Advanced Topics in Global Illumination

Assignments

- Checkpoint 7
  - Due Monday
- Final Code
  - Due Wednesday
- RenderMan
  - Due Monday, Nov 8th (note change in date)

Projects

- Approx 22 projects
- Listing of projects now on Web
- Presentation schedule
  - All projects scheduled!
  - (Whether requested or not!!!)

Projects

- Exam week presentations
  - Friday, Nov 19th
  - 12:30 – 2:30
  - 70-1620

Announcement

- Maya courses in School of Design
  - 2001-552/2001-752-01
    - Interactive 3D Animation
  - 2001-572-02
    - Shaders
    - (Intro to Maya shading)
- Contact Marla Schweppe (mkspph@rit.edu) for more info.

Logistics

- Final Report
  - Introduction
  - Approach Taken
  - Implementation Details
  - Results
  - Appendix/Code

Logistics

- Final Report
  - Introduction
    - Overview / Description
  - Approach Taken
    - If Application:
      - Platform, Language, Libraries, Tools, Compilers used
      - inputs, outputs, controls
    - If Rendering/Animation
      - Renderer used
      - How motion, shading, or textures were achieved
    - Algorithms/Techniques Employed

Logistics

- Final Report (cont'd)
  - Implementation Details
    - Overall System Architecture – dataflow diagram + explanation
    - Overall Program Architecture – list and description of major software components, classes, modules and how these modules interact.
    - If group project: Explain how integration of individual contributions were achieved.
    - If shader used: high level description of shader w/parameters

Logistics

- Final Report (cont'd)
  - Results
    - Screen shots of application/rendering
    - If application: User documentation
    - If study: conclusion of your study
    - Lessons learned
    - Possible future enhancements
  - Appendix
    - Code / Shader code
    - Save the trees: On floppy or CD if possible.

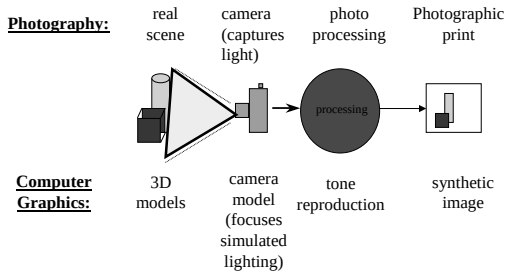
Logistics

- What I'll be looking for
  - Report
    - The process of design and implementation of the project
    - Architecture rather than code
      - Leave code details in the code comments
      - I will look at code style!
  - User Documentation
    - I'll need to know how to run it.

Logistics

- To summarize: Final Report
  - Introduction
  - Approach Taken
  - Implementation Details
  - Results
  - Appendix/Code
- Questions?

Computer Graphics as Virtual Photography



Advanced Topics in Global Illumination

- A Two Pass Global Illumination Method
- Reyes

Philosophical Questions

- Is it necessary to calculate global illumination accurately? (or when is it?)
- Is photorealism a desirable goal and to what limits should we chase it?

Bi-directional Reflectance Function

- Remember -- BDRF:

$$BDRF = f_r(\phi_i, \theta_i, \phi_r, \theta_r)$$

- Represents the physical reflective properties of the surface
- Ratio of reflected or transmitted in outgoing direction based on energy arriving from an incoming direction.
- Wavelength dependent, but does not account for energy exchange between wavelength bands.

Two Pass Method

- Remember the Phong Illumination Model

$$L(V) = \underbrace{k_a L_a}_{\text{ambient}} + \underbrace{k_d \sum_i L_i (\mathrm{S}_i \bullet \mathrm{N})}_{\text{diffuse}} + \underbrace{k_s \sum_i L_i (\mathrm{R}_i \bullet \mathrm{V})^{k_e}}_{\text{specular}}$$

- Most complete local illumination approximation.

Components of Phong Illumination

- Specular
  - Dependent upon incoming and outgoing direction
  - Mirror-like reflection
- Diffuse
  - Assumes equal reflectance in all directions
  - BDRF is constant

Which Global Illumination Technique?



[Heckbert84]



[Cohen85]

1. Why a Two-Pass Global Illumination Method?

- Ray Tracing
  - Good for specular reflections
  - Bad for diffuse reflections
  - View Dependent
  - Computationally intensive
- Radiosity
  - Good for diffuse reflections
  - Bad for specular reflections
  - View independent
  - Even more computationally intensive
- A two-pass method gives us the best of both worlds

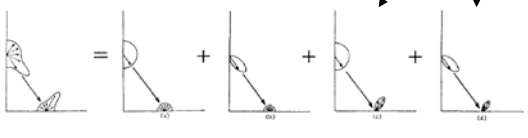
Two Pass Method

- Pass 1: View independent (radiosity)
- Pass 2: View dependent (ray tracing)

Two Pass Method

(std ray tracing)

- Four mechanisms of light transport between two surfaces



BRDF | Diffuse to Diffuse | Specular to Diffuse | Diffuse to Specular | Specular to Specular

(std radiosity)

[Wallace87]

Two Pass Method - Preprocess

- Pass 1: Use hemi-cube radiosity method
  - View independent pass
  - Will provide diffuse reflections for all objects
  - Already handles diffuse-to-diffuse reflections
  - Basic algorithm is modified to account for
    - Diffuse reflections from specular sources
    - Diffuse transmission
    - Handles curved surfaces (Wallace)
      (Sillion extended to all surface types)

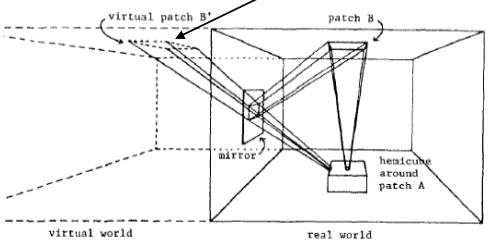
Two Pass Method - Preprocess

- Recall:
  - Form factors give fraction of light emitted at one patch that arrive at another patch
- Modification:
  - Increase form factor for patch to account for additional light arriving at the point via specular reflection.
  - And transmission

Two Pass Method - Preprocess

Like calculating reflection

- Form Factor Modification



[Wallace87]

Two Pass Method – Preprocessing Results

For each surface, the diffuse intensity emitted by the surface has been calculated, whether light is received from diffuse or specular sources.

Two Pass Method - Pass Two

- Pass 2: Ray Tracing
  - View dependent pass
  - Calculation per pixel limits work
  - Ray tracing already accounts for specular-to-specular
  - Must be modified to accurately account for diffuse-to-specular

Two Pass Method - Pass Two Diffuse-to-specular

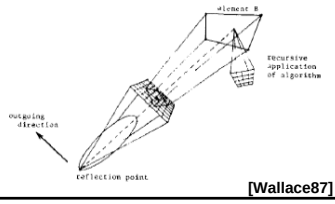
- Ideally, would consider incoming light from all directions that contribute to specular component.
- In reality, the light contributing MOST HEAVILY to specular reflection comes from direction of incident ray.
- This allows limiting integration to solid angle over which the weighted intensity is significant, rather than whole hemisphere.

Two Pass Method - Pass Two The Reflection Frustum

- Light is collected from a solid angle surrounding the ray of incidence
- The smaller angle, the more mirror-like (material properties)
- Solid angle is approximated by point samples
- Light to be considered for specular reflection is determined by a weighted average of the sampled rays

The Reflection Frustum

- Sampling intensities arrive through reflection frustum
- Incoming specularity is obtained recursively, reducing resolution each level



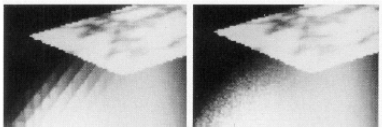
[Wallace87]

The Reflection Frustum

- Acts like integration with BDRF providing pre-computed importance weights
- Used a 10 x 10 pixel array
- Used z-buffer
- Diffuse component - Gouraud shading
- Anti-aliasing performed by jittered rotation of frustum
- Solution similar to that used in distributed ray tracing

Two Pass Method

- Reflection frustum - example



(a) (b)

Figure 6. (a) Aliasing due to uniform point sampling of the incoming intensities.

(b) Result of rotating the reflection frustum around its central axis by a random amount for each image pixel (enlargement of lower left corner of image in Figure 11b).

[Wallace87]

Two Pass Method - Example



Perfectly diffuse floors



Floors after "polishing"

[Wallace87]

Two Pass Method - Summary

- Rendering in Two Passes
- Pass 1: Radiosity
  - Considers diffuse-to-diffuse reflections
  - Considers specular-to-diffuse reflections
- Pass 2: Ray Tracing
  - Considers diffuse-to-specular reflections
  - Considers specular-to-specular reflections
- The total light emitted at a point is
  - Diffuse component, as calculated in Pass 1, plus
  - Specular component, as calculated in Pass 2

Results

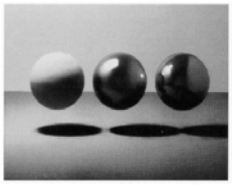


Figure 7. Stainless steel spheres with finishes ranging from completely diffuse to highly specular.

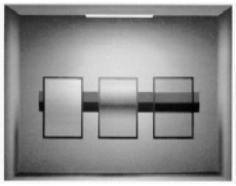


Figure 8. Three panes of translucent glass. The left-hand panel demonstrates ideal diffuse transmission.

[Wallace87]

Results

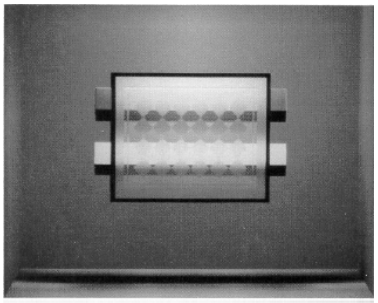
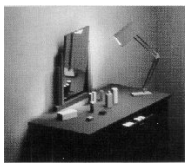


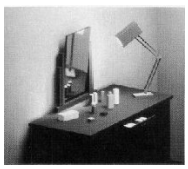
Figure 9. Etched glass.

[Wallace87]

Two Pass Method - Example



Without diffuse



Diffuse to diffuse added



Full solution

[Wallace87]

Two Pass Method - Example

- Questions?
- Break?

2. 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

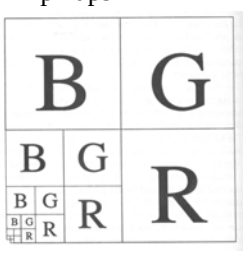
- Texture mapping efficiencies
  - Prefiltering of texture maps
  - Have texture resolution match that of patch resolution.
- Requires lots of work up front, which eliminates the need to do it at runtime.

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)

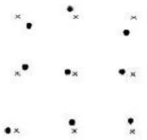
Texture Mapping

- 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

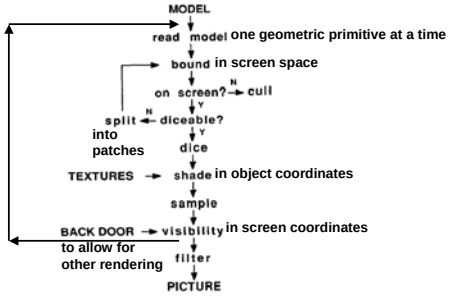
- Shading values are calculated on a single geometric entity, the micropolygon
- Flat shaded quadrilaterals, half of a pixel on each side
  - Why half of a pixel?
- Each micropolygon is represented by a single color.

REYES - Micropolygons

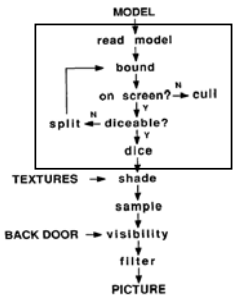


- ***Dicing*** - Geometric primitives and patches must be converted to micropolygons
- Dice along boundaries in natural coordinate system of primitive
- Done in eye space although it uses an estimate of size on screen
- Primitives may need to be converted to patches before dicing

The REYES Algorithm

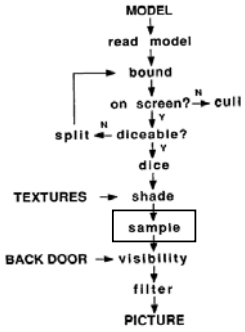


REYES - Reading in Object



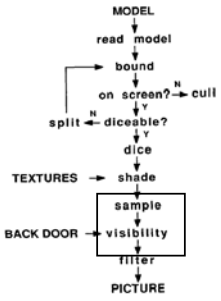
- Object computes its bounds in screen space
- Can be culled if not on screen
- Is split into patches, if necessary
- Diced, if on screen

REYES Shading



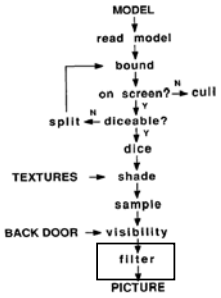
- Each micropolygon (stored in a grid) is shaded and textured.
- Note that shading is done before visibility testing -> extra work
- Object-based algorithm, i.e.., done for whole object without regard for other objects
- Enables use of vector machines/vertex shaders(?)
- Avoids reloading textures
- Controls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

REYES Visibility/Sampling

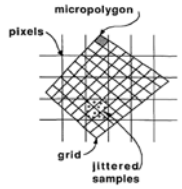


- Jitter sampling performed
- Visibility stored in z-buffer.
- Compositing is done, if required, at this step

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
  - REYES + Renderman
- Efficient global illumination is still a hot research topic.

Current Trends in Global Illumination Research circa 1998

- Radiosity
  - Clustering rather than individual surfaces
  - Real Time Object Motion
  - Meshing Techniques
- Tracing
  - Particle Tracing/Photon Mapping
  - Bi-directional Ray Tracing
- Tone Mapping/Reproduction
- Image-Based Modeling, Rendering, and Lighting

That's all folks

- Thanks for playing!
- Next week:
  - Student presentations
- One last course of business
  - Course evaluations.