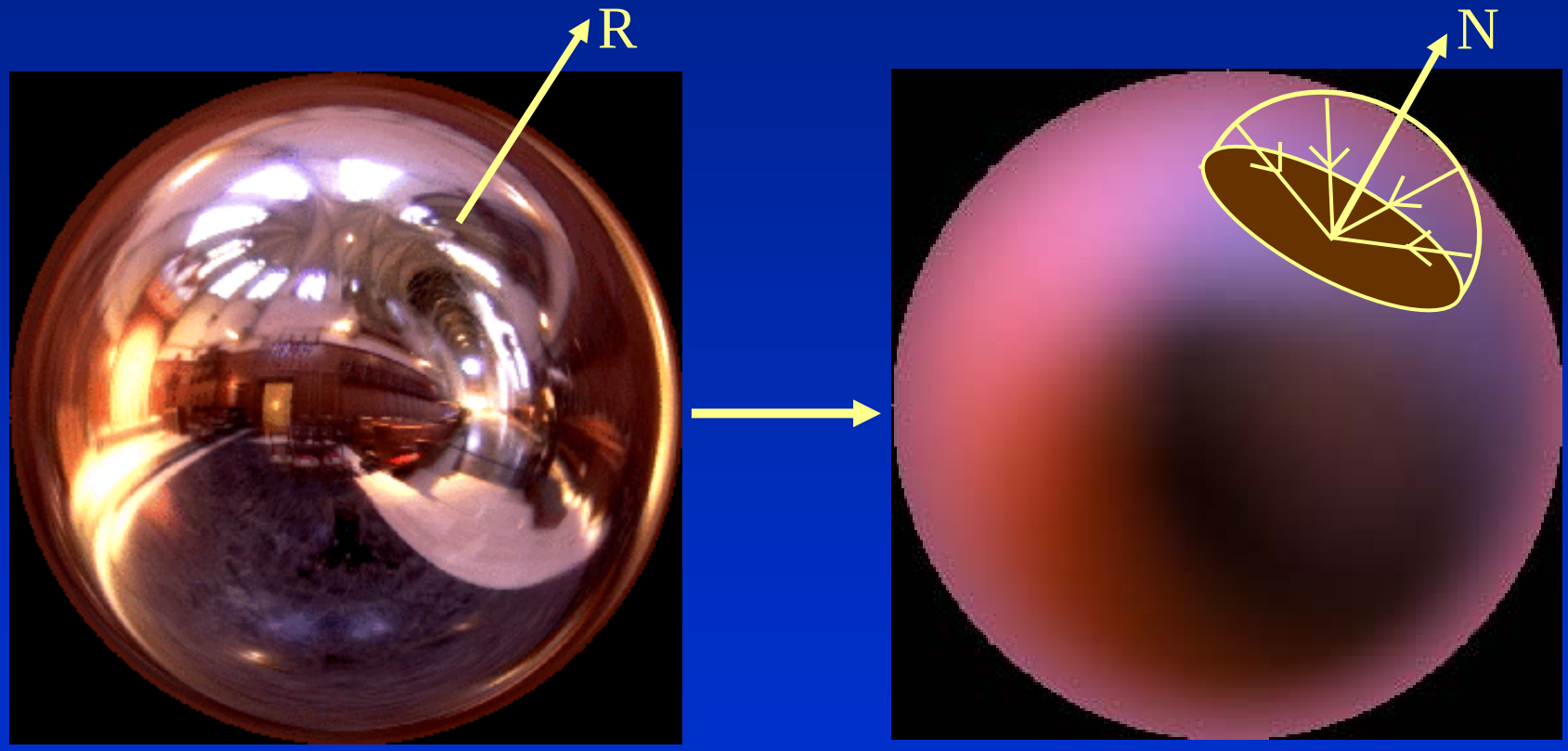
An Efficient Representation for Irradiance Environment Maps

Ravi Ramamoorthi Pat Hanrahan

Stanford University

SIGGRAPH 2001

Irradiance Environment Maps



Incident Radiance
(Illumination Environment Map)

Irradiance Environment Map

Assumptions

- Diffuse surfaces
- Distant illumination
- No shadowing, interreflection

Hence, Irradiance is a function of surface normal

$$E(\mathbf{n}) = \int_{\Omega(\mathbf{n})} L(\omega)(\mathbf{n} \cdot \omega) d\omega$$

Diffuse Reflection

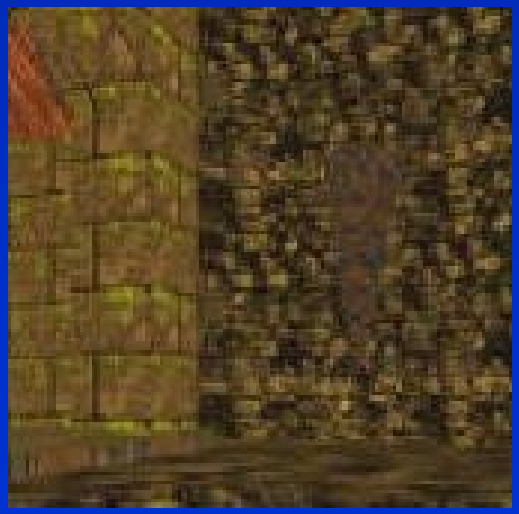
$$B(\mathbf{x}, \mathbf{n}) = \rho(\mathbf{x}) E(\mathbf{n})$$

Radiosity
(image intensity)

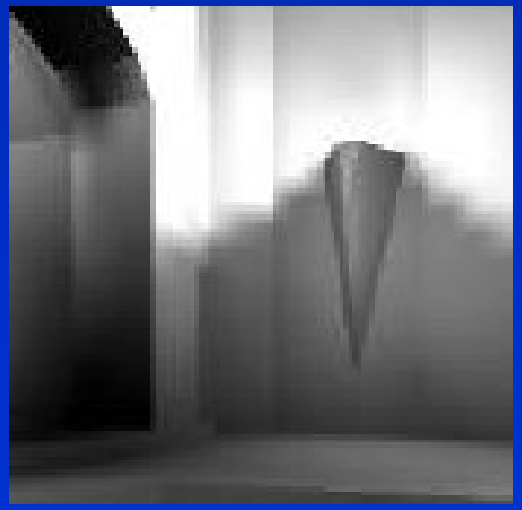
Reflectance
(albedo/texture)

Irradiance
(incoming light)

=



×

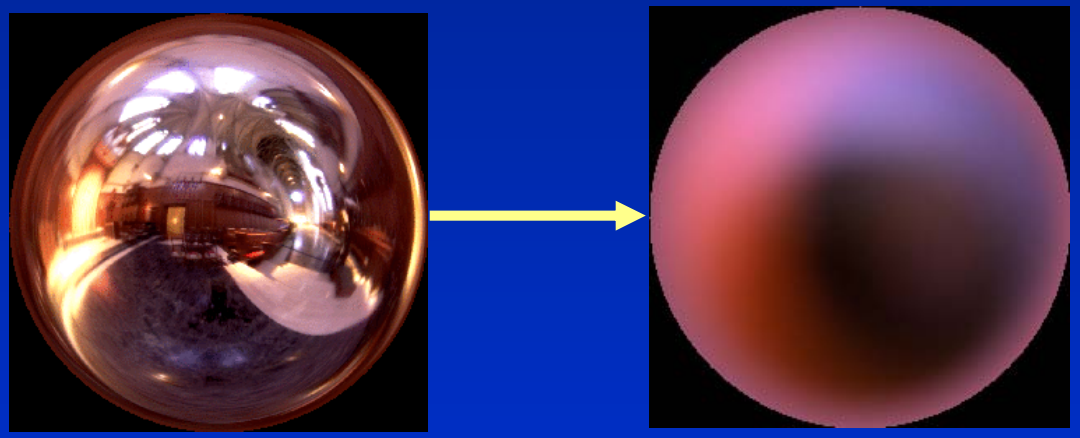


quake light map

Computing Irradiance

- Classically, hemispherical integral for each pixel

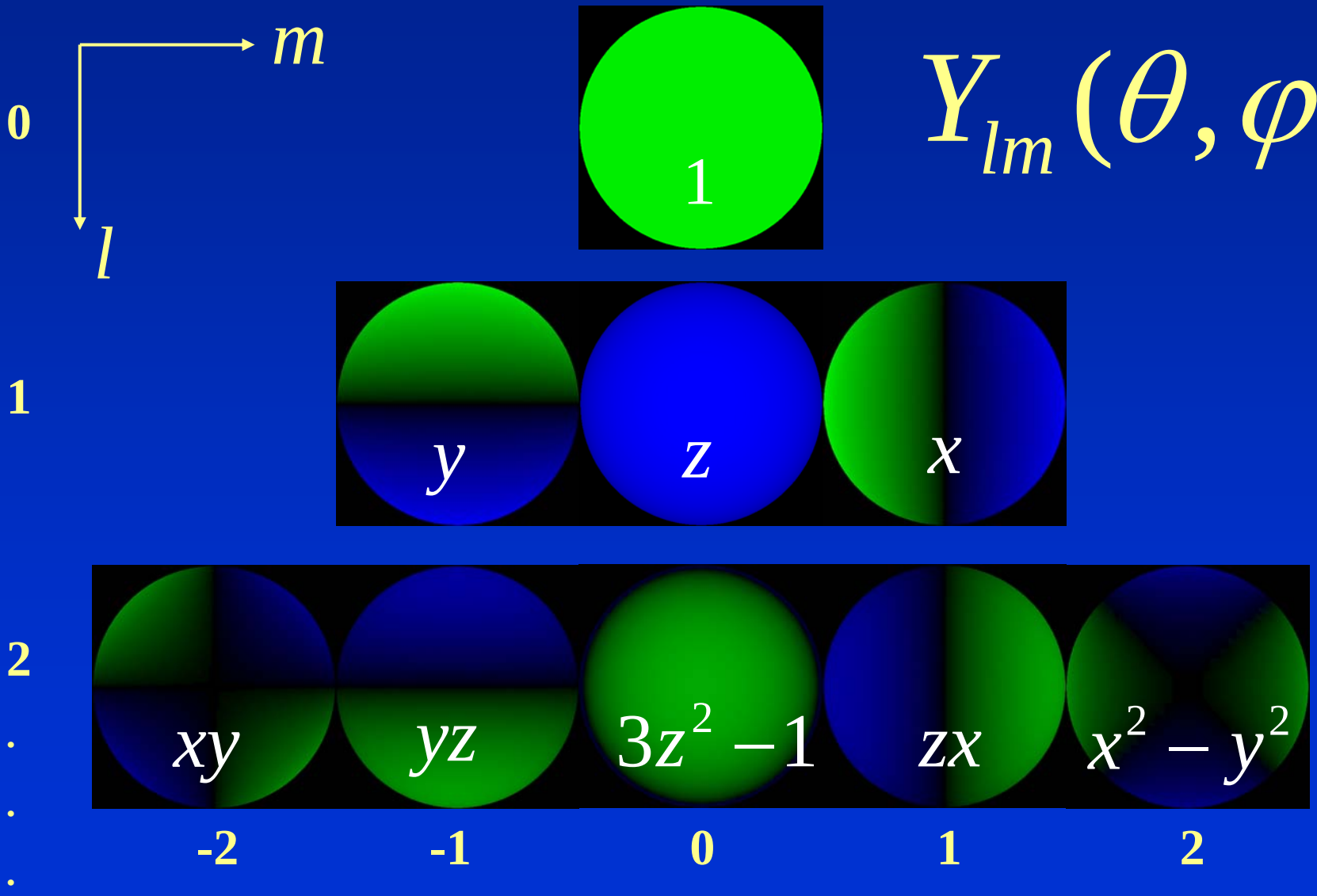
Incident Radiance



Irradiance

- Lambertian surface is like low pass filter
- Frequency-space analysis

Spherical Harmonics



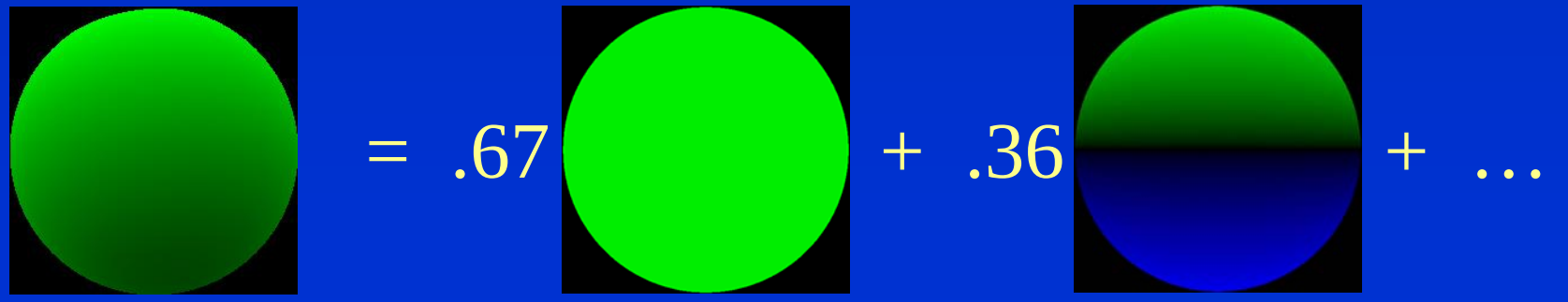
$$Y_{lm}(\theta,\varphi)$$

Spherical Harmonic Expansion

Expand lighting (L), irradiance (E) in basis functions

$$L(\theta,\phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} L_{lm} Y_{lm}(\theta,\phi)$$

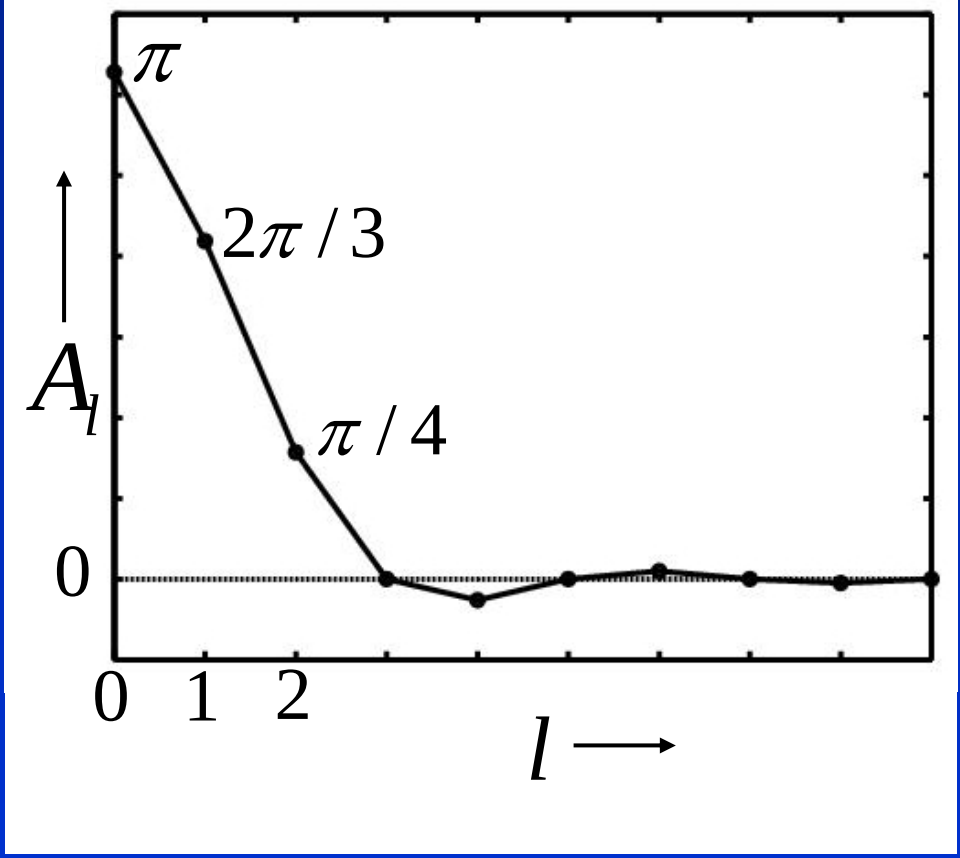
$$E(\theta,\phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} E_{lm} Y_{lm}(\theta,\phi)$$



Analytic Irradiance Formula

Lambertian surface acts like low-pass filter

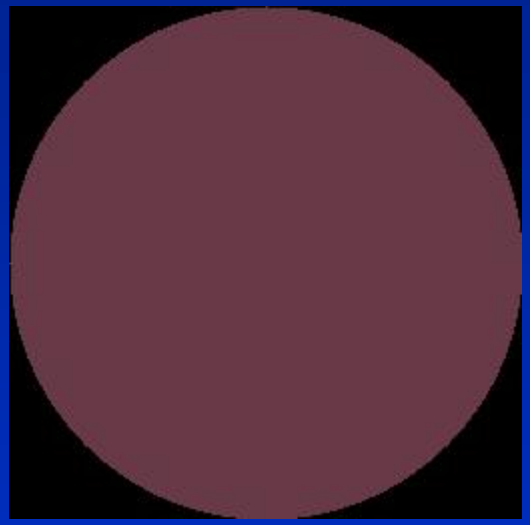
$$E_{lm} = A_l L_{lm}$$



$$A_l = 2\pi \frac{(-1)^{\frac{l}{2}-1}}{(l+2)(l-1)} \left[\frac{l!}{2^l \left(\frac{l}{2}!\right)^2} \right] \quad l \text{ even}$$

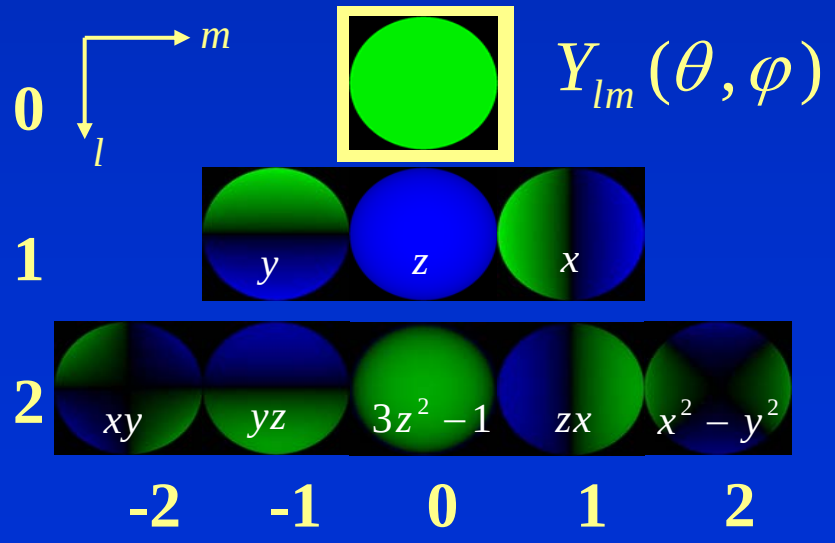
9 Parameter Approximation

Exact image



Order 0
1 term

RMS error = 25 %

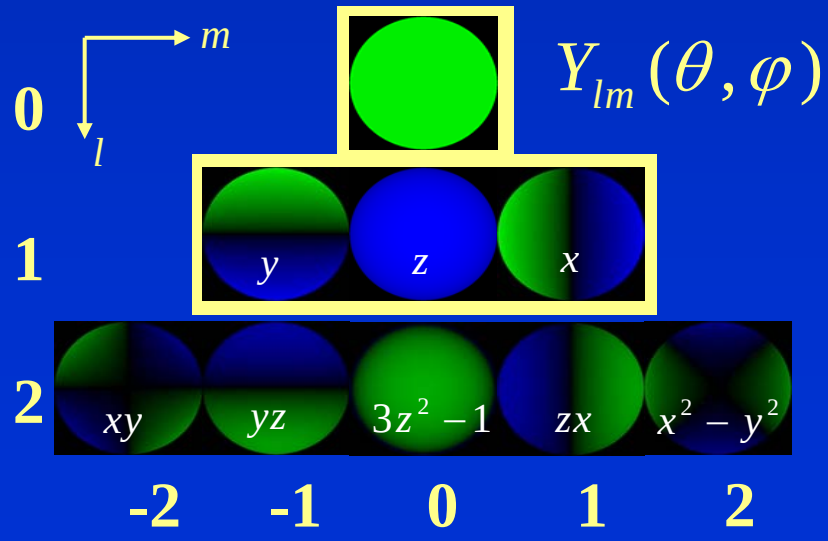


9 Parameter Approximation

Exact image

Order 1
4 terms

RMS Error = 8%



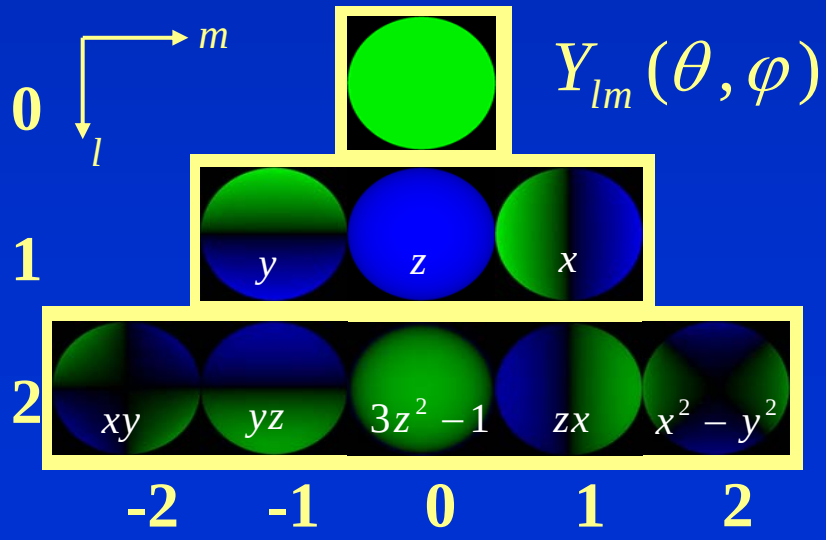
9 Parameter Approximation

Exact image

Order 2
9 terms

RMS Error = 1%

For any illumination, average error < 3% [Basri Jacobs 01]



Computing Light Coefficients

Compute 9 lighting coefficients L_{lm}

- 9 numbers instead of integrals for every pixel
- Lighting coefficients are moments of lighting

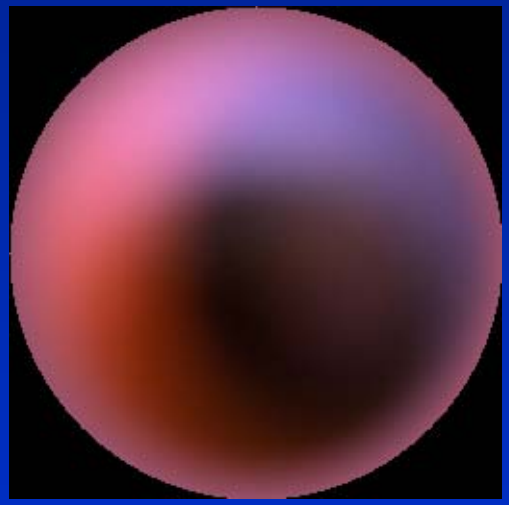
$$L_{lm} = \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} L(\theta,\phi)\, Y_{lm}(\theta,\phi)\, \sin\theta \, d\theta d\phi$$

- Weighted sum of pixels in the environment map

$$L_{lm} = \sum_{pixels(\theta,\phi)} envmap[\, pixel\,] \times basisfunc_{lm}[\, pixel\,]$$

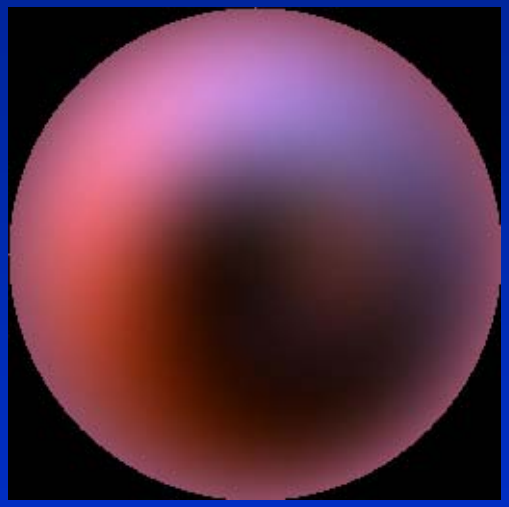
Comparison

Incident illumination 300x300



Irradiance map Texture: 256x256 Hemispherical Integration 2Hrs

Time $\propto 300 \times 300 \times 256 \times 256$



Irradiance map Texture: 256x256 Spherical Harmonic Coefficients 1sec

Time $\propto 9 \times 256 \times 256$

Rendering

- We have found the SH coefficients for irradiance which is a spherical function.
- Given a spherical coordinate, we want to calculate the corresponding irradiance quickly.

$$E(\theta,\phi) = \sum_{l,m} \hat{A}_l L_{lm} Y_{lm}(\theta,\phi)$$

Rendering

Irradiance approximated by quadratic polynomial

$$E(n) = c_4 L_{00} 1 + 2c_2 L_{11} x + 2c_2 L_{1-1} y + 2c_2 L_{10} z + c_5 L_{20} (3z^2 - 1) +$$
$$2c_1 L_{2-2} xy + 2c_1 L_{21} xz + 2c_1 L_{2-1} yz + c_1 L_{22} (x^2 - y^2)$$

$$E(n) = n^t M n$$

4x4 matrix
(depends linearly
on coefficients L_{lm})

Surface Normal vector
column 4-vector

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Hardware Implementation

$$E(n) = n^t M n$$

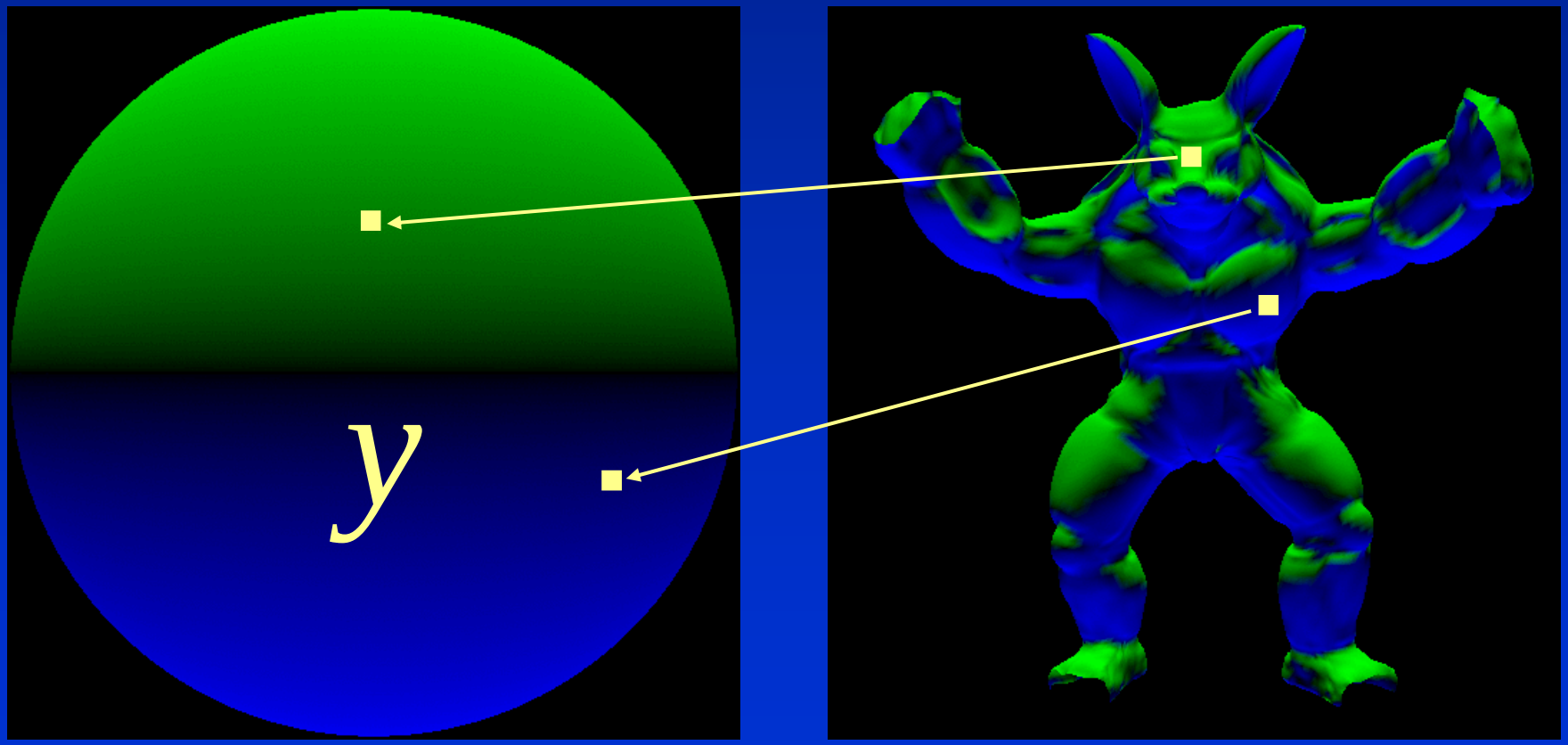
Simple procedural rendering method (no textures)

- Requires only matrix-vector multiply and dot-product
- In software or NVIDIA vertex programming hardware

```
surface float1 irradmat (matrix4 M, float3 v) {
        float4 n = {v , 1} ;
        return dot(n , M*n) ;
}
```

Complex Geometry

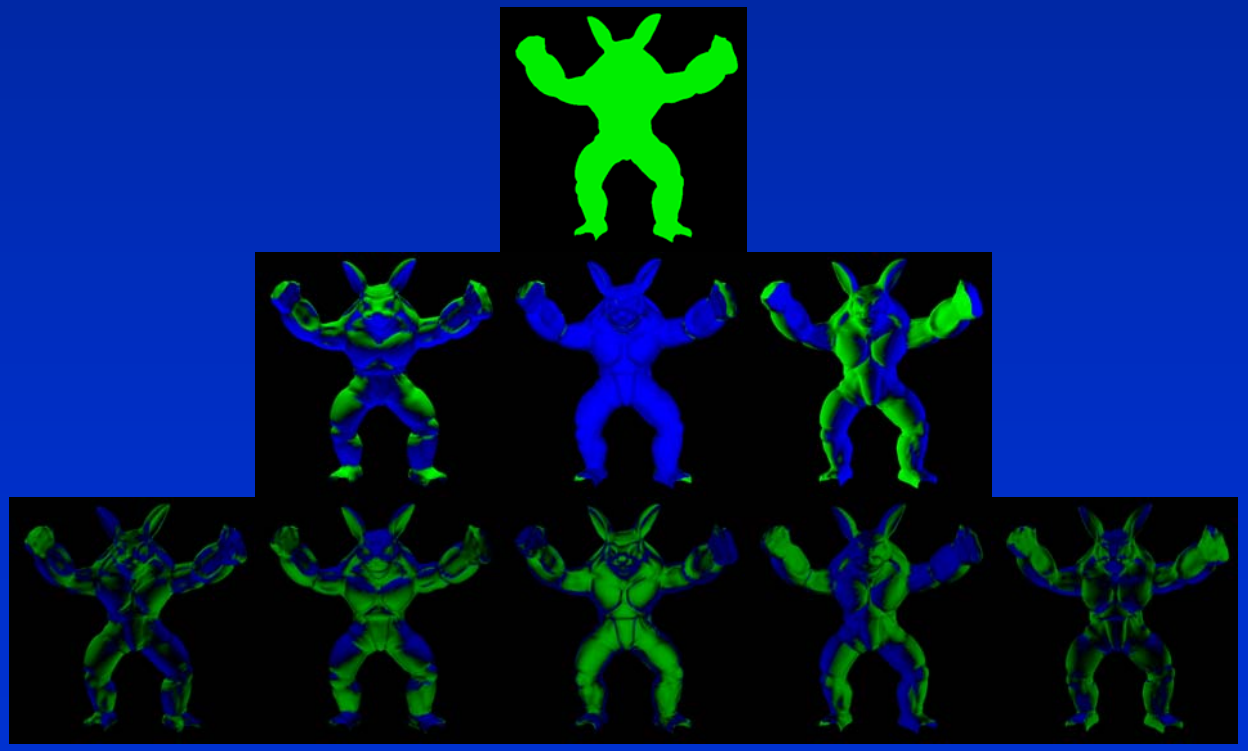
Assume no shadowing: Simply use surface normal



Lighting Design

Final image sum of 3D basis functions scaled by L_{lm}

Alter appearance by changing weights of basis functions



Results

Summary

Theory

- Analytic formula for irradiance
- Frequency-space: Spherical Harmonics
- To order 2, constant, linear, quadratic polynomials
- 9 coefficients (up to order 2) suffice

Practical Applications

- Efficient computation of irradiance
- Simple procedural rendering
- New representation, many applications

Precomputed Radiance Transfer
for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments

Peter-Pike Sloan, Microsoft Research

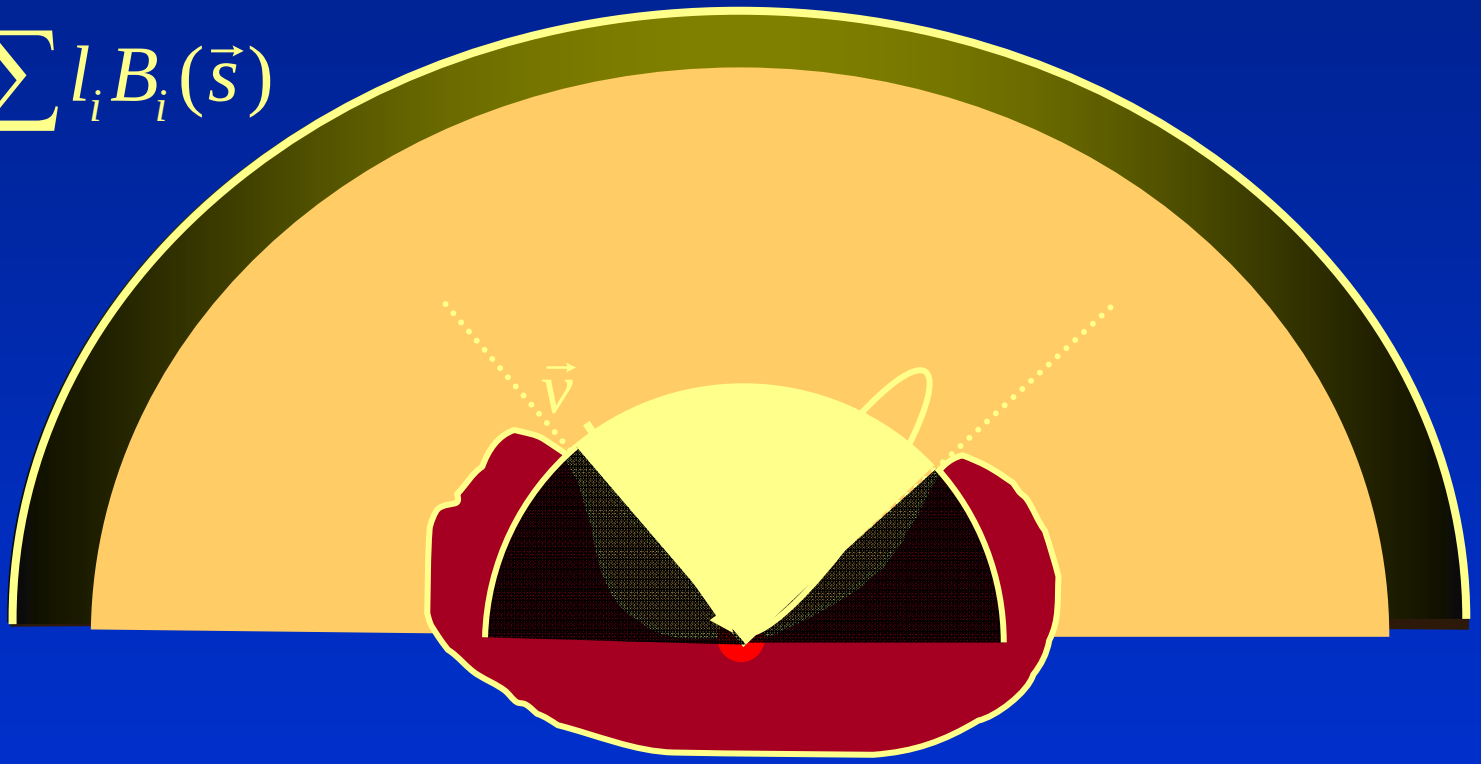
Jan Kautz, MPI Informatik

John Snyder, Microsoft Research

SIGGRAPH 2002

Basic idea

$$L(\vec{s}) \cong \sum l_i B_i(\vec{s})$$



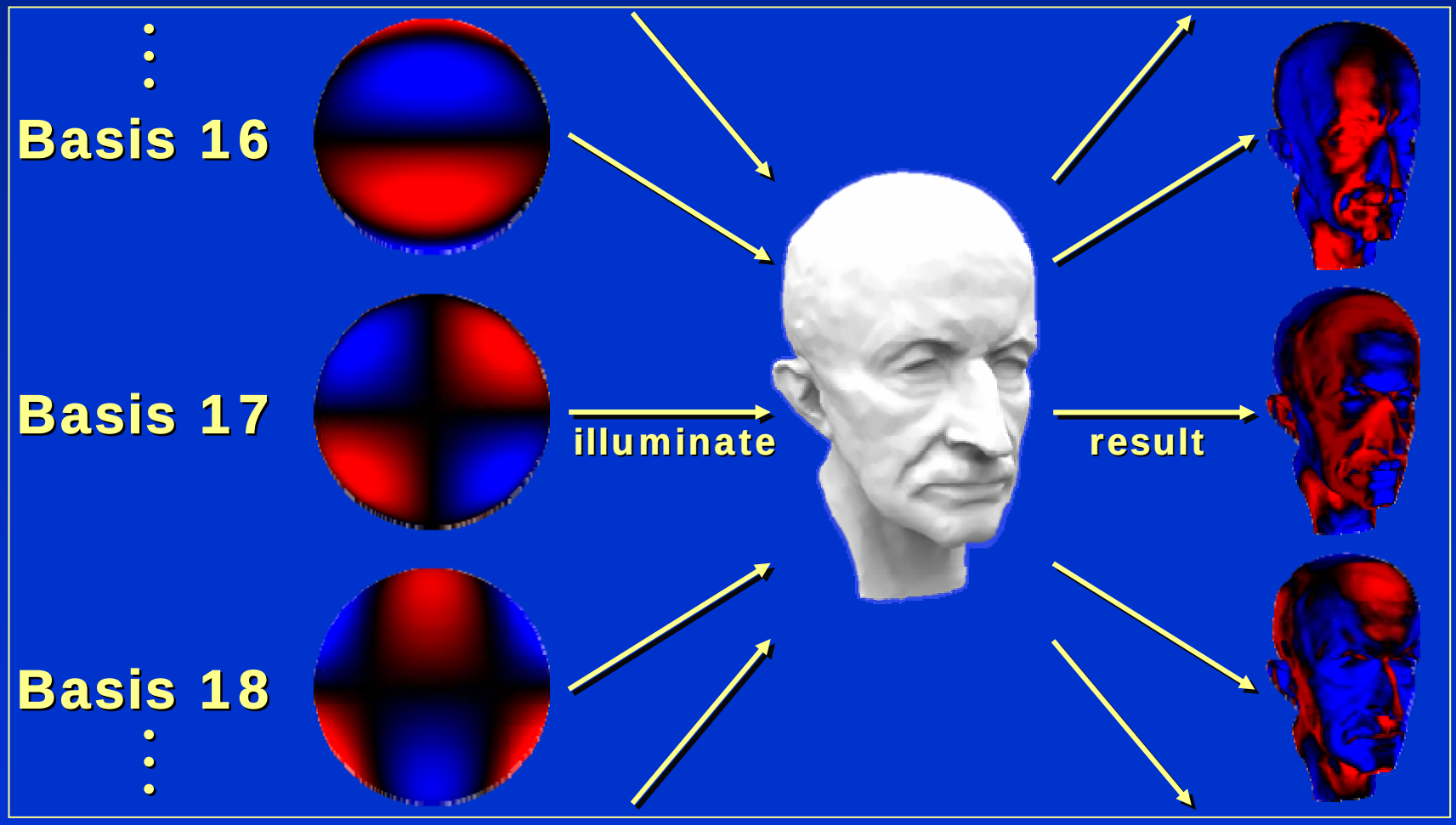
$$R(\vec{v}) = \int L(\vec{s})\, V(\vec{s})\, f(\vec{s},\vec{v})\, H_N(\vec{s})\, d\vec{s}$$

$$R(\vec{v}) \cong \sum_i l_i \int B_i(\vec{s})\, V(\vec{s})\, f(\vec{s},\vec{v})\, H_N(\vec{s})\, d\vec{s}$$

Preprocess for all *i*

Precomputation

Use 25 bases



Diffuse

No Shadows/Inter

Shadows

Shadows+Inter

Glossy

No Shadows/Inter

Shadows

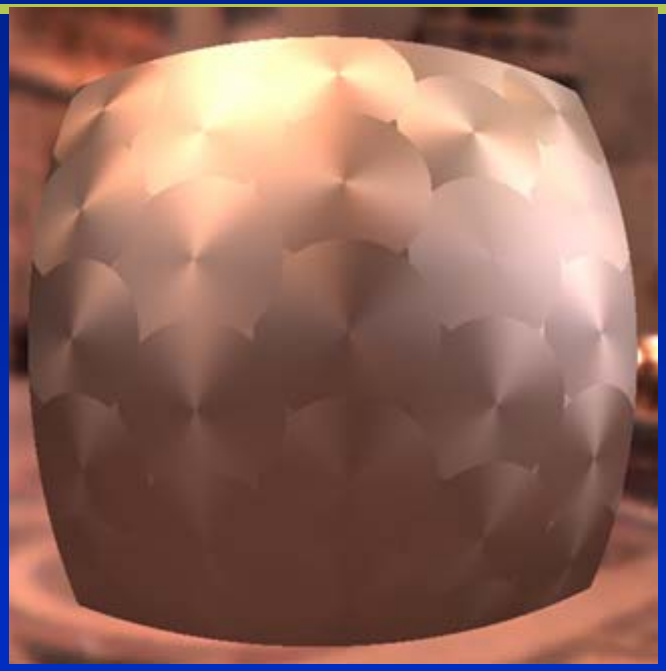
Shadows+Inter

- Glossy object, 50K mesh
- Runs at 3.6/16/125fps on 2.2Ghz P4, ATI Radeon 8500

Arbitrary BRDF

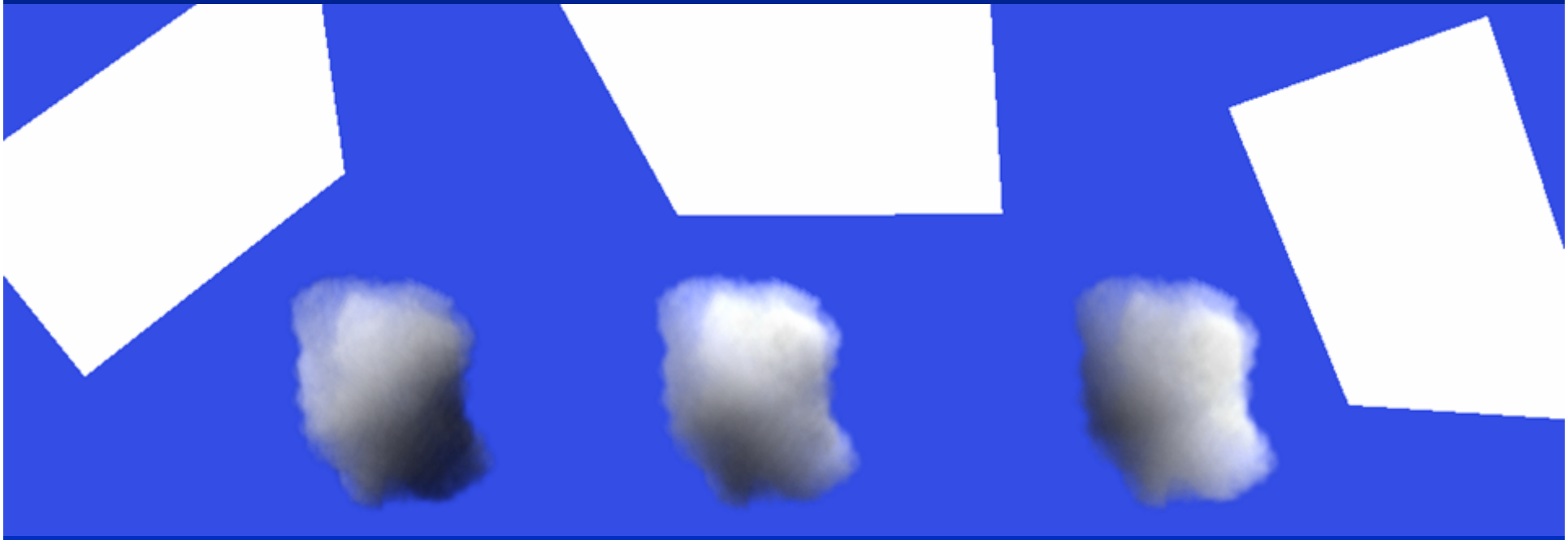
Anisotropic BRDFs

Other BRDFs



Spatially Varying

Volumes



- Diffuse volume: 32x32x32 grid
- Runs at 40fps on 2.2Ghz P4, ATI 8500
- Here: dynamic lighting