
Environment map lighting a.k.a. Image-based lighting a.k.a. Reflection mapping

Jaroslav Křivánek, KSVI, MFF UK

Jaroslav.Krivanek@mff.cuni.cz

Acknowledgement

- Some material based on Ravi Ramamoorthi's slides available from <http://inst.eecs.berkeley.edu/~cs283/fa10>
- Make sure to check out Paul Debevec's "The Story of Reflection Mapping" at <http://www.pauldebevec.com/ReflectionMapping/>

Goal

- Real-time rendering with complex lighting, shadows, and possibly also global illumination
- Infeasible in real-time game graphics – too much computation for too small a time budget
- Approaches
 - Lift some requirements, do specific-purpose tricks
 - Environment mapping, irradiance environment maps
 - SH-based lighting
 - Split the effort
 - Offline pre-computation + real-time image synthesis
 - Baked light (light maps), pre-computed radiance transfer

Environment mapping (a.k.a. image-based lighting, reflection mapping)



Miller and Hoffman, 1984

Later, Greene 86, Cabral et al, Debevec 97, ...

CG for Game Development - J. Křivánek

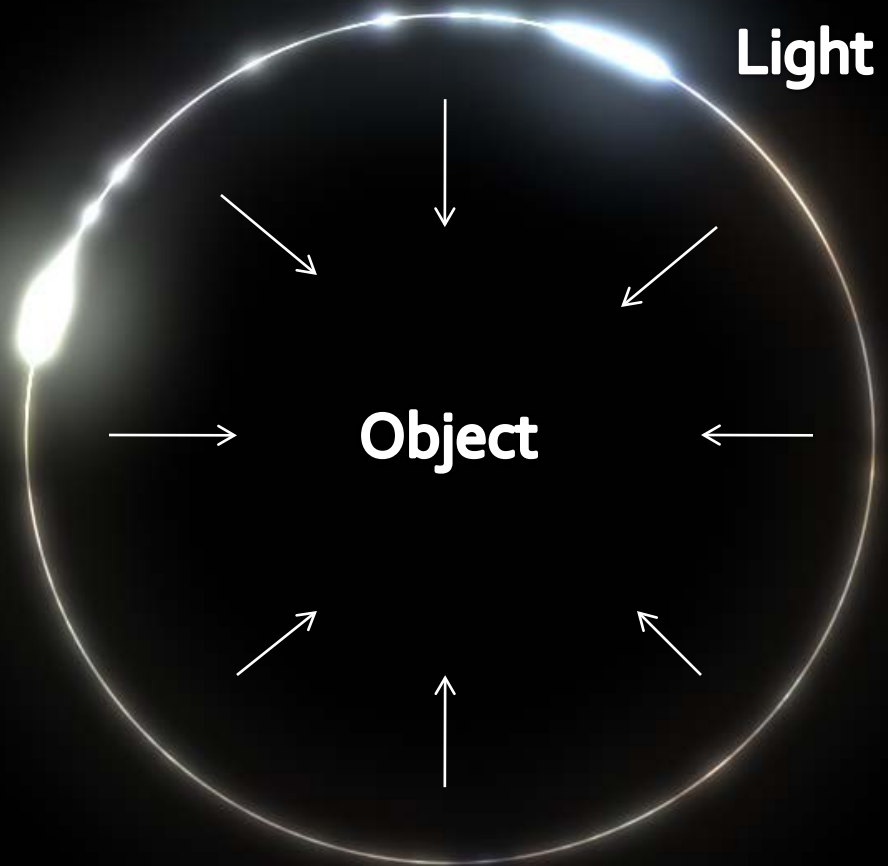
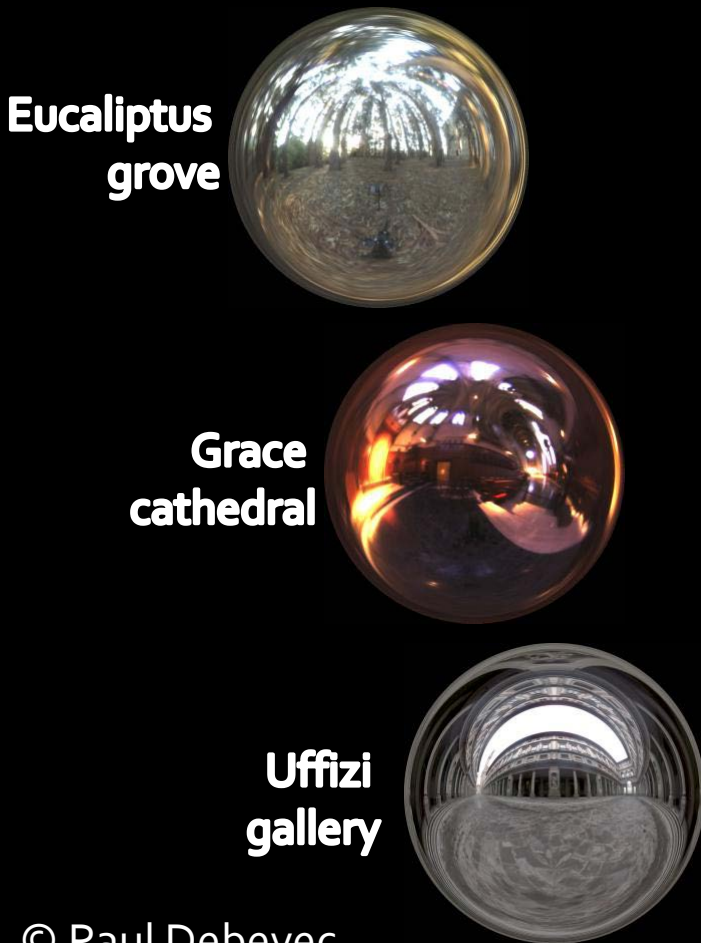
2016

Assumptions

- Distant illumination (infinite sphere around the scene)
- For real-time rendering we often assume
 - No shadowing
 - No interreflections

Image-based lighting

- Illuminating CG objects using measurements of real light (=light probes)



Point lighting

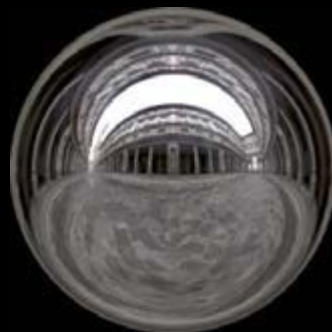
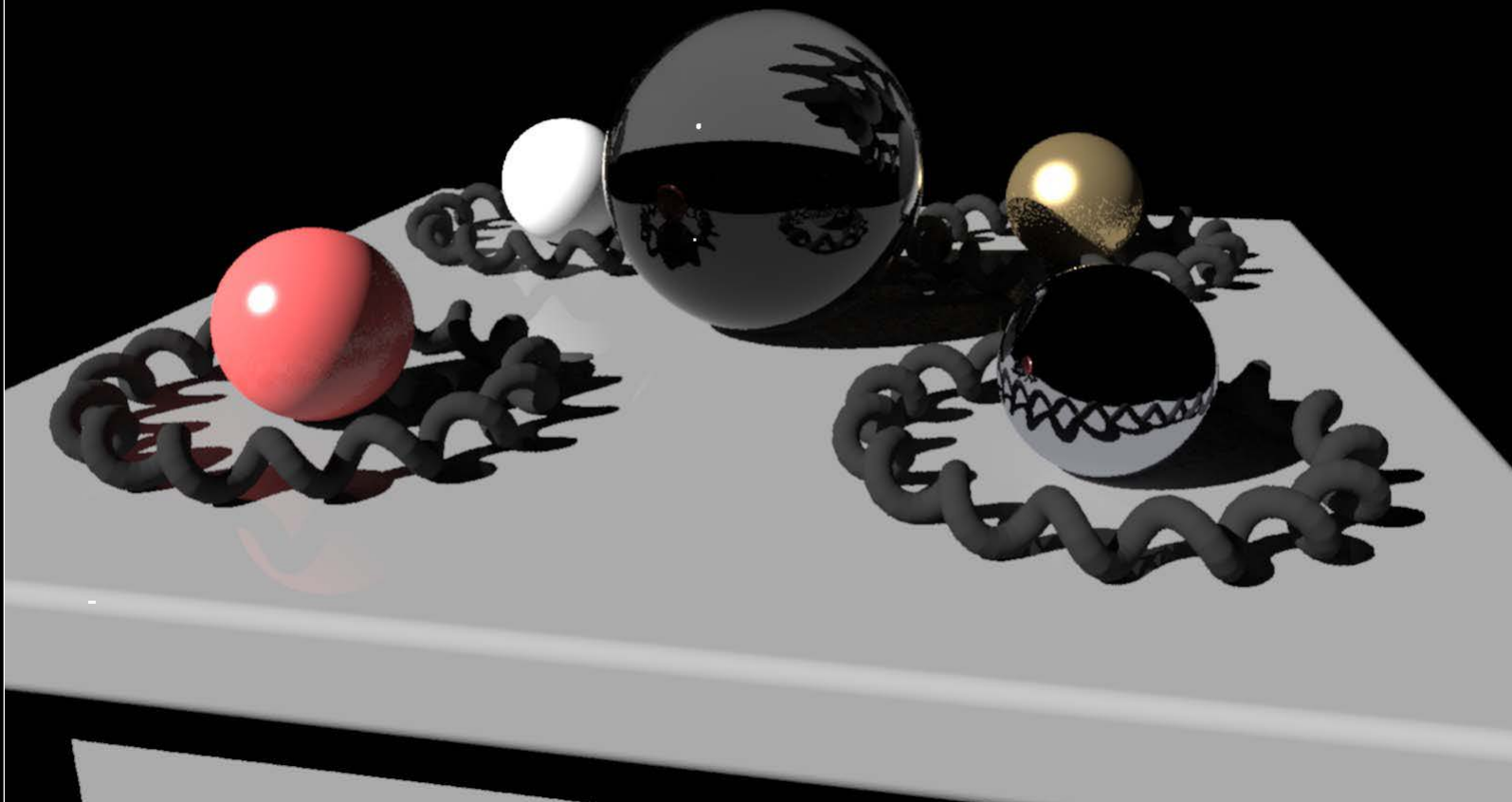


Image-based lighting

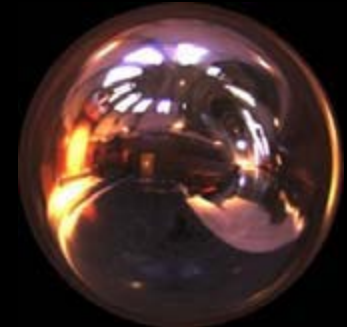


Image-based lighting

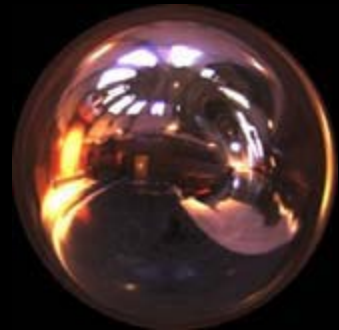
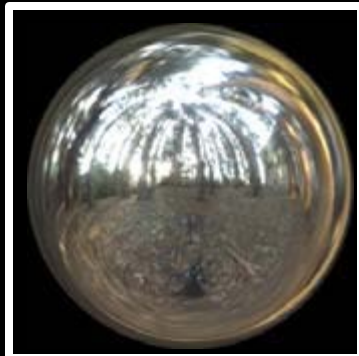


Image-based lighting

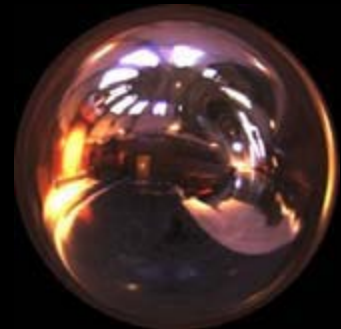
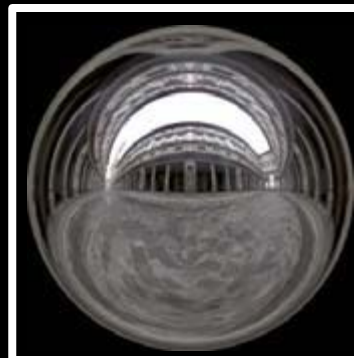
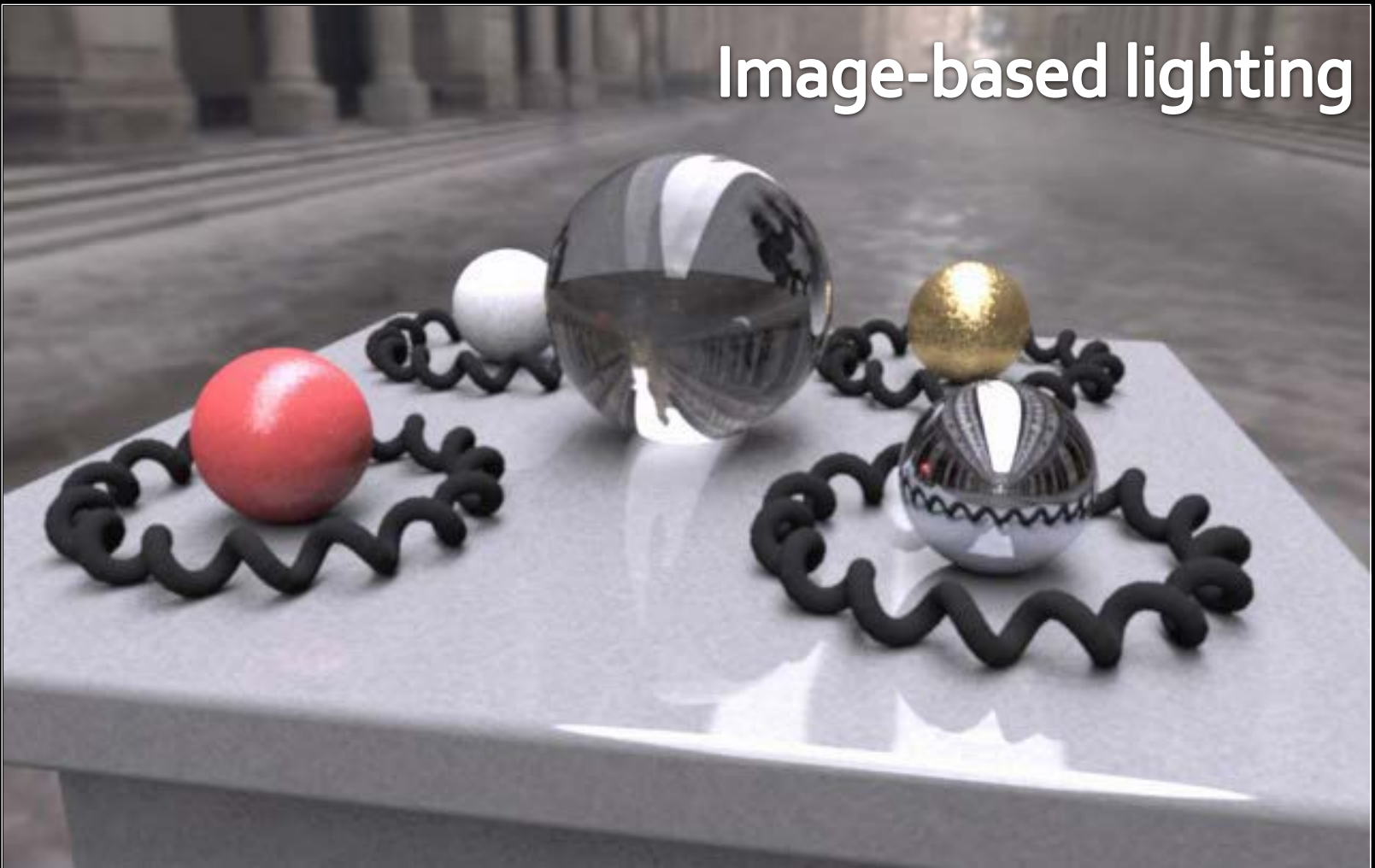
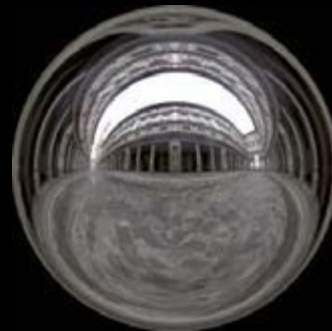
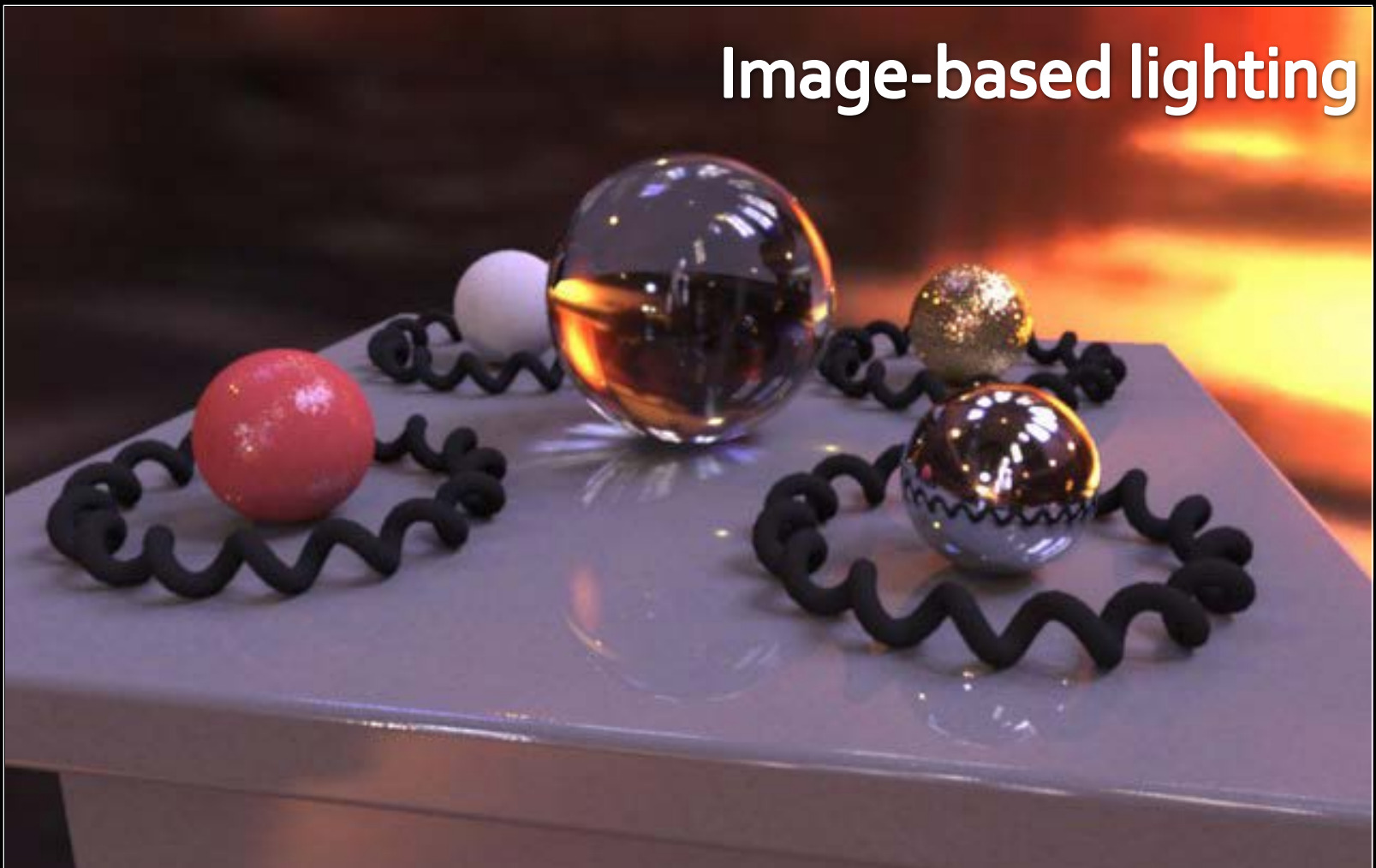


Image-based lighting



- Video

- Rendering with natural light

- <http://www.pauldebevec.com/RNL/>

- Fiat Lux

- <http://www.pauldebevec.com/FiatLux/movie/>

Mapping

Eucalyptus grove



Eucalyptus Grove Light Probe
©1999 Paul Debevec
<http://www.debevec.org/Probes>

Grace cathedral



Grace Cathedral Light Probe
©1999 Paul Debevec
<http://www.debevec.org/Probes>

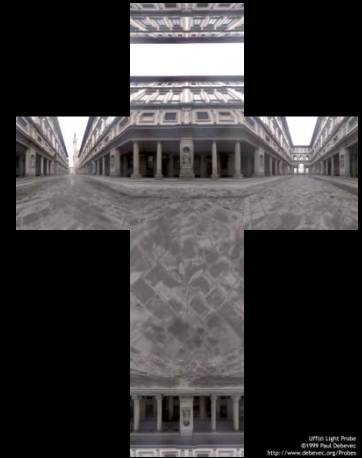
Debevec's spherical

"Latitude - longitude" (spherical coordinates)

Cube map

Mapping

Uffizi gallery



St. Peter's Cathedral



Debevec's spherical

"Latitude – longitude" (spherical coordinates)

Cube map

Mapping

- Mapping from direction in Cartesian coordinates to image UV.

```
float d = sqrt(dir.x*dir.x + dir.y*dir.y);  
float r = d>0 ? 0.159154943*acos(dir.z)/d : 0.0;  
u = 0.5 + dir.x * r;  
v = 0.5 + dir.y * r;
```



Quote from "<http://ict.debevec.org/~debevec/Probes/>"

The following light probe images were created by taking two pictures of a mirrored ball at ninety degrees of separation and assembling the two radiance maps into this registered dataset. The coordinate mapping of these images is such that the center of the image is straight forward, the circumference of the image is straight backwards, and the horizontal line through the center linearly maps azimuthal angle to pixel coordinate.

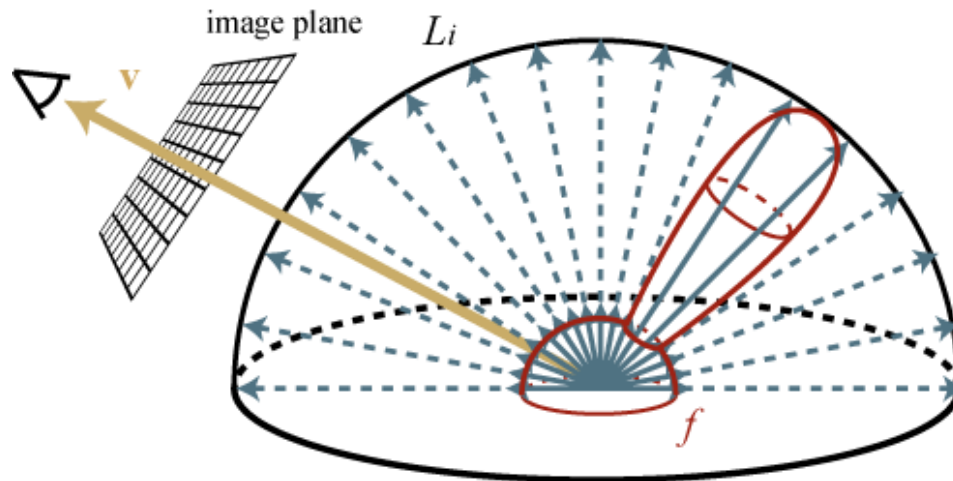
*Thus, if we consider the images to be normalized to have coordinates $\mathbf{u}=[-1,1]$, $\mathbf{v}=[-1,1]$, we have $\theta = \text{atan2}(v,u)$, $\phi = \pi * \sqrt{u*u + v*v}$. The unit vector pointing in the corresponding direction is obtained by rotating $(0,0,-1)$ by ϕ degrees around the y (up) axis and then θ degrees around the $-z$ (forward) axis. If for a direction vector in the world (D_x, D_y, D_z) , the corresponding (u,v) coordinate in the light probe image is (D_x*r, D_y*r) where $r = (1/\pi) * \text{acos}(D_z) / \sqrt{D_x^2 + D_y^2}$.*

Rendering with environment maps

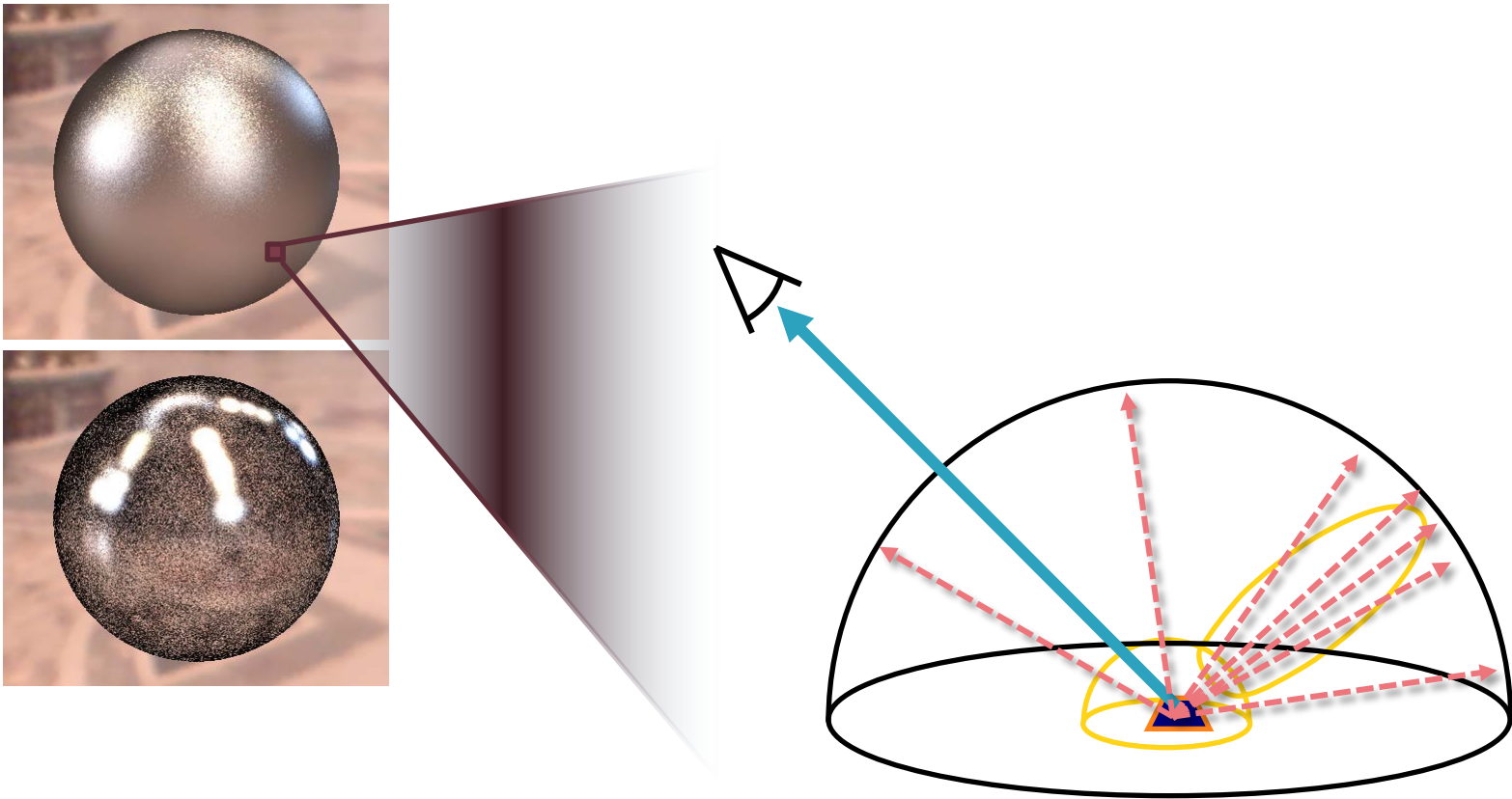
Shading due to an environment map

- Almost the same reflection equation as before
- The incident radiance L_i is due to the env. map. emission L_{em} modulated by the EM visibility V_{em}

$$L_r(\mathbf{x}, \omega_o) = \int_{H(\mathbf{x})} L_{em}(\mathbf{x}, \omega_i) \cdot V_{em}(\mathbf{x}, \omega_i) \cdot f_r(\mathbf{x}, \omega_i \rightarrow \omega_o) \cdot \cos \theta_i \, d\omega_i$$



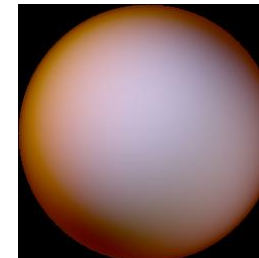
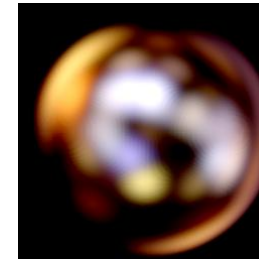
Offline rendering – Monte Carlo sampling



Real-time rendering

- MC is general, but too slow for real-time
- Real-time

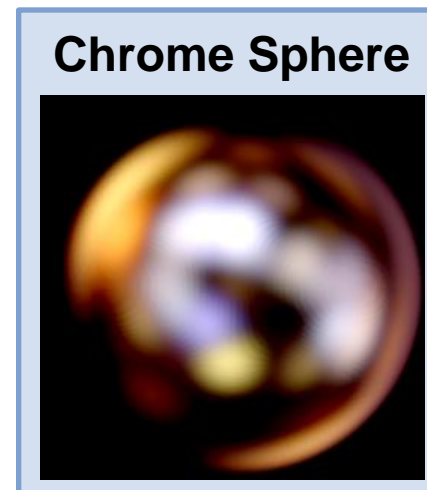
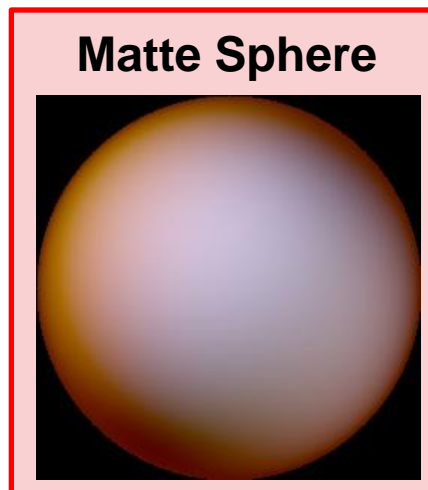
- Mirror surfaces easy
(just a texture look-up)
- What if the surface is rougher...
- Or completely diffuse?



Environment map pre-filtering

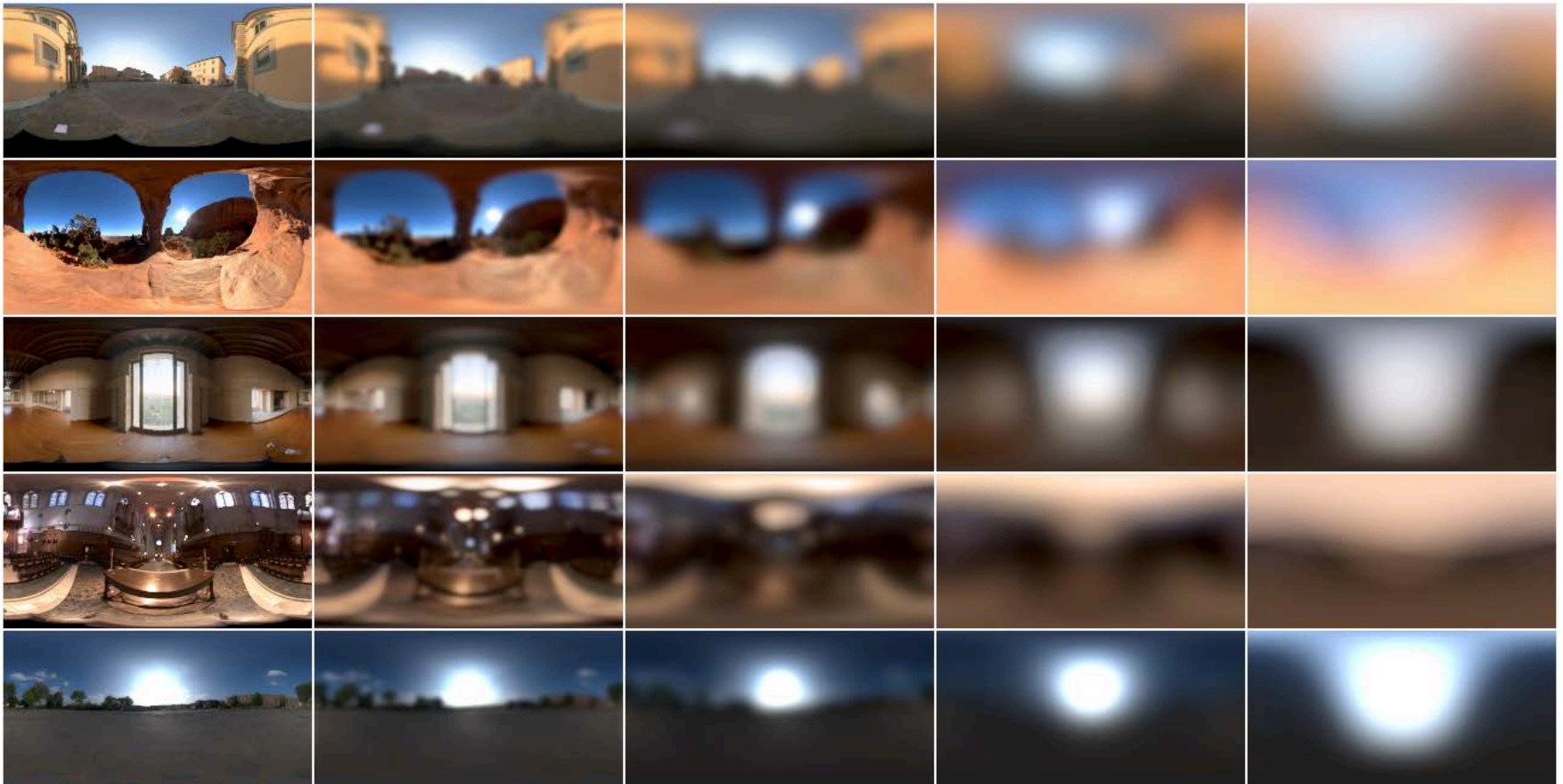
Environment map pre-filtering

- **Phong model** for rough surfaces
 - Illumination function of reflection direction R
- **Lambertian diffuse** surface
 - Illumination function of surface normal N



- Pre-filter (= blur) the EM [Miller and Hoffman, 1984]
 - **Irradiance (indexed by N)** and **Phong (indexed by R)**

Environment map pre-filtering

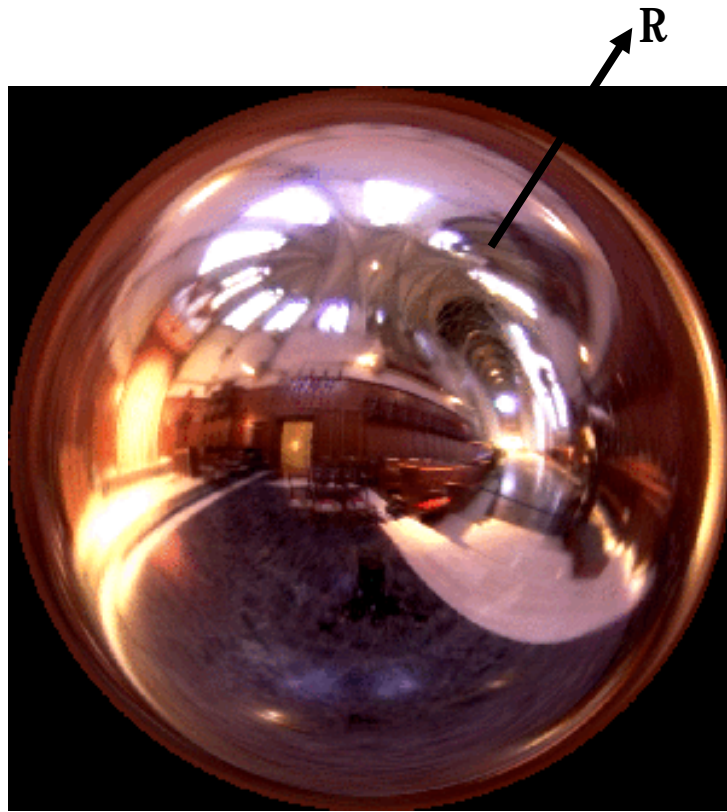


Environment map pre-filtering

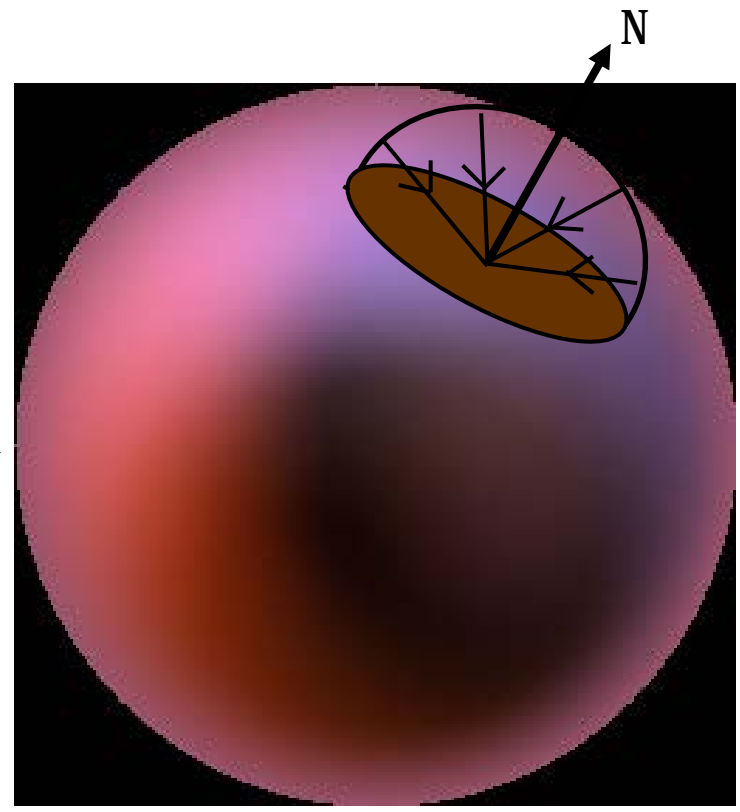
- Can't do dynamic lighting
 - Slow blurring in pre-process

Spherical harmonics-based irradiance environment maps

Spherical harmonics-based irradiance environment maps



Incident Radiance
(Illumination Environment Map)



Irradiance Environment Map

- **Diffuse (Lambertian) surfaces only!!!**

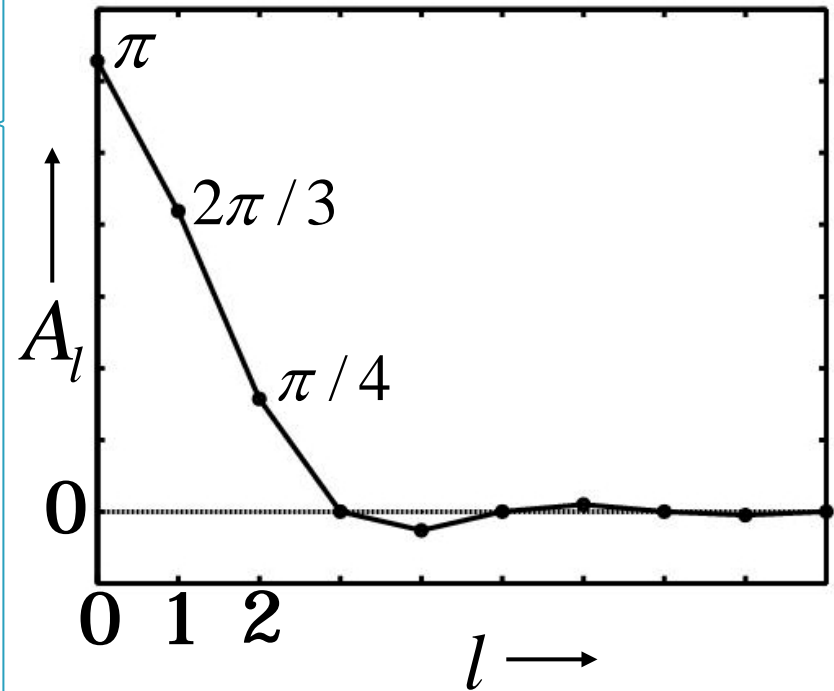
Analytic irradiance formula in SH basis

Lambertian surface acts like low-pass filter

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

SH coefficients of the irradiance EM

SH coefficients of the original EM



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

[Ramamoorthi and Hanrahan 01]

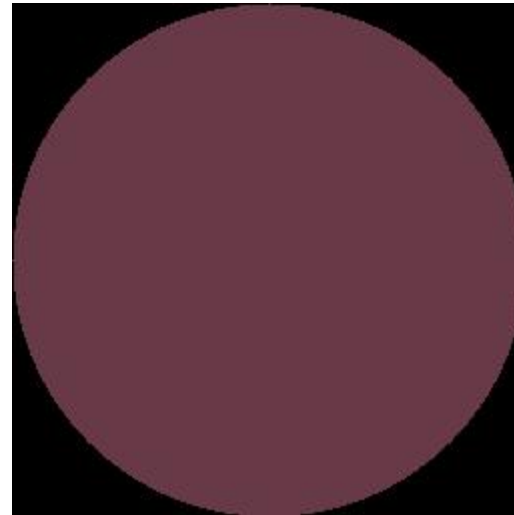
[Basri and Jacobs 01]

9-parameter approximation

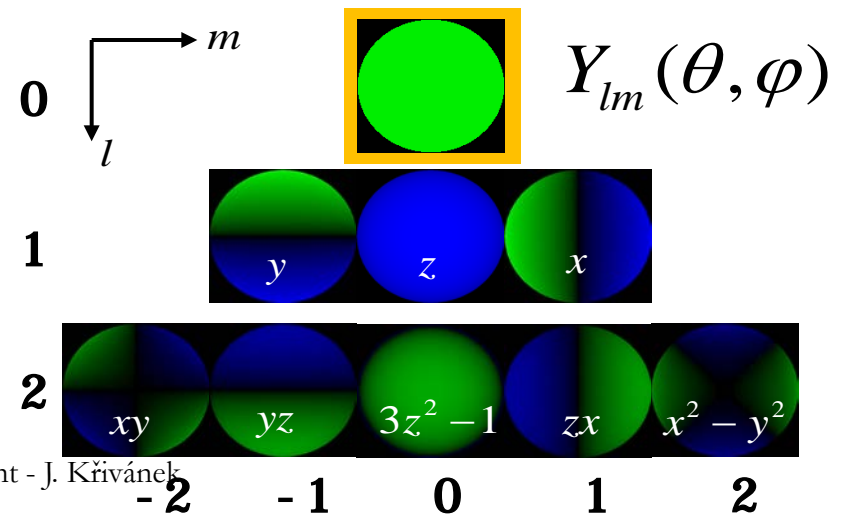
Exact image



Order 0
1 term



RMS error = 25 %



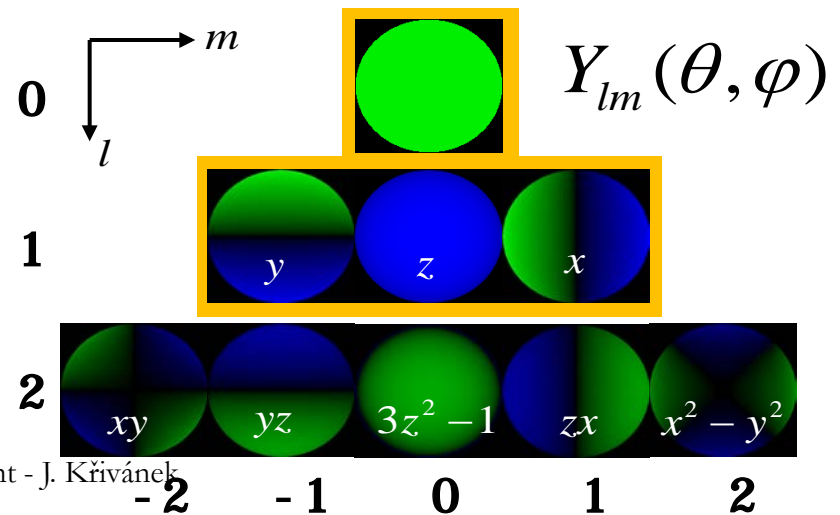
9-parameter approximation

Exact image



Order 1
4 terms

RMS Error = 8%



9-parameter approximation

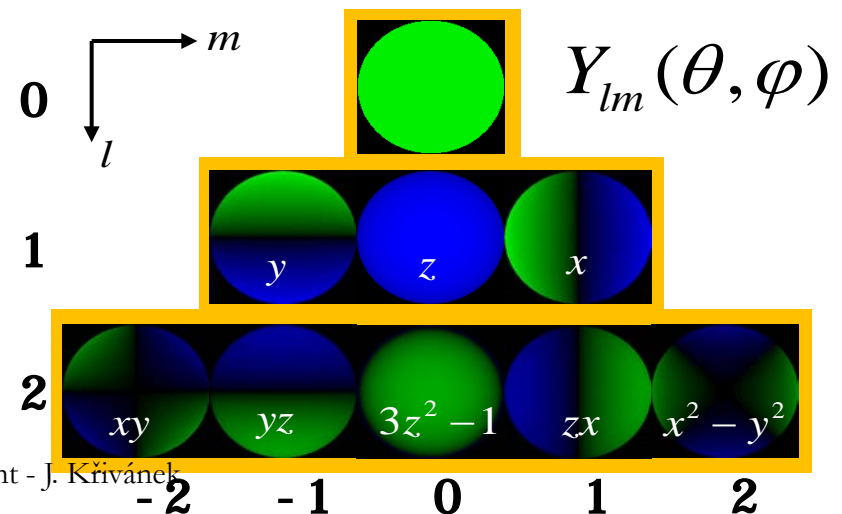
Exact image



Order 2
9 terms

RMS Error = 1%

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



Real-Time Rendering

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

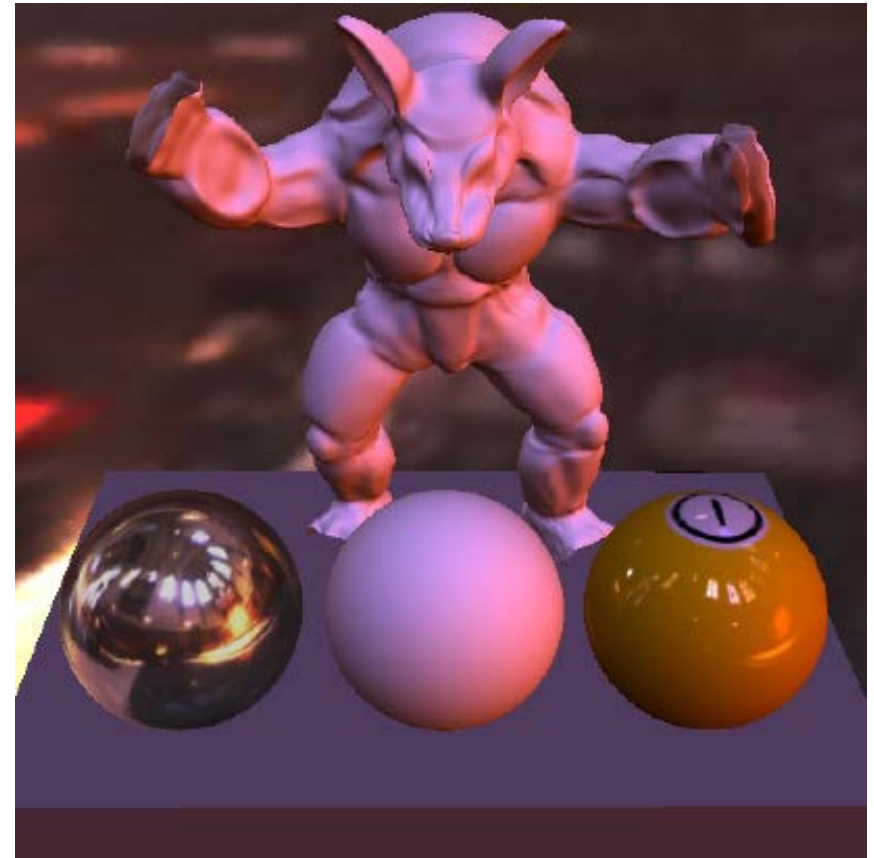
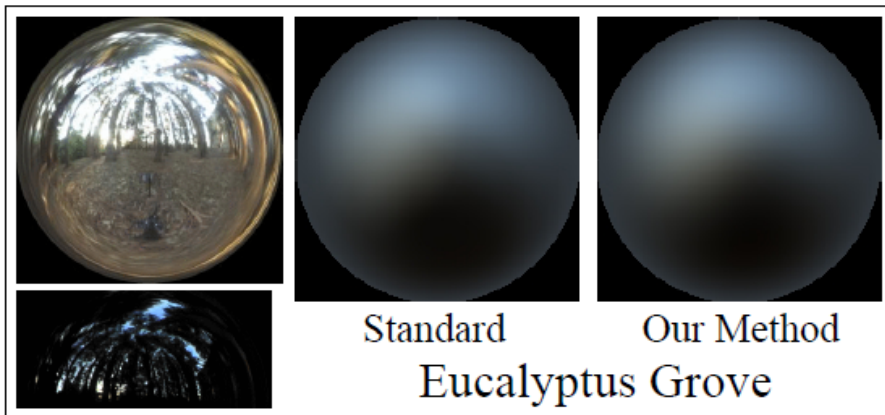
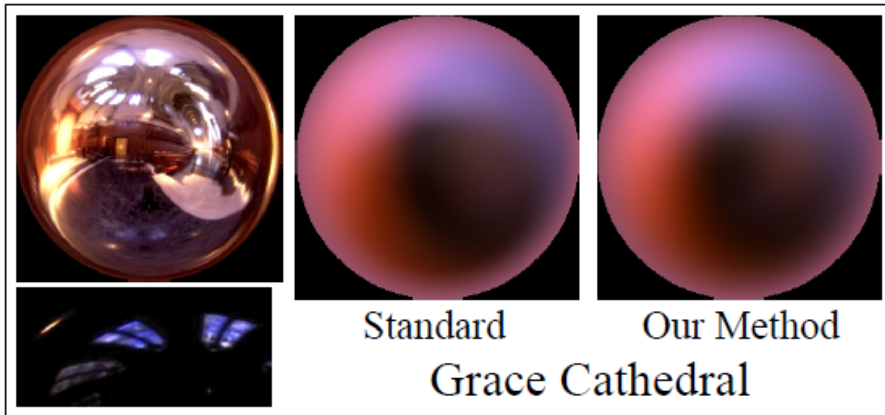
- Can be encoded in a 4x4 matrix and evaluated as above
- Simple procedural rendering method (no textures)
 - Requires only matrix-vector multiply and dot-product
 - In software or NVIDIA vertex programming hardware
- Widely used in Games (AMPED for Microsoft Xbox), Movies (Pixar, Framestore CFC, ...)

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

Algorithm

- **Preprocess** (whenever the EM changes)
 - Project the EM onto 9 SH bases functions
 - Calculate the matrix M from previous slide
 - Upload to the GPU
- During **rendering** in fragment shader
 - Fetch surface normal and the matrix M
 - Exec the code from previous slide to get the diffuse irradiance
 - Multiply by diffuse texture to get final diffuse color due to the EM

SH-based irradiance environment maps



- **Video – Ramamoorthi & Hanrahan 2001**
 - <http://graphics.stanford.edu/videos/envmap/>
- **Further information**
 - http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter10.html

Spherical harmonics-based arbitrary BRDF shading

SH-based shading of arbitrary BRDFs

■ Motivation

- Irradiance EM's could only handle diffuse surfaces
- Can we use SH for shading surfaces with arbitrary (even anisotropic) BRDFs
 - Yes

SH-based shading of arbitrary BRDFs

- [Kautz et al. 2003]
- Arbitrary, dynamic env. map
- Arbitrary BRDF
- No shadows

- SH representation

- Environment map (one set of coefficients)
- Scene BRDFs (one coefficient vector for each discretized viewing direction)






(a) point light

(b) glossy

(c) anisotropic

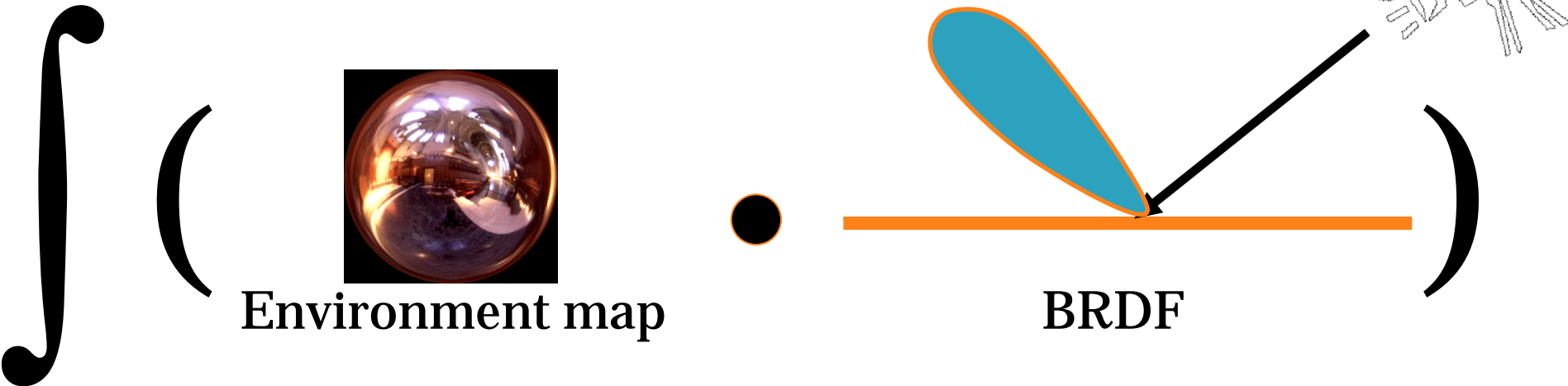


projected lighting environment			
	$n=9$	$n=25^*$	$n=49$

SH-based shading of arbitrary BRDFs

- Rendering: for each vertex / pixel, do

$$L_r(\mathbf{x}, \omega_o) = \int_{H(\mathbf{x})} L_{em}(\mathbf{x}, \omega_i) \cdot \cancel{V_{em}(\mathbf{x}, \omega_i)} \cdot f_r(\mathbf{x}, \omega_i \rightarrow \omega_o) \cdot \cos \theta_i \, d\omega_i$$



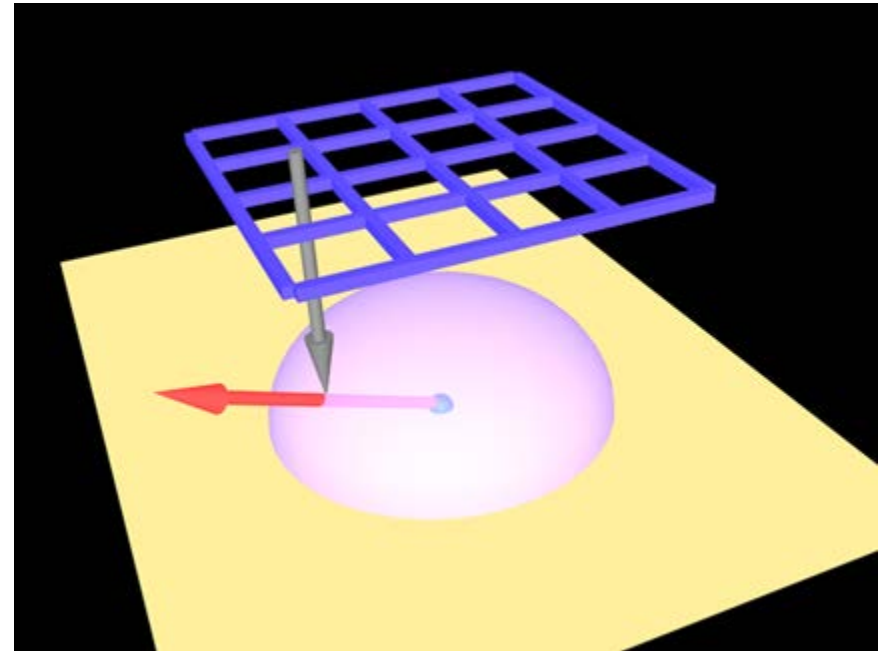
= coeff. dot product

$$L_o(\omega_o) = \sum \lambda_i f_i(\omega_o)$$

SH-based shading of arbitrary BRDFs

■ BRDF Representation

- ❑ BRDF coefficient vector $[f_i]_i$ for a given ω_o , looked up from a texture (use e.g. paraboloid mapping to map ω_o to a texture coordinate)
- ❑ BRDF coefficients pre-computed for all scene BRDFs (SH projection)



SH-based shading of arbitrary BRDFs

- BRDF is in local frame
- Environment map in global frame
- Need coordinate frame alignment -> **SH rotation**

- SH closed under rotation
 - Rotation matrix
 - Fastest known procedure is the *zxzxz*-decomposition [Kautz et al. 2003]

$$R_{SH} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \dots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \dots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \dots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Algorithm

- Preprocess
 - For each BRDF in the scene
 - For each viewing direction
 - Project the BRDF lobe onto SH basis (49-100 coefficients)
 - Whenever the EM changes
 - Project the EN onto SH basis (as many coefficients as for the “sharpest”, i.e. most specular BRDF)
- Rendering in fragment shader
 - Fetch SH coefficients for the EM
 - Fetch SH coefficients for the BRDF (current viewing direction)
 - Bring the BRDF representation to the global frame using SH rotation
 - Calculate the dot product of coefficients = final pixel color

SH-based shading of arbitrary BRDFs



Figure 3: *Brushed metal head in various lighting environments.*



(a) *varying exponent*

(b) *varying anisotropy*

Figure 4: *Spatially-Varying BRDFs.*

Filtered importance sampling

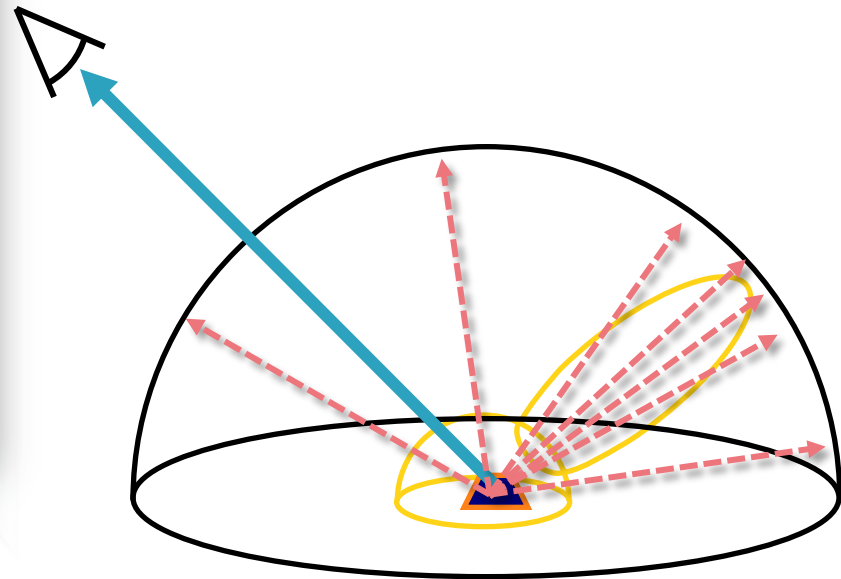
Filtered importance sampling

- Arbitrary BRDF shading
 - no SH needed
 - BRDFs can be dynamic (used for material design)
- References
 - Colbert and Křivánek 2006
 - http://http.developer.nvidia.com/GPUGems3/gpugems3_ch20.html
 - Practical implementation
 - Křivánek and Colbert 2008, EGSR
 - Theory
- Video
 - <https://www.youtube.com/watch?v=-WTOGg3M0A>

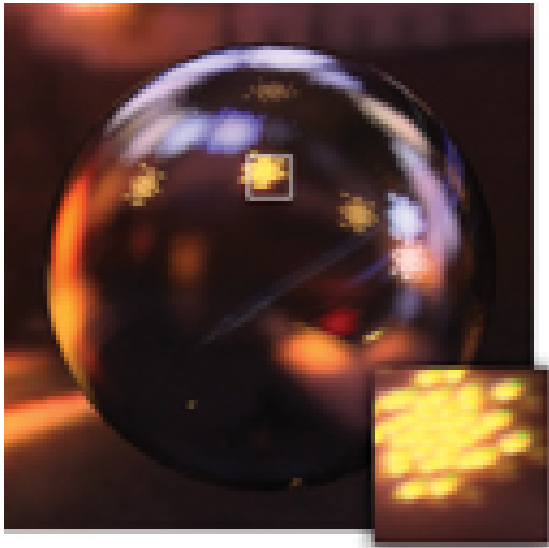
Filtered importance sampling



- Monte Carlo BRDF importance sampling
 - leaves noise

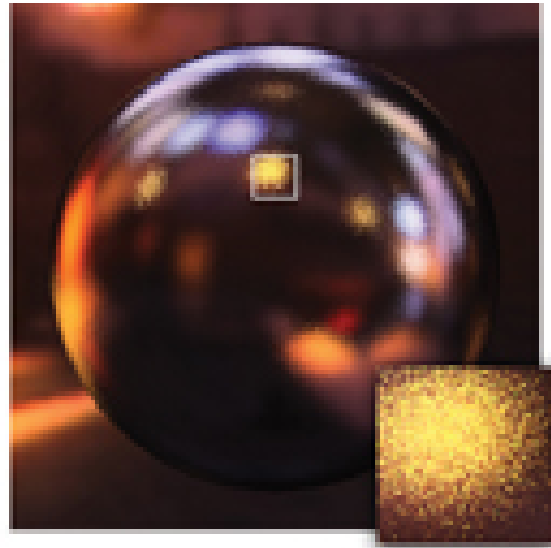


Filtered importance sampling



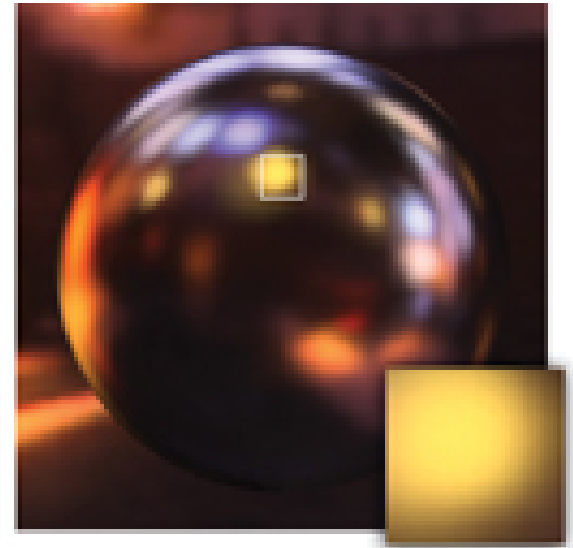
(a)

Deterministic
sampling



(b)

Monte Carlo
sampling

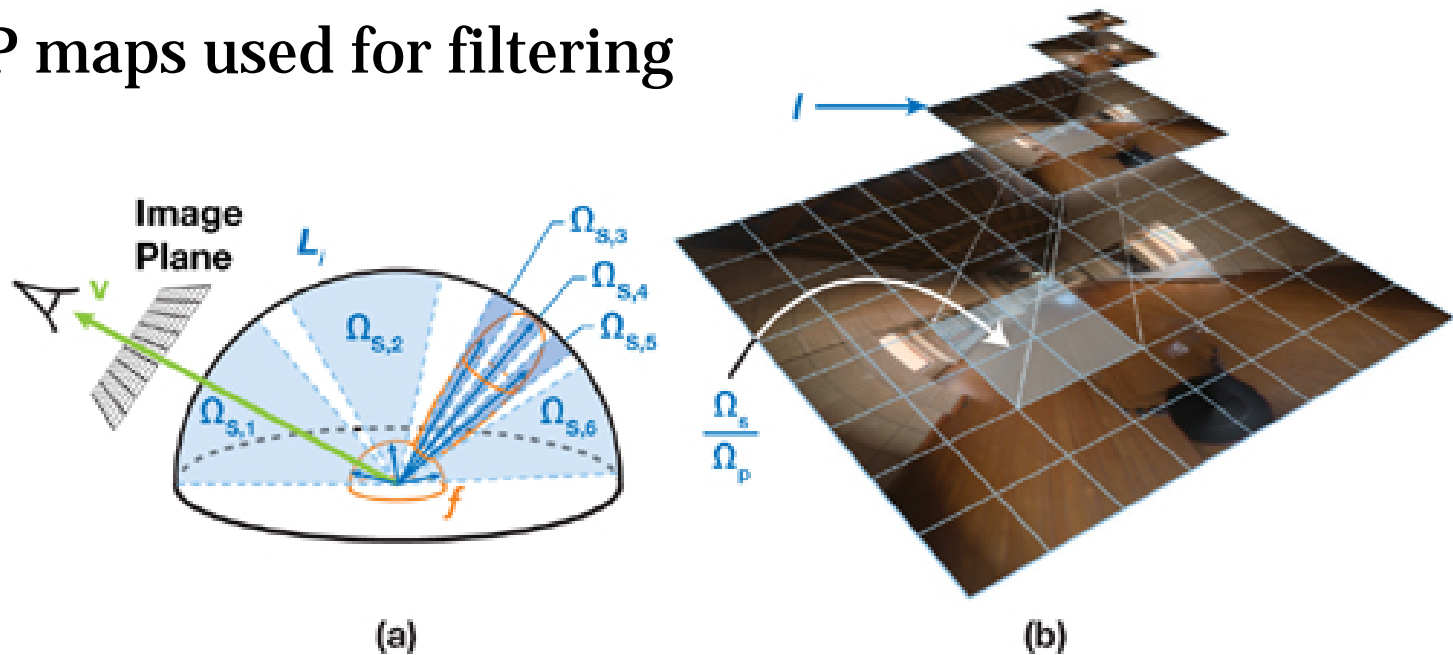


(c)

**Filtered
importance sampling**

Filtered importance sampling

- Filter width depends on probability of sampling a given direction
 - Narrow filter in the main BRDF lobe
 - Wide filter outside the lobe
- MIP maps used for filtering

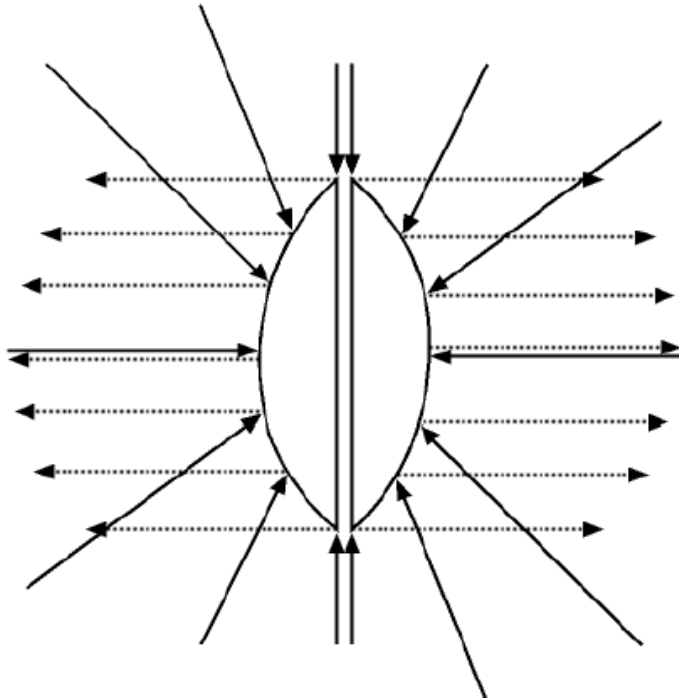


(a)

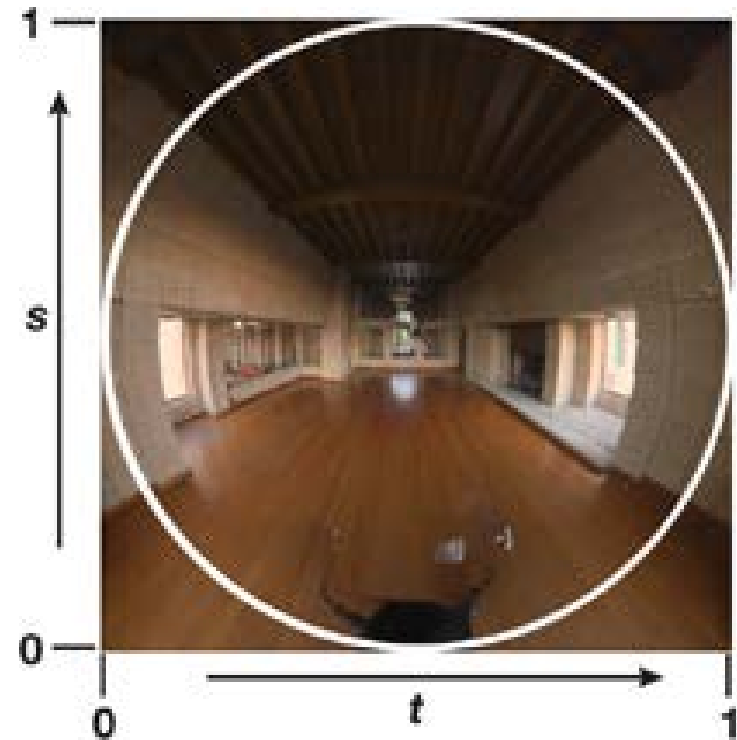
(b)

Filtered importance sampling

- Dual paraboloid mapping used to represent envmaps
 - Fast to look up, fairly low distortion



$$\text{texcoord}.st = \frac{\text{direction}.xy}{\text{direction}.z + 1}$$



(a)

Filtered importance sampling – Algorithm

- Preprocess
 - Convert EM into dual paraboloid map
 - Create MIP map for the two halves of the EM
 - Pregenerate a low-discrepancy set of “random” tuples to be used for BRDF sampling
- Rendering in fragment shader
 - for $i = 1$ to N
 - Generate a direction with BRDF importance sampling using pregenerated random tuple[i]
 - Calculate the probability density (pdf) for that direction
 - Based on the BRDF, determine the MIP map level
 - Look-up the EM, add to the average over all samples

Filtered importance sampling

- Used in substance painter

□ Check out video: https://www.youtube.com/watch?v=-fpW9C5il_U



Environment mapping summary

- Very popular for interactive rendering
- Extensions handle complex materials
- Limited to distant lighting assumption