Real-Time High Quality Rendering

GAMES202, Lingqi Yan, UC Santa Barbara

Lecture 5:
Real-Time Environment Mapping




Announcement

- Assignment 1 has been released
  - Due in 1.5 weeks
- No class next week (traveling)
  - No streaming and no recording
  - Will resume when I'm back
- Will soon start recruiting GAMES101 graders

Last Lecture

- More on PCF and PCSS
- Variance soft shadow mapping
- MIPMAP and Summed-Area Variance Shadow Maps
- Moment shadow mapping

Today

- Finishing up on shadows
  - Distance field soft shadows
- Shading from environment lighting
  - The split sum approximation
- Shadow from environment lighting

Why Distance Field Soft Shadows

Sebastian Aaltonen
@SebAaltonen

Replying to @knarkowicz @aras_p and 2 others

SDF ray-traced shadows are faster than shadow maps. The only thing limiting Fortnite having 100% SDF shadows is the memory cost of having high res SDF per object and skinned characters. Thus they use 1 cascade for near shadows and SDF everywhere else.

12:20 PM · Mar 28, 2018 ·

2 Retweets **23** Likes

Sebastian Aaltonen @SebAaltonen · Mar 28, 2018

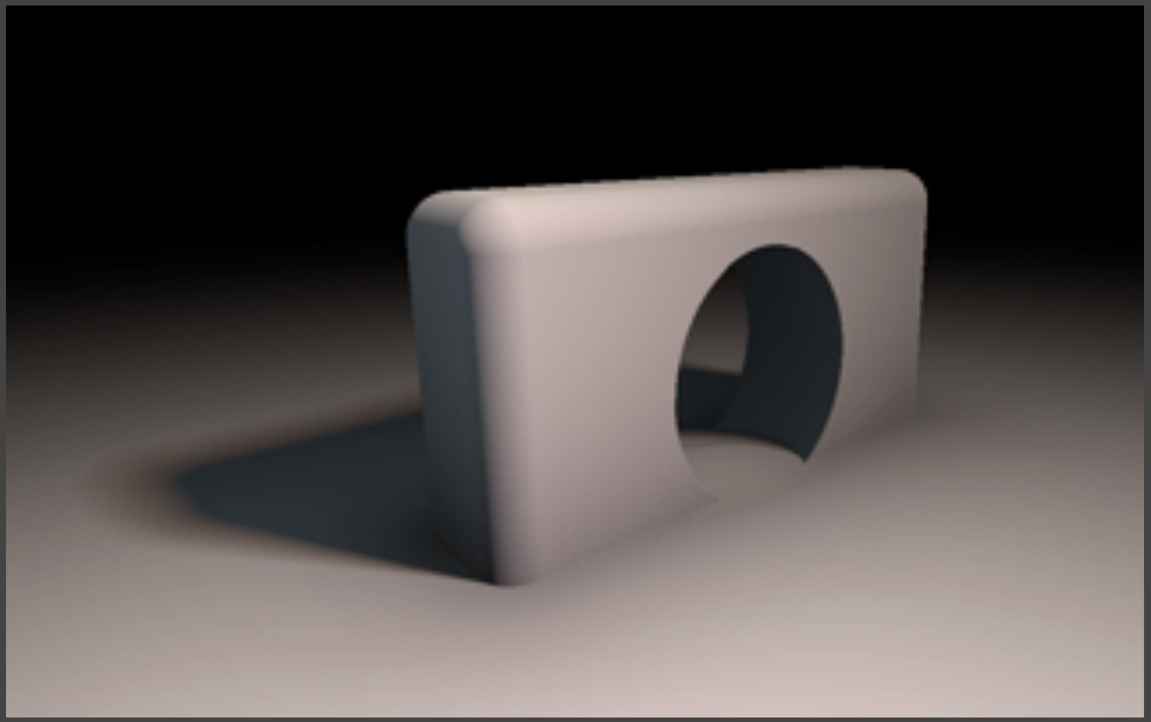
Replying to @SebAaltonen @knarkowicz and 3 others

Our tech shows that SDF shadows also work fine for dense SDF geometry at close ranges too and beat rendering equiv 10M triangle mesh to 3 shadow cascades. Also SDF shadows look way better than raster shadows with proper penumbras and no acne / undesampling / peter panning.

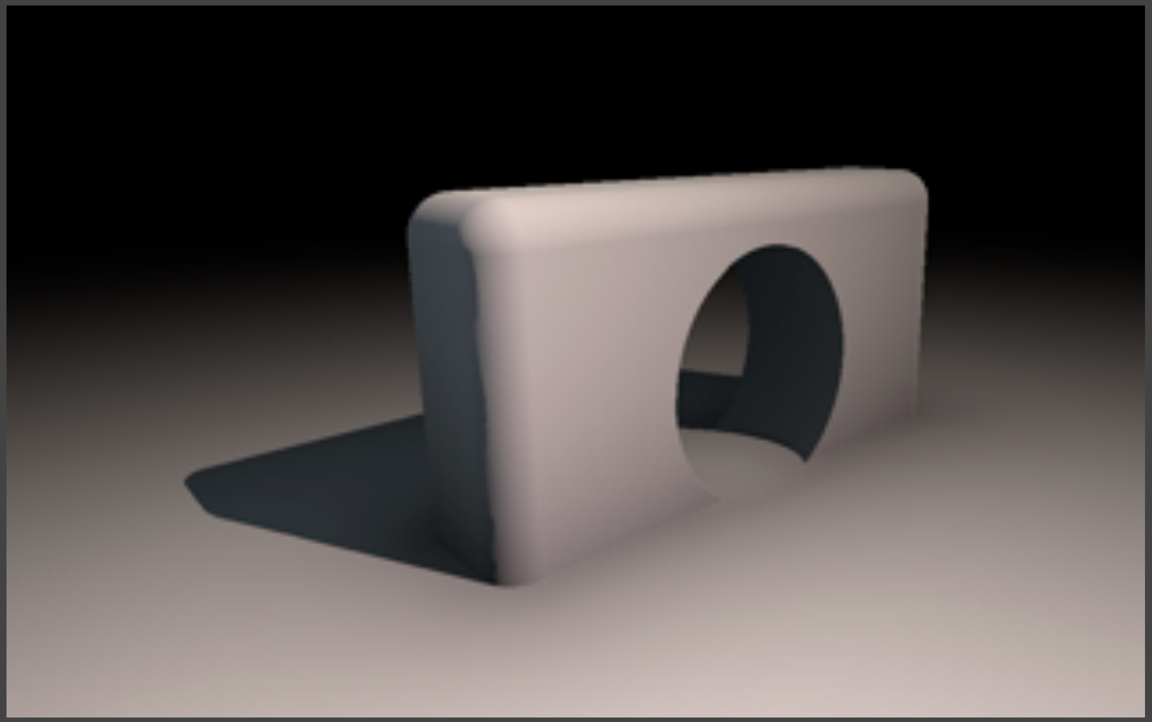
1 6

Some tweets by an indie game developer

Distance Field Soft Shadows



Soft shadow and penumbra computed using distance fields



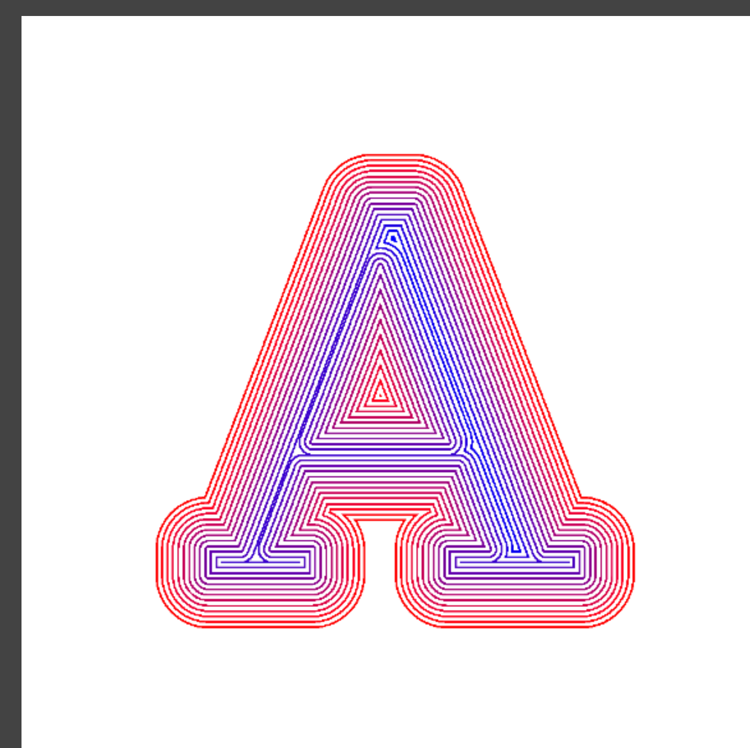
Hard shadow

https://www.iquilezles.org/www/articles/rmshadows/rmshadows.htm

From GAMES101: Distance Functions

Distance functions:

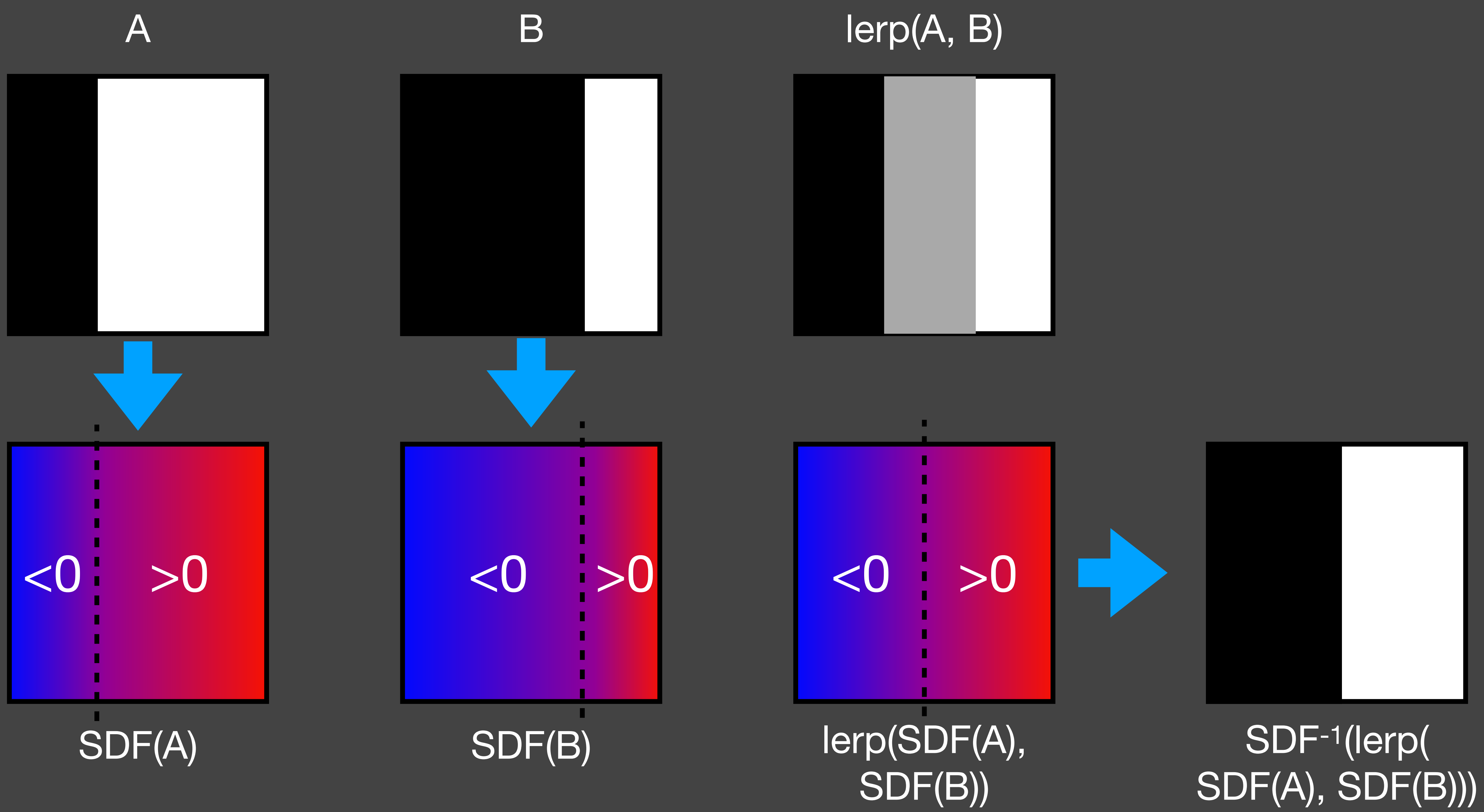
At any point, giving the minimum distance (could be **signed** distance) to the closest location on an object



https://stackoverflow.com/questions/43613256/

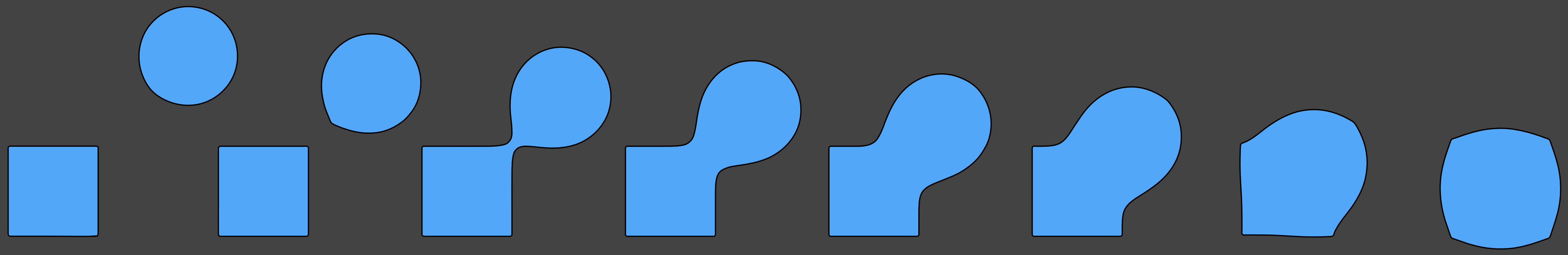
From GAMES101: Distance Functions

An Example: Blending (linear interp.) a moving boundary



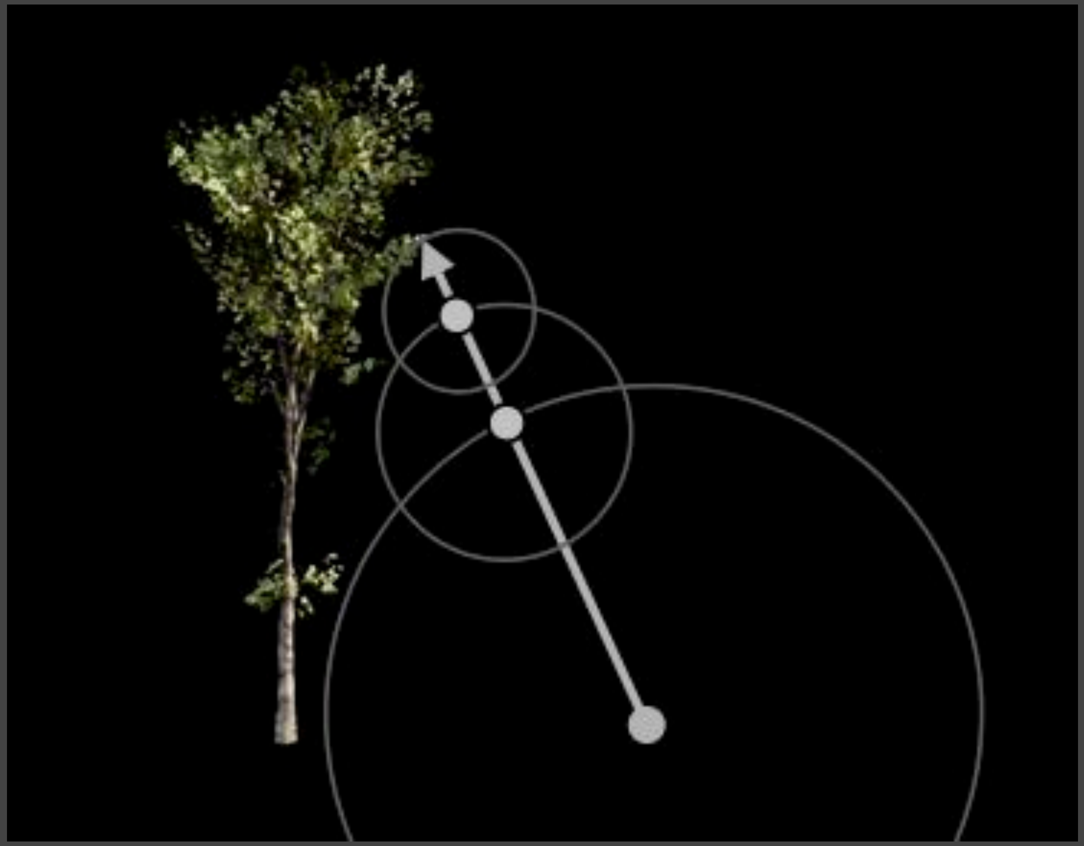
From GAMES101: Distance Functions

- Can blend any two distance functions d1, d2



The Usages of Distance Fields

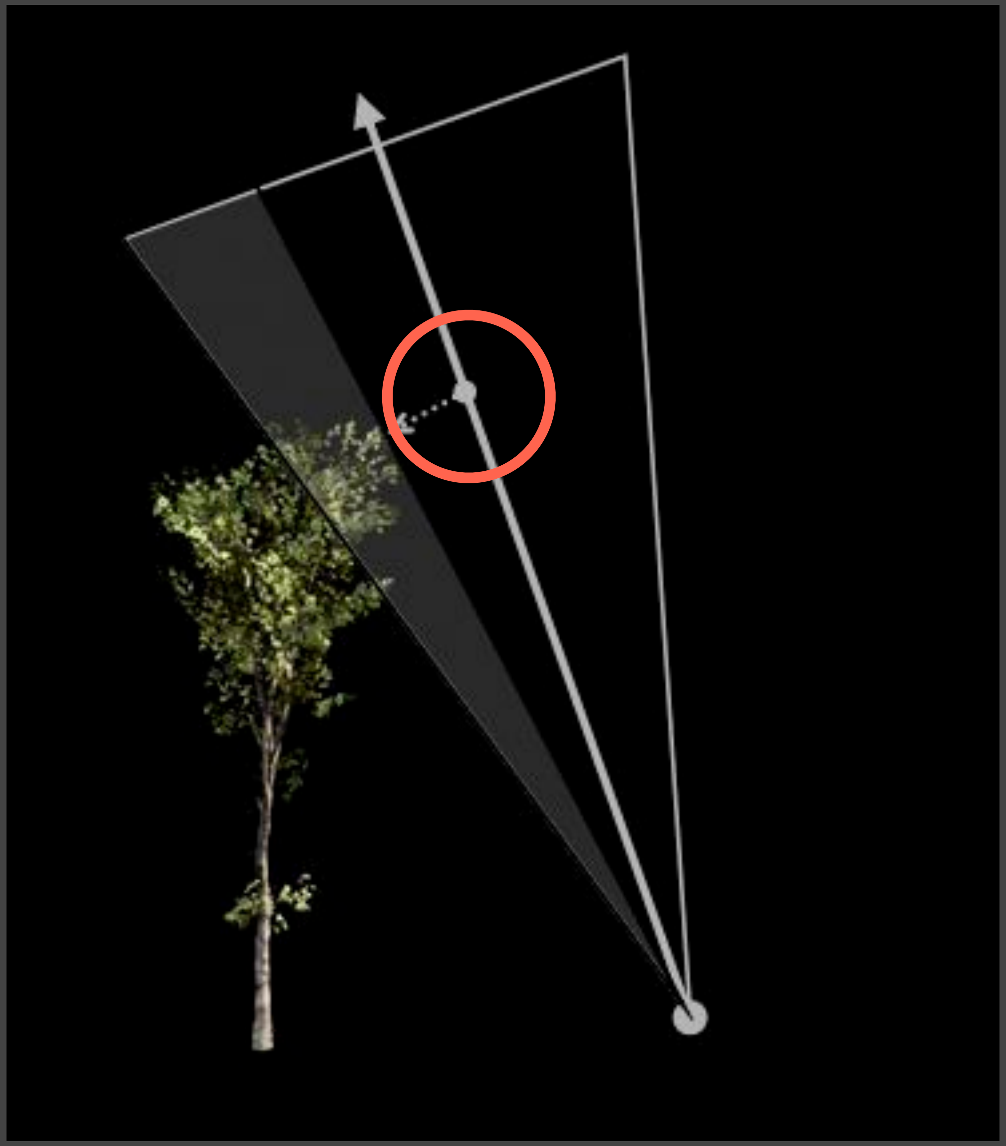
- Usage 1
  - Ray marching (sphere tracing) to perform ray-SDF intersection
  - Very smart idea behind this:
  - The value of SDF == a “safe” distance around
  - Therefore, each time at p, just travel SDF(p) distance



https://docs.unrealengine.com/en-US/BuildingWorlds/LightingAndShadows/MeshDistanceFields/index.html

The Usages of Distance Fields

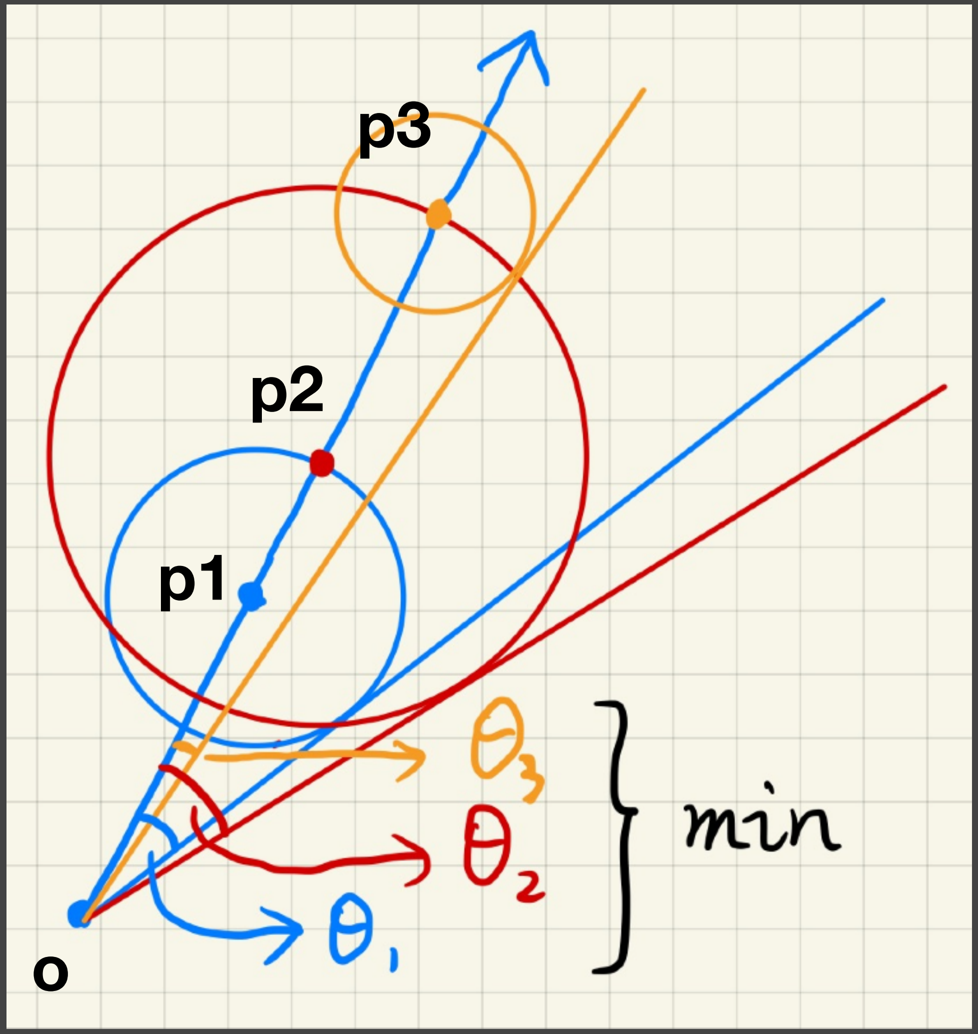
- Usage 2
  - Use SDF to determine the (approx.) percentage of occlusion
  - the value of SDF -> a “safe” angle seen from the eye
- Observation
  - Smaller “safe” angle <-> less visibility



https://docs.unrealengine.com/en-US/BuildingWorlds/LightingAndShadows/MeshDistanceFields/index.html

Distance Field Soft Shadows

- During ray matching
  - Calculate the "safe" angle from the eye at every step
  - Keep the minimum
  - How to compute the angle?

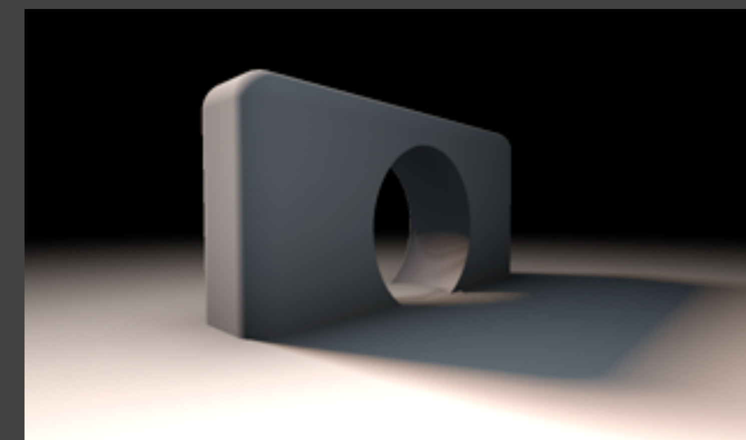


Distance Field Soft Shadows

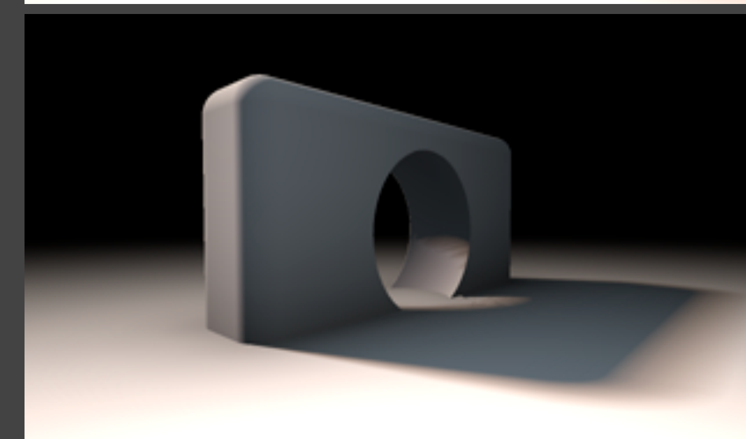
- How to compute the angle?

$$\arcsin \frac{\mathrm{SDF}(p)}{p-o} \qquad \min\left\{\frac{k\cdot \mathrm{SDF}(p)}{p-o}, 1.0\right\}$$

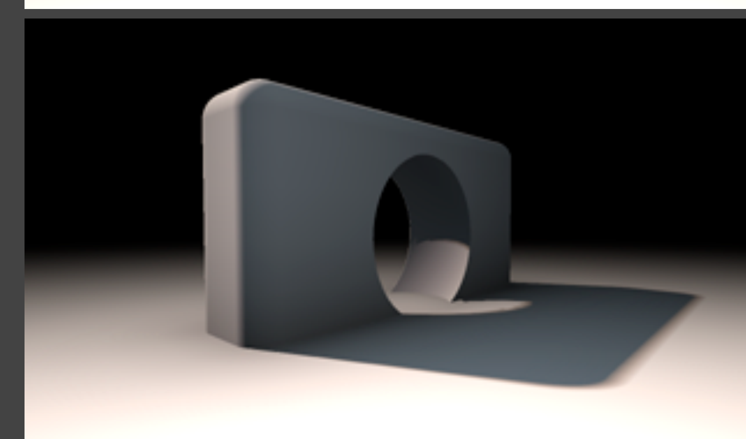
$k = 2$



$k = 8$



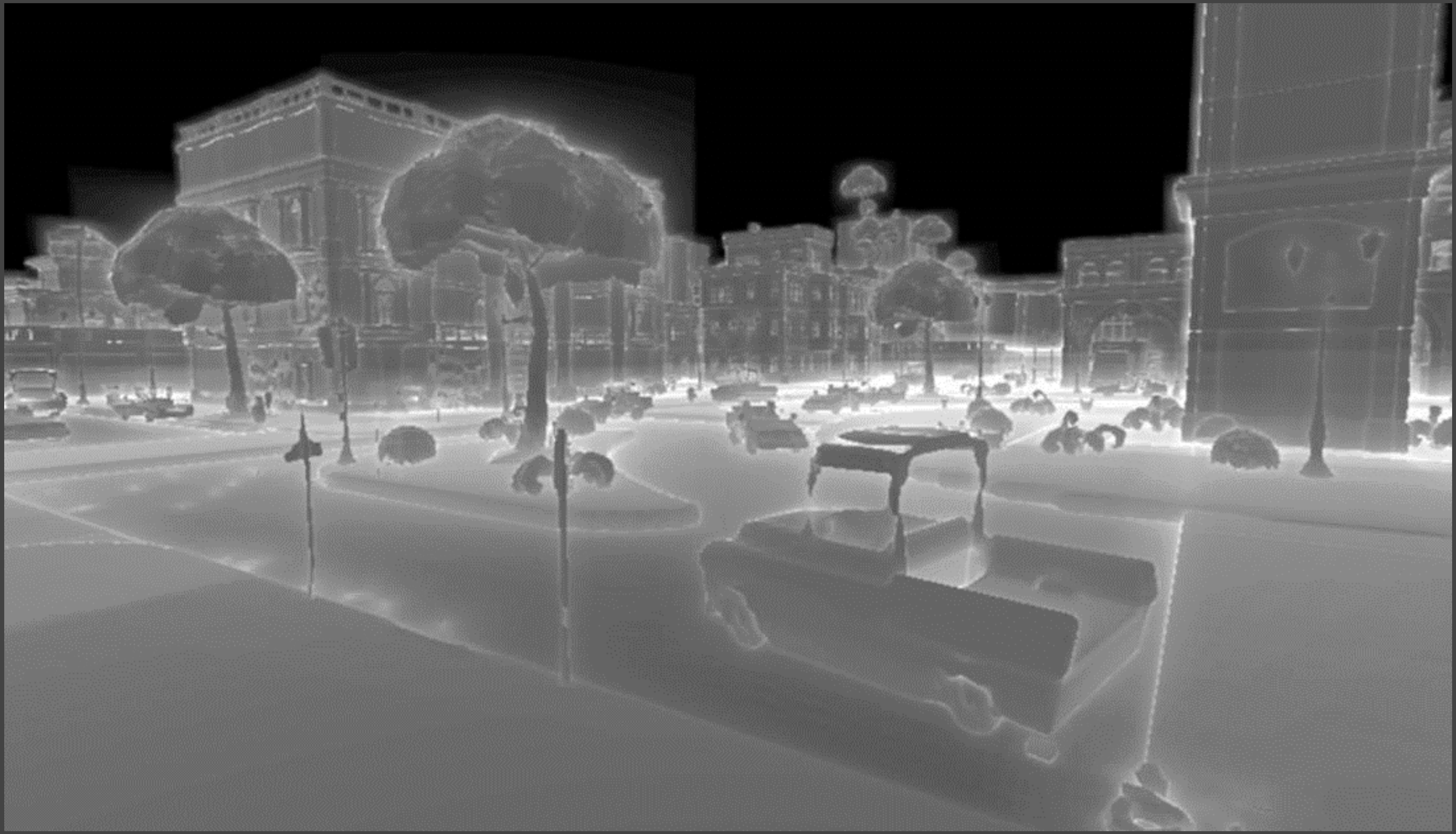
$k = 32$



- Larger k <-> earlier cutoff of penumbra <-> harder

[https://www.iquilezles.org/www/articles/rmshadows/rmshadows.htm]]

Distance Field: Visualization



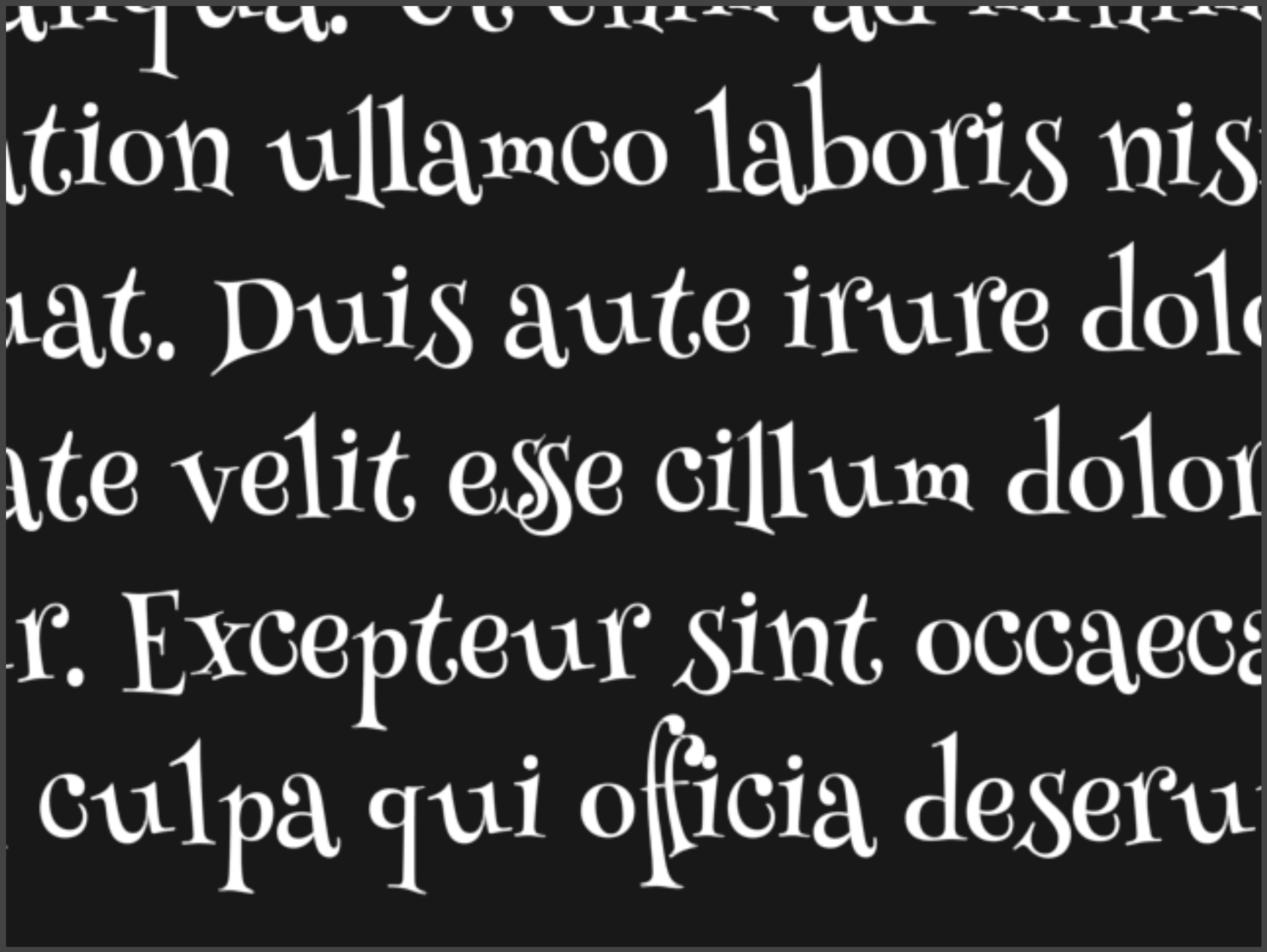
https://docs.unrealengine.com/en-US/BuildingWorlds/LightingAndShadows/MeshDistanceFields/index.html

Pros and Cons of Distance Field

- Pros
  - Fast*
  - High quality
- Cons
  - Need precomputation
  - Need heavy storage*
  - Artifact?

Another Interesting Application

- Antialiased / infinite resolution characters in RTR



https://github.com/protectwise/troika/tree/master/packages/troika-three-text

Questions?

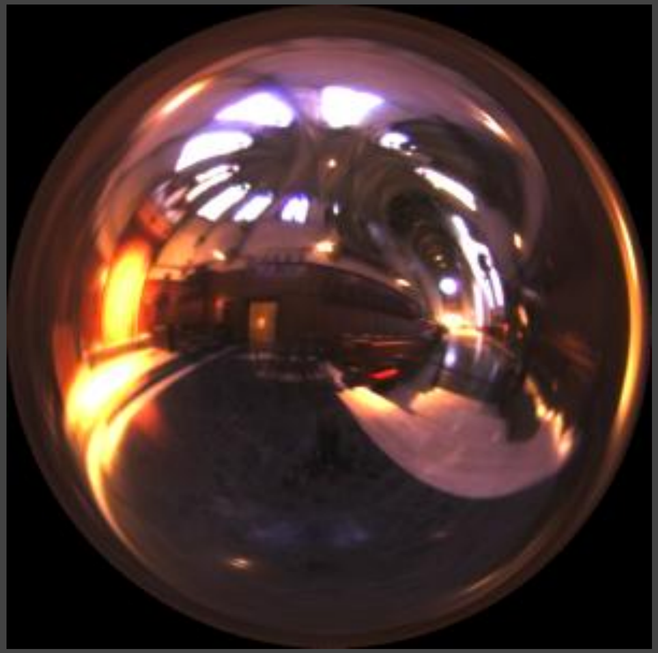
Today

- Finishing up on shadows
    - Distance field soft shadows
- Shading from environment lighting
    - The split sum approximation
- Shadow from environment lighting

Recap: Environment Lighting

- An image representing
  distant lighting from all directions

- Spherical map vs. cube map



Shading from Environment Lighting

- Informally named Image-Based Lighting (IBL)
- How to use it to shade a point (without shadows)?
  - Solving the rendering equation

$$L_o(\mathrm{p}, \omega_o) = \int_{\Omega^+} L_i(\mathrm{p}, \omega_i) f_r(\mathrm{p}, \omega_i, \omega_o) \cos\theta_i V(\mathrm{p}, \omega_i) \,\mathrm{d}\omega_i$$

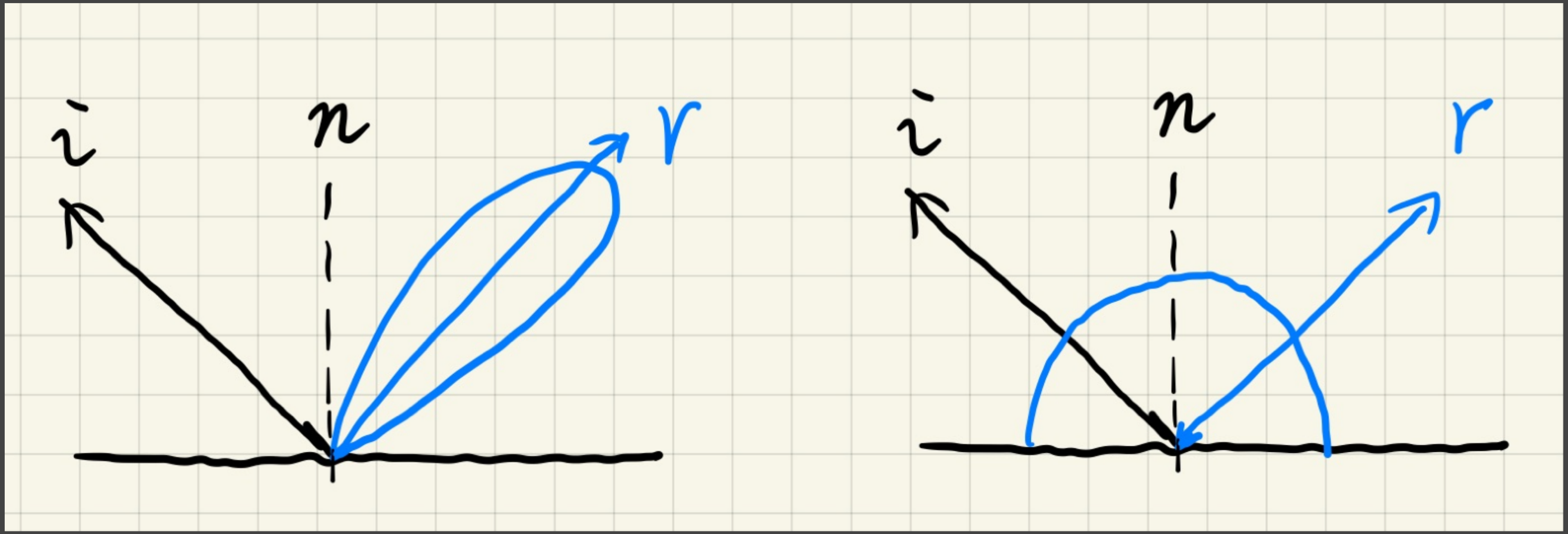
For all directions from the upper hemisphere

Shading from Environment Lighting

- General solution — Monte Carlo integration
  - Numerical
  - Large amount of samples required
- Problem — can be slow
  - In general, sampling is not preferred in shaders*
  - **Can we avoid sampling?**

Shading from Environment Lighting

- Observation
  - If the BRDF is glossy — small support!
  - If the BRDF is diffuse — smooth!
  - Does the observation remind you of something?



The Classic Approximation

- Recall: the approximation
  - Note the slight edit on Ω_G here

$$\int_{\Omega} f(x)g(x)\,\mathrm{d}x \approx \frac{\int_{\Omega_G} f(x)\,\mathrm{d}x}{\int_{\Omega_G}\,\mathrm{d}x} \cdot \int_{\Omega} g(x)\,\mathrm{d}x$$

- Conditions for acceptable accuracy?

The Split Sum: 1st Stage

- BRDF satisfies the accuracy condition in any case
  - We can safely take the lighting term out!

$$L_o(p,\omega_o) \approx \boxed{\frac{\int_{\Omega_{f_r}} L_i(p,\omega_i)\,\mathrm{d}\omega_i}{\int_{\Omega_{f_r}} \mathrm{d}\omega_i}} \cdot \int_{\Omega^+} f_r(p,\omega_i,\omega_o)\cos\theta_i\,\mathrm{d}\omega_i$$

- Note: different usage in shadows (taking vis. out)

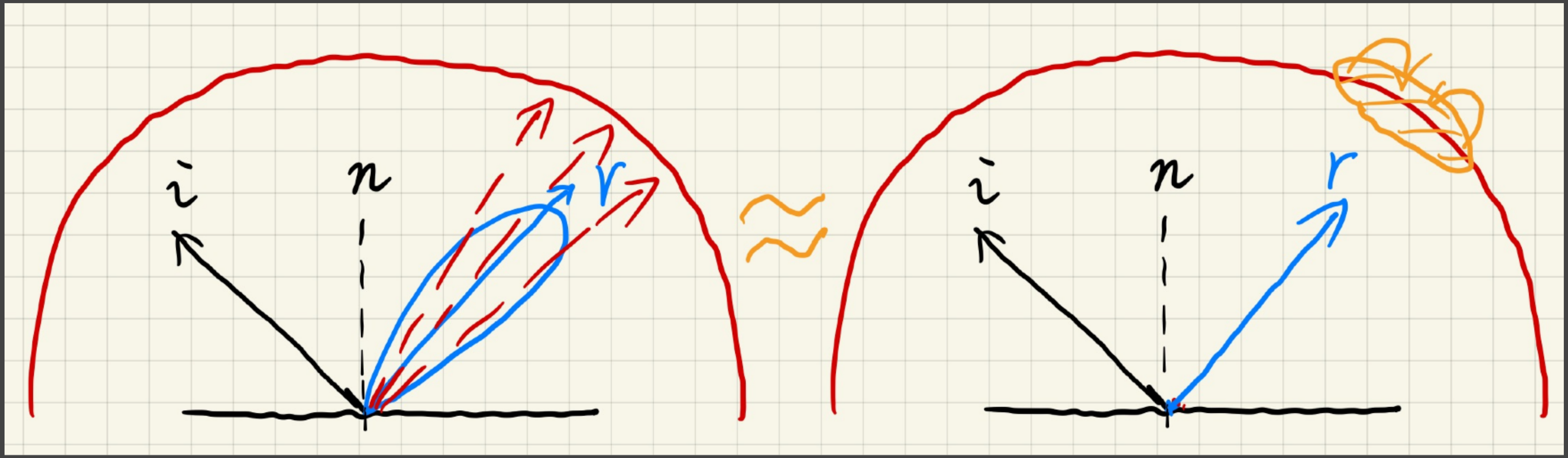
$$L_o(\mathrm{p},\omega_o) \approx \boxed{\frac{\int_{\Omega^+} V(\mathrm{p},\omega_i)\,\mathrm{d}\omega_i}{\int_{\Omega^+} \mathrm{d}\omega_i}} \cdot \int_{\Omega^+} L_i(\mathrm{p},\omega_i) f_r(\mathrm{p},\omega_i,\omega_o)\,\cos\theta_i\,\mathrm{d}\omega_i$$

The Split Sum: 1st Stage

- Prefiltering of the environment lighting
  - Pre-generating a set of differently filtered environment lighting
  - Filter size in-between can be approximated via trilinear interp.

The Split Sum: 1st Stage

- Then query the pre-filtered environment lighting at the r (mirror reflected) direction!



The Split Sum: 2nd Stage

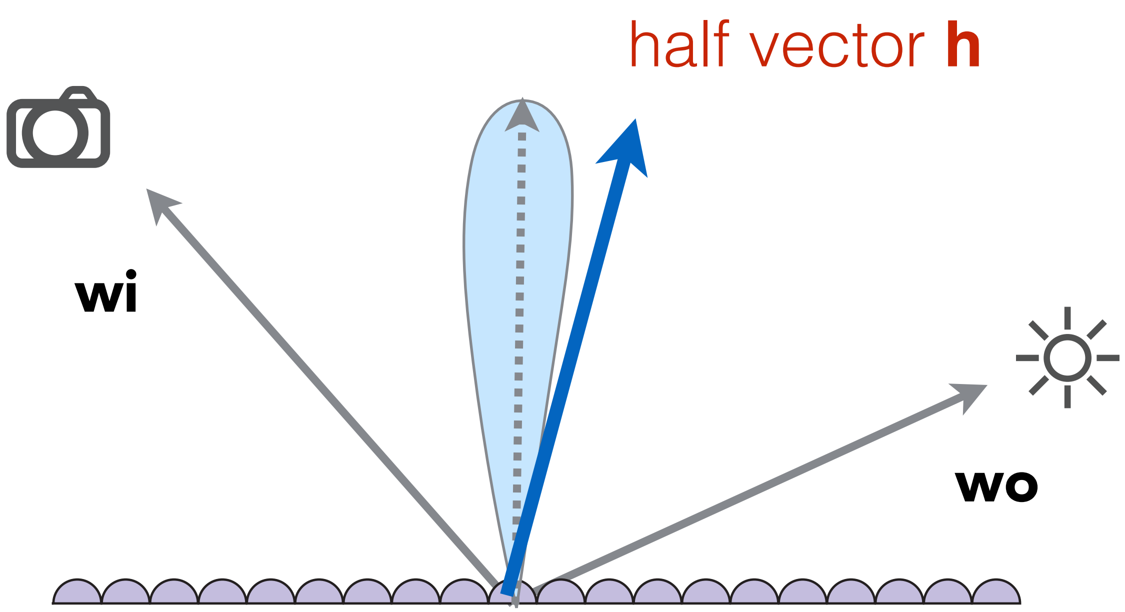
- The second term is still an integral
  - How to avoid sampling this term?

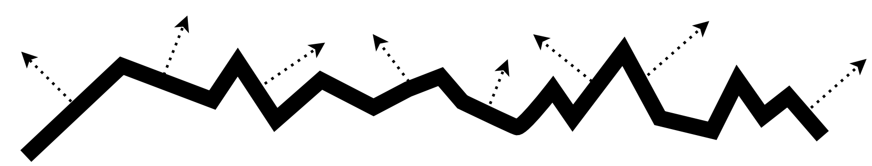
$$L_o(p,\omega_o) \approx \frac{\int_{\Omega_{f_r}} L_i(p,\omega_i)\,\mathrm{d}\omega_i}{\int_{\Omega_{f_r}} \mathrm{d}\omega_i} \cdot \boxed{\int_{\Omega^+} f_r(p,\omega_i,\omega_o)\cos\theta_i\,\mathrm{d}\omega_i}$$

- Idea
  - Precompute its value for all possible combinations of variables roughness, color (Fresnel term), etc.
  - But we'll need a huge table with extremely high dimemsions

Recall: Microfacet BRDF

- What kind of microfacets reflect wi to wo?
(hint: microfacets are mirrors)





Fresnel term — shadowing-masking term — **distribution of normals**

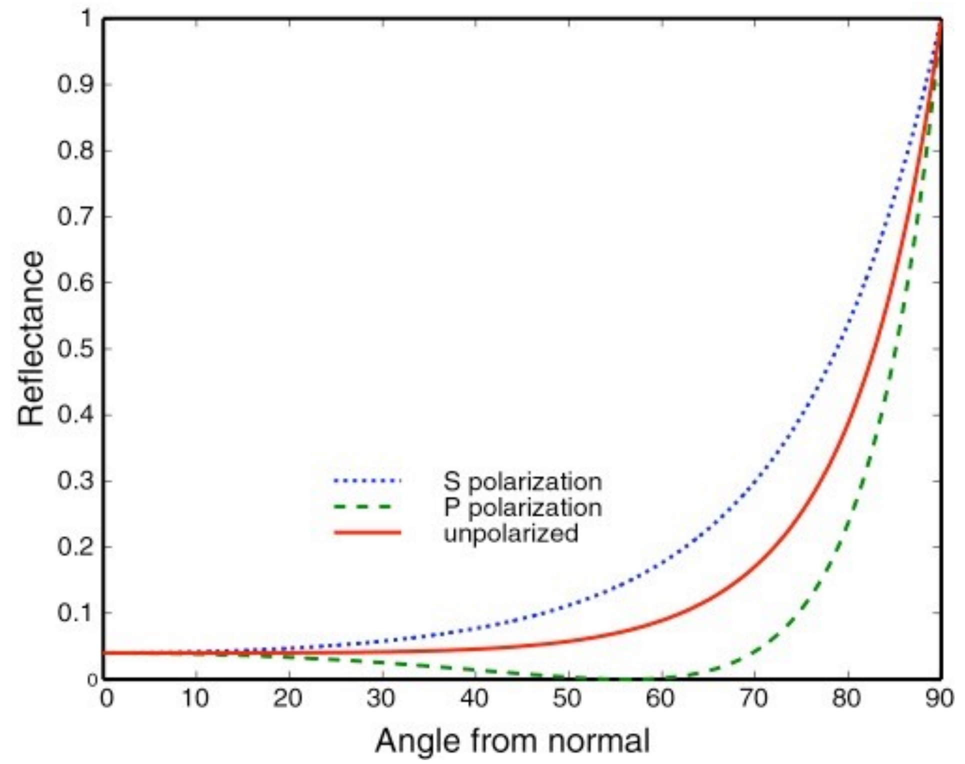
$$f(\mathbf{i},\mathbf{o}) = \frac{\mathbf{F}(\mathbf{i},\mathbf{h})\,\mathbf{G}(\mathbf{i},\mathbf{o},\mathbf{h})\,\mathbf{D}(\mathbf{h})}{4(\mathbf{n},\mathbf{i})(\mathbf{n},\mathbf{o})}$$

The Fresnel Term and the NDF

Fresnel term: the Schlick's approximation

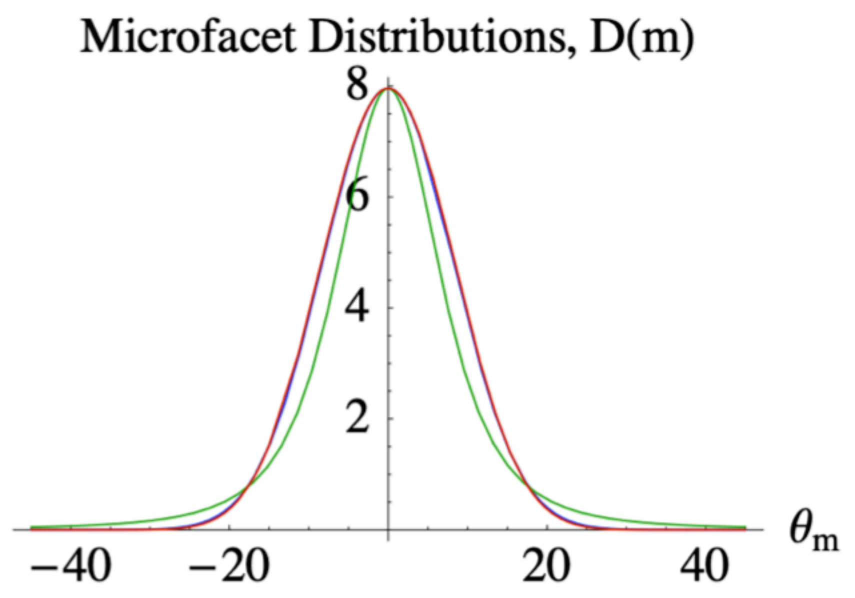
$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5$$

$$R_0 = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2$$



The NDF term: e.g. Beckmann distribution

$$D(h) = \frac{e^{-\frac{\tan^2\theta_h}{\alpha^2}}}{\pi\alpha^2\cos^4\theta_h}$$



The Split Sum: 2nd Stage

- Idea & Observation
  - Try to split the variables again!
  - The Schlick approximated Fresnel term is much simpler: Just the “base color” R_0 and the half angle θ
- Taking the Schlick’s approximation into the 2nd term
  - The “base color” is extracted!

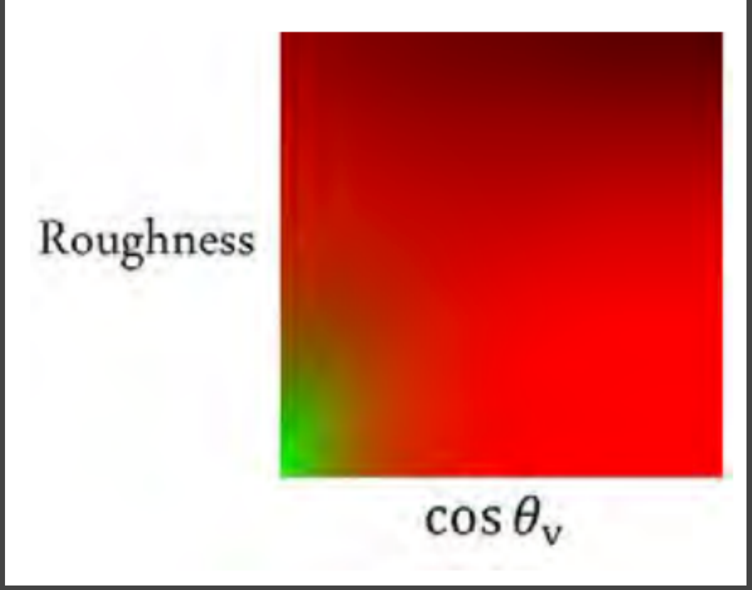
$$\int_{\Omega^+} f_r(p,\omega_i,\omega_o)\cos\theta_i\,\mathrm{d}\omega_i \approx R_0\int_{\Omega^+}\frac{f_r}{F}\left(1-(1-\cos\theta_i)^5\right)\cos\theta_i\,\mathrm{d}\omega_i + \int_{\Omega^+}\frac{f_r}{F}(1-\cos\theta_i)^5\cos\theta_i\,\mathrm{d}\omega_i$$

The Split Sum: 2nd Stage

- Both integrals can be precomputed

$$\int_{\Omega^+} f_r(p,\omega_i,\omega_o)\cos\theta_i\,\mathrm{d}\omega_i \approx R_0 \int_{\Omega^+} \frac{f_r}{F}\left(1-(1-\cos\theta_i)^5\right)\cos\theta_i\,\mathrm{d}\omega_i + \int_{\Omega^+} \frac{f_r}{F}(1-\cos\theta_i)^5\cos\theta_i\,\mathrm{d}\omega_i$$

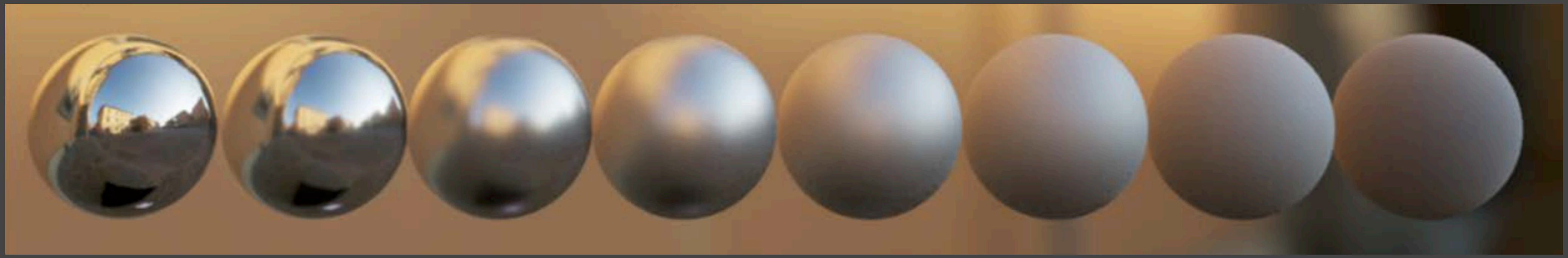
- Each integral produces one value for each (roughness, incident angle) pair
  - Therefore, each integral results in a 2D table (texture)



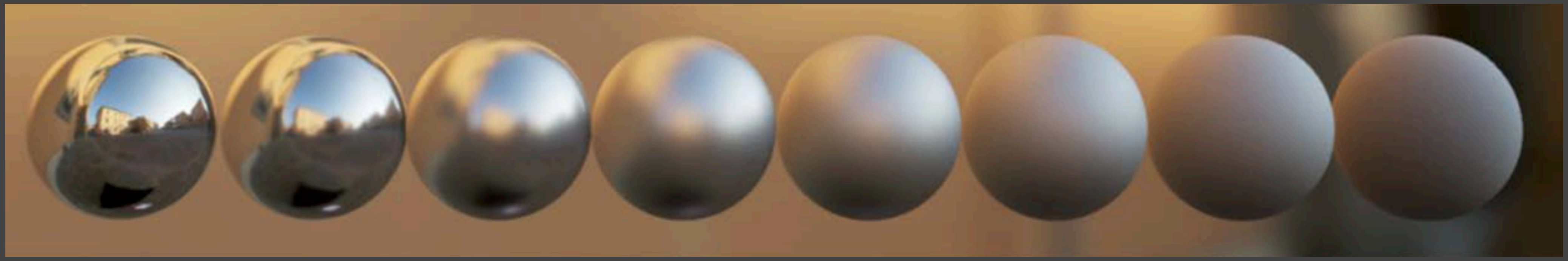
The Split Sum Approximation

- Finally, completely avoided sampling
- Very fast and almost identical results

Reference



Split sum



The Split Sum Approximation

- In the industry
  - Integral -> Sum

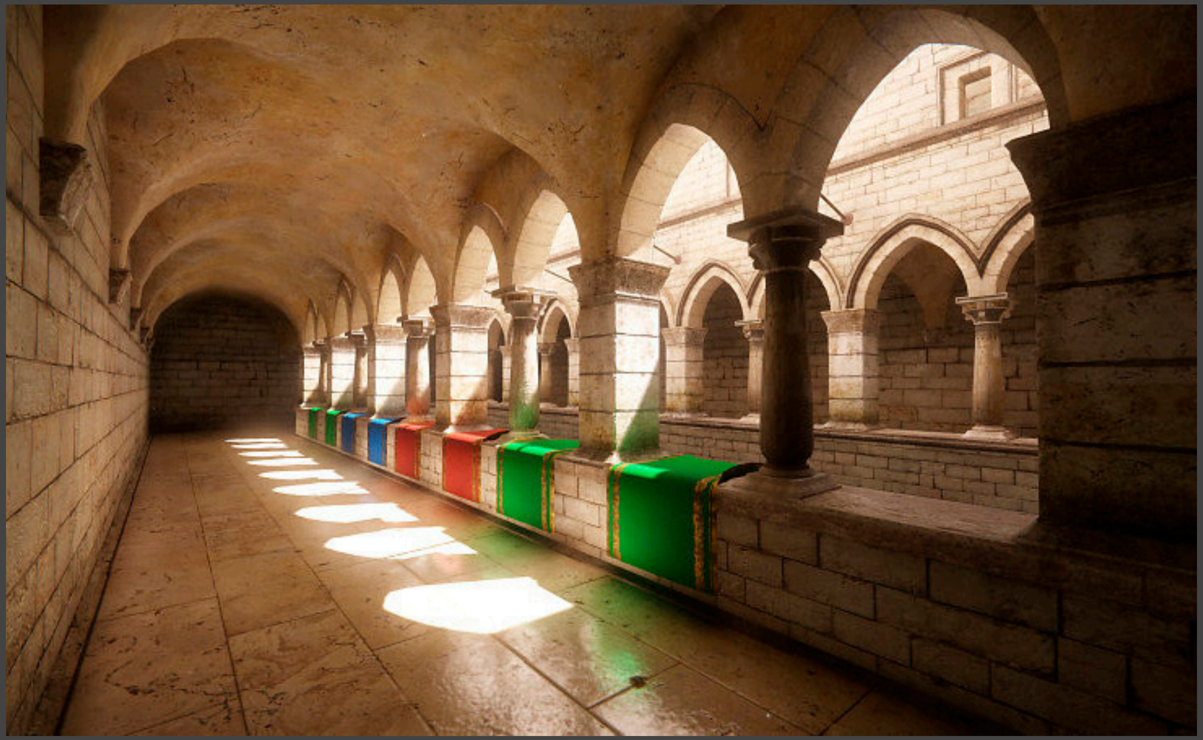
$$\frac{1}{N}\sum_{k=1}^{N}\frac{L_i(\mathbf{l}_k)f(\mathbf{l}_k,\mathbf{v})\cos\theta_{\mathbf{l}_k}}{p(\mathbf{l}_k,\mathbf{v})} \approx \left(\frac{1}{N}\sum_{k=1}^{N}L_i(\mathbf{l}_k)\right)\left(\frac{1}{N}\sum_{k=1}^{N}\frac{f(\mathbf{l}_k,\mathbf{v})\cos\theta_{\mathbf{l}_k}}{p(\mathbf{l}_k,\mathbf{v})}\right)$$

- That's why it's called split sum rather than "split integral"

Questions?

Next Lecture

- Stepping into real-time global illumination!
  - In 3D
  - In the image space
  - By precomputation
- We'll start with 3D methods
  - LPV, VXGI, RTXGI, etc.



[VXGI by NVIDIA]

Thank you!