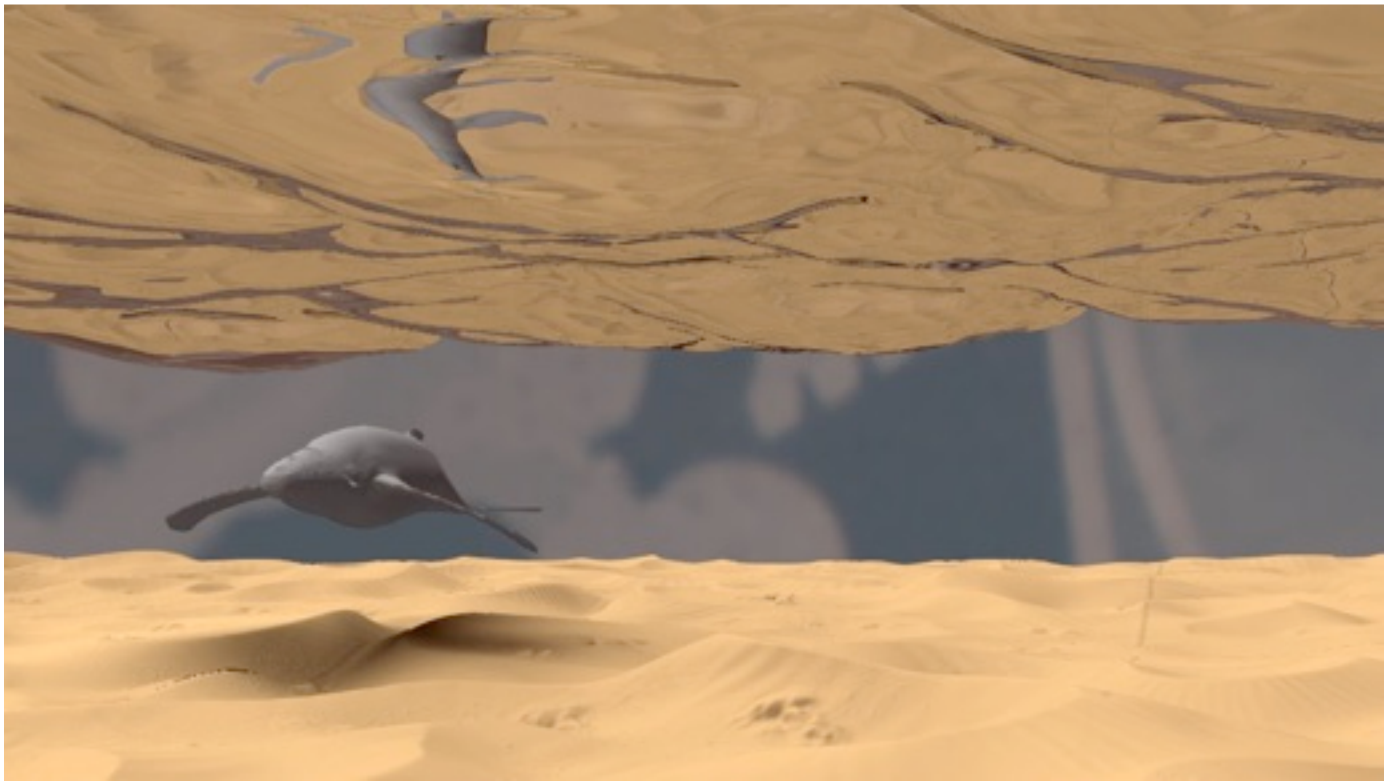
Practical Shading of Height Fields and Meshes using Spherical Harmonics Exponentiation

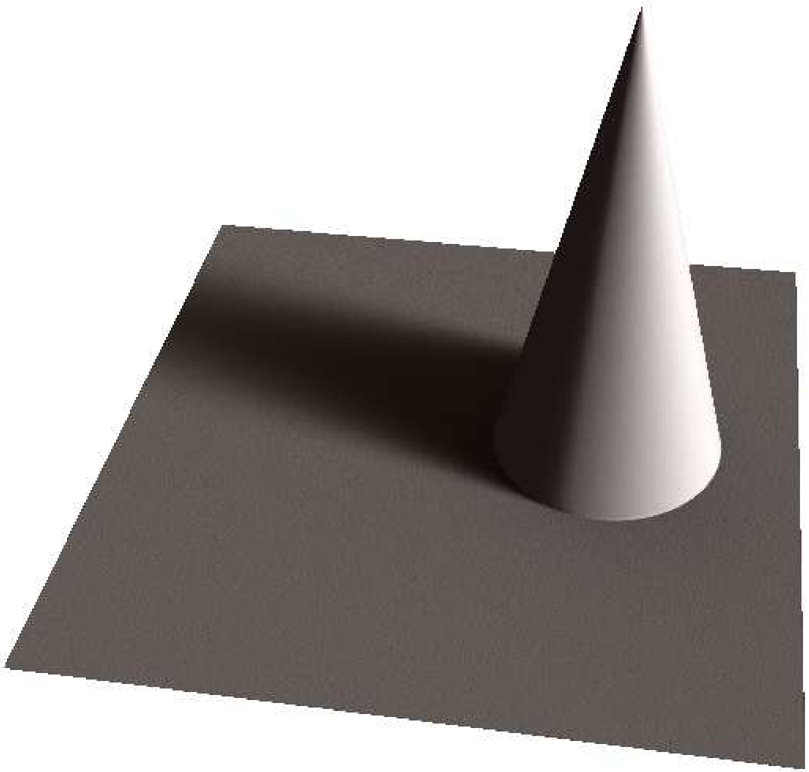


Aude Giraud

Derek Nowrouzezahrai

Université de Montréal

Goals & Motivation

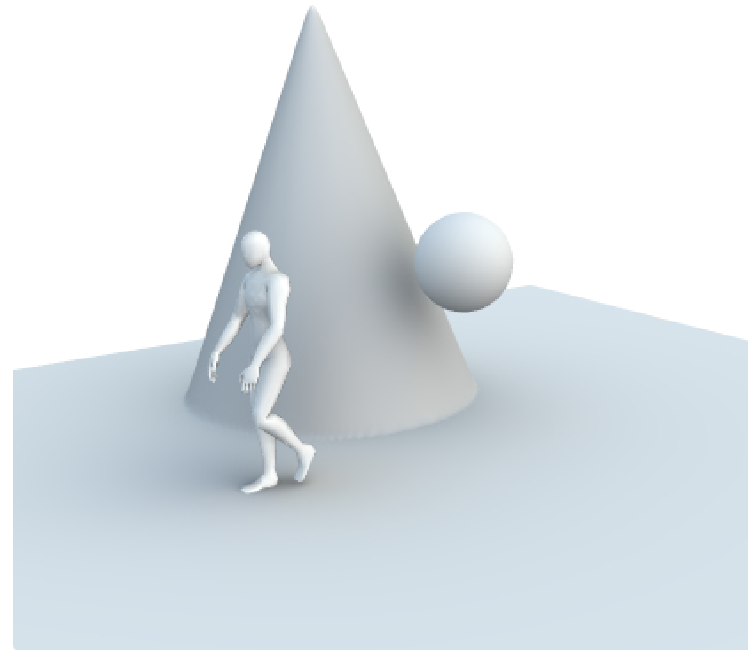


[SN08]

[RWS*06;SGNS07]

Goals & Motivation

[RWS*06;SGNS07]



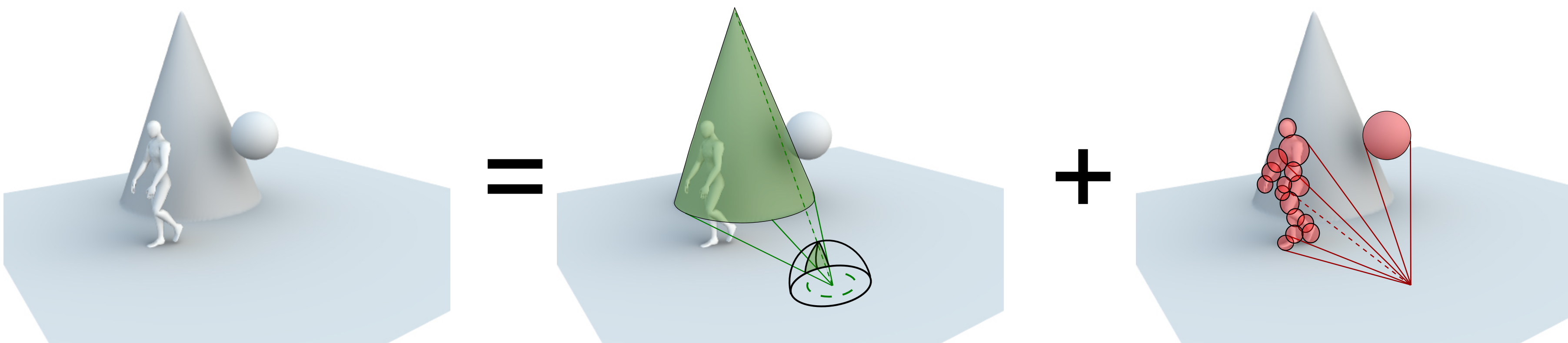
Our results

Contributions

- unifying SH exponentiation on HFs and meshes
  - dynamic geometry and HF visibility (no precomputation)
  - diffuse and glossy BRDFs in log SH

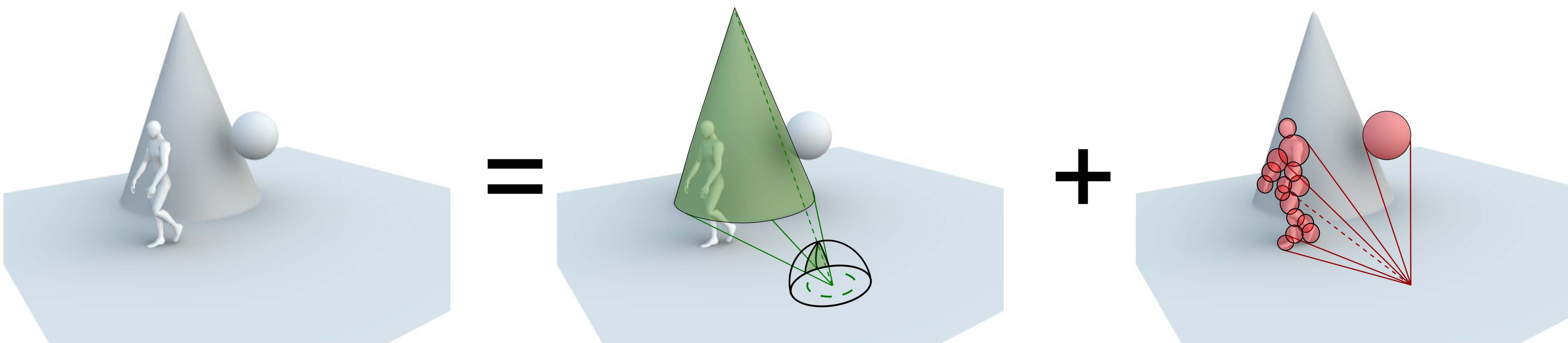
Contributions

- unifying SH exponentiation on HFs and meshes
  - dynamic geometry and HF visibility (no precomputation)
  - diffuse and glossy BRDFs in log SH



Contributions

- unifying SH exponentiation on HFs and meshes
  - dynamic geometry and HF visibility (no precomputation)
  - diffuse and glossy BRDFs in log SH



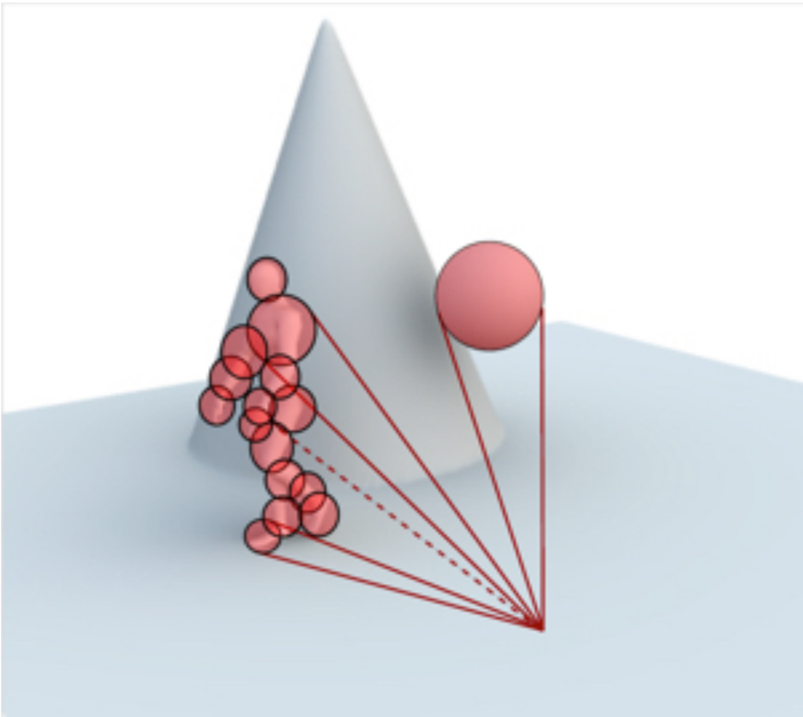
- real-time performance and simple implementation
- limitation: only ***soft direct illumination***
- applications:
  - landscape rendering (flight simulators, mapping/navigation)
  - interactive gaming

Accumulating Log-SH Visibility

Given spherical log-SH visibility for

Accumulating Log-SH Visibility

Given spherical log-SH visibility for

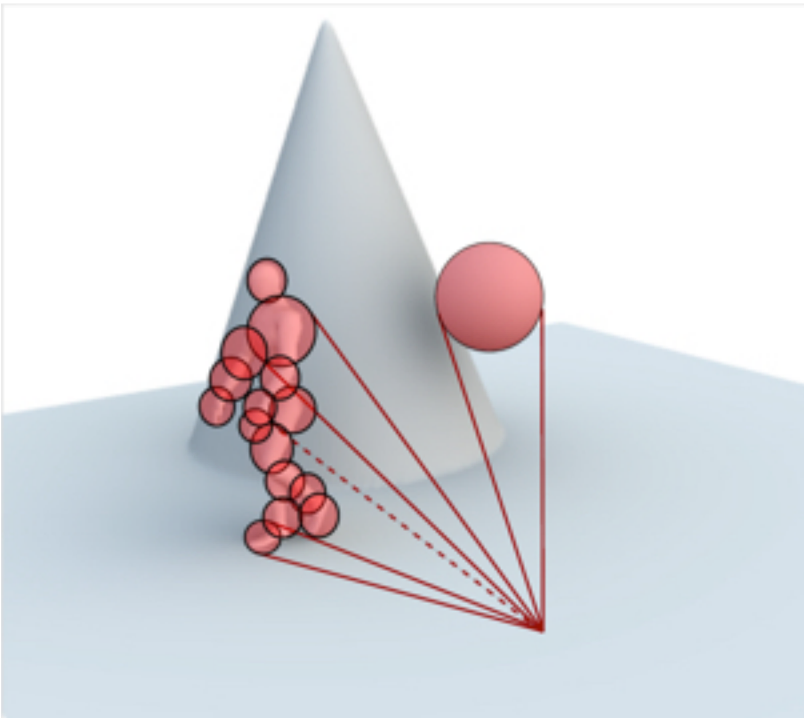


dynamic blocker "meshes"

$$\{\mathbf{v}_{\log}^{0}, \mathbf{v}_{\log}^{1}, \cdots, \mathbf{v}_{\log}^{\text{B-1}}\}$$

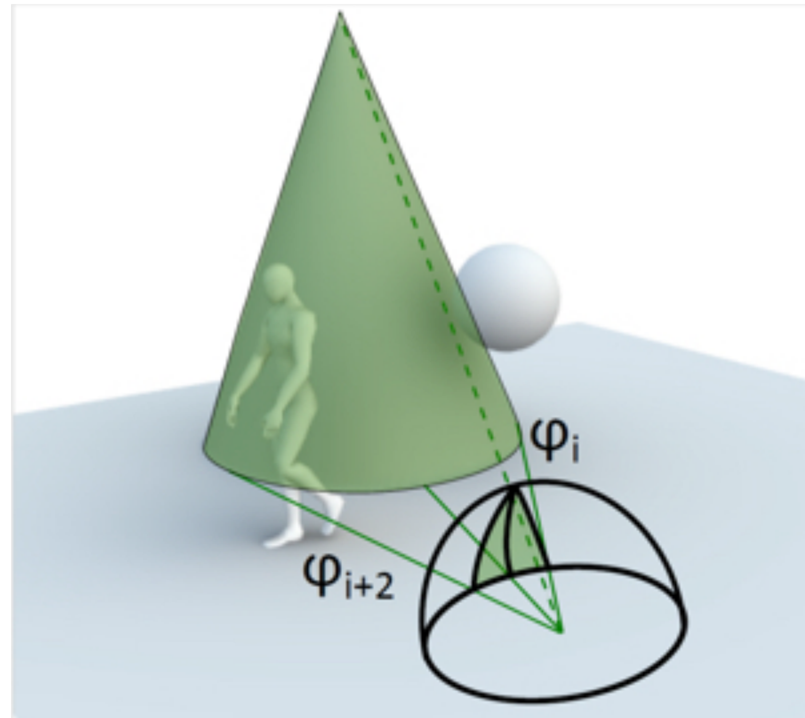
Accumulating Log-SH Visibility

Given spherical log-SH visibility for



dynamic blocker “meshes”

$$\left\{\mathbf{v}_{\log}^{0}, \mathbf{v}_{\log}^{1}, \cdots, \mathbf{v}_{\log}^{\text{B-1}}\right\}$$

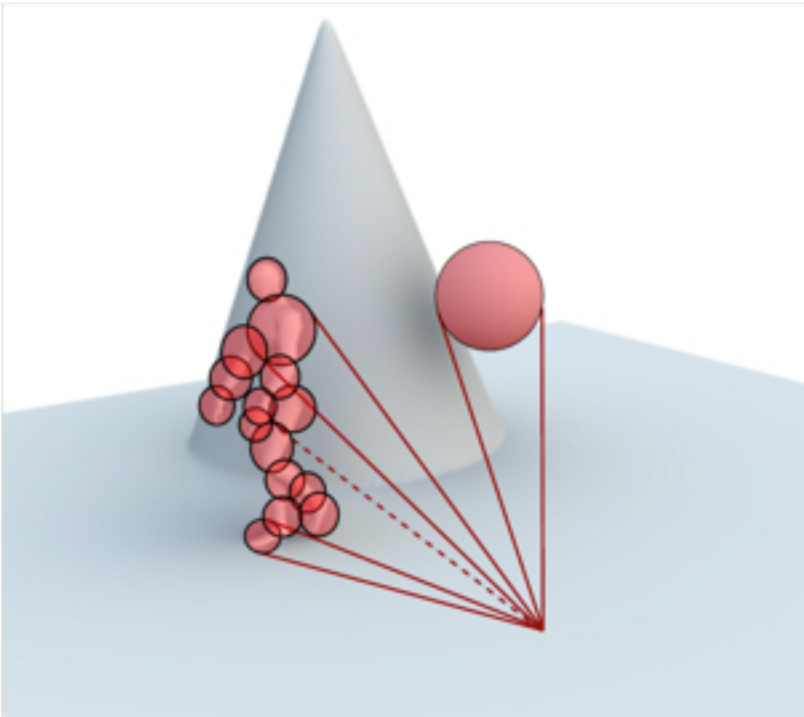


dynamic height field geometry

$$\mathbf{v}_{\log}^{\text{HF}}$$

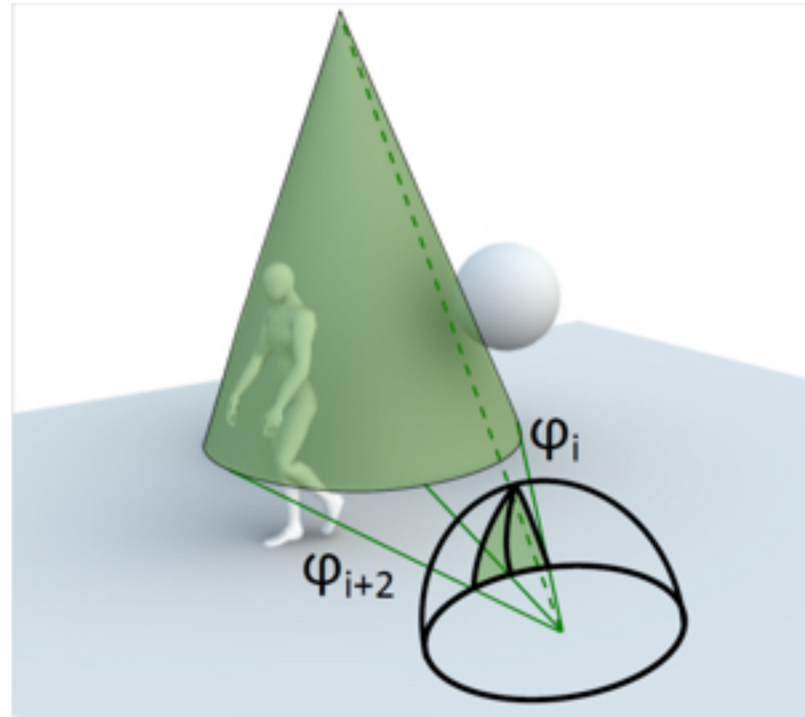
Accumulating Log-SH Visibility

Given spherical log-SH visibility for



dynamic blocker "meshes"

$$\{\mathbf{v}^{0}_{\log}, \mathbf{v}^{1}_{\log}, \cdots, \mathbf{v}^{\mathrm{B\text{-}1}}_{\log}\}$$



dynamic height field geometry

$$\mathbf{v}^{\mathrm{HF}}_{\log}$$

- the **total** log-SH visibility vector is $\mathbf{V}_{\log} = \mathbf{v}^{\mathrm{HF}}_{\log} + \sum_{b=0}^{B-1} \mathbf{v}^{\mathrm{b}}_{\log}$

SH Exponentiation [RWS*06]

SH Exponentiation [RWS*06]

- Given any log-SH coefficient vector $\mathbf{f}_{\log}$, we use **SH exponentiation** to compute the (primal-domain) SH coefficients $\mathbf{f}$

SH Exponentiation [RWS*06]

- Given any log-SH coefficient vector $\mathbf{f}_{\log}$, we use **SH exponentiation** to compute the (primal-domain) SH coefficients $\mathbf{f}$

- We use the **HYBrid** SH exponentiation method **[RWS*06]**
- A series expansion of the exponential, projected into SH

SH Exponentiation [RWS*06]

- Given any log-SH coefficient vector $\mathbf{f}_{\log}$, we use **SH exponentiation** to compute the (primal-domain) SH coefficients $\mathbf{f}$

- We use the **HYBrid** SH exponentiation method **[RWS*06]**
- A series expansion of the exponential, projected into SH

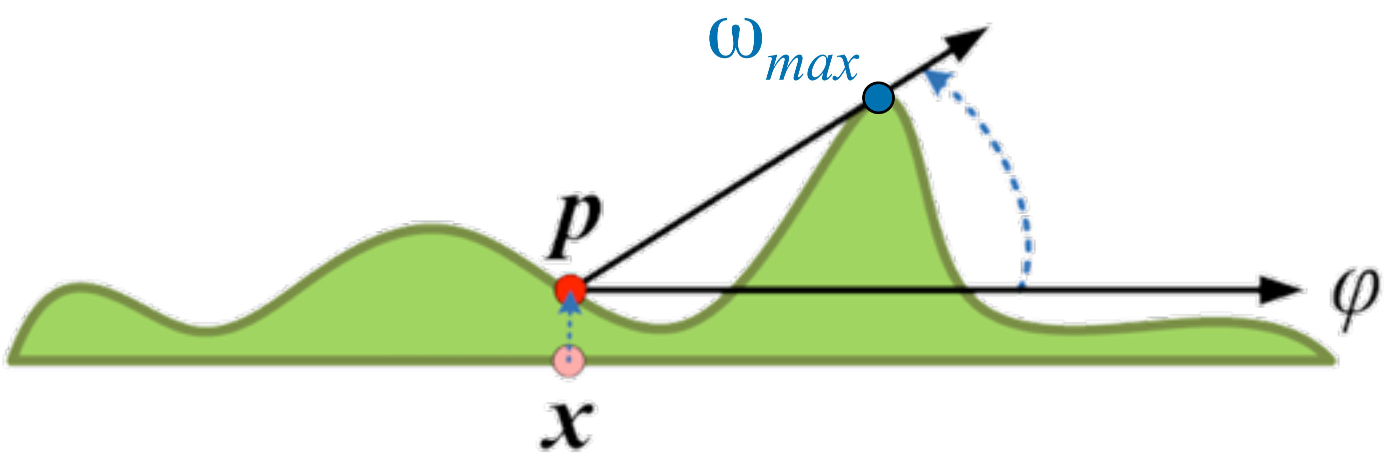
- Improved numerical stability with:
  - DC isolation
  - optimal linear-order approximation
  - SH scaling & squaring product accumulation

$$\mathbf{f} = \exp\left(\mathbf{f}_{\log}\right) \approx \mathbf{1} + \mathbf{f}_{\log} + \frac{\mathbf{f}_{\log}^2}{2} + \frac{\mathbf{f}_{\log}^3}{3!} + \cdots$$

Summary of Main Ideas

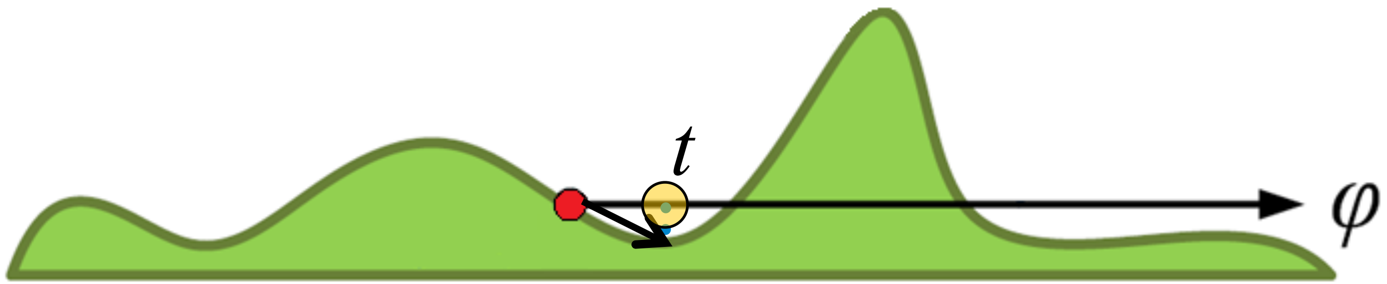
1. compute *HF self-visibility* (in *log-SH space*)
   - create multi-resolution height ***pyramids***
   - sample from pyramid levels
   - pre-filter data
   - compose visibility ***analytically*** in log-space
2. compute *HF cast-visibility* (onto meshes)
3. compute *mesh cast-visibility* (onto HF) **and** *self-visibility*
4. accumulate total spherical visibility
5. compute log-SH BRDF and perform final shading

HF Definitions and Notation [SN08]



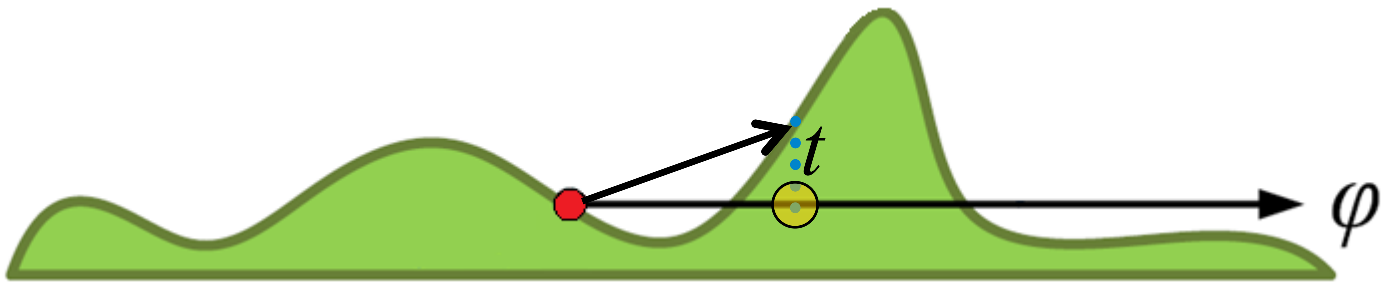
Need to find maximum blocking angle ω_{max} along direction φ.

Calculating the Max Blocking Angle



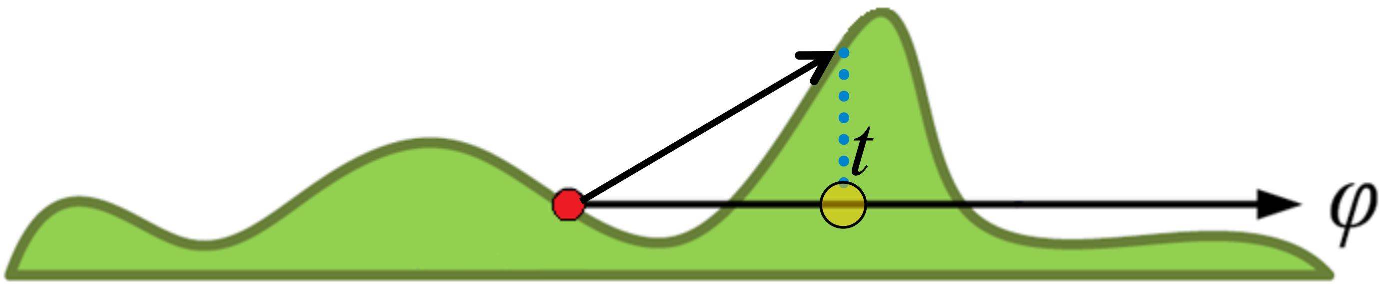
[SN08]

Calculating the Max Blocking Angle



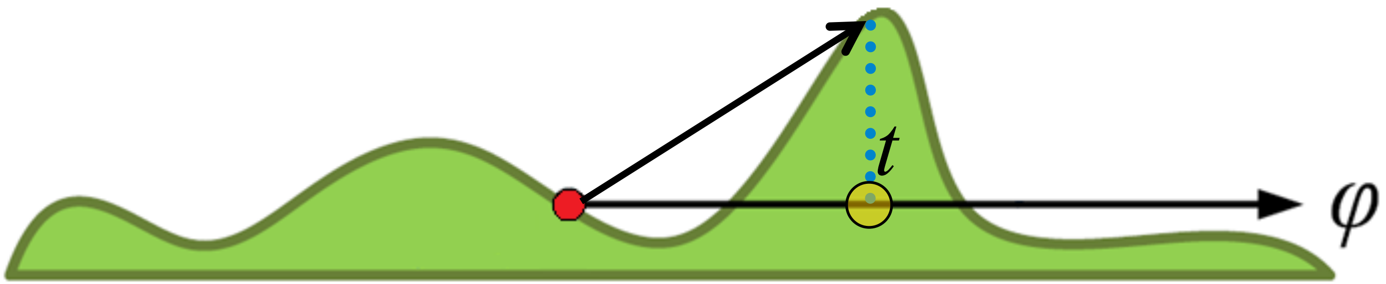
[SN08]

Calculating the Max Blocking Angle



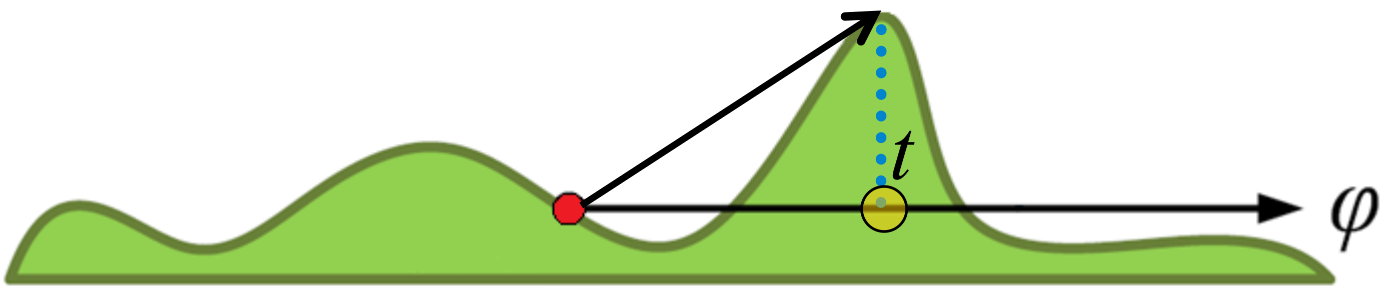
[SN08]

Calculating the Max Blocking Angle



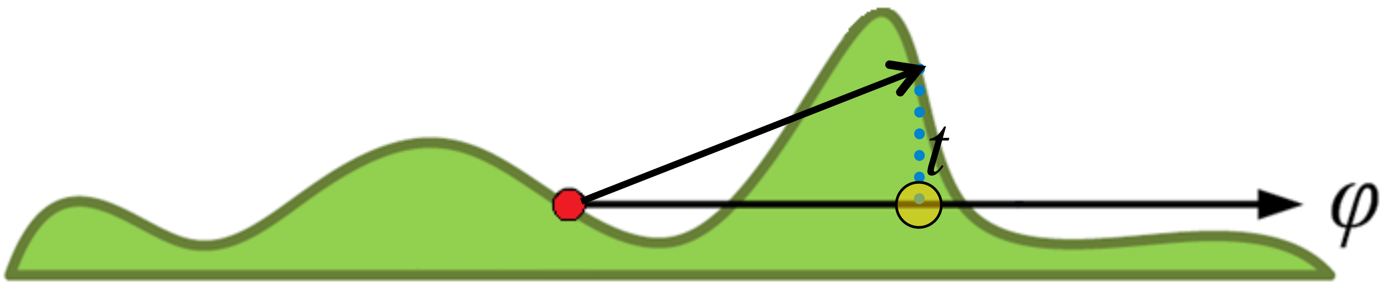
[SN08]

Calculating the Max Blocking Angle



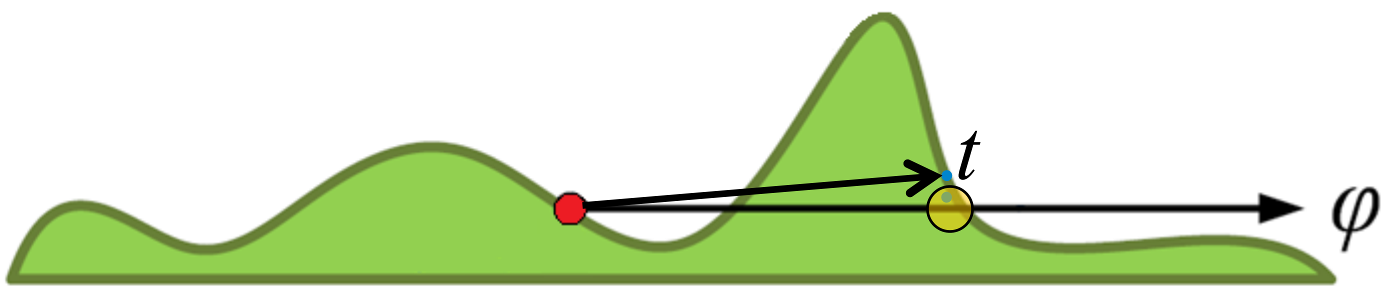
[SN08]

Calculating the Max Blocking Angle



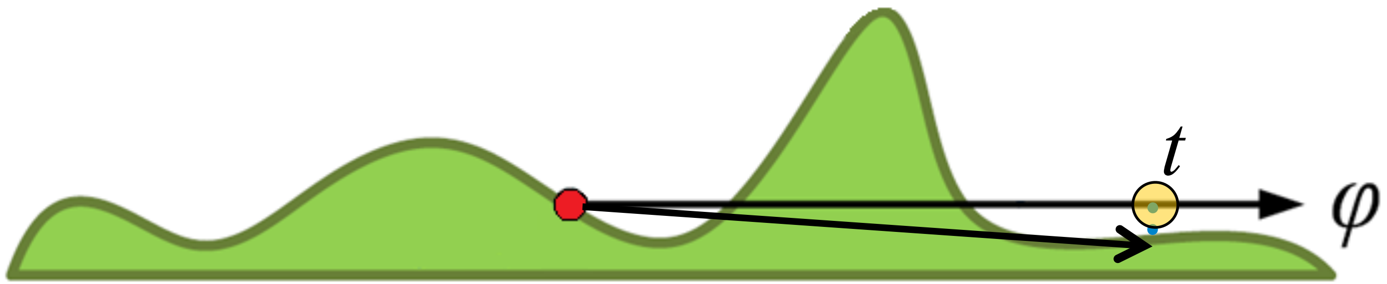
[SN08]

Calculating the Max Blocking Angle



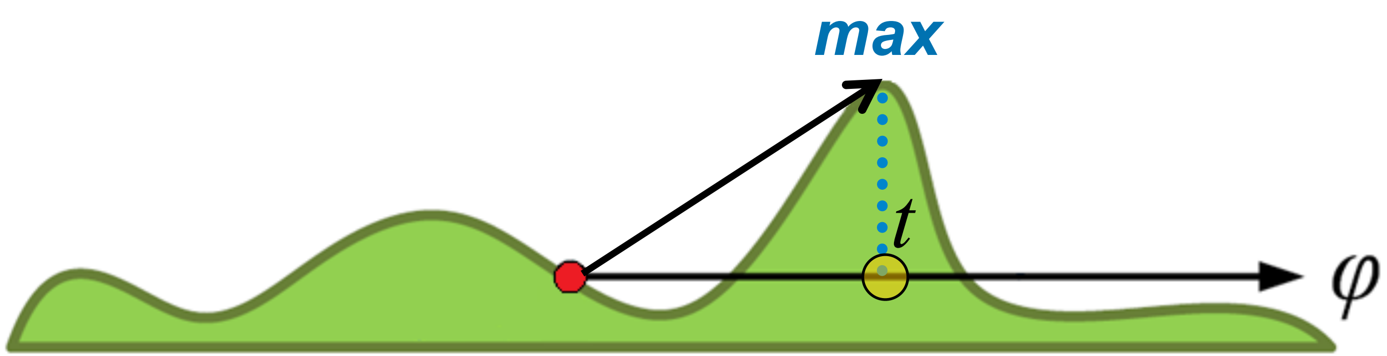
[SN08]

Calculating the Max Blocking Angle



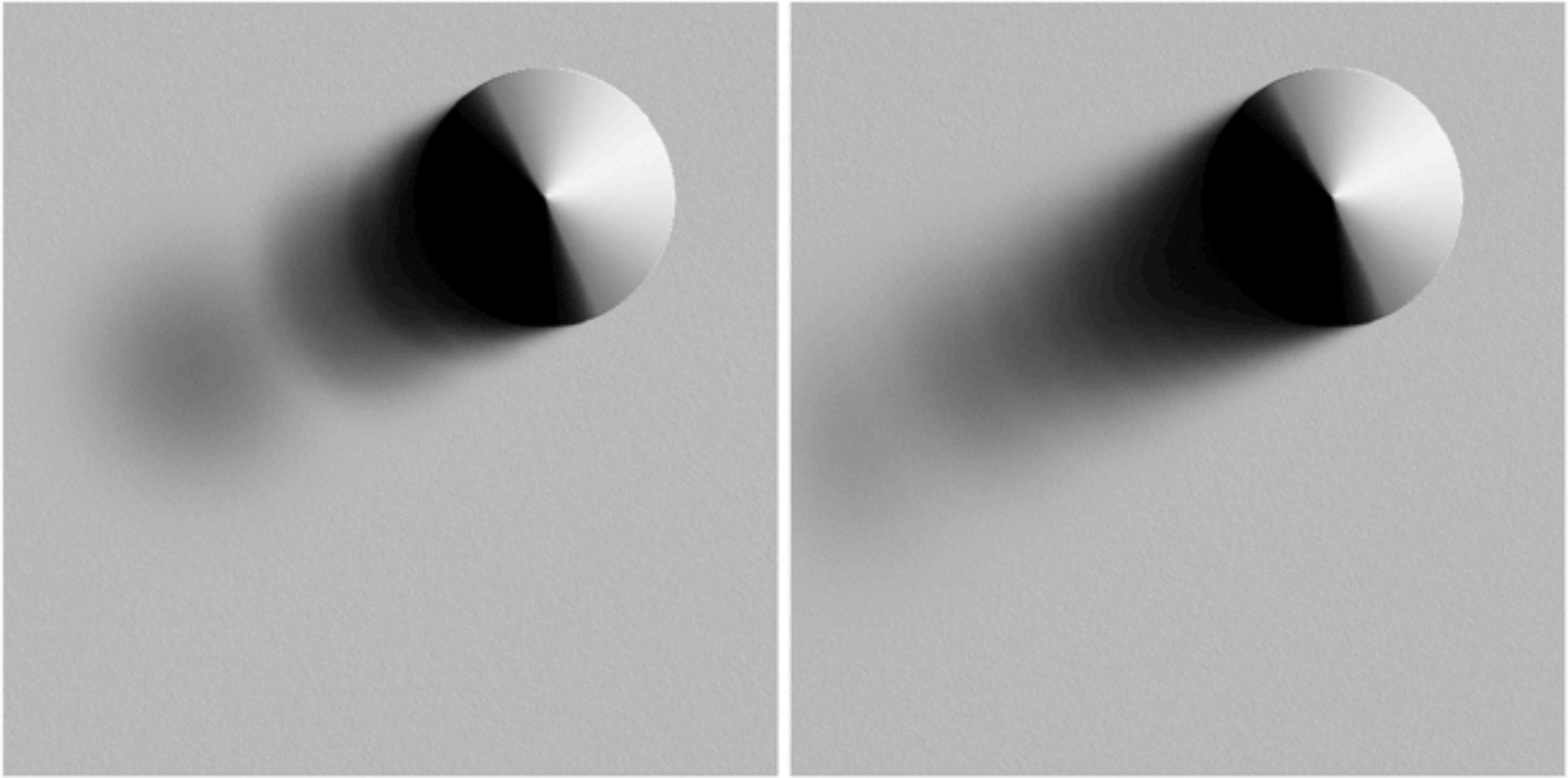
[SN08]

Calculating the Max Blocking Angle

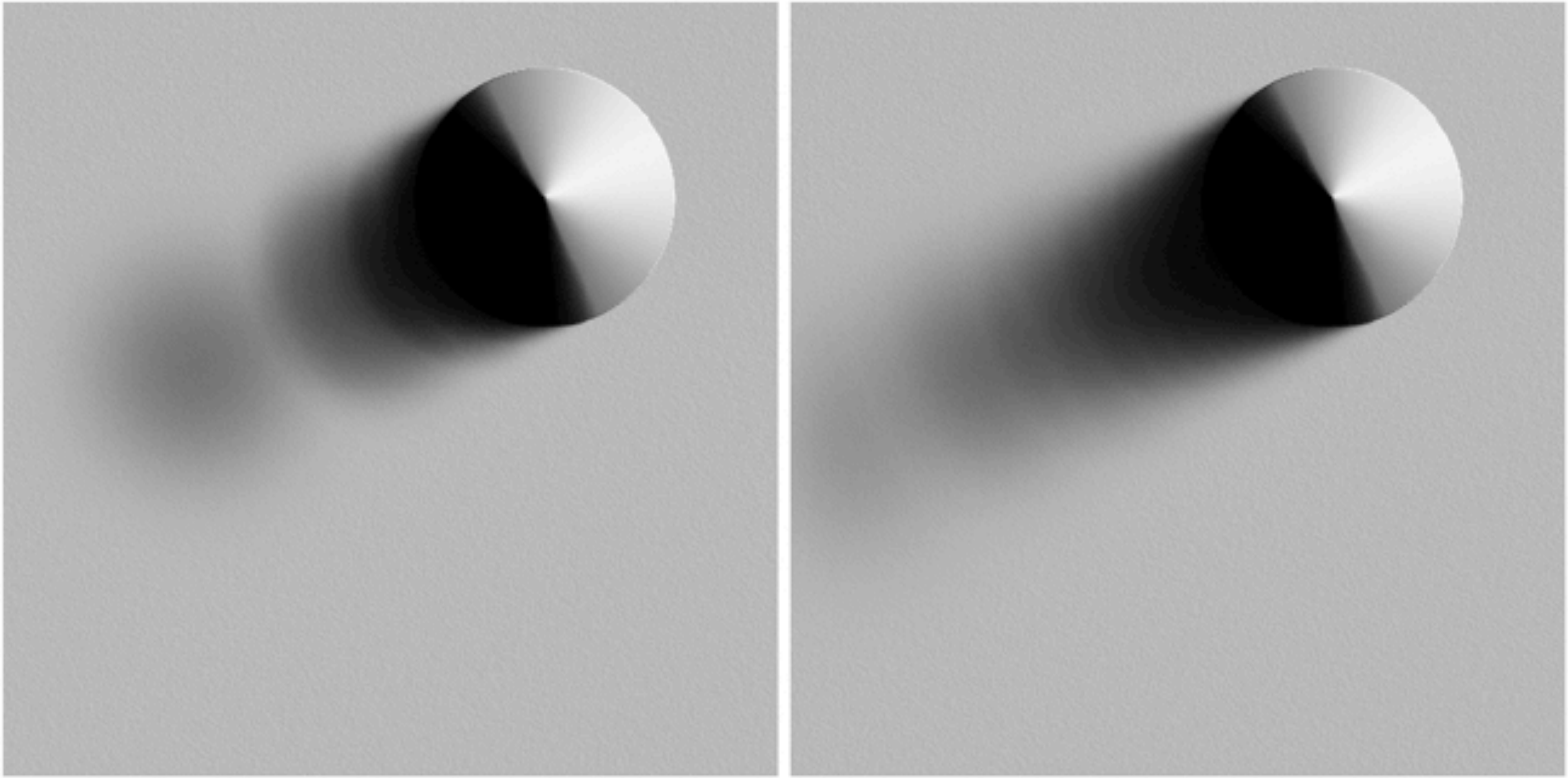


[SN08]

Brute Force Sampling [SN08]

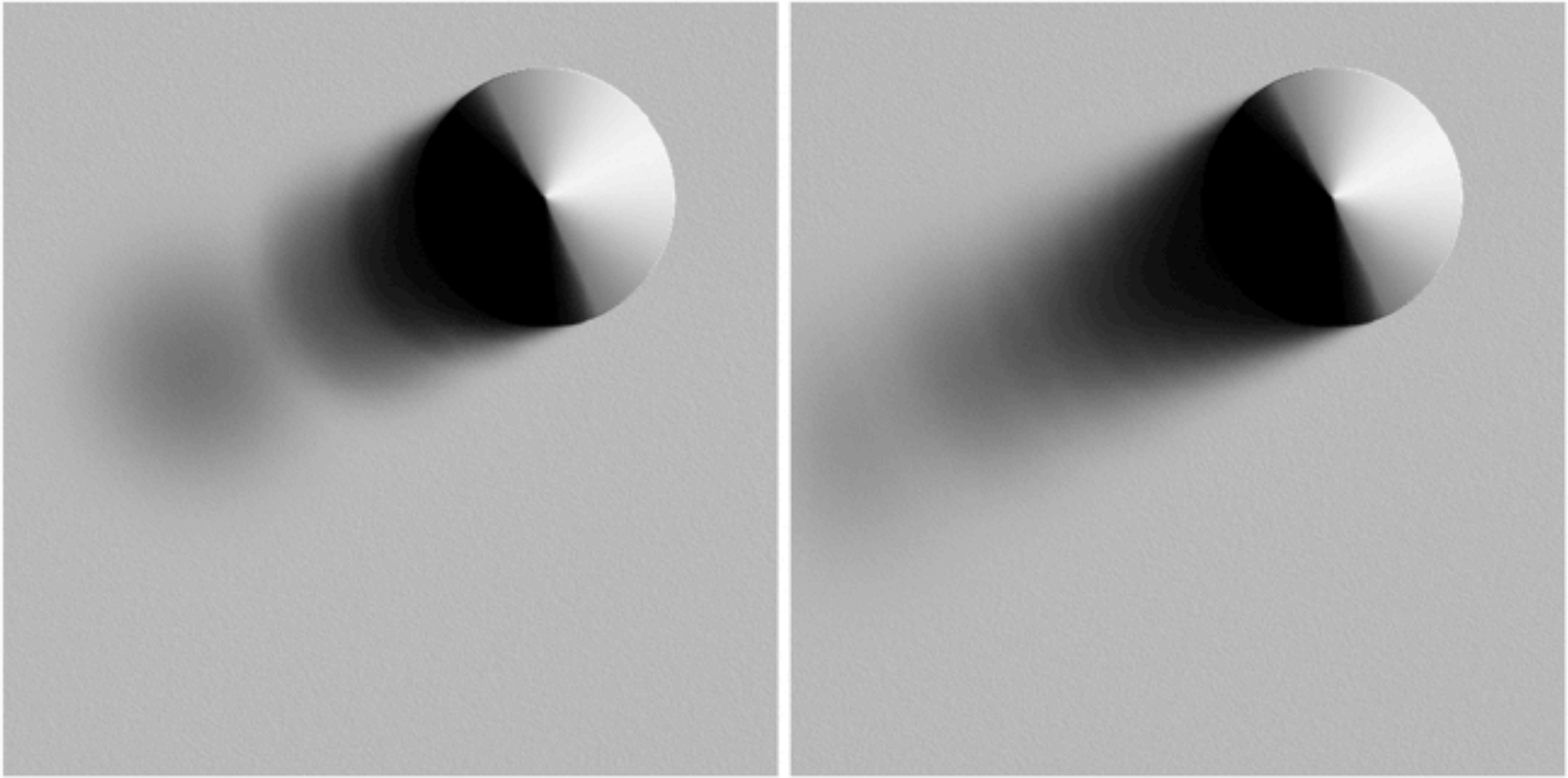


Brute Force Sampling [SN08]



Problem: *aliasing* – need *many* samples in t.

Brute Force Sampling [SN08]



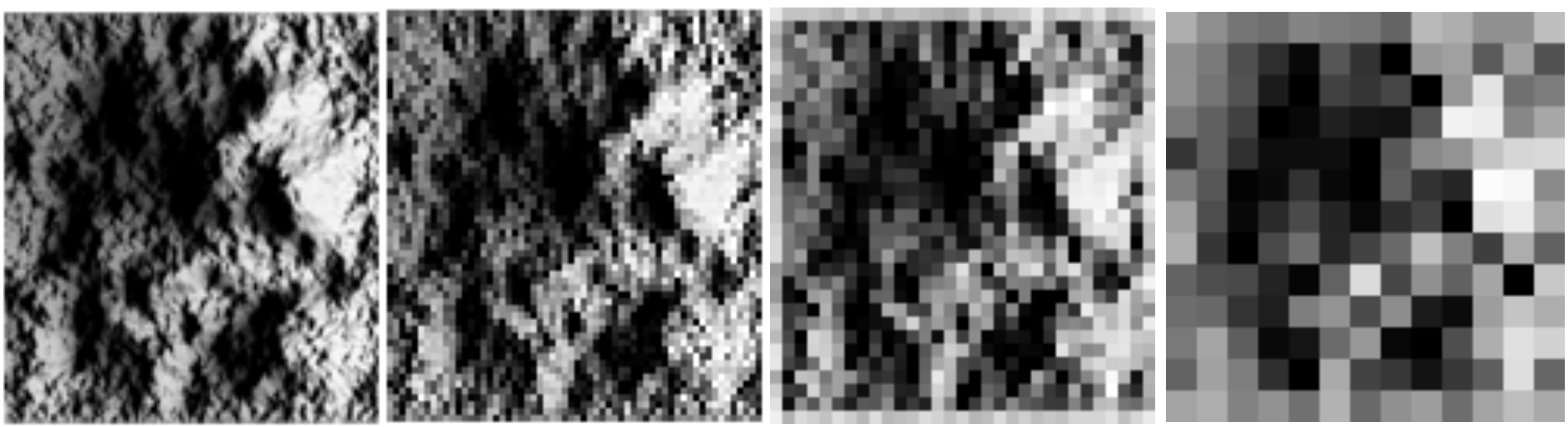
Problem: *aliasing* – need *many* samples in t.
Solution: *prefilter* data, apply *multi-scale sampling*.

Multi-Resolution Height Sampling [SN08]

f_i	height pyramid level i
$\tau_i = 2^{f(i)}$	sampling distance for level i

Multi-Resolution Height Sampling [SN08]

f_i	height pyramid level i
$\tau_i = 2^{f(i)}$	sampling distance for level i

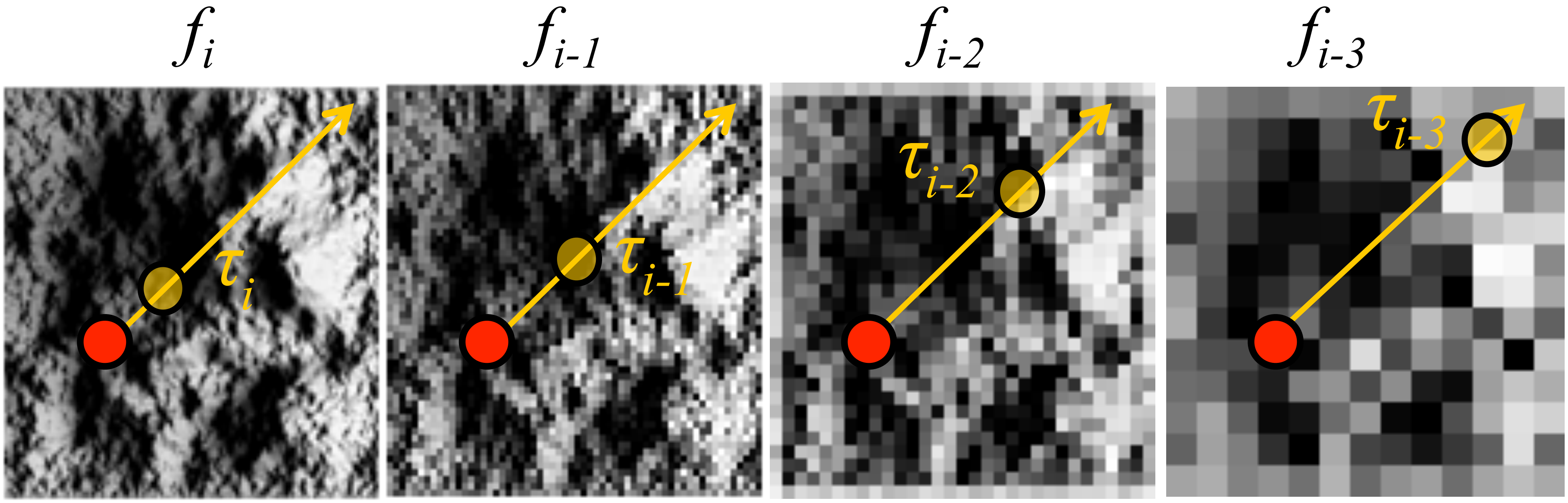


Multi-Resolution Height Sampling [SN08]

f_i height pyramid level i

$\tau_i = 2^{f(i)}$ sampling distance for level i

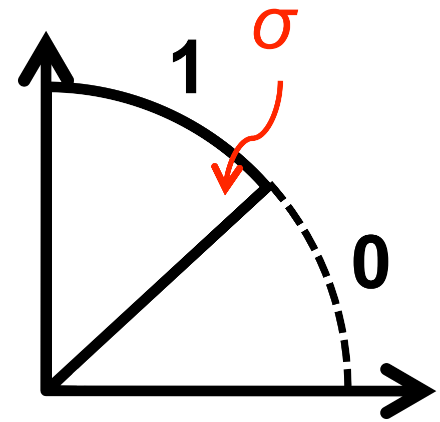
Sample coarser levels further from x.



Elevation Visibility

- starting with binary *visibility* for an elevation slice:

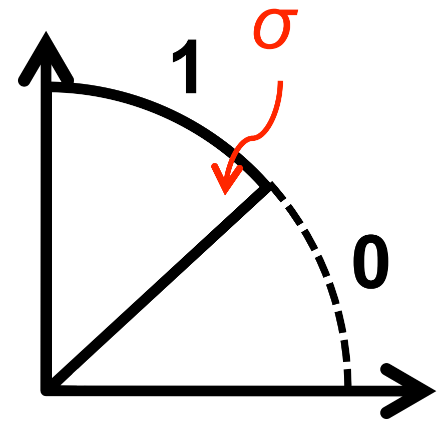
$$v(\omega;\sigma) = \begin{cases} 0, & \text{if } \omega \le \sigma \\ 1, & \text{otherwise.} \end{cases}$$



Elevation Visibility

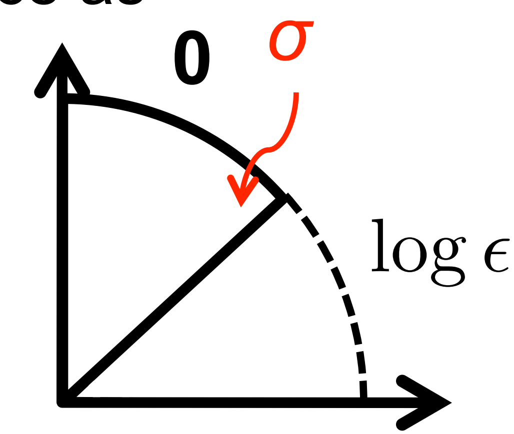
- starting with binary *visibility* for an elevation slice:

$$v(\omega;\sigma) = \begin{cases} 0, & \text{if } \omega \leq \sigma \\ 1, & \text{otherwise.} \end{cases}$$

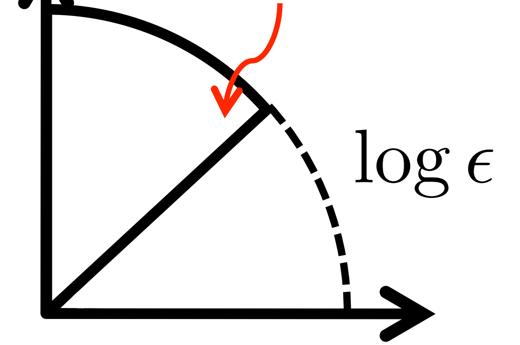


- we can express the *log-visibility* for the slice as

$$v_{\log}(\omega;\sigma) = \begin{cases} \log\epsilon, & \text{if } \omega \leq \sigma \\ 0, & \text{otherwise.} \end{cases}$$



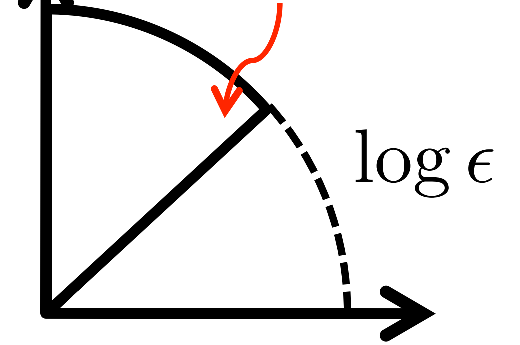
$$v_{\log}(\omega;\sigma) = \begin{cases} \log\epsilon, & \text{if } \omega \le \sigma \\ 0, & \text{otherwise.} \end{cases}$$



and represent it analytically in the **Normalized Legendre Polynomial** (**NLP**) basis:

$$\mathbf{v}_{\log}(\sigma) = \int_{\pi/2-\sigma}^{\pi} (\log\epsilon)\ \hat{\mathbf{P}}(\cos\theta)\ \sin\theta \mathrm{d}\theta$$

$$v_{\log}(\omega;\sigma) = \begin{cases} \log\epsilon, & \text{if } \omega \le \sigma \\ 0, & \text{otherwise.} \end{cases}$$



and represent it analytically in the **Normalized Legendre Polynomial** (**NLP**) basis:

$$\mathbf{v}_{\log}(\sigma) = \int_{\pi/2-\sigma}^{\pi} (\log\epsilon)\ \hat{\mathbf{P}}(\cos\theta)\ \sin\theta \mathrm{d}\theta$$

$$= \log\epsilon \times \left[\frac{\sin\sigma+1}{\sqrt{2}}, \frac{-3\cos^2\sigma}{2\sqrt{6}}, \frac{-5\sin\sigma\cos^2\sigma}{2\sqrt{10}}, \frac{7\cos^2\sigma(-4+5\cos^2\sigma)}{8\sqrt{14}}\right]$$

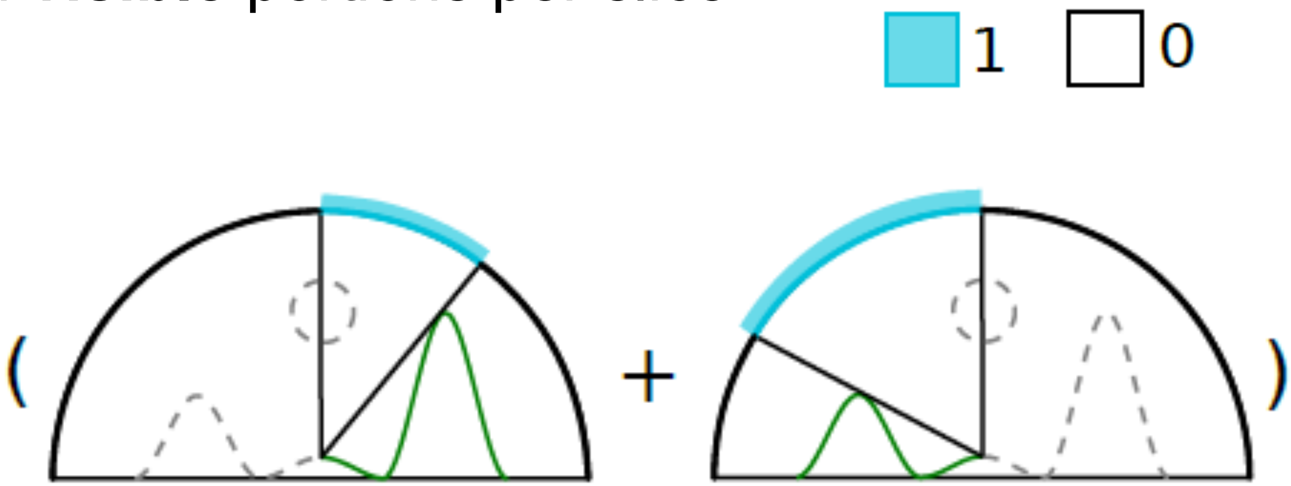
Accumulating HF Visibility

- in the primal domain: can **sum** SH visibility for each slice

Accumulating HF Visibility

- in the primal domain: can **sum** SH visibility for each slice

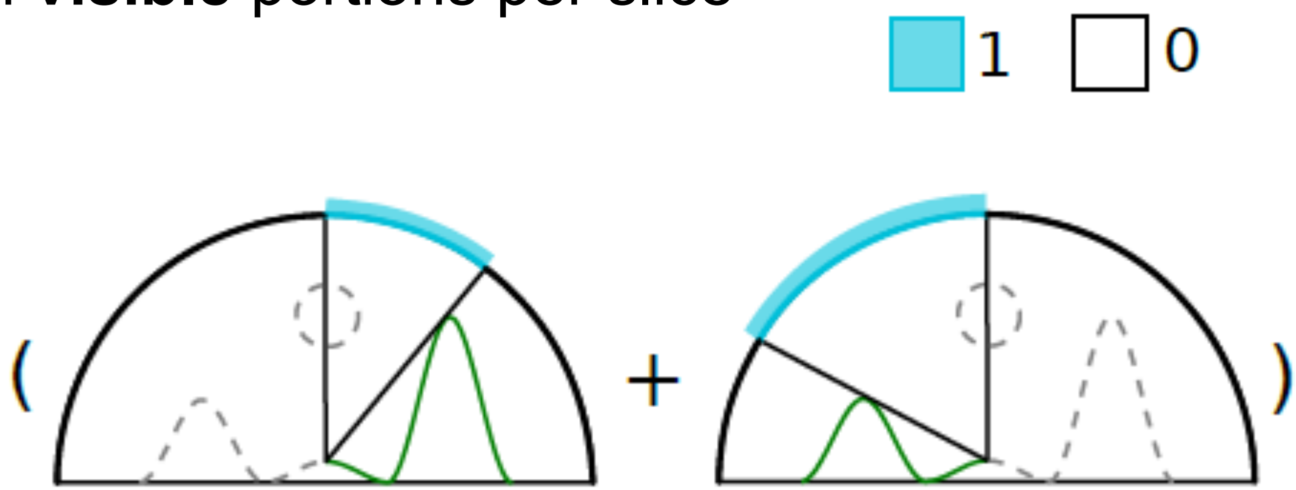
- initialize the total visibility to 0 (**fully occluded**)
- **add** in **visible** portions per slice



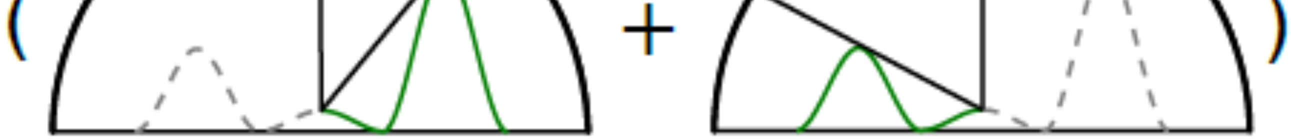
Accumulating HF Visibility

- in the primal domain: can **sum** SH visibility for each slice

- initialize the total visibility to 0 (**fully occluded**)
- **add** in **visible** portions per slice

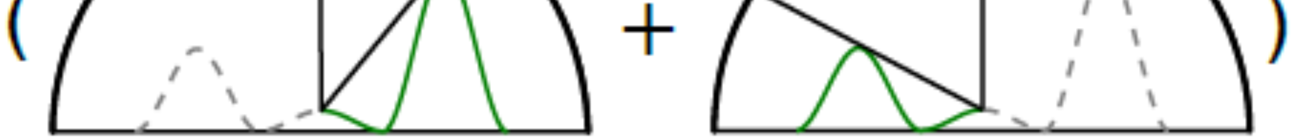


- but, in the log domain: **sum**s correspond to **products**
- how do we accumulate products of visibility?



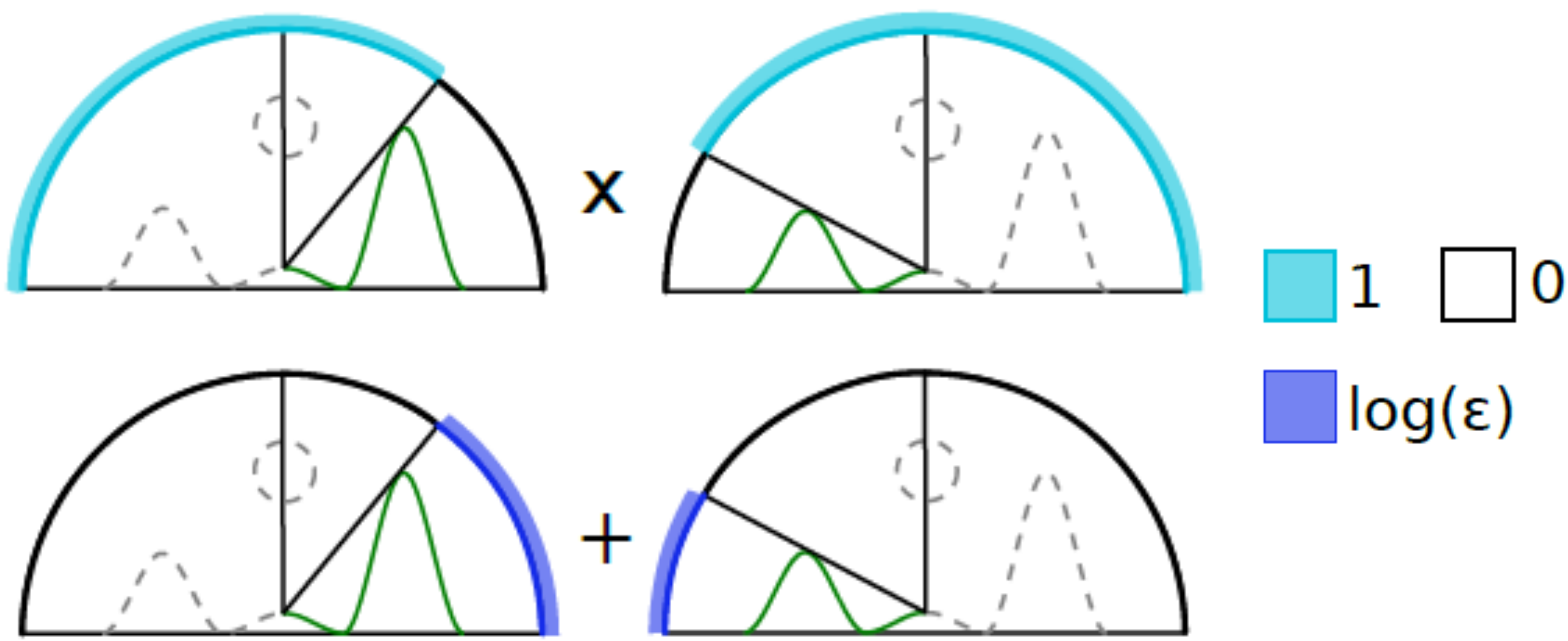
- but, in the log domain: **sum**s correspond to **products**
- how do we accumulate products of visibility?

- begin by initializing total *log*-visibility to 1 (**full visibility**)
- **multiply** in the **occluded** portions
  - do this by **summing** the log-visibility



- but, in the log domain: **sum**s correspond to **products**
- how do we accumulate products of visibility?

- begin by initializing total *log*-visibility to 1 (**full visibility**)
- **multiply** in the **occluded** portions
  - do this by **summing** the log-visibility

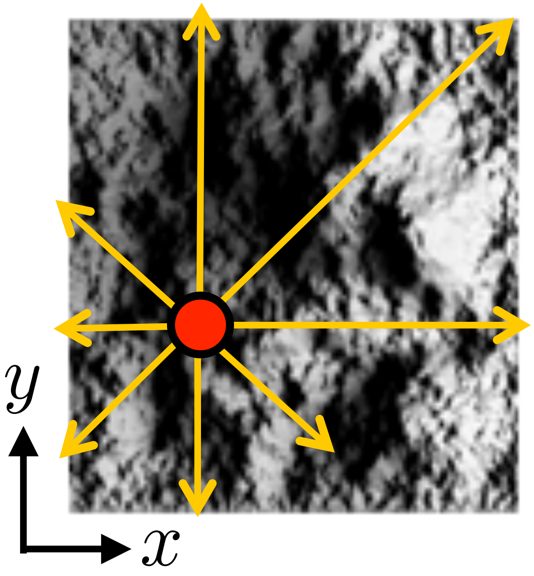


Visibility Slice Interpolation [NS09]

- already computed log-ZH azimuthal visibility, per-direction

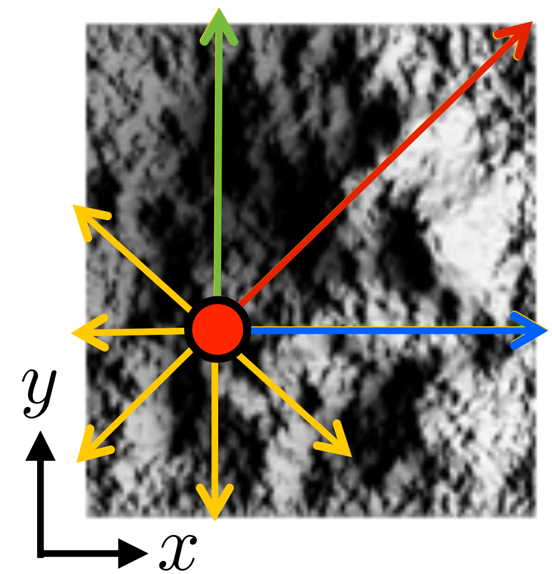
Visibility Slice Interpolation [NS09]

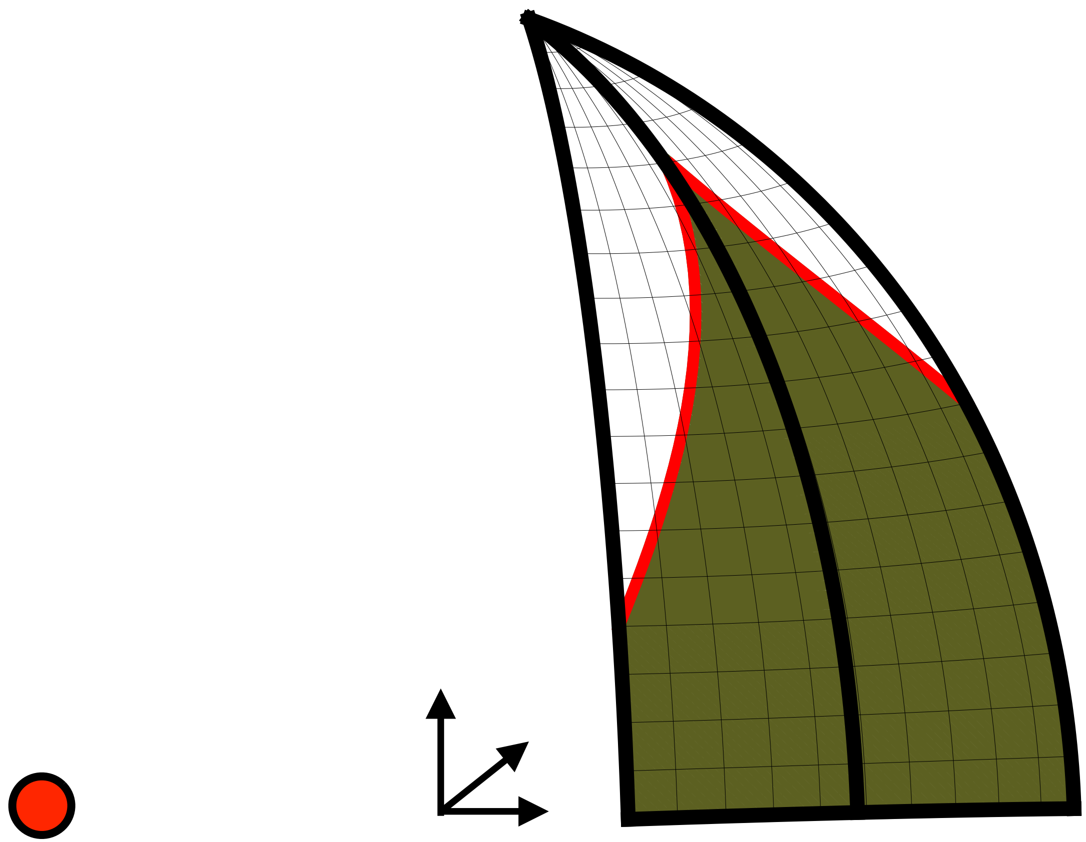
- already computed log-ZH azimuthal visibility, per-direction
- can combine and interpolate azimuthal **log-SH elevation** coefficients together to form **full log-SH spherical visibility**



Visibility Slice Interpolation [NS09]

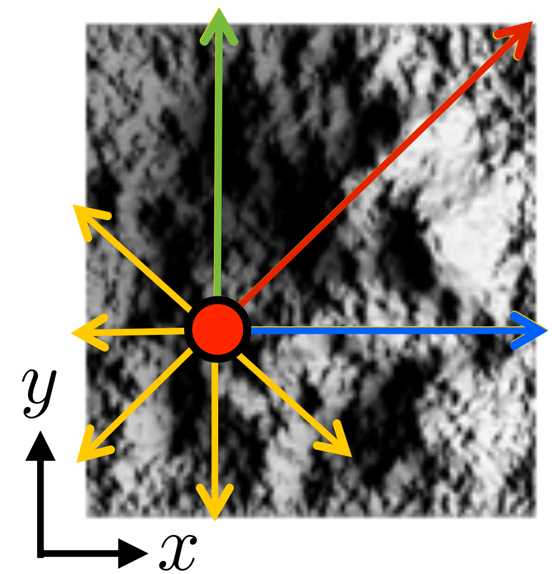
- already computed log-ZH azimuthal visibility, per-direction
- can combine and interpolate azimuthal **log-SH elevation** coefficients together to form **full log-SH spherical visibility**

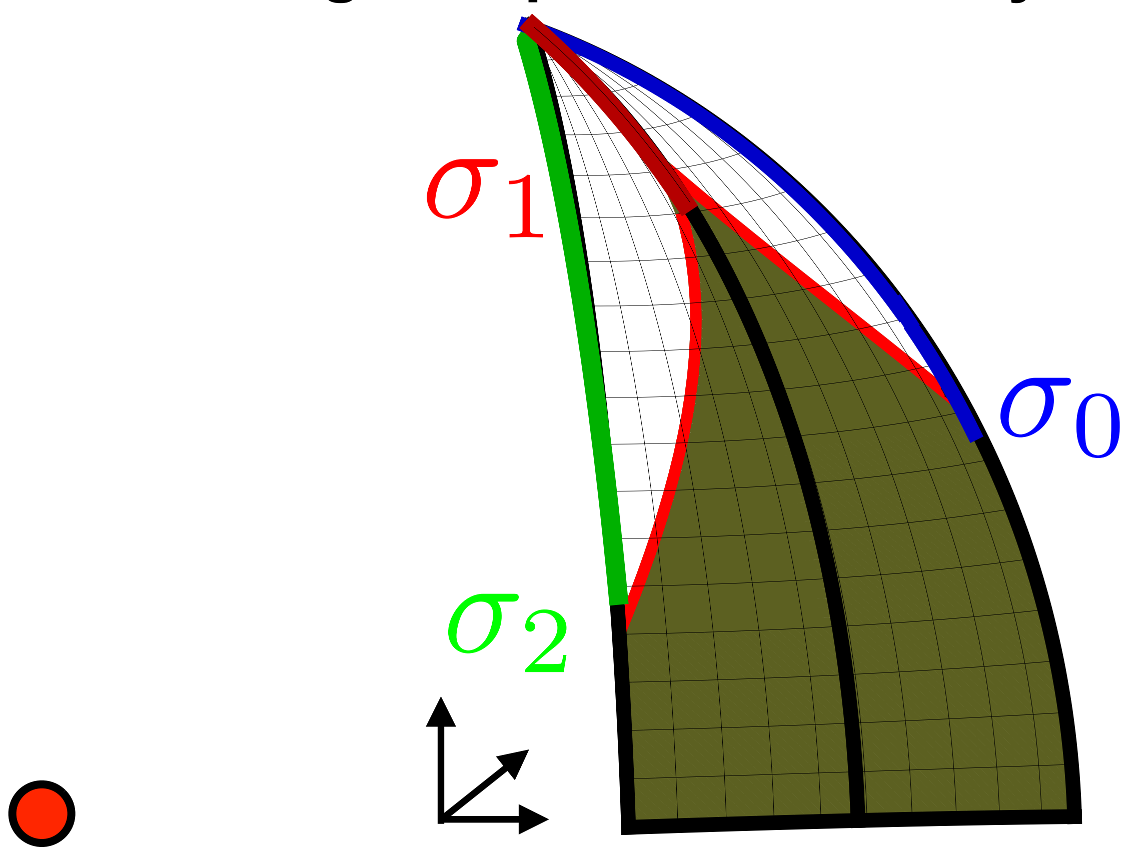


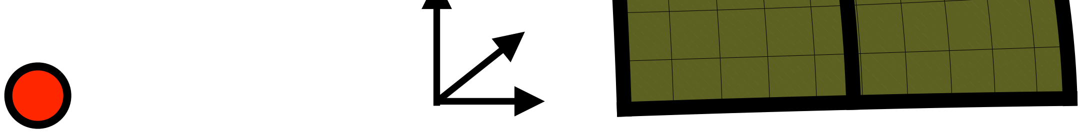


Visibility Slice Interpolation [NS09]

- already computed log-ZH azimuthal visibility, per-direction
- can combine and interpolate azimuthal **log-SH elevation** coefficients together to form **full log-SH spherical visibility**







- requires 1 precomputed interpolation + projection matrix

$$\mathbf{v}_{\log}^{\text{wedge}} = \left[\quad \mathbb{M}_{lin} \quad \right] \begin{bmatrix} \mathbf{v}_{\log}(\sigma_0) \\ \mathbf{v}_{\log}(\sigma_1) \end{bmatrix}$$

- **rotate** and **sum** across each wedge's $\mathbf{v}_{\log}^{\text{wedge}}$ to form final log-SH vector $\mathbf{v}_{\log}^{\text{HF}}$

Summary of Main Ideas

1. compute *HF self-visibility* (in *log-SH space*)

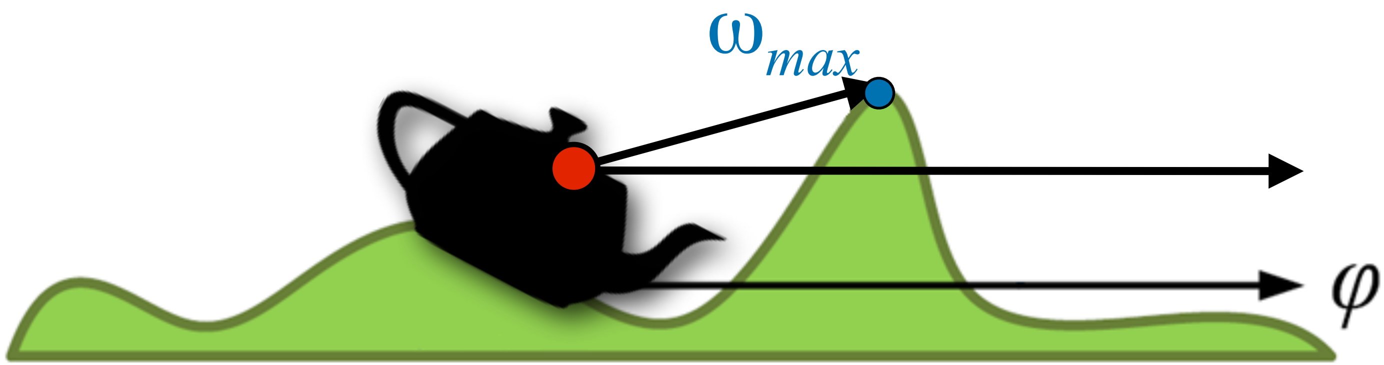
2. compute *HF cast-visibility* (onto meshes)
   - ***repeat*** multi-resolution marching
   - ***offset*** the height field queries

3. compute *mesh cast-visibility* (onto HF) **and** *self-visibility*

4. accumulate total spherical visibility

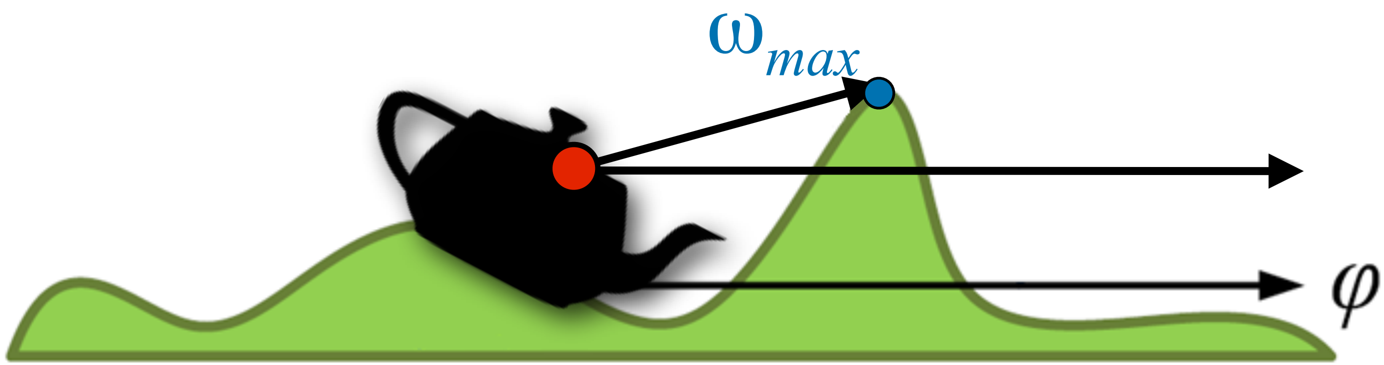
5. compute log-SH BRDF and perform final shading

Height Field Cast Visibility onto Meshes



Need to find ω_{max} on **mesh** **shading point** along each direction φ

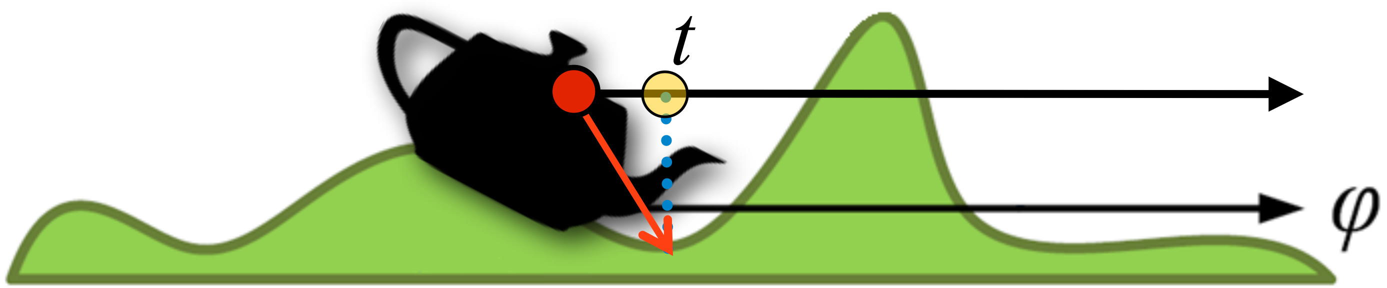
Height Field Cast Visibility onto Meshes



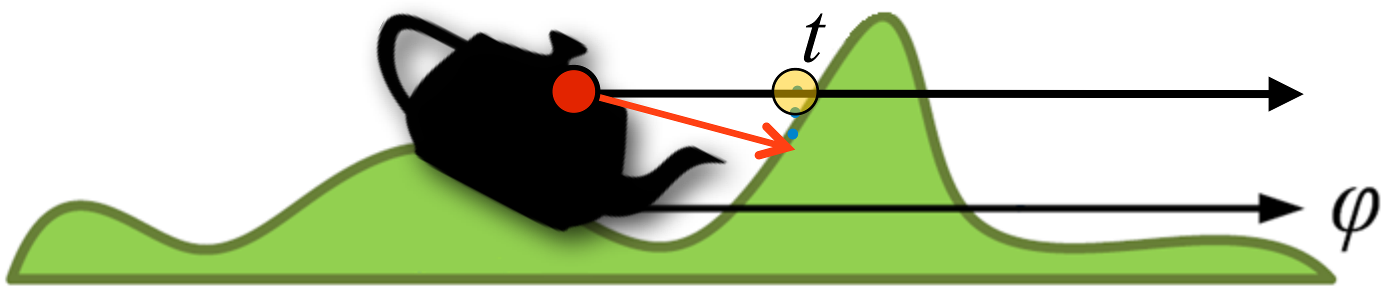
Need to find ω_{max} on **mesh** **shading point** along each direction φ

- Assume an infinite plane for the HF base elevation
  - minimum blocking angle can't go negative

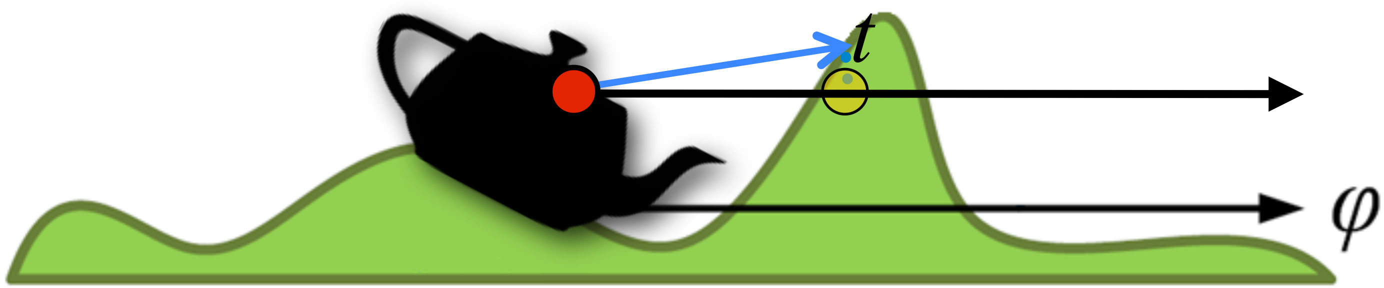
Calculating the Max Blocking Angle



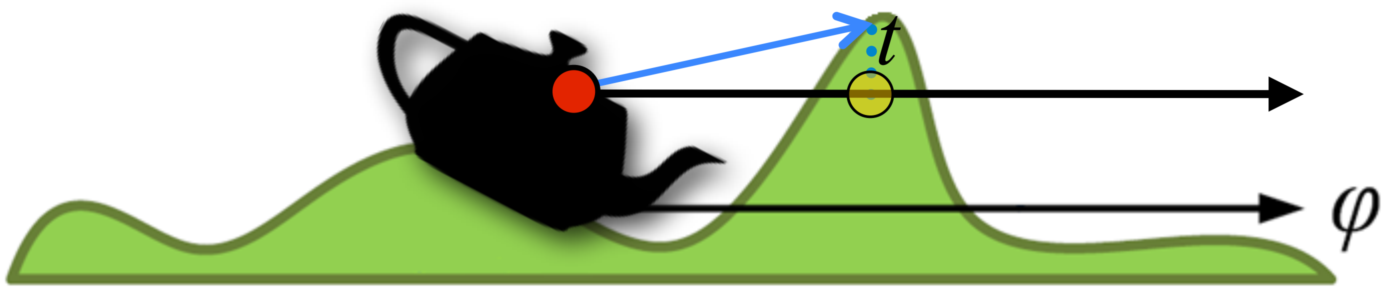
Calculating the Max Blocking Angle



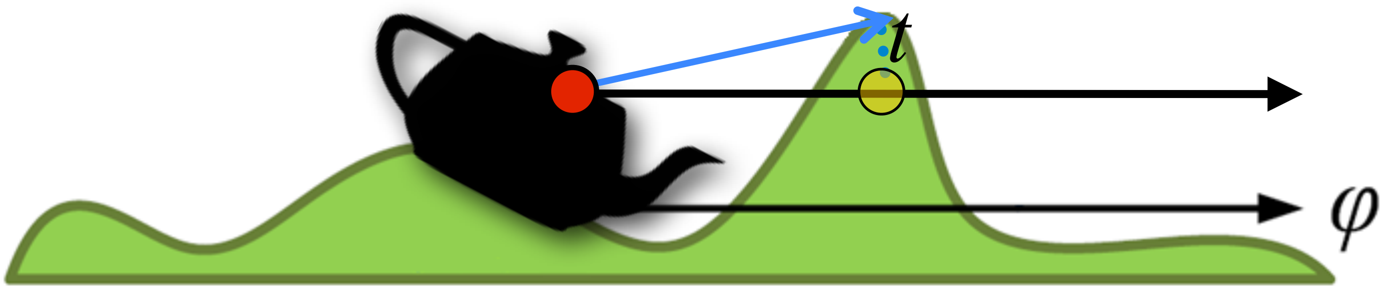
Calculating the Max Blocking Angle



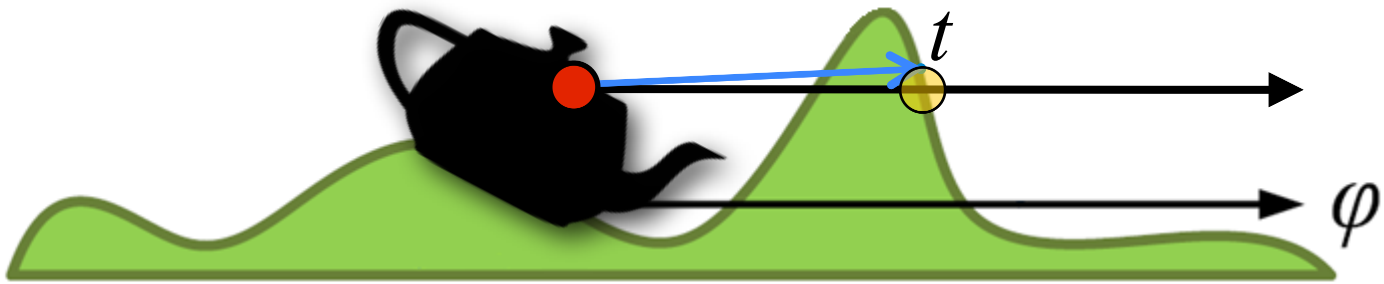
Calculating the Max Blocking Angle



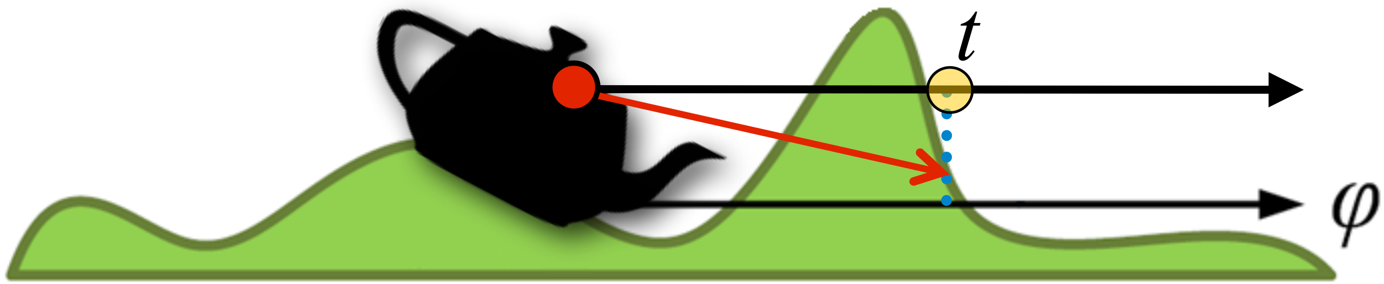
Calculating the Max Blocking Angle



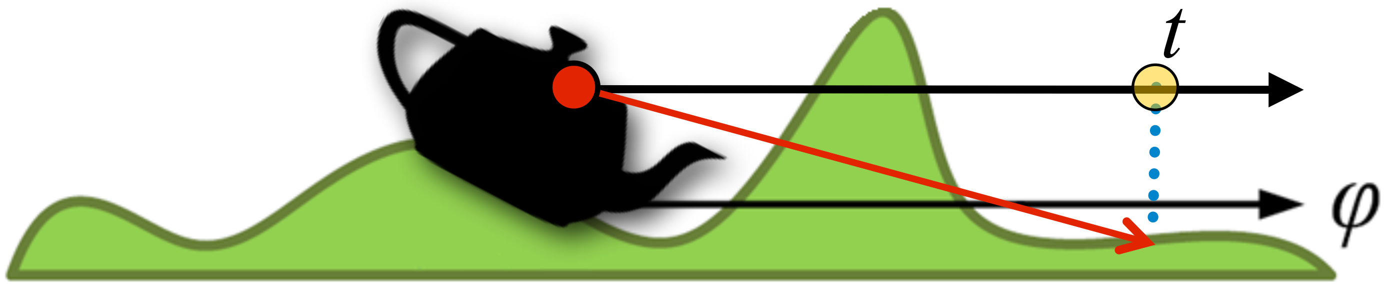
Calculating the Max Blocking Angle



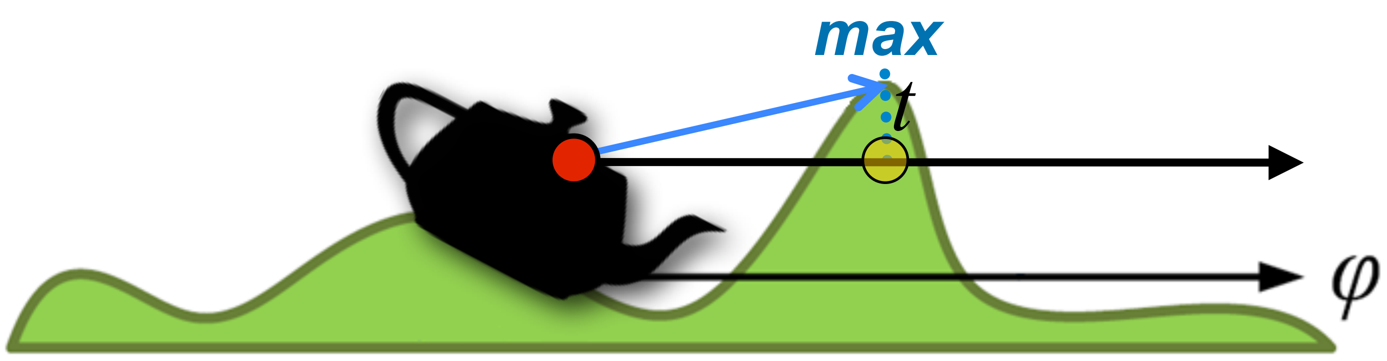
Calculating the Max Blocking Angle



Calculating the Max Blocking Angle



Calculating the Max Blocking Angle



Summary of Main Ideas

1. compute *HF self-visibility* (in *log-SH space*)

2. compute *HF cast-visibility* (onto meshes)

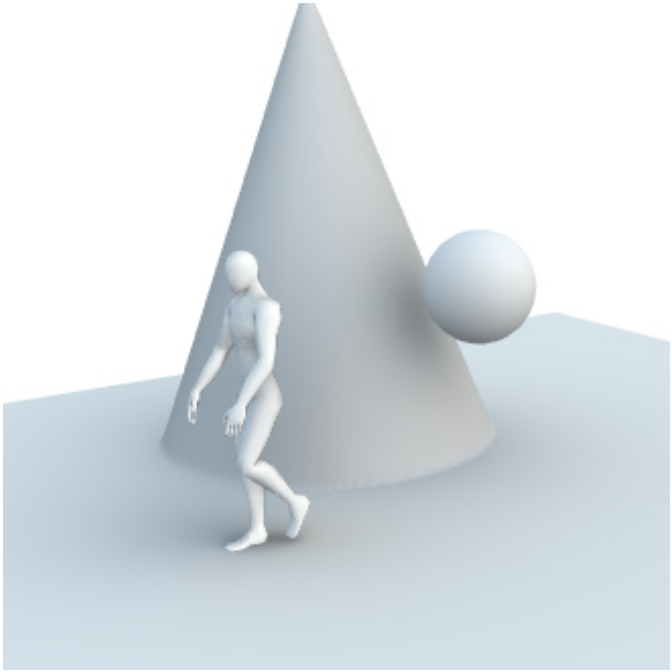
3. compute *mesh cast-visibility* (onto HF) **and** *self-visibility*
   - extend traditional SH exponentiation approach **[RWS*06;SGNS07]**
   - ***decompose*** dynamic mesh blockers into spheres
   - ***compute*** & ***accumulate log-SH visibility*** for spherical blockers
     - on the mesh shading points
     - ***repeat*** over the HF shading points
   - intelligently cull the sphere set during accumulation
     - reduces numerical accumulation error

4. accumulate total spherical visibility

5. compute log-SH BRDF and perform final shading

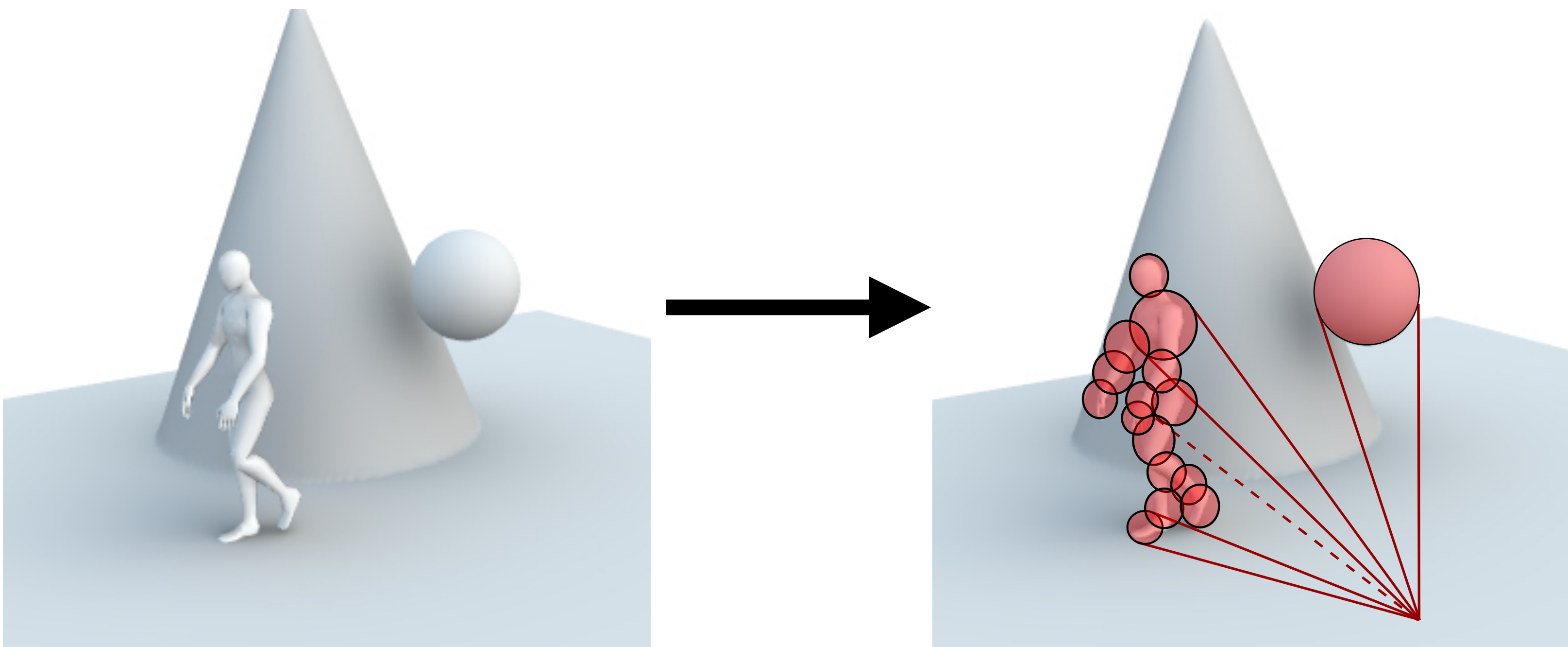
Spherical Blockers [RWS*06]

- approximate dynamic meshes with a set of spheres
  - precomputed once
  - skinned dynamically during animation/deformation



Spherical Blockers [RWS*06]

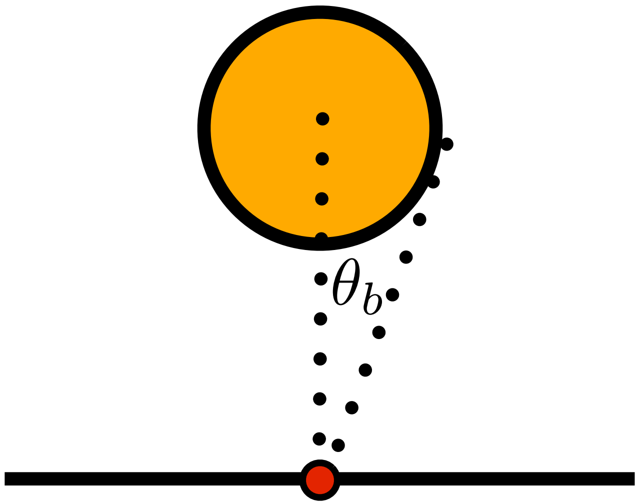
- approximate dynamic meshes with a set of spheres
  - precomputed once
  - skinned dynamically during animation/deformation



Spherical Blocker Log SH Visibility

- can compute log-visibility SH coefficients ***analytically***
- begin with a canonical alignment:

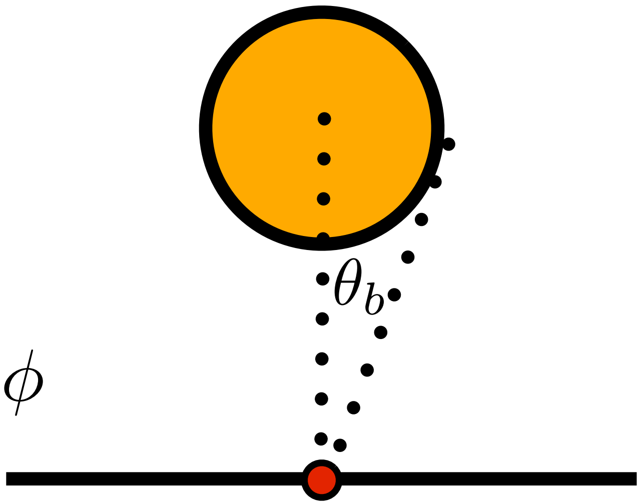
$$\theta_b = \arcsin\left(r/d\right)$$



Spherical Blocker Log SH Visibility

- can compute log-visibility SH coefficients ***analytically***
- begin with a canonical alignment:

$$\theta_b = \arcsin\left(r/d\right)$$

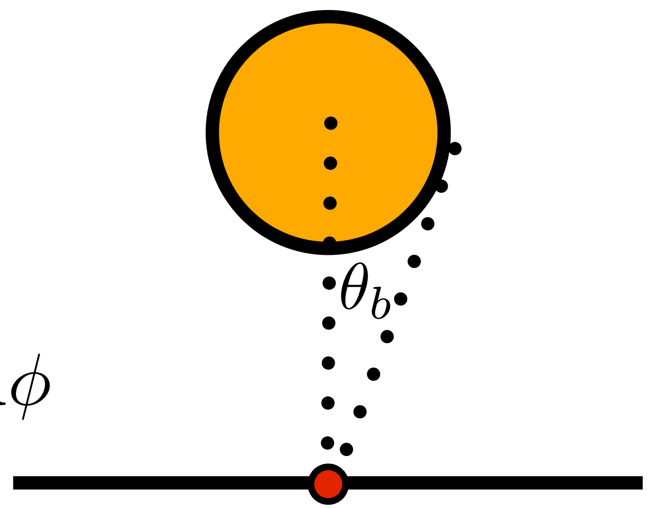


$$\int_{\theta=0}^{\theta_b} \int_{\phi=0}^{2\pi} (\log \epsilon)\ y_l^0(\theta,\phi)\ \sin\theta \, \mathrm{d}\theta \mathrm{d}\phi$$

Spherical Blocker Log SH Visibility

- can compute log-visibility SH coefficients ***analytically***
- begin with a canonical alignment:

$$\theta_b = \arcsin\left(r/d\right)$$

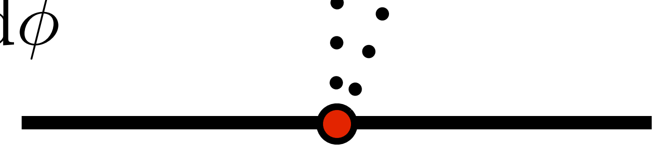


$$\int_{\theta=0}^{\theta_b}\int_{\phi=0}^{2\pi}(\log\epsilon)\ y_l^0(\theta,\phi)\ \sin\theta\,\mathrm{d}\theta\mathrm{d}\phi$$

- solve analytically (we use order-4 SH, so 4 ZH coefficients)

$$\mathbf{v}_l^{\log} = \log\epsilon\times\left[-\sqrt{\pi}(-1+\cos\theta_b),\ \frac{\sqrt{3\pi}}{2}\sin^2\theta_b,\right.$$
$$\left.\frac{\sqrt{5\pi}}{2}\cos\theta_b\sin^2\theta_b,\ \frac{\sqrt{7\pi}}{16}(3+5\cos(2\theta_b))\sin^2\theta_b\right]$$

$$\int_{\theta=0} \int_{\phi=0} (\log \epsilon)\; y_l^0(\theta, \phi)\; \sin\theta \, \mathrm{d}\theta \mathrm{d}\phi$$

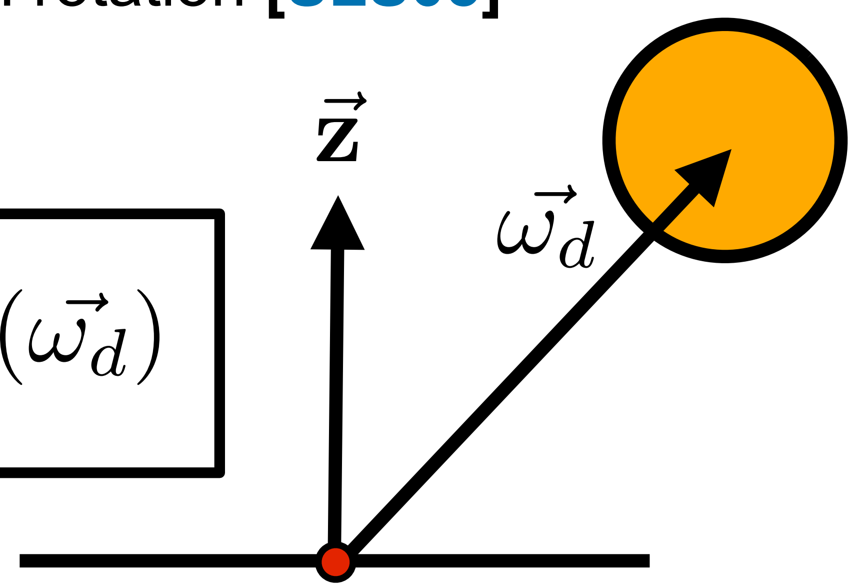


- solve analytically (we use order-4 SH, so 4 ZH coefficients)

$$\mathbf{v}_l^{\log} = \log\epsilon \times \left[-\sqrt{\pi}(-1+\cos\theta_b),\ \tfrac{\sqrt{3\pi}}{2}\sin^2\theta_b,\ \tfrac{\sqrt{5\pi}}{2}\cos\theta_b\sin^2\theta_b,\ \tfrac{\sqrt{7\pi}}{16}(3+5\cos(2\theta_b))\sin^2\theta_b \right]$$

- align to shading frame with ZH rotation **[SLS05]**

$$\boxed{\mathbf{v}_{l,m}^{\log} = \sqrt{\frac{4\pi}{2l+1}}\, \mathbf{v}_l^{\log}\, y_l^m(\vec{\omega_d})}$$



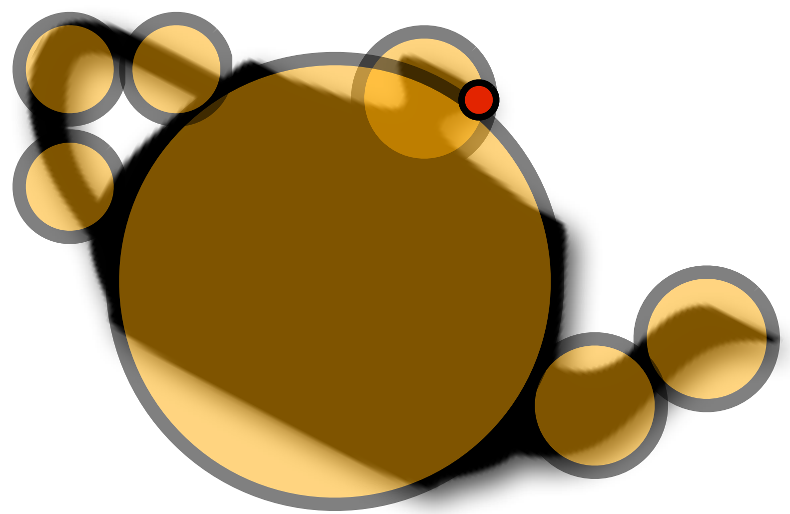
Spherical Blocker Self- & Cast- Shadows

Spherical Blocker Self- & Cast- Shadows

- accumulate spherical blocker occlusion for both:

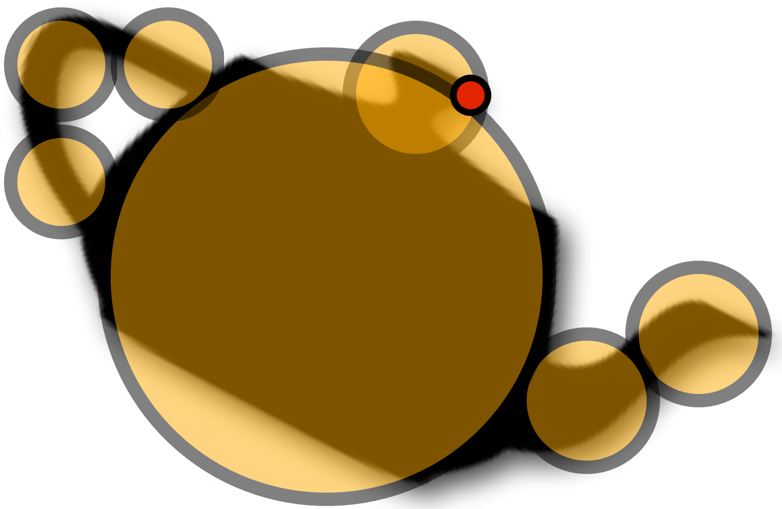
Spherical Blocker Self- & Cast- Shadows

- accumulate spherical blocker occlusion for both:
  - dynamic object *self-occlusion*

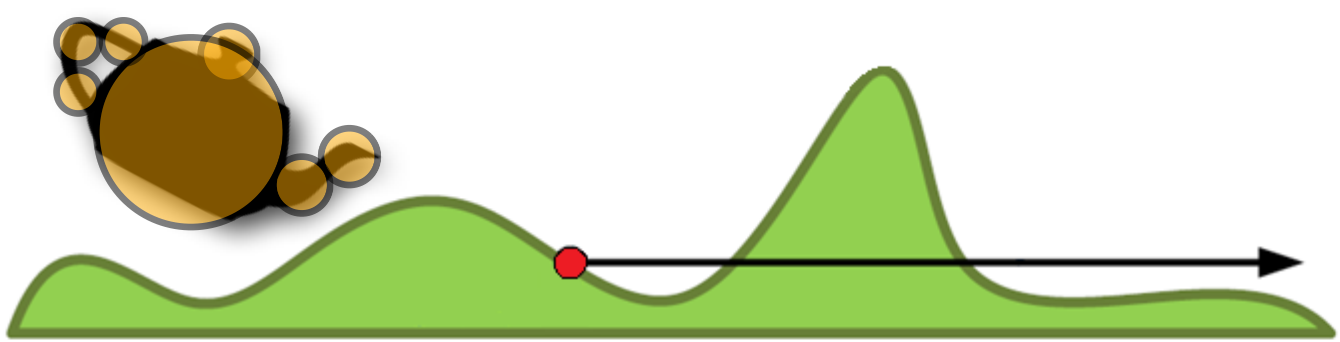


Spherical Blocker Self- & Cast- Shadows

- accumulate spherical blocker occlusion for both:
  - dynamic object *self-occlusion*



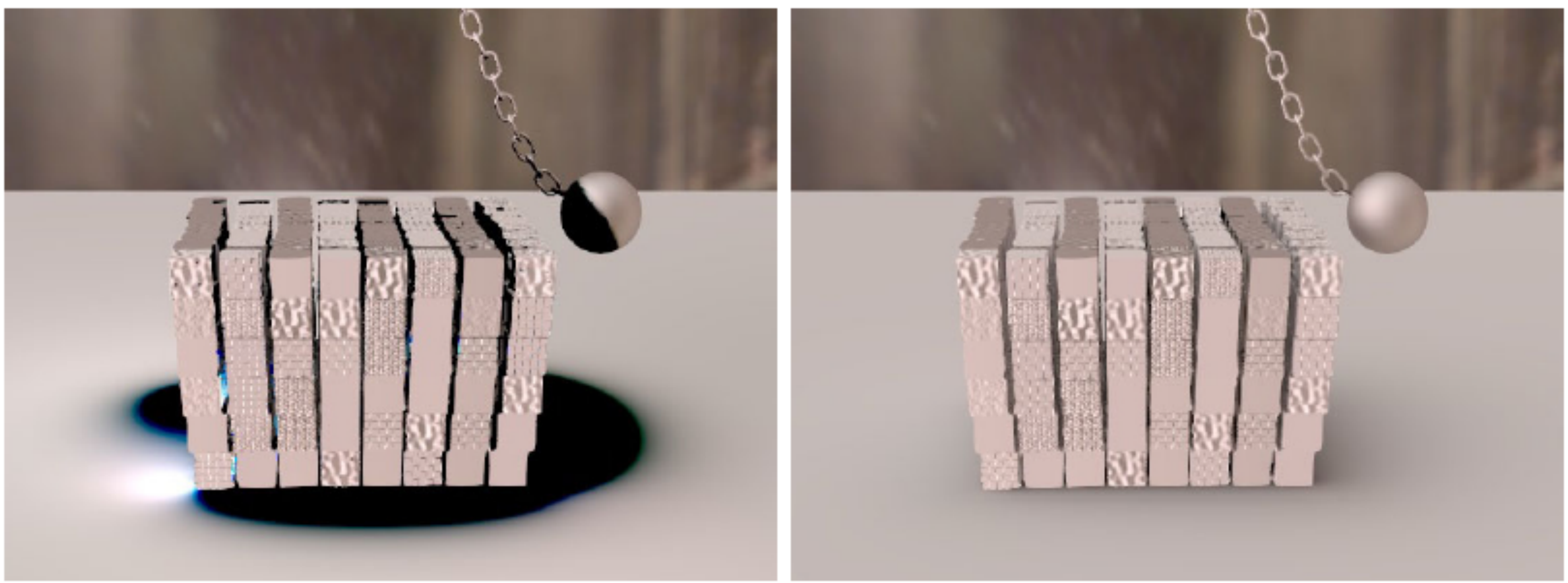
  - **and** dynamic object *cast-occlusion* onto the HF



Ratio Attenuation

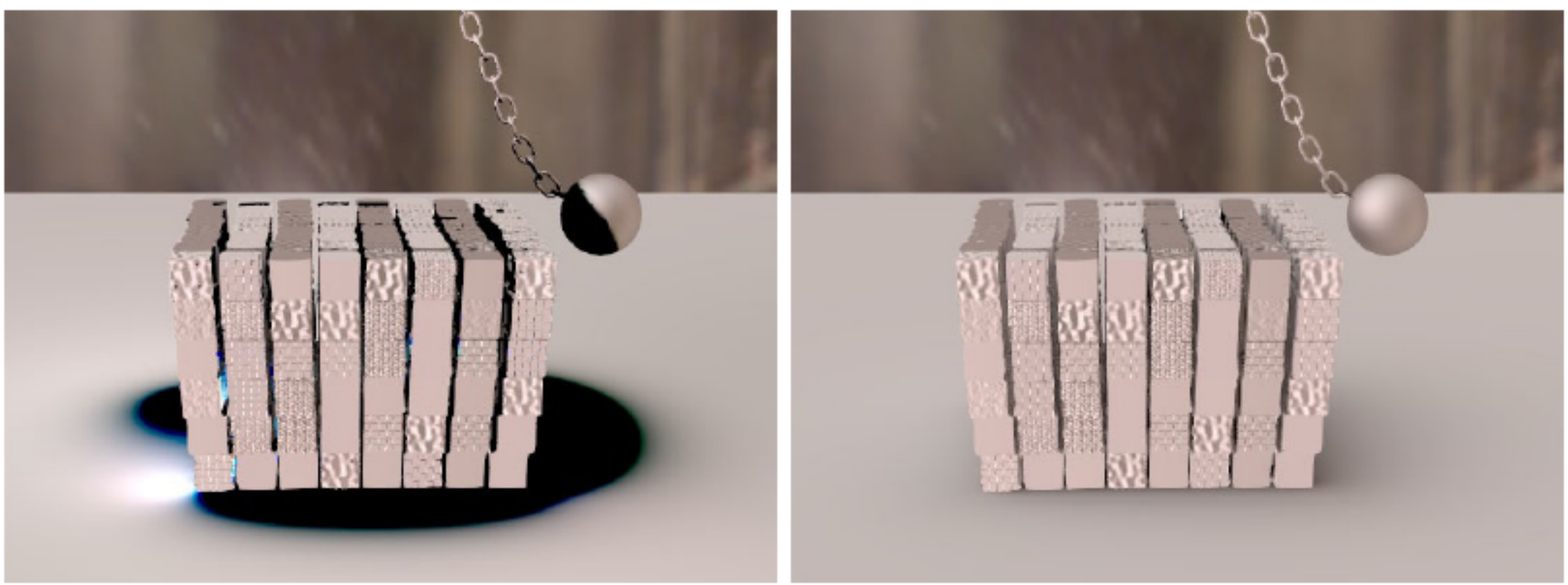
Ratio Attenuation

- SH exponentiation suffers from accumulation error when there are **many overlapping** blocker spheres



Ratio Attenuation

- SH exponentiation suffers from accumulation error when there are **many overlapping** blocker spheres
- we reduce accumulation error by:
  - weighting log-SH visibility by blocker solid angle, and
  - only accumulating blockers in upper shading hemisphere



Summary of Main Ideas

1. compute *HF self-visibility* (in *log-SH space*)

2. compute *HF cast-visibility* (onto meshes)

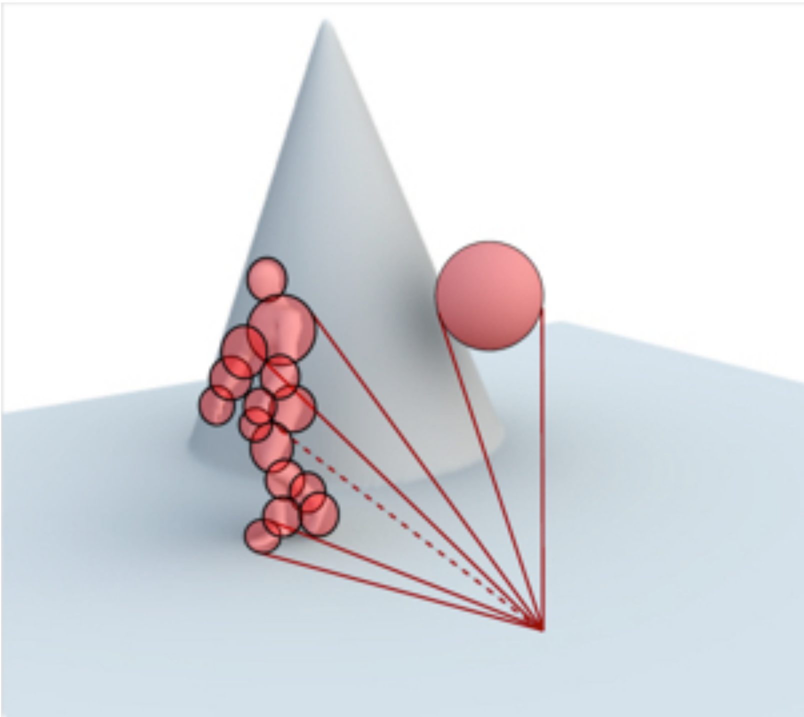
3. compute *mesh cast-visibility* (onto HF) **and** *self-visibility*

4. accumulate total spherical visibility
   - combine **per-slice** HF (log) visibility to form **full** spherical visibility **[NS09]**
   - ***accumulate*** dynamic mesh blocker log-visibility and HF log-visibility
   - perform **SH exponentiation**

5. compute log-SH BRDF and perform final shading

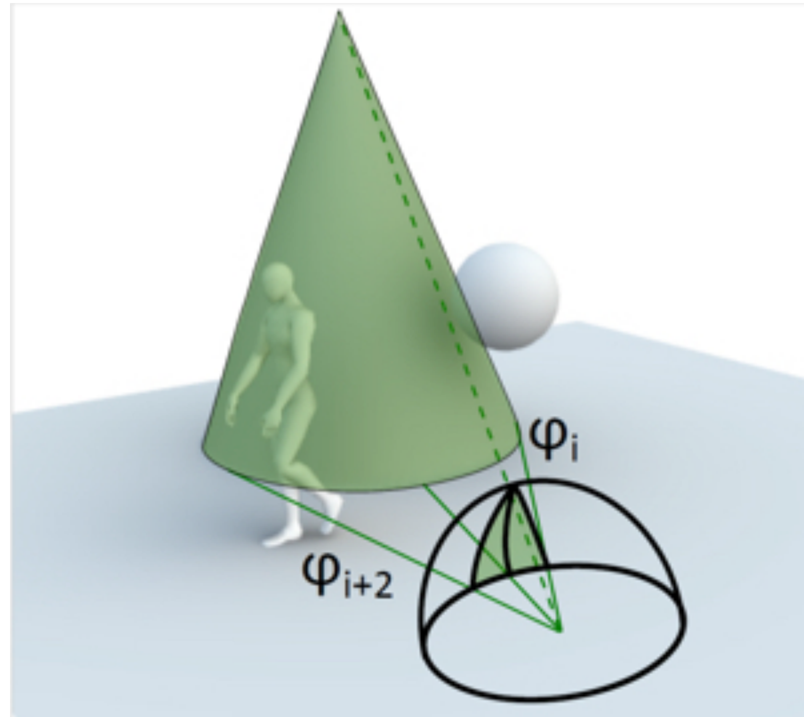
Accumulate Log-SH Visibility

Given spherical log-SH visibility for



dynamic blocker “meshes”

$$\left\{\mathbf{v}_{\log}^{0}, \mathbf{v}_{\log}^{1}, \cdots, \mathbf{v}_{\log}^{\mathrm{B-1}}\right\}$$

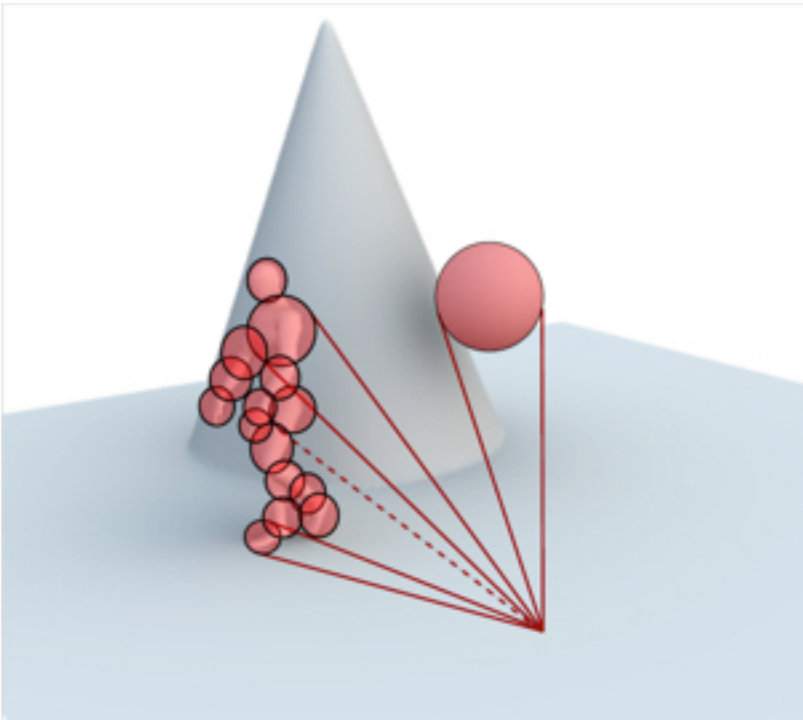


dynamic height field geometry

$$\mathbf{v}_{\log}^{\mathrm{HF}}$$

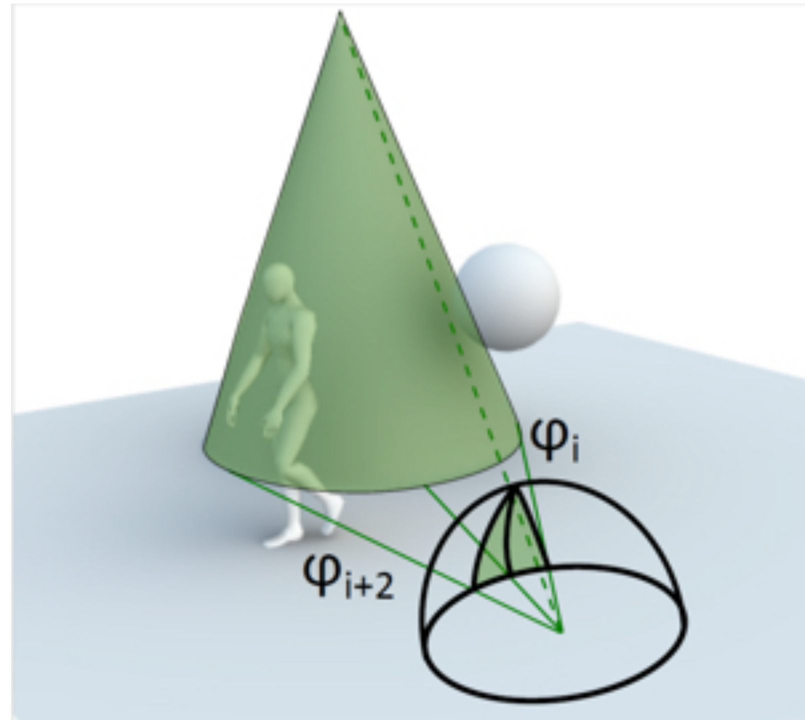
- the **total** log-SH visibility vector is $\mathbf{V}_{\log} = \mathbf{v}_{\log}^{\mathrm{HF}} + \sum_{b=0}^{B-1} \mathbf{v}_{\log}^{\mathrm{b}}$

Given spherical log-SH visibility for



dynamic blocker "meshes"

$$\{\mathbf{v}_{\log}^{0}, \mathbf{v}_{\log}^{1}, \cdots, \mathbf{v}_{\log}^{\mathrm{B\text{-}1}}\}$$



dynamic height field geometry

$$\mathbf{v}_{\log}^{\mathrm{HF}}$$

- the **total** log-SH visibility vector is $\mathbf{V}_{\log} = \mathbf{v}_{\log}^{\mathrm{HF}} + \sum_{b=0}^{B-1} \mathbf{v}_{\log}^{\mathrm{b}}$

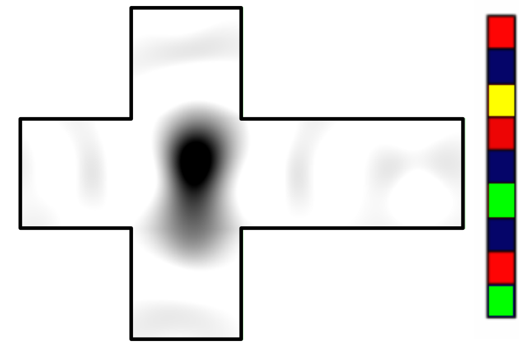
$$\mathbf{V} = \exp\left(\mathbf{V}_{\log}\right) \approx \mathbf{1} + \mathbf{V}_{\log} + \frac{{\mathbf{V}_{\log}}^{2}}{2} + \frac{{\mathbf{V}_{\log}}^{3}}{3!} + \cdots$$

Summary of Main Ideas

1. compute *HF self-visibility* (in *log-SH space*)
2. compute *HF cast-visibility* (onto meshes)
3. compute *mesh cast-visibility* (onto HF) **and** *self-visibility*
4. accumulate total spherical visibility
5. compute log-SH BRDF and perform final shading
   - simplify triple-product shading to double-product shading
   - ***formulate*** view-evaluated BRDF in log-SH space
   - ***accumulate*** BRDF with multi-product visibility

Traditional Triple Product SH Shading

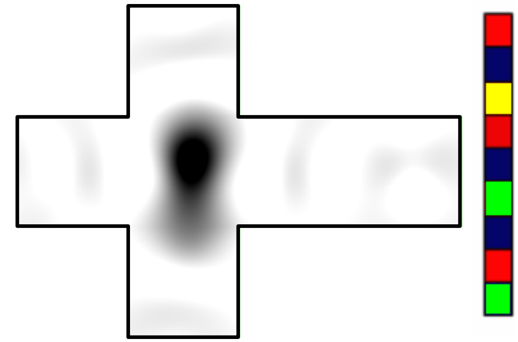
$$\mathbf{V} = \exp\left(\mathbf{V}_{\log}\right)$$ spherical SH visibility



Traditional Triple Product SH Shading

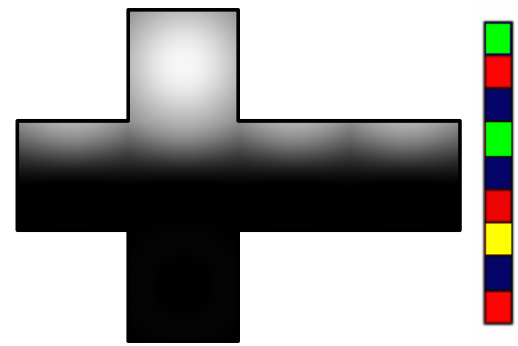
$$\mathbf{V} = \exp\left(\mathbf{V}_{\log}\right)$$

spherical SH visibility



$$\mathbf{f}_r(\omega_o)$$

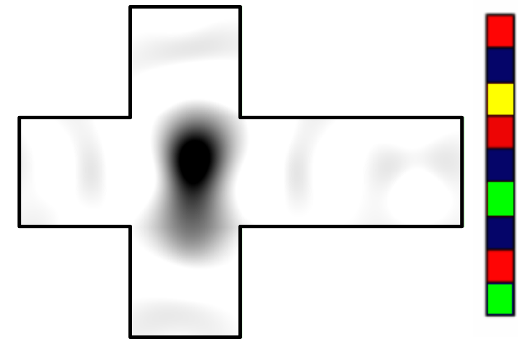
view-evaluated BRDF



Traditional Triple Product SH Shading

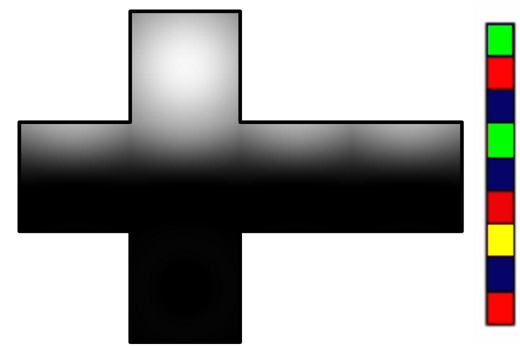
$$\mathbf{V} = \exp\left(\mathbf{V}_{\log}\right)$$

spherical SH visibility



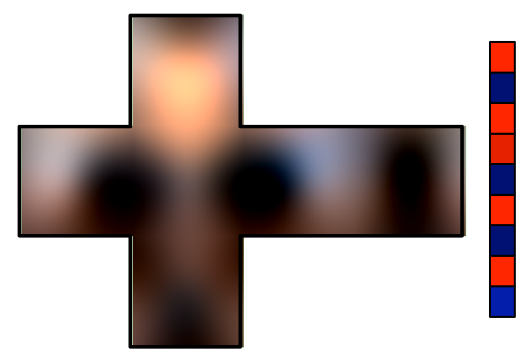
$$\mathbf{f}_r(\omega_o)$$

view-evaluated BRDF

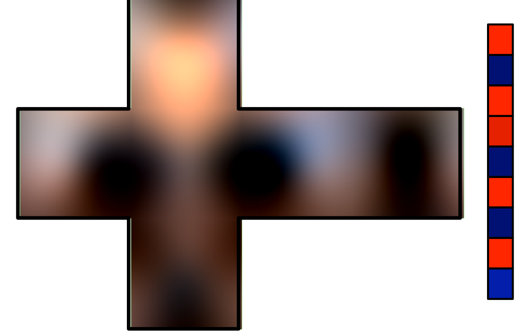


$$\mathbf{L}_e$$

lighting environment



$\mathbf{L}_e$ lighting environment



- final shading traditionally ([**RWS*06;SGNS07**]) computed with triple-product SH integration:

$$L_o(\omega_o) = \sum_{ijk} [\mathbf{L}_e]_i \, [\mathbf{V}]_j \, [\mathbf{f}_r(\omega_o)]_k \, \Gamma_{ijk}$$

where

$$\Gamma_{ijk} = \int_{S^2} y_i(\omega) \, y_j(\omega) \, y_k(\omega) \, \mathrm{d}\omega$$

are the SH *tripling coefficients*, a sparse order-3 tensor.

- Triple product shading computation is still costly!

Log-BRDF Shading

- We already use log-space to perform a multi-product

$$\mathbf{V} = \exp\left(\mathbf{V}_{\log}\right) \approx \prod_{b=0}^{B-1} \mathbf{V}_b$$

Log-BRDF Shading

- We already use log-space to perform a multi-product

$$\mathbf{V} = \exp\left(\mathbf{V}_{\log}\right) \approx \prod_{b=0}^{B-1} \mathbf{V}_b$$

- Triple-product shading composes the BRDF-weighted visibility (***transfer***) in the primal domain with a product

$$\mathbf{T} = \mathbf{f}_r(\omega_o) \times \prod_{b=0}^{B-1} \mathbf{V}_b$$

Log-BRDF Shading

- We already use log-space to perform a multi-product

$$\mathbf{V} = \exp\left(\mathbf{V}_{\log}\right) \approx \prod_{b=0}^{B-1} \mathbf{V}_b$$

- Triple-product shading composes the BRDF-weighted visibility (***transfer***) in the primal domain with a product

$$\mathbf{T} = \mathbf{f}_r(\omega_o) \times \prod_{b=0}^{B-1} \mathbf{V}_b$$

- <u>Idea</u>: use log-space to compose transfer with a **sum**

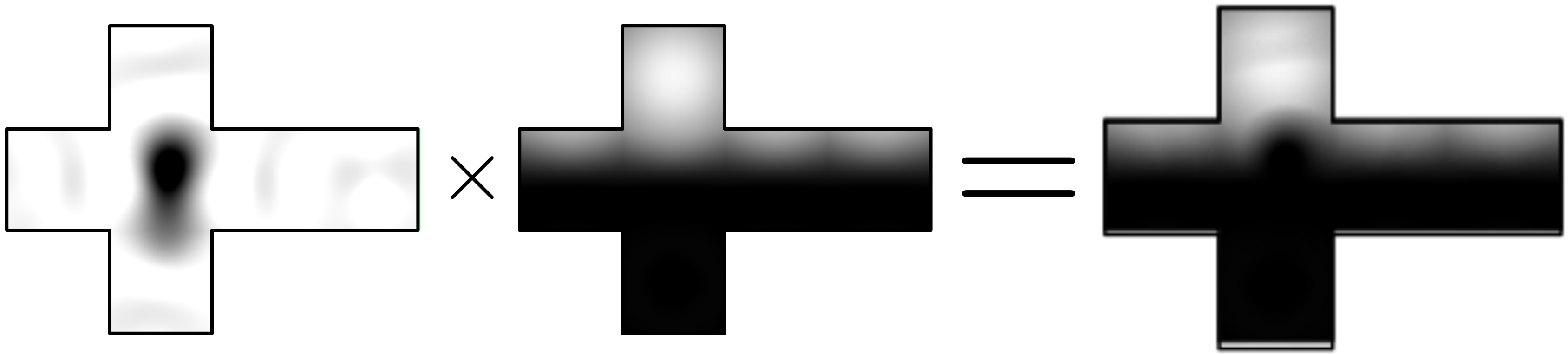
- <u>Idea</u>: use log-space to compose transfer with a **sum**

$$\mathbf{T} = \exp\left(\left[\mathbf{f}_r(\omega_o)\right]_{\log} + \mathbf{V}_{\log}\right) \approx \mathbf{f}_r(\omega_o) \times \prod_{b=0}^{B-1} \mathbf{V}_b$$

- <u>Idea</u>: use log-space to compose transfer with a **sum**

$$\mathbf{T} = \exp\left(\left[\mathbf{f}_r(\omega_o)\right]_{\log} + \mathbf{V}_{\log}\right) \approx \mathbf{f}_r(\omega_o) \times \prod_{b=0}^{B-1} \mathbf{V}_b$$

$\mathbf{T}$

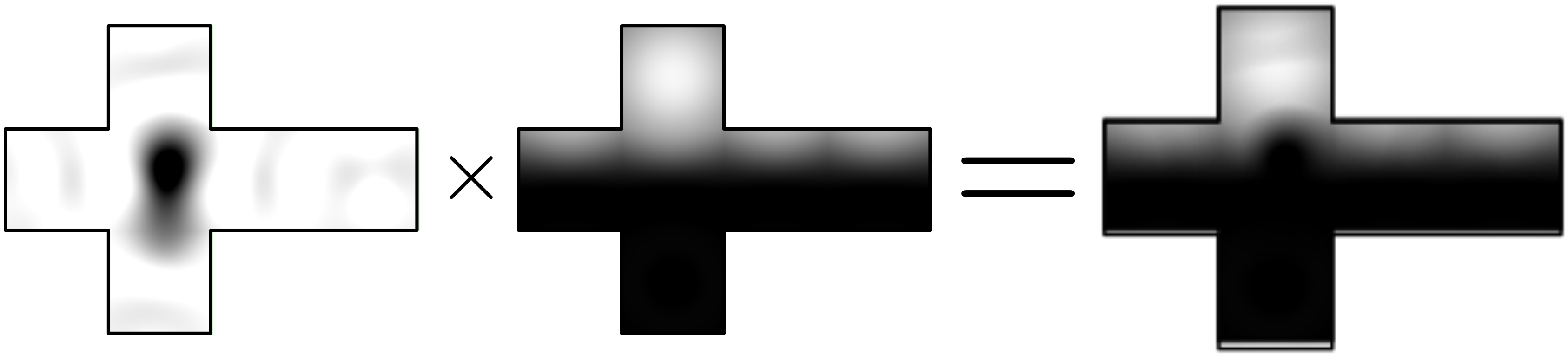


SH transfer

- Idea: use log-space to compose transfer with a **sum**

$$\mathbf{T} = \exp\left(\left[\mathbf{f}_r(\omega_o)\right]_{\log} + \mathbf{V}_{\log}\right) \approx \mathbf{f}_r(\omega_o) \times \prod_{b=0}^{B-1} \mathbf{V}_b$$

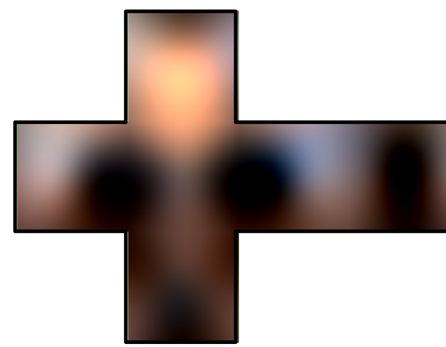
$\mathbf{T}$



SH transfer

$\mathbf{L}_e$

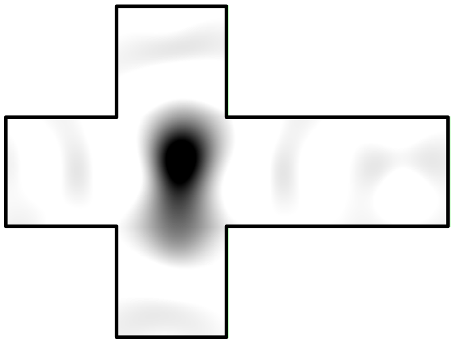
lighting environment



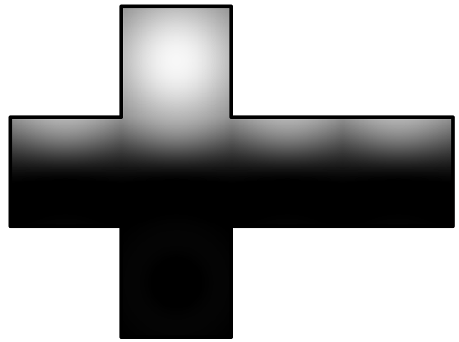
- Idea: use log-space to compose transfer with a **sum**

$$\mathbf{T} = \exp\left(\left[\mathbf{f}_r(\omega_o)\right]_{\log} + \mathbf{V}_{\log}\right) \approx \mathbf{f}_r(\omega_o) \times \prod_{b=0}^{B-1} \mathbf{V}_b$$

$\mathbf{T}$



$\times$

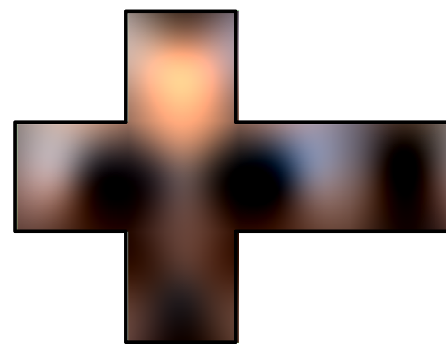


$=$

SH transfer

$\mathbf{L}_e$

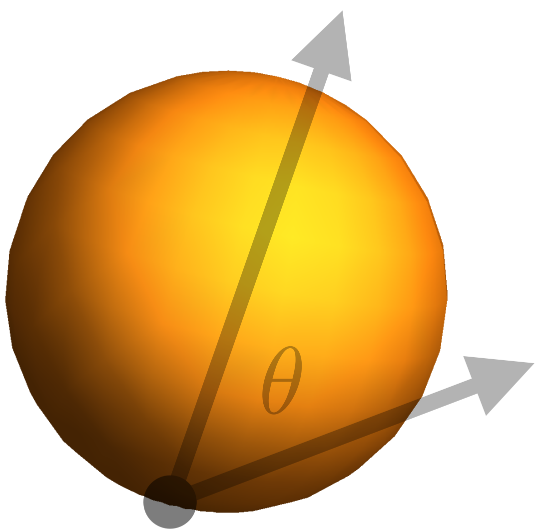
lighting environment



- Now shading requires a cheap *double-product* SH integral!
  - but how do we compute the ***log-BRDF*** *SH coefficients*?

Log-BRDF SH Coefficients

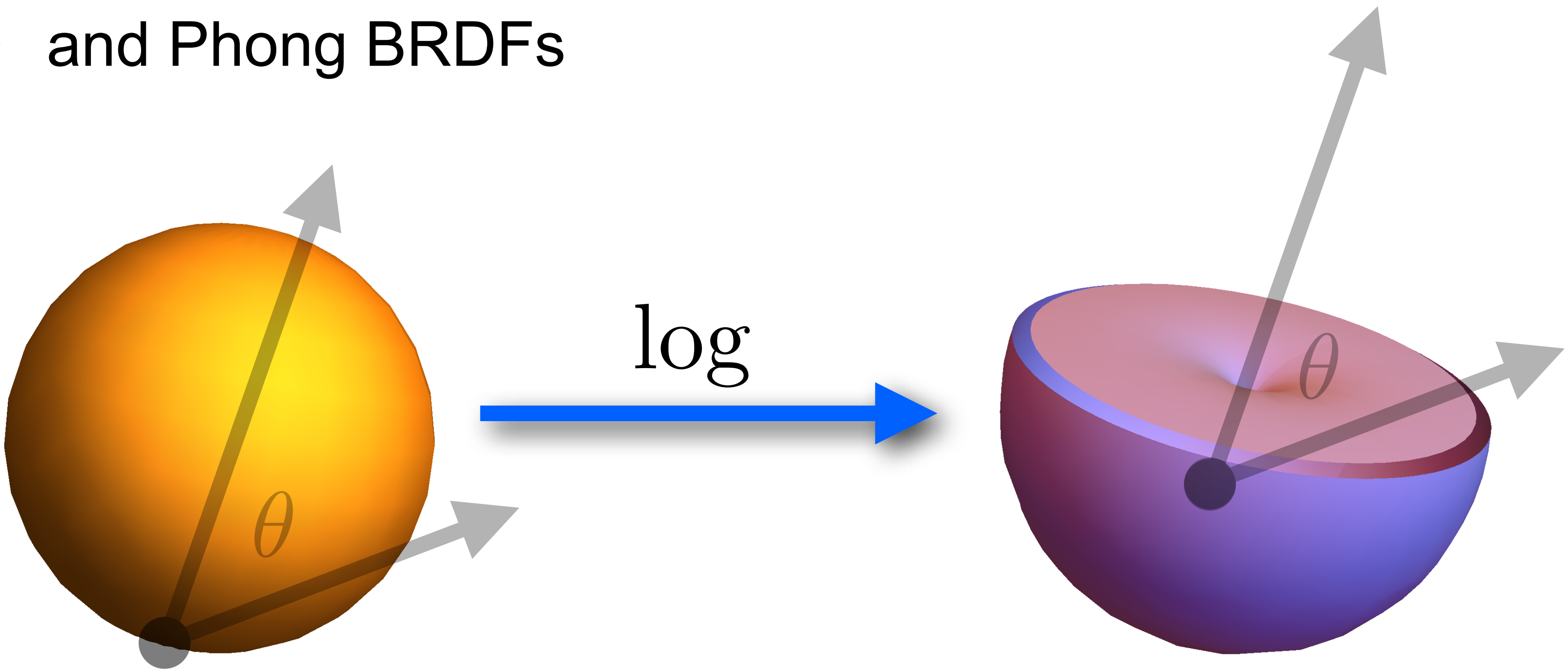
- We compute the log-**ZH** BRDF coefficients *numerically* for:
  - diffuse BRDFs,
  - and Phong BRDFs



$$f_r(\theta) = \frac{\alpha+1}{2\pi} \max(\cos^\alpha \theta, 0)$$

Log-BRDF SH Coefficients

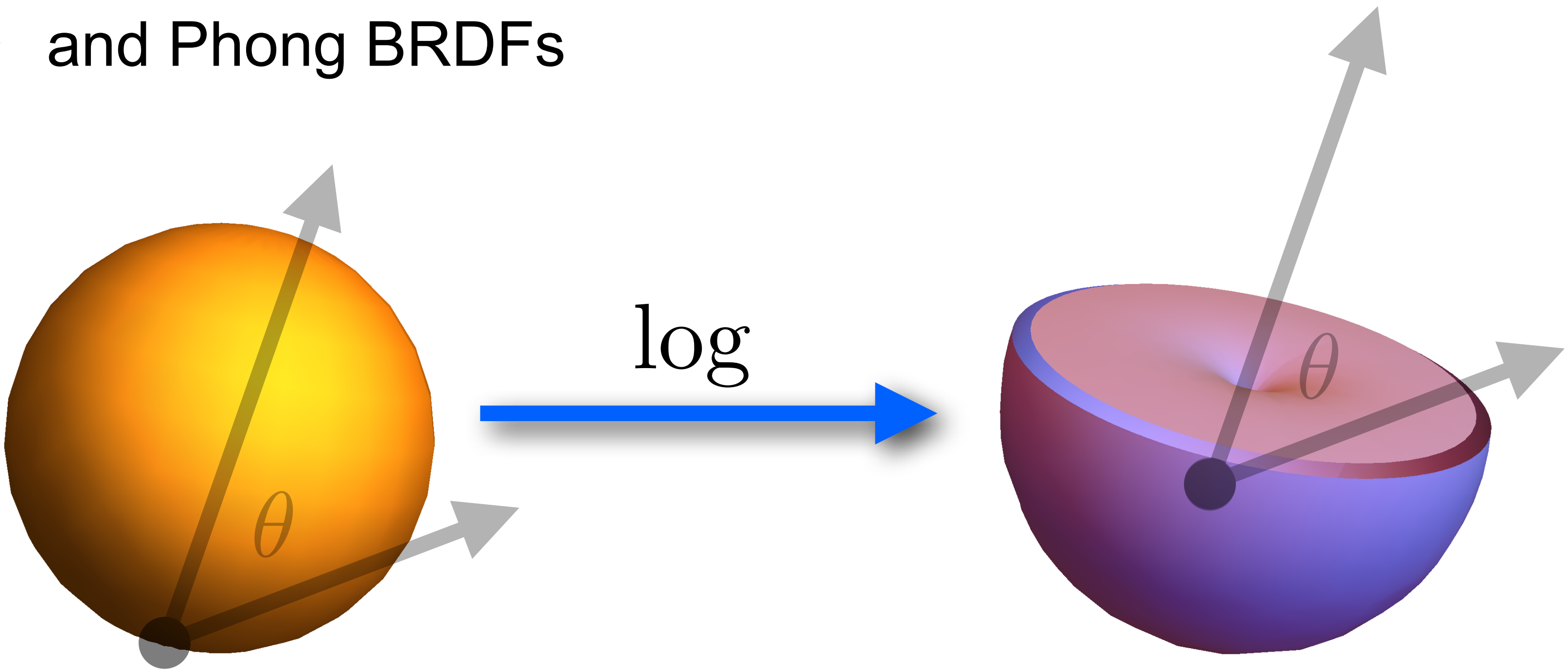
- We compute the log-**ZH** BRDF coefficients *numerically* for:
  - diffuse BRDFs,
  - and Phong BRDFs



$$f_r(\theta) = \frac{\alpha+1}{2\pi} \max(\cos^\alpha \theta, 0)$$

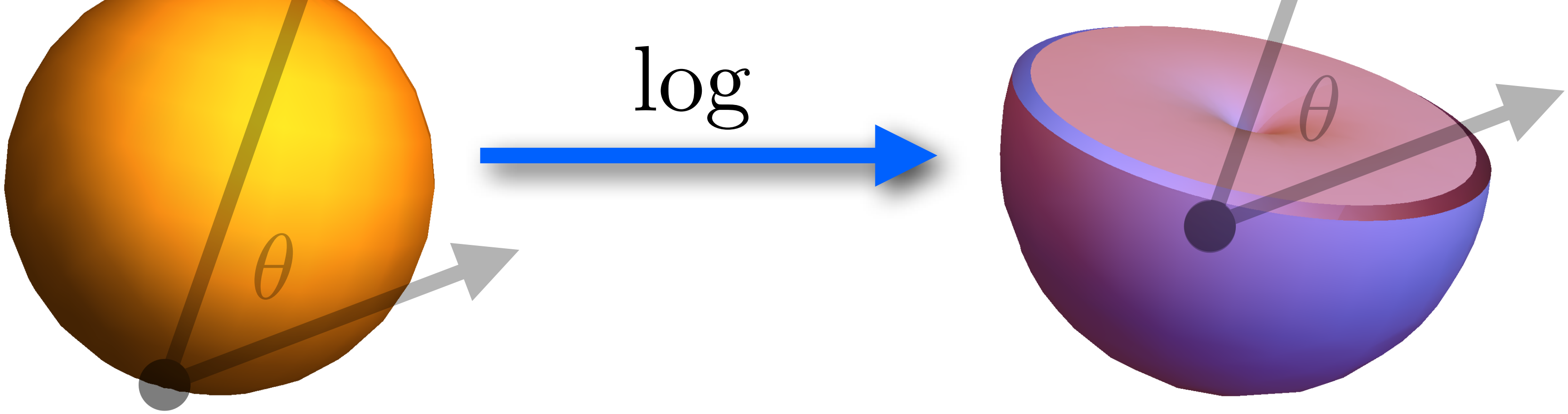
Log-BRDF SH Coefficients

- We compute the log-**ZH** BRDF coefficients *numerically* for:
  - diffuse BRDFs,
  - and Phong BRDFs



$$f_r(\theta) = \frac{\alpha+1}{2\pi}\max(\cos^\alpha\theta, 0)$$

- Need to treat hemispherical clamping carefully!



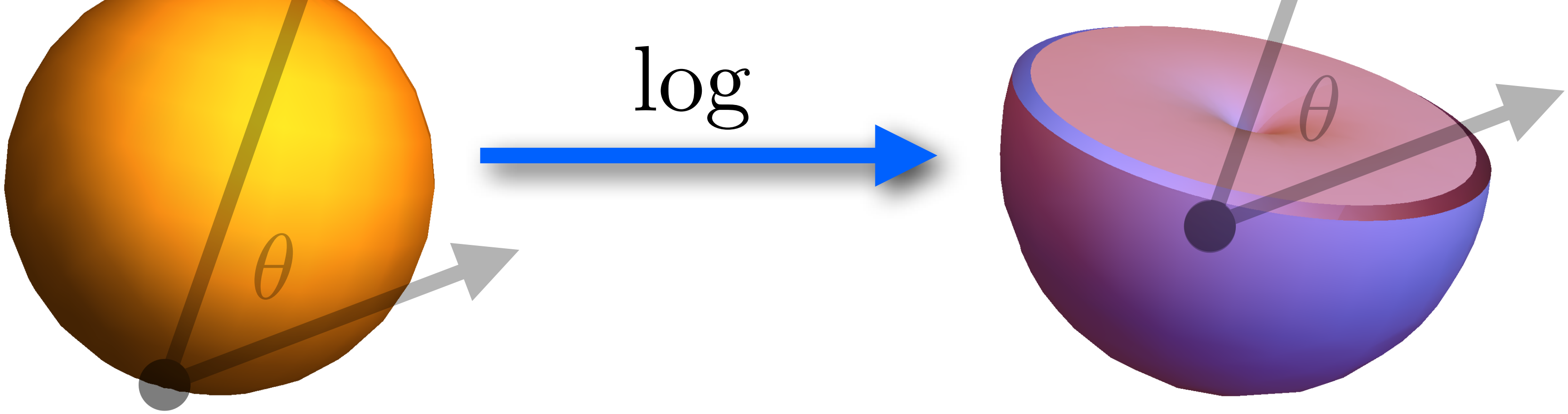
$$f_r(\theta) = \tfrac{\alpha+1}{2\pi}\max(\cos^\alpha\theta, 0)$$

- Need to treat hemispherical clamping carefully!

- Canonical-frame ZH log-BRDF coefficients are then:

$$f_{l,0}^{\log} = \int_{H^{2+}} \log\left(\tfrac{\alpha+1}{2\pi}\max(\cos^\alpha\omega_\theta, \epsilon)\right)\; y_l^0(\omega)\,\mathrm{d}\omega +$$

$$\int_{H^{2-}} (\log\epsilon)\; y_l^0(\omega)\,\mathrm{d}\omega$$



$$f_r(\theta) = \tfrac{\alpha+1}{2\pi} \max(\cos^\alpha \theta, 0)$$

- Need to treat hemispherical clamping carefully!

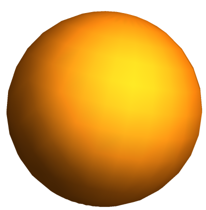
- Canonical-frame ZH log-BRDF coefficients are then:

$$f_{l,0}^{\log} = \int_{H^{2+}} \log\left(\tfrac{\alpha+1}{2\pi} \max(\cos^\alpha \omega_\theta, \epsilon)\right)\ y_l^0(\omega)\,\mathrm{d}\omega + \int_{H^{2-}} (\log \epsilon)\ y_l^0(\omega)\,\mathrm{d}\omega$$

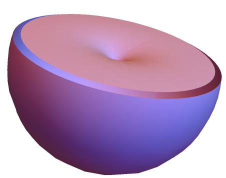
- We compute & tabulate order-4 ZH coefficients numerically

Log-BRDF Error

- In a ***worse case*** lighting scenario, log-SH BRDF shading still maintains a cosine-like fall-off profile



SH

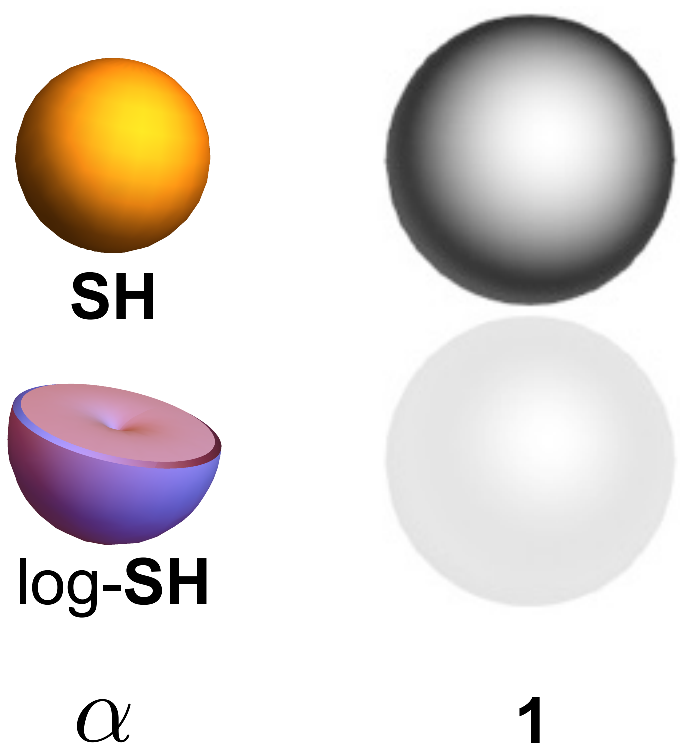


log-**SH**

α

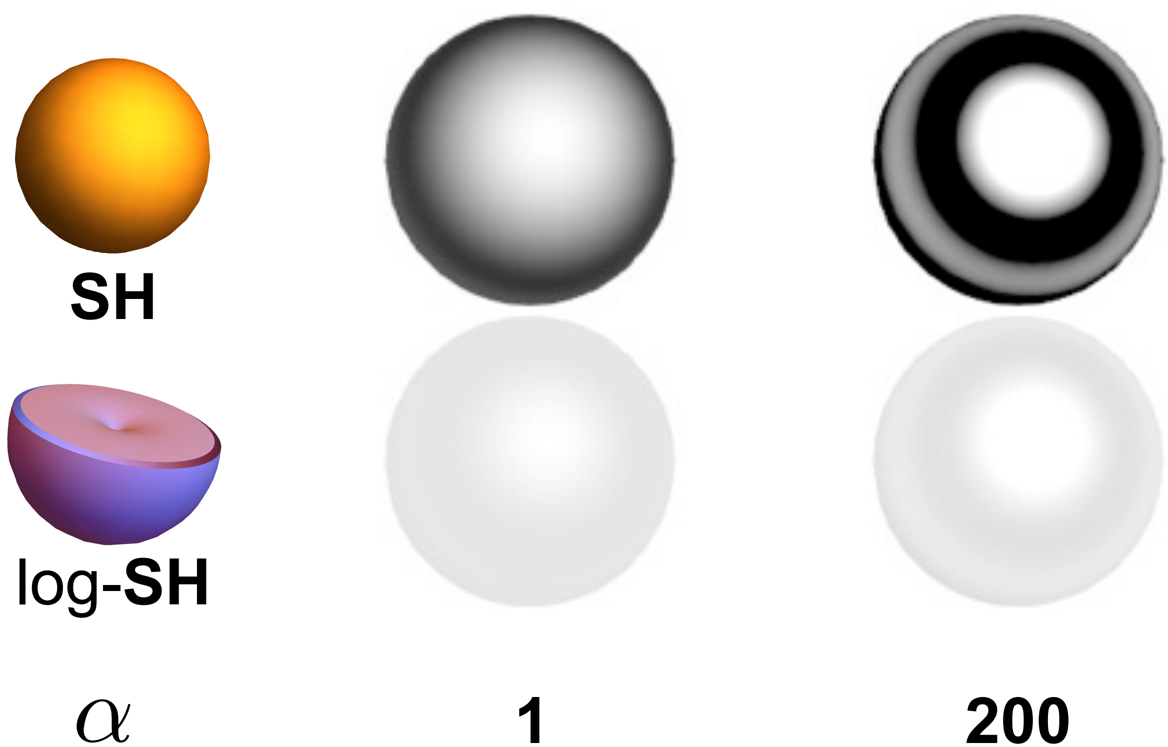
Log-BRDF Error

- In a ***worse case*** lighting scenario, log-SH BRDF shading still maintains a cosine-like fall-off profile



Log-BRDF Error

- In a ***worse case*** lighting scenario, log-SH BRDF shading still maintains a cosine-like fall-off profile



Results

Results

- Hybrid image/object-space renderer
  - spherical blockers splatted onto screen **[SGNS07]**
  - multi-resolution HF ray-marching in HF object-space
  - rendered at 960 x 540 with (avg.) pixel coverage of 83%.

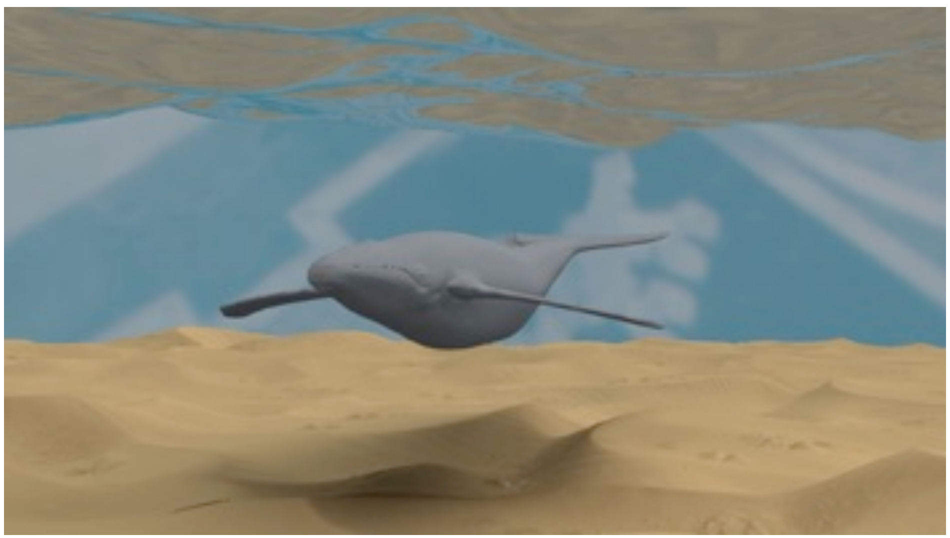
Results

- Hybrid image/object-space renderer
  - spherical blockers splatted onto screen [**SGNS07**]
  - multi-resolution HF ray-marching in HF object-space
  - rendered at 960 x 540 with (avg.) pixel coverage of 83%.

Wrecking Ball	Whale in Ocean	Cone Man
402 blockers + HF	50 blockers + HF	25 blockers + HF
15Hz (GTX 480)	10 Hz (GTX 480)	68 Hz (GTX 480)

Results

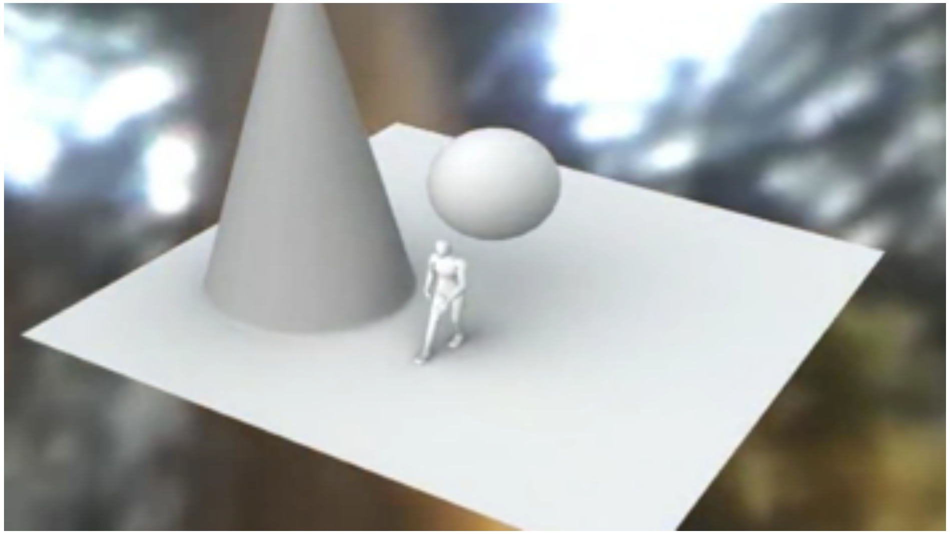
- Hybrid image/object-space renderer
  - spherical blockers splatted onto screen [**SGNS07**]
  - multi-resolution HF ray-marching in HF object-space
  - rendered at 960 x 540 with (avg.) pixel coverage of 83%.

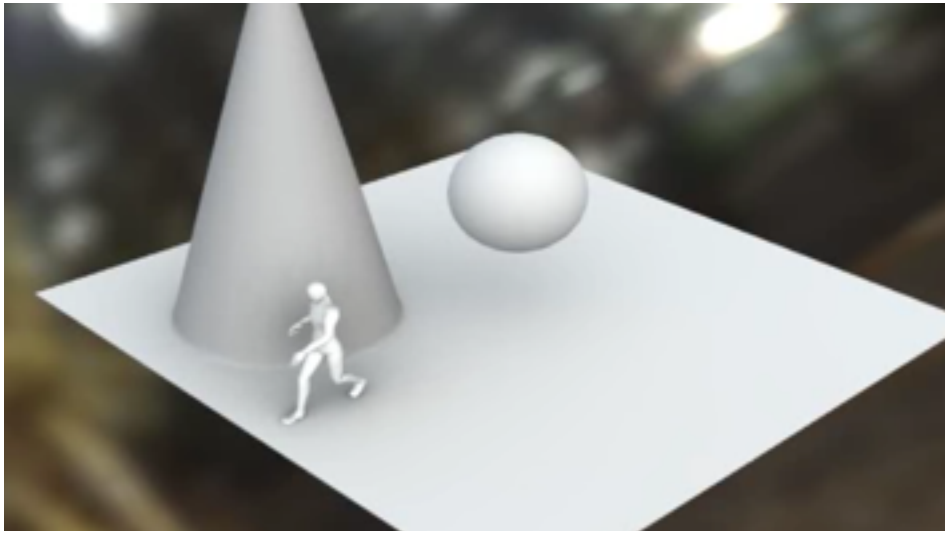


Wrecking Ball	Whale in Ocean	Cone Man
402 blockers + HF	50 blockers + HF	25 blockers + HF
15Hz (GTX 480)	10 Hz (GTX 480)	68 Hz (GTX 480)

Results

- Hybrid image/object-space renderer
  - spherical blockers splatted onto screen [SGNS07]
  - multi-resolution HF ray-marching in HF object-space
  - rendered at 960 x 540 with (avg.) pixel coverage of 83%.





Wrecking Ball	Whale in Ocean	Cone Man
402 blockers + HF	50 blockers + HF	25 blockers + HF
15Hz (GTX 480)	10 Hz (GTX 480)	68 Hz (GTX 480)

Conclusions

Conclusions

- Combine *soft shadowing* from environment lighting for scenes with **dynamic blockers** and **dynamic HFs**

Conclusions

- Combine *soft shadowing* from environment lighting for scenes with **dynamic blockers** and **dynamic HFs**
- Extend multi-resolution marching to non-HF objects
  - offset marching and infinite plane assumption

Conclusions

- Combine *soft shadowing* from environment lighting for scenes with **dynamic blockers** and **dynamic HFs**
- Extend multi-resolution marching to non-HF objects
  - offset marching and infinite plane assumption
- Novel log-SH visibility composition for HF slices
  - analytic Legendre polynomial coefficients for log-visibility elevation functions

Conclusions

- Combine *soft shadowing* from environment lighting for scenes with **dynamic blockers** and **dynamic HFs**

- Extend multi-resolution marching to non-HF objects
  - offset marching and infinite plane assumption

- Novel log-SH visibility composition for HF slices
  - analytic Legendre polynomial coefficients for log-visibility elevation functions

- Propose Log-SH BRDF formulation to reduce triple-product shading to double-product shading

Future Work

Future Work

- infinite plane assumption when marching non-HF elements
  - leverage negative blocking angle formulation of **[NS09]**

Future Work

- infinite plane assumption when marching non-HF elements
  - leverage negative blocking angle formulation of **[NS09]**

- analytic log-BRDF formulation with better hemi-clamping

Future Work

- infinite plane assumption when marching non-HF elements
  - leverage negative blocking angle formulation of **[NS09]**
- analytic log-BRDF formulation with better hemi-clamping
- indirect lighting accumulation in log-SH space

Future Work

- infinite plane assumption when marching non-HF elements
  - leverage negative blocking angle formulation of **[NS09]**

- analytic log-BRDF formulation with better hemi-clamping

- indirect lighting accumulation in log-SH space

- generalize geometry
  - local height field displacements
  - tiled height field representations
  - non-spherical blockers

We acknowledge the helpful suggestions of the anonymous reviewers.

Thanks! Any questions?