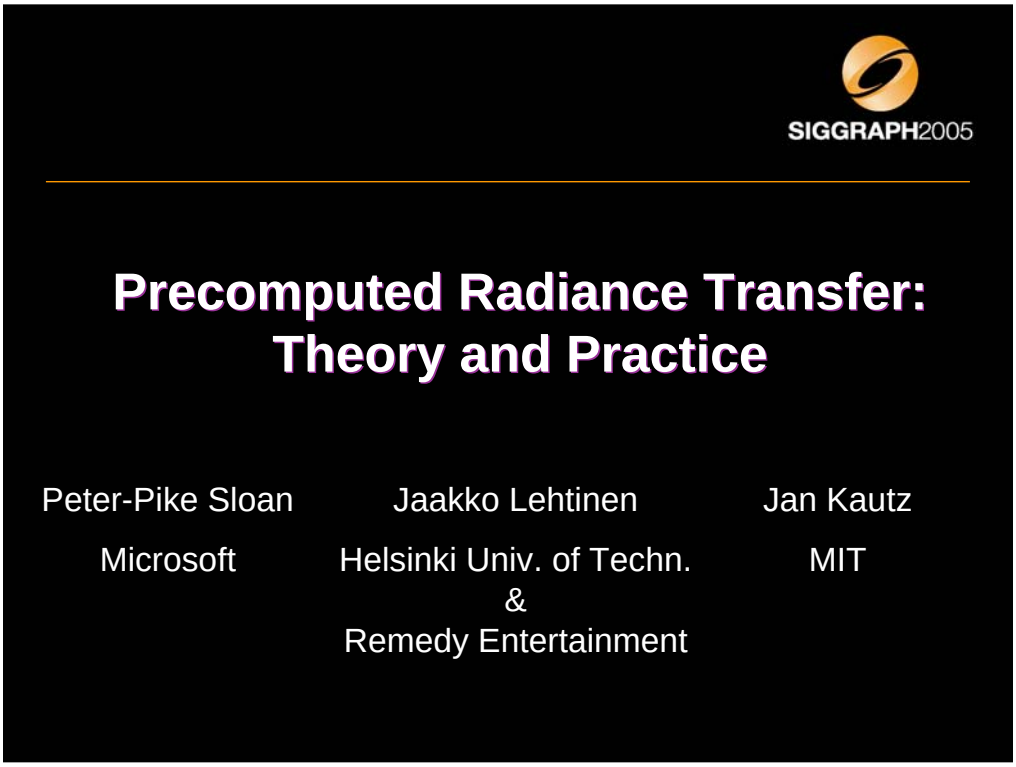
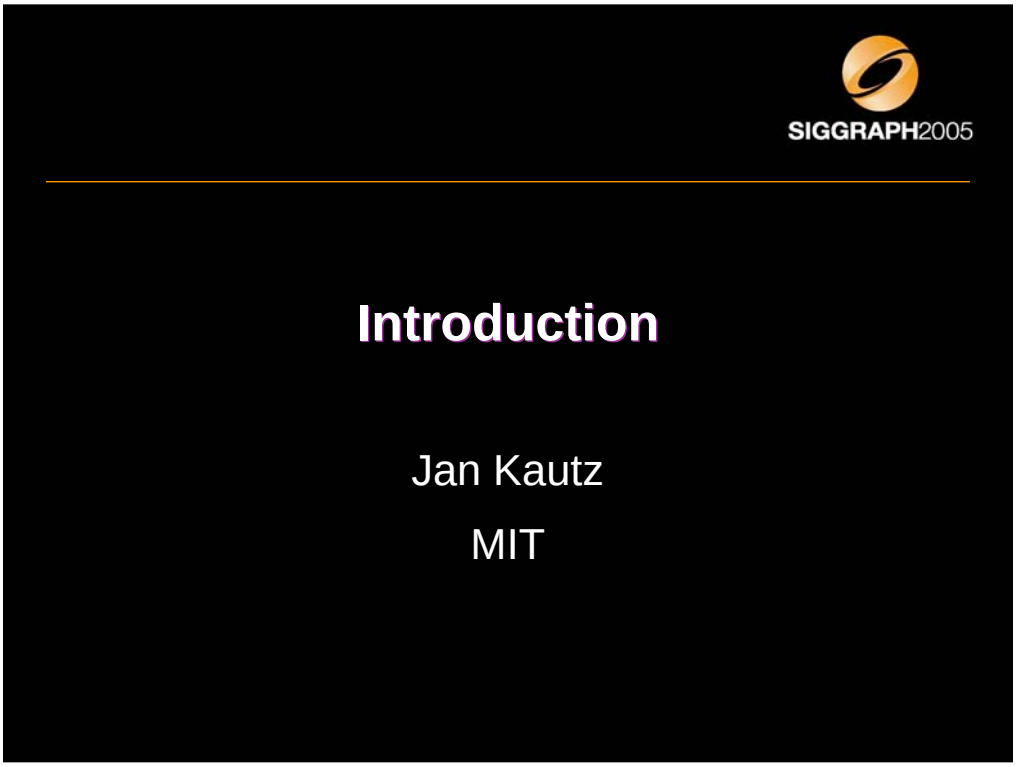
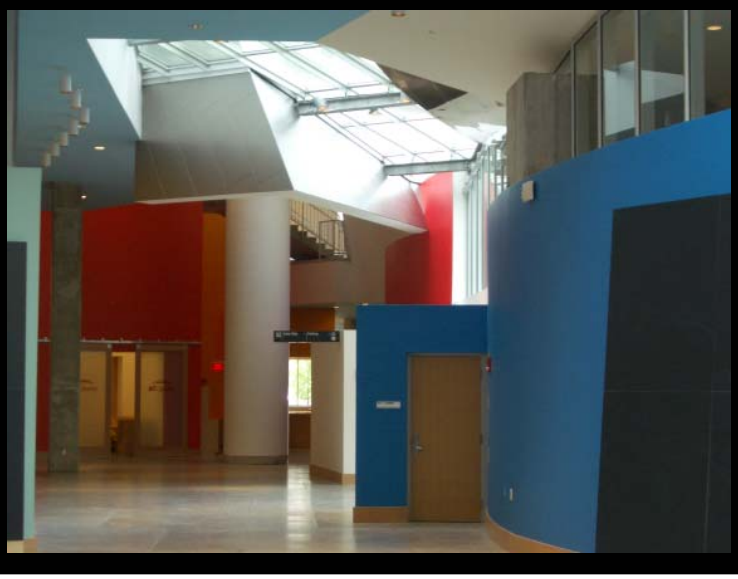
Precomputed Radiance Transfer:
Theory and Practice


SIGGRAPH2005
Precomputed Radiance Transfer:
Theory and Practice
Peter-Pike Sloan
Microsoft
Jaakko Lehtinen
Helsinki Univ. of Techn.
&
Remedy Entertainment
Jan Kautz
MIT


SIGGRAPH2005
Introduction
Jan Kautz
MIT

Introduction



We see here an example of a real-world scene which has a lot of visual complexity and richness. Generating synthetic images that come close to this is an extremely challenging problem.

Having them animate and respond to a users control is even more daunting.

We will discuss some of the issues that need to be addressed to meet this challenge.

There are many types of scene complexity which operate individually and in synergy with each other to generate the visual complexity of the resulting image.

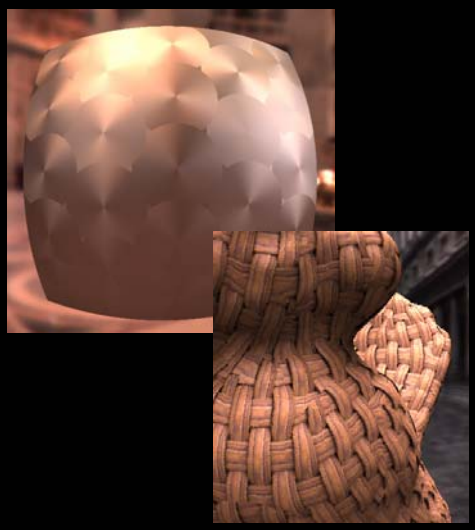
Geometric Complexity

- Real-world scenes have a lot of large-scale detail

Material Complexity

SIGGRAPH2005

- Models how light interacts with a surface
  - "Structure" below visible scale
  - At visible scale but not geometry



Lighting Complexity

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - “Smooth” (low frequency) lighting
  - Completely general

Lighting Complexity

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - “Smooth” (low frequency) lighting
  - Completely general

Lighting Complexity

SIGGRAPH2005

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - "Smooth" (low frequency) lighting
  - Completely general

Lighting Complexity

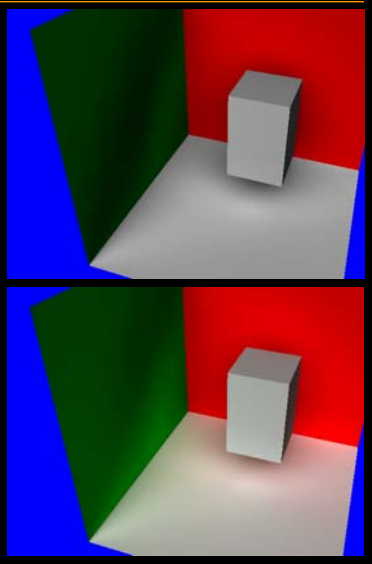
- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - “Smooth” (low frequency) lighting
  - Completely general

Transport Complexity

SIGGRAPH2005

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)

Transport Complexity

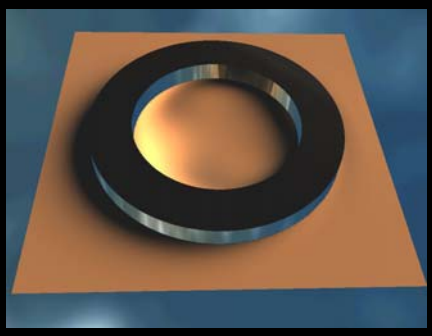


- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)

Transport Complexity

SIGGRAPH2005

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)

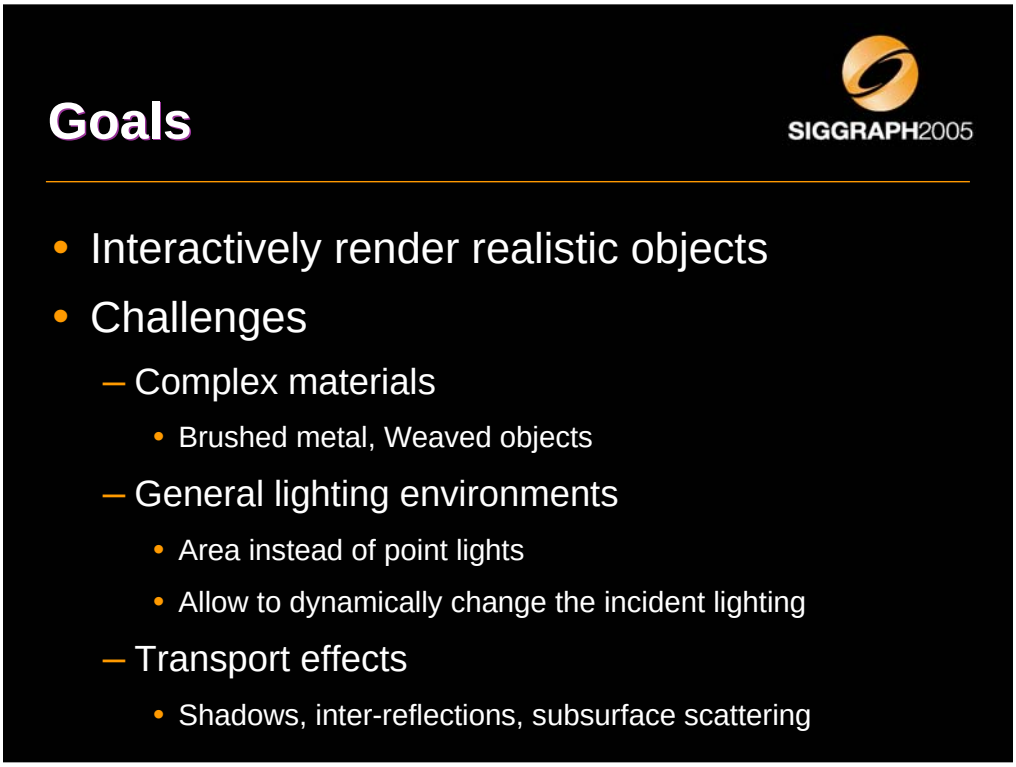


Transport Complexity

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)

Some of all of this

- Real scenes have all of these forms of complexity
- High realism on one is not necessarily interesting without the others
  - Complex materials lit by single point light
  - Complex lighting environments on diffuse surfaces with no shadows



The primary goal of this course is to accurately render objects in general lighting environments at interactive rates.

There are several challenges that must be overcome to achieve this goal.

Real objects have complex and spatially varying material properties – so we need to support general reflection models and handle spatial variation.

General lighting environments are much more compelling compared to simple point lighting models.

Run time integration over general area lighting models is expensive with traditional techniques.

We would like to support complex transport effects – soft shadows from area lights, inter-reflection, caustics and subsurface scattering, while maintaining interactive rendering rates.

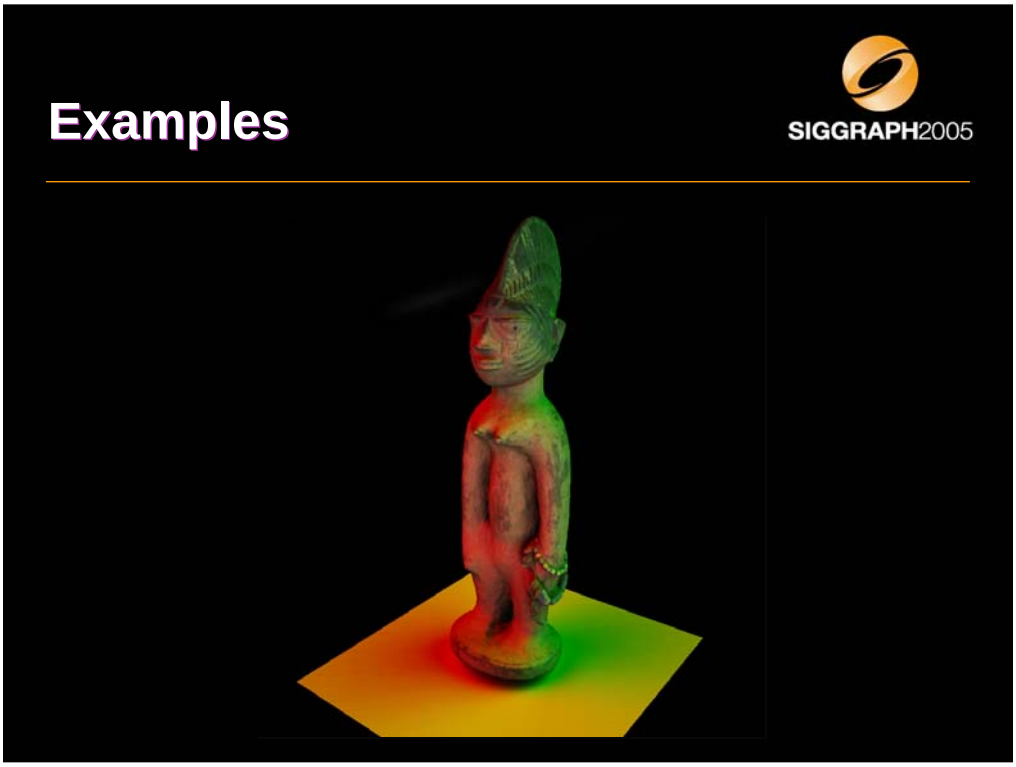
PRT

- What is Precomputed Radiance Transfer?
  - A way to shade objects under different illumination
    - Includes direct & indirect lighting, caustics, subsurface scat.
    - Any kind of transport is possible
    - Real-time, allowing lighting to change
  - Illumination can come from all directions (env-map)
    - Usually assumed to be far away
    - Depending on implementation, possible restrictions on lighting (e.g. low-frequency)

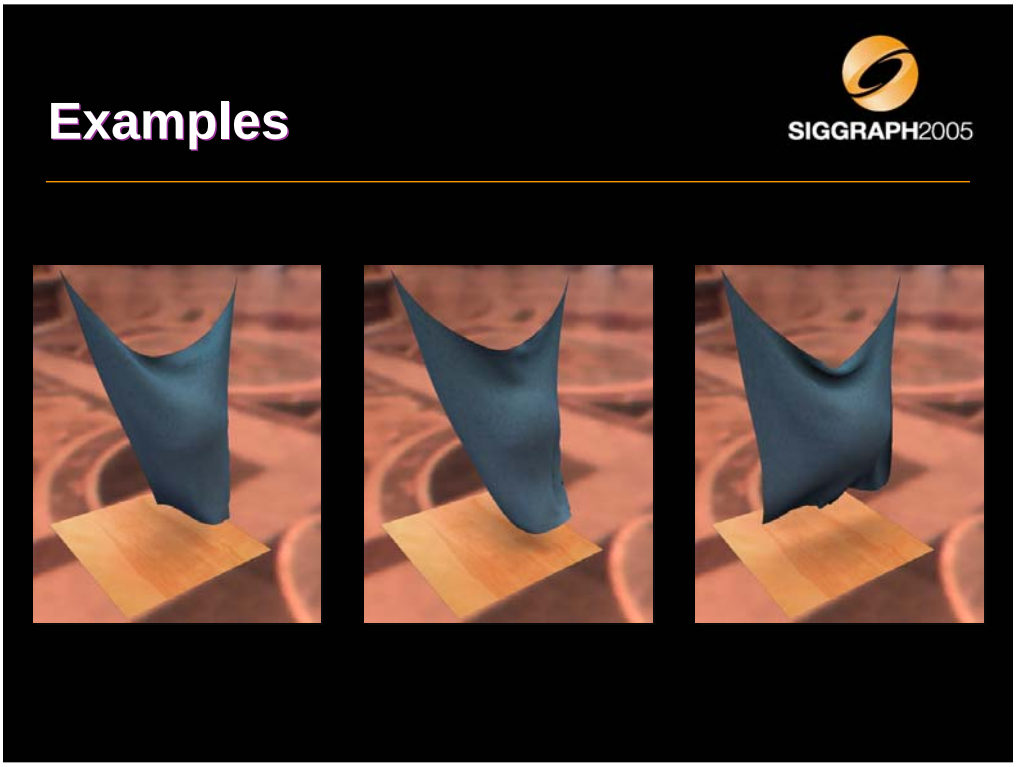
PRT

- What does it not do?
  - It does not do full global illumination for arbitrary dynamic scenes
  - Objects have to be static!
    - First ideas on how to deal with animations are appearing
  - Object-to-object interaction is limited
    - Some new algorithms coming out though

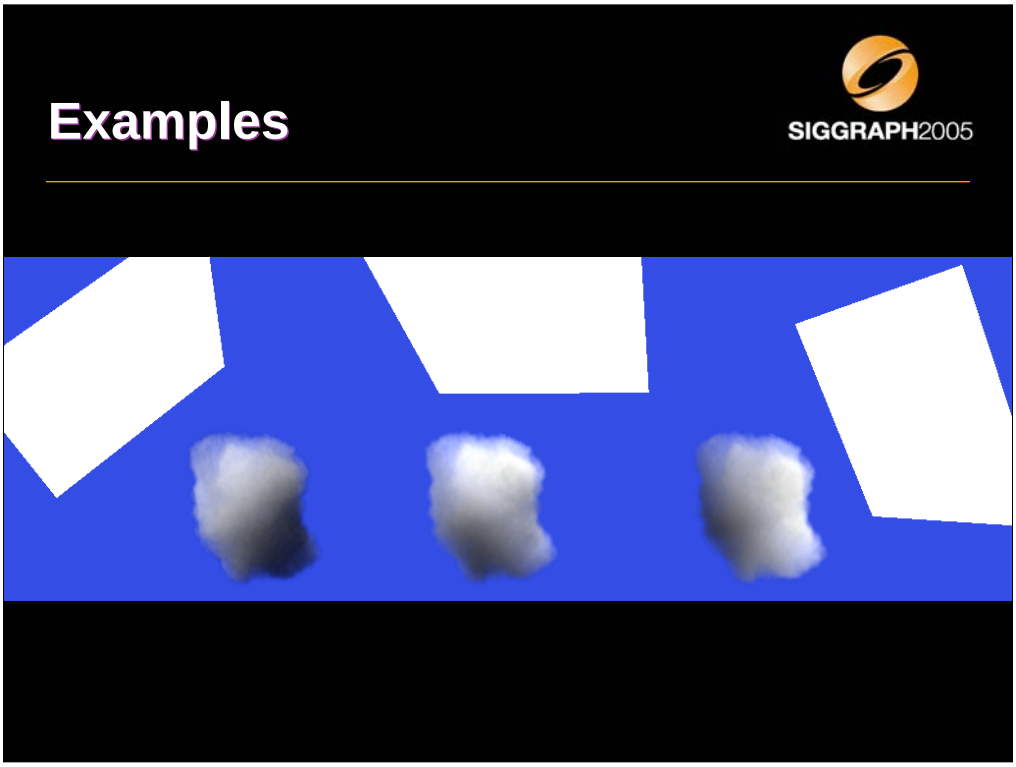
Bust illuminated with a forest environment. Note the soft shadows.



Statue illuminated by two area sources, one green and one red. Number and sizes of light sources don't occur any performance penalty when PRT is used.



Some more examples. Note the nice soft shadows.

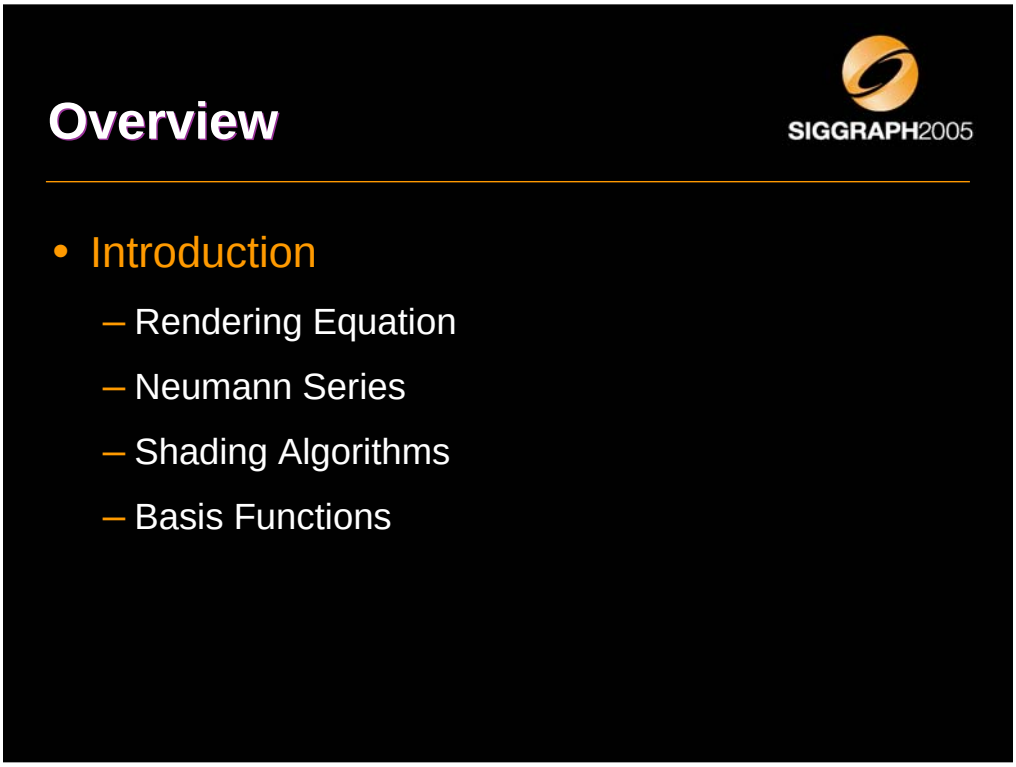


PRT can also be used in volumes.

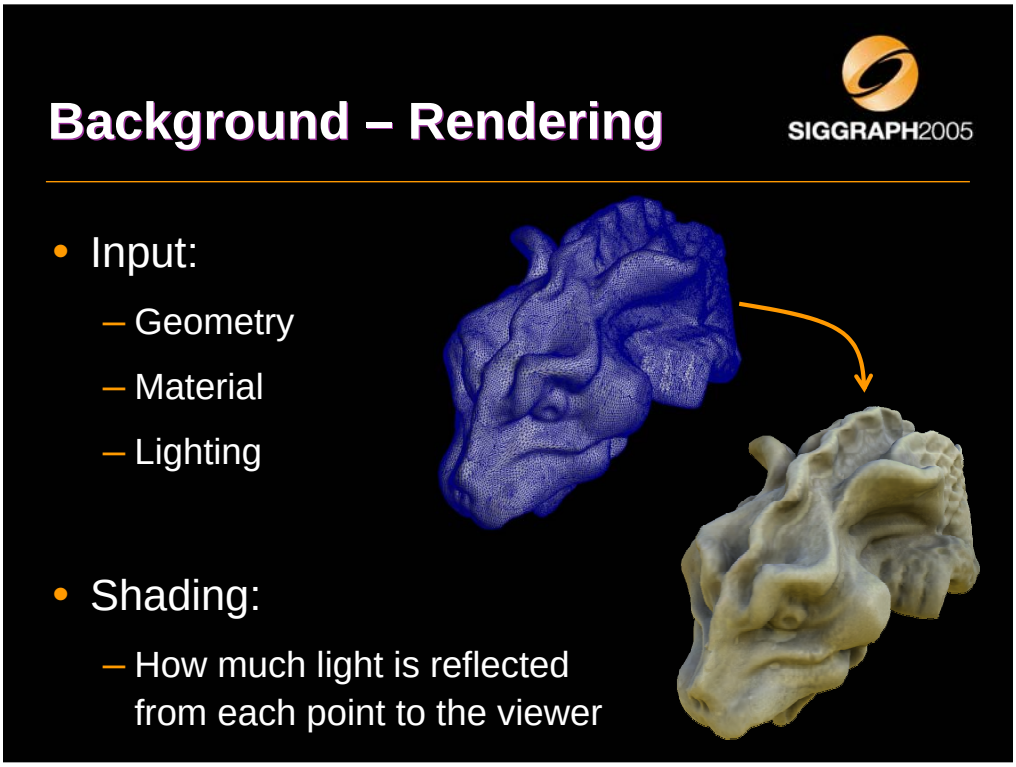

SIGGRAPH2005
Demos

Course Overview

- 8:30 – 9:00 Introduction (Jan Kautz)
- 9:00 – 9:15 Diffuse Precomputed Radiance Transfer (Jan Kautz)
- 9:15 – 10:15 General Precomputed Radiance Transfer (Jaakko Lehtinen)
- Break
- 10:30 – 11:00 SH Light Representations (Jan Kautz)
- 11:00 – 11:30 Practical PRT I (Peter-Pike Sloan)
- 11:30 – 12:00 Practical PRT II (Peter-Pike Sloan)
- 12:00 – 12:15 Conclusions/Summary (Peter-Pike Sloan)



In this introduction, we will learn about the Rendering Equation, its Neumann expansion, different shading algorithms, and basis functions.

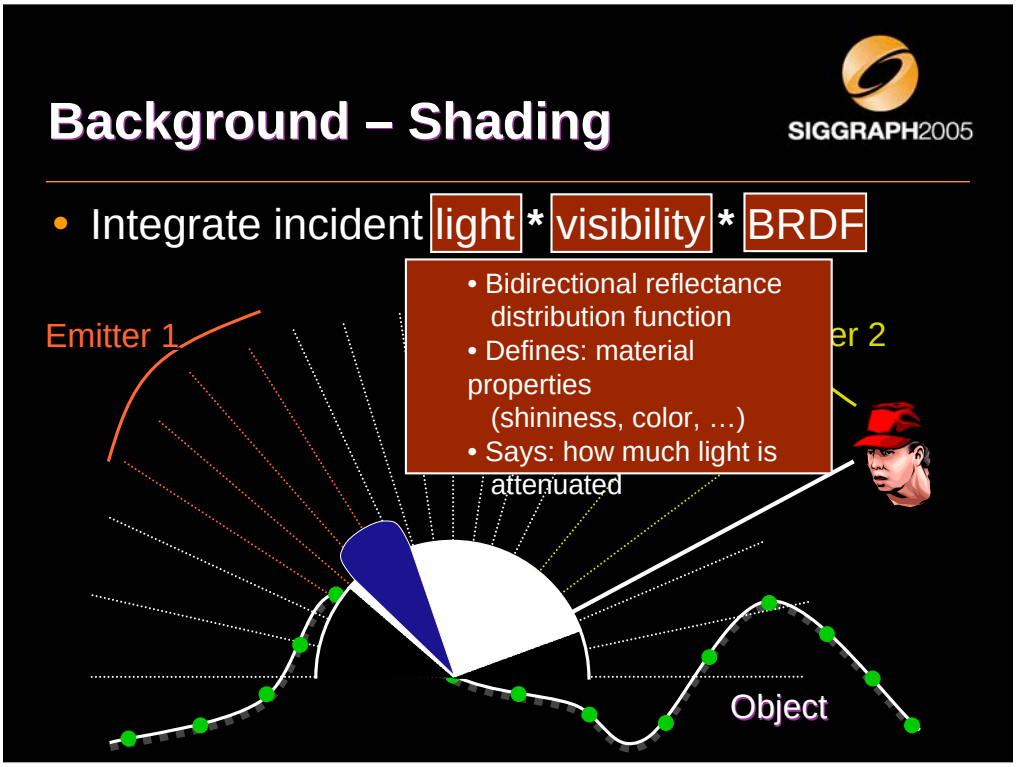


Loosely put, rendering takes input (geometry, materials, lights) and produces an image.

Shading computes how much light is reflect from each visible point to the viewer

There are many different ways to compute an image given the input. Precomputed Radiance Transfer is a technique to accelerate that process.

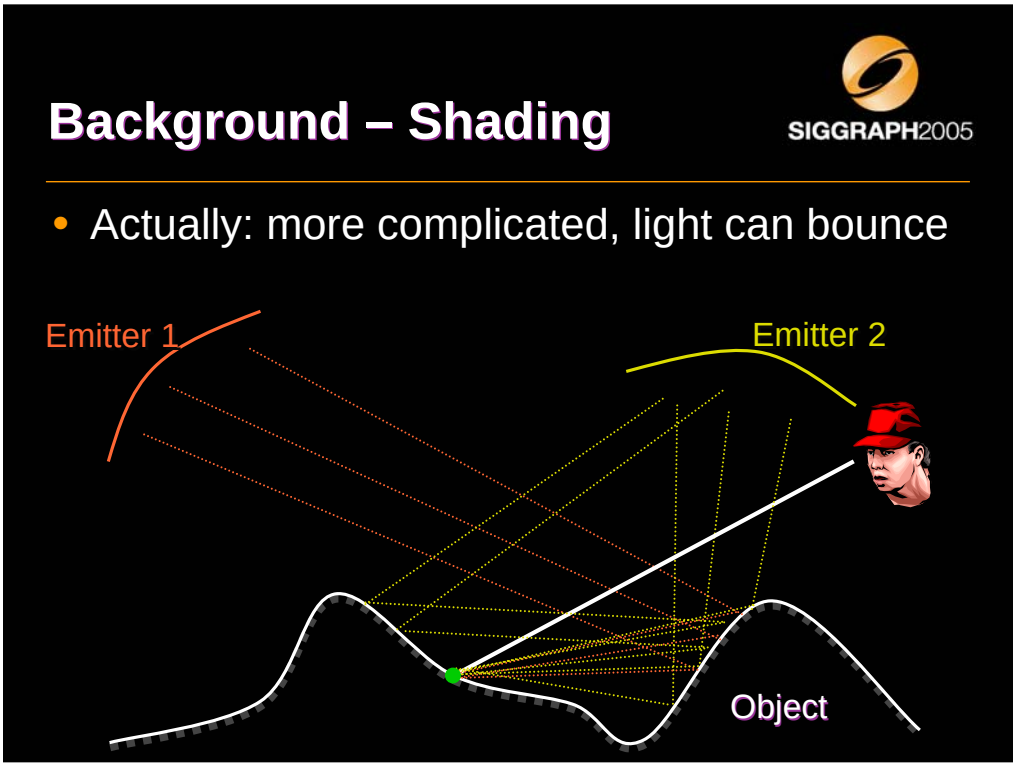
PRT deals with the shading computation (how much light is reflected from every visible point). The most general formulation for that is the so-called Rendering Equation. We will have a look at it in the following slides.



Shading needs to do the following:

Integrate over all incoming light directions, multiply with the binary visibility function and multiply with the material properties (e.g. here: light is coming from mostly the blue cone of directions).

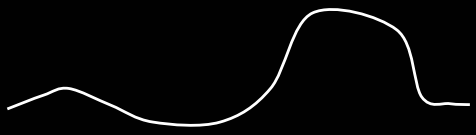
This computation has to be repeated for every point on the surface, which makes it a fairly costly operation.



In reality this is even more complicated, as light can bounce around on the object, before it is finally reflected towards the viewer.

Rendering Equation

$$L\left(p \to \vec{d}\right) = L_e\left(p \to \vec{d}\right) + \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L\left(p \leftarrow \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

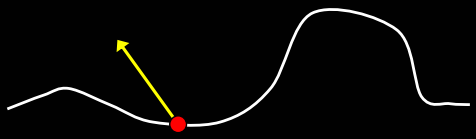


Given an object illuminated in a lighting environment, the rendering equation models the equilibrium of the flow of light in the scene. It can be used to determine how light a visible point reflects towards the viewer. We will walk through a hemispherical formulation of this equation.

Rendering Equation

SIGGRAPH2005

$$L(p \to \vec{d}) = L_e(p \to \vec{d}) + \int_{\Omega} f_r(p, \vec{s} \to \vec{d}) L(p \leftarrow \vec{s}) H_{N_p}(-\vec{s}) ds$$

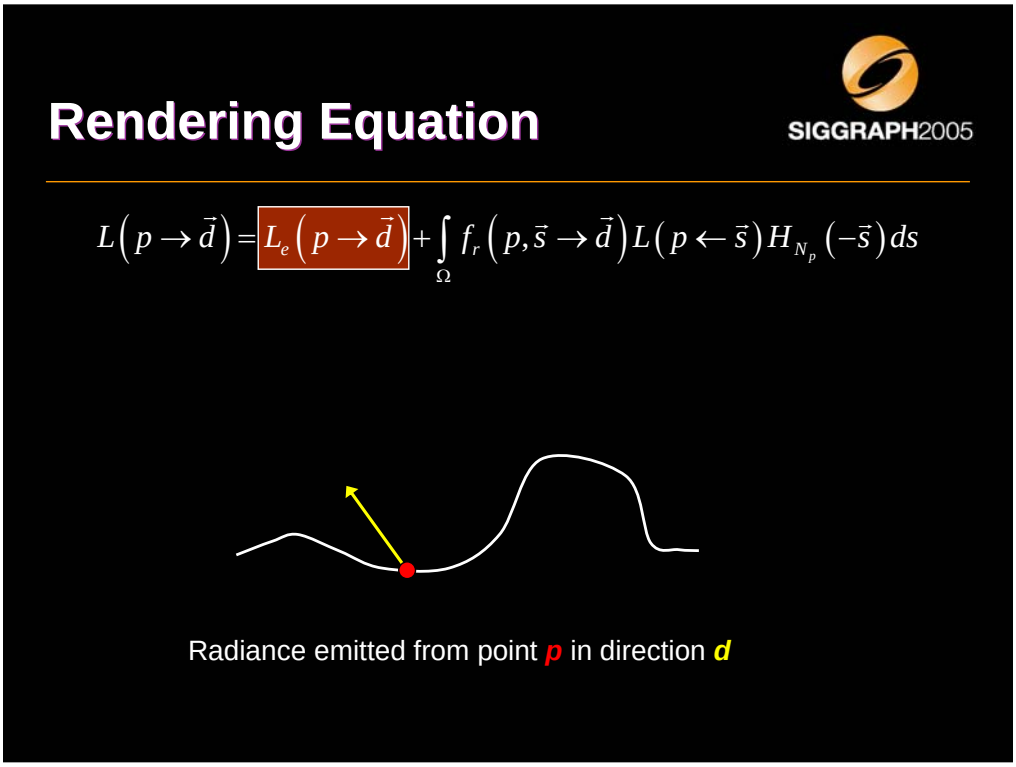


Radiance leaving point ***p*** in direction ***d***

The desired quantity is the radiance leaving a point on the object P in a given direction d.

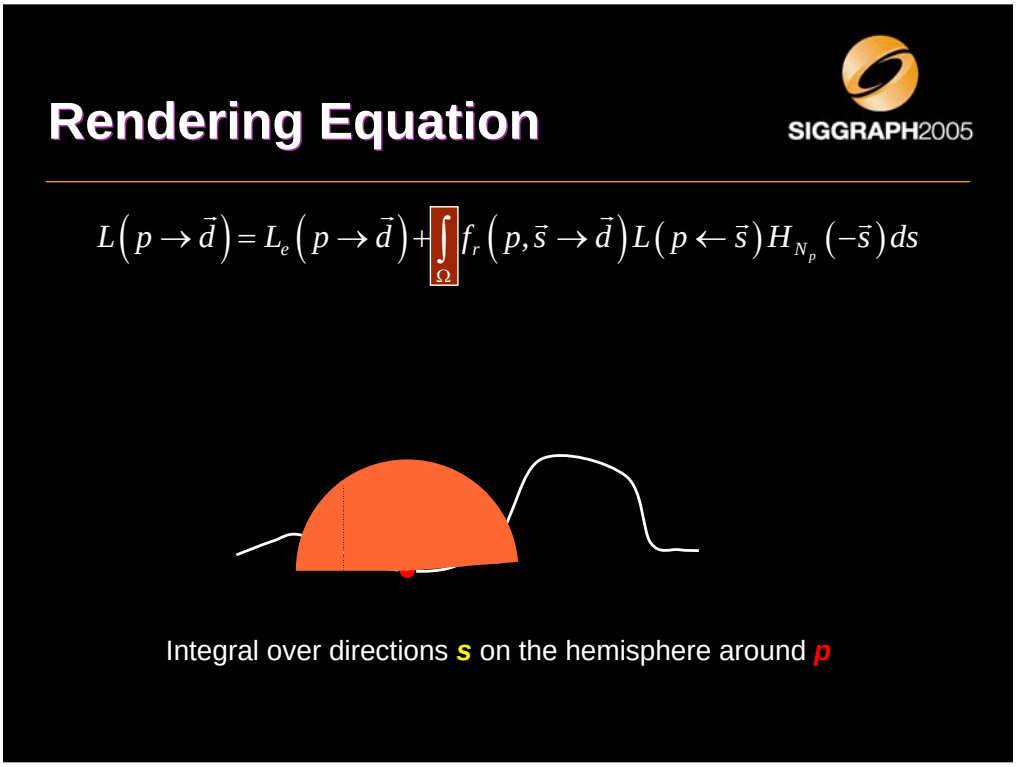
Radiance is the intensity of light from a point to a certain direction.

Rendering Equation



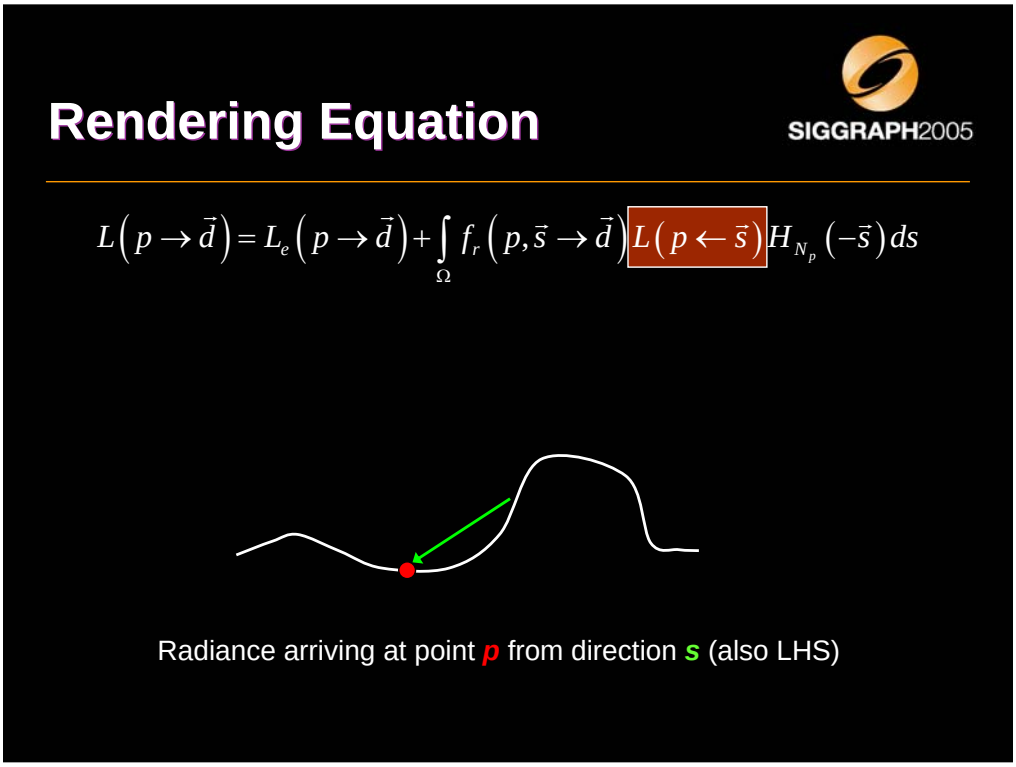
The first term is the radiance emitted directly from the point in the given direction. In our work we will assume that no objects emit light, they are just lit by a distant lighting environment (the source radiance function.)

Rendering Equation



This is followed by an integral over the hemisphere around the point, where s is used to denote a direction on this hemisphere

Rendering Equation

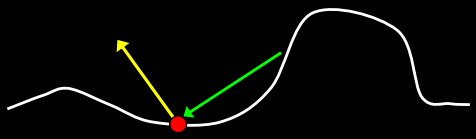


The second term is the radiance arriving at point P from the direction S, note that this is also the variable we are solving for so this is an integral equation.

Rendering Equation

SIGGRAPH2005

$$L\left(p \to \vec{d}\right) = L_e\left(p \to \vec{d}\right) + \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L\left(p \leftarrow \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

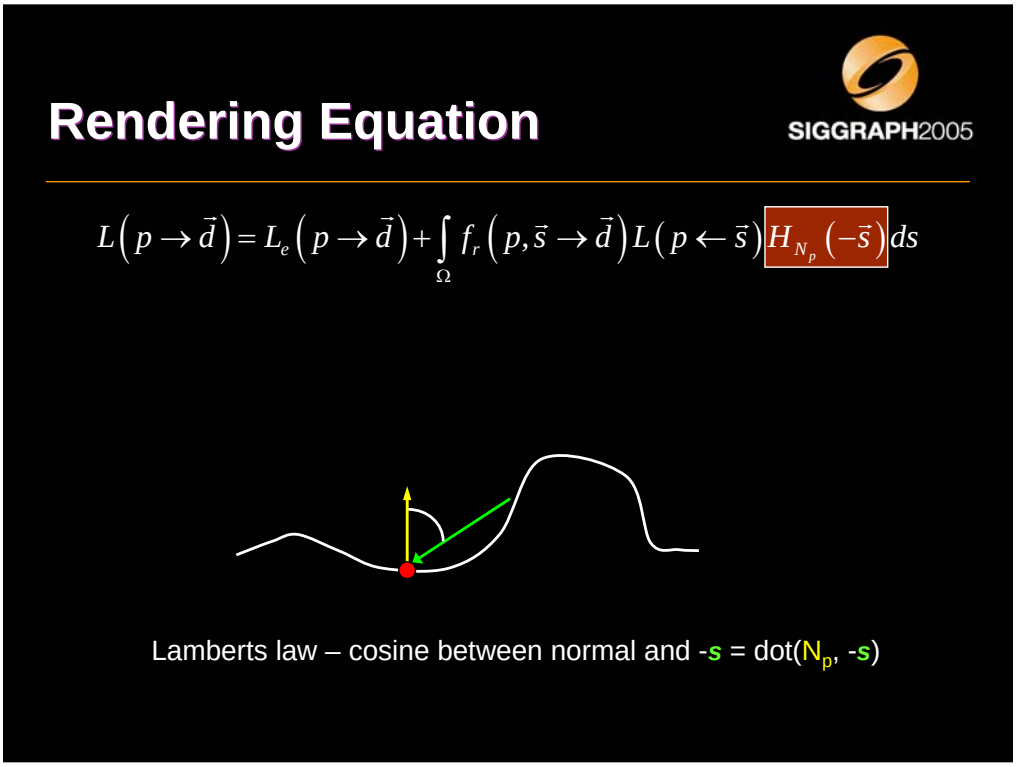


BRDF at point ***p*** evaluated for incident direction ***s*** in outgoing direction ***d***

BRDF: defines look of material. 4D function: $f_r^{\,p}(\vec{s} \to \vec{d})$

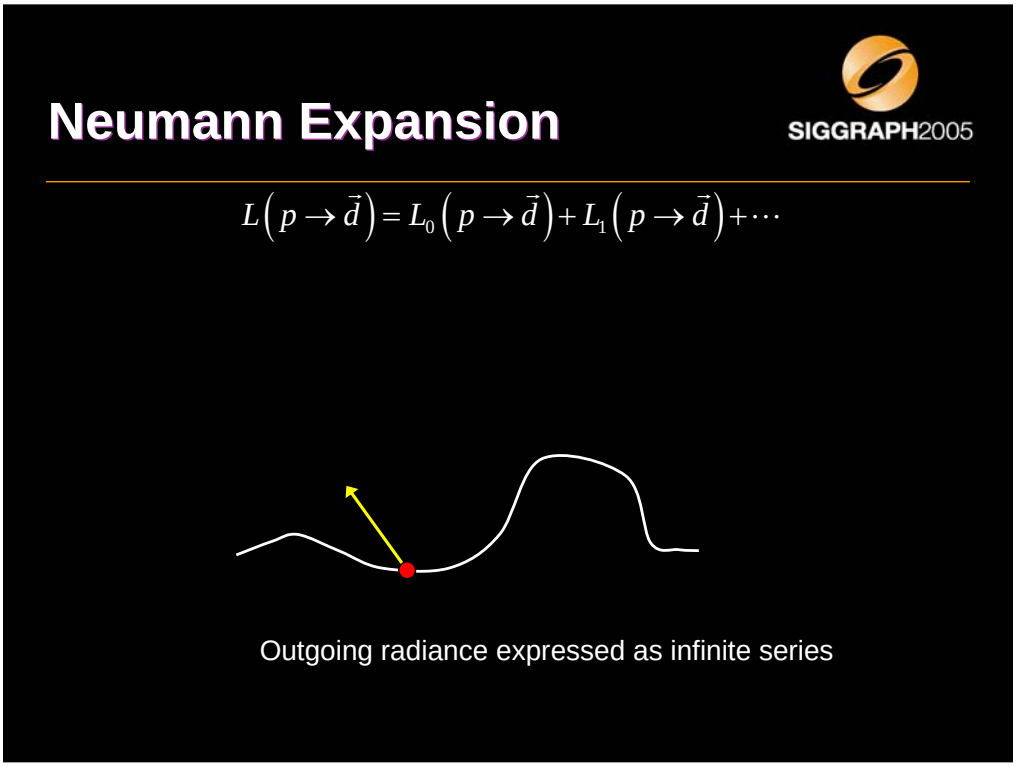
The 1st factor inside the integral is the BRDF of the surface at point P, the BRDF is a 4D function that models what percent of light for some input direction (s) leaves in some outgoing direction (d).

Rendering Equation



The final term is the cosine term that comes from lamberts law – due to projected area.

Neumann Expansion

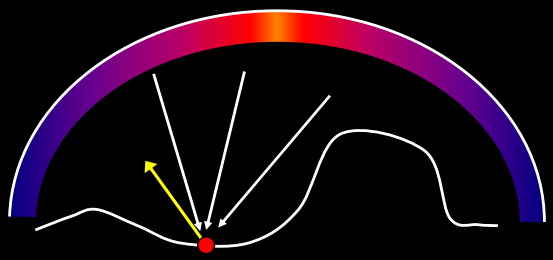


One convenient way to reason about the solution to this integral equation is by using a Neumann expansion of this expression, where outgoing radiance is expressed as an infinite series.

Neumann Expansion

SIGGRAPH2005

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$



Direct lighting arriving at point p – from distant environment

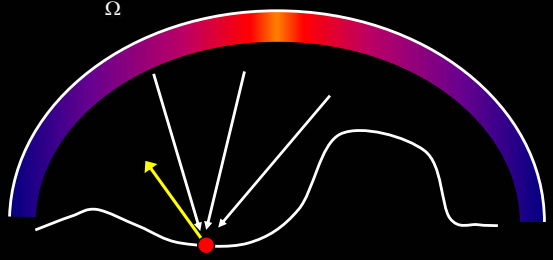
The first term in this series is the direct lighting arriving at point P from a distant lighting environment – the source radiance environment we referred to earlier.

Neumann Expansion

SIGGRAPH2005

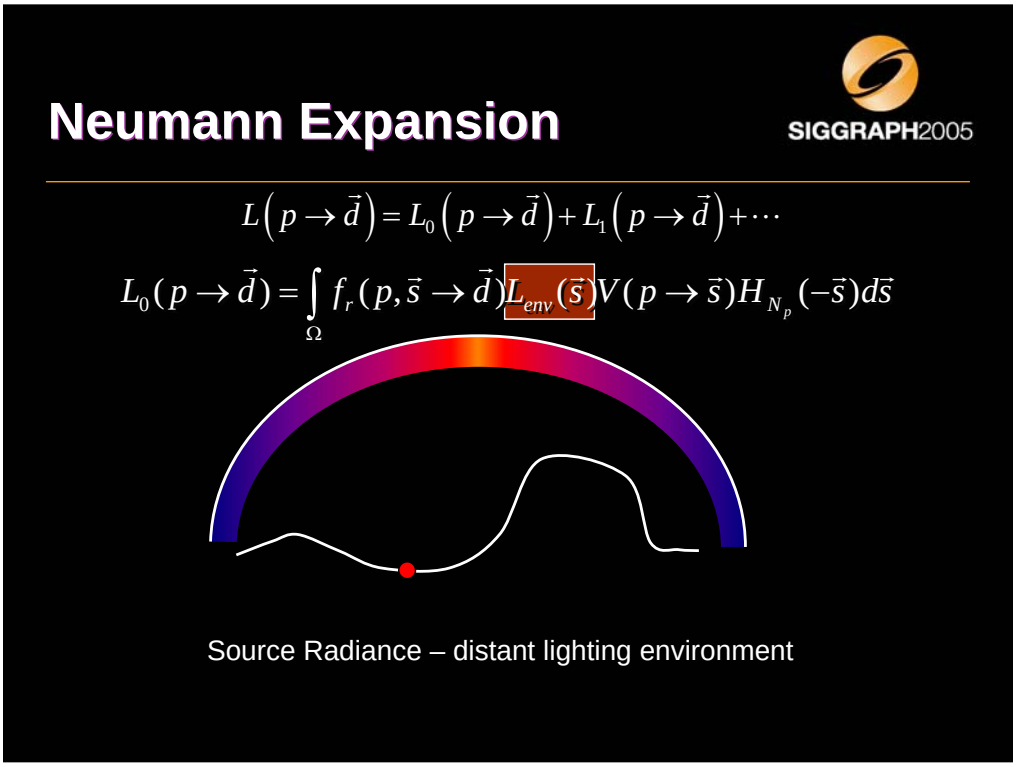
$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L_0(p \to \vec{d}) = \int_{\Omega} f_r(p, \vec{s} \to \vec{d}) L_{env}(\vec{s}) V(p \to \vec{s}) H_{N_p}(-\vec{s}) d\vec{s}$$



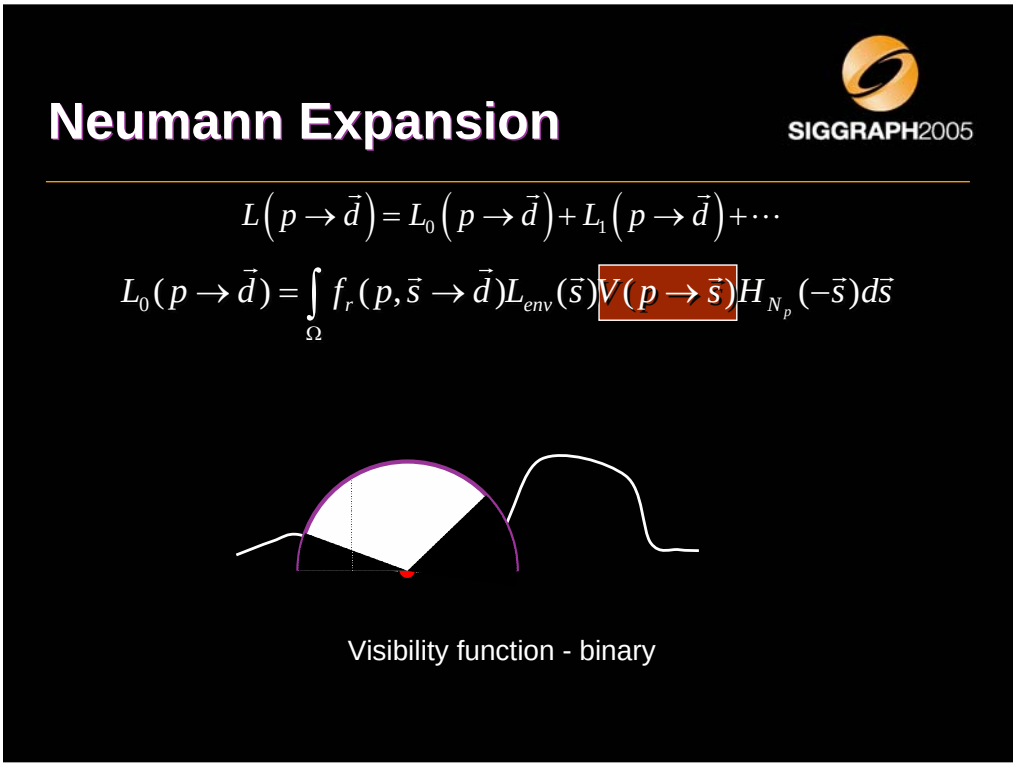
Direct lighting arriving at point p – from distant environment

This term is an integral over the hemisphere at the point p.



One of the new factors in this expression is Lenv – the source radiance function we are assuming that this is the only source of light in the scene. It is assumed to be far away, so there is no dependence on the position p (same incident radiance at every p) --- usually an environment map is used to represent this distant lighting. This is just a conventional integral, the source radiance function only exists inside the integral.

Neumann Expansion



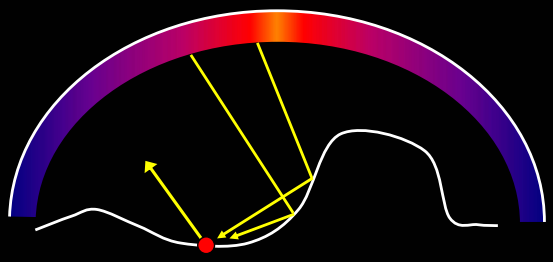
The next new factor is the visibility function – this is a binary function that is 1 in a given direction if the point can “see” the distant lighting environment, and zero otherwise.

L0 models how light that directly reaches the surface contributes to outgoing radiance.

Neumann Expansion

SIGGRAPH2005

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$



All paths from source that take 1 bounce

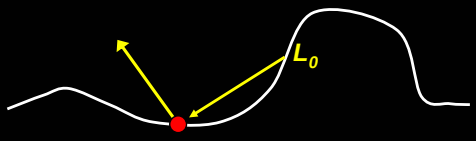
The next term in the expansion models all paths from the source radiance function that reach the given point after a single bounce and contribute to outgoing radiance in the given direction.

Neumann Expansion

SIGGRAPH2005

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L_1\left(p \to \vec{d}\right) = \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L_0\left(p \leftarrow \vec{s}\right)\left(1 - V\left(p \to \vec{s}\right)\right) H_{N_p}\left(-\vec{s}\right) ds$$



All paths from source that take 1 bounce

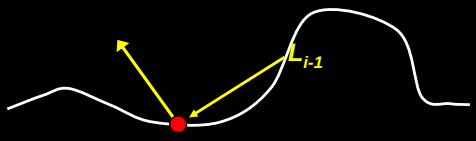
This is also just a conventional integral where the previous term (L_0) is inside of the integral.

Neumann Expansion

SIGGRAPH2005

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L_i\left(p \to \vec{d}\right) = \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L_{i-1}\left(p \leftarrow \vec{s}\right) \left(1 - V\left(p \to \vec{s}\right)\right) H_{N_p}\left(-\vec{s}\right) ds$$



All paths from source that take i bounces

In general the i-th bounce models how all of the energy from the “previous bounce” contributes to outgoing radiance in the given direction.

Overview

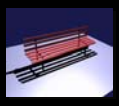
- Introduction
  - Rendering Equation
  - Neumann Series
  - Shading Algorithms
  - Basis Functions

Shading Algorithms

SIGGRAPH2005

- Real-time Rendering:
  Approximations to Rendering Equation
  - Direct lighting – indirect lighting: ambient term

  - Point lights – no area lights



    - Hard shadows – no soft shadows
  - Distant environment maps – no nearby emitters

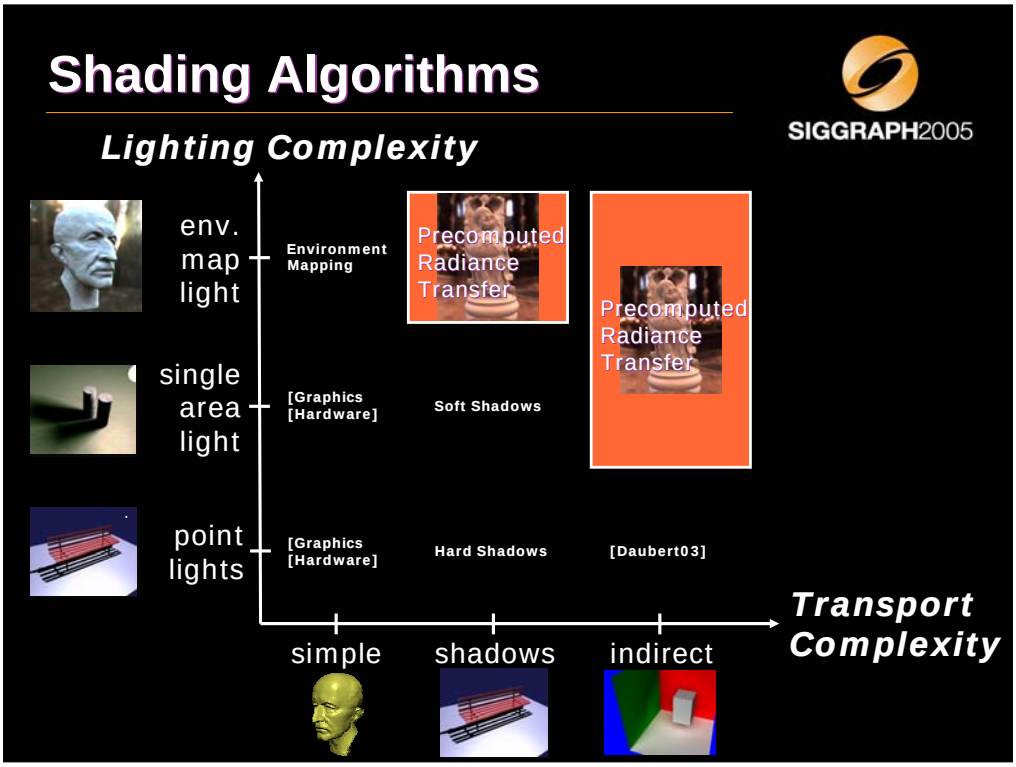
In real-time rendering, we usually make approximation to the full rendering equation, as it is too expensive to compute on the fly.

Common approximations are:

- only direct lighting (no indirect illumination at all, a simple ambient term can make up for it somewhat)

- point lights (area lights cast soft shadows, which are harder to compute, since the light source needs to be sampled)

- distant environment maps (this means that the same incident light arrives at all points of an object, and so there is no need to recompute it)



There are various previous algorithms that deal with real-time shading.

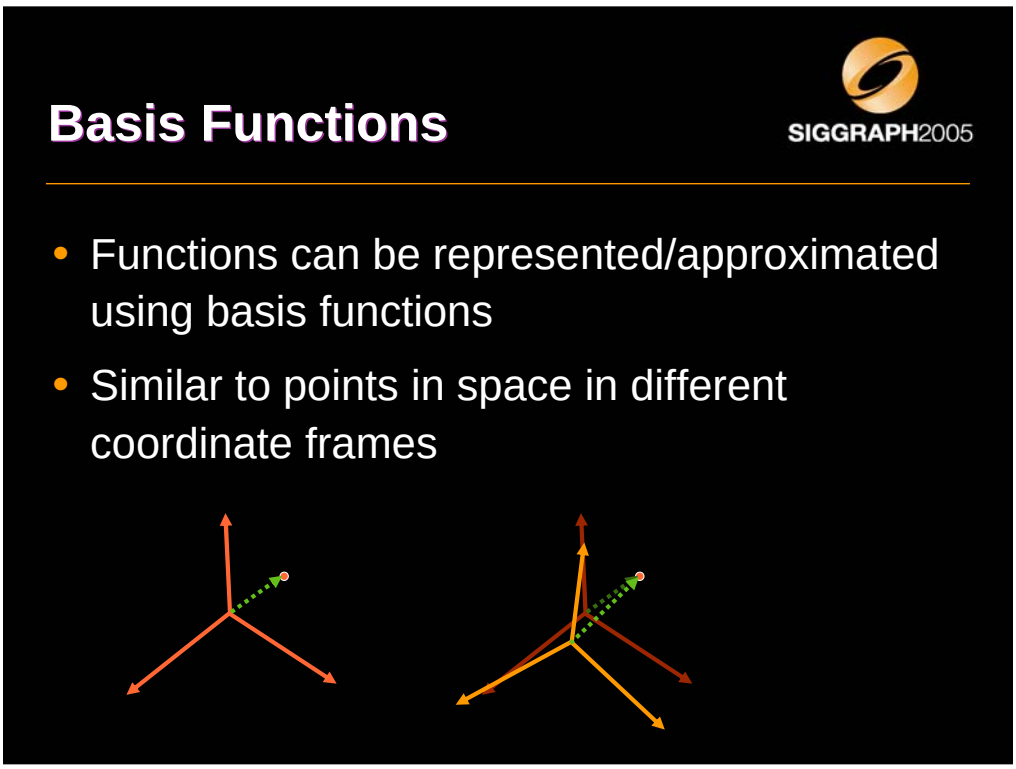
The graph here shows with what type of lighting complexity versus transport complexity they can deal with.

Precomputed Radiance Transfer fills the gap for natural illumination, which can arrive from all directions (distant environment map).

Overview

- Introduction
  - Rendering Equation
  - Neumann Series
  - Shading Algorithms
  - Basis Functions

Basis Functions

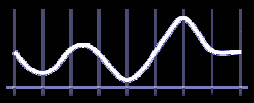


Basis Functions

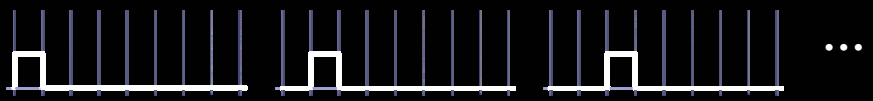
SIGGRAPH2005

Figures are courtesy of
Robin Green

- Given some function $f(t)$:



- Given basis functions $B_i(t)$:



$B_1(t)$ $B_2(t)$ $B_3(t)$

Basis Functions

SIGGRAPH2005

- Project into the function space:

$$\int \times = c_1$$

$$\int \times = c_2$$

$$\int \times = c_3$$

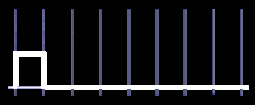
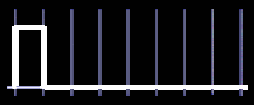
- Coefficients c_i represent function

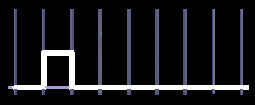
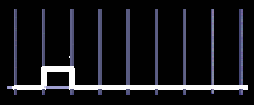
Projection is done by integrating the product of the function f(t) and the basis functions B_i(t)

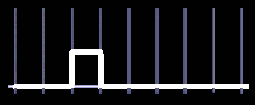
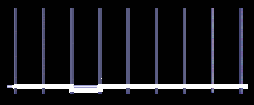
Basis Functions

SIGGRAPH2005

- Reconstruct original function
  - Weight each basis function with coefficient

$c_1 \times$  $=$ 

$c_2 \times$  $=$ 

$c_3 \times$  $=$ 

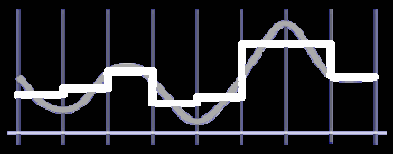
Reconstruction is done by scaling the basis functions with the coefficients c_i.

Basis Functions

SIGGRAPH2005

- Reconstruct original function
  - Sum them all up

$$\sum_{i=1}^{N} c_i B_i(x) \quad =$$



- In this case: approximation
- Benefit: fewer numbers needed

These scaled basis functions are then summed up.

Basis Functions

- Function: $f(t)$ (real-valued)
- Projection: $c_i = \int_\Omega f(t) B_i(t) dt =: \langle f(t), B_i(t) \rangle$
- Reconstruction: $f^R(t) = \sum_i^N c_i B_i(t)$
- Optimal for N basis functions (least-squares)

Basis Functions

- Orthogonal basis functions

$$\left\langle B_i(t), B_j(t) \right\rangle = \begin{cases} 0 & i \neq j \\ \neq 0 & i = j \end{cases}$$

- Orthonormal basis functions

$$\left\langle B_i(t), B_i(t) \right\rangle = 1$$

- Previously shown projection only holds for orthonormal basis functions!

Basis Functions

- Non-orthogonal basis functions

$$\left\langle B_i(t), B_j(t) \right\rangle = ?$$

- Projection cannot be done directly
- Define dual basis $\tilde{B}_k(t)$:

$$\left\langle B_i(t), \tilde{B}_k(t) \right\rangle = \begin{cases} 1 & i = k \\ 0 & \text{else} \end{cases}$$

Basis Functions

- Dual Basis $\tilde{B}_k(t)$
  - Is linear combination of basis functions $B_i(t)$
  - Compute using following equations

$$A_{ij} = \int B_i(t) B_j(t) dt$$

$$\tilde{B}_k(t) = \sum_j A^{-1}_{kj} B_j(t)$$

  - Matrix $\mathbf{A}$ is the Gram matrix of the basis

Basis Functions

- Projection into non-ortho. basis with dual

$$c_i = \left\langle f(t), \tilde{B}_i(t) \right\rangle$$

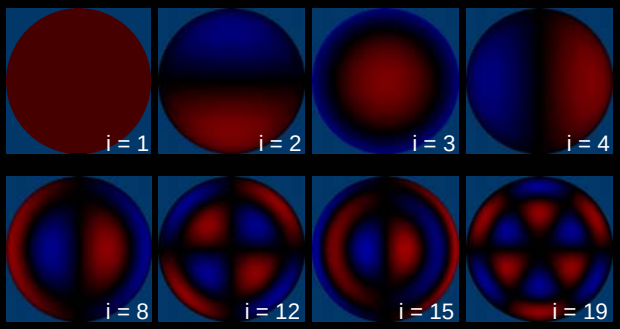
- Reconstruction with non-orthogonal basis

$$f^R(t) = \sum_i^N c_i B_i(t)$$

Basis Functions

SIGGRAPH2005

- Example: Spherical Harmonics $Y_l^m(\theta,\varphi)$
  - Orthonormal basis functions over the sphere
  - Similar to Fourier series



Basis Functions – Spherical Harmonics

- Recurrence formula for SH (band l, mode m)

$$Y_l^m(\theta,\varphi) = \begin{cases} \sqrt{2}K_l^m \cos(m\varphi)P_l^m(\cos\theta), & m > 0 \\ \sqrt{2}K_l^m \sin(-m\varphi)P_l^{-m}(\cos\theta), & m < 0 \\ K_l^0 P_l^0(\cos\theta), & m = 0 \end{cases}$$

- Here $P_l^m(\cos\theta)$ are the associated Legendre polynomials

The exact definitions can be looked up on the web. If only the first few bands are used, it is more efficient to use explicit formulas, which can be derived by using Maple for example. Here are the exact definitions for the first 25 basis functions. Input is a direction (in Cartesian coordinates). The evaluated basis function is written to ylm_array[].

```
float x = dir[0];
float y = dir[1];
float z = dir[2];
float x2, y2, z2;

ylm_array[0] = 0.282095f;     //l,m = 0,0   // 1/sqrt(4pi)
ylm_array[1] = 0.488603f * y; //1,-1    //sqrt(3/4pi)
ylm_array[2] = 0.488603f * z; //1,0
ylm_array[3] = 0.488603f * x; //1,1

x2 = x*x; y2 = y*y; z2 = z*z;
ylm_array[4] = 1.092548f * x * y; //2,-2   // sqrt(15/4pi)
ylm_array[5] = 1.092548f * y * z; //2,-1
ylm_array[6] = 0.315392f * (3.f*z2 - 1.f ); //2,0  // sqrt(5/16pi)
ylm_array[7] = 1.092548f * x * z; //2, 1
ylm_array[8] = 0.546274f * (x2 - y2); //2,2  // sqrt( 15/16pi )

const float fY30const = 0.37317633259011539141414395913199f; //0.25f*sqrt(7.f/M_PI);
const float fY31const = 0.45704579946446573615802069691 6f; //1.f/8.0f*sqrt(42.f/M_PI);
const float fY32const = 1.44530572132027702769469007719 9f; //0.25f*sqrt(105.f/M_PI);
const float fY33const = 0.59004358992664351034561027754 1f; //1.f/8.f*sqrt(70.f/M_PI);
ylm_array[ 9] = fY33const*y*(3.f*x2 - y2); //3,-3
ylm_array[10] = fY32const*2.f*x*y*z; // 3,-2
ylm_array[11] = fY31const*y*(5.f*z2-1.f); // 3,-1
ylm_array[12] = fY30const*z*(5.f*z2-3.f); // 3,0
ylm_array[13] = fY31const*x*(5.f*z2-1.f); // 3,1
ylm_array[14] = fY32const*z*(x2-y2); // 3,2
ylm_array[15] = fY33const*x*(x2-3.f*y2); // 3,3
```

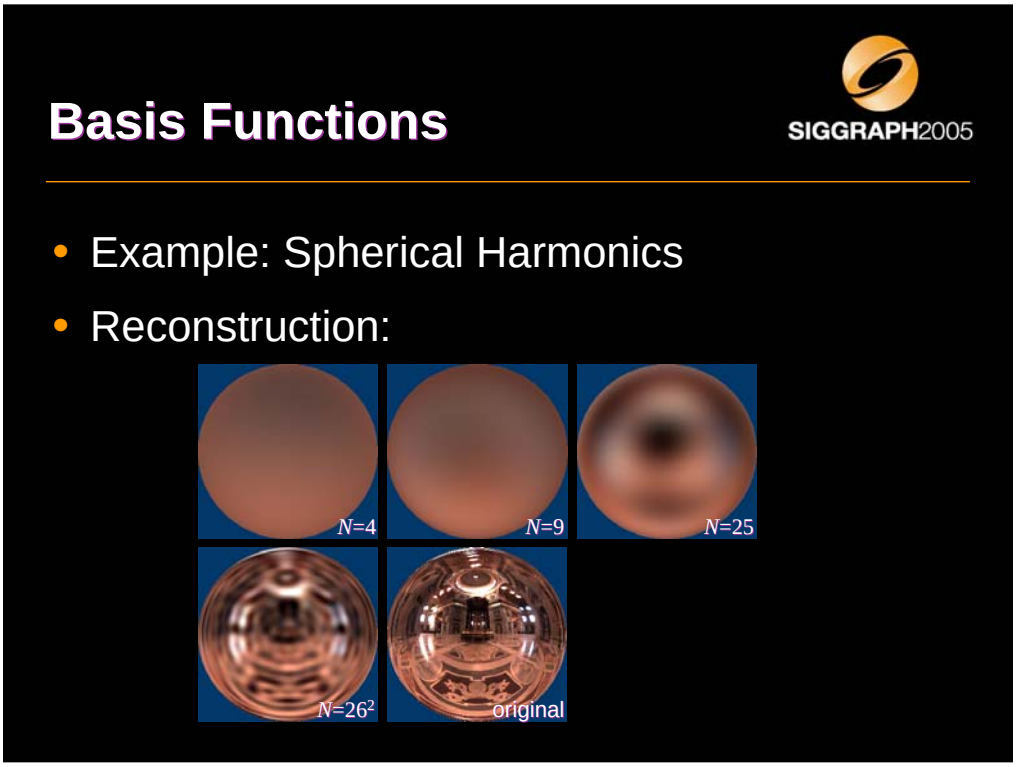
Basis Functions – Spherical Harmonics

- Example definitions (in Cartesian coords)

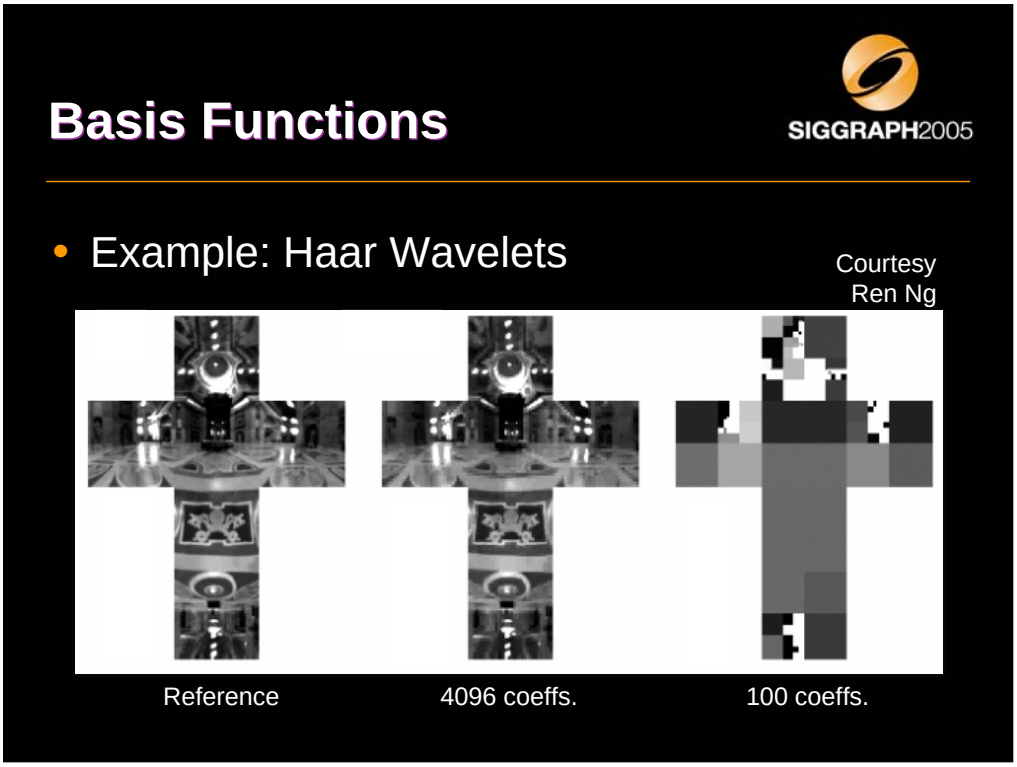
$$Y_1^0 = \frac{1}{2}\sqrt{\frac{3}{\pi}}z \quad Y_2^0 = \frac{1}{4}\sqrt{\frac{5}{\pi}}\left(2z^2 - x^2 - y^2\right) \quad Y_2^2 = \frac{1}{2}\sqrt{\frac{15}{\pi}}\left(x^2 - y^2\right)$$

$$Y_1^1 = \frac{1}{2}\sqrt{\frac{3}{\pi}}x \quad Y_2^1 = \frac{1}{2}\sqrt{\frac{15}{\pi}}zx$$

and so on...



Here an example environment map is projected in spherical harmonics and then back (using 4, 9, 25, and 26^2 basis functions).



Alternatively, the Haar wavelet basis functions can be used (instead of SH). We will talk briefly about this later on.

It should be noted, that wavelets are good at representing all-frequency detail (e.g. with 100 coeffs the bright windows are represented well, the not so important darker areas (floor) is represented with less accuracy).

The blocky appearance of the projection with 100 coefficients isn't as bad as it might look, if the BRDF is relatively dull, i.e., the environment will be blurred really heavily. In that case, the blocky appearance does not matter. In case of a very shiny BRDF, the blockiness will appear as blocky reflections.

Introduction

- Questions?


SIGGRAPH2005