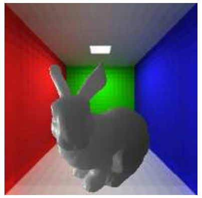
Graphics Hardware 2003

GPU Algorithms for Radiosity and Subsurface Scattering

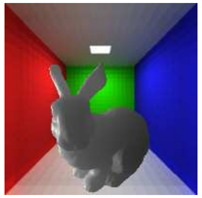
Nathan A. Carr, Jesse D. Hall, John C. Hart

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN



Matrix Radiosity & Diffuse Subsurface Scattering

- Both can be solved using discretization of the scene into patches.
- Both involve computation of light transport between pairs of patches:
  - **Link:** a connection between pairs of patches that may undergo light interaction.
  - **Form Factor:** percentage of light leaving a patch that reaches another patch in the scene.
- Diffuse subsurface scattering is an easier problem:
  - single integration of light across links.
- Radiosity requires iteration of moving light across links until some form of convergence is reached.
  - PDE equilibrium simulation (iterated light integration)



Radiosity: Overview

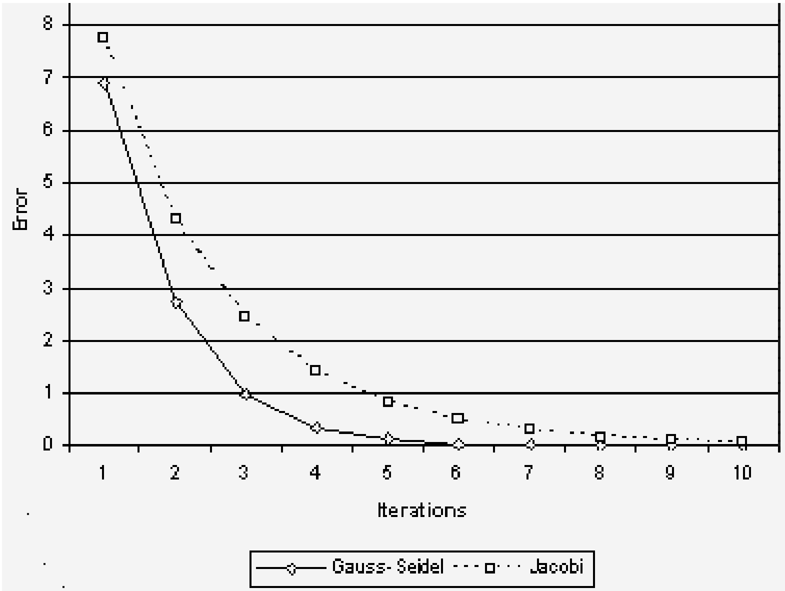
- Goal is to solve radiosity matrix $\mathbf{K}B=E$ on GPU
  - Experiment to test recent GPU floating-point horsepower
  - Leaves results on GPU for display
- Use Jacobi iteration for solution
  - Converges slower than Gauss-Seidel
  - But more parallelism than Gauss-Seidel
- Use natural representation
  - Matrix is 2D texture, vectors are 1D textures
  - Complexity limited by max Pbuffer size (2048 x 2048)

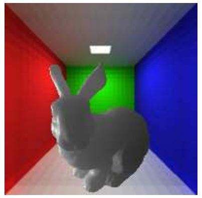
Jacobi v. Gauss-Seidel

- Jacobi iteration
  - Classical: $B_i^{(k+1)} = E_i - \Sigma_{j \neq i} \mathbf{K}_{ij} B_j^{(k)}$
  - Dependence free: $B_i^{(k+1)} = E - \mathbf{K}B^{(k)} + B^{(k)}$

$\mathbf{K}_{ii} = 1$

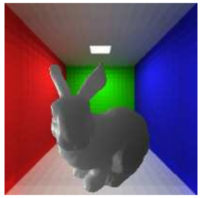
- Gauss-Seidel
  - Needs dependence: $B_i = E_i - \Sigma_{j \neq i} \mathbf{K}_{ij} B_j$
  - But converges 2x Jacobi!
- GPU Gauss-Seidel
  - *n* passes (Kruger & Westermann S03)
- GPU Jacobi
  - *n*/254 passes (unrolled)





Radiosity: Details

- Each Jacobi iteration requires multiple passes due to fragment shader instruction limits
- First: $R_i = \sum_{j=1}^{n} \mathbf{K}_{ij} B_j$ $(\lceil n/254 \rceil$ passes)
- Finally: $B_i' = B_i + E_i - R_i$ (1 pass)
- Each output element is computed in parallel
- Could interpolate to vertices on GPU



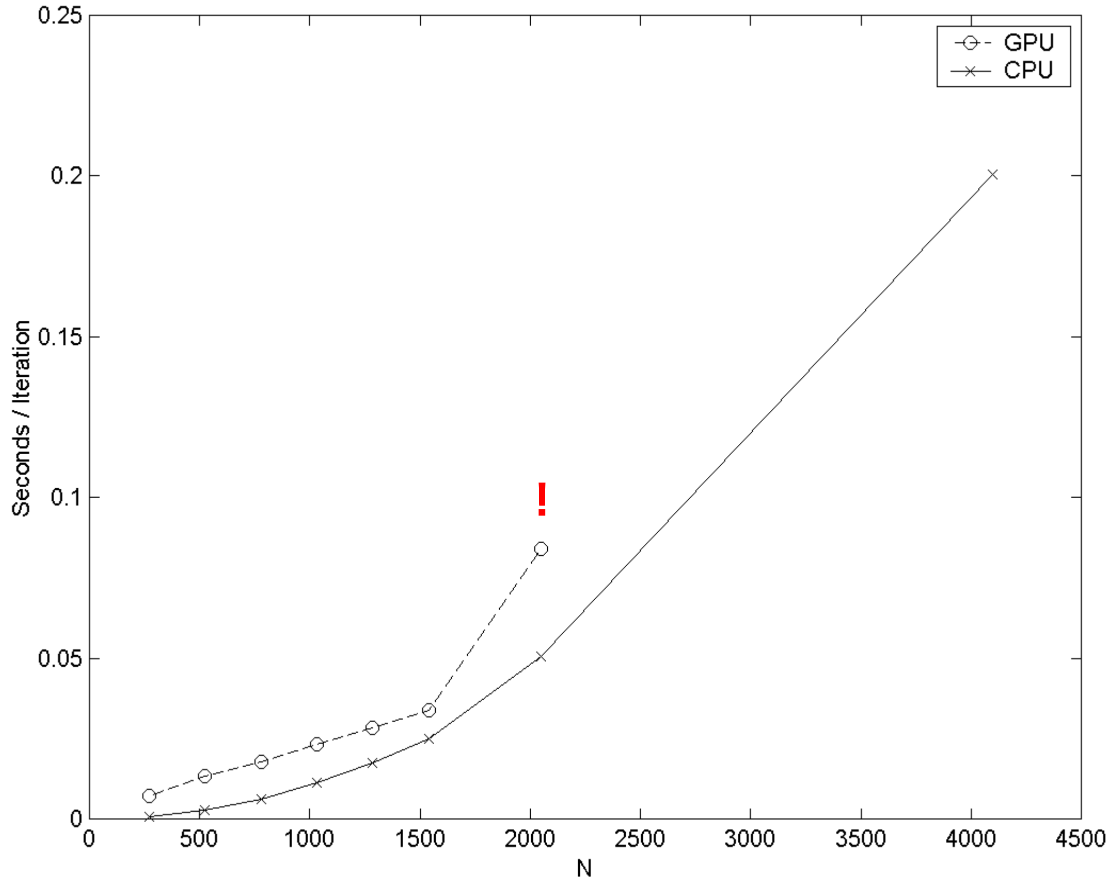
Radiosity: Results

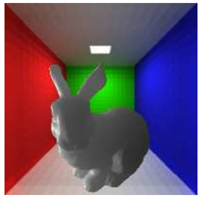
CPU: Athlon 2800+

- Gauss-Seidel
- 40 iter/s, 190M flops
- Bandwidth-limited

GPU: GFFX 5900 Ultra

- Jacobi
- 30 iter/s, 141M flops
- Compute-limited

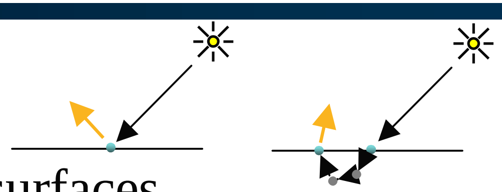




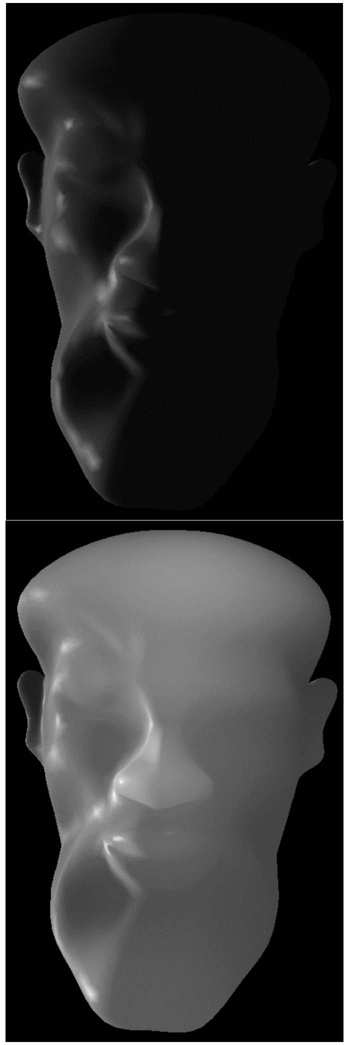
Radiosity: Conclusion

- Currently slower than on CPU
  - Compute speed increases faster than bandwidth
- Better organization needed for complex scenes
  - Could also make interpolating results easier
  - Might improve performance (caching)
- Other radiosity methods
  - GPU Progressive Refinement (http://www.cs.unc.edu/~coombe/research/radiosity)
  - Hierarchical Radiosity on the GPU?

Subsurface Scattering

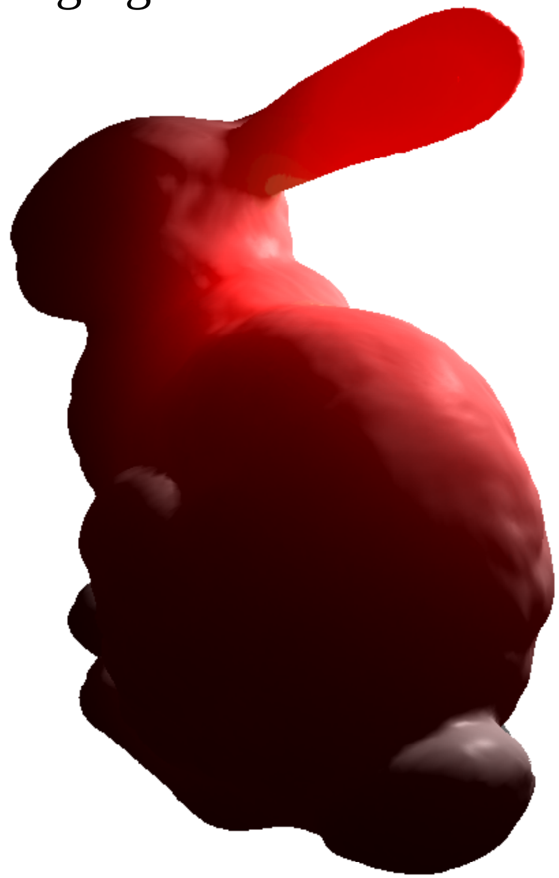


- BRDF Model
  - Inaccurate for many non-metal surfaces
  - Assumption: Light leaves a surface at the same point it impacted the surface.
- Developing A New Model…
  - Hanrahan & Kreuger S93 (single scattering)
  - Jensen *et. al.* S01 (complete model- BSSRDF)
  - Jensen *et. al.* S02 (diffuse multiple scattering, hierarchical)
- Towards Interactive Rates…
  - Lensch *et. al.* PG02 (atlas, radiosity-like)
  - Hao *et. al.* I3D03 (per-vertex local scatter)
  - Sloan *et. al.* S03 (pre-computed radiance xfer)



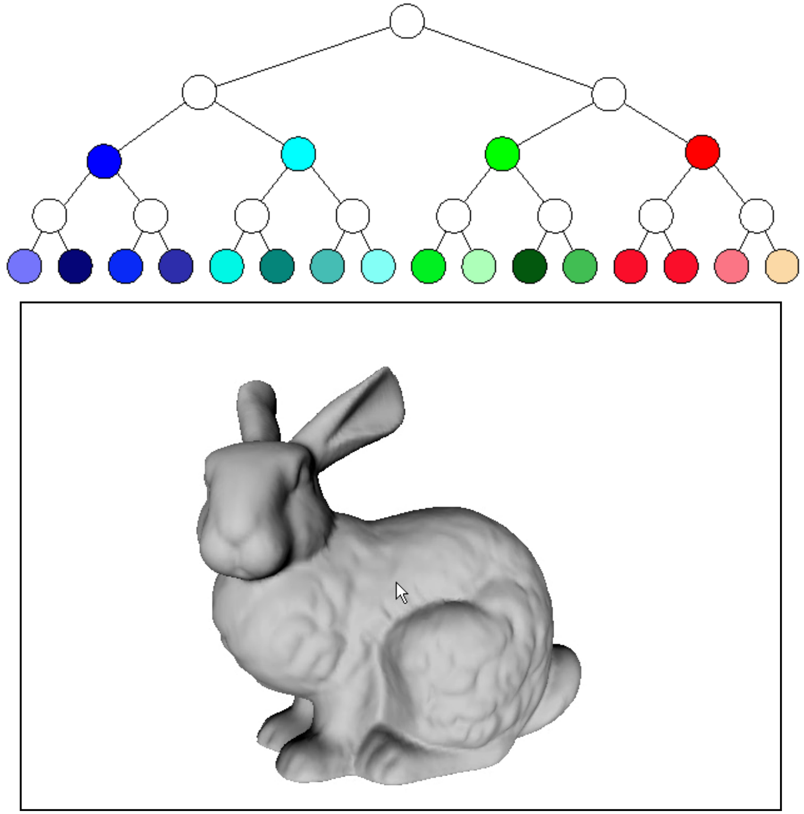
Our Method: Advantages

- Diffuse multiple scattering (ala Jensen *et al.* S02)
  - Removes dependence between incident and exiting light direction
  - Reduces dimensionality of the problem
  - Accurate to with a few percent
- Hierarchical
- Fully GPU based
  - Fragment shader implementation
  - Integrates with automatic mip-map generation hardware
- Decouples shading frequency from tessellation and screen resolution
  - “Per-texel” shading



Surface Hierarchies and Parameterization

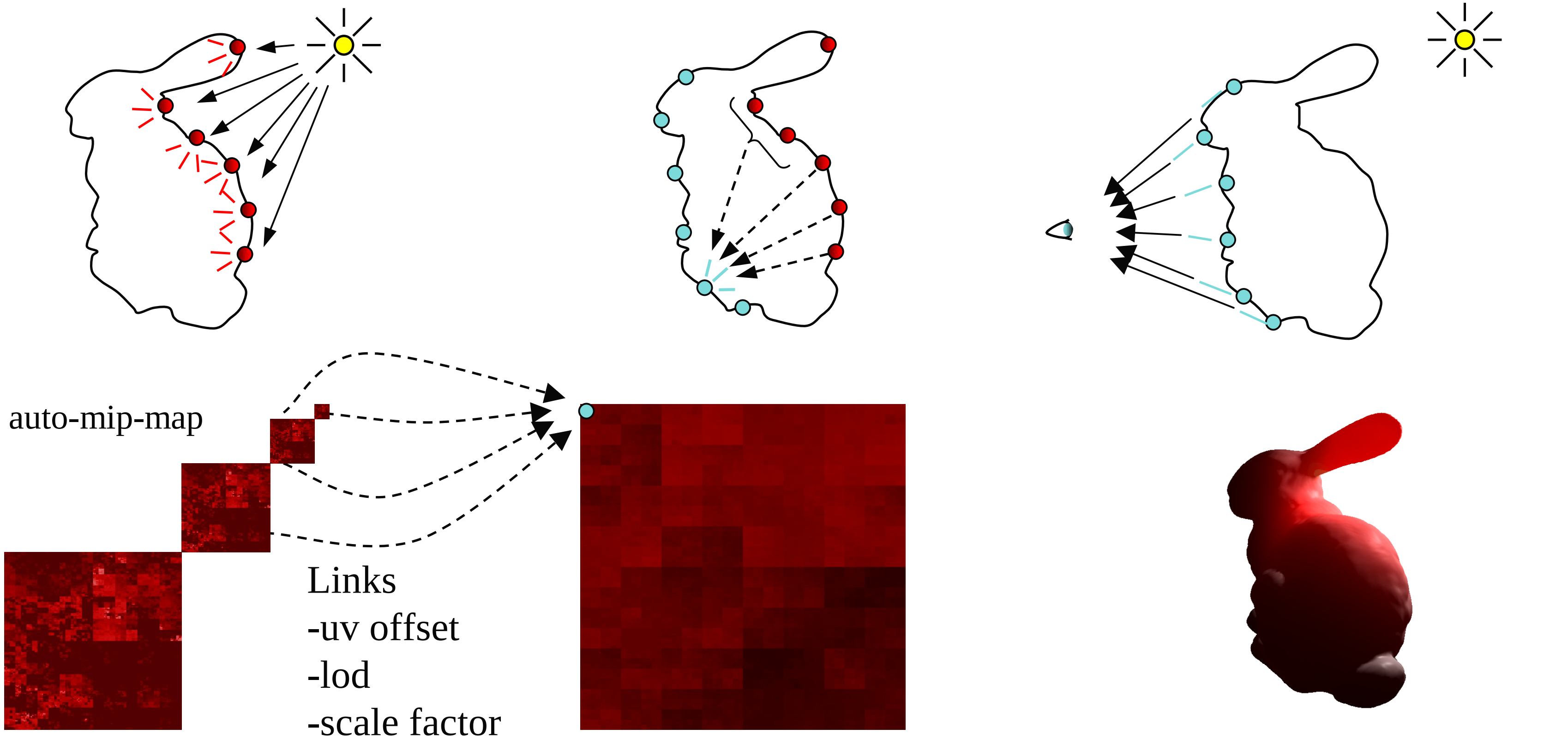
- Provides Surface Hierarchy
- Obeys GPU rasterization rules
  - Render directly into texture atlas using render to texture No seams!
  - Automatic mip-map generation. (Fast integration).
- Supports GPU filtering for anti-aliasing
  - Bilinear
  - Mip-mapping



Mip-mappable Texture Atlas

Carr & Hart. Meshed Atlases for Real-Time Procedural Solid Texturing, TOG 2002.

Overview



Pass1: Transmitted Radiance Map	Pass 2: Scattered Irradiance Map	Pass 3: Final Result

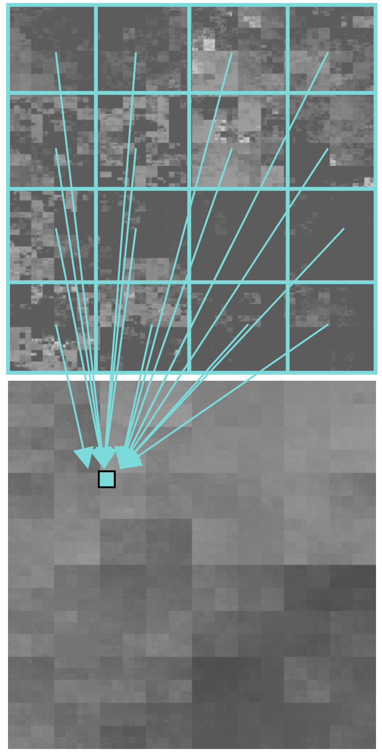
Pre-process

Form factor calculation:

- Run a simulation
  - Monte-carlo, ray-marching
  - Support for non-homogenous material
- Use Jensen’s analytic approximation (SIGGRAPH 2001)
  - Assumes surface is locally flat
  - Easy to implement
  - Results look good!

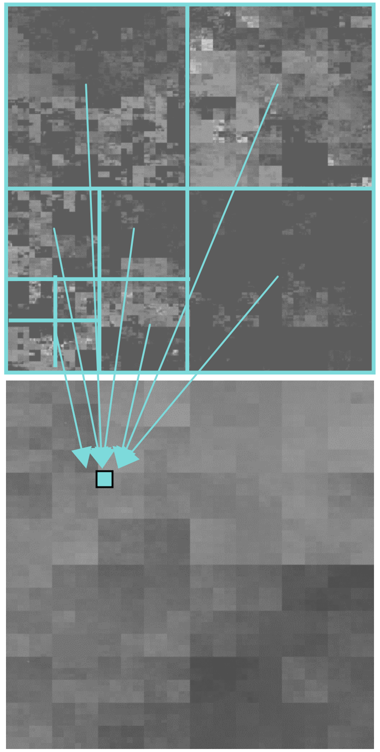
Pre-process: Uniform Links

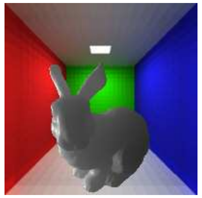
- Each texel *i* needs to represent a *link* to all other texels *j*
- Instead link texel *i* to a cluster *j*
- *Fij* ⇔ factor between texel *i* and cluster *j*
- Store *Fij* records at each texel
- LOD, uv cluster location same for all texels
  - Store as fragment program constants
- Only requires storage of form factor records per-texel



Pre-process: Adaptive Links

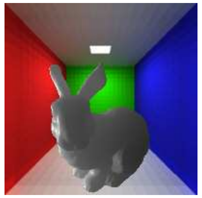
- Create links adaptively per texel.
- Benefit:
  - Higher accuracy for fewer links possible.
  - More efficient..
    - Subsurface scattering has exponential decay. May not need to store links from to all surface regions
- Downside:
  - Varying uv offset and LOD per texel. More texel records required
  - Care must be taken in choosing link locations to avoid seam artifacts



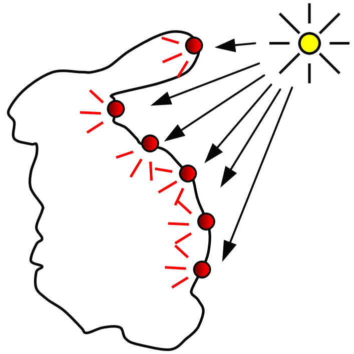


Implementation Details: Storing Links

- Required texture space for link information is significant.
  - Adaptive Link Locations:
    - uv offset
    - LOD –which mip-map level
    - form factor
  - Static Links (LOD, uv offset in constant registers)
- Vector Quantize form factors
  - 256 rgb element code book – stored as 1D texture
  - Each form factor is reduced to an 8 bit value.
  - Additional texture lookup required in fragment shader
- Place uv offsets + lod in lookup table
  - (offset + lod) may be reduced to 8 bit value
  - Extra texture lookup required

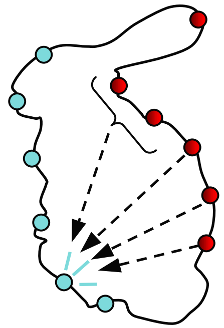


Pass 1: Radiosity Map

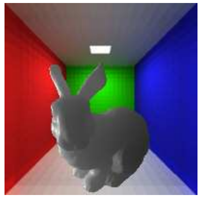


- No restriction on lighting model/method
  - Shadow maps, shadow volumes.
  - Bump mapping
  - Environment maps
  - Pre-computed radiance transfer. (Spherical Harmonics).
  - Monte-carlo ray-tracing
  - Radiosity
- Resolution of the map may be chosen arbitrarily (performance versus quality)
  - We tried both 512x512 and 1024x1024
  - We found you can get away “cheap”.
    - 512x512 map
    - Per-vertex lighting, etc…

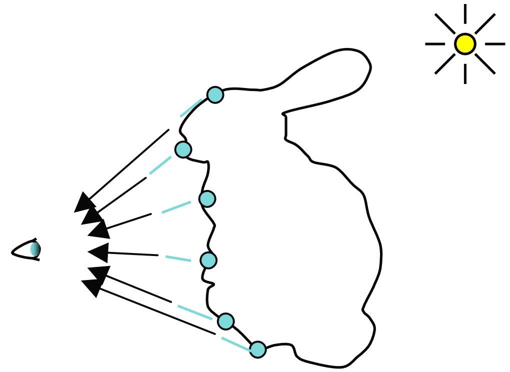
Pass 2: Scattered Irradiance Map



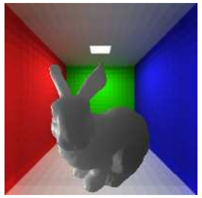
- Size of map may be arbitrarily chosen (quality versus performance).
  - Subsurface scattered irradiance tends to be low frequency.
  - We used 512x512, and 1024x1024
- Most expensive pass
  - Many texture lookups required per-texel
    - High bandwidth cost
    - Adaptive links more expensive



Pass 3: Final Rendering

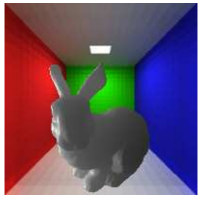


- Compute the incident lighting on the mesh.
  - Do higher quality rendering (e.g. per-pixel lighting)
- Texture map (add) the results from pass 2 (scattered irradiance map)
  - modulated by Fresnel.



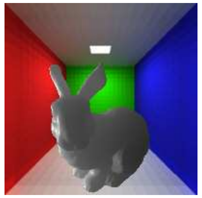
Results…

- A real-time demo..
  - 512x512 map
  - Static Links
    - 16 (4 megabytes)
    - 64 (16 megabytes)



Future Work

- Subsurface scattering on dynamically deforming models.
- Adaptive refinement on the GPU
  - Dynamic link creation and patch subdivision
  - More efficient exploration of light paths
- Single Scattering on the GPU



Acknowledgements

Work supported by:

- NVIDIA Corp.
- NSF ITR ACI-0113968