Precomputed Radiance Transfer

Harrison McKenzie Chapter
Two Principle Areas:

- **Real-time Graphics**
  - Games
  - Rasterizing Triangles

- **Global Illumination**
  - Movies
  - Ray Tracing
  - Radiosity
  - Photon-Mapping
Quality vs. Speed

- Global Illumination strategies are capable of rendering images of amazing quality by simulating object interactions.

- Unfortunately, modeling throngs of photons bumping into things can take a long time.
- And we want to render our scene at least 30 times a second!

- Conventionally, we just cheat and hope nobody notices.
  - Still have problems with good soft shadows and reflections.
Precomputed Radiance Transfer does a little bit of both

- Model the way that an object maps incoming radiance to out-going radiance
- Divides the lighting function for an object into two parts
  - A Spherical Harmonic projection of the function that maps the scene's light to color
  - A Spherical Harmonic projection of all the lights in the scene at a particular object
Advantages

- Allows us to do a lot of the hard work in the lighting calculation as a preprocess for each vertex

- As part of that pre-process, we can 'Bake' in certain characteristics about how the object interacts with itself
  - Self-Shadowing

- Final rendering procedure is very fast -- the lighting integral reduces to an N term dot product of SH coefficients
PRT Algorithm Overview

- Preprocess Mesh
  - Create sampling vectors
  - Compute the input radiance at each vertex in each sample direction
  - Add light to SH representation

- Render Mesh
  - For each vertex, take the dot product of the SH transfer vector at that point, and the light's SH representation
Spherical Harmonics

- Spherical Harmonics are a set of basis functions which functions can be represented with

- We can *project* functions into the SH basis

- We can *reconstruct* functions using a linear combination of the basis functions
Spherical Harmonics 2

- The SH representation's ability to capture detail is tied to the number of "bands" which are used.
- More bands = more fine grain detail.
- More bands = more coefficients which need to be stored.
Computing Incident Radiance

- At a point:
  - For every sample vector:
    - Find how that vector's light would affect the color
    - Normalize by sphere's area
  - Add each normalized value to the SH coefficient for each band
Rendering

- Project lights in the scene to SH, relative to the model
  - Similar to projecting points on the mesh
  - Far less lights than points, computationally ok
- Light at any vertex is:
  - Dot product of the light SH projection and the point's SH representation
  - N-dimensional dot product, where N bands * bands
Work so far:
My work in this area

- I'm moving the calculations over to the GPU for the rendering step.

- Multiple problems to solve
  - Data Representation (SH coefficients)
  - Interpolation Strategies
  - Dataset size and compression

- Most research implementations perform the lighting calculations on the CPU

- Real-time games move these calculations onto the GPU to increase parallelism
Current GPU Implementation
Questions?