and Rendering

- When writing a Renderman shader
  - Illuminance can be used regardless of the method of computation method.
- Separation of shading from a given rendering technique.

Rendering

- Summary
  - Rendering Equation
  - Ray Tracing
  - Radiosity
  - Two-Pass Global Illumination Method
  - Photon Mapping
  - REYES + Renderman
- Efficient global illumination is still a hot research topic.

Questions