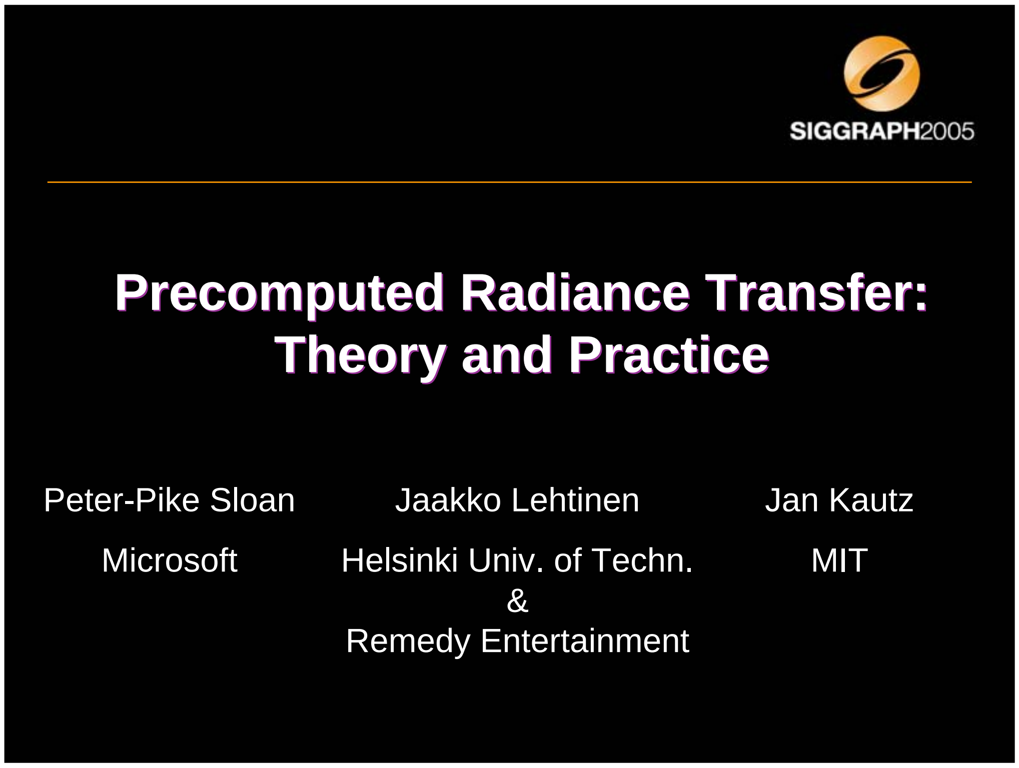
SIGGRAPH2005



SIGGRAPH2005

Precomputed Radiance Transfer: Theory and Practice

Peter-Pike Sloan
Microsoft

Jaakko Lehtinen
Helsinki Univ. of Techn.
&
Remedy Entertainment

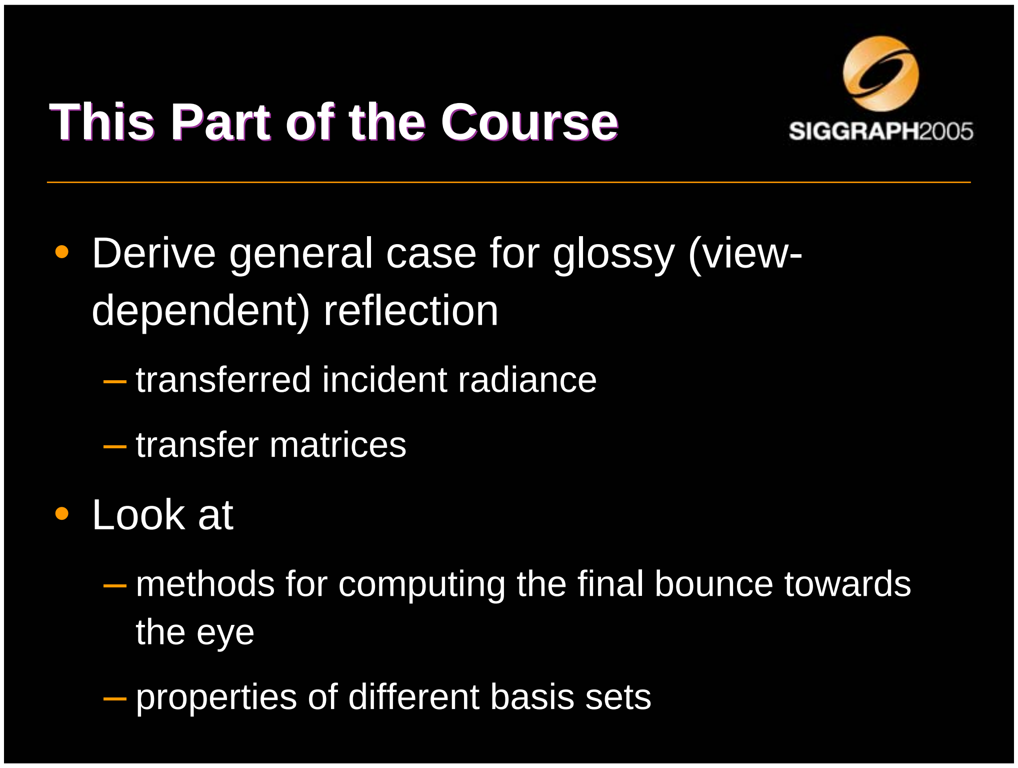
Jan Kautz
MIT

SIGGRAPH2005

General PRT

Jaakko Lehtinen

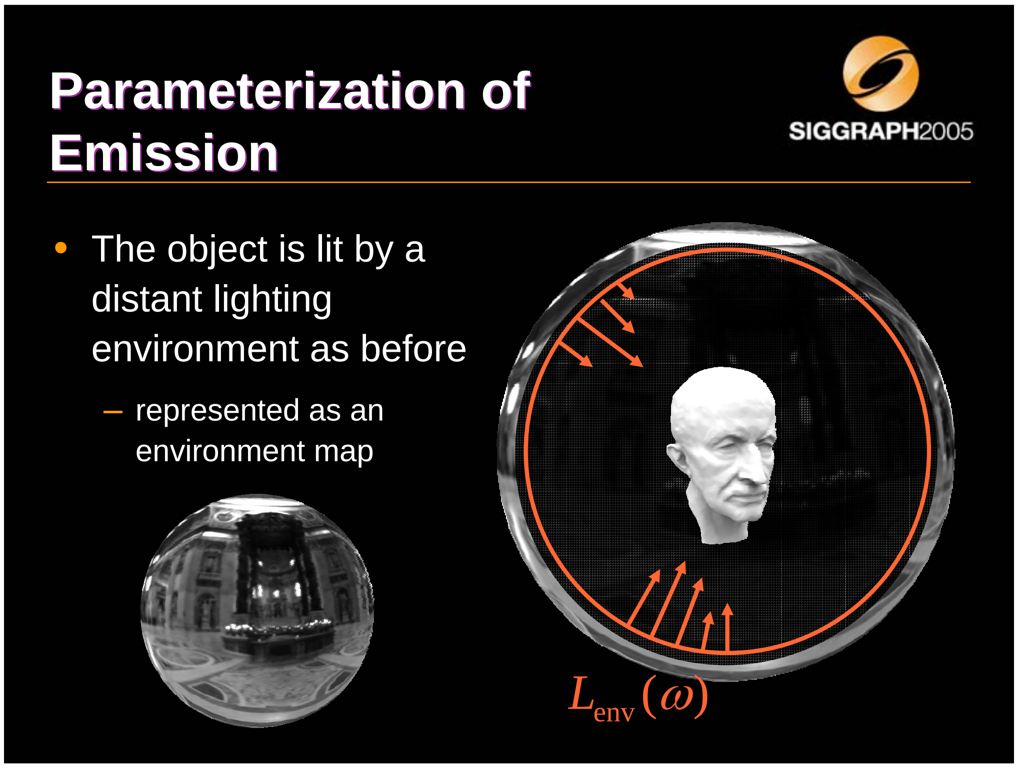
Helsinki University of Technology,
Remedy Entertainment Ltd.



This Part of the Course

SIGGRAPH2005

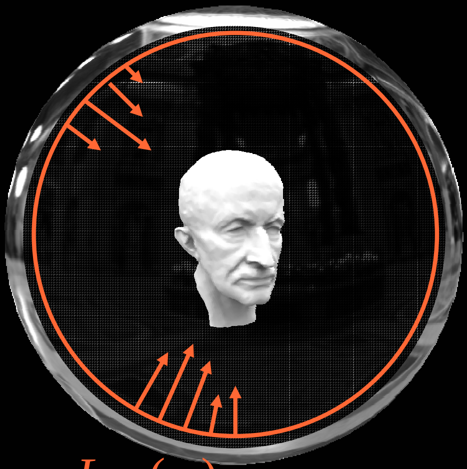
- Derive general case for glossy (view-dependent) reflection
  - transferred incident radiance
  - transfer matrices
- Look at
  - methods for computing the final bounce towards the eye
  - properties of different basis sets



Parameterization of Emission

SIGGRAPH2005

- The object is lit by a distant lighting environment as before
  - represented as an environment map

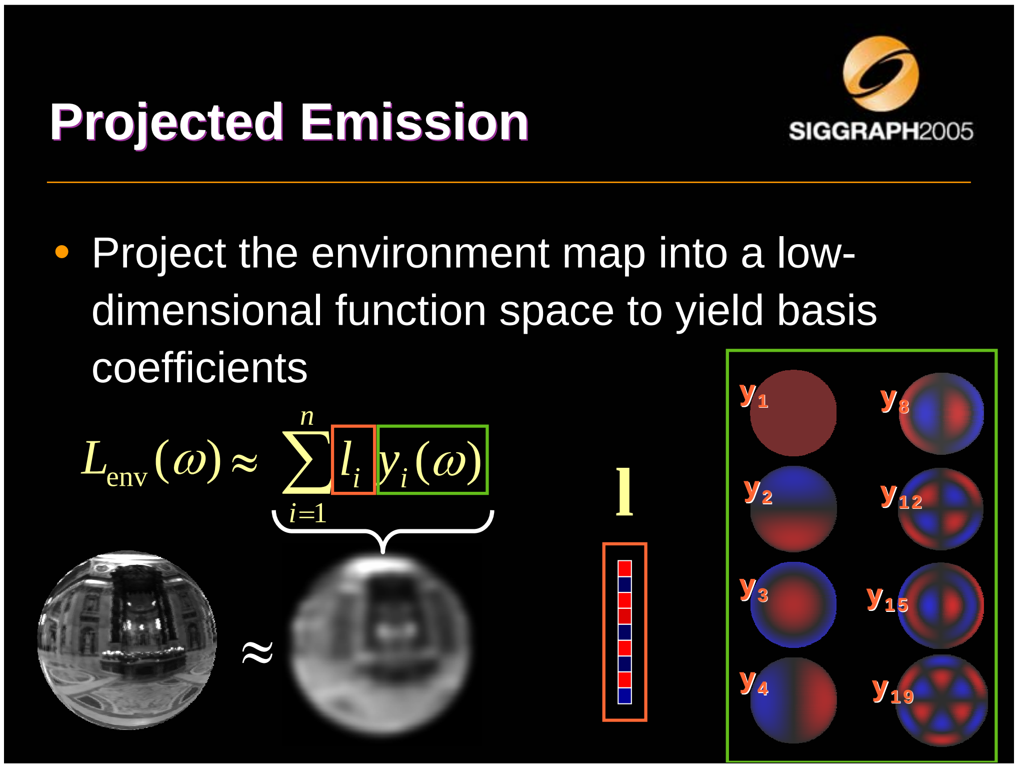


$L_{\text{env}}(\omega)$

As before, we'll be looking at an object sitting in free space, distant from the lighting environment that illuminates it. Extensions are possible, but we'll go through this simple case not to be overly general.

"Distant" means that location on the object does not affect the directional distribution of incident light. In other words, in absence of occlusion, the direct light incident upon all points on the object is the same, i.e., L_env(p,omega) = L_env(omega) for all p.

We usually represent the lighting environment using an a so-called "light probe" image, but it is also possible to use analytic area lights (cones, etc.). These will be covered in a subsequent part of the course.



Projected Emission

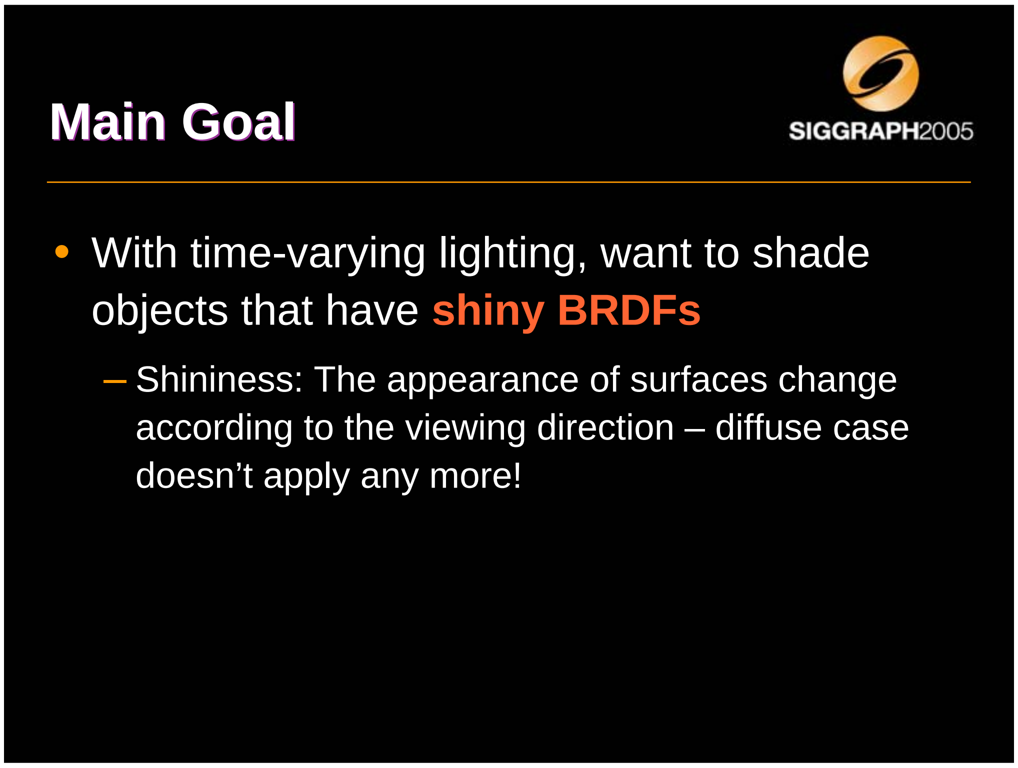
SIGGRAPH2005

- Project the environment map into a low-dimensional function space to yield basis coefficients

$$L_{\text{env}}(\omega) \approx \underbrace{\sum_{i=1}^{n} \boxed{l_i} \boxed{y_i(\omega)}}$$

≈
l
y1
y2
y3
y4
y8
y12
y15
y19

As described in the earlier section, the environment map that we use for lighting the object is projected into a function basis {y_i} with i=1,…,n. This yields an approximation to the original environment map, and the approximation is fully defined by the vector **l** of coefficients.



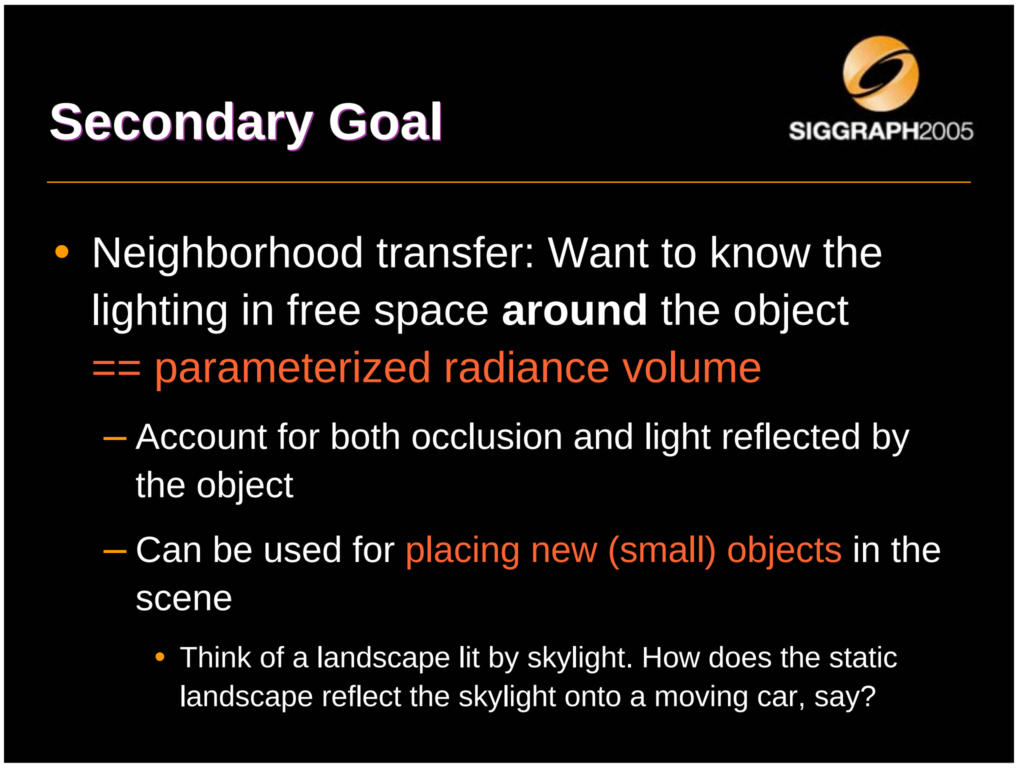
Main Goal

SIGGRAPH2005

- With time-varying lighting, want to shade objects that have **shiny BRDFs**
  - Shininess: The appearance of surfaces change according to the viewing direction – diffuse case doesn’t apply any more!

What we want to do: Global illumination with glossy reflections and time-varying lighting. Shiny BRDFs mean that the appearance of a point changes depending on where it is looked at from. This means that the diffuse case described earlier doesn’t work any more.

Here we deal exclusively with scenes and objects that are rigid, i.e., they do not change their shapes. It is possible to rotate the object (that corresponds just to an opposite rotation of the incident lighting). Real-time Global Illumination for deforming scenes is very challenging.



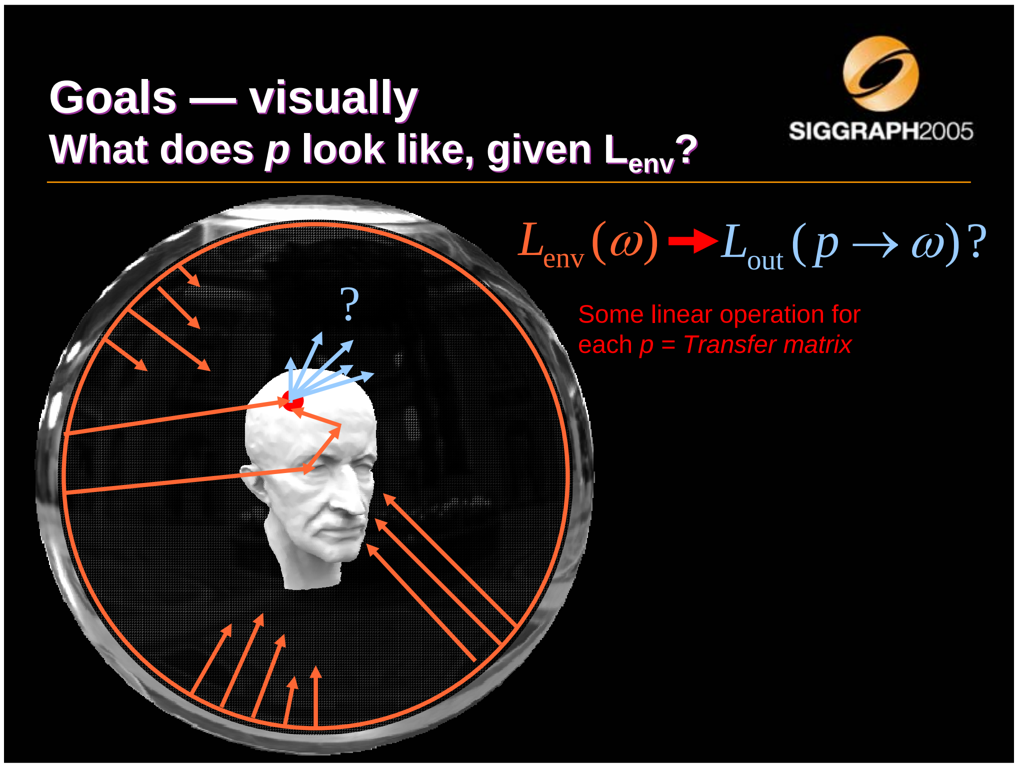
Secondary Goal

SIGGRAPH2005

- Neighborhood transfer: Want to know the lighting in free space **around** the object
  == parameterized radiance volume
  - Account for both occlusion and light reflected by the object
  - Can be used for placing new (small) objects in the scene
    - Think of a landscape lit by skylight. How does the static landscape reflect the skylight onto a moving car, say?

We also want to approximate the lighting in FREE SPACE around the object. **Why this is useful**: Think of the object being a landscape scene with mountains, hills, valleys etc. If we can compute the lighting incident to points in free space in the landscape, we can shade moving objects (vehicles, characters, say) with lighting that is affected by the environment, and thus get color bleeding and soft shadowing effects onto the moving objects.

However, the effect of the object on the scene – shadows, etc. -- needs to be handled separately, then. Some work into that direction has been presented by my Finnish colleagues at I3D this year, and will be presented here at Siggraph this week by people from MSR Asia.



Goals — visually

What does *p* look like, given L_{env}?

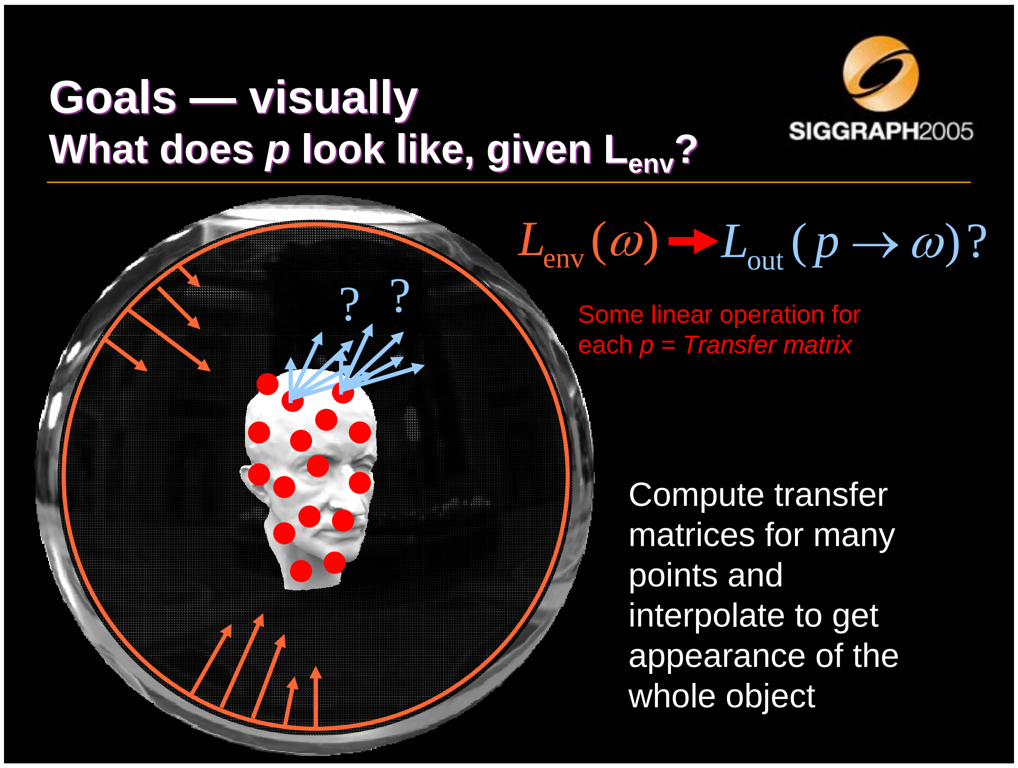
SIGGRAPH2005

$L_{\text{env}}(\omega) \rightarrow L_{\text{out}}(p \rightarrow \omega)?$

Some linear operation for each *p* = *Transfer matrix*

So, what we want to know is: Given some projected environment map, what do points on the object look like? That is, what is the directional distribution of outgoing light from each point? If we know these values, we can directly assign to pixels in an image.

The appearance of each point is linearly related to the distant lighting environment, so we are looking for some linear operator that takes as input the lighting environment, and produces the appearance of the point as output. In the end, we will represent this appearance in some linear basis, so the linear mapping will be, for each point, a matrix that maps the coefficients of incident, distant lighting environment into coefficients for outgoing radiance for the point p.



Goals — visually

What does *p* look like, given L_{env}?

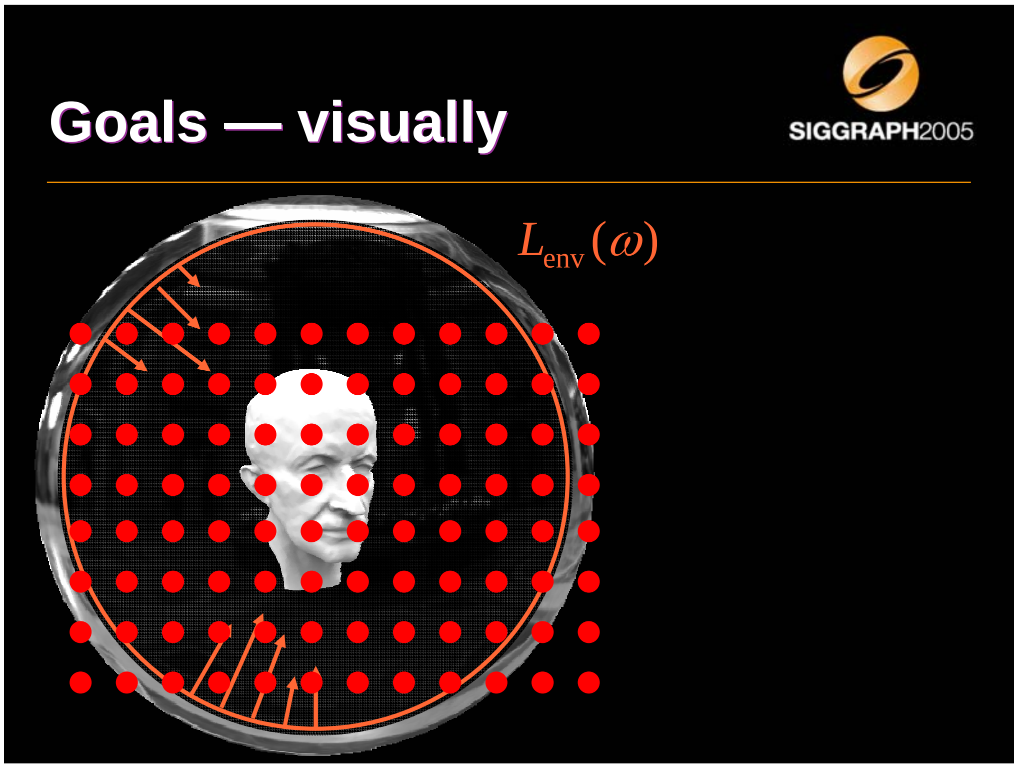
SIGGRAPH2005

$L_{env}(\omega) \rightarrow L_{out}(p \rightarrow \omega)?$

Some linear operation for each *p* = *Transfer matrix*

Compute transfer matrices for many points and interpolate to get appearance of the whole object

We’ll compute this mapping for many points on the object, so that we can get a reasonable approximation for the appearance of the whole object by interpolating from these samples.

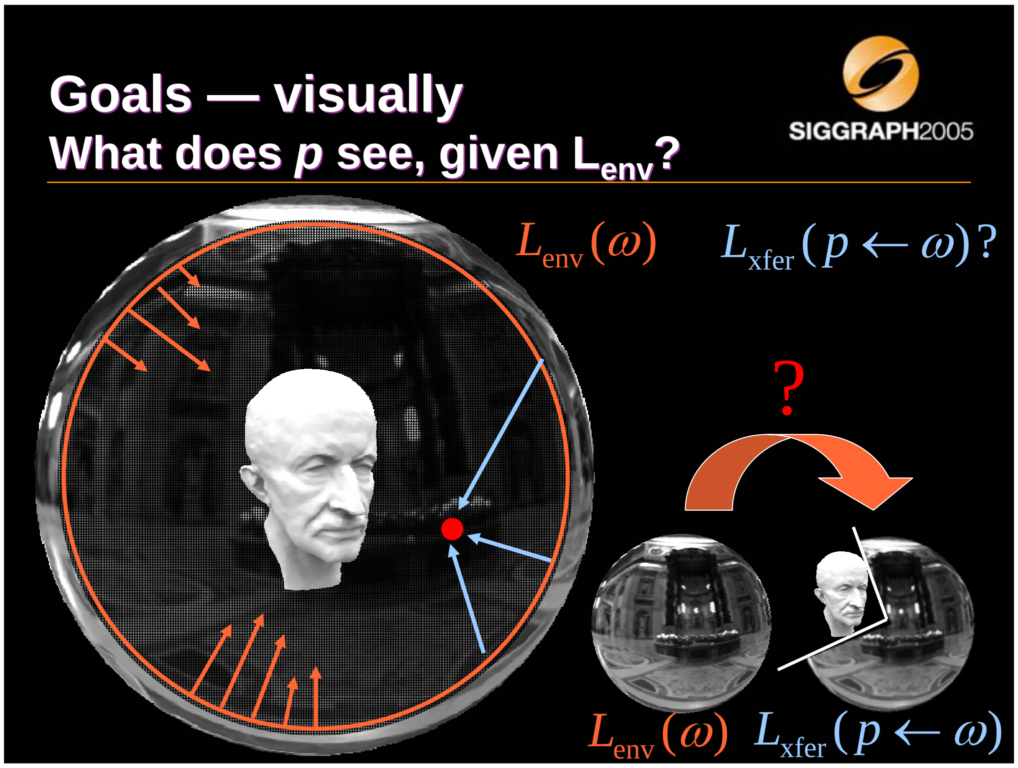


Goals — visually

SIGGRAPH2005

The other question: What do the points near but not on the object “see” when the object is illuminated using an environment map? In other words, how does the presence of the object affect the lighting that impinges upon these points in free space?

Note that we are NOT talking about a camera for rendering pictures. Instead, we are interested in placing new, tiny objects in the scene, and thus we need to know the lighting incident to the point in order to shade the new object realistically.



Goals — visually

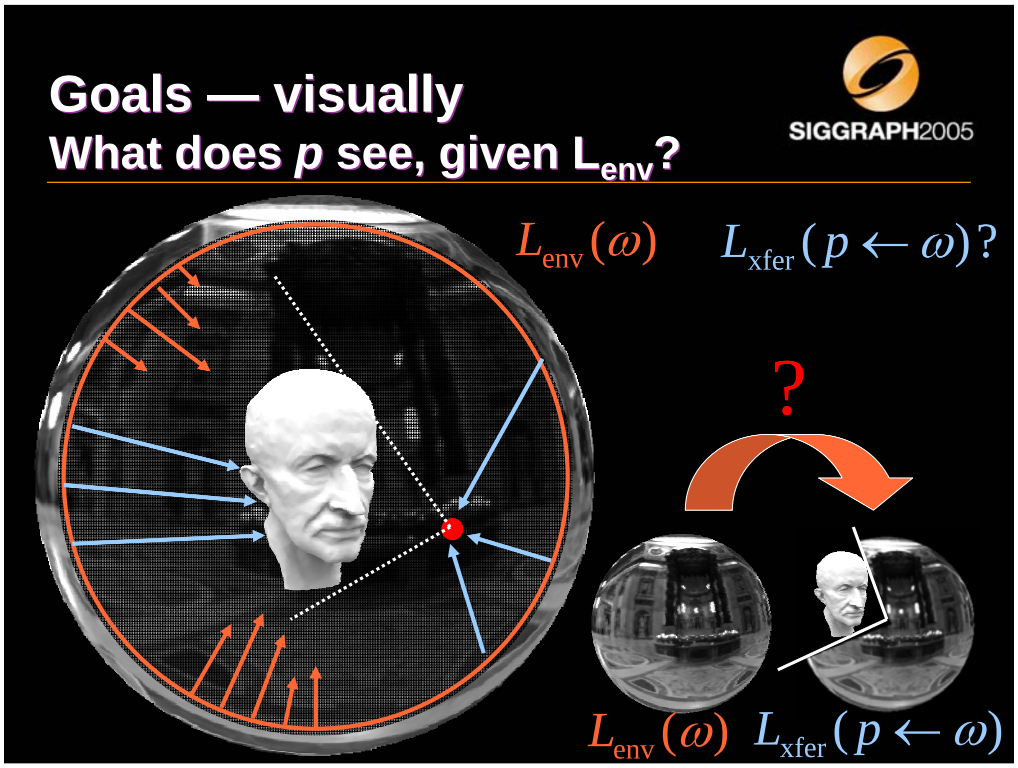
What does *p* see, given L_{env}?

SIGGRAPH2005
$L_{\text{env}}(\omega)$
$L_{\text{xfer}}(p \leftarrow \omega)$?
?
$L_{\text{env}}(\omega)$
$L_{\text{xfer}}(p \leftarrow \omega)$

Let’s concentrate on a single point somewhere near the object.

Here you see the distant lighting environment, and on the right we have a schematic drawing of what point p sees when the scene is lit using the source environment map.

Some light proceeds directly from the lighting environment to point p; that happens in those directions where the object doesn’t block the view from p.

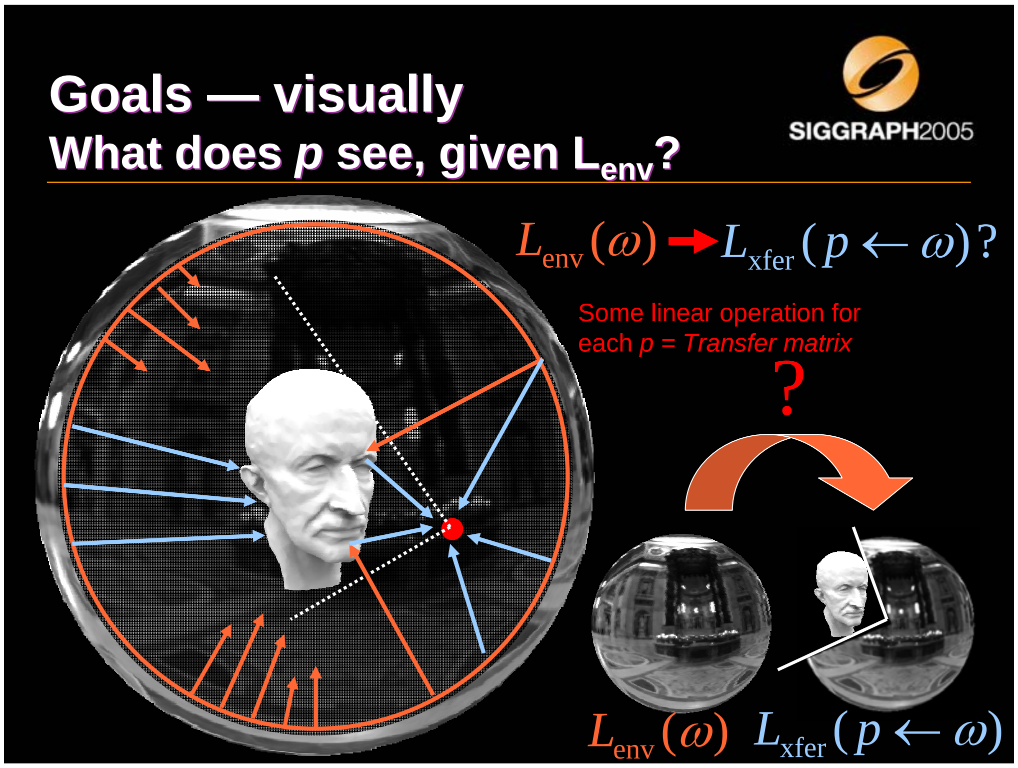


Goals — visually

What does *p* see, given L_{env}?

SIGGRAPH2005
$L_{\text{env}}(\omega)$
$L_{\text{xfer}}(p \leftarrow \omega)$?
?
$L_{\text{env}}(\omega)$
$L_{\text{xfer}}(p \leftarrow \omega)$

Some light that leaves the distant environment will be blocked by the object. That happens in directions where the object blocks the view from p to the environment. We denote that by cutting away a part of the environment here on the right.



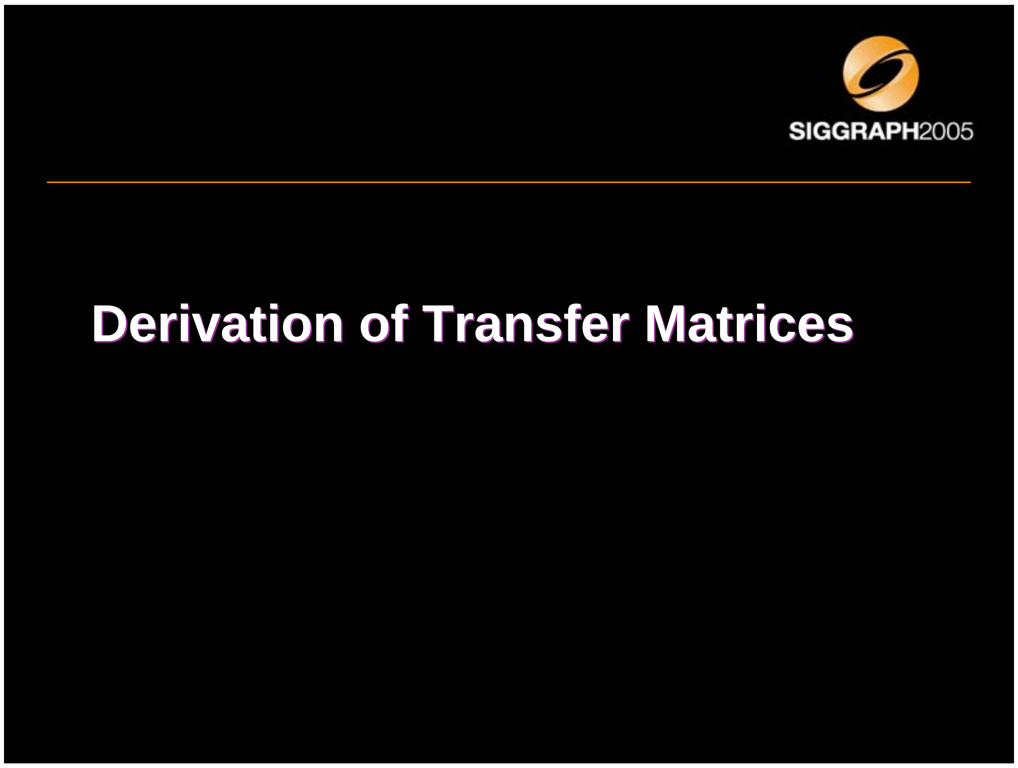
Goals — visually

What does *p* see, given L_{env}?

SIGGRAPH2005
$L_{\mathrm{env}}(\omega) \rightarrow L_{\mathrm{xfer}}(p \leftarrow \omega)?$
Some linear operation for each *p* = *Transfer matrix*
?
$L_{\mathrm{env}}(\omega)$
$L_{\mathrm{xfer}}(p \leftarrow \omega)$

Some light takes a bounce or multiple bounces off the object before impinging upon point p. In other words, p sees an image of the what our object looks like when lit by the environment map.

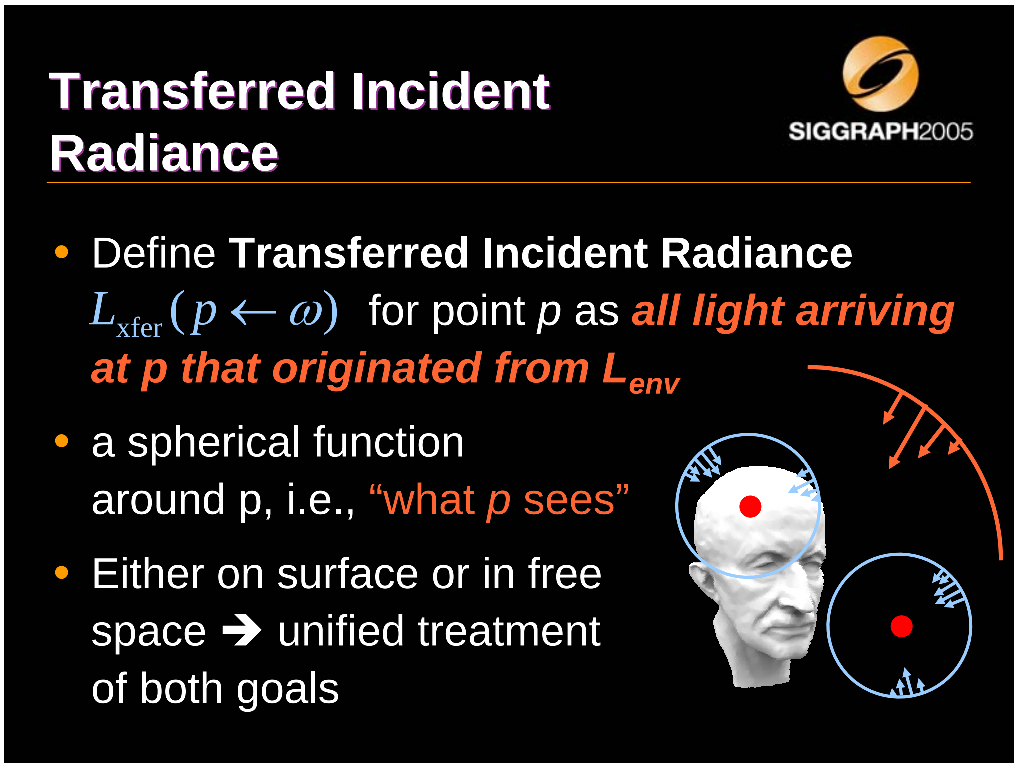
The relationship between the lighting environment and what point p sees is linear; we will in the following project the light impinging on p in a finite function space, and thus this relationship will be characterized by a matrix. This matrix will in general be different for each point.



Derivation of Transfer Matrices

SIGGRAPH2005

In this section we will derive the exact relationship of the distant lighting environment and the lighting after it has interacted with the scene.



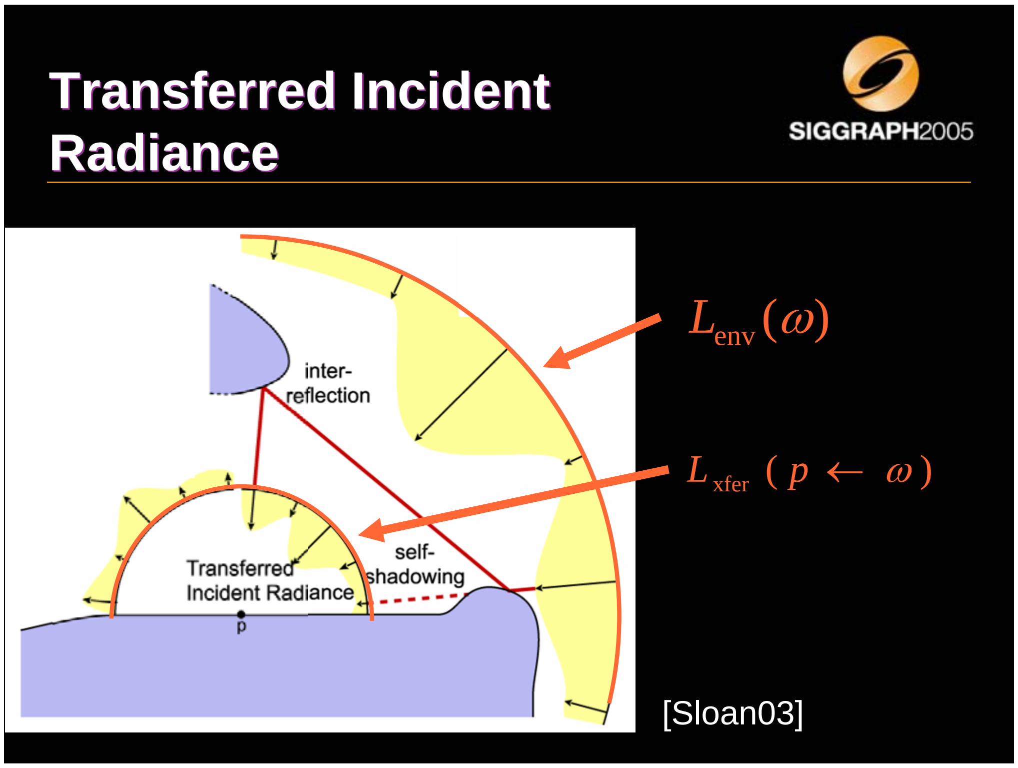
Transferred Incident Radiance

SIGGRAPH2005

- Define **Transferred Incident Radiance** $L_{\mathrm{xfer}}(p \leftarrow \omega)$ for point *p* as ***all light arriving at p that originated from L_{env}***
- a spherical function around p, i.e., "what *p* sees"
- Either on surface or in free space ➔ unified treatment of both goals

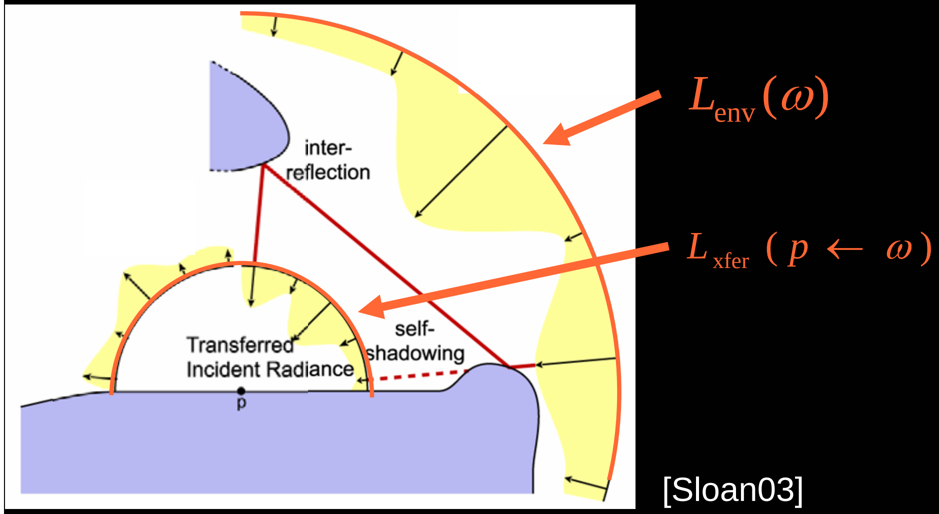
To get going, we first define a helper quantity called *transferred incident radiance*.

For each point in the scene, be it on the surface of the object or in free space, we define it as the radiance incident onto that point from each direction. This is different from the distant environment, since the light is blocked and reflected by the object before impinging upon the point. In other words, it is a spherical function for each point in the scene, and it defines "what point p sees", and it of course changes as the lighting environment changes.

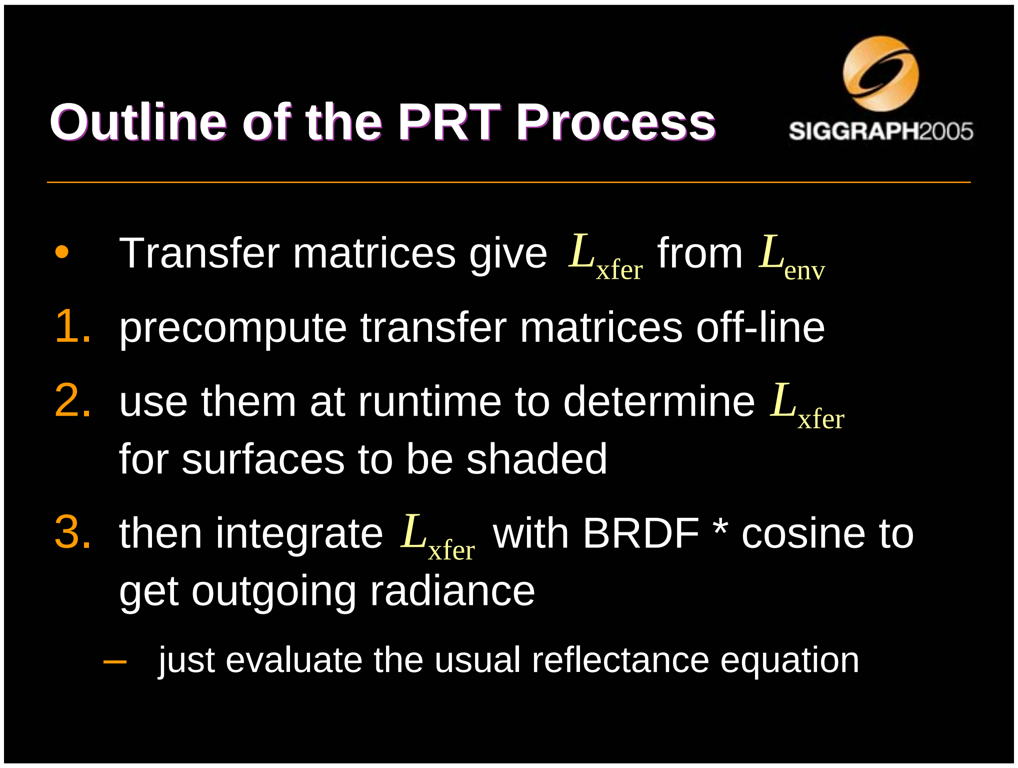


Transferred Incident Radiance

SIGGRAPH2005



[Sloan03]



Outline of the PRT Process

- Transfer matrices give L_{xfer} from L_{env}

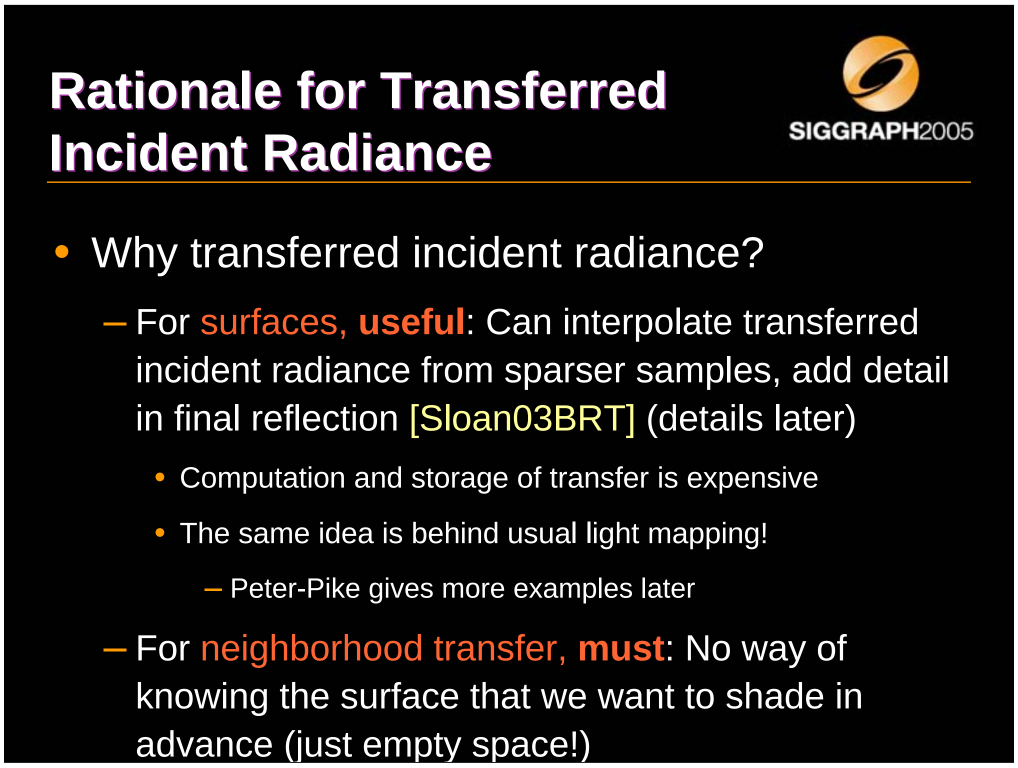
1. precompute transfer matrices off-line
2. use them at runtime to determine L_{xfer} for surfaces to be shaded
3. then integrate L_{xfer} with BRDF * cosine to get outgoing radiance
   – just evaluate the usual reflectance equation

The basic idea is that we'll precompute the linear operators (that is, matrices) that map our distant lighting environment into transferred incident radiance for discrete set of points in the scene, and at runtime we'll use them for computing transferred incident radiance at those points, given the current lighting environment. Obviously, the light going out from a point on the object is the sum of emitted light and reflected light, and we get reflected light by computing the usual reflectance integral of the transferred incident lighting. This yields final reflected radiance towards the virtual camera.

The matrices that map the incident lighting into transferred incident radiance will be precomputed off-line.

At runtime, given the current lighting environment, these matrices will be used for computing the transferred incident radiance at points that we want to shade.

Then the transferred incident radiance is reflected once more for computing the intensity of light flowing from *p* towards the viewing direction omega_out.



Rationale for Transferred Incident Radiance

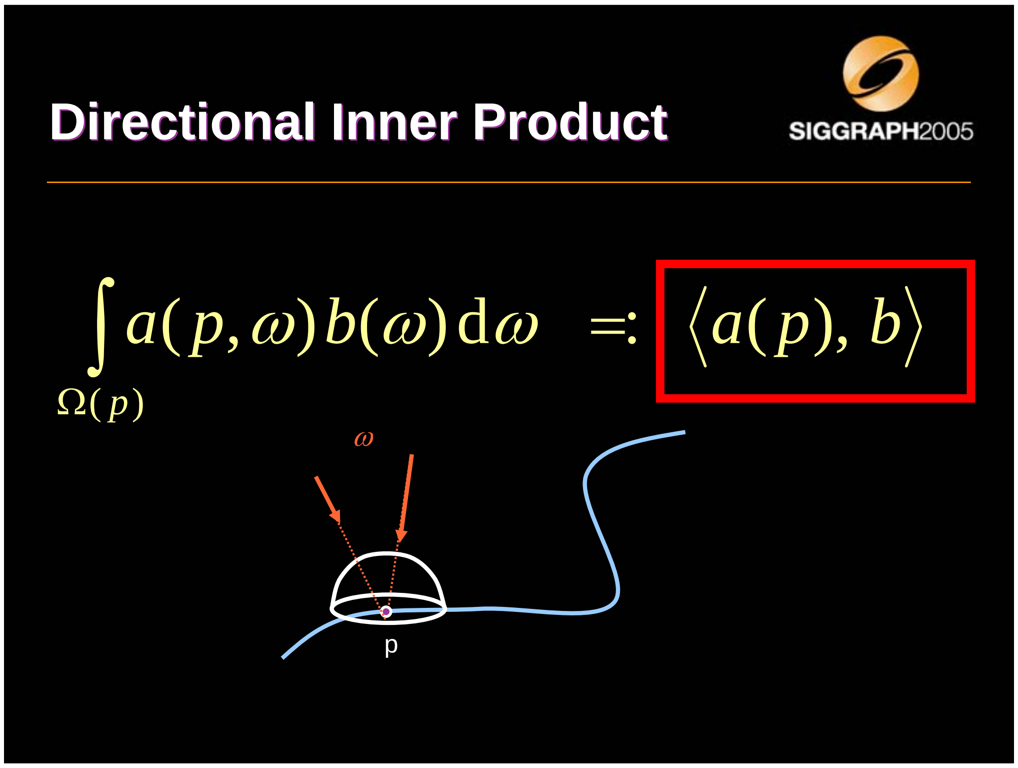
SIGGRAPH2005

- Why transferred incident radiance?
  - For surfaces, **useful**: Can interpolate transferred incident radiance from sparser samples, add detail in final reflection [Sloan03BRT] (details later)
    - Computation and storage of transfer is expensive
    - The same idea is behind usual light mapping!
      - Peter-Pike gives more examples later
  - For neighborhood transfer, **must**: No way of knowing the surface that we want to shade in advance (just empty space!)

So why bother with transferred incident radiance and not compute outgoing radiance directly? For surfaces of the object, this is certainly possible. But it is also possible to decouple the sampling rates of transferred incident radiance and outgoing radiance. The rationale is that transferred incident radiance often (but not always) varies quite slowly over space, whereas the outgoing radiance from a surface often has high spatial frequencies. Thus, the transferred incident radiance can often be interpolated from sparser samples, and turned into higher-frequency outgoing radiance (*bi-scale radiance transfer*). Of course, we want to avoid as much work as possible, and computing the transfer is expensive.

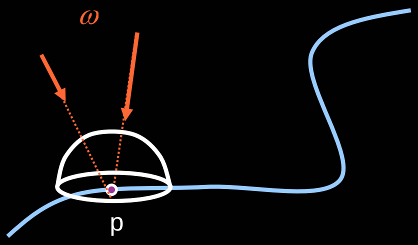
The same idea makes usual light maps work: You have a coarser representation of incident irradiance (from the light map) that gets turned into higher-frequency outgoing radiance by multiplication with a higher-resolution texture map. In effect you interpolate incident lighting from sparser samples and add detail in the final reflection. Peter-Pike has more examples of this in a later part of the course.

If we want to place a new object in the scene, we need to know the light incident upon the object from all directions, and thus we need transferred incident radiance.



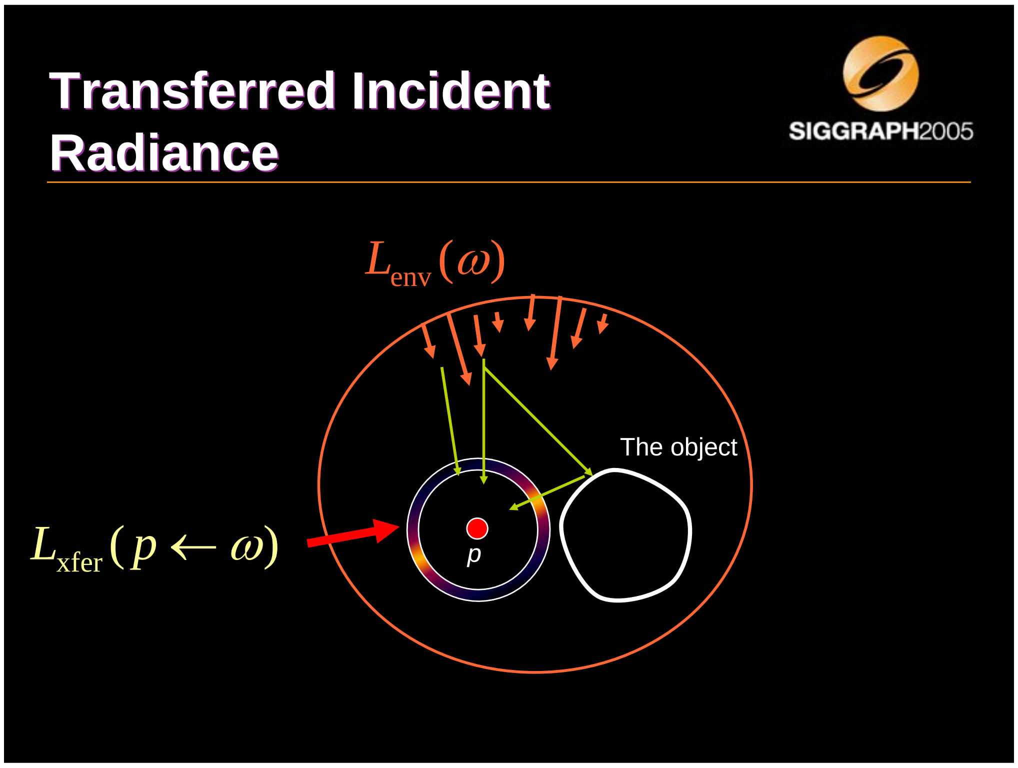
Directional Inner Product

$$\int_{\Omega(p)} a(p,\omega)\,b(\omega)\,\mathrm{d}\omega \;=:\; \boxed{\langle a(p),\, b\rangle}$$



In the rest of the presentation we'll denote a directional inner product using wedges like this.

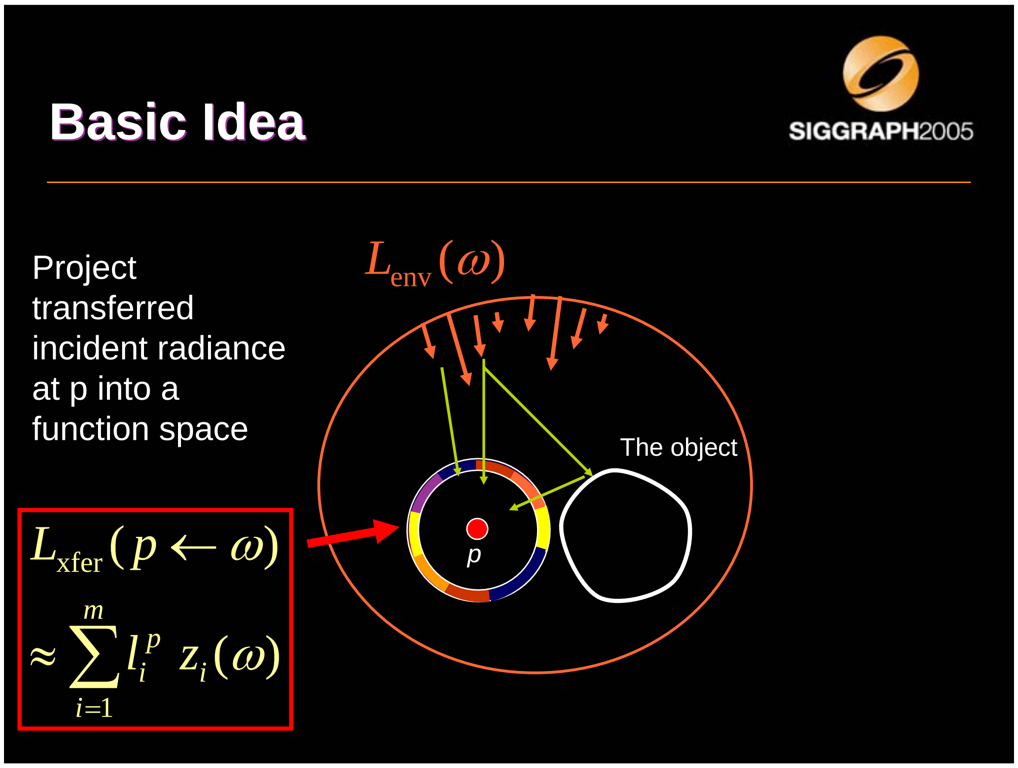
The arguments to this inner product are always two functions: One that varies both in space (p) and direction (omega), and another that varies only in the direction domain (omega). We'll use this notation for getting projection coefficients for different functions.



Transferred Incident Radiance

SIGGRAPH2005

As we've stated before, transferred incident radiance at point p is a spherical function that tells you "what p sees" when the scene is lit by Lenv.



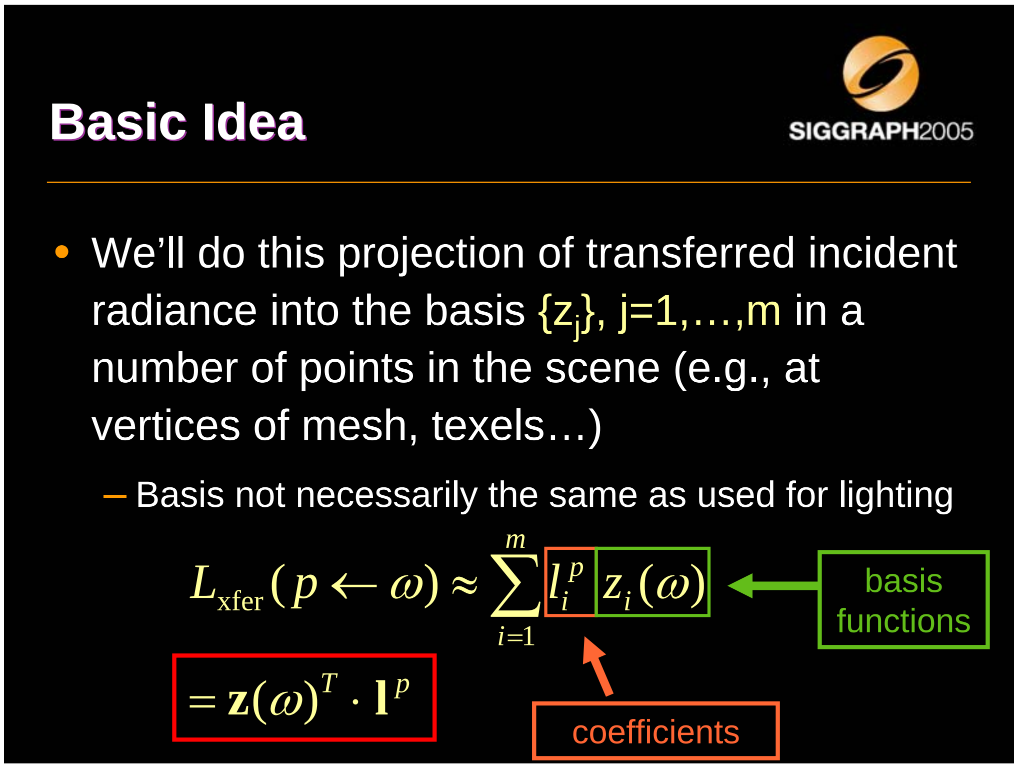
Basic Idea

SIGGRAPH2005

Project transferred incident radiance at p into a function space

$$L_{\text{xfer}}(p \leftarrow \omega) \approx \sum_{i=1}^{m} l_i^p \, z_i(\omega)$$

To start the way towards deriving the transfer matrix that maps the distant illumination into transferred lighting, we project the transferred lighting into a finite, spherical function space spanned by functions {z}; that is, we **discretize** transferred incident radiance and represent it using a finite number of basis coefficients that we denote by l^p.



Basic Idea

SIGGRAPH2005

- We’ll do this projection of transferred incident radiance into the basis $\{z_j\}$, $j=1,\ldots,m$ in a number of points in the scene (e.g., at vertices of mesh, texels…)
  - Basis not necessarily the same as used for lighting

$$L_{\text{xfer}}(p \leftarrow \omega) \approx \sum_{i=1}^{m} \boxed{l_i^p}\boxed{z_i(\omega)}$$

basis functions

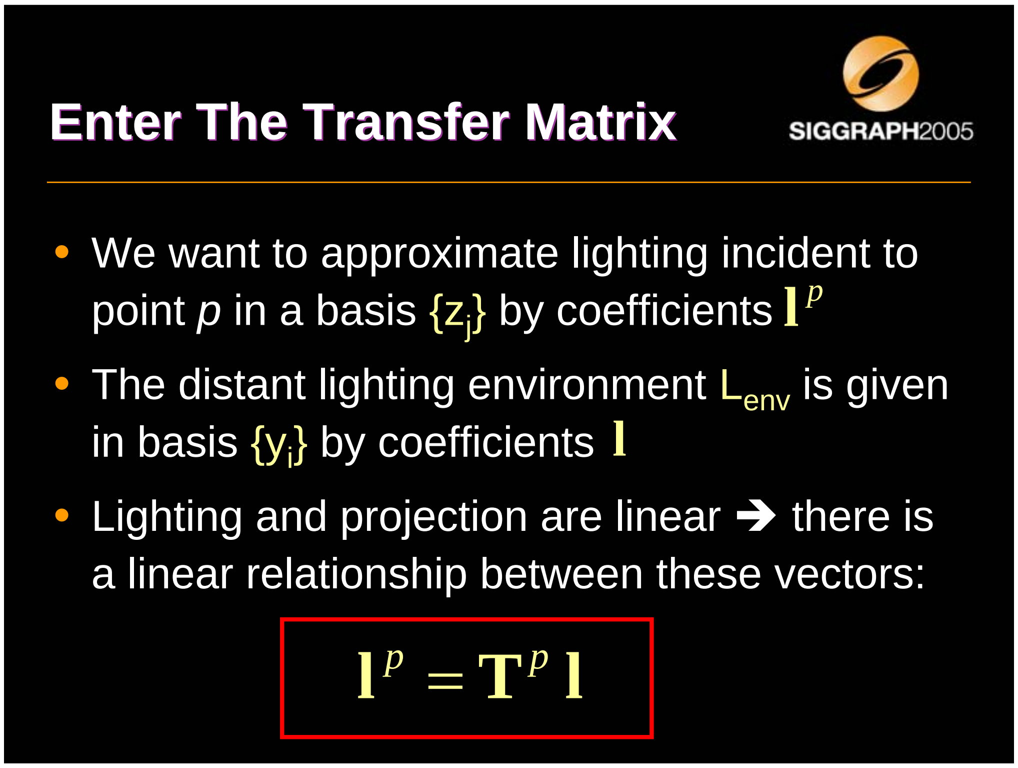
$$\boxed{= \mathbf{z}(\omega)^T \cdot \mathbf{l}^p}$$

coefficients

We’ll start our way towards deriving the transfer matrices that map the incident illumination into transferred incident radiance.

It all starts from representing transferred incident radiance for point p in a function space spanned by spherical basis functions z. The coefficient vector that describes this approximation of transferred incident radiance is denoted by **l**^p.

Note that what we’ll do here corresponds to a simple discretization of the continuous rendering equation, but we skip the details of this.



Enter The Transfer Matrix

SIGGRAPH2005

- We want to approximate lighting incident to point *p* in a basis $\{z_j\}$ by coefficients $\mathbf{l}^p$
- The distant lighting environment L_{env} is given in basis $\{y_i\}$ by coefficients $\mathbf{l}$
- Lighting and projection are linear ➔ there is a linear relationship between these vectors:

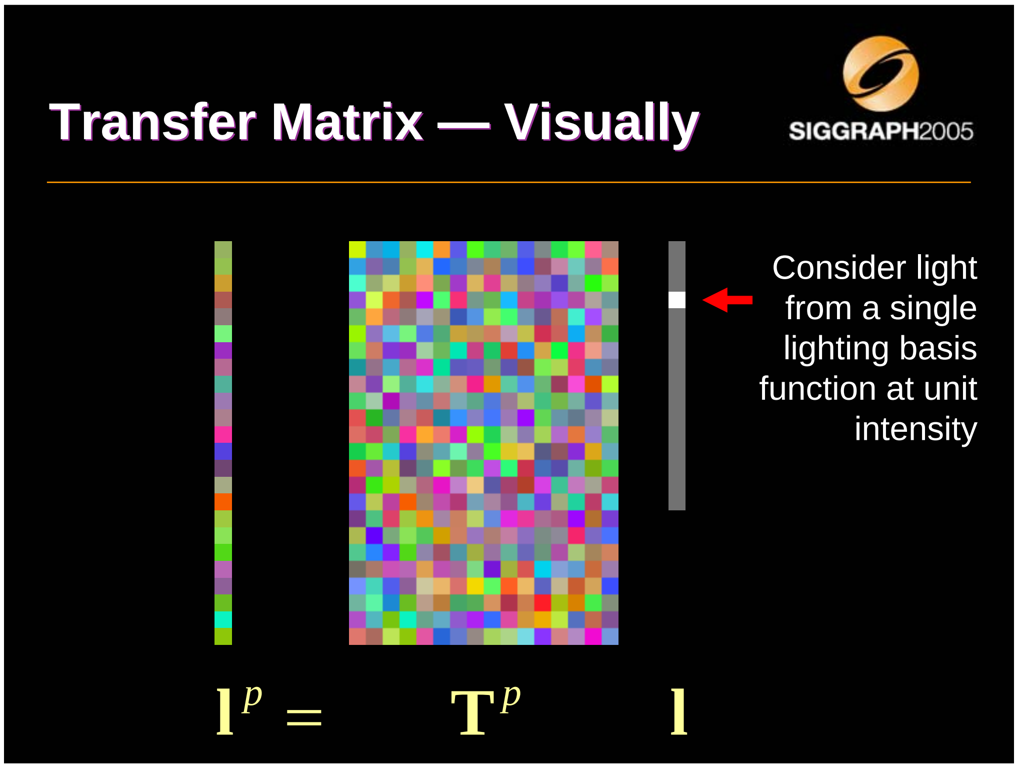
$$\boxed{\mathbf{l}^p = \mathbf{T}^p\,\mathbf{l}}$$

So, we want a “spherical image” that gives the light incident to p; since we use a basis expansion, the image is defined by the coefficient vector **l**^p.

The coefficients **l**^p are unknown, the coefficients **l** that define the distant incident lighting we know.

What we want to do is find their relationship; this relationship between the two vectors is linear, and thus it can be expressed using a matrix. We call this matrix the **transfer matrix** for point p: It maps the distant, incident illumination to an approximation of the lighting that reaches *p*.

We’ll come to what the matrix looks like next…



Transfer Matrix — Visually

SIGGRAPH2005

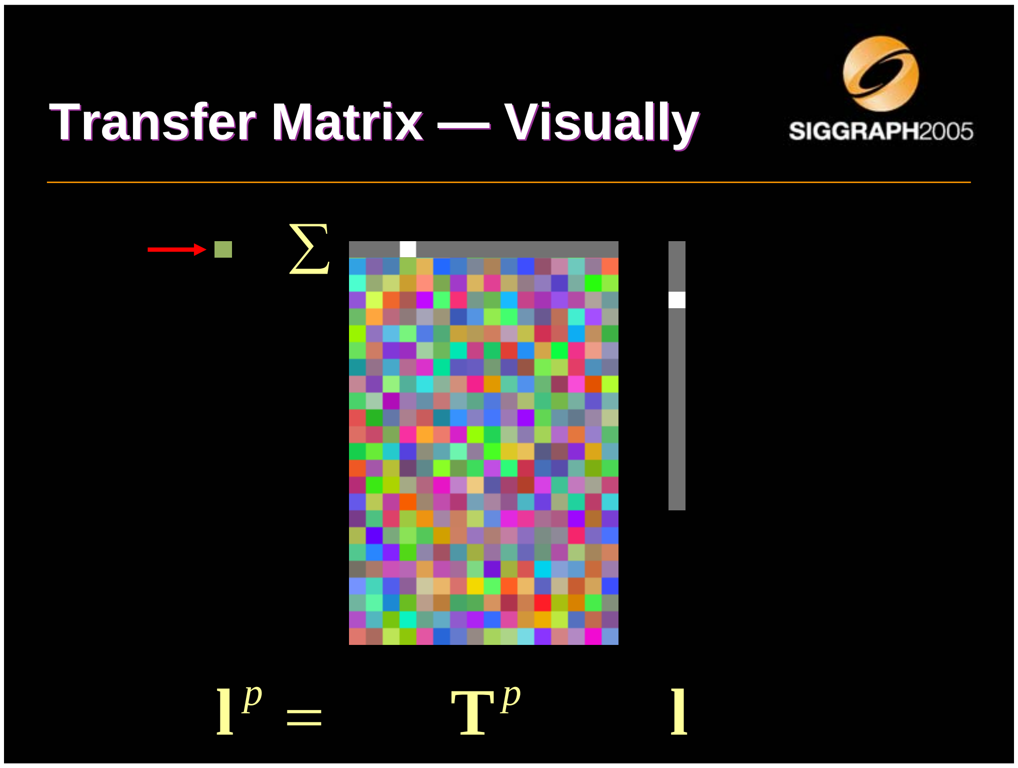
Consider light from a single lighting basis function at unit intensity

$$\mathbf{l}^p = \mathbf{T}^p \; \mathbf{l}$$

Let’s think about that for a while. What happens if the scene is lit by just a single lighting basis function y_i, i.e., there is just a single ONE in the emission vector **l** and the rest is just zero?

That’s marked there in the visual depiction as the white box in the gray vector.

If we think of the matrix-vector product, what happens here?

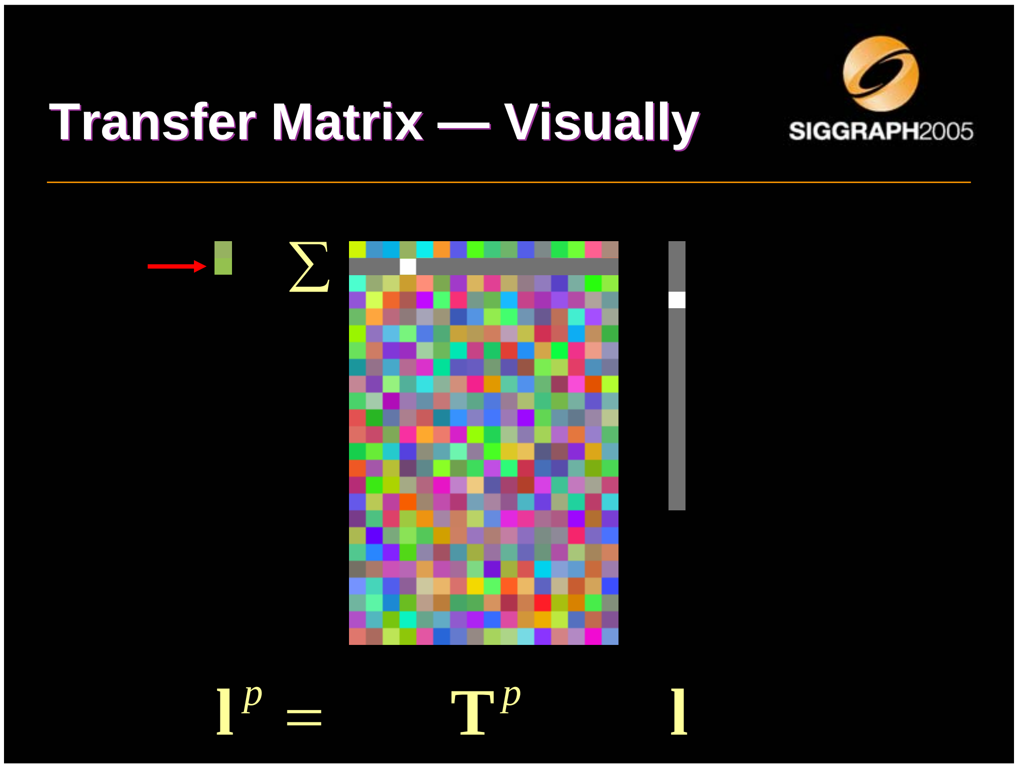


Transfer Matrix — Visually

SIGGRAPH2005

$$\mathbf{l}^p = \sum \mathbf{T}^p \quad \mathbf{l}$$

Take the dot product of the vector l with the first row == copy the fourth entry over to the result

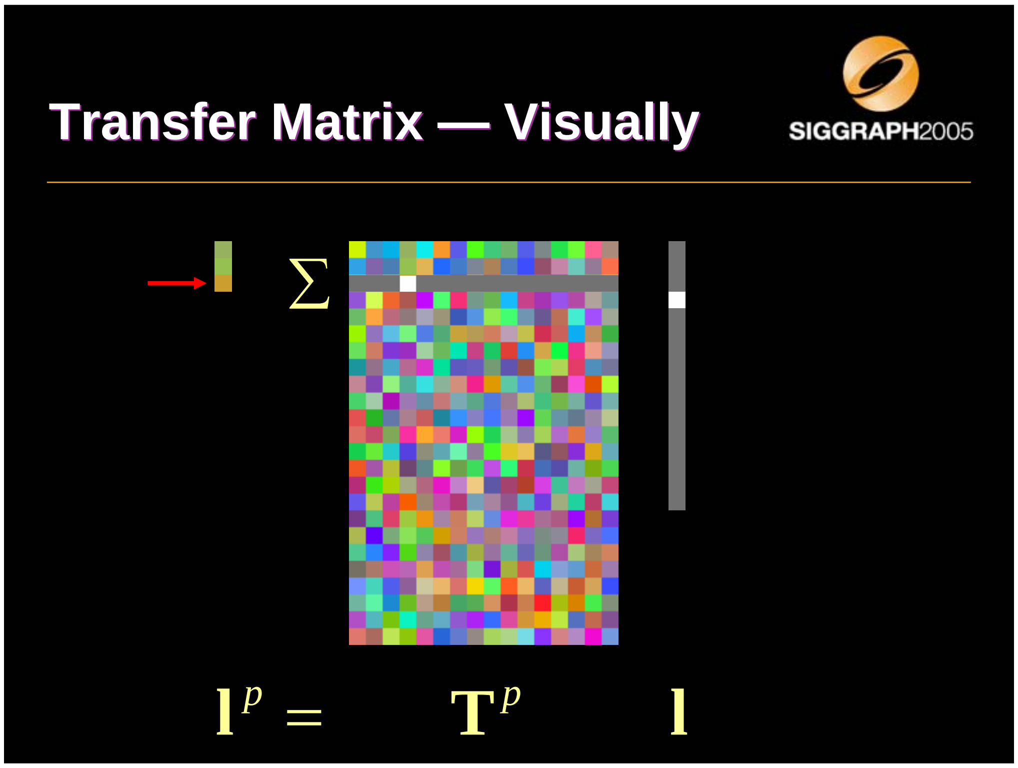


Transfer Matrix — Visually

SIGGRAPH2005

$$\mathbf{l}^p = \sum \mathbf{T}^p \; \mathbf{l}$$

..

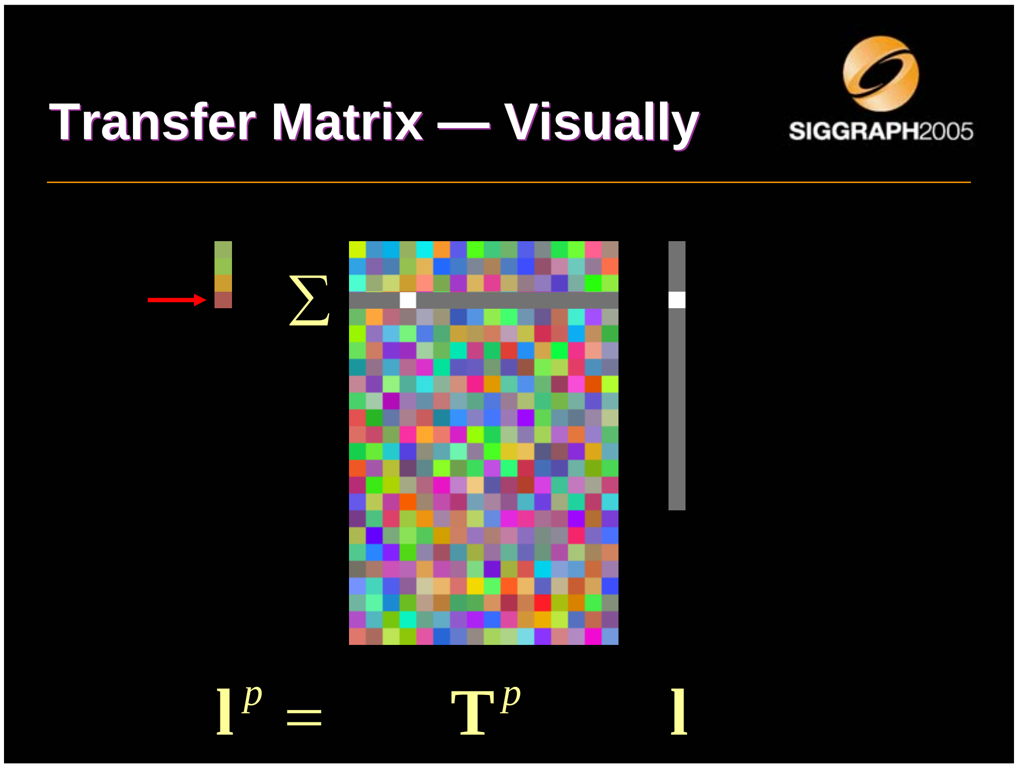


Transfer Matrix — Visually

SIGGRAPH2005

$$\mathbf{l}^p = \sum \mathbf{T}^p \, \mathbf{l}$$

..

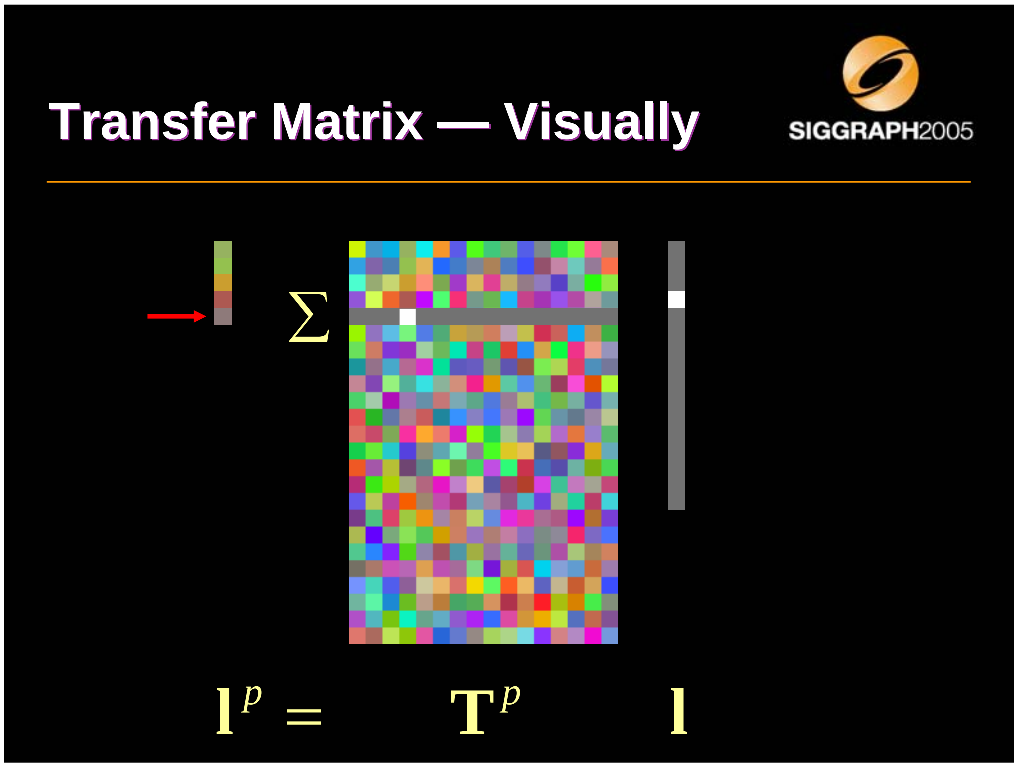


Transfer Matrix — Visually

SIGGRAPH2005

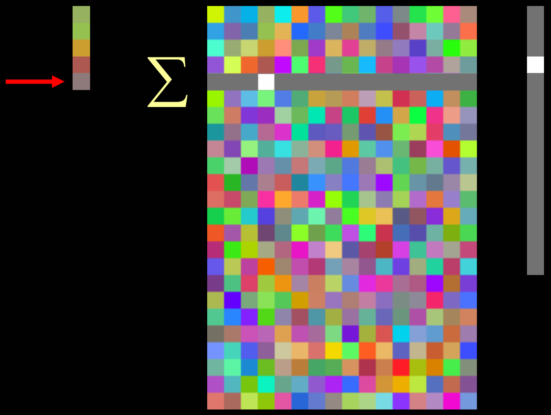
$$\mathbf{l}^p = \sum \mathbf{T}^p \, \mathbf{l}$$

..

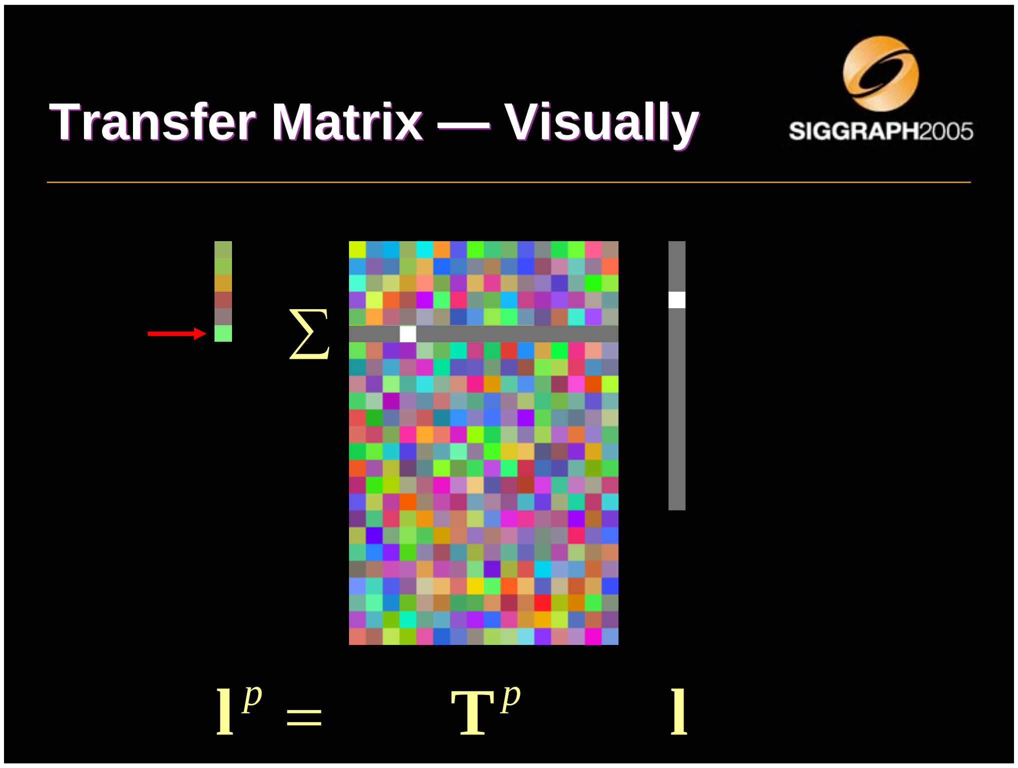


Transfer Matrix — Visually

SIGGRAPH2005

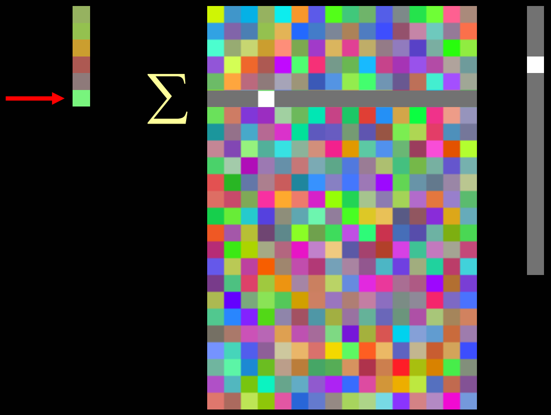


$$\mathbf{l}^p = \mathbf{T}^p \; \mathbf{l}$$

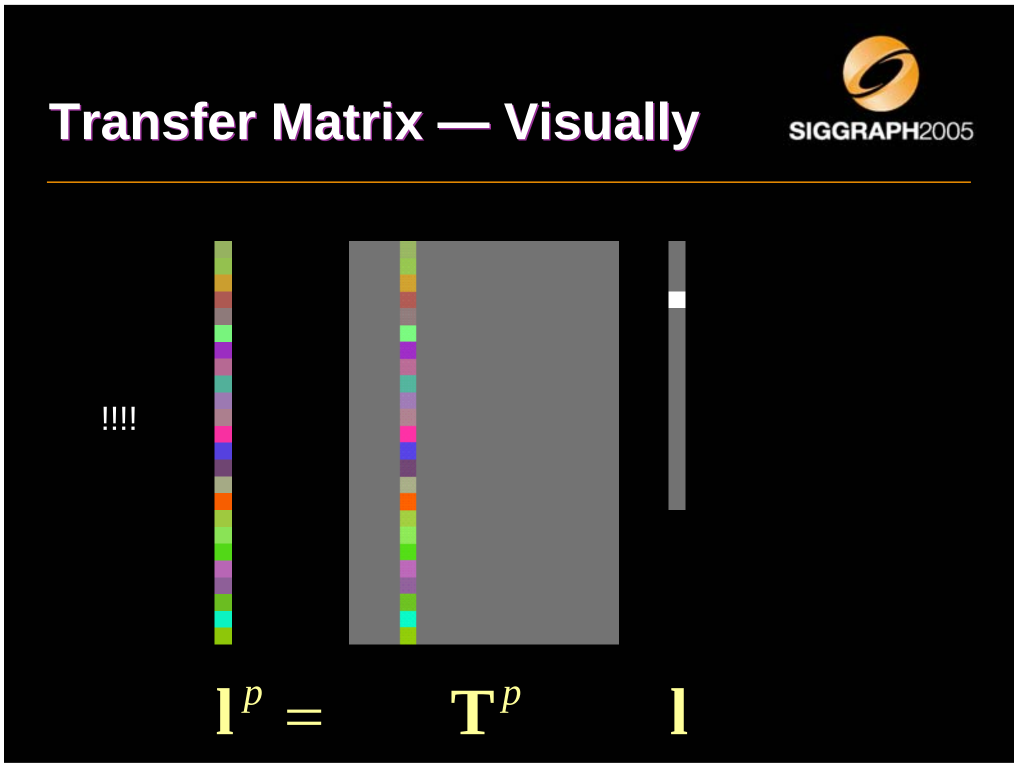


Transfer Matrix — Visually

SIGGRAPH2005



$$\mathbf{l}^p = \sum \mathbf{T}^p \; \mathbf{l}$$



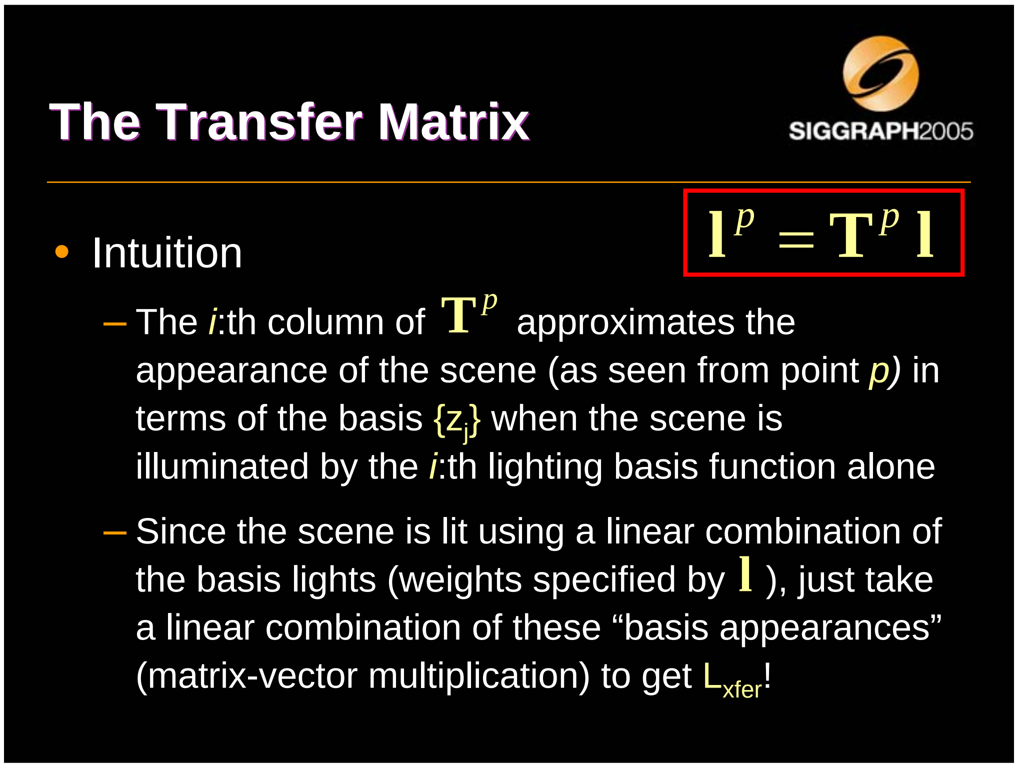
Transfer Matrix — Visually

SIGGRAPH2005

$$\mathbf{l}^{p} = \mathbf{T}^{p} \quad \mathbf{l}$$

In the end, we’ve just copied the fourth column of the matrix over to the result.

Now think back to what the coefficients in **l**^p are supposed to be: The basis coefficients that represent transferred incident radiance at point p in terms of the basis functions {z}!



The Transfer Matrix

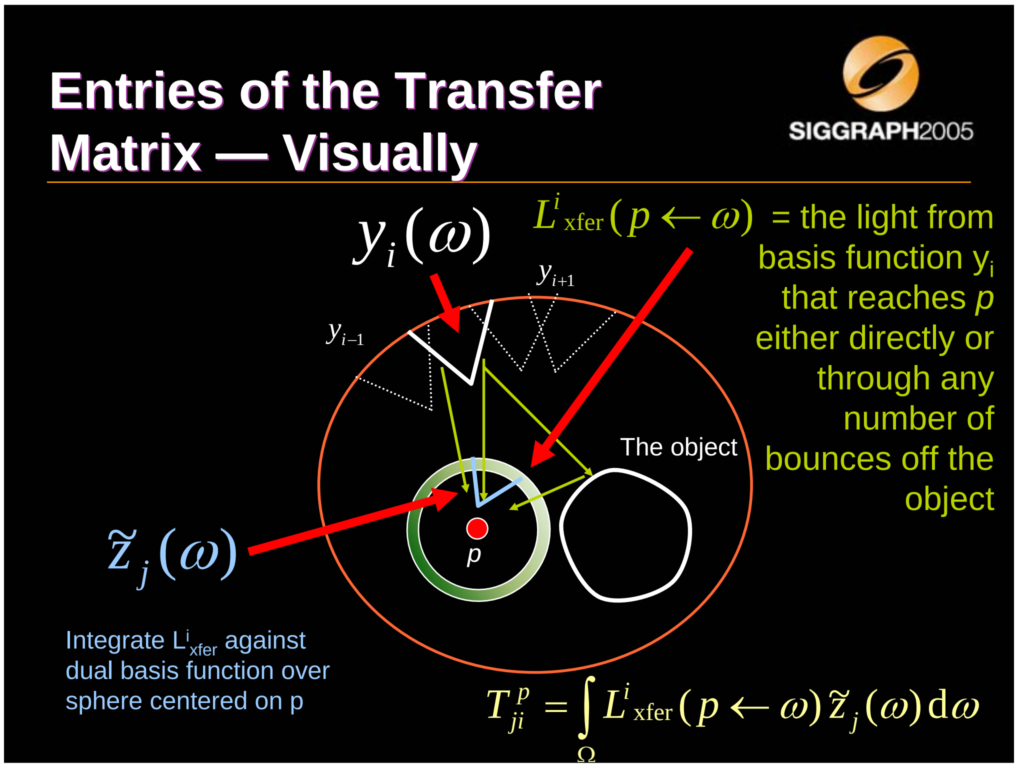
SIGGRAPH2005

$$\boxed{\mathbf{l}^{p} = \mathbf{T}^{p}\,\mathbf{l}}$$

- Intuition
  - The *i*:th column of $\mathbf{T}^{p}$ approximates the appearance of the scene (as seen from point *p)* in terms of the basis $\{z_j\}$ when the scene is illuminated by the *i*:th lighting basis function alone
  - Since the scene is lit using a linear combination of the basis lights (weights specified by $\mathbf{l}$), just take a linear combination of these “basis appearances” (matrix-vector multiplication) to get L_{xfer}!

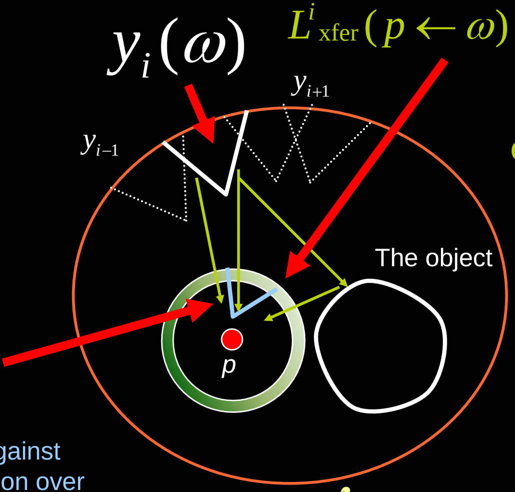
This leads us to conclude that the i:th column of the matrix is a **basis appearance:** it describes the lighting at p when the scene is lit using the i:th lighting basis function alone.

And since the scene is lit by a linear combination of basis lights, getting the transferred incident radiance at p just involves taking a linear combination of these basis appearances with weights taken from the emission vector **l**. That’s just matrix-vector multiplication!



Entries of the Transfer Matrix — Visually

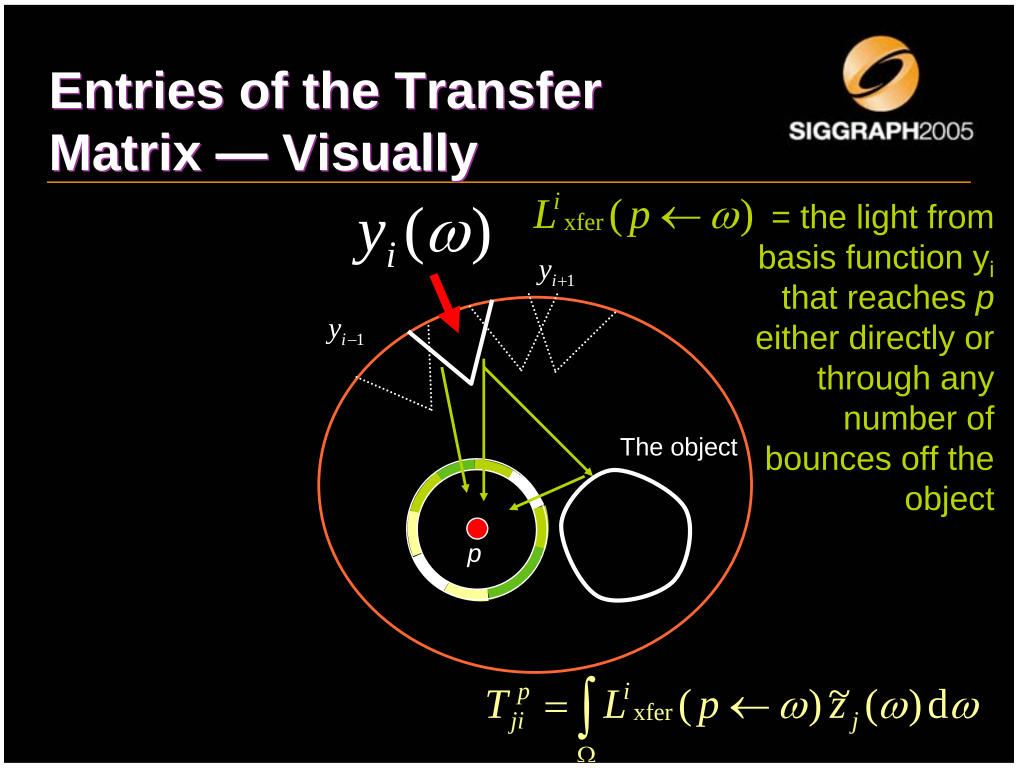
SIGGRAPH2005



$L^i{}_{\text{xfer}}(p \leftarrow \omega)$ = the light from basis function y_i that reaches *p* either directly or through any number of bounces off the object

Integrate $L^i{}_{\text{xfer}}$ against dual basis function over sphere centered on p

$$T^p_{ji} = \int_\Omega L^i{}_{\text{xfer}}(p \leftarrow \omega)\tilde{z}_j(\omega)\,\mathrm{d}\omega$$



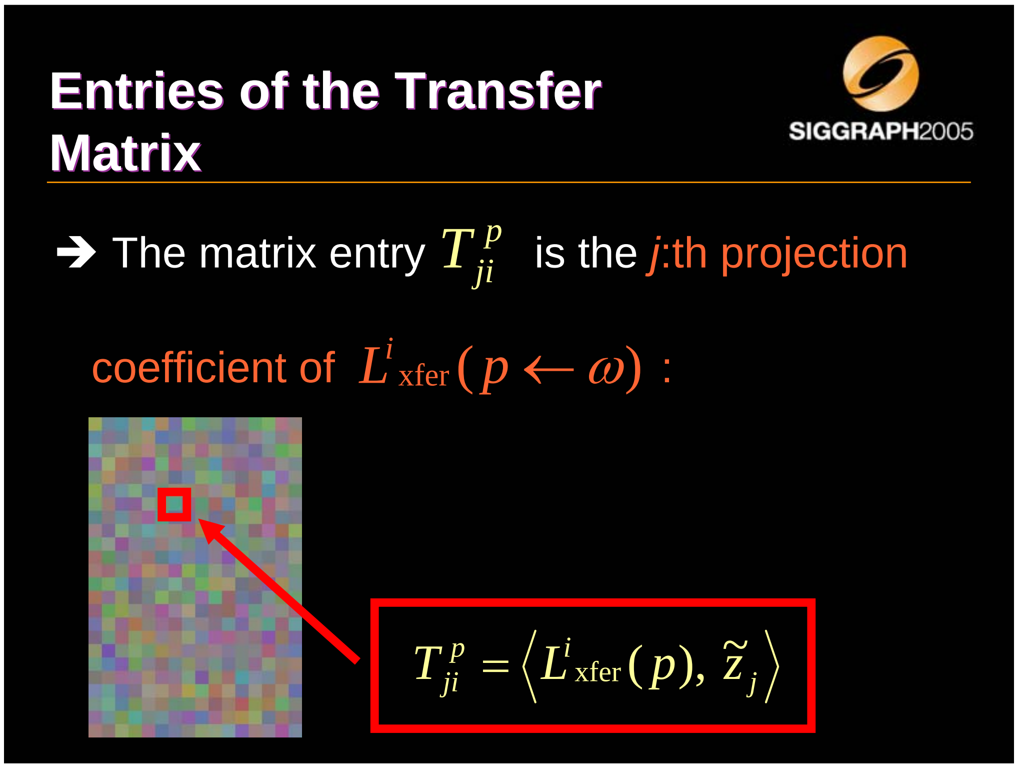
Entries of the Transfer Matrix — Visually

SIGGRAPH2005

$L^i_{\text{xfer}}(p \leftarrow \omega)$ = the light from basis function y_i that reaches *p* either directly or through any number of bounces off the object

$$T^p_{ji} = \int_\Omega L^i_{\text{xfer}}(p \leftarrow \omega)\,\tilde{z}_j(\omega)\,\mathrm{d}\omega$$

The y and z functions are drawn as “hat” functions for illustrational purposes only. Such functions *can* be used for computing transfer, but other basis functions, possibly with global supports like the Spherical Harmonics, are often used.

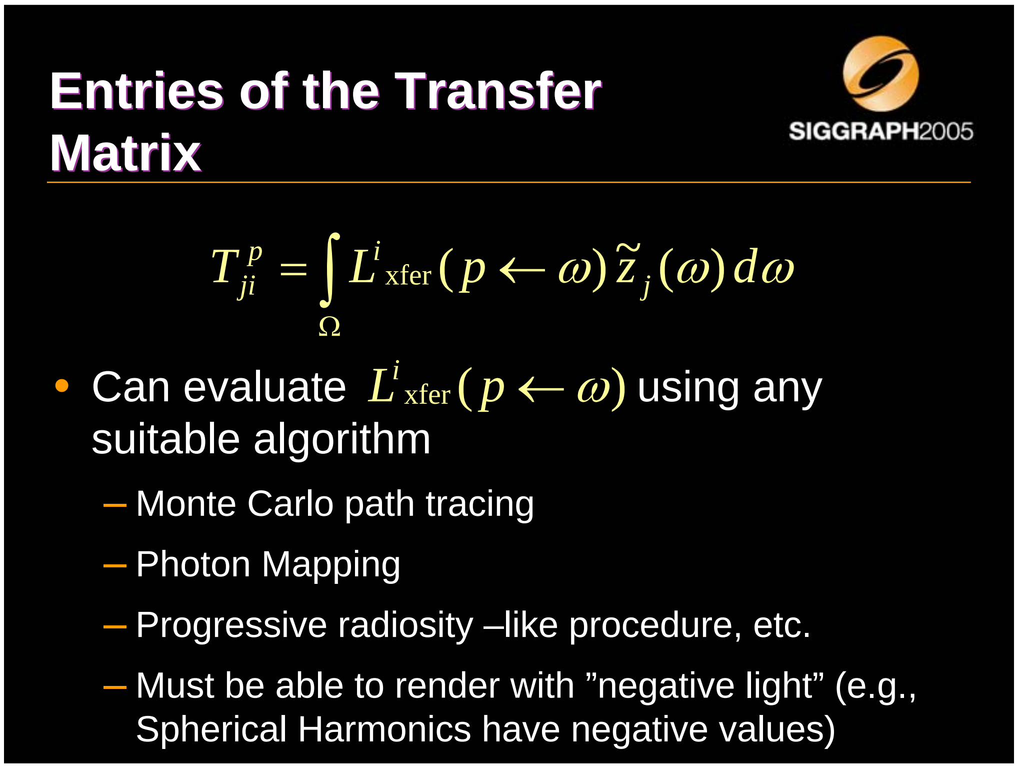


Entries of the Transfer Matrix

SIGGRAPH2005

➔ The matrix entry T_{ji}^{p} is the *j*:th projection coefficient of $L^{i}_{\text{xfer}}(p \leftarrow \omega)$:

$$\boxed{T_{ji}^{p} = \left\langle L^{i}_{\text{xfer}}(p),\, \tilde{z}_j \right\rangle}$$



Entries of the Transfer Matrix

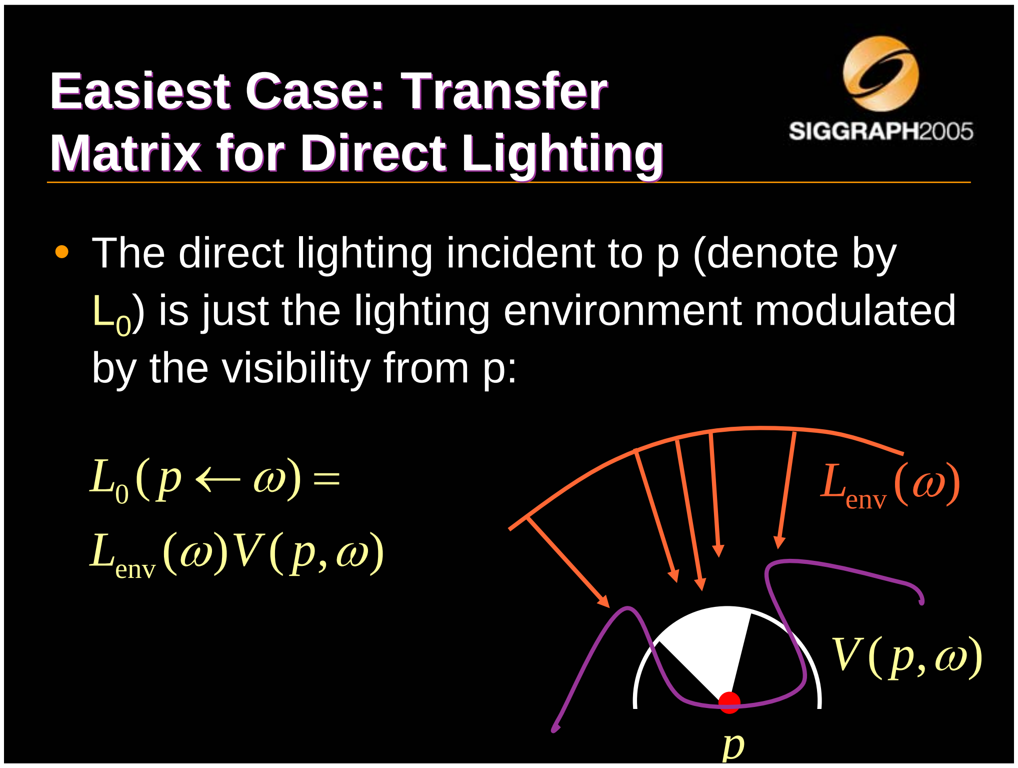
SIGGRAPH2005

$$T^{p}_{ji} = \int_{\Omega} L^{i}_{\text{xfer}}(p \leftarrow \omega)\,\tilde{z}_j(\omega)\,d\omega$$

- Can evaluate $L^{i}_{\text{xfer}}(p \leftarrow \omega)$ using any suitable algorithm
  - Monte Carlo path tracing
  - Photon Mapping
  - Progressive radiosity –like procedure, etc.
  - Must be able to render with ”negative light” (e.g., Spherical Harmonics have negative values)

All you need is to be able to compute the radiance that is incident to the point p when the scene is lit using the i:th light basis function alone! NOTE that the lighting basis functions can be negative, so the renderer must be able to handle “negative light” as well.

Next we’ll consider the simple case of direct illumination only; that is, we consider only the light incident to the object that has been shadowed by the object, but has not bounced on the surface before impinging upon p. This simple case will show that this perhaps abstract-looking computation can really be simple in some cases.

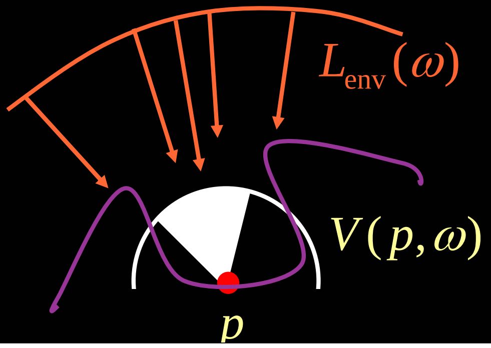


Easiest Case: Transfer Matrix for Direct Lighting

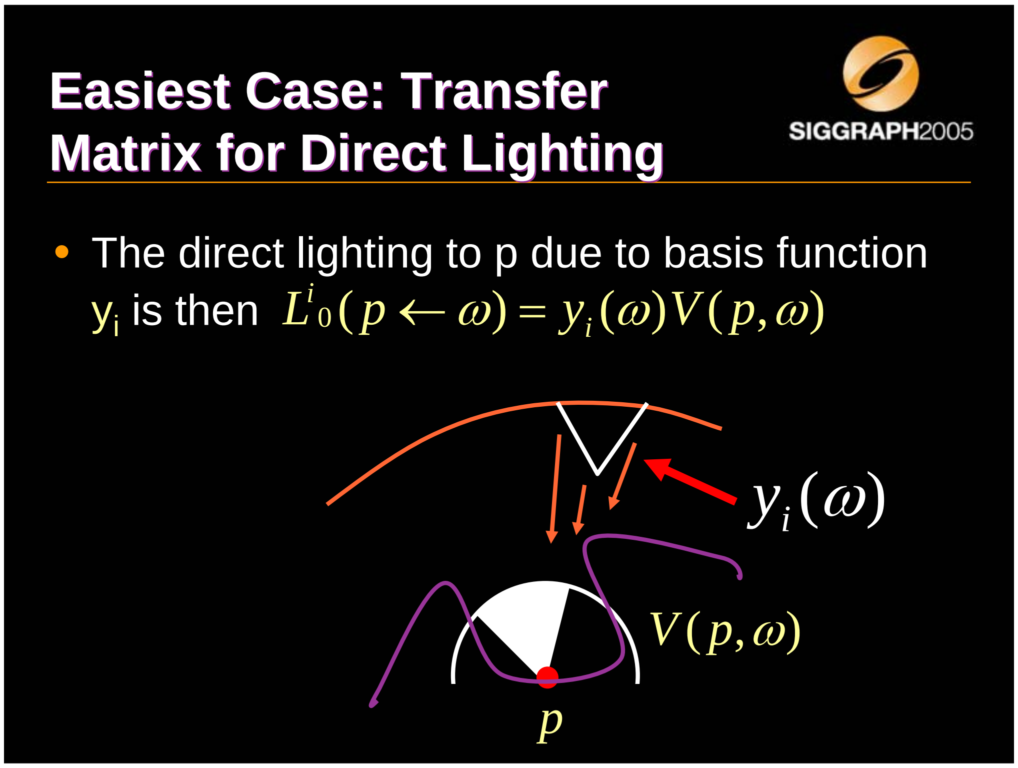
SIGGRAPH2005

- The direct lighting incident to p (denote by L_0) is just the lighting environment modulated by the visibility from p:

$$L_0(p \leftarrow \omega) = L_{\text{env}}(\omega) V(p, \omega)$$



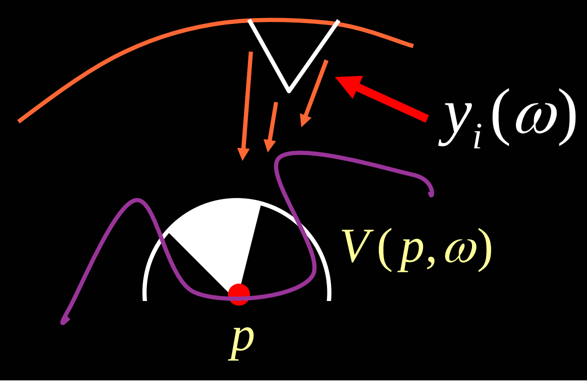
The first term, the direct lighting term L_0, is just our lighting environment L_env times the visibility function V(p,w) that encodes whether or not the object blocks the sightline from p towards w: If p can see the environment at direction w, the lighting is taken from the environment; otherwise there is no direct light from that direction, i.e., L_0(p ← w) = 0 for that direction.



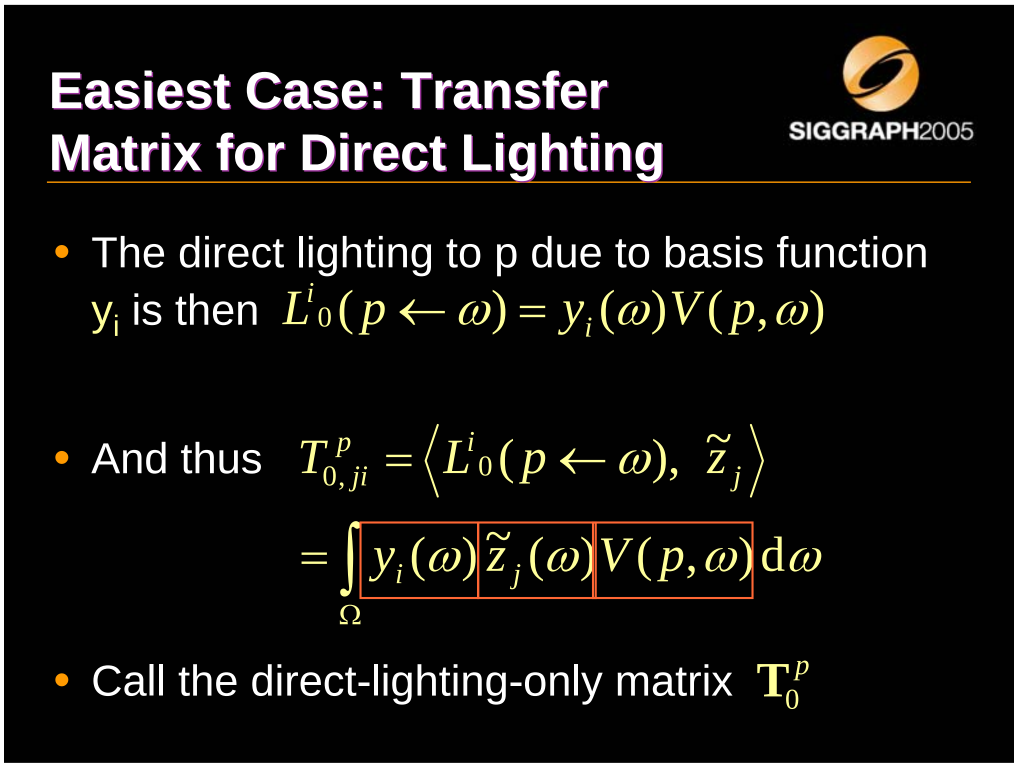
Easiest Case: Transfer Matrix for Direct Lighting

SIGGRAPH2005

- The direct lighting to p due to basis function y_i is then $L^i_0(p \leftarrow \omega) = y_i(\omega) V(p,\omega)$



The matrix entry T_ji is the integral of the visibility function times the i:th lighting basis function times the j:th transferred incident radiance coefficient!



Easiest Case: Transfer Matrix for Direct Lighting

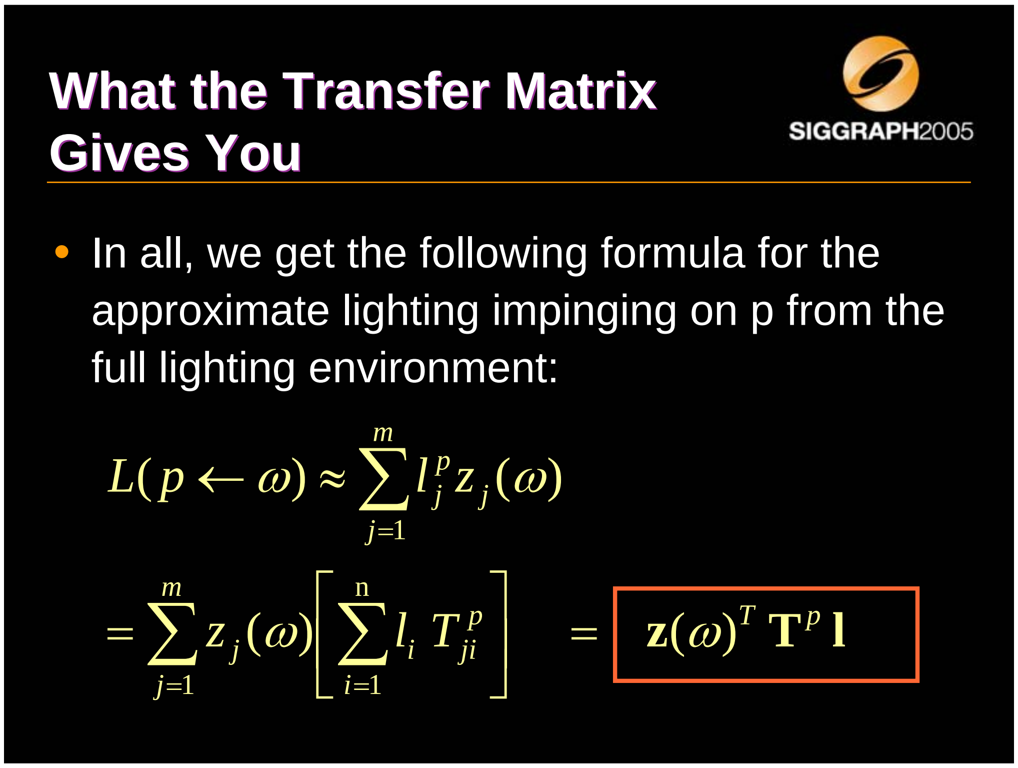
SIGGRAPH2005

- The direct lighting to p due to basis function y_i is then $L^i{}_0(p \leftarrow \omega) = y_i(\omega) V(p,\omega)$
- And thus $T^p_{0,ji} = \left\langle L^i{}_0(p \leftarrow \omega),\ \tilde{z}_j \right\rangle$

$$= \int_\Omega \boxed{y_i(\omega)}\boxed{\tilde{z}_j(\omega)}\boxed{V(p,\omega)}\,\mathrm{d}\omega$$

- Call the direct-lighting-only matrix $\mathbf{T}^p_0$

The matrix entry T_ji is the integral of the visibility function times the i:th lighting basis function times the j:th transferred incident radiance coefficient!



What the Transfer Matrix Gives You

SIGGRAPH2005

- In all, we get the following formula for the approximate lighting impinging on p from the full lighting environment:

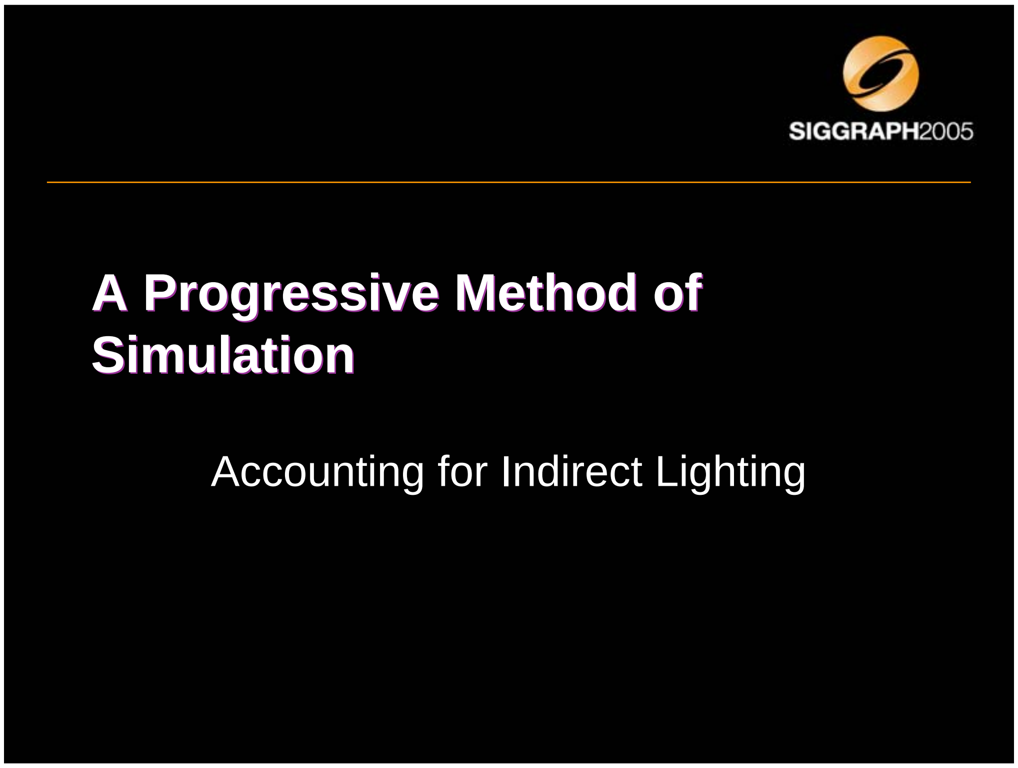
$$L(p \leftarrow \omega) \approx \sum_{j=1}^{m} l_j^p z_j(\omega)$$

$$= \sum_{j=1}^{m} z_j(\omega) \left[\sum_{i=1}^{n} l_i \, T_{ji}^p \right] \quad = \boxed{\mathbf{z}(\omega)^T \, \mathbf{T}^p \, \mathbf{l}}$$

If we substitute the previous into the equation that gives the approximated transferred incident radiance from the its basis coefficients, we end up with the following.

The sum in the middle can be written in matrix form as given on the second line, where we have stacked the values of all the basis functions z_j, evaluated in direction omega, together to form the vector **z**.

The transfer matrices T0 map the incident illumination (expressed in basis {y} by vector **l**) into direct, shadowed incident radiance at the point p, expressed in basis {z}.

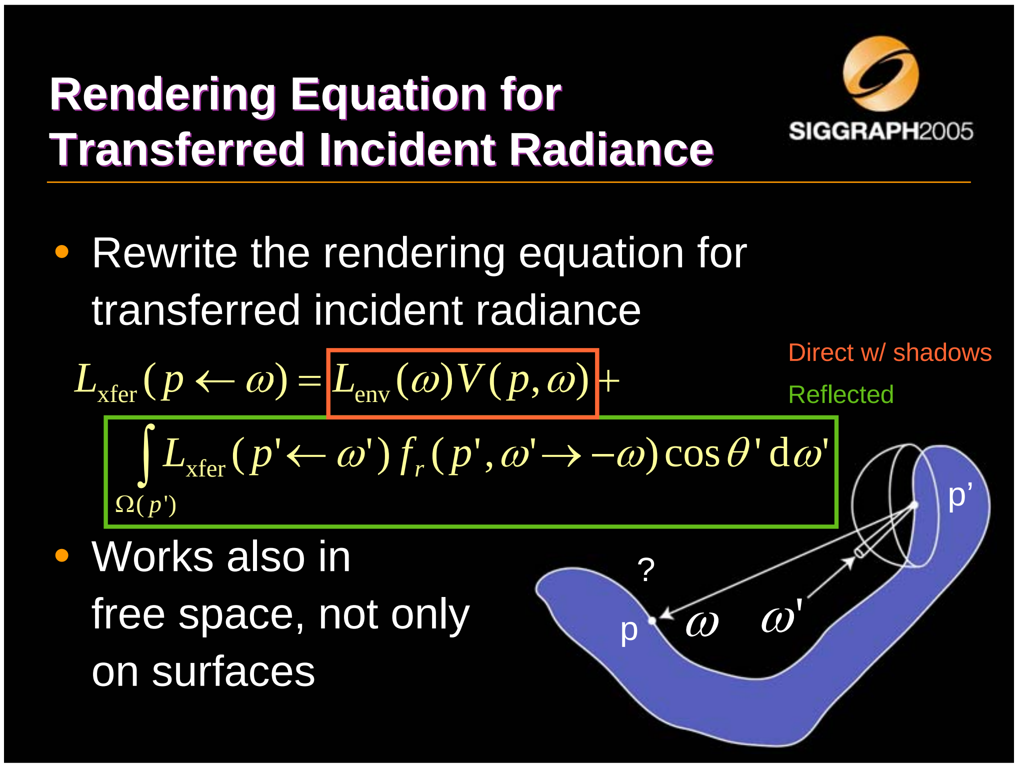


A Progressive Method of Simulation

Accounting for Indirect Lighting

The next section derives a particular method for computing the transfer matrices that account for indirect lighting as well. It resembles a progressive gathering radiosity method, and bears a close resemblance to an early non-diffuse global illumination method of Sillion and others from Siggraph '91.

In what follows, we'll derive a recursive formula that computes the transfer matrices that correspond to k+1 bounces of light, given that we know the transfer matrices that account for k bounces for light everywhere in the scene. As we already have a method for computing T_0, the transfer matrix for direct lighting, we are able to account for an arbitrary number of bounces this way. And as lighting is linear, the transfer matrix T^p that accounts for all bounces is just the sum of the matrices T_0^p, T_1^p, … that each account for a particular number of bounces.



Rendering Equation for Transferred Incident Radiance

SIGGRAPH2005

- Rewrite the rendering equation for transferred incident radiance

Direct w/ shadows

Reflected

$$L_{\text{xfer}}(p \leftarrow \omega) = \boxed{L_{\text{env}}(\omega) V(p,\omega)} + \boxed{\int_{\Omega(p')} L_{\text{xfer}}(p' \leftarrow \omega') f_r(p', \omega' \to -\omega) \cos\theta' \, d\omega'}$$

- Works also in free space, not only on surfaces

p'
?
p
ω
ω'

In order to derive the method, we'll use a "transposed" version of the rendering equation; one that we have written specifically using transferred incident radiance as the unknown, not outgoing radiance as is usually done. These equations are equivalent; if we know the solution to the other, we get the other pretty easily from it.

What the equation says: The lighting incident to p is the sum of

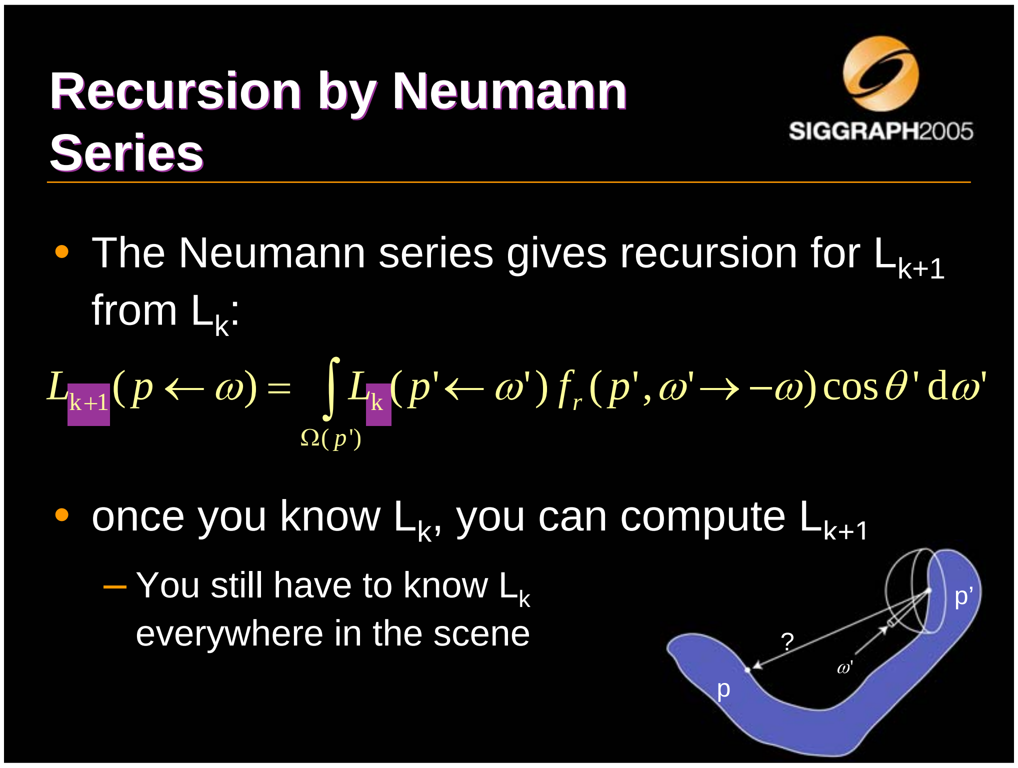
•direct lighting from the lighting environment, shadowed by the object

•Light reflected by the scene towards p.

Note that the first term is only non-zero if the ray from p towards omega doesn't intersect the scene anywhere, and the second term can only be non-zero if the ray does intersect the scene.

P' is the point that the ray from p towards omega intersects. Note that the usual reflectance integral is performed above the point p' here, not p as in the usual version of the rendering equation.

This works just as well for points p not on the surfaces, but in free space.

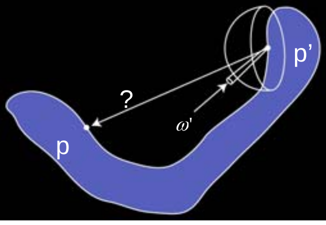


Recursion by Neumann Series

- The Neumann series gives recursion for L_{k+1} from L_k:

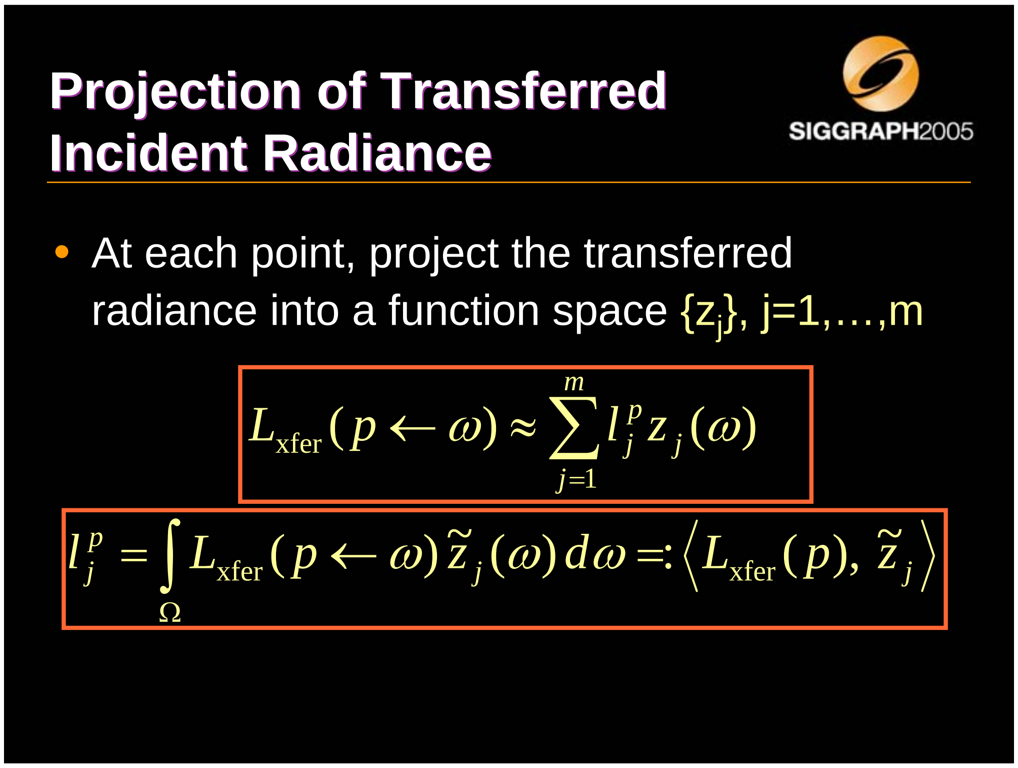
$$L_{k+1}(p \leftarrow \omega) = \int_{\Omega(p')} L_k(p' \leftarrow \omega')\, f_r(p', \omega' \rightarrow -\omega) \cos\theta' \, \mathrm{d}\omega'$$

- once you know L_k, you can compute L_{k+1}
  - You still have to know L_k everywhere in the scene



The incident version of the rendering equation can be solved using Neumann series just as well as the outgoing version: The lighting upon p is the sum of direct light, light that has taken one, two, three, etc. bounces.

Given that we know the lighting in the scene that has taken k bounces: Then the Neumann series gives the relationship of this known k-times reflected light and k+1 –times reflected light, i.e., if we know the incident radiance from the previous bounce, the next one is obtained from the reflectance equation.



Projection of Transferred Incident Radiance

SIGGRAPH2005

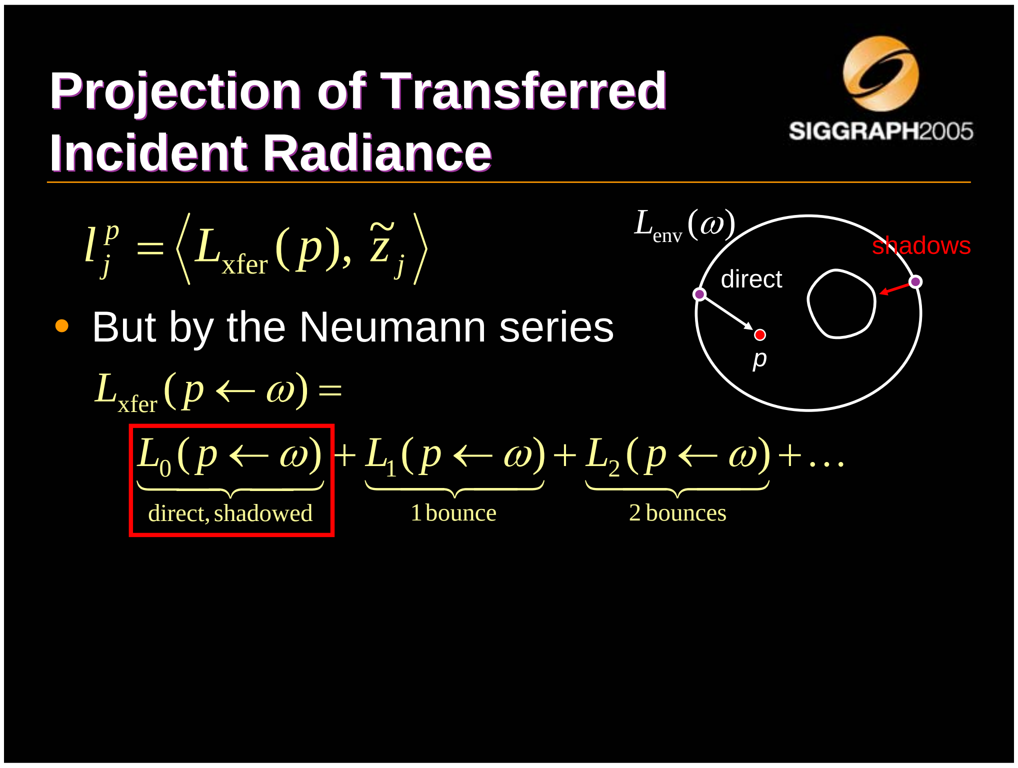
- At each point, project the transferred radiance into a function space $\{z_j\}$, j=1,…,m

$$\boxed{L_{\text{xfer}}(p \leftarrow \omega) \approx \sum_{j=1}^{m} l_j^p z_j(\omega)}$$

$$\boxed{l_j^p = \int_{\Omega} L_{\text{xfer}}(p \leftarrow \omega)\tilde{z}_j(\omega)\,d\omega =: \left\langle L_{\text{xfer}}(p), \tilde{z}_j \right\rangle}$$

To derive transfer matrices for all bounces, we’ll start from the projection equation: We want to express L_xfer at p in terms of the basis functions z. In the end this will yield the transfer matrices for all bounces.

As we saw earlier: In order to get the j:th projection coefficients, we have to integrate the function against the dual basis function z^tilde.



Projection of Transferred Incident Radiance

$$l_j^p = \left\langle L_{\text{xfer}}(p),\ \tilde{z}_j \right\rangle$$

- But by the Neumann series

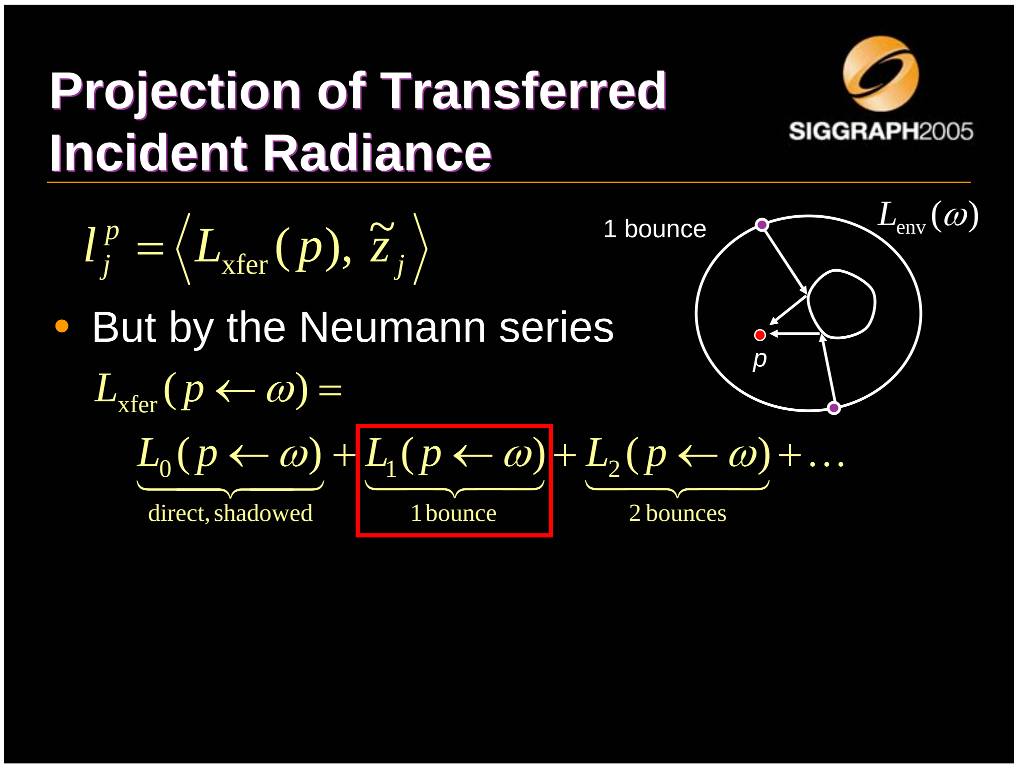
$$L_{\text{xfer}}(p \leftarrow \omega) =$$

$$\boxed{\underbrace{L_0(p \leftarrow \omega)}_{\text{direct, shadowed}}} + \underbrace{L_1(p \leftarrow \omega)}_{\text{1 bounce}} + \underbrace{L_2(p \leftarrow \omega)}_{\text{2 bounces}} + \ldots$$

$L_{\text{env}}(\omega)$
shadows
direct
p
SIGGRAPH2005

As we saw earlier, the rendering equation may be solved by the Neumann series as the sum of direct lighting, light bounced off the object once, twice, etc.

That is, the radiance incident to p is the sum of radiance directly from the lighting environment L that is shadowed by the object…



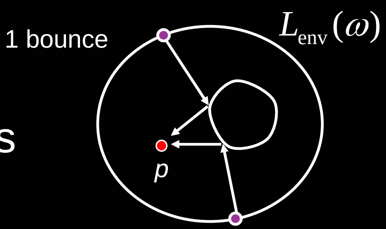
Projection of Transferred Incident Radiance

$$l_j^p = \langle L_{\text{xfer}}(p), \tilde{z}_j \rangle$$

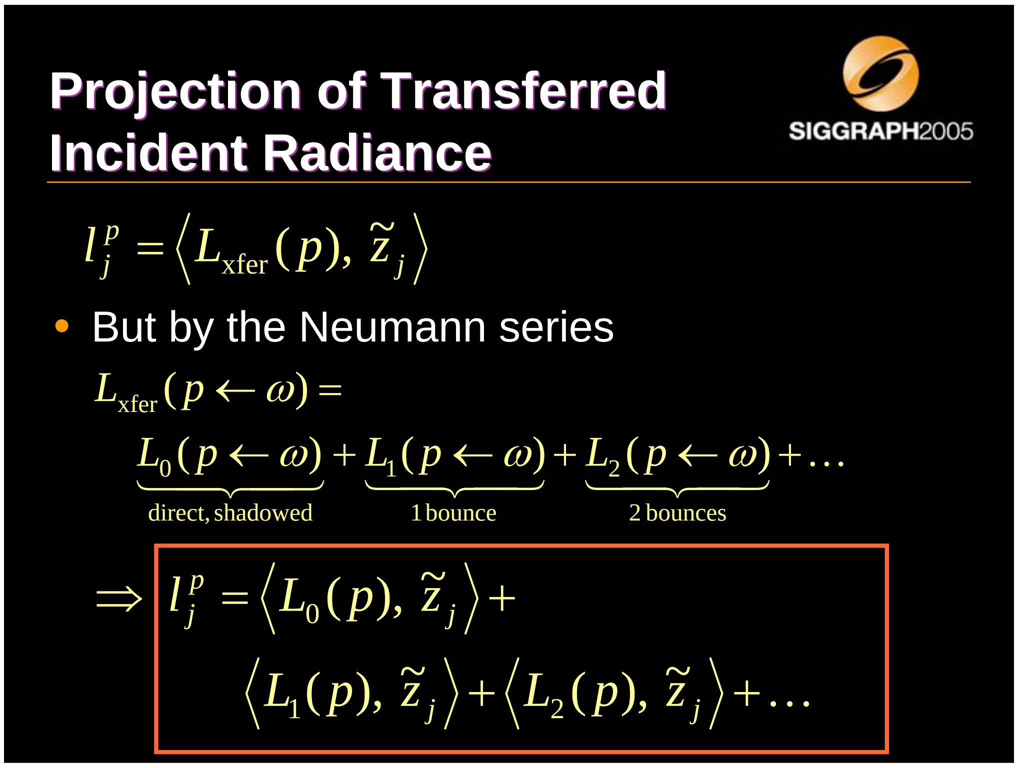
- But by the Neumann series

$$L_{\text{xfer}}(p \leftarrow \omega) =$$

$$\underbrace{L_0(p \leftarrow \omega)}_{\text{direct, shadowed}} + \boxed{\underbrace{L_1(p \leftarrow \omega)}_{\text{1 bounce}}} + \underbrace{L_2(p \leftarrow \omega)}_{\text{2 bounces}} + \ldots$$



…the light that reaches p through one bounce off the object…



Projection of Transferred Incident Radiance

$$l_j^p = \left\langle L_{\text{xfer}}(p),\ \tilde{z}_j \right\rangle$$

- But by the Neumann series

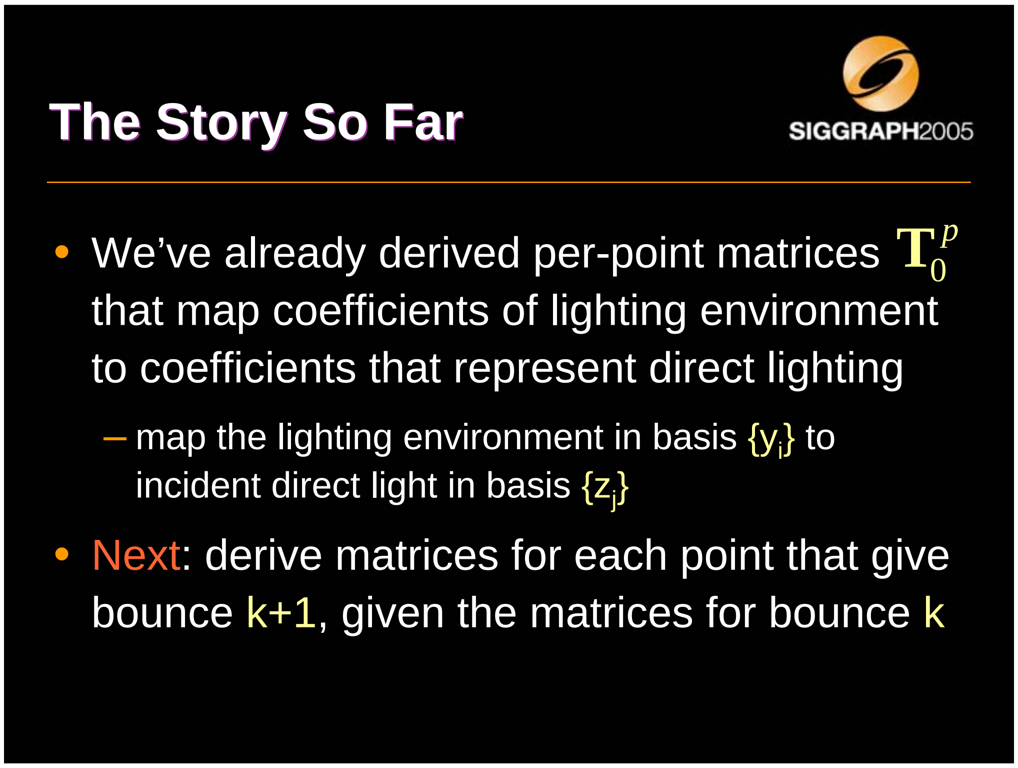
$$L_{\text{xfer}}(p \leftarrow \omega) =$$

$$\underbrace{L_0(p \leftarrow \omega)}_{\text{direct, shadowed}} + \underbrace{L_1(p \leftarrow \omega)}_{\text{1 bounce}} + \underbrace{L_2(p \leftarrow \omega)}_{\text{2 bounces}} + \ldots$$

$$\Rightarrow \boxed{\begin{aligned} l_j^p &= \left\langle L_0(p),\ \tilde{z}_j \right\rangle + \\ &\left\langle L_1(p),\ \tilde{z}_j \right\rangle + \left\langle L_2(p),\ \tilde{z}_j \right\rangle + \ldots \end{aligned}}$$

…two bounces, and so on.

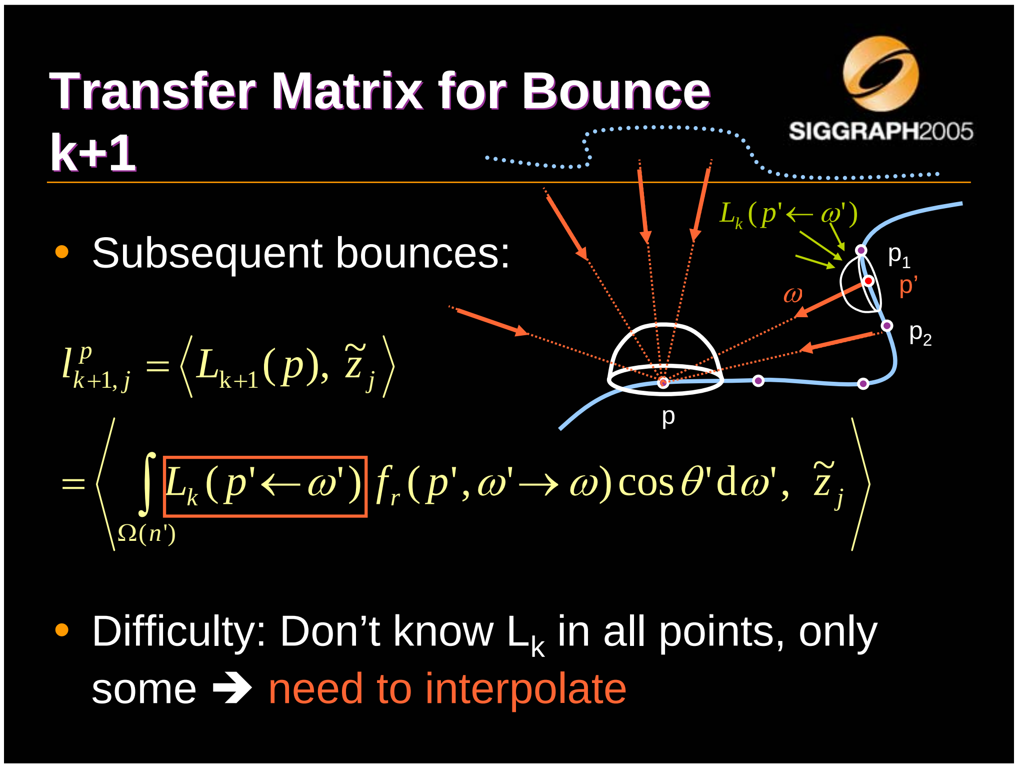
Since the inner product is linear, the projection coefficients can be computed by projecting each bounce of light separately and adding the results together.



The Story So Far

SIGGRAPH2005

- We've already derived per-point matrices $\mathbf{T}_0^p$ that map coefficients of lighting environment to coefficients that represent direct lighting
  - map the lighting environment in basis $\{y_i\}$ to incident direct light in basis $\{z_j\}$
- Next: derive matrices for each point that give bounce k+1, given the matrices for bounce k


SIGGRAPH2005
$L_k(p'\leftarrow\omega')$
p_1
p'
ω
p_2
p

Transfer Matrix for Bounce k+1

- Subsequent bounces:

$$l^p_{k+1,j} = \langle L_{k+1}(p), \tilde{z}_j \rangle$$

$$= \left\langle \int_{\Omega(n')} \boxed{L_k(p'\leftarrow\omega')}\, f_r(p',\omega'\rightarrow\omega)\cos\theta' \mathrm{d}\omega',\ \tilde{z}_j \right\rangle$$

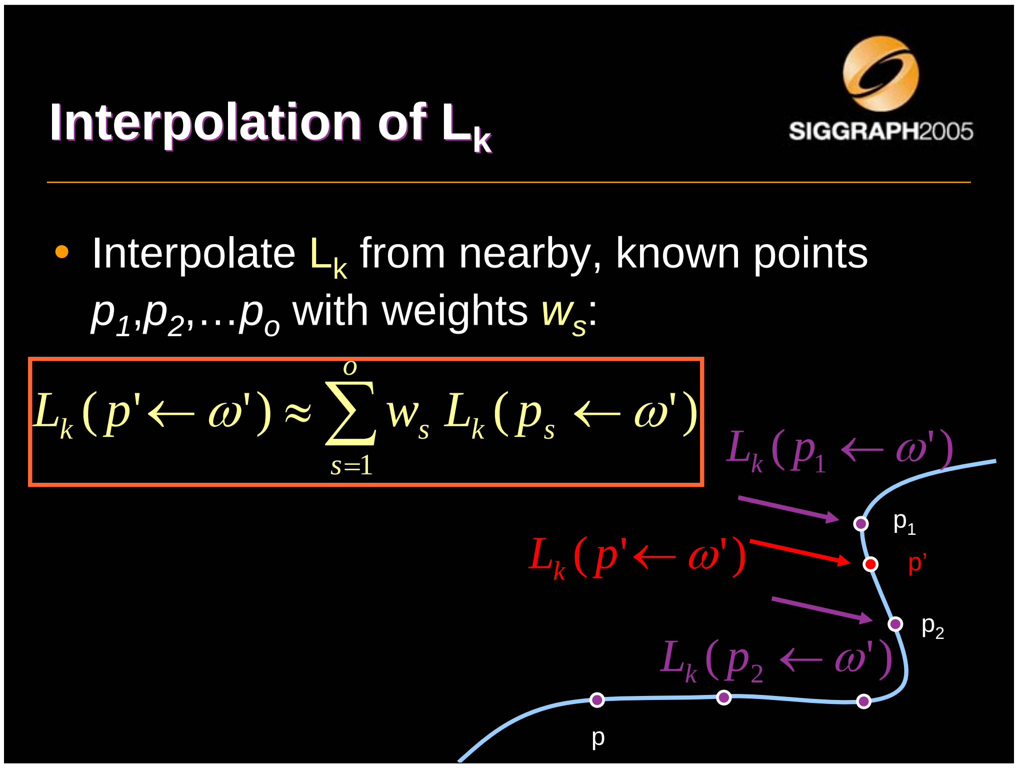
- Difficulty: Don't know L_k in all points, only some ➔ need to interpolate

We'll start the derivation of the matrices T_^p{k+1} from the projection coefficients l^p_k.

The projection of the bounce k+1 at point p is the directional inner product (at p) of the radiance reflected towards p from the previous bounce k and the dual basis functions z_j^tilde. That is, for a large number of directions around p, we have to evaluate the k+1 –bounce radiance reflected towards p from each direction. That reflected radiance is determined by reflecting the previous bounce at the points where the rays from p hit by performing the usual reflectance integration at the points where our rays hit. **NOTE** that if a ray does not hit the object, there is no light of bounce k+1 coming from that direction!

We will present efficient methods for evaluating the reflectance integral later when we talk about producing outgoing radiance from transferred incident radiance.

Notice that the rays from p may hit the surface of the object anywhere, not just at the points where we know the previous bounce k. We know an approximation to transferred incident radiance from the previous bounce at the finite set of points p, not in all points in the scene ➔ to get L_k for a point p' that falls between the samples, interpolate linearly from the nearest known points.



Interpolation of L_k

SIGGRAPH2005

- Interpolate L_k from nearby, known points $p_1, p_2, \ldots p_o$ with weights w_s:

$$\boxed{L_k(p' \leftarrow \omega') \approx \sum_{s=1}^{o} w_s \, L_k(p_s \leftarrow \omega')}$$

$L_k(p_1 \leftarrow \omega')$
$L_k(p' \leftarrow \omega')$
$L_k(p_2 \leftarrow \omega')$
p_1
p'
p_2
p

Interpolate transferred incident radiance from nearest known points.

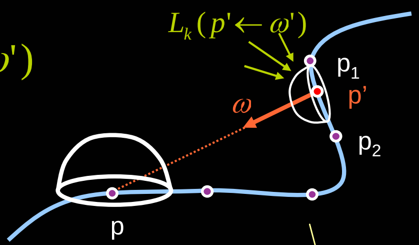
The w_s are interpolation weights (positive, sum to one).

In practice we know L_k at the vertices of our model (or texels, if we sample transfer using textures). This means we can easily interpolate using barycentric coordinates (from vertices) or bilinearly (from textures).

Transfer Matrix for Bounce k+1

SIGGRAPH2005

$$L_k(p'\leftarrow\omega') \approx \sum_{s=1}^{o} w_s\, L_k(p_s\leftarrow\omega')$$



$$\Rightarrow l^{p}_{k+1,j} = \left\langle L_{k+1}(p),\ \tilde{z}_j \right\rangle$$

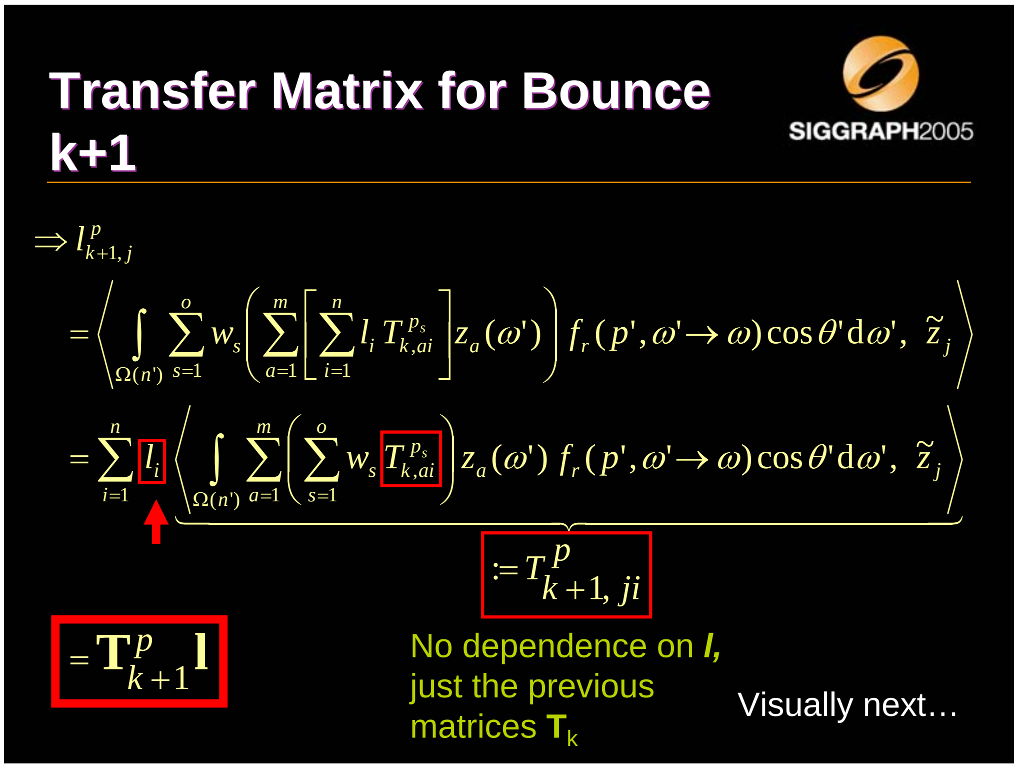
$$= \left\langle \int_{\Omega(n')} \left(\sum_{s=1}^{o} w_s\, L_k(p_s\leftarrow\omega') \right) f_r(p',\omega'\to\omega)\cos\theta'\,\mathrm{d}\omega',\ \tilde{z}_j \right\rangle$$

$$= \left\langle \int_{\Omega(n')} \sum_{s=1}^{o} w_s \underbrace{\left(\sum_{a=1}^{m} l^{p_s}_{k,a}\, z_a(\omega') \right)}_{L_k(p_s\leftarrow\omega') \approx \sum_{a=1}^{m} l^{p_s}_{k,a}\, z_a(\omega')} f_r(p',\omega'\to\omega)\cos\theta'\,\mathrm{d}\omega',\ \tilde{z}_j \right\rangle$$

Now we have the tools to compute the next bounce.

When evaluating the radiance reflected from the last bounce by point p’ towards p, substitute the interpolated transferred incident radiance into the reflectance integral.

The transferred incident radiance through k bounces we know; we can compute its coefficients using the matrices for k bounces T_k.



Transfer Matrix for Bounce k+1

SIGGRAPH2005

$$\Rightarrow l^{p}_{k+1,j}$$

$$= \left\langle \int_{\Omega(n')} \sum_{s=1}^{o} w_s \left(\sum_{a=1}^{m} \left[\sum_{i=1}^{n} l_i \, T^{p_s}_{k,ai} \right] z_a(\omega') \right) f_r(p',\omega' \to \omega) \cos\theta' \mathrm{d}\omega',\ \tilde{z}_j \right\rangle$$

$$= \sum_{i=1}^{n} \boxed{l_i} \underbrace{\left\langle \int_{\Omega(n')} \sum_{a=1}^{m} \left(\sum_{s=1}^{o} w_s \boxed{T^{p_s}_{k,ai}} \right) z_a(\omega') \, f_r(p',\omega' \to \omega) \cos\theta' \mathrm{d}\omega',\ \tilde{z}_j \right\rangle}_{\boxed{:= T^{p}_{k+1,\, ji}}}$$

$$\boxed{= \mathbf{T}^{p}_{k+1} \mathbf{l}}$$

No dependence on ***l*,**
just the previous
matrices $\mathbf{T}_k$

Visually next…

At the points from which we interpolate, transferred incident radiance of the previous bounce is linear in the incident coefficients (induction hypothesis – we've proved this for direct lighting, i.e., k=0, already).

On the top row we substitute this linear relationship to the previous equation,

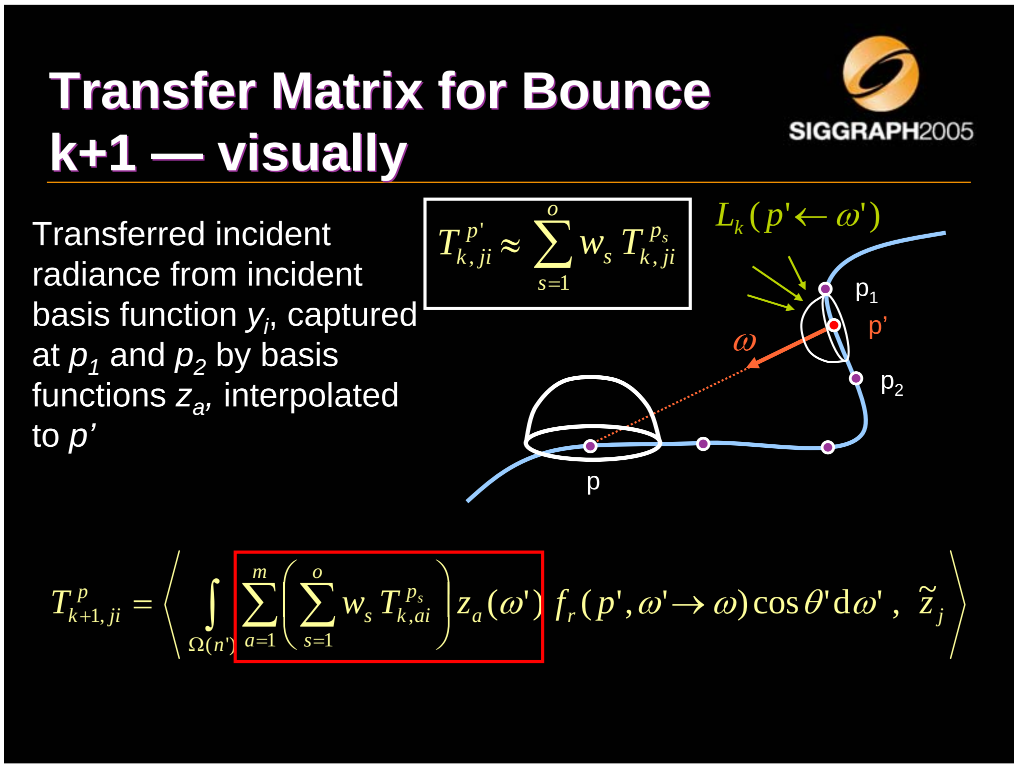
Then move the sum out of the inner product by linearity,

And notice that what's left inside the inner product is just integrals of basis functions, the BRDF and the entries of transfer matrices for the previous bounce, and that there are again two free indices,

and that this is again just a matrix-vector multiplication with the incident lighting coefficients **l**. We define the stuff inside the the inner product as T_{k+1}, the transfer matrix for bounce k+1.

!!! You should notice that the coefficients **l** for the lighting environment DO NOT affect the transfer matrix T_k+1 – it is computed purely from suitable integrals and interpolations of the transfer matrices T_k.

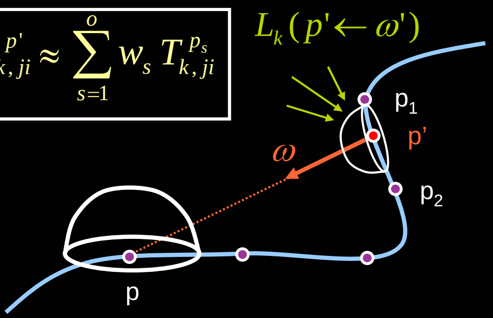
This is just to convince you that it works, a more graphical explanation is coming up next…



Transfer Matrix for Bounce k+1 — visually

Transferred incident radiance from incident basis function y_i, captured at p_1 and p_2 by basis functions z_a, interpolated to p'

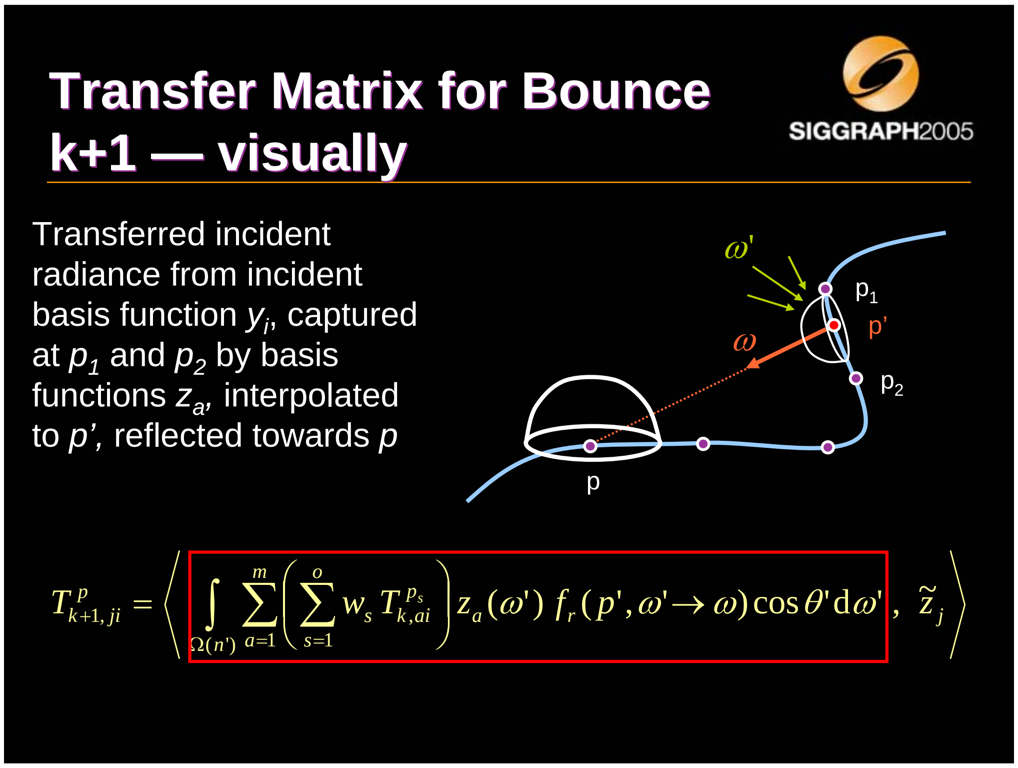
$$T_{k,ji}^{p'} \approx \sum_{s=1}^{o} w_s \, T_{k,ji}^{p_s}$$



$$T_{k+1,ji}^{p} = \left\langle \int_{\Omega(n')} \boxed{\sum_{a=1}^{m}\left(\sum_{s=1}^{o} w_s \, T_{k,ai}^{p_s}\right) z_a(\omega')} \, f_r(p',\omega' \rightarrow \omega)\cos\theta' \, \mathrm{d}\omega' \, , \, \tilde{z}_j \right\rangle$$

When evaluating the projection of transferred incident radiance of bounce k+1 from lighting basis function y_i to point p, we have to compute the radiance reflected towards p from the previous bounce. To do this, we have to evaluate a directional inner product for many directions omega around p.

For a single direction omega, the boxed formula is just the transferred incident radiance from y_i after k bounces at the point p’ where the ray from p towards omega hits the surface of the object, interpolated in the fashion we just described.

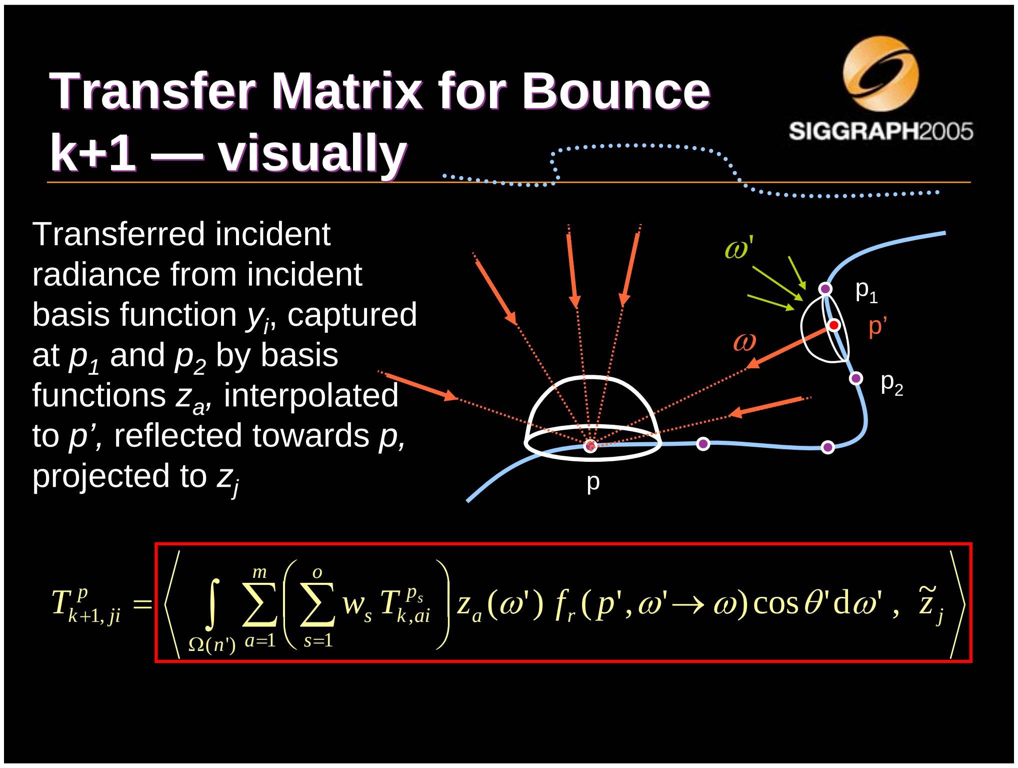


Transfer Matrix for Bounce k+1 — visually

Transferred incident radiance from incident basis function y_i, captured at p_1 and p_2 by basis functions z_a, interpolated to p', reflected towards p

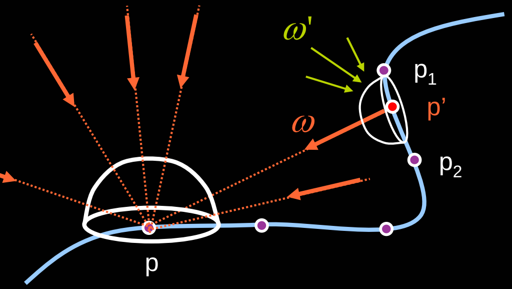
$$T^{p}_{k+1,ji} = \left\langle \boxed{\int_{\Omega(n')} \sum_{a=1}^{m} \left(\sum_{s=1}^{o} w_s T^{p_s}_{k,ai} \right) z_a(\omega')\, f_r(p',\omega' \to \omega) \cos\theta' \, \mathrm{d}\omega'}, \ \tilde{z}_j \right\rangle$$

The boxed formula is now the radiance reflected by point p’ from the previous bounce towards p; this is just the usual reflectance equation.



Transfer Matrix for Bounce k+1 — visually

Transferred incident radiance from incident basis function y_i, captured at p_1 and p_2 by basis functions z_a, interpolated to *p'*, reflected towards *p*, projected to z_j

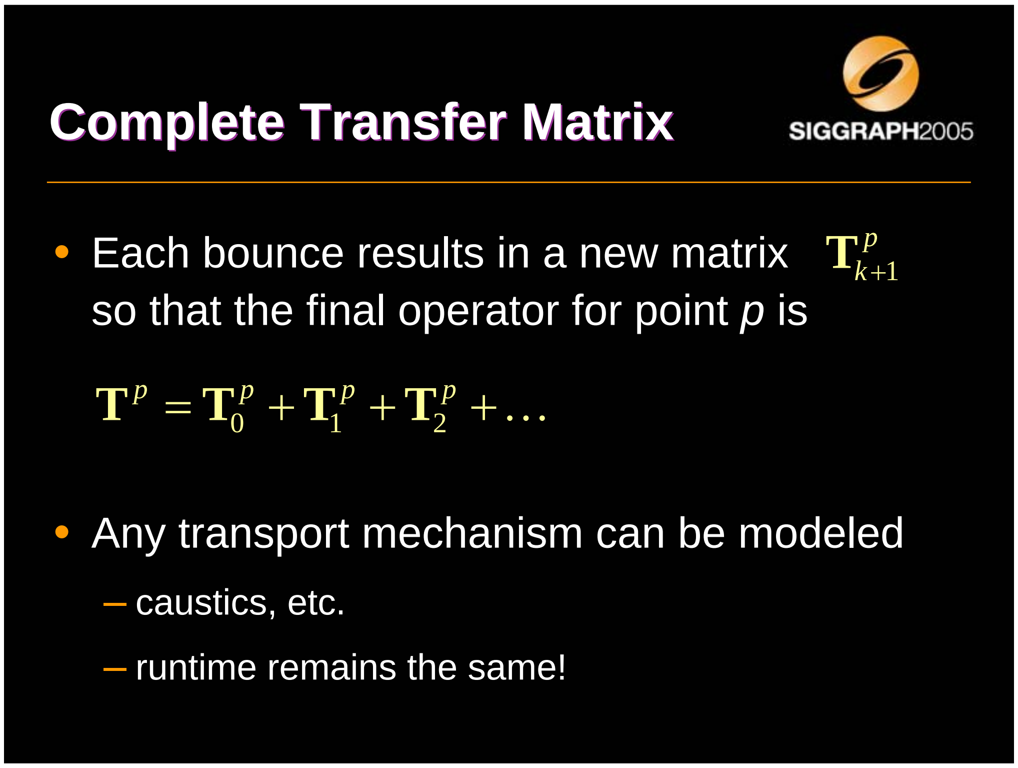


$$T^{p}_{k+1,ji} = \left\langle \int_{\Omega(n')} \sum_{a=1}^{m} \left(\sum_{s=1}^{o} w_s T^{p_s}_{k,ai} \right) z_a(\omega')\, f_r(p',\omega' \to \omega) \cos\theta' \mathrm{d}\omega' \,,\ \tilde{z}_j \right\rangle$$

And finally, as we repeat this process for many different omegas, we project transferred incident radiance from y_i after k+1 bounces into the function space spanned by the z functions.

The inner product (=outer integral <>) is just a “gather” operation as seen from point p: Shoot a number of rays from p, determine the light reflected towards p from where the rays hit (using interpolation from nearby samples), and project that light into the basis z.

If we use spherical harmonics for representing the transferred incident radiance, this bears quite some resemblance to an early non-diffuse radiosity method from 1991 by Sillion and others; they used spherical harmonics for representing outgoing radiance from surface points, and used a similar gathering scheme. They worked with fixed lighting, though.



Complete Transfer Matrix

SIGGRAPH2005

- Each bounce results in a new matrix $\mathbf{T}_{k+1}^{p}$ so that the final operator for point *p* is

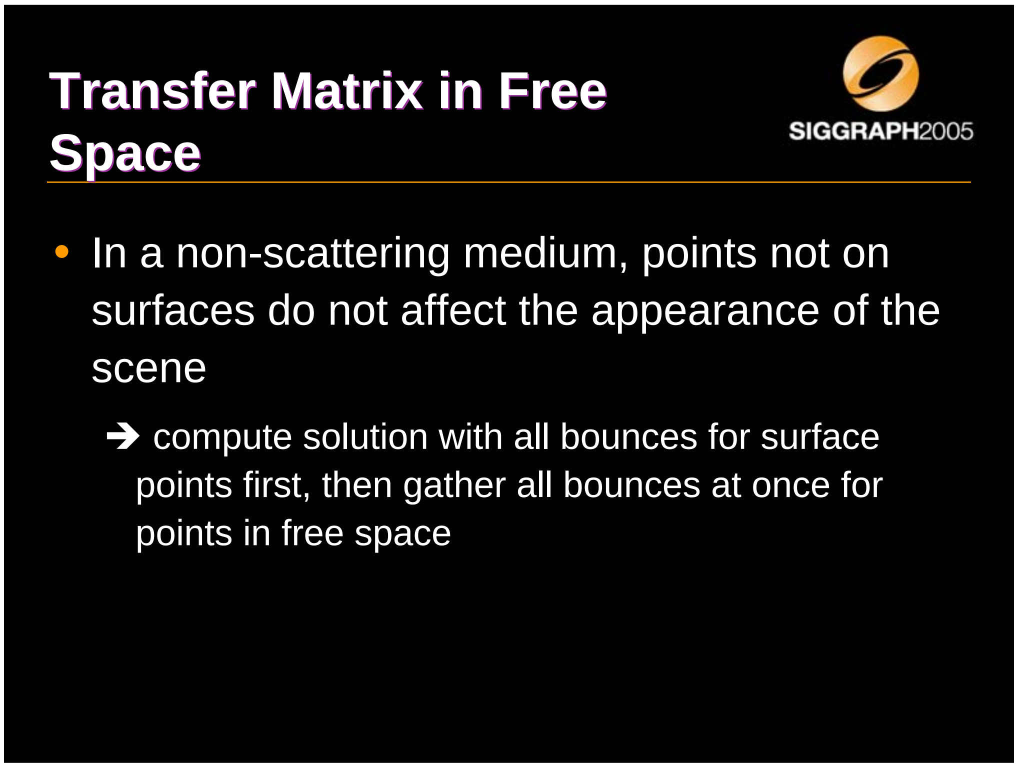
$$\mathbf{T}^{p} = \mathbf{T}_{0}^{p} + \mathbf{T}_{1}^{p} + \mathbf{T}_{2}^{p} + \ldots$$

- Any transport mechanism can be modeled
  - caustics, etc.
  - runtime remains the same!

In the beginning we gave an explicit formula for T_0 and now we have a recursion for T_{k+1} from T_k ➔ we can compute transfer matrices for all bounces.

The compound transfer matrix that accounts for all bounces is obtained from these by just summing them up.

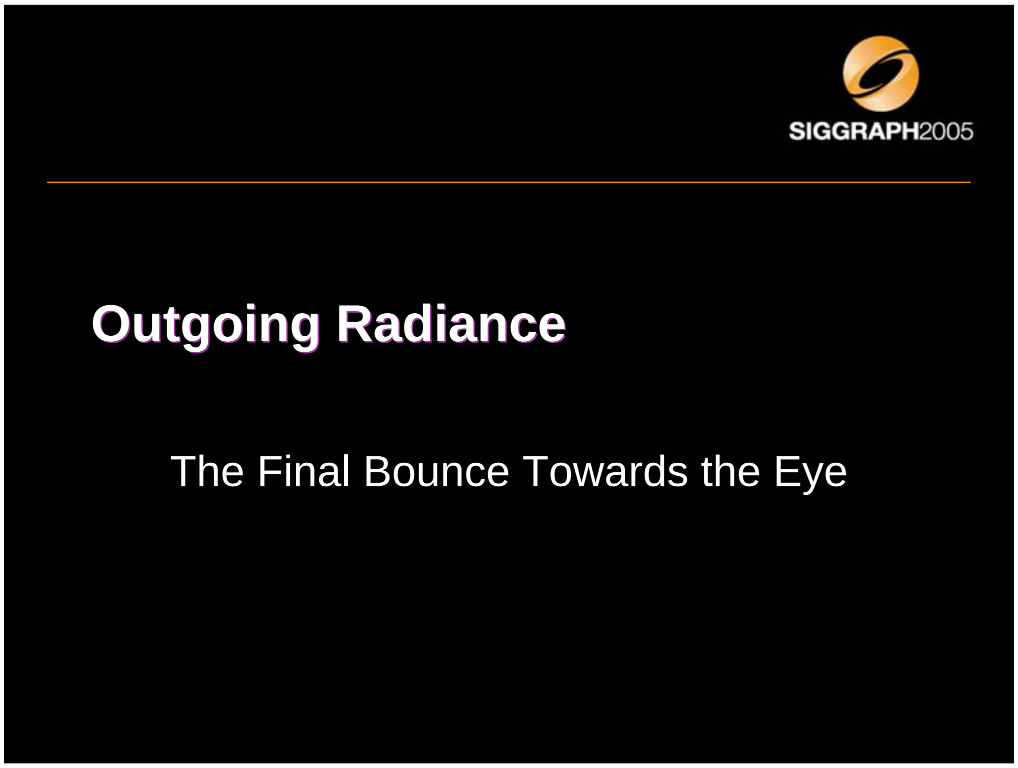
It is worth noticing that nothing in the previous derivation prevents inclusion of subsurface scattering; all you need to do is simulate with your favorite method when determining the matrices T_{k+1} from the previous bounce.



Transfer Matrix in Free Space

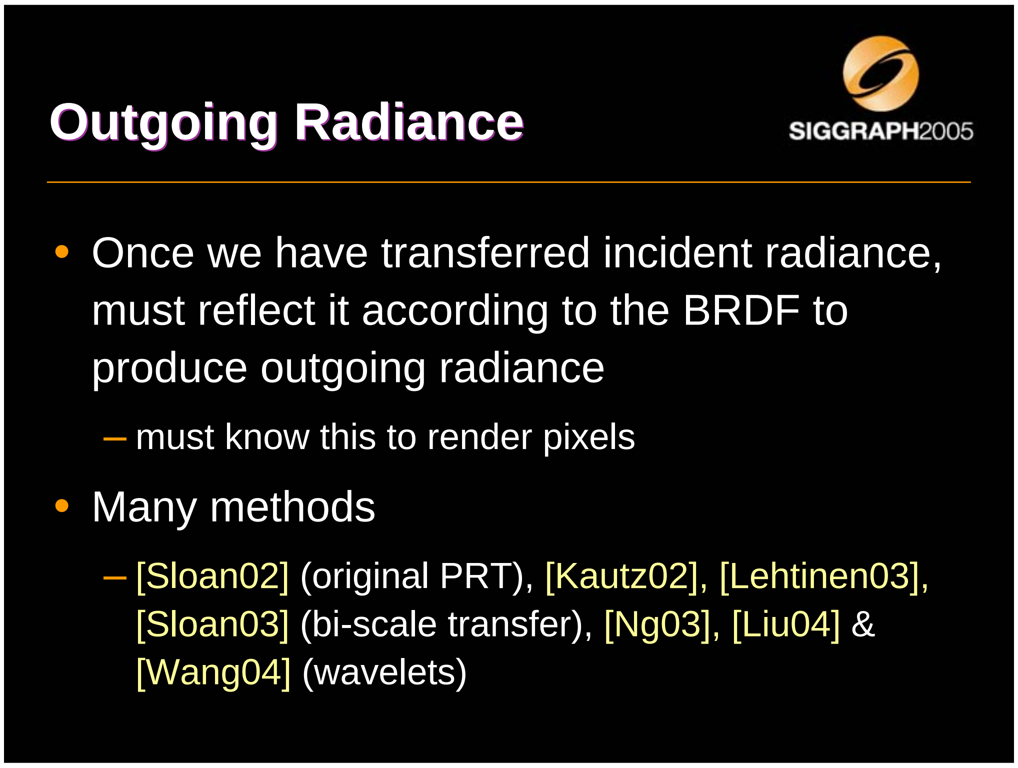
SIGGRAPH2005

- In a non-scattering medium, points not on surfaces do not affect the appearance of the scene
  - ➔ compute solution with all bounces for surface points first, then gather all bounces at once for points in free space



Outgoing Radiance

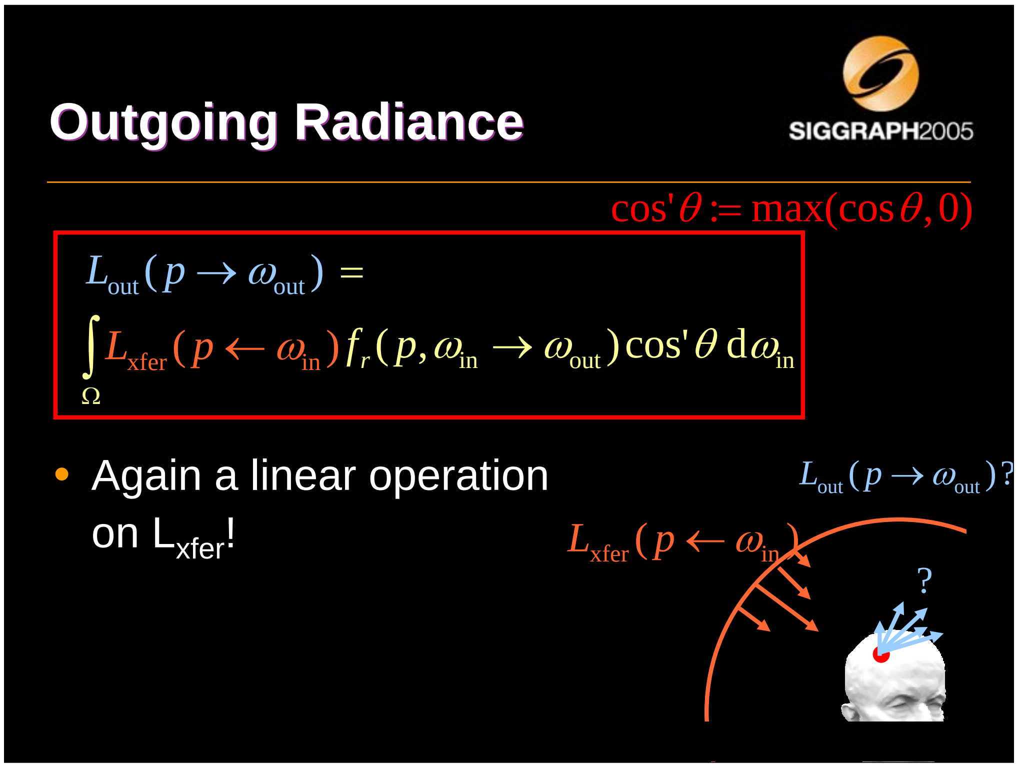
The Final Bounce Towards the Eye



Outgoing Radiance

SIGGRAPH2005

- Once we have transferred incident radiance, must reflect it according to the BRDF to produce outgoing radiance
  - must know this to render pixels
- Many methods
  - [Sloan02] (original PRT), [Kautz02], [Lehtinen03], [Sloan03] (bi-scale transfer), [Ng03], [Liu04] & [Wang04] (wavelets)



Outgoing Radiance

$\cos'\theta := \max(\cos\theta, 0)$

$$\boxed{\begin{aligned} &L_{\mathrm{out}}(p \rightarrow \omega_{\mathrm{out}}) = \\ &\int_{\Omega} L_{\mathrm{xfer}}(p \leftarrow \omega_{\mathrm{in}}) f_r(p, \omega_{\mathrm{in}} \rightarrow \omega_{\mathrm{out}}) \cos'\theta \, \mathrm{d}\omega_{\mathrm{in}} \end{aligned}}$$

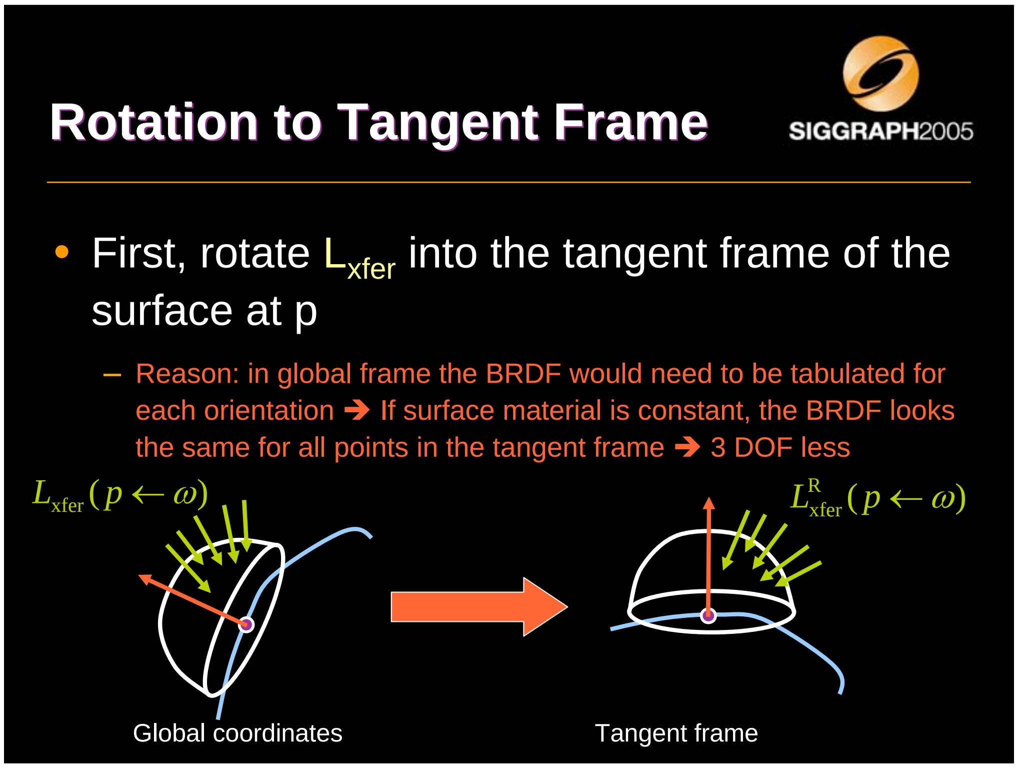
- Again a linear operation on L_{xfer}!

$L_{\mathrm{out}}(p \rightarrow \omega_{\mathrm{out}})?$
$L_{\mathrm{xfer}}(p \leftarrow \omega_{\mathrm{in}})$
?

We'll add another linear operator O^p that maps the transferred incident radiance at p into outgoing radiance.

O may be a **matrix**, in which case it maps the transferred incident radiance into full, spherical outgoing radiance expressed in a new function basis, or it can be a **vector** that depends on the outgoing viewing direction, in which case it is just dotted with the transferred incident radiance coefficients to produce outgoing radiance into the viewing direction.

In order to define the integration domain as the whole sphere instead of the hemisphere centered at the normal as done usually, we'll change the cosine term cos theta = dot(normal, omega_in) to read cos' theta := max(dot(normal, omega_in), 0) to signify clamping of the cosine to zero from below.



Rotation to Tangent Frame

- First, rotate L_{xfer} into the tangent frame of the surface at p
  - Reason: in global frame the BRDF would need to be tabulated for each orientation ➔ If surface material is constant, the BRDF looks the same for all points in the tangent frame ➔ 3 DOF less

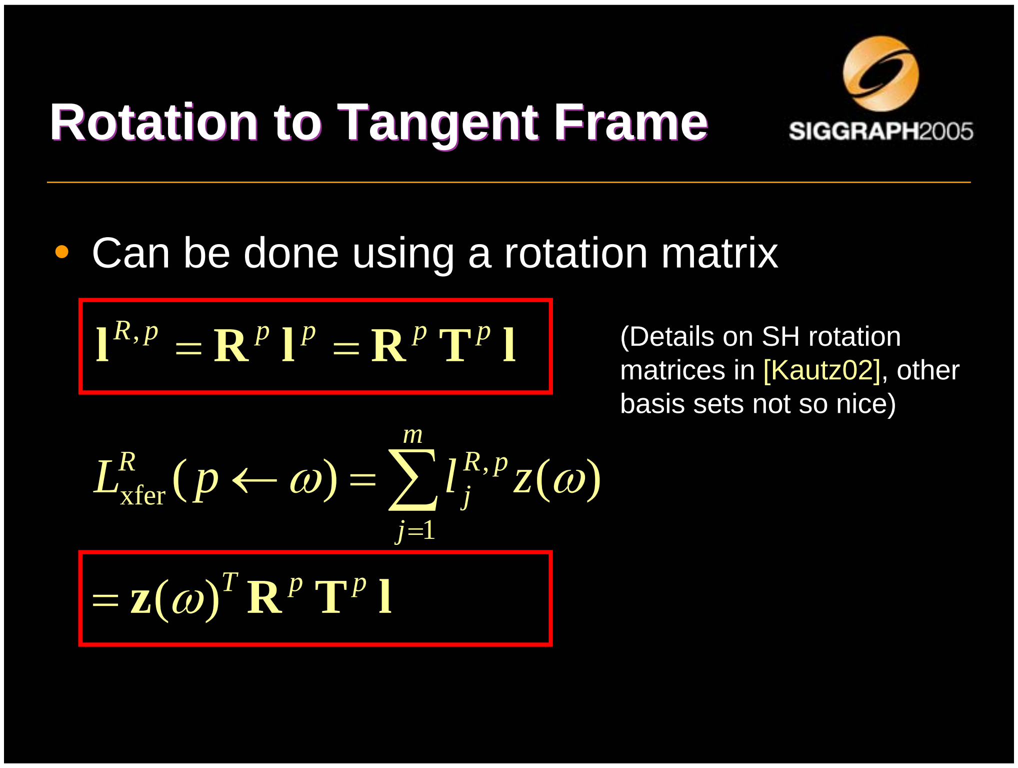
SIGGRAPH2005
$L_{xfer}(p \leftarrow \omega)$
$L^{R}_{xfer}(p \leftarrow \omega)$
Global coordinates
Tangent frame

In order to simplify things a bit, we will perform the outgoing radiance integration in the canonical frame of each point. This canonical space is the so-called tangent space.

Note that this has nothing to do with the orientation of the object; only the orientation of the vertex' tangent space w.r.t. the object space.

The rationale for this is that in the tangent space the BRDF takes the simplest possible form; indeed, if the computation would be done in global coordinates, we would need to perform different computations for surface points oriented differently. This would add a significant storage burden.

However, this step is not strictly necessary. For instance Ng et al. [2004] (triple product wavelet integrals) make the BRDF higher-dimensional by adding the surface orientation as another dimension, and compressing this high-D signal with wavelets.



Rotation to Tangent Frame

SIGGRAPH2005

- Can be done using a rotation matrix

$$\boxed{\mathbf{l}^{R,p} = \mathbf{R}^{p}\,\mathbf{l}^{p} = \mathbf{R}^{p}\,\mathbf{T}^{p}\,\mathbf{l}}$$

(Details on SH rotation matrices in [Kautz02], other basis sets not so nice)

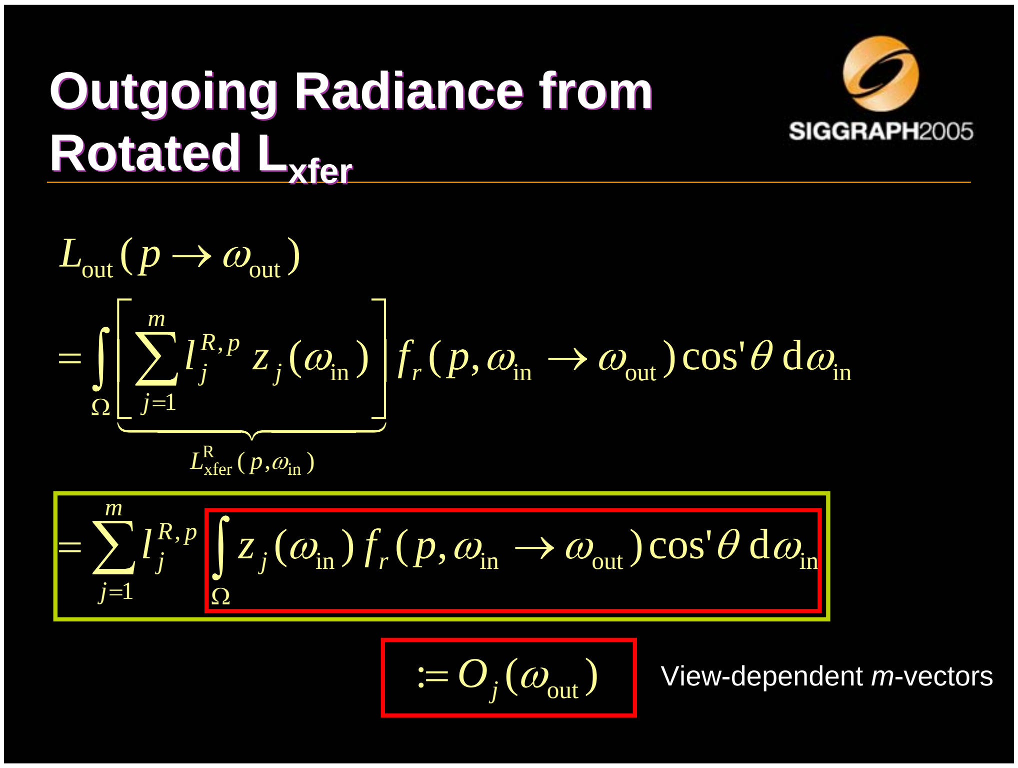
$$L^{R}_{\text{xfer}}(p \leftarrow \omega) = \sum_{j=1}^{m} l_j^{R,p}\, z(\omega)$$

$$\boxed{= \mathbf{z}(\omega)^{T}\,\mathbf{R}^{p}\,\mathbf{T}^{p}\,\mathbf{l}}$$

Since rotation is a linear operation, we can get coefficients of rotated transferred incident radiance by applying a suitable rotation matrix **R** to the coefficients of transferred incident radiance. We denote this by **l**^{R,p}.

Only the Spherical Harmonics and other “steerable” bases are nice in the rotation sense, since they are closed under rotation. This means that any SH expansion can be rotated an arbitrary amount, and still represented **exactly** using the same basis functions. This means the projected lighting environment does not suffer from spurious wobbling or aliasing under rotation.

Other basis sets not necessarily closed under rotation, which means that the rotation cannot be represented exactly with the same basis set. This might result in temporal “wobbling” and similar aliasing artifacts.



Outgoing Radiance from Rotated L_{xfer}

SIGGRAPH2005

$$L_{\text{out}}(p \to \omega_{\text{out}})$$

$$= \int_\Omega \underbrace{\left[\sum_{j=1}^{m} l_j^{R,p}\, z_j(\omega_{\text{in}})\right]}_{L^{\text{R}}_{\text{xfer}}(p,\omega_{\text{in}})} f_r(p,\omega_{\text{in}} \to \omega_{\text{out}}) \cos'\theta \, \mathrm{d}\omega_{\text{in}}$$

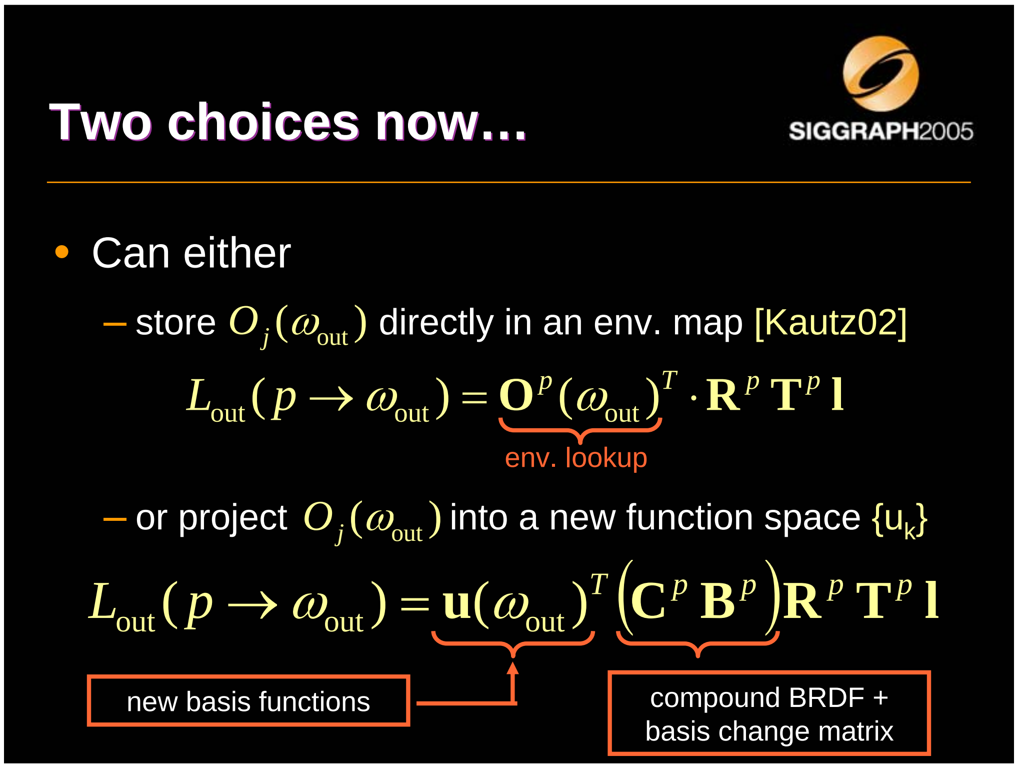
$$\boxed{= \sum_{j=1}^{m} l_j^{R,p} \boxed{\int_\Omega z_j(\omega_{\text{in}})\, f_r(p,\omega_{\text{in}} \to \omega_{\text{out}}) \cos'\theta \, \mathrm{d}\omega_{\text{in}}}}$$

$$\boxed{:= O_j(\omega_{\text{out}})}$$ View-dependent *m*-vectors

In order to get outgoing radiance from transferred, rotated incident radiance, we'll first plug its basis expansion into the usual reflectance equation.

The components of the vector **l**^{R,p} are the basis coefficients of transferred, rotated incident radiance.

The integral in the red box only depends on the basis index j and the outgoing angle – thus, they are view-dependent vectors of length m (the number of basis functions used for representing transferred incident radiance). They can be stored in an environment map, and outgoing radiance can be computed simply by looking the appropriate vector with the view direction and dotting it with the **l**^{R,p}.



Two choices now…

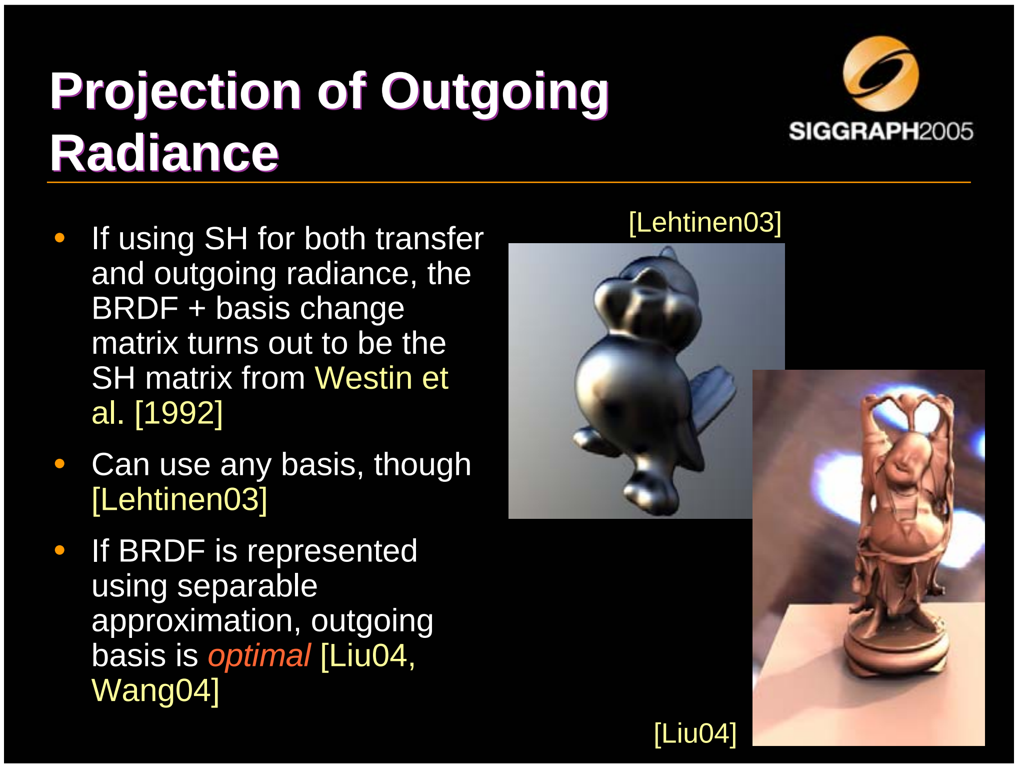
SIGGRAPH2005

- Can either
  - store $O_j(\omega_{\text{out}})$ directly in an env. map [Kautz02]

$$L_{\text{out}}(p \to \omega_{\text{out}}) = \underbrace{\mathbf{O}^p(\omega_{\text{out}})}_{\text{env. lookup}}{}^T \cdot \mathbf{R}^p\,\mathbf{T}^p\,\mathbf{l}$$

  - or project $O_j(\omega_{\text{out}})$ into a new function space $\{u_k\}$

$$L_{\text{out}}(p \to \omega_{\text{out}}) = \underbrace{\mathbf{u}(\omega_{\text{out}})}_{\boxed{\text{new basis functions}}}{}^T \underbrace{\left(\mathbf{C}^p\,\mathbf{B}^p\right)}_{\boxed{\text{compound BRDF + basis change matrix}}}\mathbf{R}^p\,\mathbf{T}^p\,\mathbf{l}$$



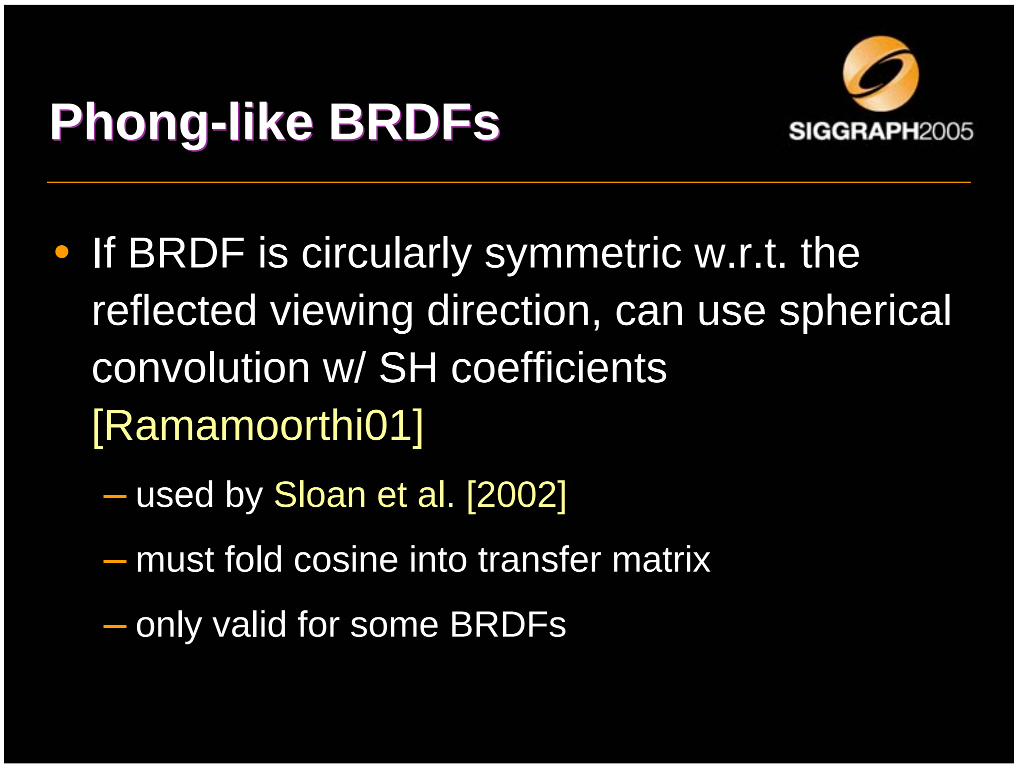
Projection of Outgoing Radiance

SIGGRAPH2005

- If using SH for both transfer and outgoing radiance, the BRDF + basis change matrix turns out to be the SH matrix from Westin et al. [1992]
- Can use any basis, though [Lehtinen03]
- If BRDF is represented using separable approximation, outgoing basis is *optimal* [Liu04, Wang04]

[Lehtinen03]

[Liu04]

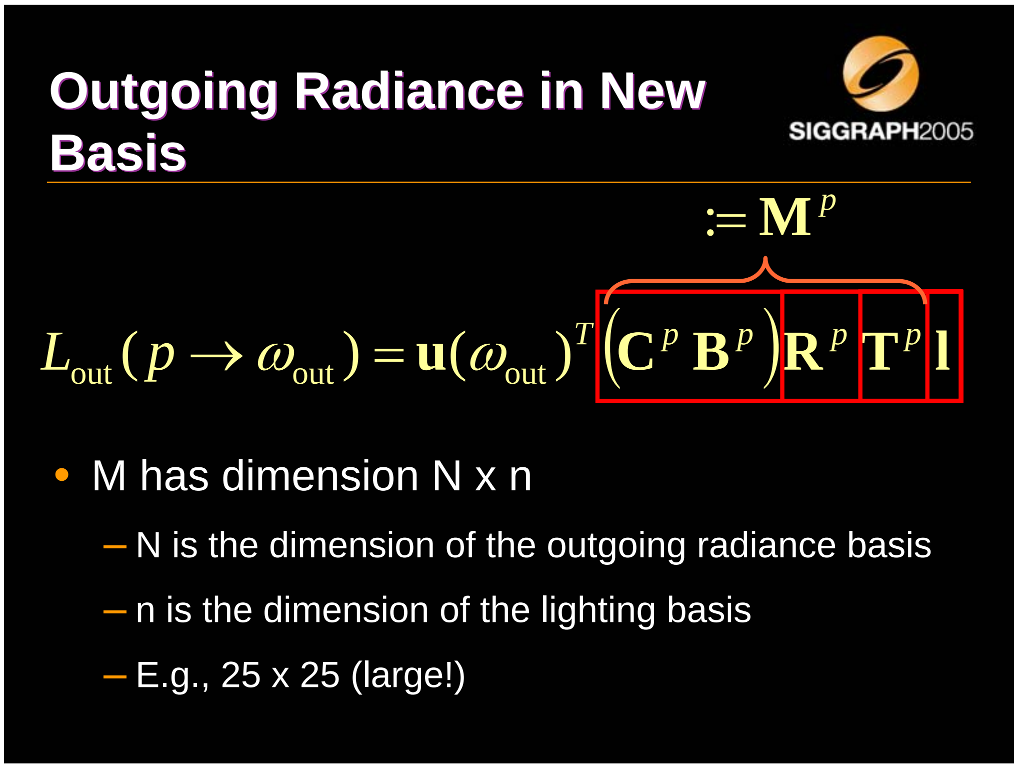


Phong-like BRDFs

SIGGRAPH2005

- If BRDF is circularly symmetric w.r.t. the reflected viewing direction, can use spherical convolution w/ SH coefficients [Ramamoorthi01]
  - used by Sloan et al. [2002]
  - must fold cosine into transfer matrix
  - only valid for some BRDFs

This is what was used by Sloan and others in 2002, but the method has limited applicability because it cannot support all BRDFs.



Outgoing Radiance in New Basis

SIGGRAPH2005

$$L_{\text{out}}(p \to \omega_{\text{out}}) = \mathbf{u}(\omega_{\text{out}})^T \overbrace{\boxed{\left(\mathbf{C}^p\,\mathbf{B}^p\right)}\boxed{\mathbf{R}^p}\boxed{\mathbf{T}^p}}^{:=\mathbf{M}^p}\boxed{\mathbf{l}}$$

- M has dimension N x n
  - N is the dimension of the outgoing radiance basis
  - n is the dimension of the lighting basis
  - E.g., 25 x 25 (large!)

Let’s walk through the whole chain of transformations:

First we have the incident lighting, that is, its basis coefficients l.

My multiplication by T^p, the incident lighting is turned into transferred incident radiance, i.e., an approximation to what point p sees when the scene is lit using the lighting environment specified by l.

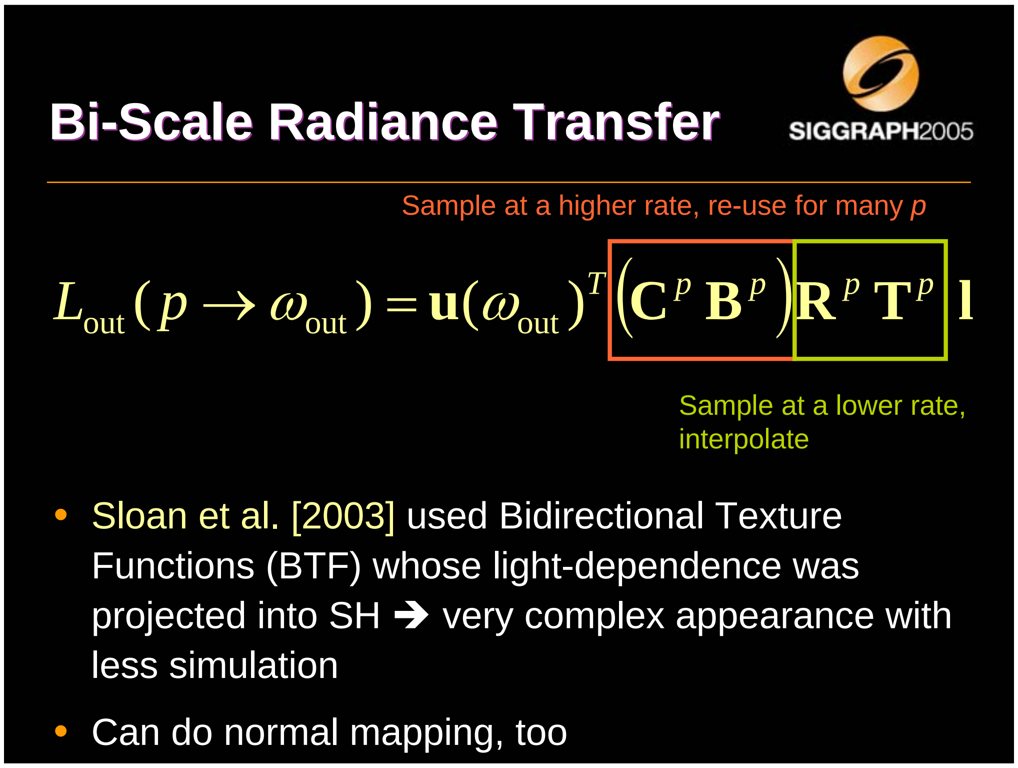
Multiplied by R^p, the transferred incident radiance is rotated into the tangent frame at point p.

Finally, it is turned into outgoing radiance coefficients by multiplication with the compound BRDF + basis change matrix CB.

To get the radiance from p to the viewing direction, the basis functions u are evaluated in the viewing direction and modulated using the outgoing radiance coefficients. Done!

M is a compound radiance transfer operator; it maps the distant, incident illumination into the outgoing radiance from point p, i.e., its “appearance”.

For instance, using 4th order (25-term) SH for incident radiance, and the same basis for outgoing radiance, we have a 25x25 matrix per point p.



Bi-Scale Radiance Transfer

SIGGRAPH2005

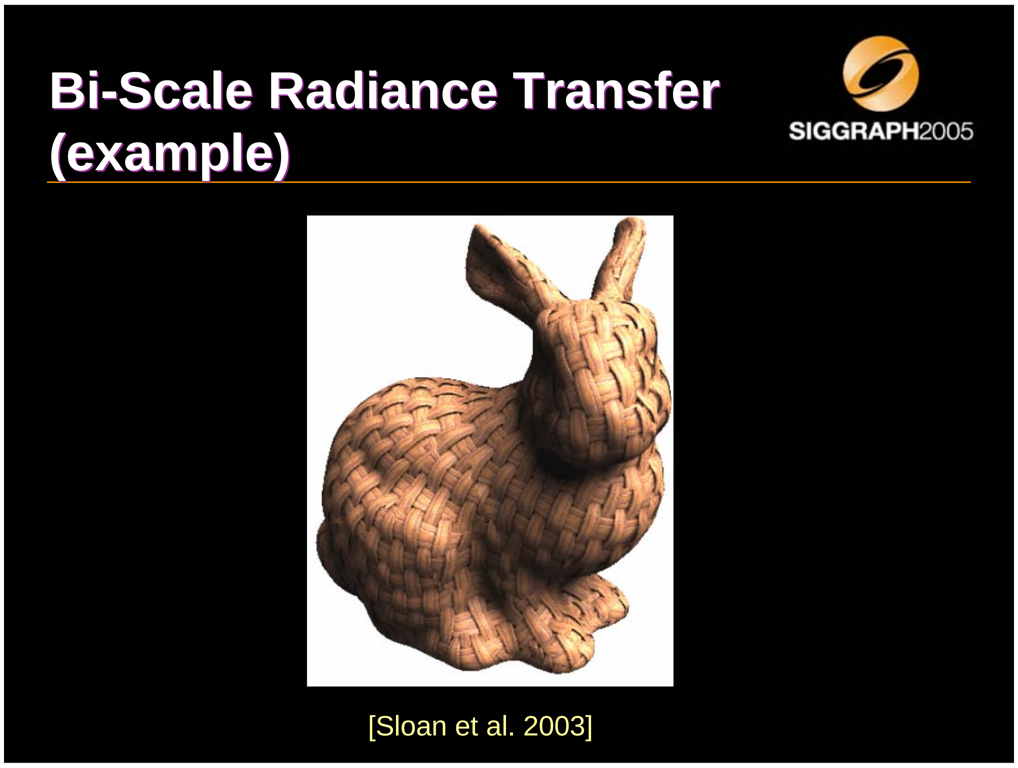
$$L_{\text{out}}(p \to \omega_{\text{out}}) = \mathbf{u}(\omega_{\text{out}})^T \overset{\text{Sample at a higher rate, re-use for many } p}{\boxed{\left(\mathbf{C}^p\,\mathbf{B}^p\right)}}\underset{\text{Sample at a lower rate, interpolate}}{\boxed{\mathbf{R}^p\,\mathbf{T}^p}}\,\mathbf{l}$$

- Sloan et al. [2003] used Bidirectional Texture Functions (BTF) whose light-dependence was projected into SH ➔ very complex appearance with less simulation
- Can do normal mapping, too

The simulation and storage of transfer is expensive, and often the radiance incident onto the surfaces of the scene varies much more slowly than the outgoing radiance.

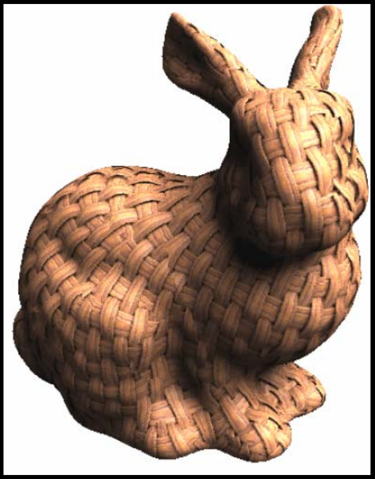
This motivates splitting the general equation into two parts.

The transfer part we sample sparsely and interpolate, while the operator that maps transferred incident radiance into outgoing radiance we sample more densely. Since it is defined only locally, we can re-use the same C*B for many points on the surface.



Bi-Scale Radiance Transfer (example)

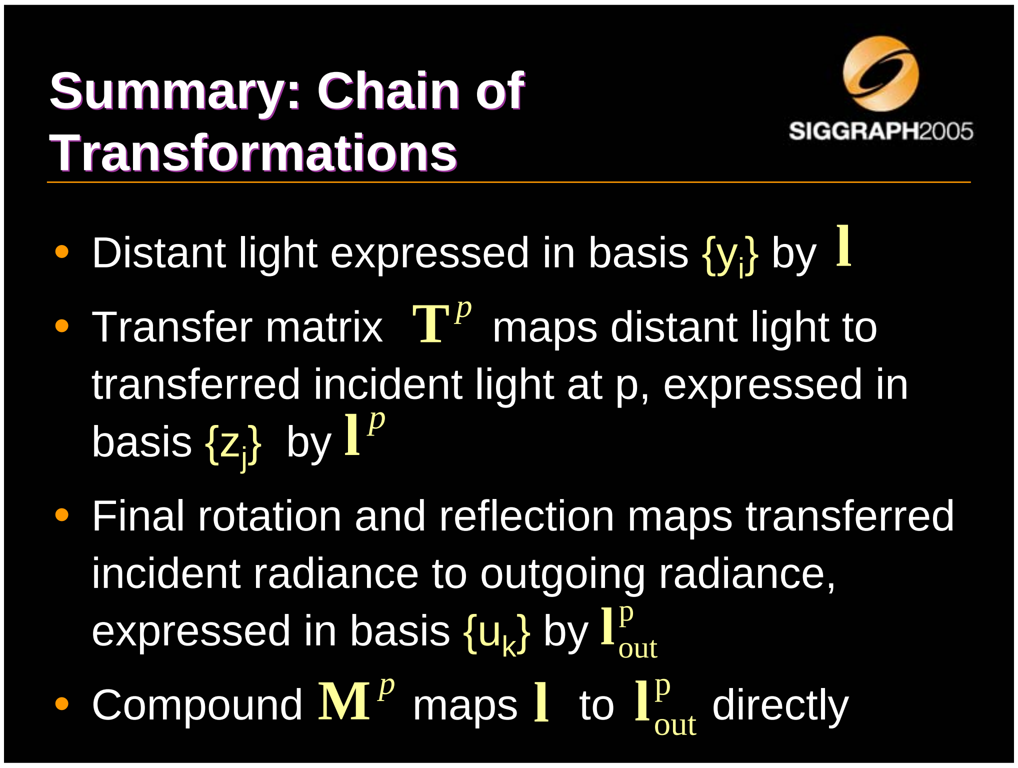
SIGGRAPH2005



[Sloan et al. 2003]

Notice the fine texture on the surface, and how the weave patterns shadow and mask each other.

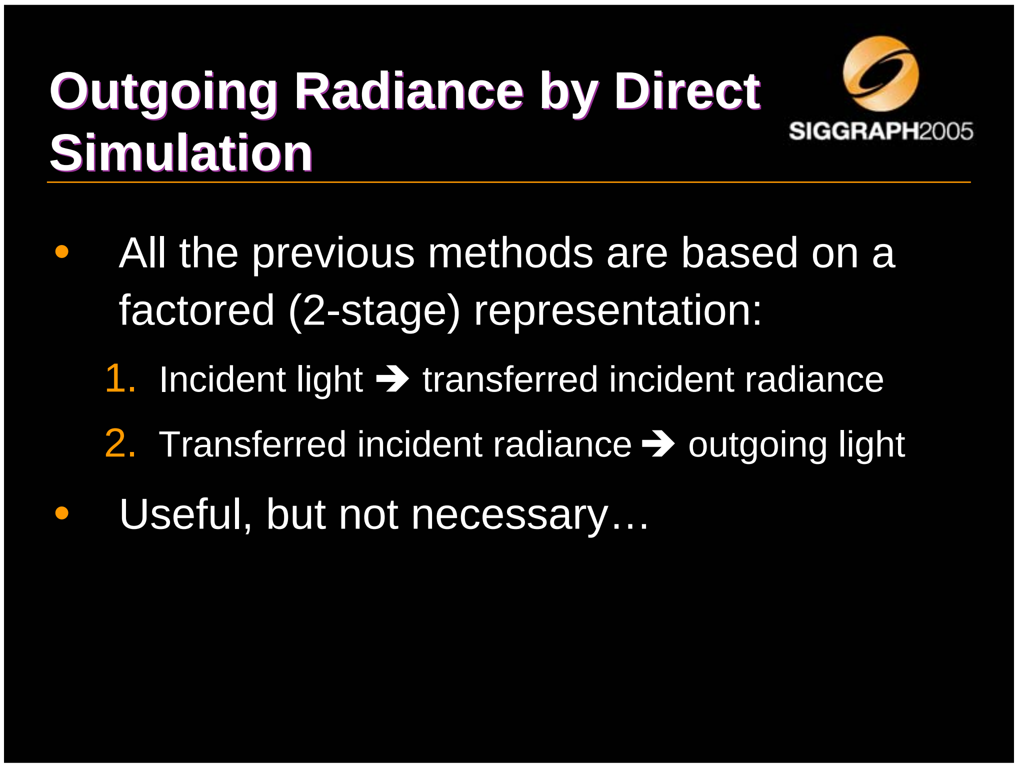
Simulation of similar quality this without the bi-scale factorization would result in an excessive amount of data. Much of it would be unnecessary too; the small, scale effects usually do not affect the macro-scale transfer (large shadows and interreflections) too much.



Summary: Chain of Transformations

SIGGRAPH2005

- Distant light expressed in basis $\{y_i\}$ by $\mathbf{l}$
- Transfer matrix $\mathbf{T}^p$ maps distant light to transferred incident light at p, expressed in basis $\{z_j\}$ by $\mathbf{l}^p$
- Final rotation and reflection maps transferred incident radiance to outgoing radiance, expressed in basis $\{u_k\}$ by $\mathbf{l}^p_{out}$
- Compound $\mathbf{M}^p$ maps $\mathbf{l}$ to $\mathbf{l}^p_{out}$ directly

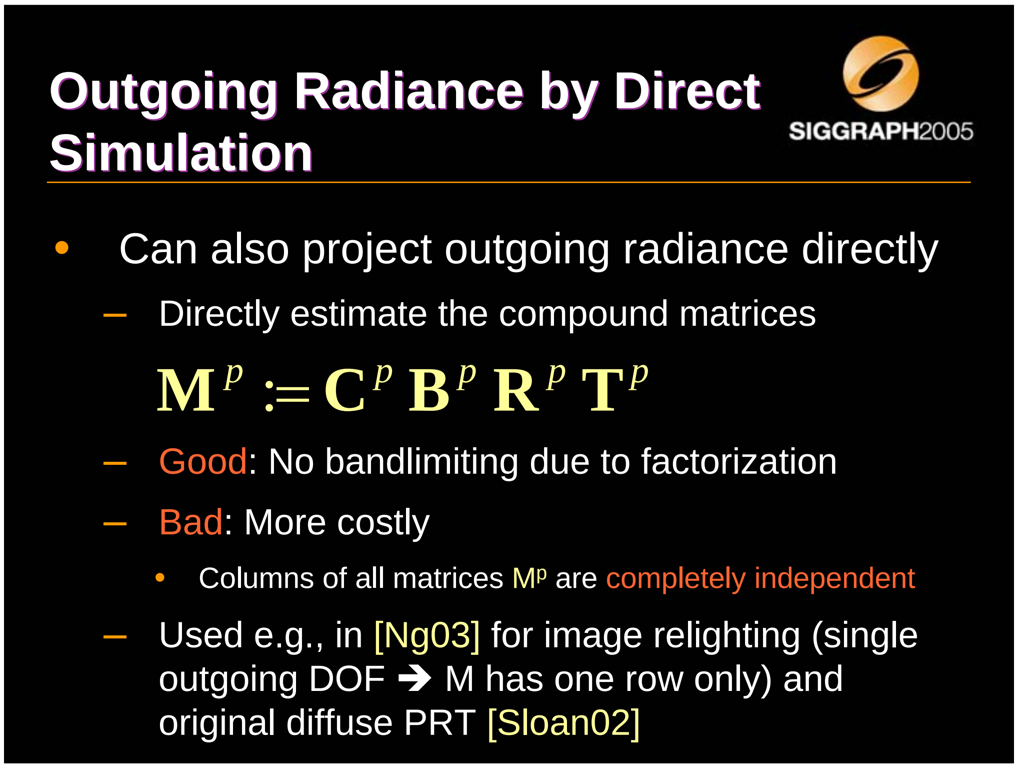


Outgoing Radiance by Direct Simulation

SIGGRAPH2005

- All the previous methods are based on a factored (2-stage) representation:
  1. Incident light ➔ transferred incident radiance
  2. Transferred incident radiance ➔ outgoing light
- Useful, but not necessary…

The two-stage method that we've presented here is not the only possible way.



Outgoing Radiance by Direct Simulation

SIGGRAPH2005

- Can also project outgoing radiance directly
  - Directly estimate the compound matrices

    $$\mathbf{M}^p := \mathbf{C}^p \mathbf{B}^p \mathbf{R}^p \mathbf{T}^p$$

  - Good: No bandlimiting due to factorization
  - Bad: More costly
    - Columns of all matrices M^p are completely independent
  - Used e.g., in [Ng03] for image relighting (single outgoing DOF ➔ M has one row only) and original diffuse PRT [Sloan02]

In this case each column of M directly answers the question: “What does point p look like when the scene is illuminated by basis light i?”

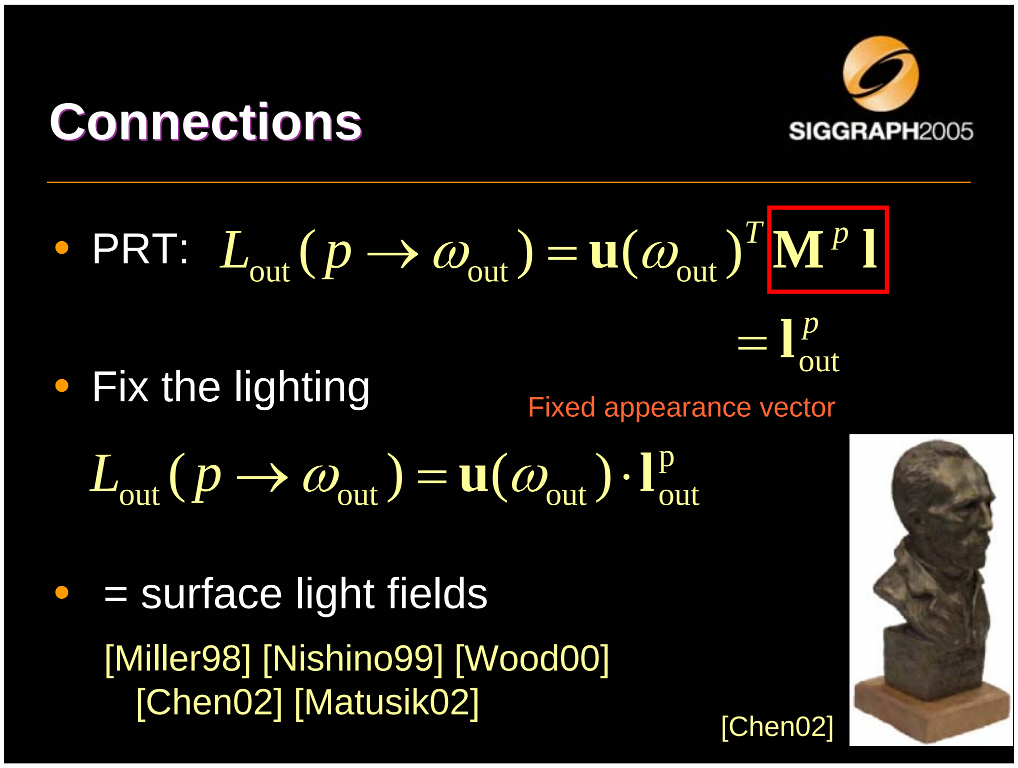
Because the computation is not split anywhere, all the matrices must be computed separately by for instance Monte Carlo path tracing.

Note! All image relighting methods basically do this, but with a single DOF for the outgoing radiance (only light towards the camera in a fixed position)

Because all results are path traced, we can represent arbitrarily complex transport paths regardless of the sampling on surfaces.

But because of the this, cannot utilize a Neumann series and intermediate results for other points, and thus must compute all bounces for all points separetely.

While all this is certainly possible, this method is computationally heavier than the methods we’ve presented before, and it doesn’t allow for the decoupling of the sampling rates for transfer and final reflection.



Connections

SIGGRAPH2005

- PRT: $L_{\text{out}}(p \to \omega_{\text{out}}) = \mathbf{u}(\omega_{\text{out}})^T \boxed{\mathbf{M}^p \mathbf{l}}$

$$= \mathbf{l}^p_{\text{out}}$$

- Fix the lighting

Fixed appearance vector

$$L_{\text{out}}(p \to \omega_{\text{out}}) = \mathbf{u}(\omega_{\text{out}}) \cdot \mathbf{l}^p_{\text{out}}$$

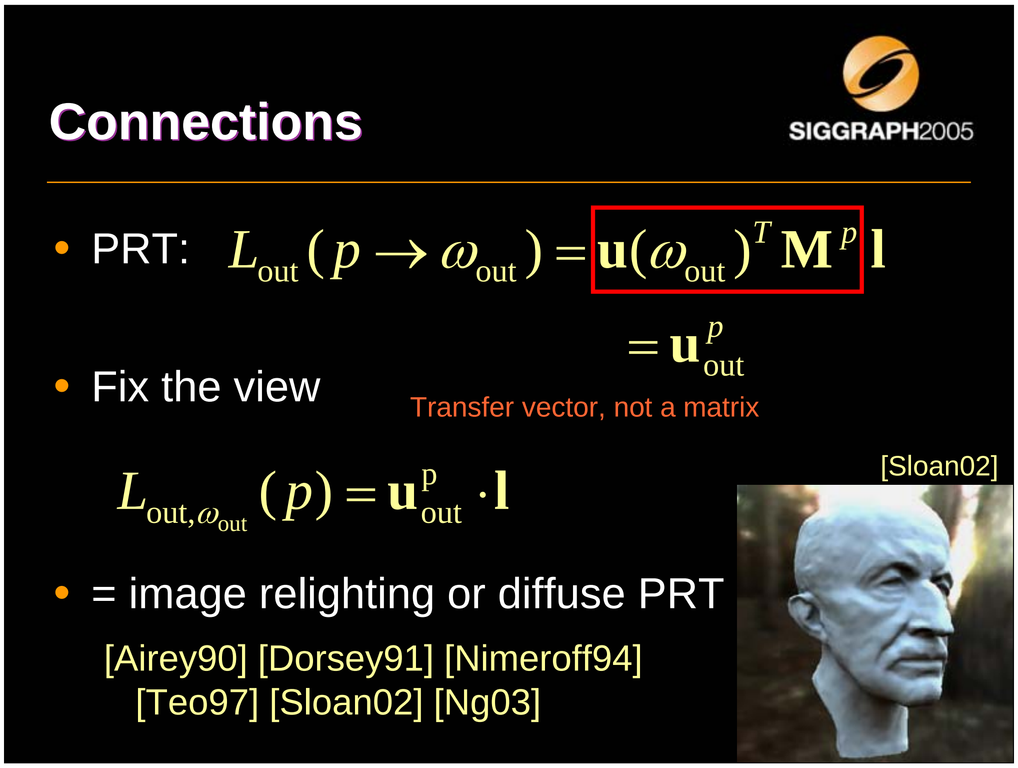
- = surface light fields

[Miller98] [Nishino99] [Wood00] [Chen02] [Matusik02]

[Chen02]

We can ask: What if we fix the lighting? Then for each p, the product M^p l can be precomputed; this vector directly encodes the appearance of point p.

This is a surface light field. Note that we cannot rotate the object w.r.t. the lighting environment any more, but we can look at it from any viewpoint.



Connections

SIGGRAPH2005

- PRT: $L_{\text{out}}(p \rightarrow \omega_{\text{out}}) = \boxed{\mathbf{u}(\omega_{\text{out}})^T \mathbf{M}^p}\,\mathbf{l}$

$$= \mathbf{u}^p_{\text{out}}$$

- Fix the view

Transfer vector, not a matrix

$$L_{\text{out},\omega_{\text{out}}}(p) = \mathbf{u}^{\text{p}}_{\text{out}} \cdot \mathbf{l}$$

- = image relighting or diffuse PRT

[Airey90] [Dorsey91] [Nimeroff94] [Teo97] [Sloan02] [Ng03]

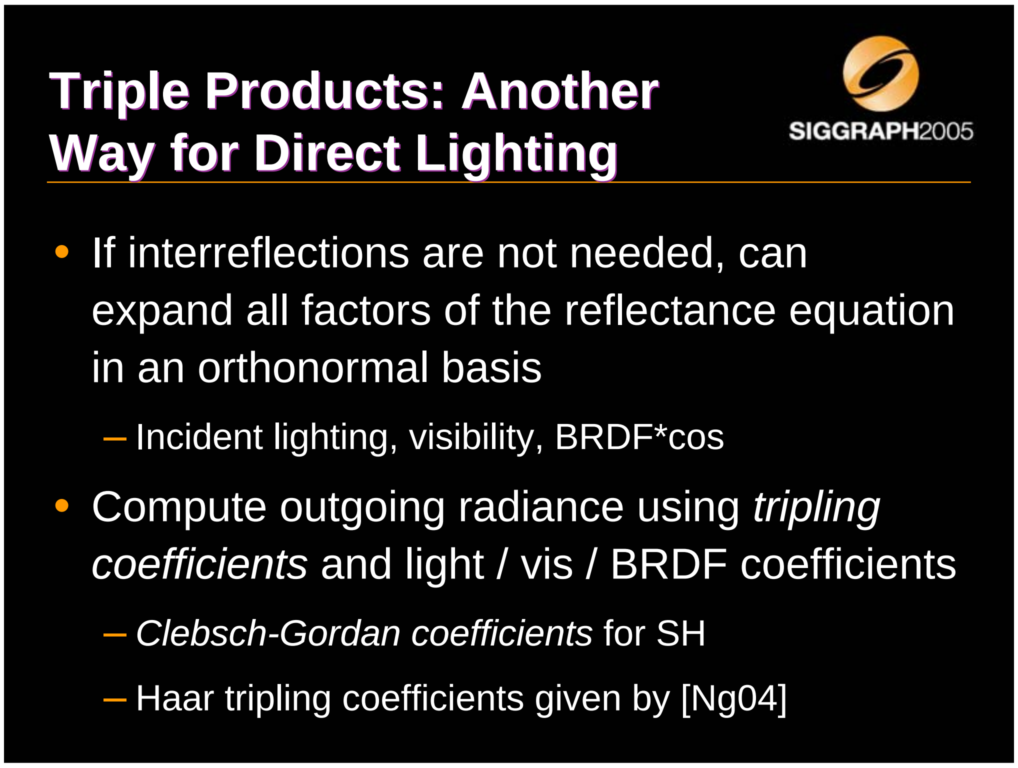
[Sloan02]

And what if we fix the view? This means, for instance, fixing the camera so that it cannot move; we are rendering a fixed picture of the object.

Then the basis function vector u is fixed, and its product with M^p can be precomputed as before.

Now the lighting may vary, but the view not – this is image relighting.

Also, if all surfaces are diffuse, their appearance is captured by a single vector, not matrix; i.e., diffuse PRT (what was described earlier) may be seen as a special case of this.

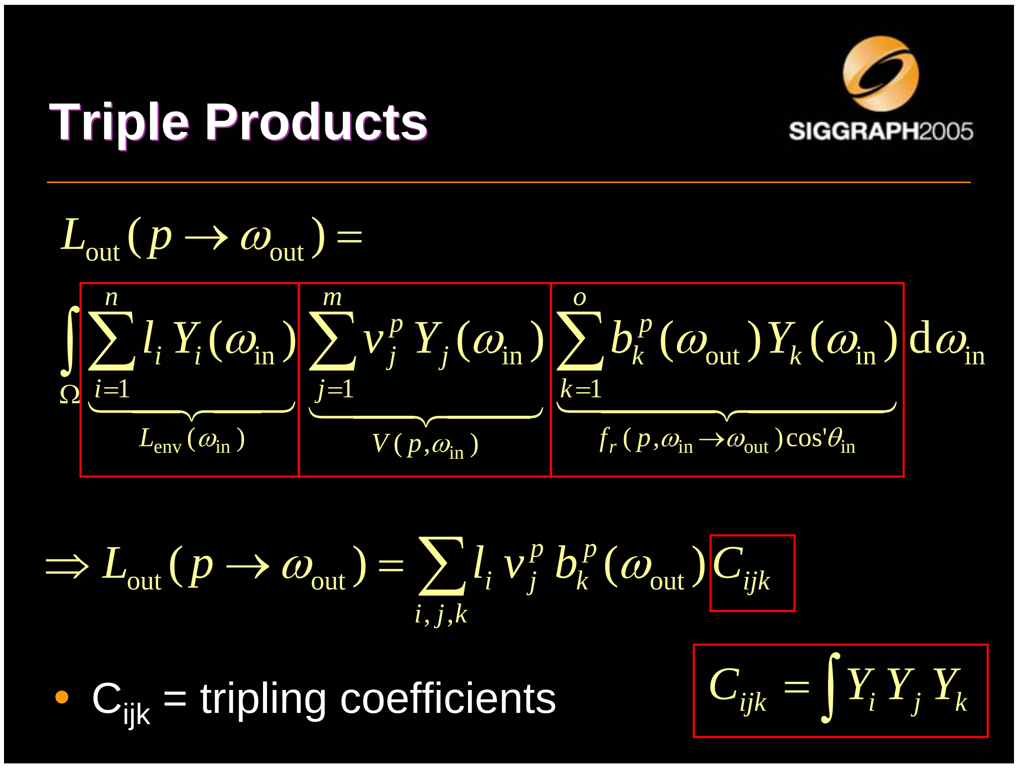


Triple Products: Another Way for Direct Lighting

SIGGRAPH2005

- If interreflections are not needed, can expand all factors of the reflectance equation in an orthonormal basis
  - Incident lighting, visibility, BRDF*cos
- Compute outgoing radiance using *tripling coefficients* and light / vis / BRDF coefficients
  - *Clebsch-Gordan coefficients* for SH
  - Haar tripling coefficients given by [Ng04]

Here we assume that the lighting environment, visibility and BRDF*cos are all projected into the same function space, although that is not strictly necessary, as triple products can be defined also for mixed bases.



Triple Products

SIGGRAPH2005

$$L_{\text{out}}(p \to \omega_{\text{out}}) =$$

$$\int_\Omega \boxed{\underbrace{\sum_{i=1}^{n} l_i\, Y_i(\omega_{\text{in}})}_{L_{\text{env}}(\omega_{\text{in}})}} \boxed{\underbrace{\sum_{j=1}^{m} v_j^p\, Y_j(\omega_{\text{in}})}_{V(p,\omega_{\text{in}})}} \boxed{\underbrace{\sum_{k=1}^{o} b_k^p(\omega_{\text{out}})\, Y_k(\omega_{\text{in}})}_{f_r(p,\omega_{\text{in}} \to \omega_{\text{out}})\cos'\theta_{\text{in}}}}\, \mathrm{d}\omega_{\text{in}}$$

$$\Rightarrow L_{\text{out}}(p \to \omega_{\text{out}}) = \sum_{i,j,k} l_i\, v_j^p\, b_k^p(\omega_{\text{out}}) \boxed{C_{ijk}}$$

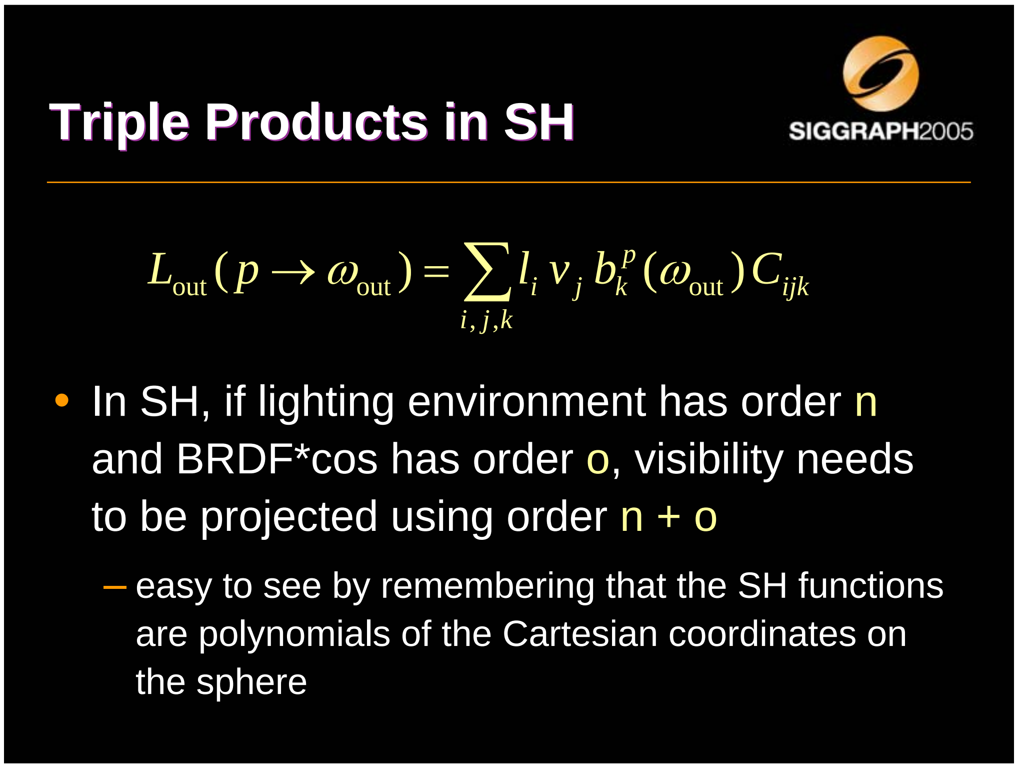
- C_{ijk} = tripling coefficients

$$\boxed{C_{ijk} = \int Y_i\, Y_j\, Y_k}$$

The outgoing radiance from point p (without interreflections) is a **tri-linear function of the lighting environment, visibility and BRDF*cos**, i.e., the resulting value is linear separately in each of these functions.

All the basis expansion sums and coefficients can be moved out of the reflectance integral, and so we are left with a trilinear expression with the vectors l, v, and b and the tripling coefficients C_ijk.

The tripling coefficients are actually a 3D tensor, as can be seen from the triple sum expression. Note that the tensor has high symmetry, since the indices i,j,k can be freely permuted and the tripling coefficients remain the same – this is obvious from their definition. Note that if we used different basis sets for the three functions, this symmetry would be lost.

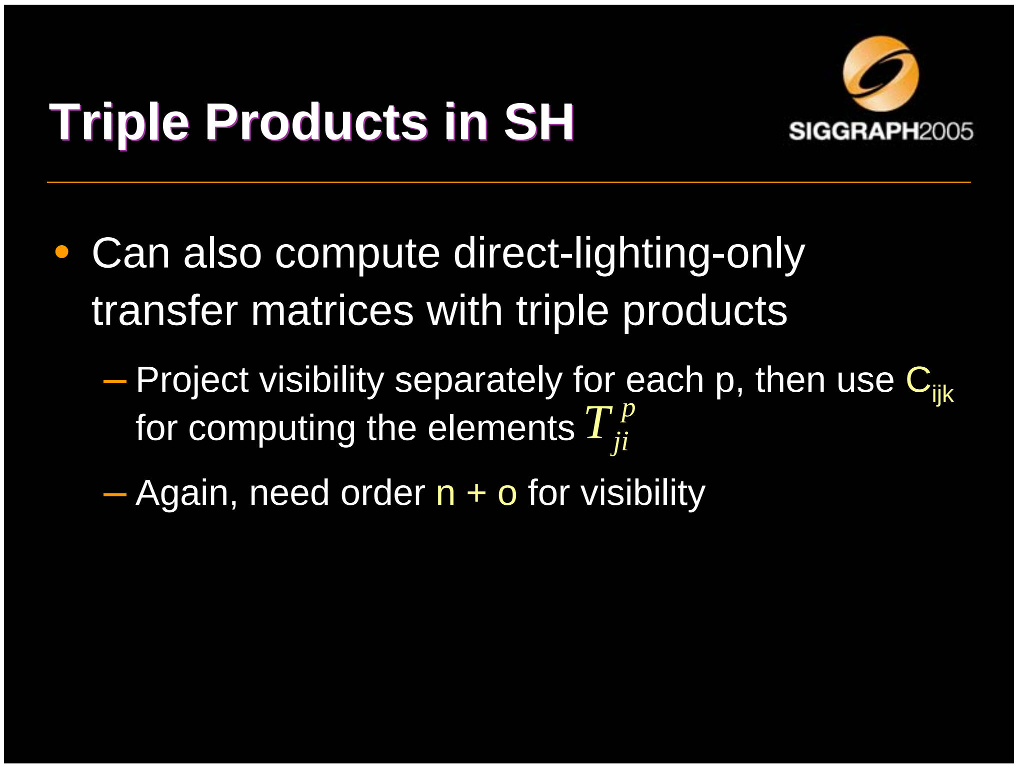


Triple Products in SH

SIGGRAPH2005

$$L_{\text{out}}(p \to \omega_{\text{out}}) = \sum_{i,j,k} l_i \, v_j \, b_k^p(\omega_{\text{out}}) \, C_{ijk}$$

- In SH, if lighting environment has order n and BRDF*cos has order o, visibility needs to be projected using order n + o
  - easy to see by remembering that the SH functions are polynomials of the Cartesian coordinates on the sphere



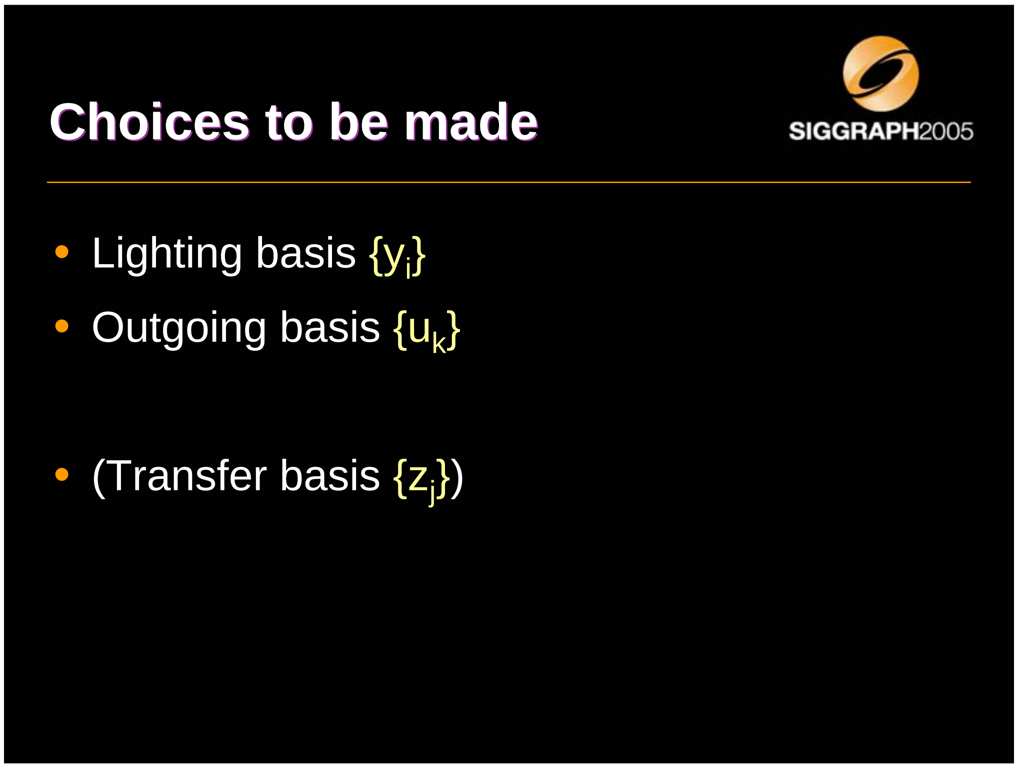
Triple Products in SH

SIGGRAPH2005

- Can also compute direct-lighting-only transfer matrices with triple products
  - Project visibility separately for each p, then use C_{ijk} for computing the elements T_{ji}^{p}
  - Again, need order n + o for visibility

Properties of Different Basis Sets

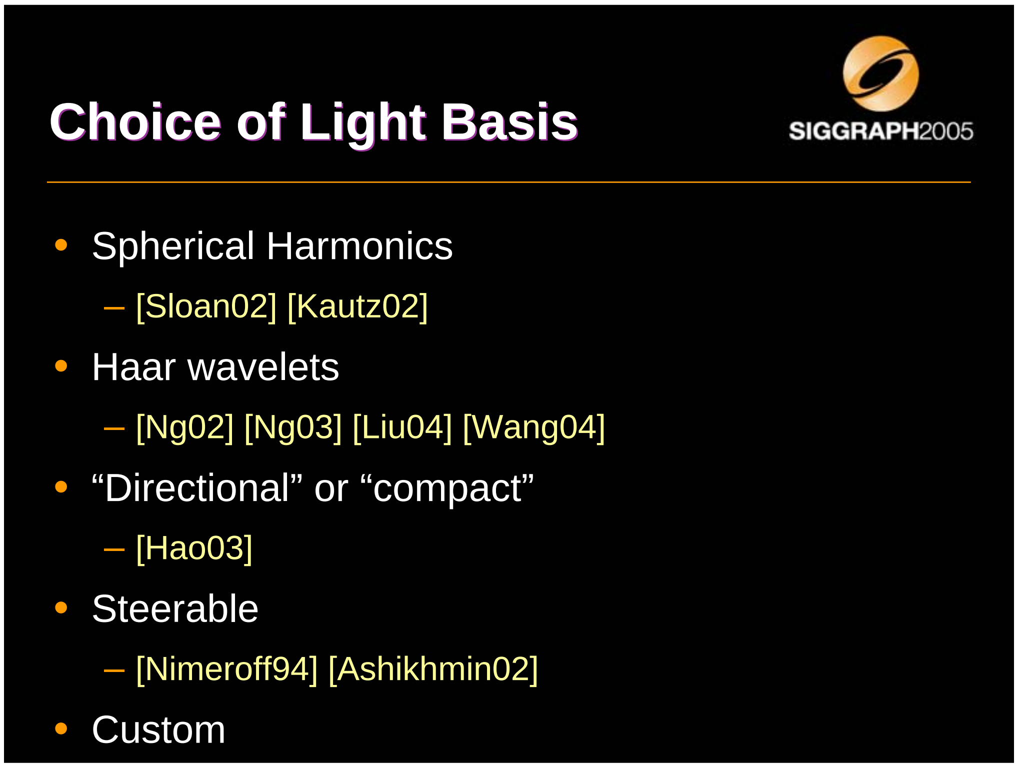
SIGGRAPH2005



Choices to be made

SIGGRAPH2005

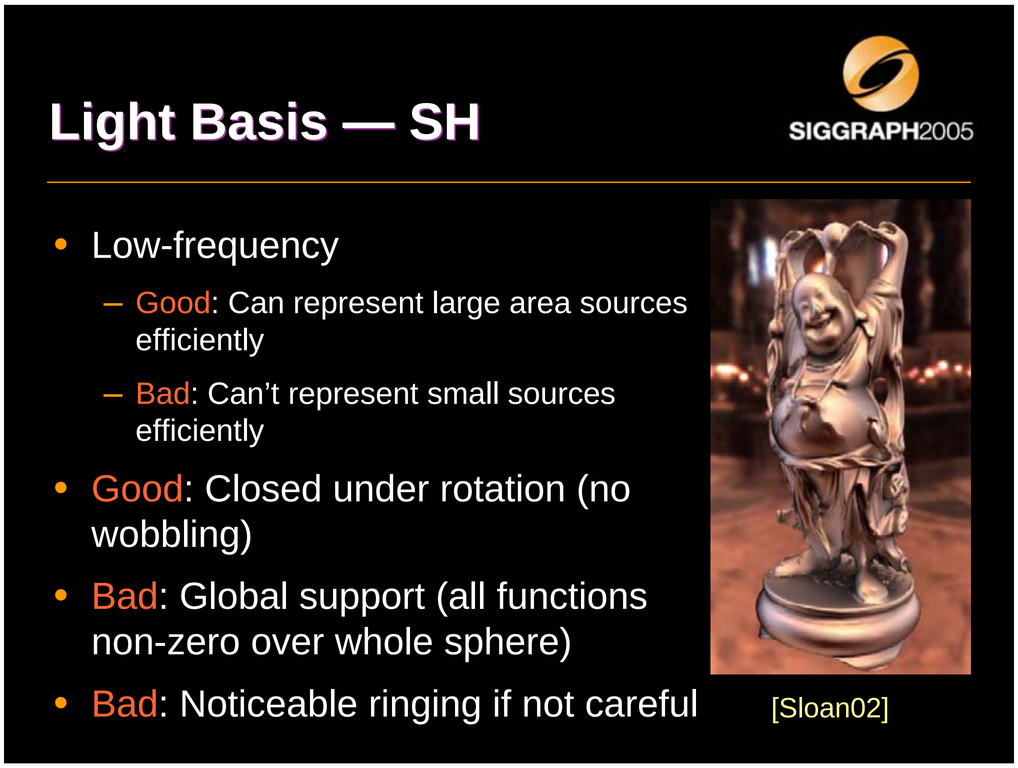
- Lighting basis $\{y_i\}$
- Outgoing basis $\{u_k\}$
- (Transfer basis $\{z_j\}$)



Choice of Light Basis

SIGGRAPH2005

- Spherical Harmonics
  - [Sloan02] [Kautz02]
- Haar wavelets
  - [Ng02] [Ng03] [Liu04] [Wang04]
- “Directional” or “compact”
  - [Hao03]
- Steerable
  - [Nimeroff94] [Ashikhmin02]
- Custom

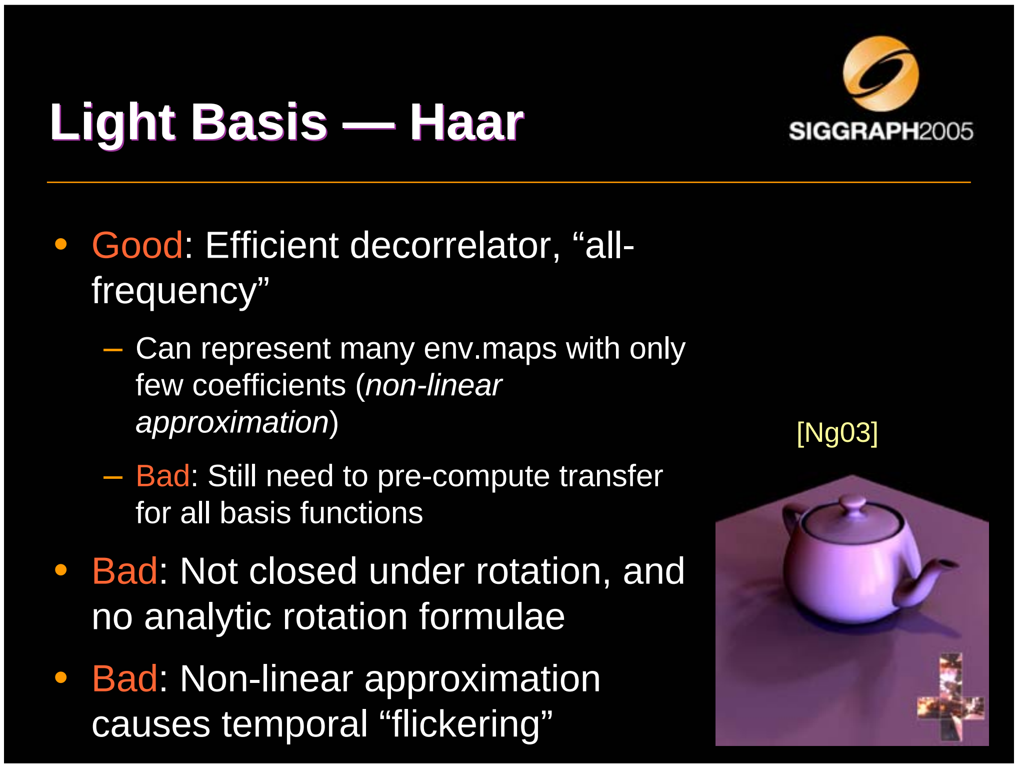


Light Basis — SH

SIGGRAPH2005

- Low-frequency
  - Good: Can represent large area sources efficiently
  - Bad: Can’t represent small sources efficiently
- Good: Closed under rotation (no wobbling)
- Bad: Global support (all functions non-zero over whole sphere)
- Bad: Noticeable ringing if not careful

[Sloan02]

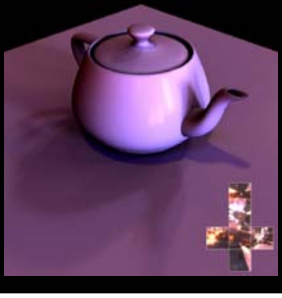


Light Basis — Haar

SIGGRAPH2005

- Good: Efficient decorrelator, “all-frequency”
  - Can represent many env.maps with only few coefficients (*non-linear approximation*)
  - Bad: Still need to pre-compute transfer for all basis functions
- Bad: Not closed under rotation, and no analytic rotation formulae
- Bad: Non-linear approximation causes temporal “flickering”

[Ng03]

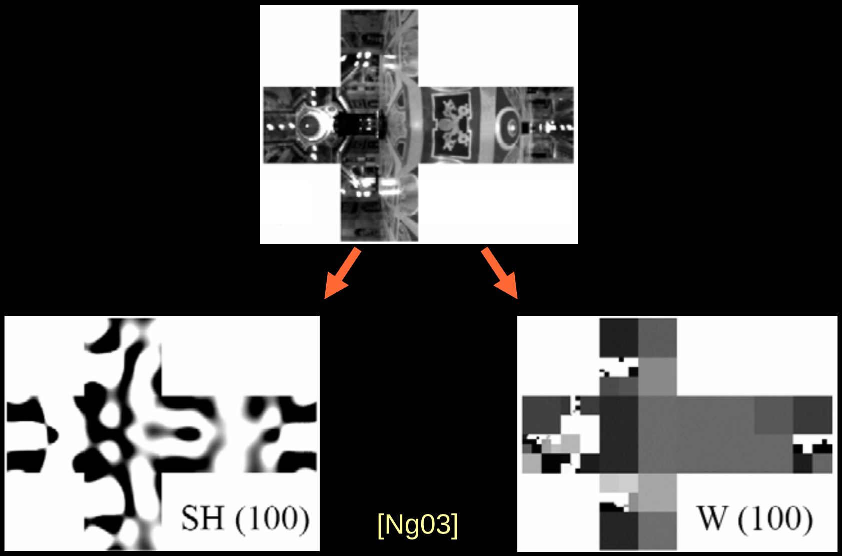


Non-linear approximation means only using those basis functions whose projection coefficients are sufficiently far from zero. The basis functions that carry significant energy for an environment map change for each environment; thus, if we want to render using arbitrary lighting, the transfer must be computed for each basis function, even if only a small subset of them would be required for rendering with any given fixed environment.

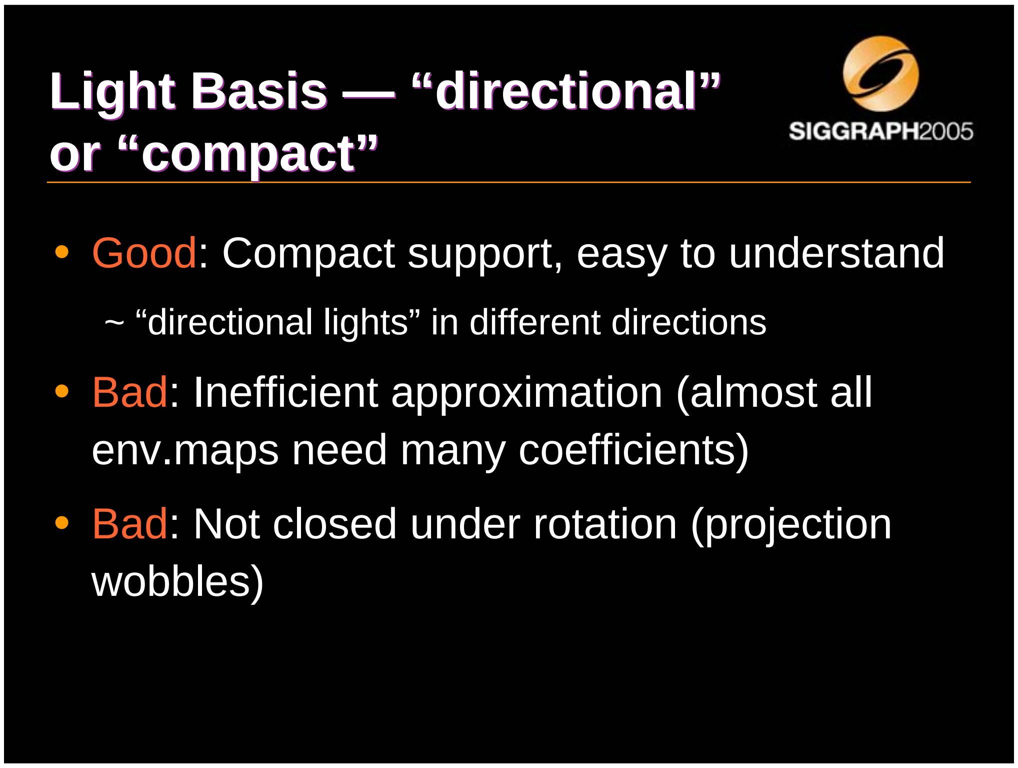
This is a lot of work if we want to support all-frequency lighting, but if the price can be paid, results are very convincing.

Light Basis – SH vs. Haar

SIGGRAPH2005



[Ng03]



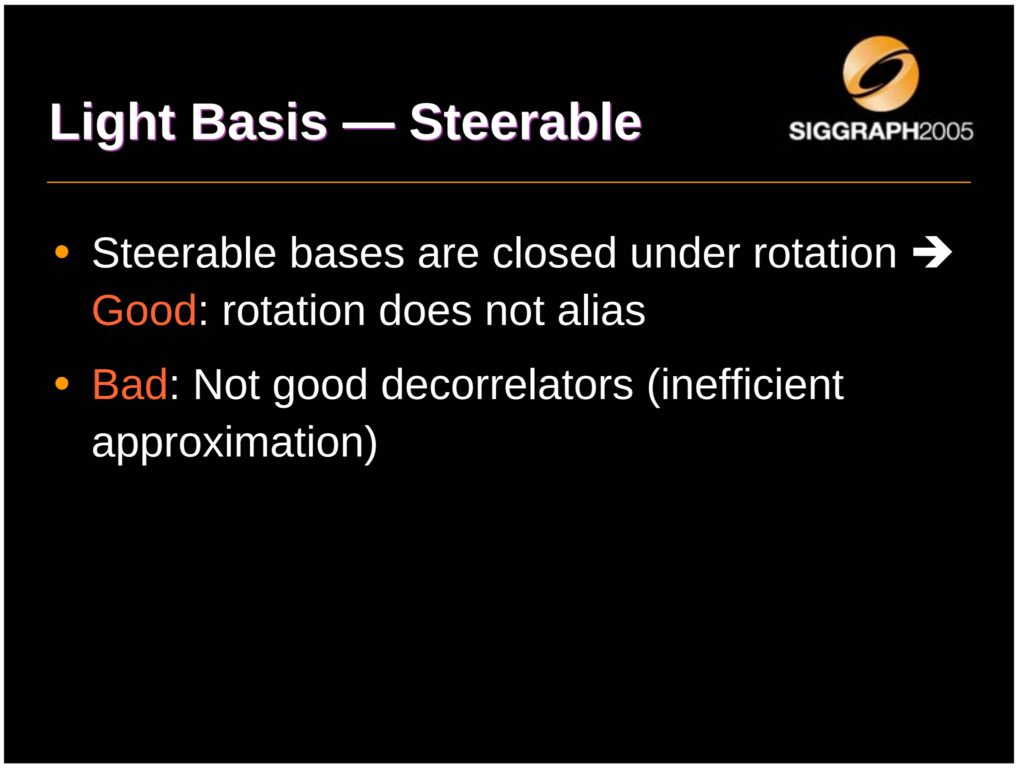
Light Basis — “directional” or “compact”

SIGGRAPH2005

- Good: Compact support, easy to understand

  ~ “directional lights” in different directions
- Bad: Inefficient approximation (almost all env.maps need many coefficients)
- Bad: Not closed under rotation (projection wobbles)

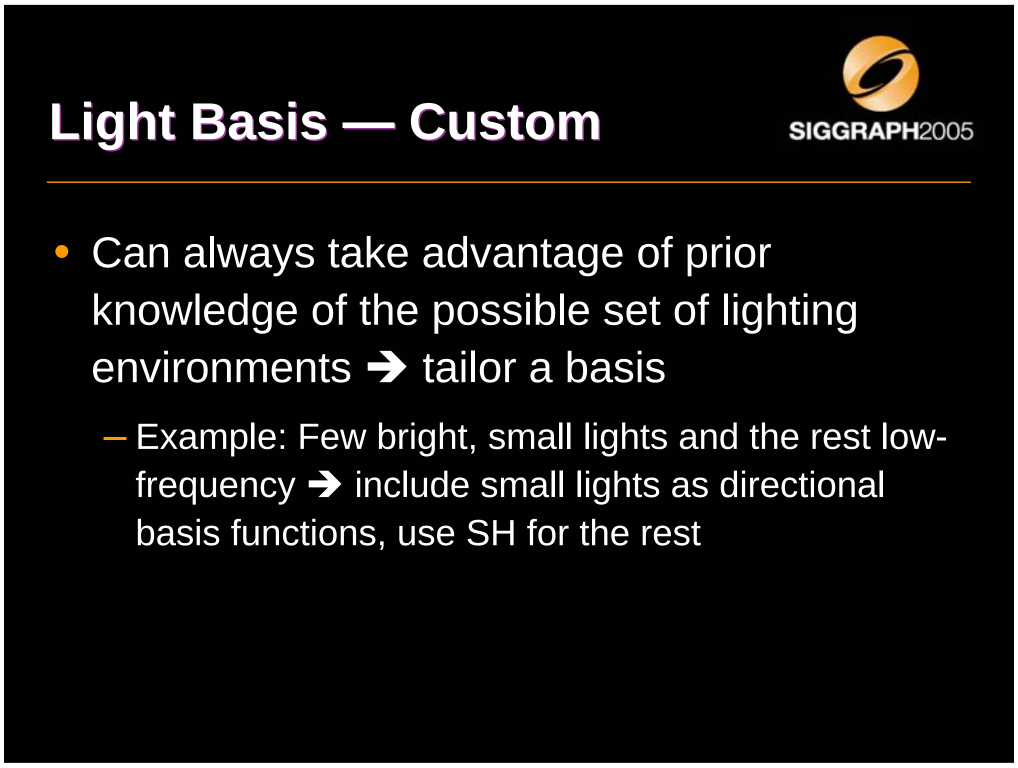
All the bad sides of Haar wavelets, but not the good one (do not project to sparse vectors).



Light Basis — Steerable

SIGGRAPH2005

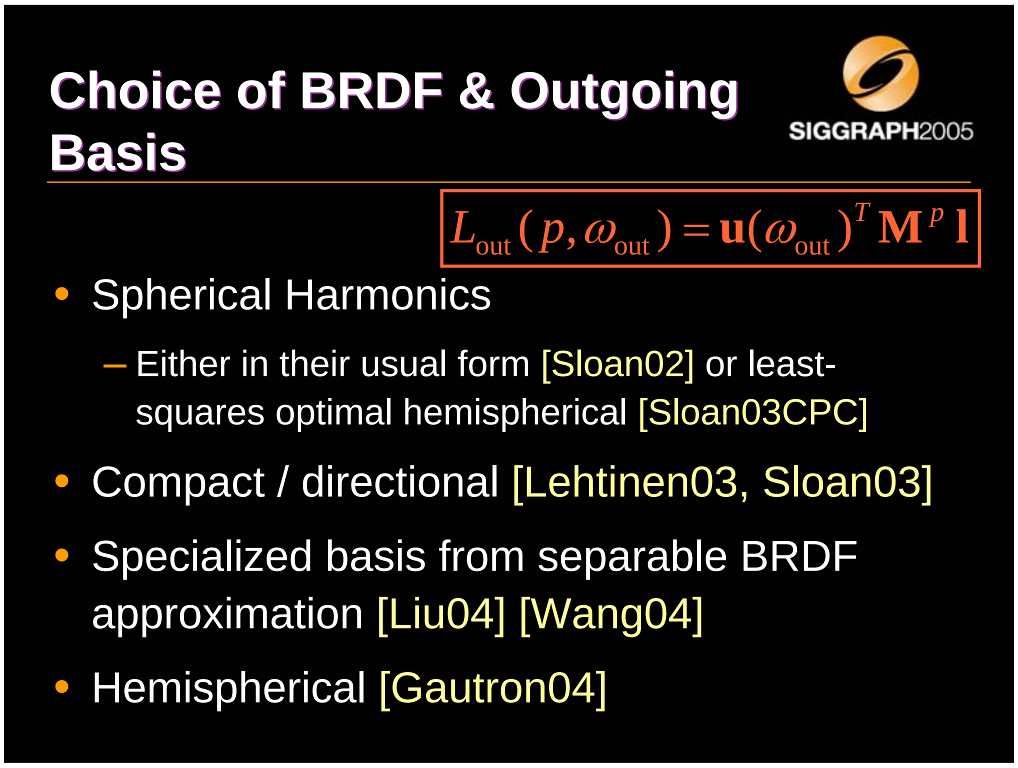
- Steerable bases are closed under rotation ➔ Good: rotation does not alias
- Bad: Not good decorrelators (inefficient approximation)



Light Basis — Custom

SIGGRAPH2005

- Can always take advantage of prior knowledge of the possible set of lighting environments ➔ tailor a basis
  - Example: Few bright, small lights and the rest low-frequency ➔ include small lights as directional basis functions, use SH for the rest



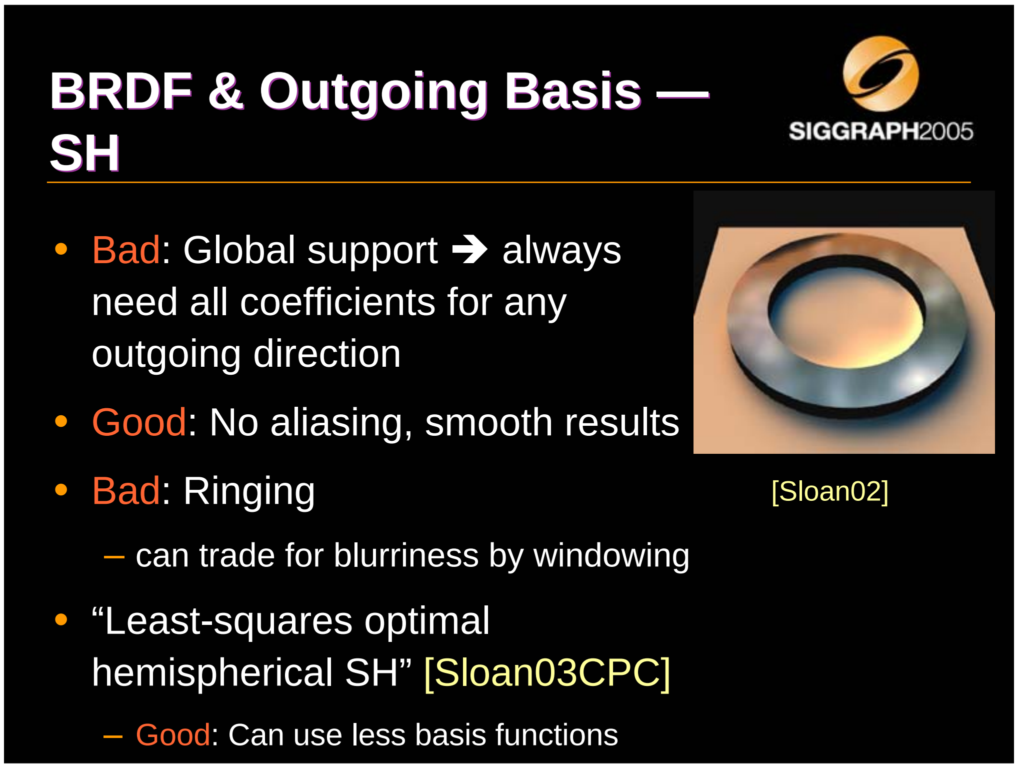
Choice of BRDF & Outgoing Basis

SIGGRAPH2005

$$L_{\text{out}}(p,\omega_{\text{out}}) = \mathbf{u}(\omega_{\text{out}})^T \mathbf{M}^p \mathbf{l}$$

- Spherical Harmonics
  - Either in their usual form [Sloan02] or least-squares optimal hemispherical [Sloan03CPC]
- Compact / directional [Lehtinen03, Sloan03]
- Specialized basis from separable BRDF approximation [Liu04] [Wang04]
- Hemispherical [Gautron04]

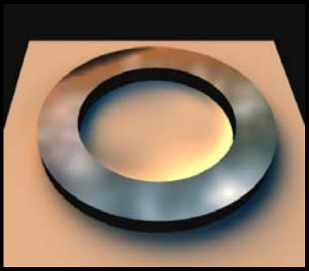
These are not all of the possibilities. For instance, wavelet bases could be employed here. The simplest case, the Haar wavelet, is not well suited for direct visualization of outgoing radiance, however; its expressive power corresponds to a piecewise constant directional basis!

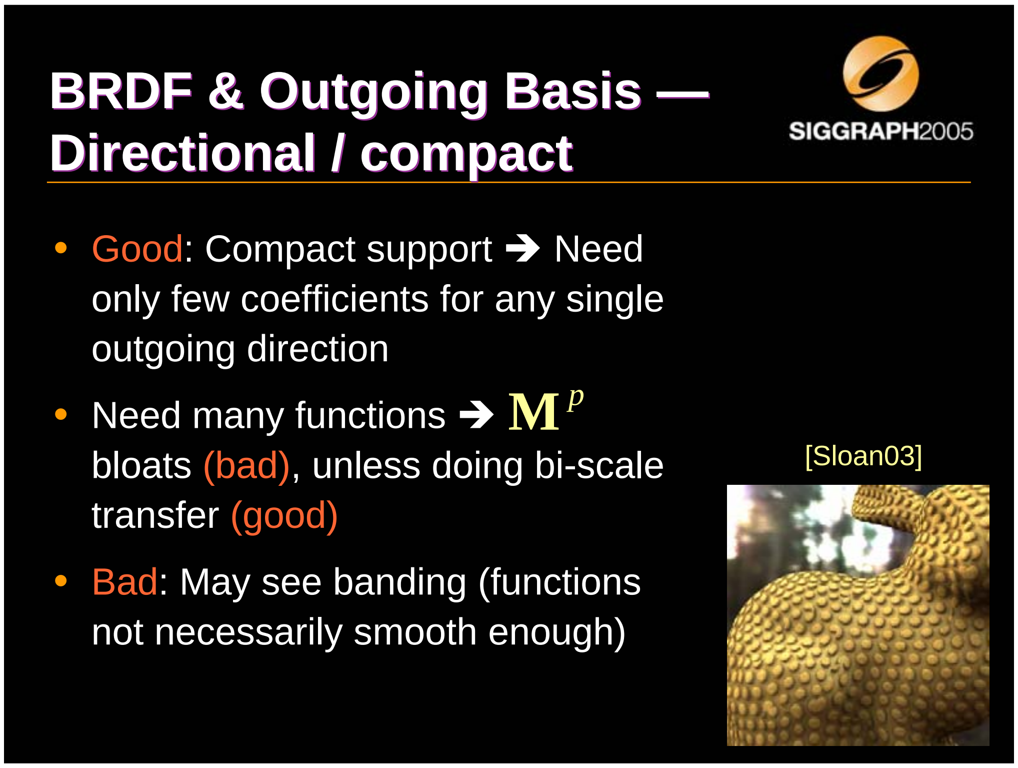


BRDF & Outgoing Basis — SH

SIGGRAPH2005

- Bad: Global support ➔ always need all coefficients for any outgoing direction
- Good: No aliasing, smooth results
- Bad: Ringing
  - can trade for blurriness by windowing
- “Least-squares optimal hemispherical SH” [Sloan03CPC]
  - Good: Can use less basis functions


[Sloan02]

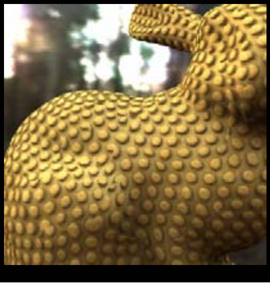


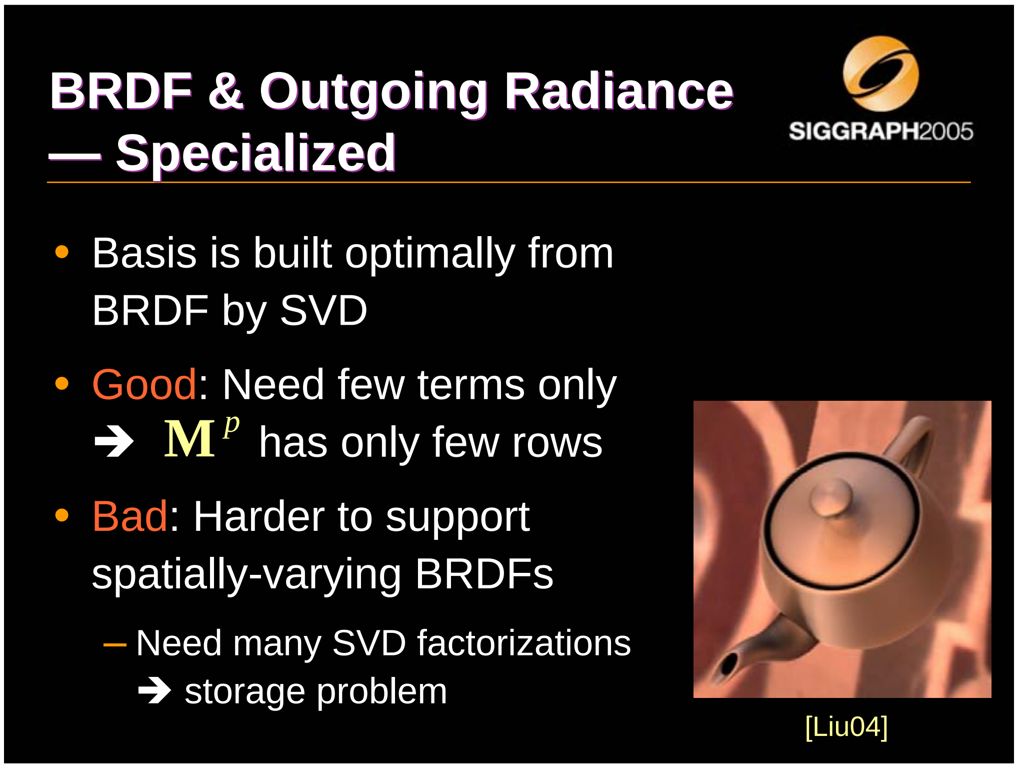
BRDF & Outgoing Basis — Directional / compact

SIGGRAPH2005

- Good: Compact support ➔ Need only few coefficients for any single outgoing direction
- Need many functions ➔ $\mathbf{M}^p$ bloats (bad), unless doing bi-scale transfer (good)
- Bad: May see banding (functions not necessarily smooth enough)

[Sloan03]

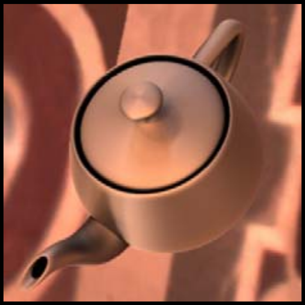




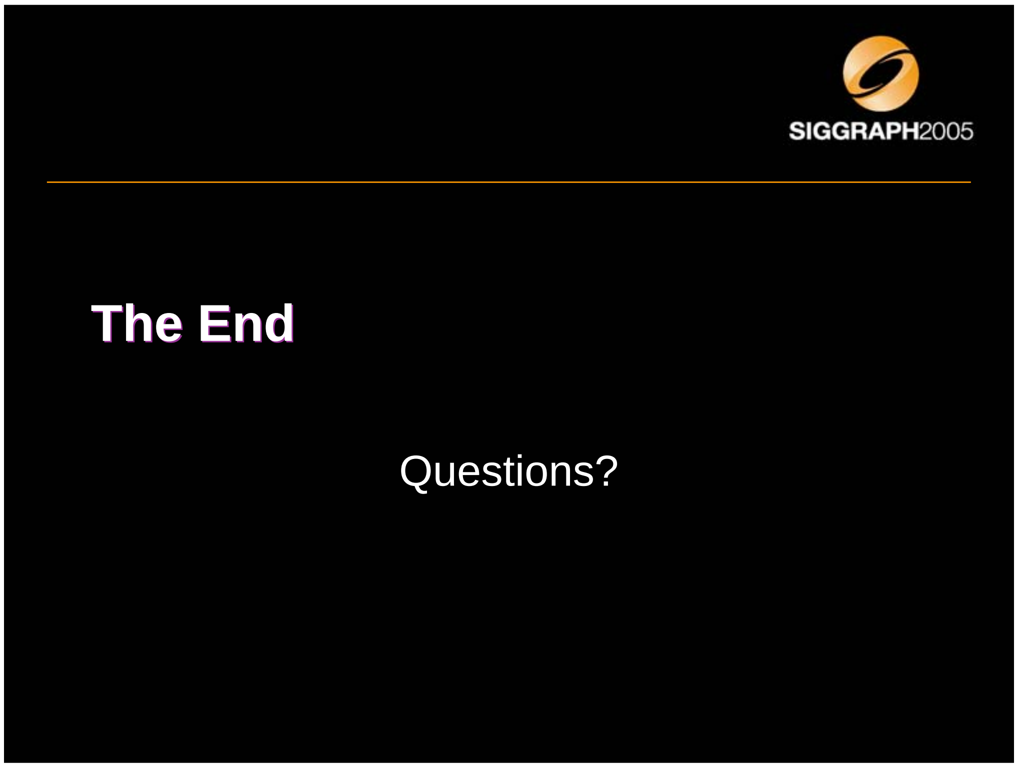
BRDF & Outgoing Radiance — Specialized

SIGGRAPH2005

- Basis is built optimally from BRDF by SVD
- Good: Need few terms only
  ➔ $\mathbf{M}^{p}$ has only few rows
- Bad: Harder to support spatially-varying BRDFs
  - Need many SVD factorizations
    ➔ storage problem



[Liu04]



SIGGRAPH2005

The End

Questions?

SIGGRAPH2005