Notes

- Assignment 1 is out, due October 12
  - Inverse Kinematics
  - Evaluating Catmull-Rom splines for motion curves
- Wednesday: may be late (will get someone to leave a note if so)

REYES

- Invented at Lucasfilm (later Pixar) by Cook et al. SIGGRAPH '87
- Geometry is diced up into grids of micropolygons (quads about one pixel big)
- Each micropolygon is "shaded" in parallel to get a colour+opacity (RGBA)
- Then sent to "hiding" to determine in which point samples it makes a contribution
- Each point sample keeps a sorted list of visible points, composites them together when done
- Filter blends point samples to get final pixels

Why REYES

- No compromises in visual quality (antialiasing, motion blur, transparency, etc.) compared with e.g. OpenGL
- Very efficient in optimized implementations (e.g. PhotoRealistic Renderman from Pixar) with predictable runtimes compared to raytracing
  - Handle complex scenes robustly
- Huge flexibility from shading architecture
  - Modern GPU’s are catching up now…

Shading

- A shader is a small program that computes the RGBA of a micropolygon
- Writing shaders to get the right look is a hugely important part of production
- Shaders probably need to know about surface normal, surface texture coordinates, active lights, …
  - E.g. ubiquitous Phong model
- But can do a whole lot more!

Displacement shaders

- Can actually move the micropolygon to a new location
  - Allows for simple geometry (e.g. sphere) to produce complex results (e.g. baseball)
  - A flexibility no other renderer allows so easily
- Also could perturb surface normal independently (bump mapping)---so even if geometry remains simple and fixed, appearance can be complex

Shadows and lighting

- Part of the shader will figure out if point is in shadow or not
  - Typically using precomputed “shadow maps” but could also use ray-tracing!
- Huge flexibility for cheating (“photosurrealism”)
  - Each object can pick and choose which lights illuminate it, which shadows it’s in, change the location/direction/intensity of lights, ...
  - More control than e.g. ray-tracing

Surface shading

- Implement whatever formula you want for how light bounces off surface to viewer
- Can look up texture maps or compute procedural textures
- Can layer surface shaders on top of each other
- Reflections: can do ray-tracing for high accuracy, but usually use “environment mapping”
  - Assume object is small compared to distance to surroundings
  - Then look up reflected light in a texture based on direction (2D), not position+direction (5D)
  - Can capture or synthesize environment map

Atmospheric shading

- Can further adjust micropolygon colour to account for fog, atmospheric attenuation (blue mountains), etc.
- Could do it with more accurate ray-tracing
- Usually a simple formula based on distance to camera works fine

RenderMan

- Pixar defines a standard API for high quality renderers
  - Think OpenGL, but with quality trumping performance
  - “Postscript for 3D”
- Today, many software packages implement (at least part of) the Renderman standard
- Two parts to the API
  - Direct calls, e.g. RiTranslate(0.3, 0.4, -1);
    - Very similar to OpenGL
  - RenderMan Interface Bytestream (RIB) files: save calls in text file for later processing
- Also a shading language (based on C)

RenderMan resources

- Pixar website has official API reference
- “The RenderMan Companion”, Upstill
- “Advanced RenderMan”, Apodaca and Gritz
- SIGGRAPH course notes
- www.renderman.org

Differential Rendering

- For adding CG shadows in a real scene
- Put in rough estimate of true scene geometry, colour, material etc.
- Render rough estimate once without CG object, once with CG object:
  - Difference is shadow (or caustics, ...)
  - Composite onto real scene (addition)
- Then render CG object alone
  - Composite "atop"

Image-based lighting

- For getting CG lights just right (can be easily faked though) or environment map for reflections
- Take high-dynamic range (HDR) image of silver ball
- Do the math to get the environment map
- Reference: Paul Debevec, "Rendering Synthetic Objects into Real Scenes…", SIGGRAPH'98
  - www.debevec.org

Example