Samurai Shading

in

Jasmin Patry
Sucker Punch Productions

Introduction

GHOST

- Sucker Punch is a part of Sony Interactive Entertainment Worldwide Studios (soon PlayStation Studios)
- Peak size: 160 people (including 25 QA)
- Previous games include:
  - *Sly Cooper* (*1*, *2*, & *3*)
  - *Infamous* (*1*, *2*, *Festival of Blood*, *Second Son*, *First Light*)

Introduction

- Recently released *Ghost of Tsushima* for the PlayStation 4

- Our goals after *Infamous: Second Son* and *First Light* included improving the quality of our materials and lighting
  - Particularly for characters
- **Goal:** Support new types of materials, including:
  - Materials with strong anisotropic specular response
  - Materials with strong asperity scattering component
- **Goal:** Improve fidelity of skin shading
- **Goal:** Make it easy to use detail maps based on scanned materials
  - For all types of maps (albedo, gloss, specular, normal, etc.)

- Art direction called for “stylized realism”
- Lighting models are physically based
- Materials authored with physically plausible values, some photogrammetry
- Lighting can deviate from physical correctness
  - Artists can globally adjust to achieve desired look

- Most lighting in *Ghost* is deferred:
  - Lambertian diffuse
    - Optional translucency
    - Optional simplified asperity scattering (fuzziness)
  - Isotropic GGX specular
    - Optional colored specular via “metallic” channel
- Other materials are forward lit, e.g. when using:
  - Anisotropic GGX specular
  - Full anisotropic asperity scattering BRDF
  - Subsurface scattering

Introduction | Overview

- This talk is divided into the following four topics:
  - Anisotropic specular maps and filtering
  - Anisotropic asperity scattering BRDF
  - Skin shading improvements
  - Physically based detail maps (as bonus slides)





Anisotropic Specular

Anisotropic Specular

- In previous games, we supported GGX anisotropic specular aligned with mesh tangent space
- Wanted to support arbitrary spatially varying orientation
  - Plus normal variance filtering
- The SGGX paper by Heitz et al. (2015) gave us ideas on ways to accomplish both goals

- LEAN Mapping (Olano and Baker, 2010)
  - Uses a Beckmann-like NDF
  - Requires choice of scale factor for texture storage
    - 16-bit textures preferable
  - For flat normals, gives equivalent results to our algorithm
  - Can be adapted to work with GGX-based BRDFs using:

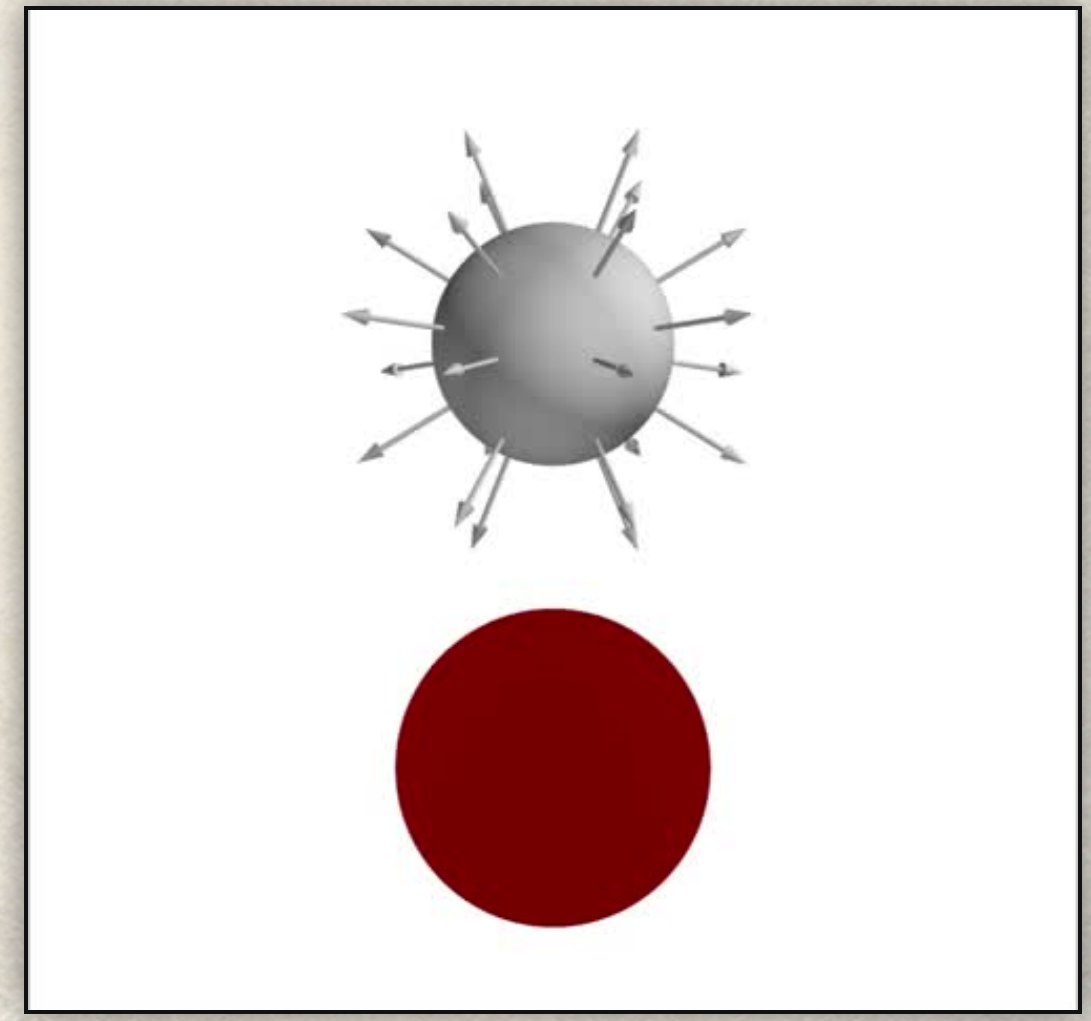
$$\sigma = \frac{1}{\sqrt{2}}\alpha$$

    - σ: Beckmann roughness
    - This is the approach that we used in the following comparisons

- LEADR Mapping (Dupuy, Heitz, et al., 2013)
  - Incorporates displacement mapping
  - Accounts for entire BRDF:
    - NDF
    - Masking-shadowing
    - Diffuse microfacets
  - Uses Beckmann NDF, and same storage and NDF filtering as LEAN
  - Similar to our approach, except that we're:
    - Using the GGX NDF and associated shadow-masking term
    - Using SGGX-based filtering instead of Beckmann-based
    - Not including diffuse microfacets or displacement mapping

Anisotropic Specular | SGGX Recap

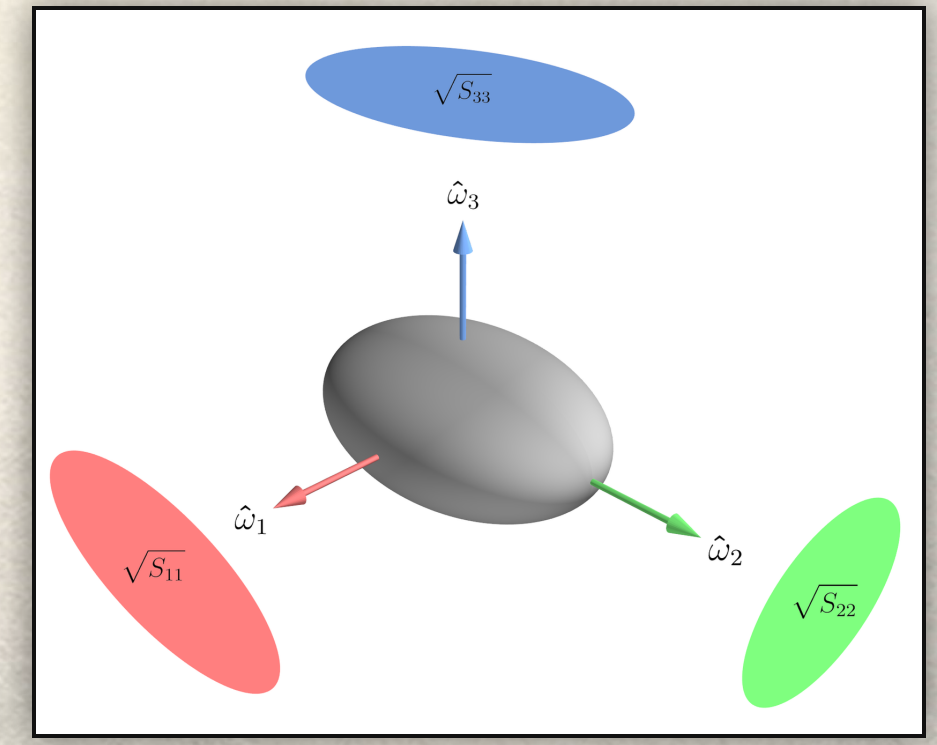
- SGGX developed as a *microflake distribution*
  - Extends GGX NDF to spherical domain
  - NDF of ellipsoid
  - GGX is a special case of SGGX



- SGGX characterized by an $\mathbf{S}$ matrix

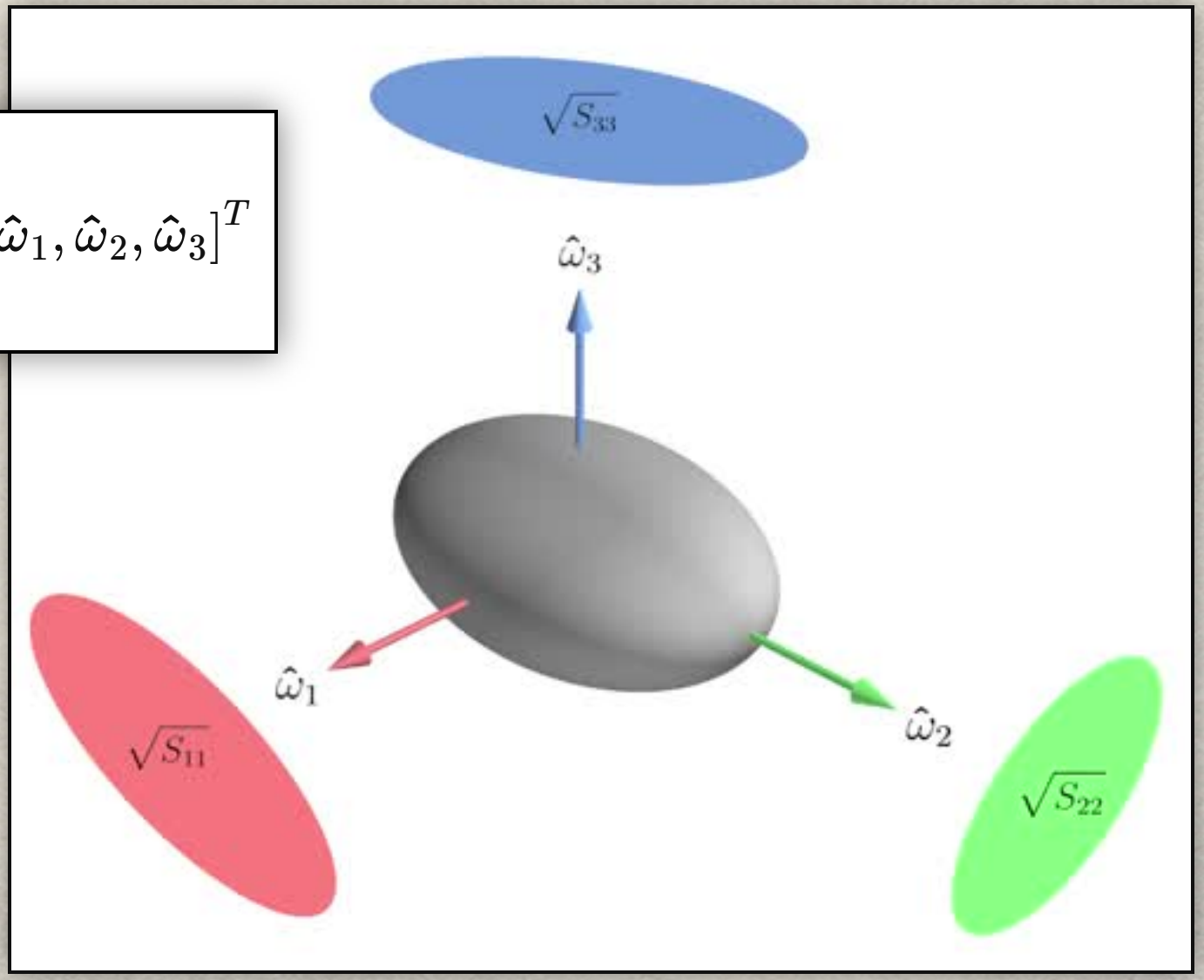
$$\mathbf{S} = [\hat{\omega}_1, \hat{\omega}_2, \hat{\omega}_3] \begin{bmatrix} S_{11} & 0 & 0 \\ 0 & S_{22} & 0 \\ 0 & 0 & S_{33} \end{bmatrix} [\hat{\omega}_1, \hat{\omega}_2, \hat{\omega}_3]^T$$

- $\hat{\omega}_i$: principal axes of ellipsoid
  - Also eigenvectors of $\mathbf{S}$
- S_{ii}: Square of ellipsoid projected area
  - Eigenvalues of $\mathbf{S}$



Anisotropic Specular | SGGX Recap

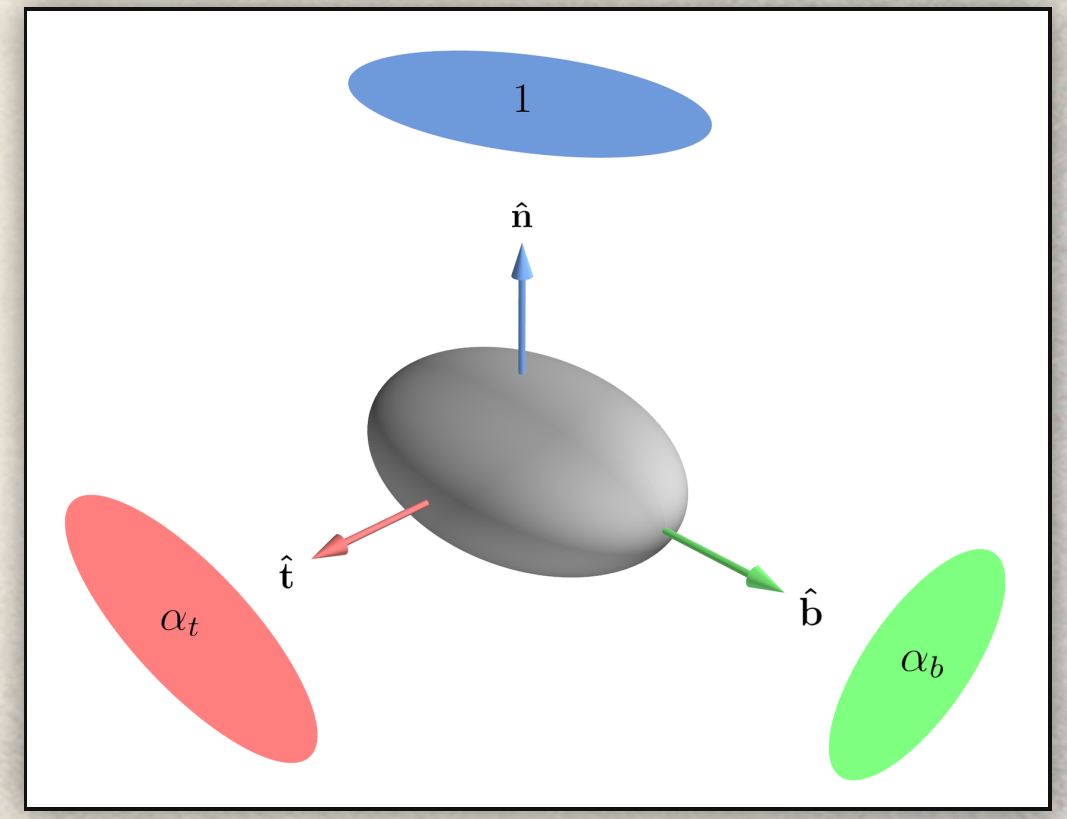
$$\mathbf{S} = [\hat{\omega}_1, \hat{\omega}_2, \hat{\omega}_3] \begin{bmatrix} S_{11} & 0 & 0 \\ 0 & S_{22} & 0 \\ 0 & 0 & S_{33} \end{bmatrix} [\hat{\omega}_1, \hat{\omega}_2, \hat{\omega}_3]^T$$



- GGX is a special case:

$$\mathbf{S} = \left[\hat{\mathbf{t}}, \hat{\mathbf{b}}, \hat{\mathbf{n}}\right] \begin{bmatrix} \alpha_t^2 & 0 & 0 \\ 0 & \alpha_b^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\hat{\mathbf{t}}, \hat{\mathbf{b}}, \hat{\mathbf{n}}\right]^T$$

- $\hat{\mathbf{t}}$: Tangent
- $\hat{\mathbf{b}}$: Bitangent
- $\hat{\mathbf{n}}$: Normal
- α: GGX roughness

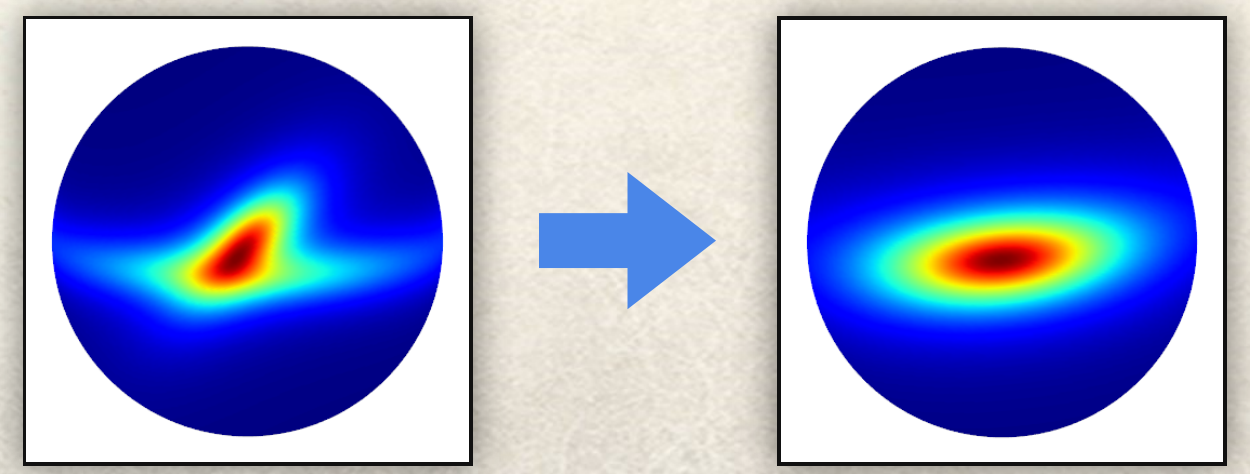


- Using the normal-mapped basis defined by $\hat{\mathbf{t}}_n$, $\hat{\mathbf{b}}_n$, and $\hat{\mathbf{n}}$, GGX can also be written as:

$$\mathbf{S} = \left[\hat{\mathbf{t}}_n, \hat{\mathbf{b}}_n, \hat{\mathbf{n}}\right] \begin{bmatrix} S_{xx} & S_{xy} & 0 \\ S_{xy} & S_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\hat{\mathbf{t}}_n, \hat{\mathbf{b}}_n, \hat{\mathbf{n}}\right]^T$$

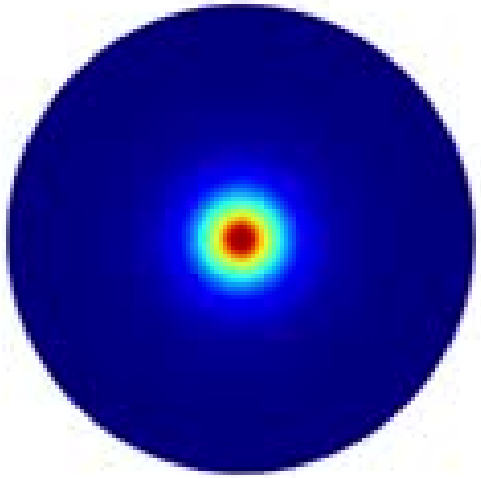
- $\hat{\mathbf{t}}_n$: Normal-mapped tangent
- $\hat{\mathbf{b}}_n$: Normal-mapped bitangent
- $\hat{\mathbf{n}}$: Normal-mapped normal
- S_{xx}, S_{xy}, S_{yy}: 2x2 GGX submatrix

- The SGGX paper describes a “parameter estimation” algorithm for determining the SGGX NDF from an arbitrary NDF
  - SGGX NDF ellipsoid has same axes as covariance eigenvectors
  - Preserves microflake projected area along these axes
- Heitz et al. also show that linearly filtering $\mathbf{S}$ is a good approximation to this algorithm

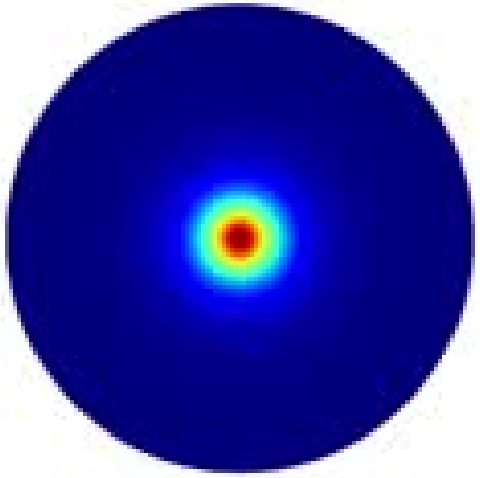


Input NDF

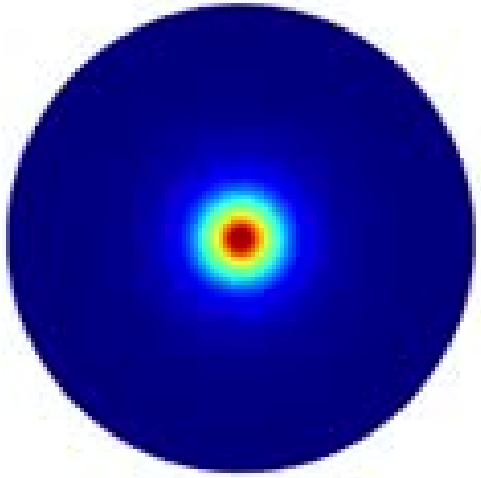
Linear Filtering



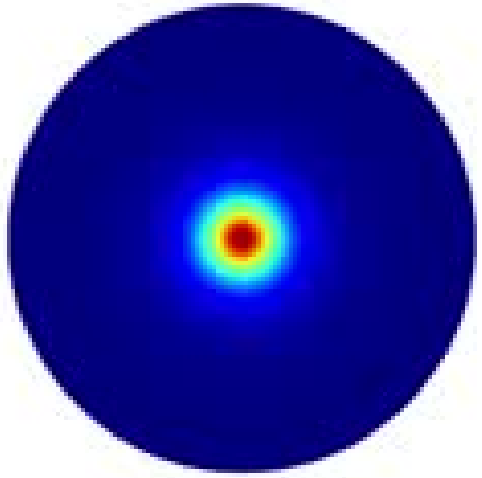
Input NDF



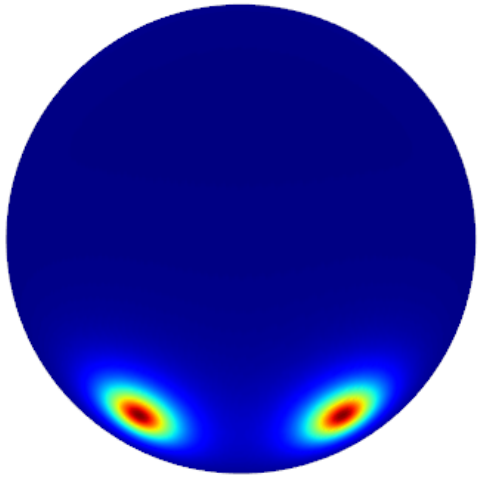
Linear Filter



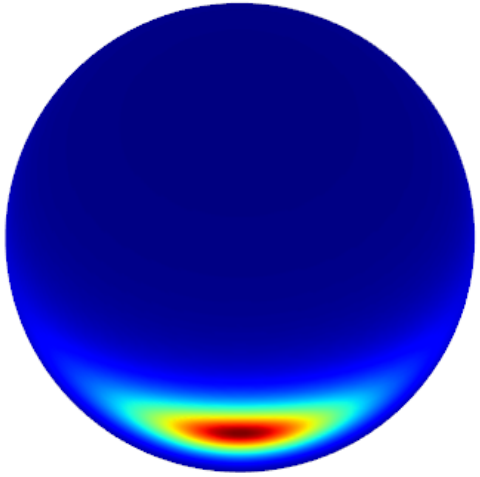
Parameter Est.



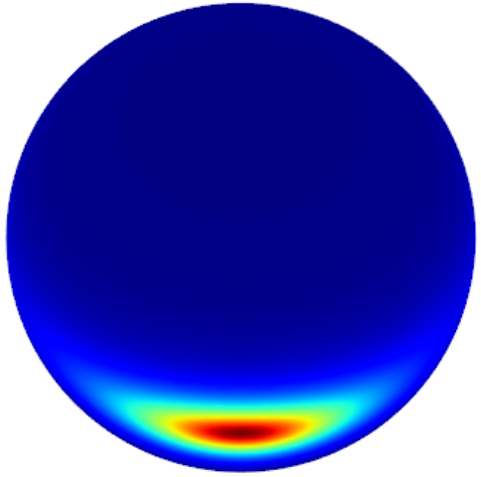
LEAN



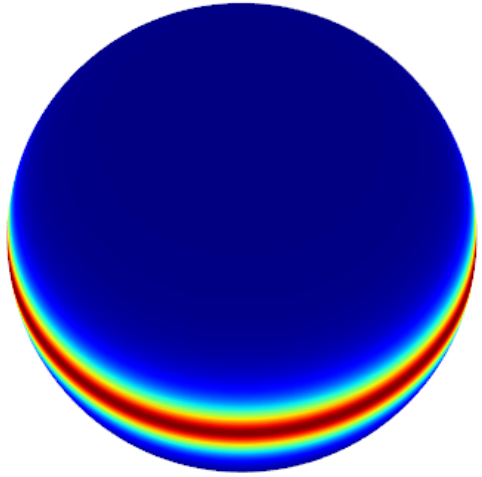
Input NDF



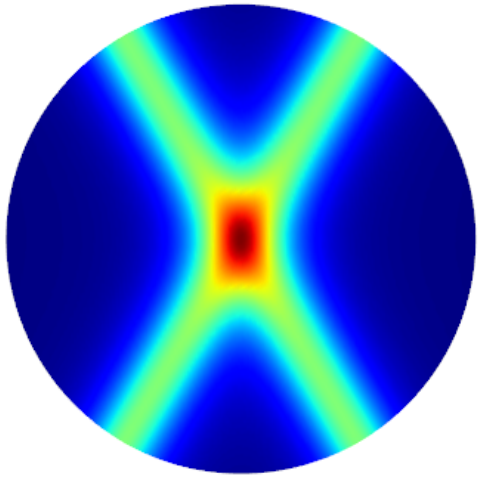
Linear Filter



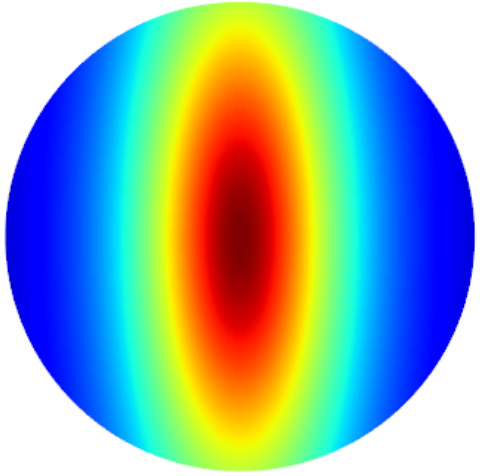
Parameter Est.



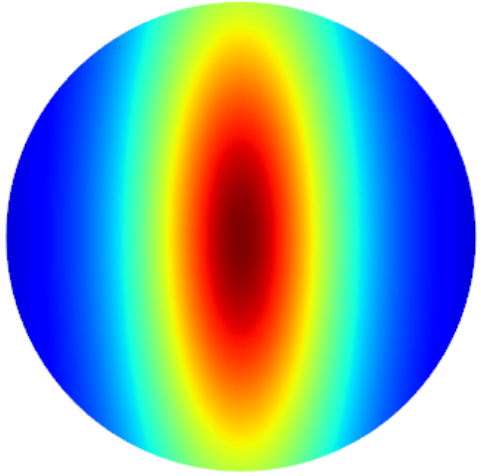
LEAN



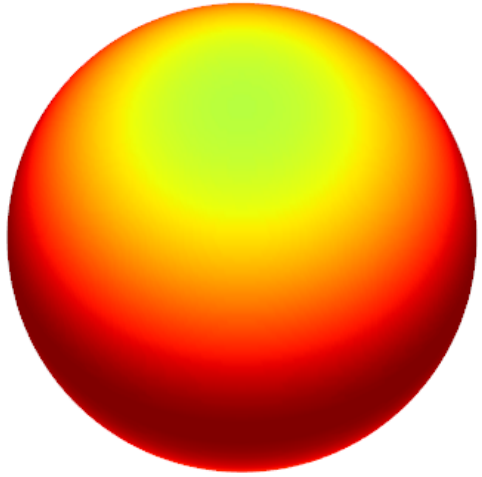
Input NDF



Linear Filter



Parameter Est.



LEAN

1. Convert normal + 2x2 anisotropic GGX matrix to 3x3 SGGX matrix $\mathbf{S}$:

$$\mathbf{S} = \mathbf{M}_{\hat{\mathbf{n}}} \begin{bmatrix} S_{xx} & S_{xy} & 0 \\ S_{xy} & S_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{M}_{\hat{\mathbf{n}}}^{T}$$

where $\mathbf{M}_{\hat{\mathbf{n}}}$ rotates $\hat{\mathbf{k}} = (0, 0, 1)$ onto $\hat{\mathbf{n}} = (x, y, z)$:

$$\mathbf{M}_{\hat{\mathbf{n}}} = \begin{bmatrix} z + \frac{y^2}{1+z} & \frac{-xy}{1+z} & x \\ \frac{-xy}{1+z} & z + \frac{x^2}{1+z} & y \\ -x & -y & z \end{bmatrix}$$

2. Linearly filter $\mathbf{S}$ matrices during mipmap generation
3. Extract new normal $\hat{\mathbf{n}}'$ and symmetric 2x2 submatrix from new $\mathbf{S}'$ matrix:

$$\begin{bmatrix} S'_{xx} & S'_{xy} & 0 \\ S'_{xy} & S'_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \frac{\mathbf{M}_{\hat{\mathbf{n}}'}^T \mathbf{S}' \mathbf{M}_{\hat{\mathbf{n}}'}}{\left(\mathbf{M}_{\hat{\mathbf{n}}'}^T \mathbf{S}' \mathbf{M}_{\hat{\mathbf{n}}'}\right)_{3,3}}$$

To extract $\hat{\mathbf{n}}'$ from $\mathbf{S}'$, use power iteration:

$$\hat{\mathbf{n}}'_0 = [0, 0, 1]$$

$$\hat{\mathbf{n}}'_i = \frac{\mathbf{S}' \hat{\mathbf{n}}'_{i-1}}{\left| \mathbf{S}' \hat{\mathbf{n}}'_{i-1} \right|}$$

4. Store to BC5 normal map + BC7 anisotropic specular (“aniso”) map with components

$$\left[\sqrt{S'_{xx}}, \frac{1}{2}\frac{S'_{xy}}{\sqrt{S'_{xx}S'_{yy}}} + \frac{1}{2}, \sqrt{S'_{yy}}\right]$$

1. Decode at run time
   - One caveat is that linear interpolation of our texture may result in a non-positive definite matrix
     - I.e., negative eigenvalues
   - Need to guard against this before using
   - We chose to extract the eigenvalues and eigenvectors
     - Need the new anisotropic tangent and bitangent for our ambient specular approximation
   - Extract α_t^2, α_b^2, $\alpha_t\alpha_b$, $\hat{\mathbf{t}}$, $\hat{\mathbf{b}}$ for use in:

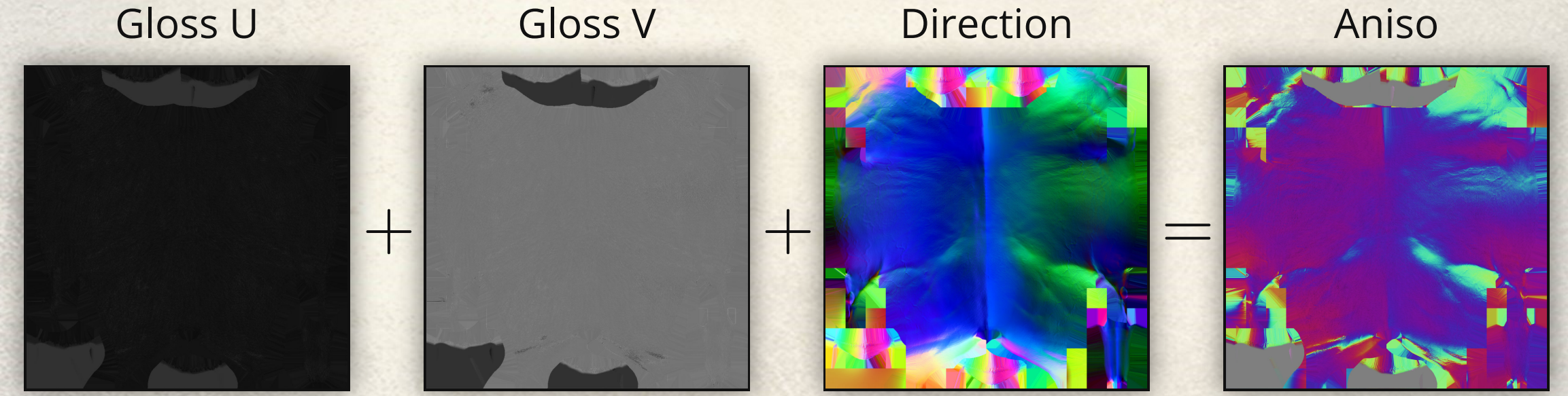
$$D(\hat{\mathbf{h}}) = \frac{(\alpha_t\alpha_b)^3}{\pi\left(\left(\hat{\mathbf{t}}\cdot\hat{\mathbf{h}}\right)^2\alpha_b^2+\left(\hat{\mathbf{b}}\cdot\hat{\mathbf{h}}\right)^2\alpha_t^2+\left(\hat{\mathbf{n}}\cdot\hat{\mathbf{h}}\right)^2(\alpha_t\alpha_b)^2\right)^2}$$

Anisotropic Specular | Our Algorithm

```
1  void DecodeSggx(float3 anisoTex, float3 tangent, float3 bitangent, out SAnisoSpecData anid)
2  {
3      const float s_alphaMin = GgxAlphaFromGloss(1.0f);
4
5      float sxx = Square(anisoTex.x);
6      float sxy = anisoTex.x * (anisoTex.y * 2.0 - 1.0) * anisoTex.z;
7      float syy = Square(anisoTex.z);
8
9      float discr = sqrt(4.0f * Square(sxy) + Square(sxx - syy));
10     float discrRcp = rcp(max(discr, 1e-6f));
11
12     anid.m_alphaTSqr = max(saturate(0.5f * (sxx + syy + discr)), Square(s_alphaMin));
13     anid.m_alphaBSqr = max(saturate(0.5f * (sxx + syy - discr)), Square(s_alphaMin));
14     anid.m_alphaTB = sqrt(anid.m_alphaTSqr * anid.m_alphaBSqr);
15
16     float cSqr = saturate(0.5f * ((sxx - syy) * discrRcp + 1.0f));
17     float c = sqrt(cSqr);
18     float s = Sign(sxy) * sqrt(1.0f - cSqr);
19
20     anid.m_tangent = c * tangent + s * bitangent;
21     anid.m_bitangent = -s * tangent + c * bitangent;
22 }
```

Anisotropic Specular | Authoring

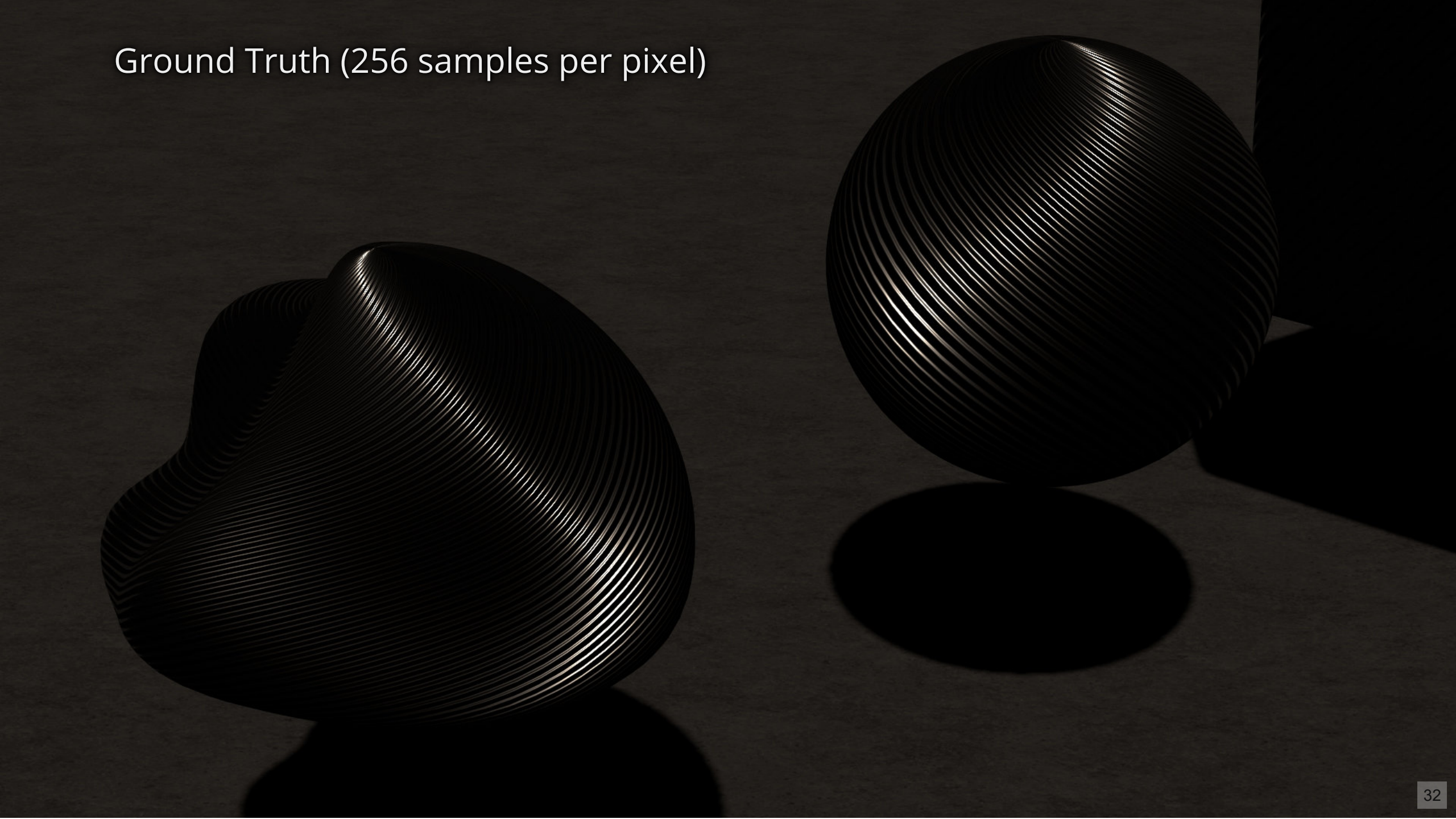
- Aniso maps are authored by artists using a custom Substance Designer node

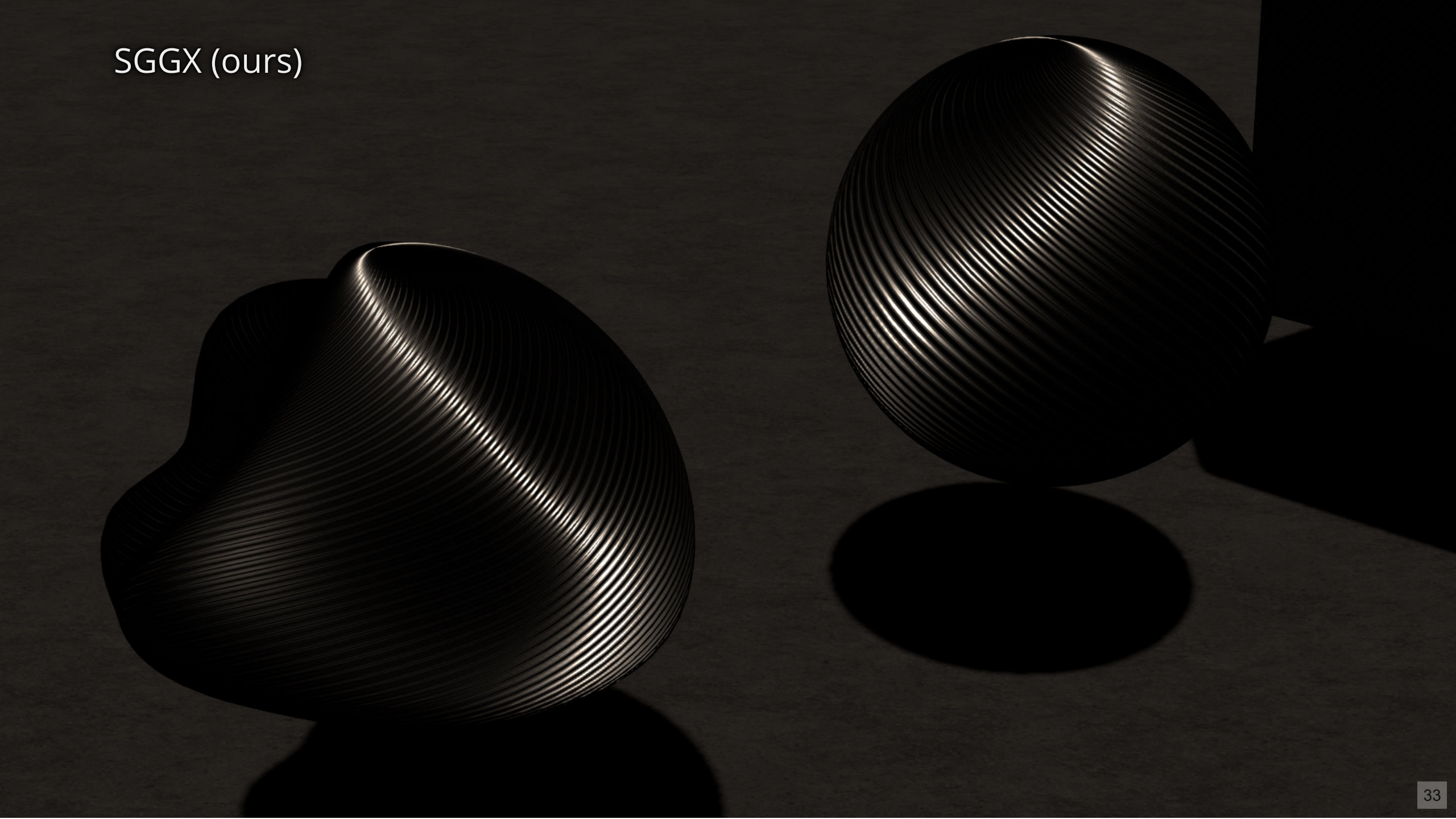


Anisotropic Specular | Results

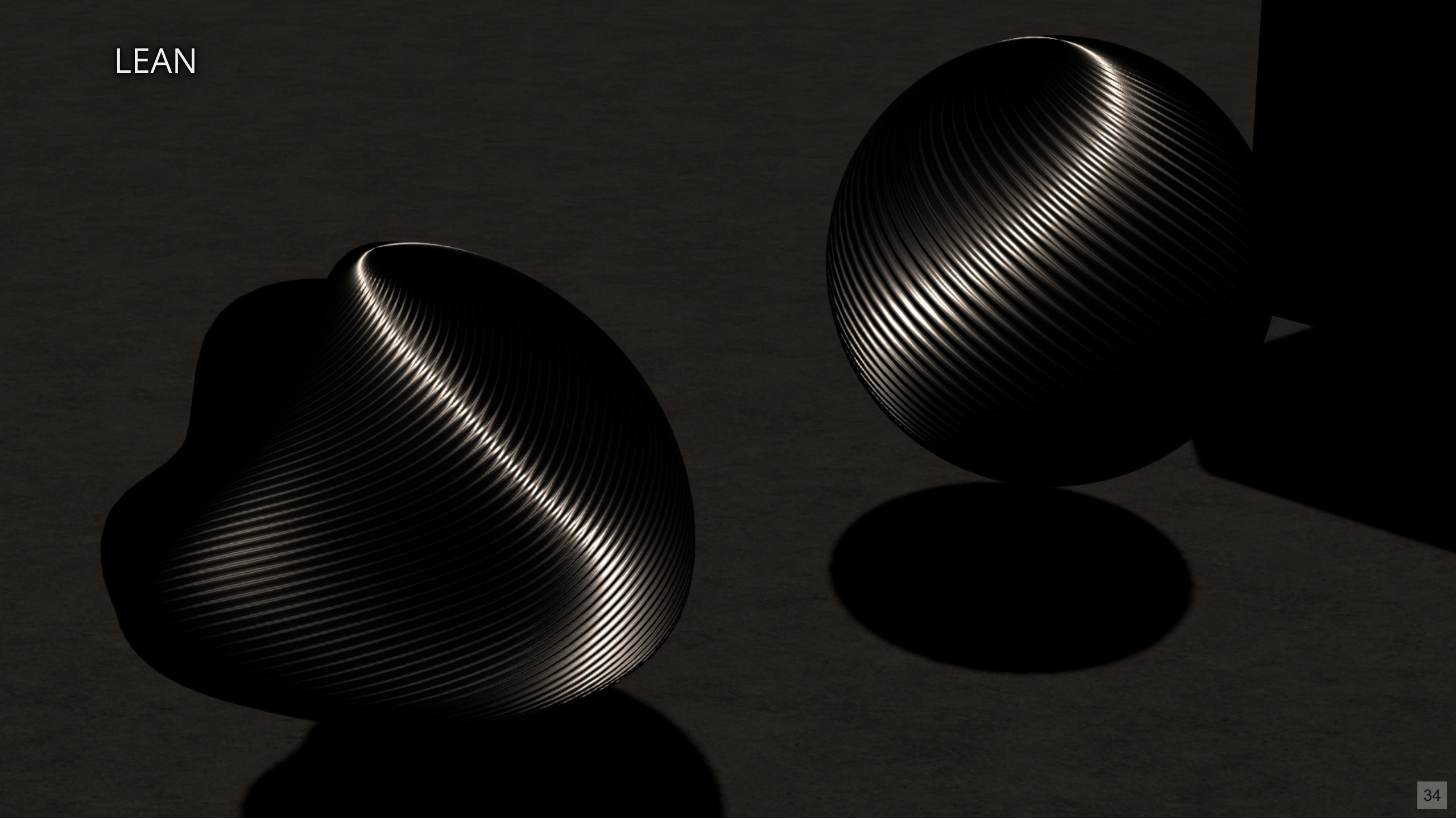
- Comparison of SGGX and LEAN filtering

Ground Truth (256 samples per pixel)

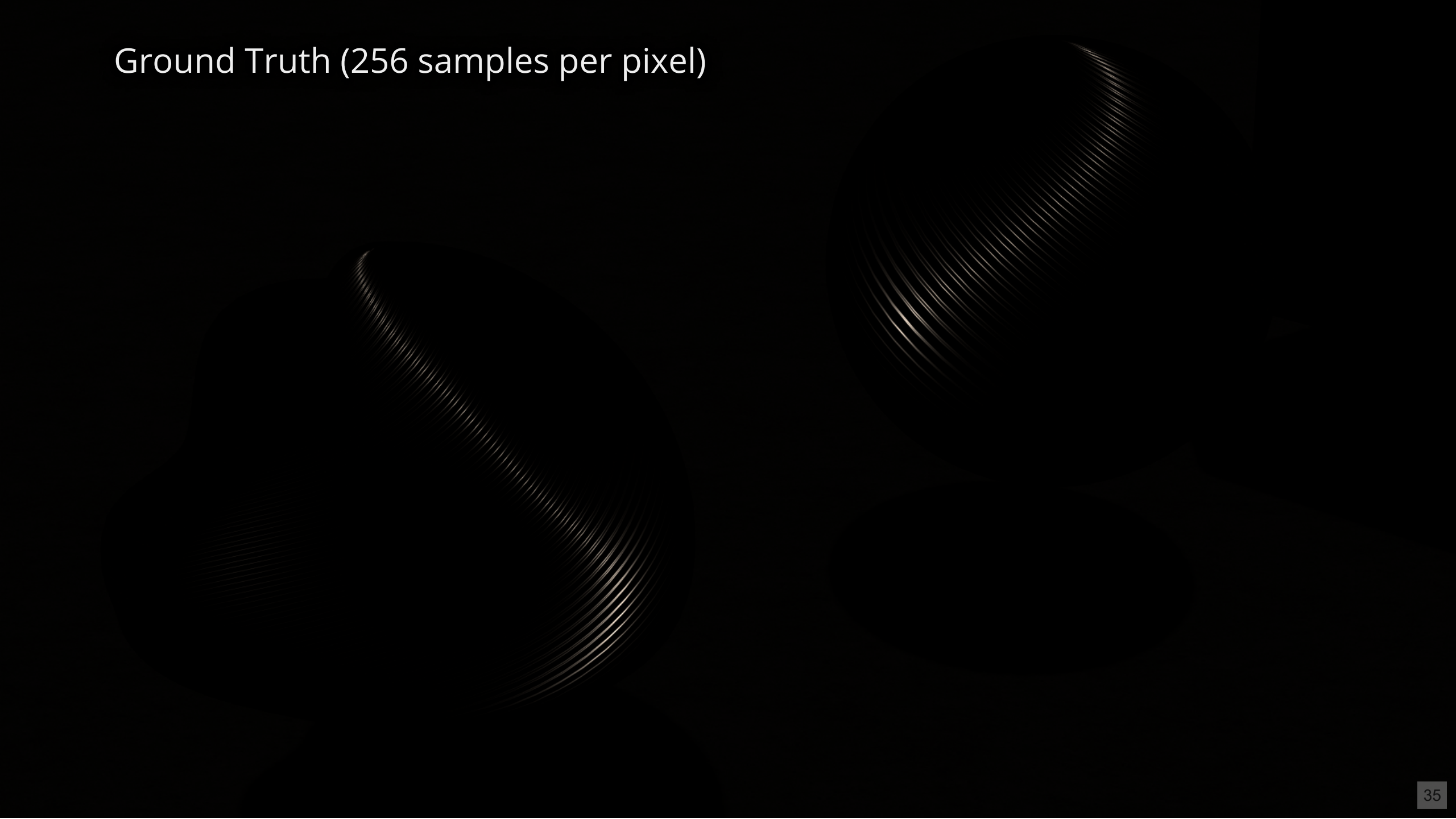



SGGX (ours)

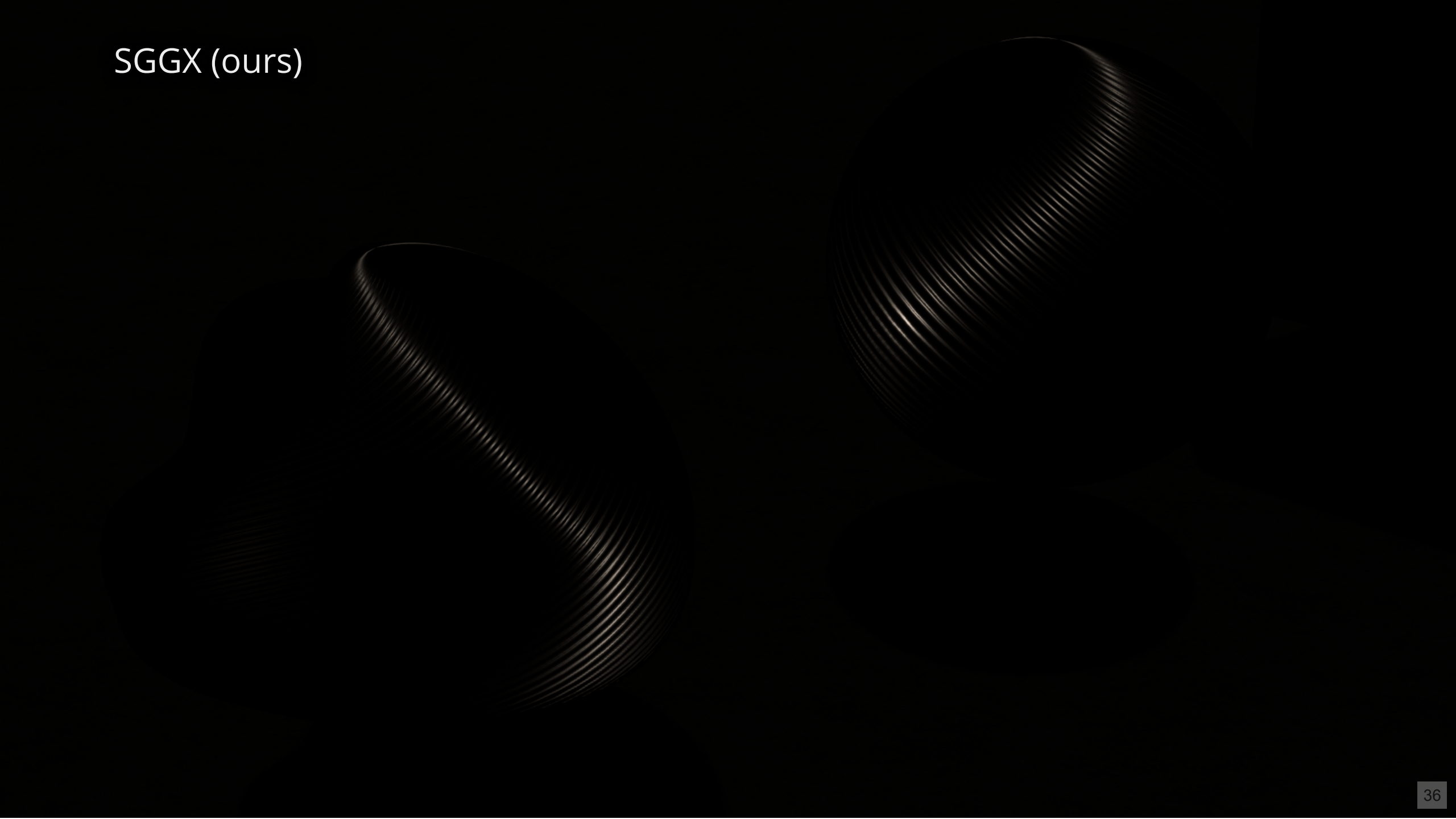
LEAN



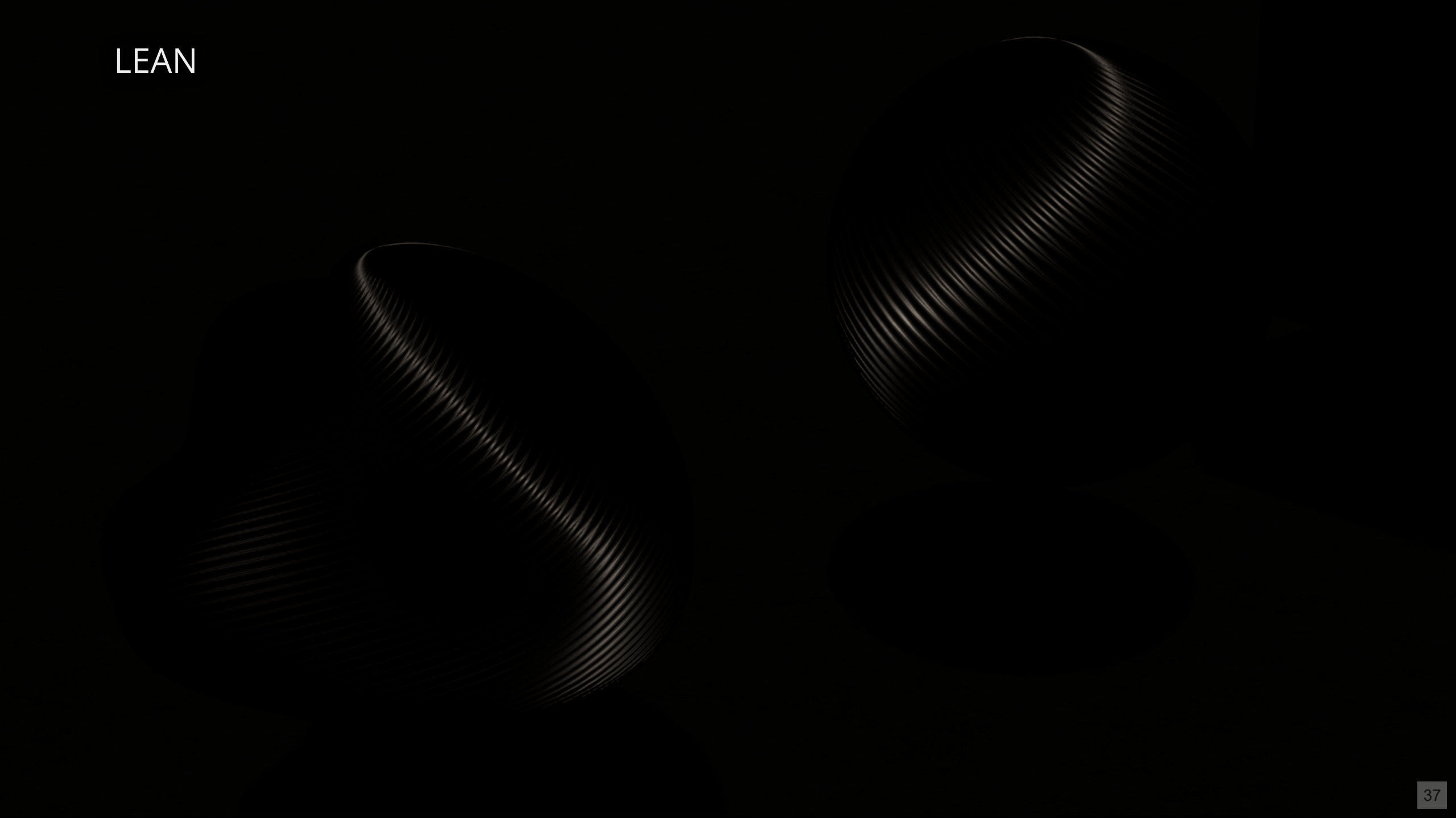
Ground Truth (256 samples per pixel)

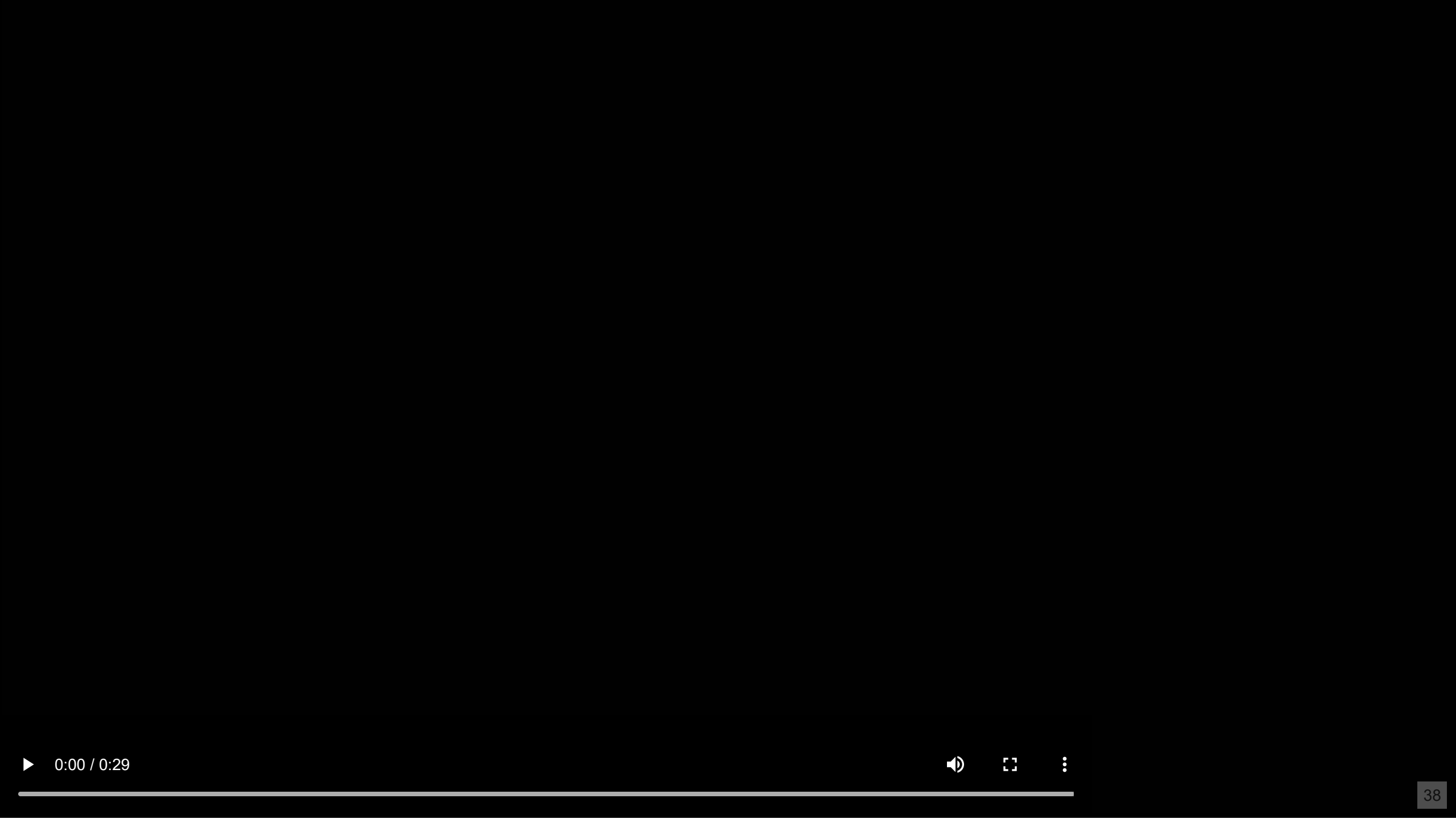


SGGX (ours)



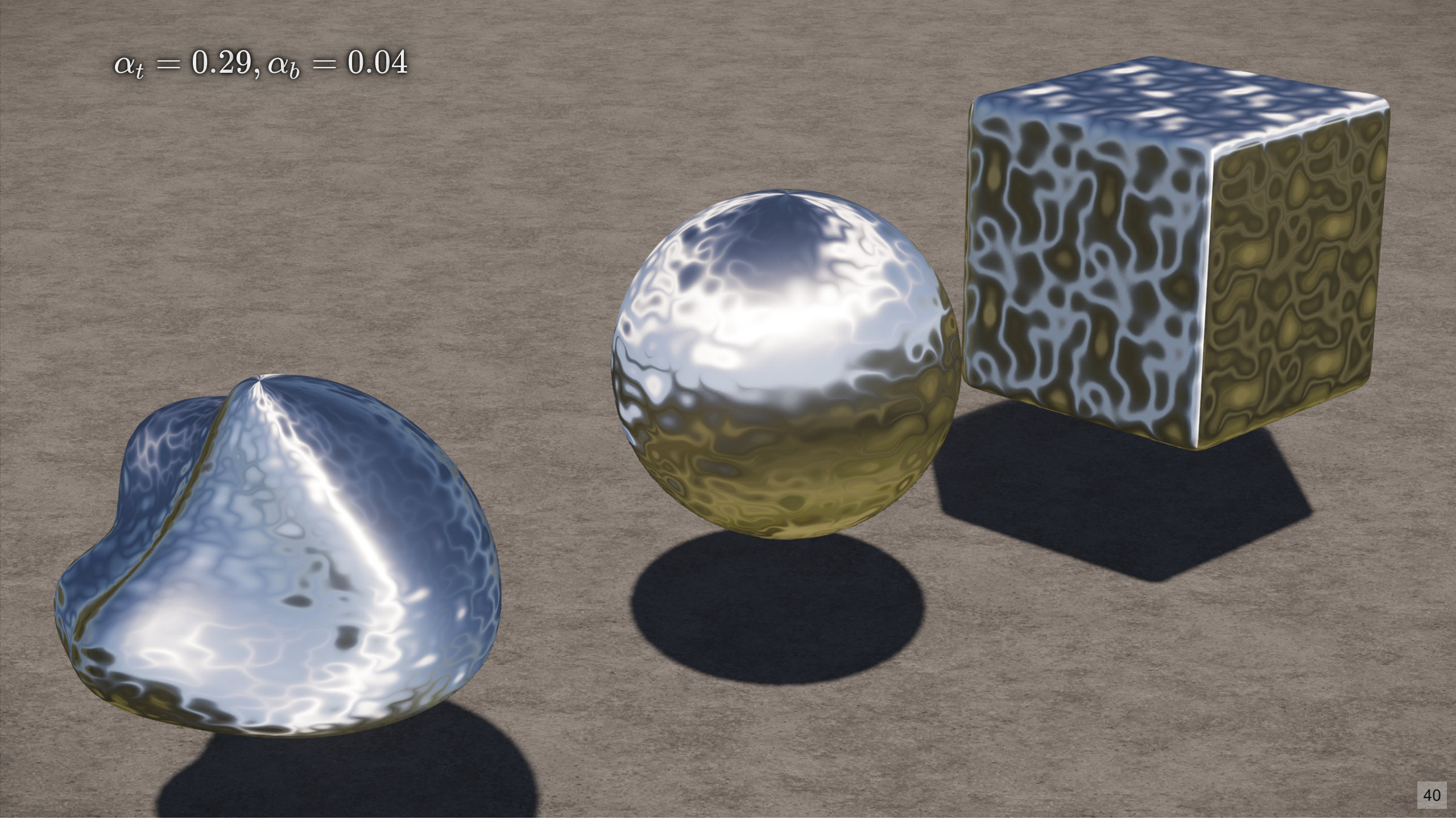
LEAN



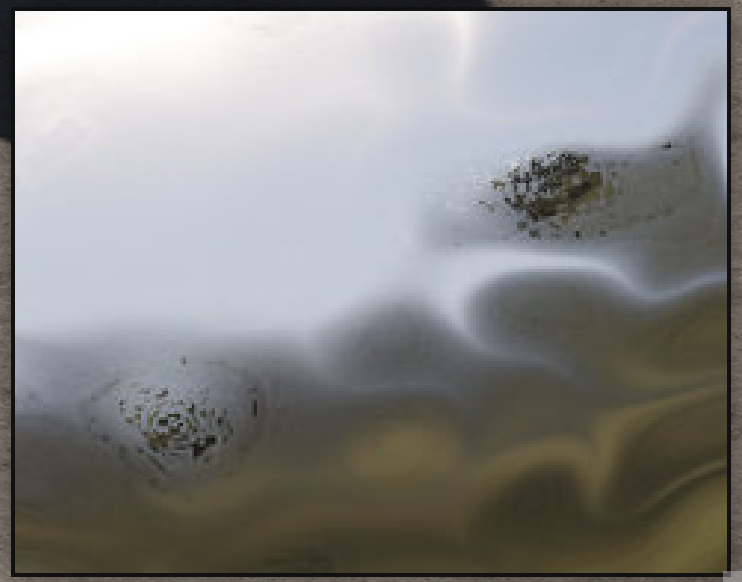

0:00 / 0:29

Anisotropic Specular | Limitations

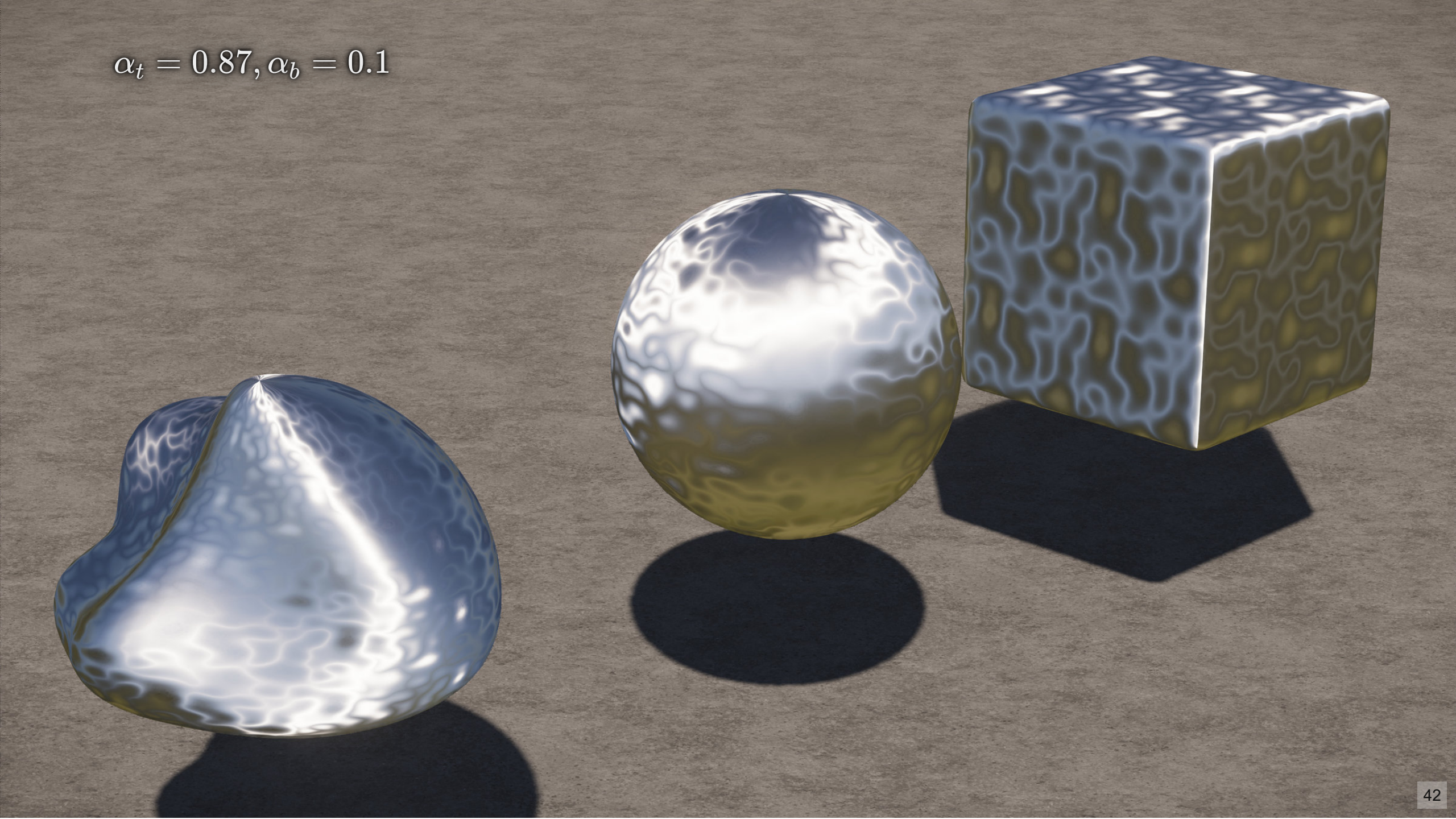
- Due to encoding, texture hardware does not interpolate $\mathbf{S}$ matrix directly, which may result in artifacts
  - In practice, for real-world materials, this did not result in problems
- Combinations of large and small α_t and α_b values do not work well


$\alpha_t = 0.29, \alpha_b = 0.04$

$\alpha_t = 0.87, \alpha_b = 0.04$



$\alpha_t = 0.87, \alpha_b = 0.1$



Anisotropic Specular | Summary

- New SGGX-based filtering approach
  - Simple to implement
  - Same number of parameters (5) as LEAN
  - Same/similar shading cost
  - Good results for strong normals and highly anisotropic NDFs
- New anisotropic roughness texture encoding
  - Based on SGGX encoding
  - Works well with BC7 compression
    - 1/4 to 1/8th the size of LEAN encoding


Anisotropic Asperity Scattering

Anisotropic Asperity Scattering BRDF

- Asperity scattering
  - … AKA sheen
  - … AKA fuzziness
  - … AKA fuzz

Anisotropic

- Asperity scattering
  - … AKA sheen
  - … AKA fuzziness
  - … AKA fuzz

Anisotropic Fuzz

- Asperity scattering
  - … AKA sheen
  - … AKA fuzziness
  - … AKA fuzz

Anisotropic Fuzz

- Asperity scattering
  - … AKA sheen
  - … AKA fuzziness
  - … AKA fuzz
- New BRDF was researched during look development

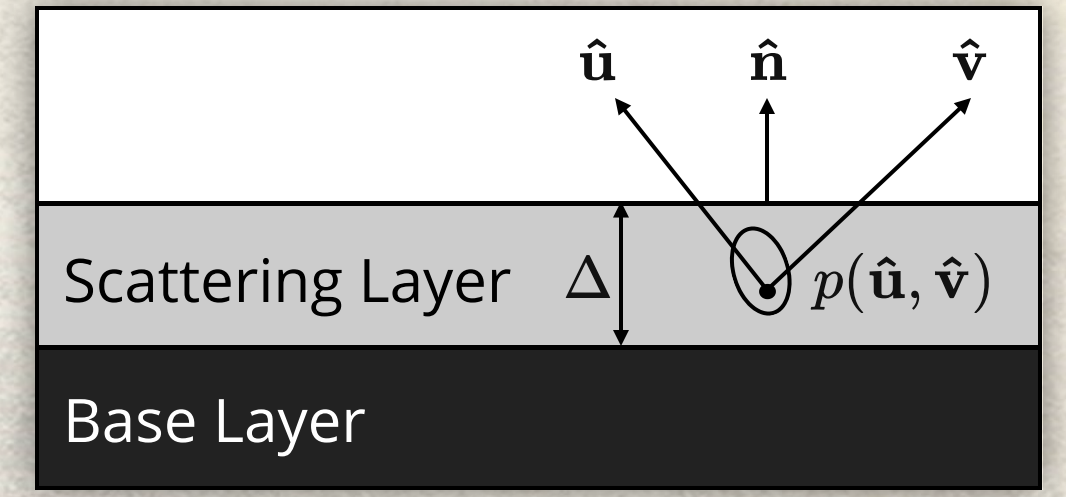
Anisotropic Fuzz

- Asperity scattering
  - … AKA sheen
  - … AKA fuzziness
  - … AKA fuzz
- New BRDF was researched during look development
- Realistic velvet, including crushed velvet, was important at the time
- In Ghost, used for moss, felt, horse coats, and some cloth.

- Our old model was ad hoc: Fresnel-like term + wrap lighting
- Neubelt & Pettineo (Ready at Dawn) presented their model in this course in 2013
  - Uses microfacet framework
  - Energy conserving
- *The Secret of Velvety Skin* (Koenderink and Pont, 2002)
  - Models fuzziness as a thin single-scattering layer
- No existing models supported anisotropy, to our knowledge

Anisotropic Fuzz | New Model

- Our new model was inspired by Koenderink and Pont's model
- Extended to support:
  - Anisotropy
  - Shadowing of base layer
  - Spherical harmonic lighting



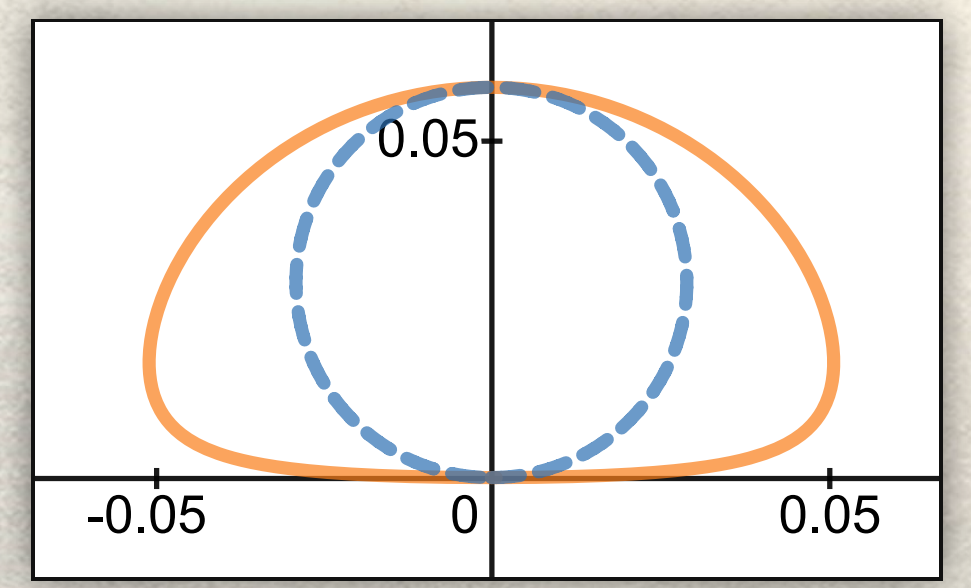
- $\hat{\mathbf{u}}$: Light direction
- $\hat{\mathbf{v}}$: View direction
- $p(\hat{\mathbf{u}}, \hat{\mathbf{v}})$: Phase function
- Δ: Thickness of scattering layer
- λ: Mean free path
- $d = \frac{\Delta}{\lambda} \ll 1$: Density

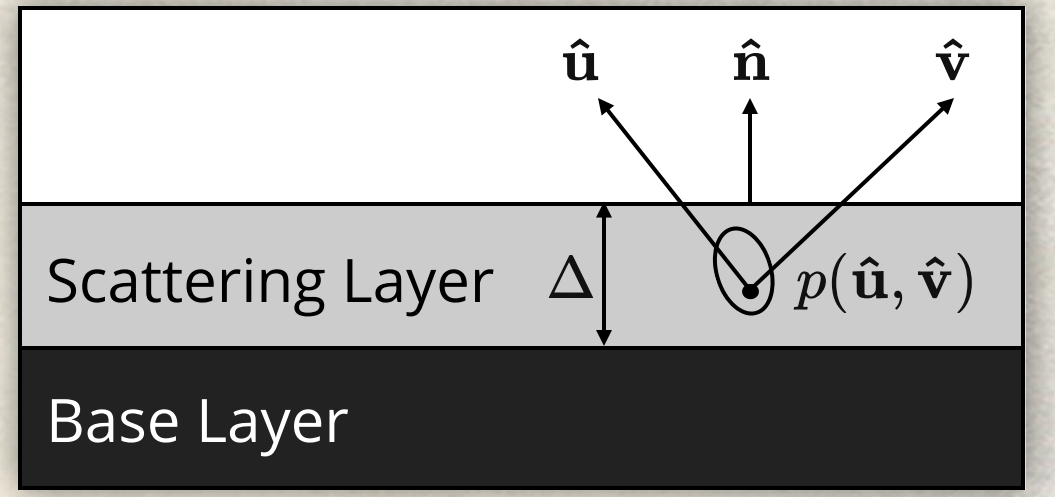
Anisotropic Fuzz | New Model

- Koenderink and Pont show that the BRDF of the scattering layer is:

$$f(\hat{\mathbf{u}}, \hat{\mathbf{v}}) = \mathbf{c}_f p(\hat{\mathbf{u}}, \hat{\mathbf{v}}) \frac{\left(1 - e^{-d \frac{(\hat{\mathbf{u}}+\hat{\mathbf{v}})\cdot\hat{\mathbf{n}}}{(\hat{\mathbf{u}}\cdot\hat{\mathbf{v}})(\hat{\mathbf{v}}\cdot\hat{\mathbf{n}})}}\right)}{(\hat{\mathbf{u}}+\hat{\mathbf{v}})\cdot\hat{\mathbf{n}}}$$

- Isotropic case:

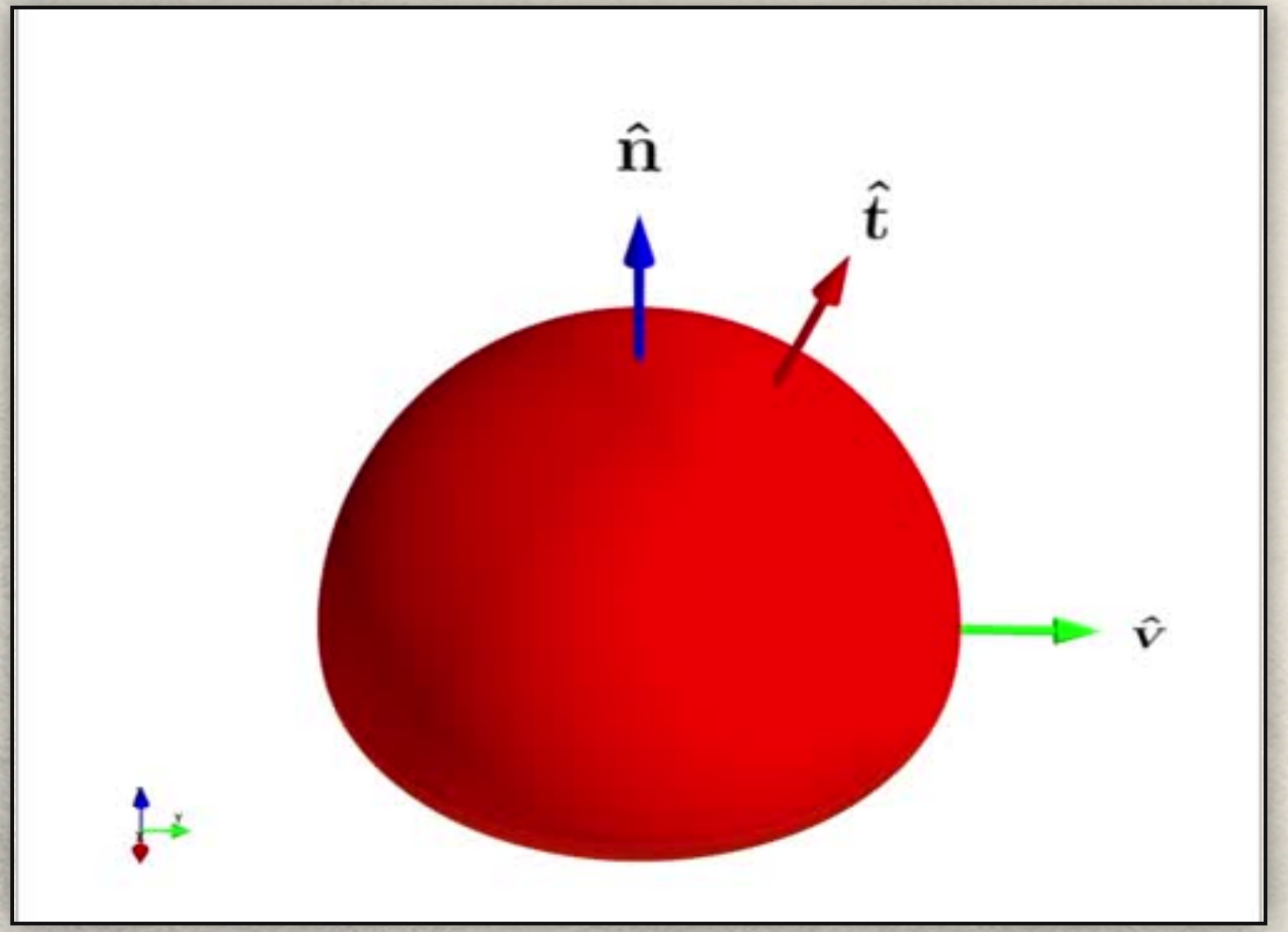




- $\hat{\mathbf{u}}$: Light direction
- $\hat{\mathbf{v}}$: View direction
- $p(\hat{\mathbf{u}}, \hat{\mathbf{v}})$: Phase function
- Δ: Thickness of scattering layer
- λ: Mean free path
- $d = \frac{\Delta}{\lambda} \ll 1$: Density
- $\mathbf{c}_f$: Albedo

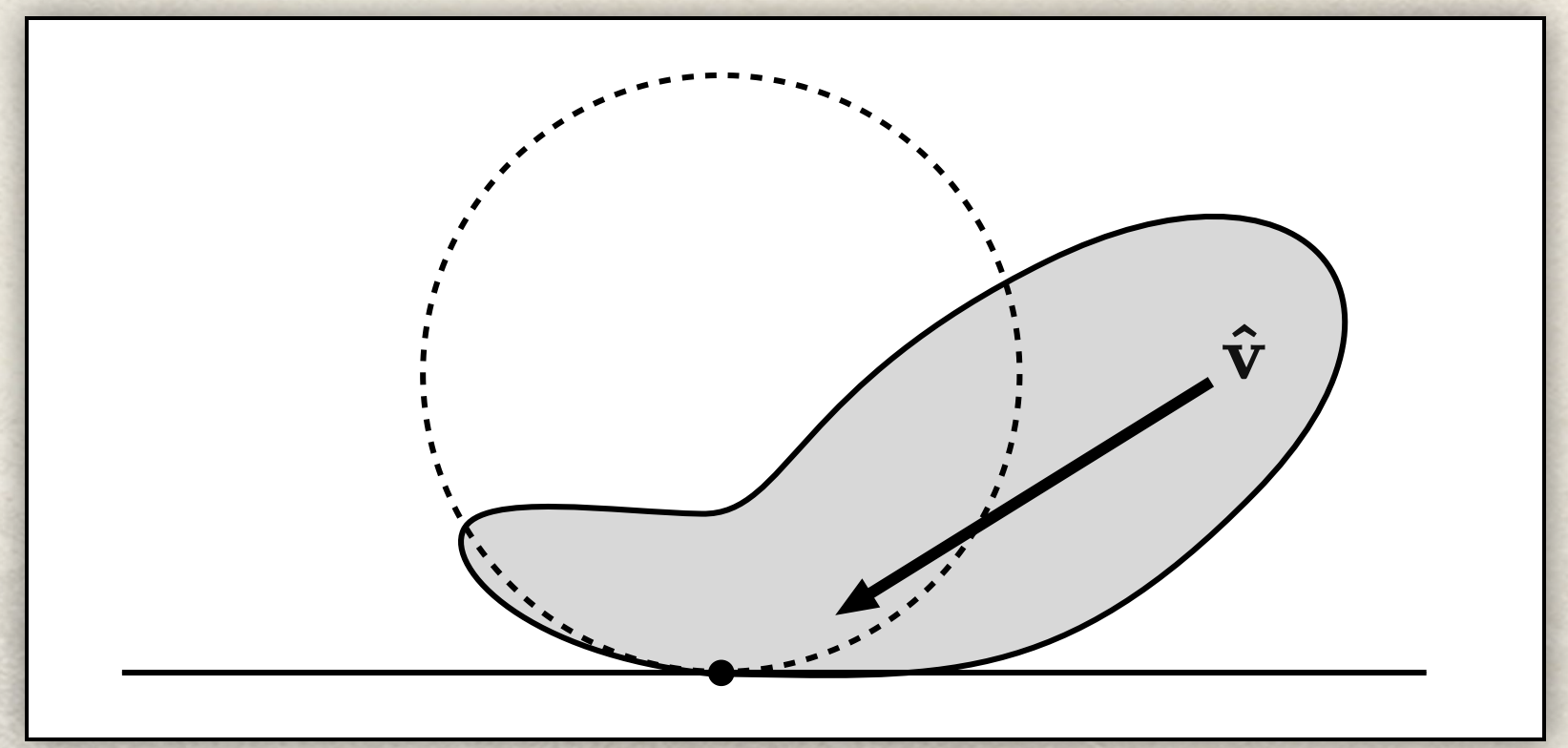
Anisotropic Fuzz | New Model

- Strongly dependent on view direction

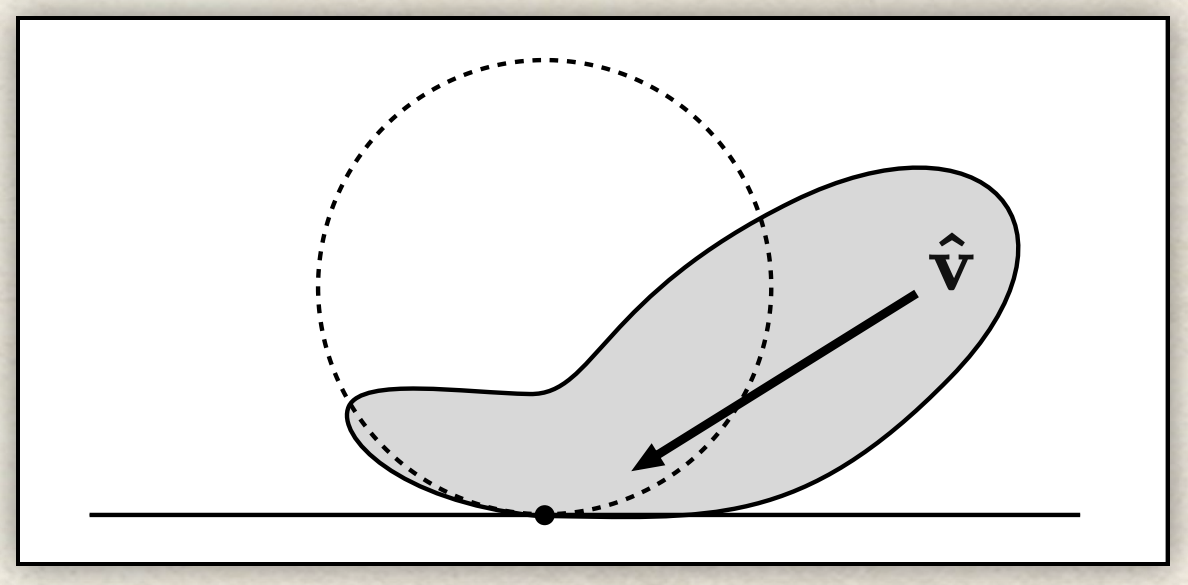


Anisotropic Fuzz | Velvet Scattering

- According to Koenderink and Pont, the scattering diagram for black velvet resembles:



- According to Koenderink and Pont, the scattering diagram for black velvet resembles:



- Strong back scattering lobe
- Smaller forward scattering lobe
- Can we find a phase function that reproduces this?

- Starting with the Schlick approximation to the Henyey-Greenstein phase function:

$$p_{\text{Schlick}}(k, \cos\theta) = \frac{1}{4\pi} \frac{1-k^2}{(1-k\cos\theta)^2}$$

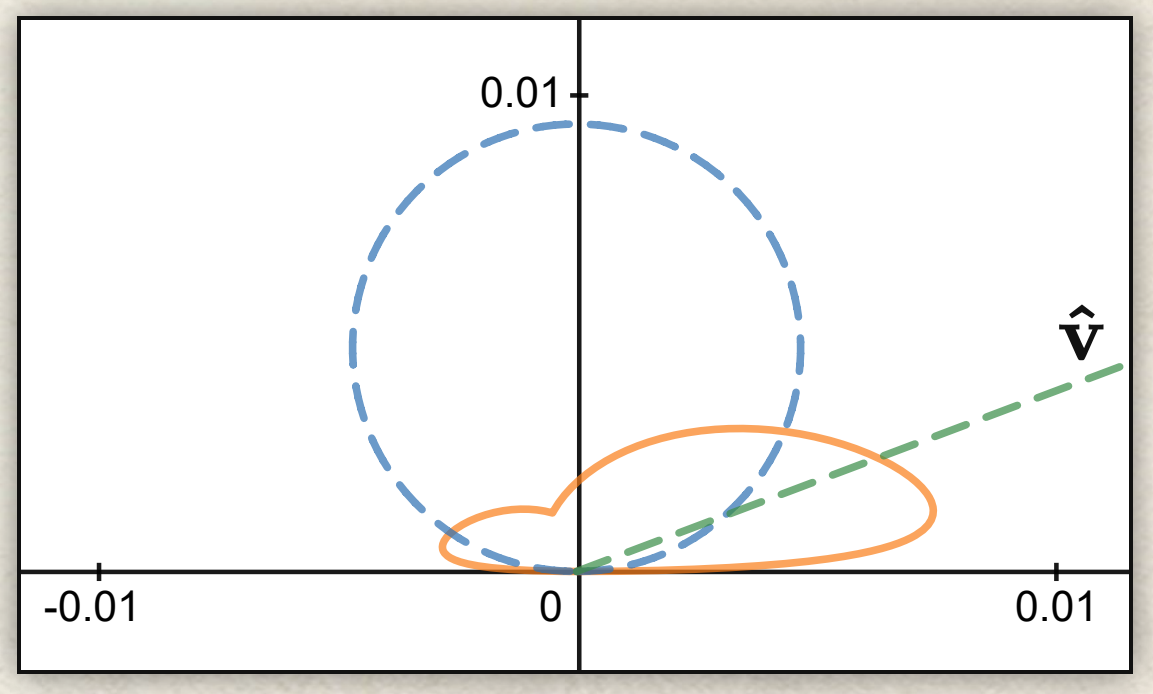
  - k: asymmetry parameter
- Make it dependent on the angle between the half vector $\hat{\mathbf{h}}$ and the fiber direction $\hat{\mathbf{t}}$:

$$p_V(k, \hat{\mathbf{h}}, \hat{\mathbf{t}}) = r(k) p_{\text{Schlick}}(k, \sin(\hat{\mathbf{t}}, \hat{\mathbf{h}}))$$

  - $r(k)$: normalization factor (approximate and conservative)

Anisotropic Fuzz | Phase Function

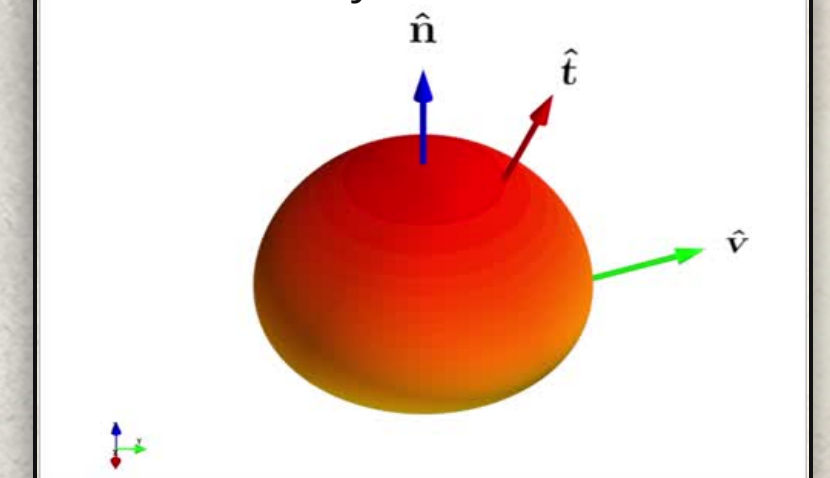
- Using this function in the BRDF, we get:



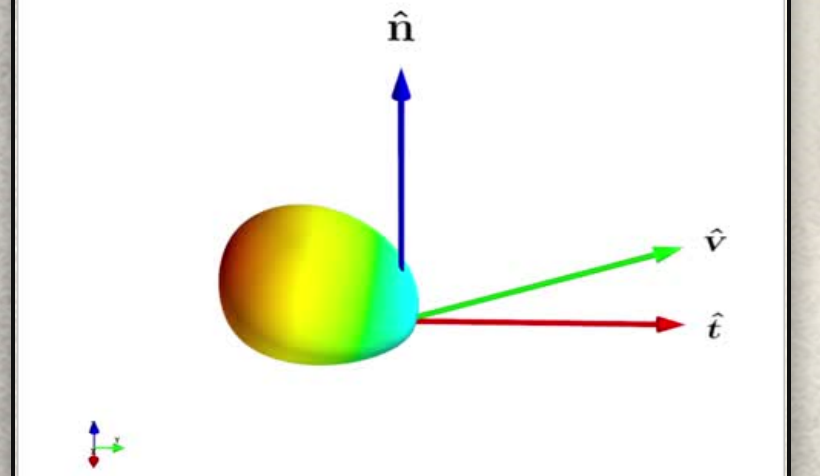
- … which is qualitatively similar, at least.

Anisotropic Fuzz | Phase Function

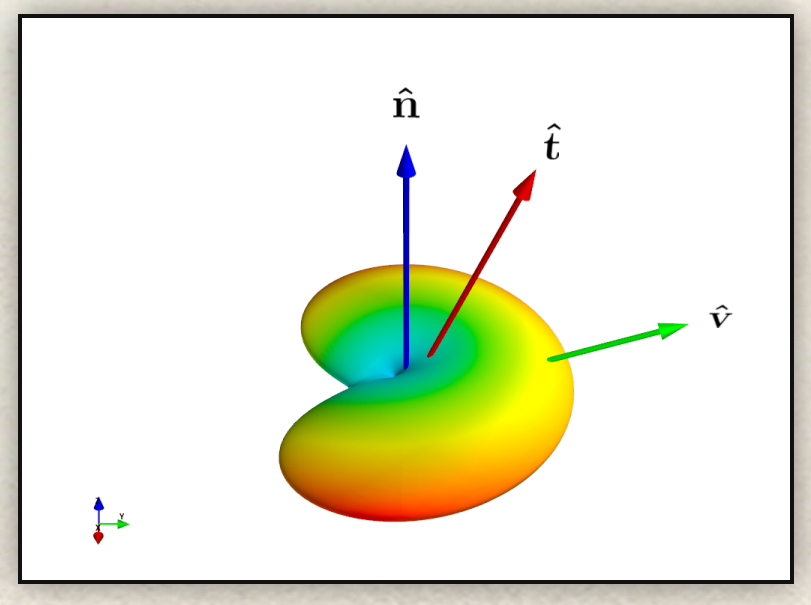
Vary Schlick k



Vary Fiber ($\hat{\mathbf{t}}$) Polar Angle

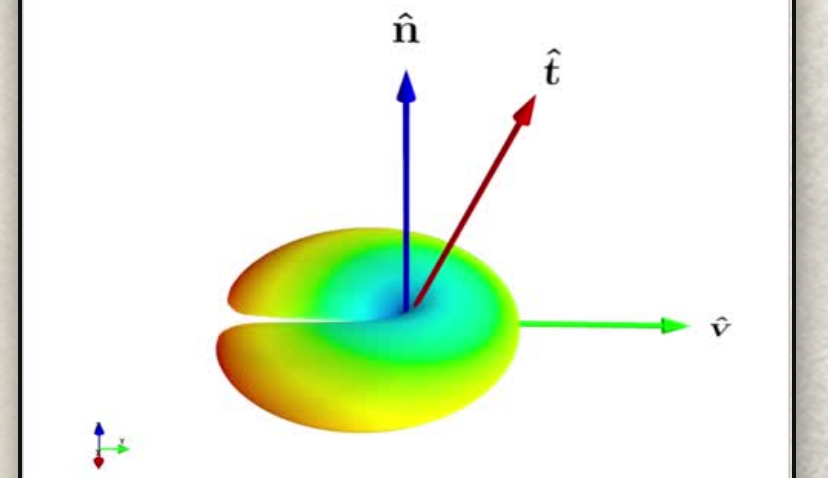


3D Plot:

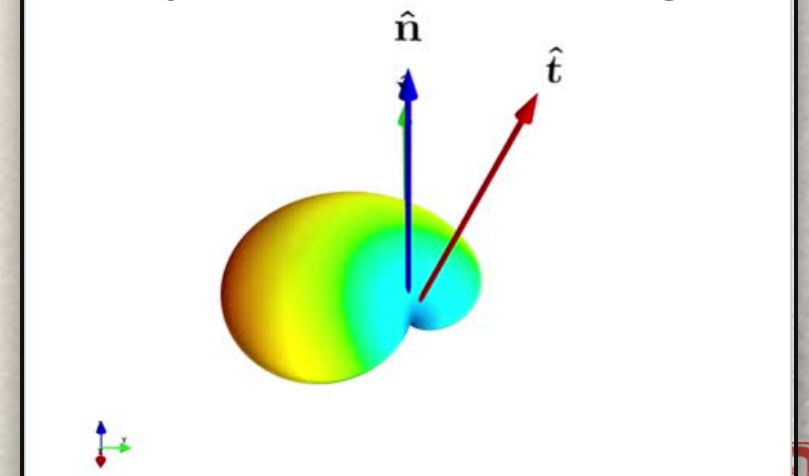


(Note: Normalized Amplitude)

Vary View ($\hat{\mathbf{v}}$) Polar Angle



Vary View ($\hat{\mathbf{v}}$) Azimuthal Angle



- More recent work: use a “fiber-like” specular SGGX phase function
  - Analytical normalization

$$D_F\left(\hat{\mathbf{h}}\right) = \frac{\alpha^3}{\pi\left(\left(\hat{\mathbf{h}}\cdot\hat{\mathbf{t}}\right)^2(1-\alpha^2)+\alpha^2\right)^2}$$

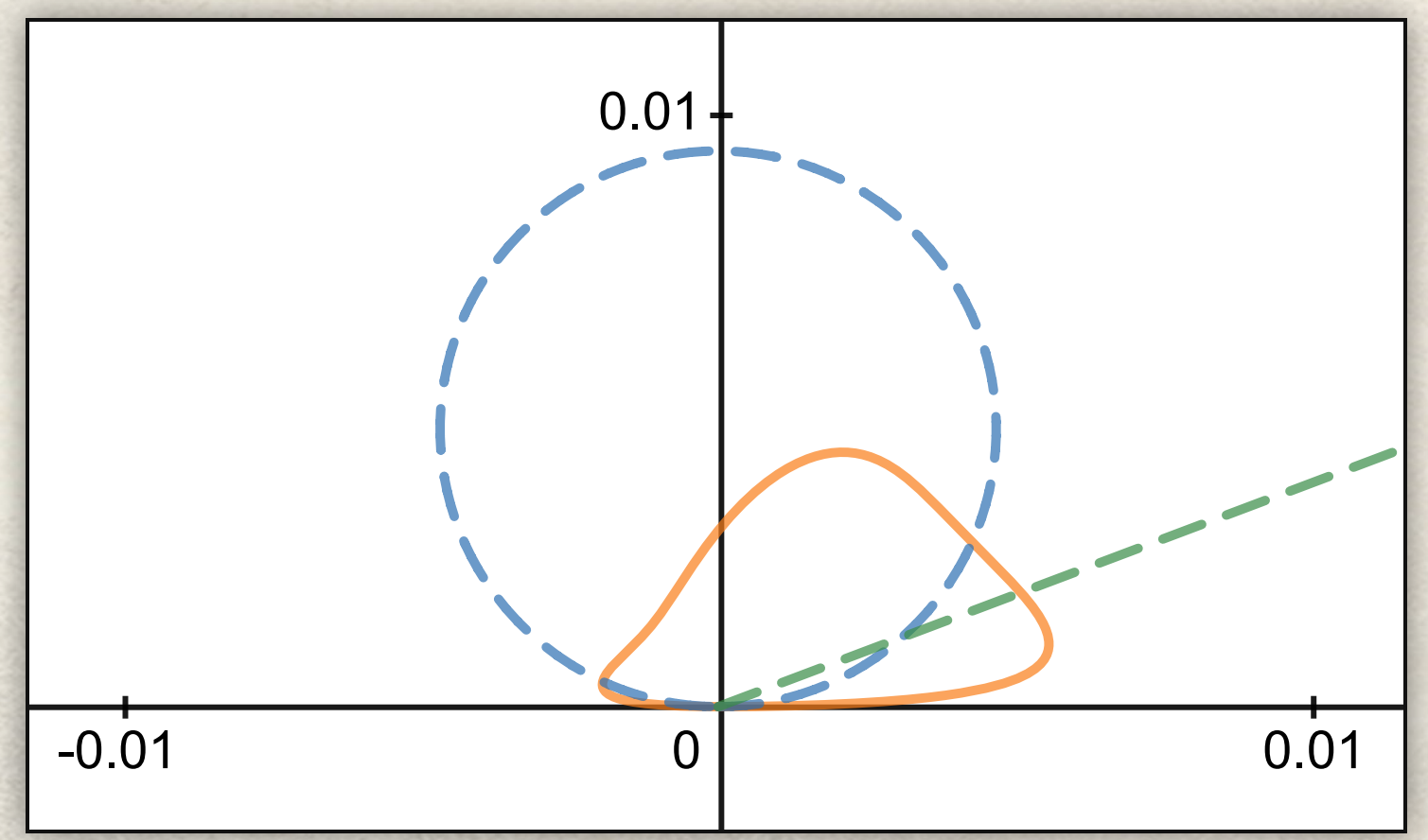
$$\sigma_F\left(\hat{\mathbf{u}}\right) = \sqrt{1-\left(\hat{\mathbf{u}}\cdot\hat{\mathbf{t}}\right)^2\left(1-\alpha^2\right)}$$

$$p_F\left(\hat{\mathbf{u}},\hat{\mathbf{v}}\right) = \frac{D\left(\hat{\mathbf{h}}\right)}{4\sigma(\hat{\mathbf{u}})}$$

- $\hat{\mathbf{u}}$: Light direction
- $\hat{\mathbf{v}}$: View direction
- $\hat{\mathbf{h}} = \frac{\hat{\mathbf{u}}+\hat{\mathbf{v}}}{|\hat{\mathbf{u}}+\hat{\mathbf{v}}|}$: Half vector
- $\hat{\mathbf{t}}$: Fiber direction
- α: GGX roughness
- D_F: (S)GGX “fiber” NDF
- σ_F: Microflake projected area
- $p_F(\hat{\mathbf{u}},\hat{\mathbf{v}})$: Phase function

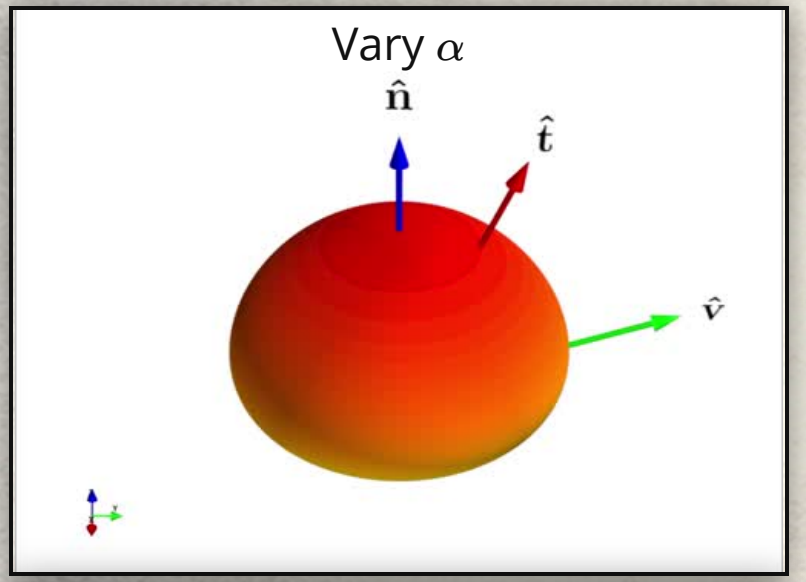
Anisotropic Fuzz | Phase Function

- Using the SGGX phase function we get:

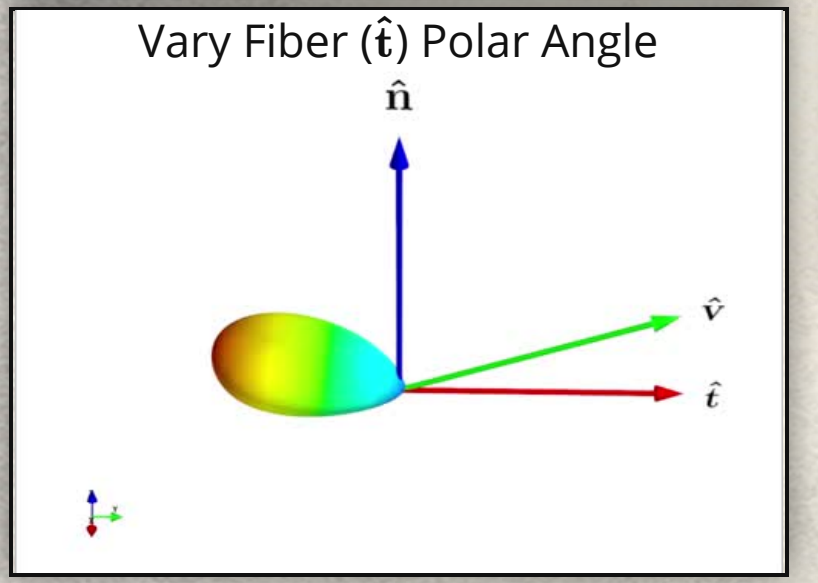


Anisotropic Fuzz | Phase Function

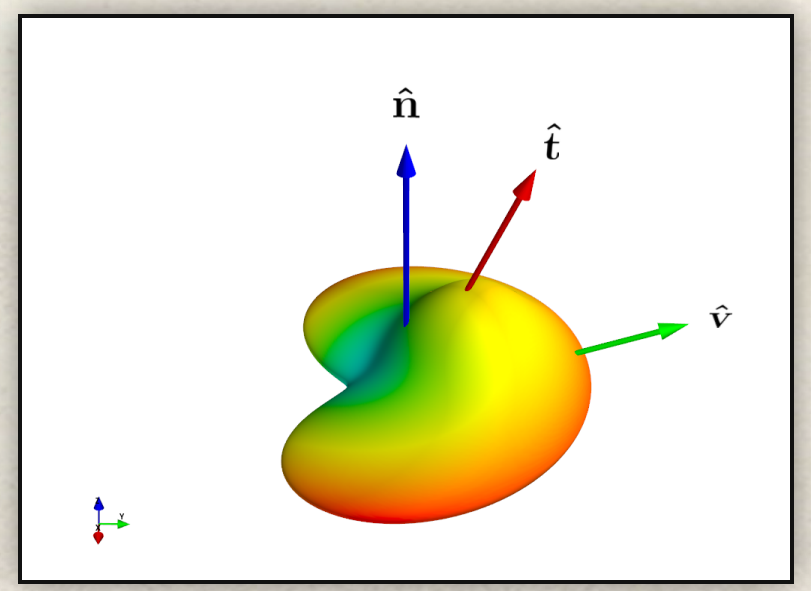
Vary α



Vary Fiber ($\hat{\mathbf{t}}$) Polar Angle

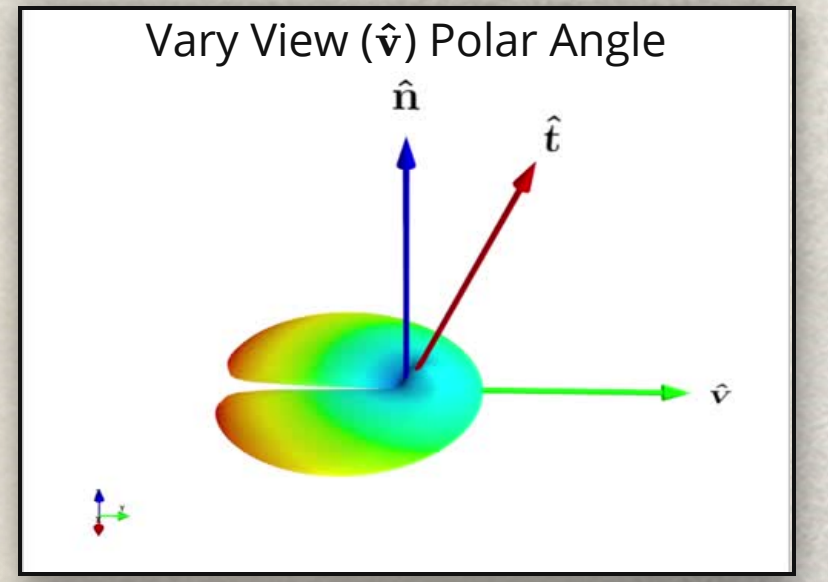


3D Plot:

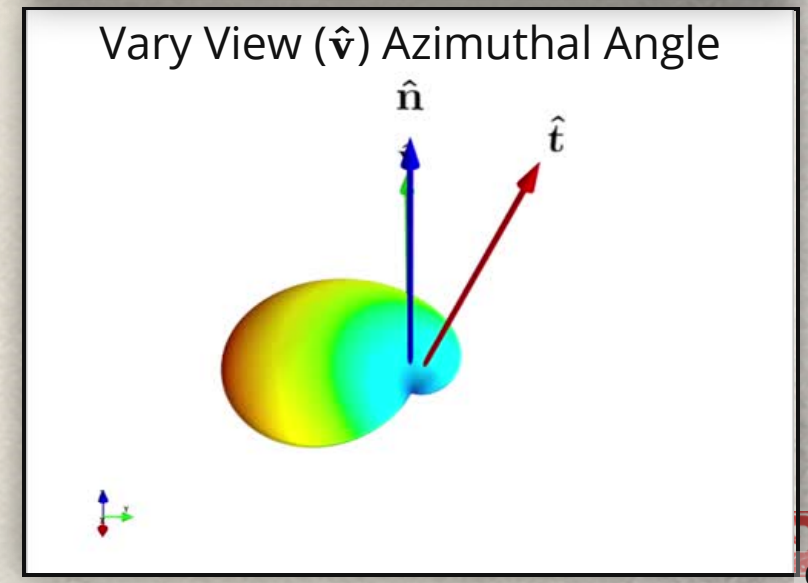


(Note: Normalized Amplitude)

Vary View ($\hat{\mathbf{v}}$) Polar Angle



Vary View ($\hat{\mathbf{v}}$) Azimuthal Angle



- Need to account for shadowing of base layer by scattering layer
- Don't account for all interactions to keep things simple(r)
- Probability of a ray penetrating to the base layer without scattering is:

$$P_p(\hat{\mathbf{u}}) = e^{-\frac{d}{\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}}}$$

- Probability of a ray reflected by the base layer escaping is:

$$P_e(\hat{\mathbf{v}}) = e^{-\frac{d}{\hat{\mathbf{v}} \cdot \hat{\mathbf{n}}}}$$

- Probability that an incident ray is scattered towards the base layer is:

$$P_s(\hat{\mathbf{u}}) = 1 - \int_{\Omega} p(\hat{\mathbf{u}}, \hat{\mathbf{v}}) \, \mathrm{d}\omega_{\hat{\mathbf{v}}}$$

$$P_p(\hat{\mathbf{u}}) = e^{-\frac{d}{\hat{\mathbf{u}}\cdot\hat{\mathbf{n}}}} \qquad P_e(\hat{\mathbf{v}}) = e^{-\frac{d}{\hat{\mathbf{v}}\cdot\hat{\mathbf{n}}}} \qquad P_s(\hat{\mathbf{u}}) = 1 - \int_\Omega p(\hat{\mathbf{u}}, \hat{\mathbf{v}})\, \mathrm{d}\omega_{\hat{\mathbf{v}}}$$

- Combining these, we get the following for the base layer attenuation:

$$\mathbf{c}_{\text{atten}}(\hat{\mathbf{u}}, \hat{\mathbf{v}}) = \left[P_p(\hat{\mathbf{u}}) + \mathbf{c}_f(1 - P_p(\hat{\mathbf{u}}))P_s(\hat{\mathbf{u}})\right] P_e(\hat{\mathbf{v}})$$

- No analytical expression for P_s (even for SGGX 😔)
- Use the following approximation:

$$P_s(\hat{\mathbf{u}}) \approx c_0 + c_1\hat{\mathbf{u}}\cdot\hat{\mathbf{n}} + c_2\hat{\mathbf{u}}\cdot\hat{\mathbf{t}} + c_3(\hat{\mathbf{u}}\cdot\hat{\mathbf{n}})^2 + c_4(\hat{\mathbf{u}}\cdot\hat{\mathbf{n}})(\hat{\mathbf{u}}\cdot\hat{\mathbf{t}}) + c_5(\hat{\mathbf{u}}\cdot\hat{\mathbf{t}})^2$$

Anisotropic Fuzz | Ambient Lighting

- Want anisotropy to continue to be visible in ambient lighting conditions
  - For us this means spherical harmonic lighting + specular probes
- Consider the rendering equation for our BRDF:

- Want anisotropy to continue to be visible in ambient lighting conditions
  - For us this means spherical harmonic lighting + specular probes
- Consider the rendering equation for our BRDF:

$$L_o(\hat{\mathbf{v}}) = \mathbf{c}_f \int_\Omega p(\hat{\mathbf{u}}, \hat{\mathbf{v}}) g(\hat{\mathbf{u}}, \hat{\mathbf{v}}) L_i(\hat{\mathbf{u}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

- Approximate using:

$$L_o(\hat{\mathbf{v}}) \approx \frac{\mathbf{c}_f}{2\pi} \int_\Omega g(\hat{\mathbf{u}}, \hat{\mathbf{v}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}} \int_\Omega p(\hat{\mathbf{u}}, \hat{\mathbf{v}}) L_i(\hat{\mathbf{u}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}},$$

$$L_o(\hat{\mathbf{v}}) \approx \frac{\mathbf{c}_f}{2\pi} \int_\Omega g(\hat{\mathbf{u}}, \hat{\mathbf{v}})\, \mathrm{d}\omega_{\hat{\mathbf{u}}} \int_\Omega p(\hat{\mathbf{u}}, \hat{\mathbf{v}}) L_i(\hat{\mathbf{u}})\, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

- Now assume that L_i is approximately constant

$$L_o(\hat{\mathbf{v}}) \approx \frac{\mathbf{c}_f}{4\pi^2} \int_\Omega g(\hat{\mathbf{u}}, \hat{\mathbf{v}})\, \mathrm{d}\omega_{\hat{\mathbf{u}}} \int_\Omega L_i(\hat{\mathbf{u}})\, \mathrm{d}\omega_{\hat{\mathbf{u}}} \int_\Omega p(\hat{\mathbf{u}}, \hat{\mathbf{v}})\, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

- We've broken it down into:

$$L_o(\hat{\mathbf{v}}) \approx \mathbf{c}_f$$

$$\times \frac{1}{2\pi} \int_\Omega g(\hat{\mathbf{u}}, \hat{\mathbf{v}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}} \approx \frac{d(1+c_0 \hat{\mathbf{v}}\cdot\hat{\mathbf{n}}+c_1(\hat{\mathbf{v}}\cdot\hat{\mathbf{n}})^2)}{d+c_2 \hat{\mathbf{v}}\cdot\hat{\mathbf{n}}+c_3(\hat{\mathbf{v}}\cdot\hat{\mathbf{n}})^3} \equiv \widetilde{G}(d, \hat{\mathbf{v}})$$

$$\times \frac{1}{2\pi} \int_\Omega L_i(\hat{\mathbf{u}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$ Easy to compute from SH coefficients

$$\times \int_\Omega p(\hat{\mathbf{u}}, \hat{\mathbf{v}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}} = 1 - P_s(\hat{\mathbf{v}})$$

- Also need to account for ambient shadowing of base layer
- First consider diffuse ambient lighting only
- Define:
  - E_p: Irradiance of the light penetrating the scattering layer
  - E_s: Irradiance of the light scattered down by the scattering layer
  - $\mathbf{c}_b$: Diffuse albedo of the base layer
  - $L_{b,o}$: Luminance of the base layer due to diffuse ambient lighting

$$L_{b,o}(\hat{\mathbf{v}}) = \frac{\mathbf{c}_b}{\pi} P_e(\hat{\mathbf{v}}) \left(E_p + E_s\right)$$

$$E_p = \int_\Omega P_p(\hat{\mathbf{u}}) L_i(\hat{\mathbf{u}}) (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

$$E_s = \mathbf{c}_f \int_\Omega \left[1 - P_p(\hat{\mathbf{u}})\right] P_s(\hat{\mathbf{u}}) L_i(\hat{\mathbf{u}}) (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

Bonus Slide

$$E_p = \int_\Omega P_p(\hat{\mathbf{u}}) L_i(\hat{\mathbf{u}}) (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

- Approximate using:

$$E_p \approx \frac{1}{\pi} \int_\Omega P_p(\hat{\mathbf{u}}) (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}} \int_\Omega L_i(\hat{\mathbf{u}}) (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

  - Exact for uniform lighting

Bonus Slide

- We've broken it down into:

$$E_p \approx$$

$$\frac{1}{\pi}\int_\Omega P_p(\hat{\mathbf{u}})(\hat{\mathbf{u}}\cdot\hat{\mathbf{n}})\,\mathrm{d}\omega_{\hat{\mathbf{u}}} \qquad \approx 1 + c_0 d + c_1 d^2 \equiv \widetilde{Q}(d)$$

$$\times \int_\Omega L_i(\hat{\mathbf{u}})(\hat{\mathbf{u}}\cdot\hat{\mathbf{n}})\,\mathrm{d}\omega_{\hat{\mathbf{u}}} \qquad \equiv E\text{, the irradiance on the surface (computed using SH)}$$

Bonus Slide

- Now for irradiance of light scattered onto base layer by scattering layer:

$$E_s = \mathbf{c}_f \int_\Omega \left[1 - P_p(\hat{\mathbf{u}})\right] P_s(\hat{\mathbf{u}}) L_i(\hat{\mathbf{u}})(\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

- Approximate using:

$$E_s \approx \frac{\mathbf{c}_f}{\pi^2} \int_\Omega \left[1 - P_p(\hat{\mathbf{u}})\right] (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}} \int_\Omega P_s(\hat{\mathbf{u}})(\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}} \int_\Omega L_i(\hat{\mathbf{u}})(\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

  - Exact for isotropic scattering and uniform lighting

Bonus Slide

- We've broken it down into:

$$E_s \approx \mathbf{c}_f$$

$$\times \frac{1}{\pi} \int_\Omega \left[1 - P_p(\hat{\mathbf{u}})\right] (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}} \quad = 1 - \frac{1}{\pi} \int_\Omega P_p(\hat{\mathbf{u}}) (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

$$\times \int_\Omega P_s(\hat{\mathbf{u}}) (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}} \quad \equiv R(\theta_f, \alpha), \text{ constant for fixed } \theta_f \text{ and } \alpha$$

$$\times \int_\Omega L_i(\hat{\mathbf{u}}) (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \, \mathrm{d}\omega_{\hat{\mathbf{u}}} \quad \equiv E, \text{ the irradiance on the surface (computed using SH)}$$

Bonus Slide

- Putting it all together, we have

$$
\begin{aligned}
L_{b,o}(\hat{\mathbf{v}}) &= \frac{\mathbf{c}_b}{\pi} P_e(\hat{\mathbf{v}}) \left(E_p + E_s\right) \\
&\approx \frac{\mathbf{c}_b}{\pi} P_e(\hat{\mathbf{v}}) \left[\widetilde{Q} E + \mathbf{c}_f \left(1 - \widetilde{Q}\right) R E\right] \\
&= \frac{\mathbf{c}_b \mathbf{c}_{\text{atten}}}{\pi} E
\end{aligned}
$$

where

$$
\mathbf{c}_{\text{atten}} = P_e(\hat{\mathbf{v}}) \left[\widetilde{Q} + \mathbf{c}_f R \left(1 - \widetilde{Q}\right)\right]
$$

Bonus Slide

- For specular ambient shadowing, we follow a similar approach
  - Replacing SH sample with e.g. filtered reflection probe sample

$$\int_{\Omega} L_i(\hat{\mathbf{u}})(\hat{\mathbf{u}} \cdot \hat{\mathbf{n}})\, \mathrm{d}\omega_{\hat{\mathbf{u}}} \rightarrow \int_{\Omega} f_{spec}(\hat{\mathbf{u}}, \hat{\mathbf{v}}) L_i(\hat{\mathbf{u}})(\hat{\mathbf{u}} \cdot \hat{\mathbf{n}})\, \mathrm{d}\omega_{\hat{\mathbf{u}}}$$

  - This lets us use the same $\mathbf{c}_{\mathrm{atten}}$ attenuation term

Bonus Slide

Anisotropic Fuzz | Simplified Deferred Version

- For more common environmental materials (such as moss), we implemented a simplified version of this BRDF for deferred shading
  - 7-bit fuzziness value in G-Buffer (mapped to [0, 1])
- Used a fixed set of parameters:
  - Density $d = 0.5$
  - Spread $= 0.9$ ($k_{\text{Schlick}} \approx -0.1$, or $\alpha_{\text{SGGX}} \approx 0.95$)
  - $\hat{\mathbf{t}}_{\text{Fiber}} = \hat{\mathbf{n}}_{\text{Vertex}}$
  - Fiber albedo $\mathbf{c}_{\text{Fiber}} = 5 \cdot \mathbf{c}_{\text{Lambert}}$
- For performance reasons, did not include base layer shadowing; instead we used:

$$L_{\text{Diffuse}} = \text{lerp}\left(L_{\text{Lambert}}, L_{\text{Fuzz}}, \text{fuzziness}\right)$$

- In the forward-shaded version, we expose a parameter to smooth normals (towards vertex normals) to give the scattering layer a softer appearance
- Environment vertex normals were too inconsistent to use directly for shading
- Instead we used wrap lighting to soften normals (McAuley 2013):

$$\hat{\mathbf{u}} \cdot \hat{\mathbf{n}} \rightarrow \frac{1+n}{2(1+w)} \left(\frac{\max(\hat{\mathbf{u}} \cdot \hat{\mathbf{n}} + w, 0)}{1+w} \right)^n \text{ with } n = 2, w = 0.2$$

Anisotropic Fuzz | Results

- Ghost didn't have any crushed velvet-like materials that made full use of the BRDF's capabilities
- To demonstrate the technique, I created a “velvety” horse by modifying the Ghost horse shaders
- Note that these results use the SGGX phase function, which is different than what shipped in Ghost
  - In general, the results are qualitatively very similar
- Disclaimer: Programmer art ahead!


No Fuzz

Fuzziness Enabled With Parameters:

- Density: 0.2
- Spread: 0.2
- Fiber Tilt: 23 degrees
- Noisy Fiber Direction Map

No Base Layer Attenuation


Enabled Base Layer Attenuation
$P_s(\hat{\mathbf{u}}) \equiv 0$

$P_s(\hat{\mathbf{u}})$ implemented

Vary Density d $[0.0, 0.5]$

Vary Fiber Color Value

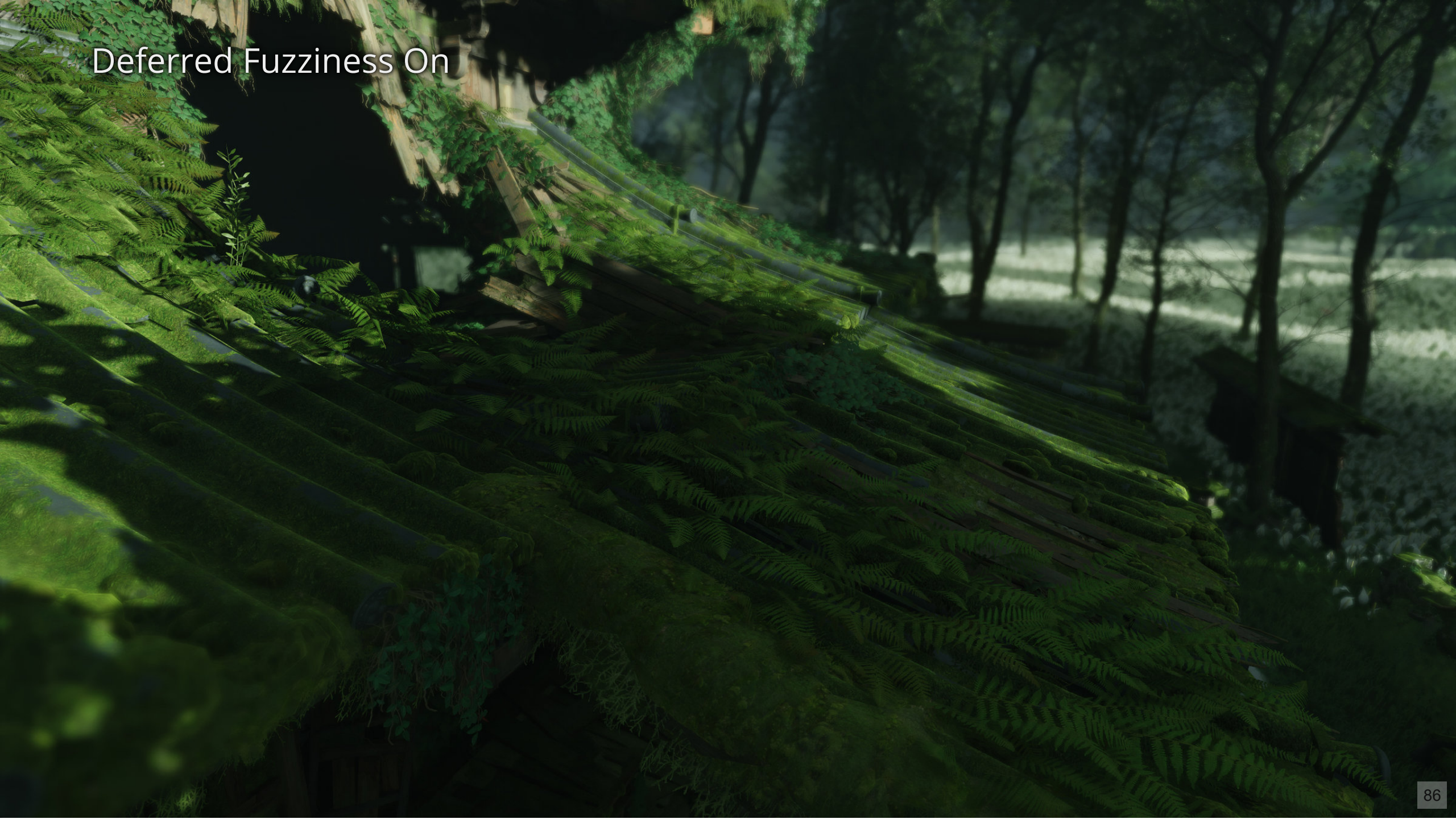
Vary Fiber Color Saturation

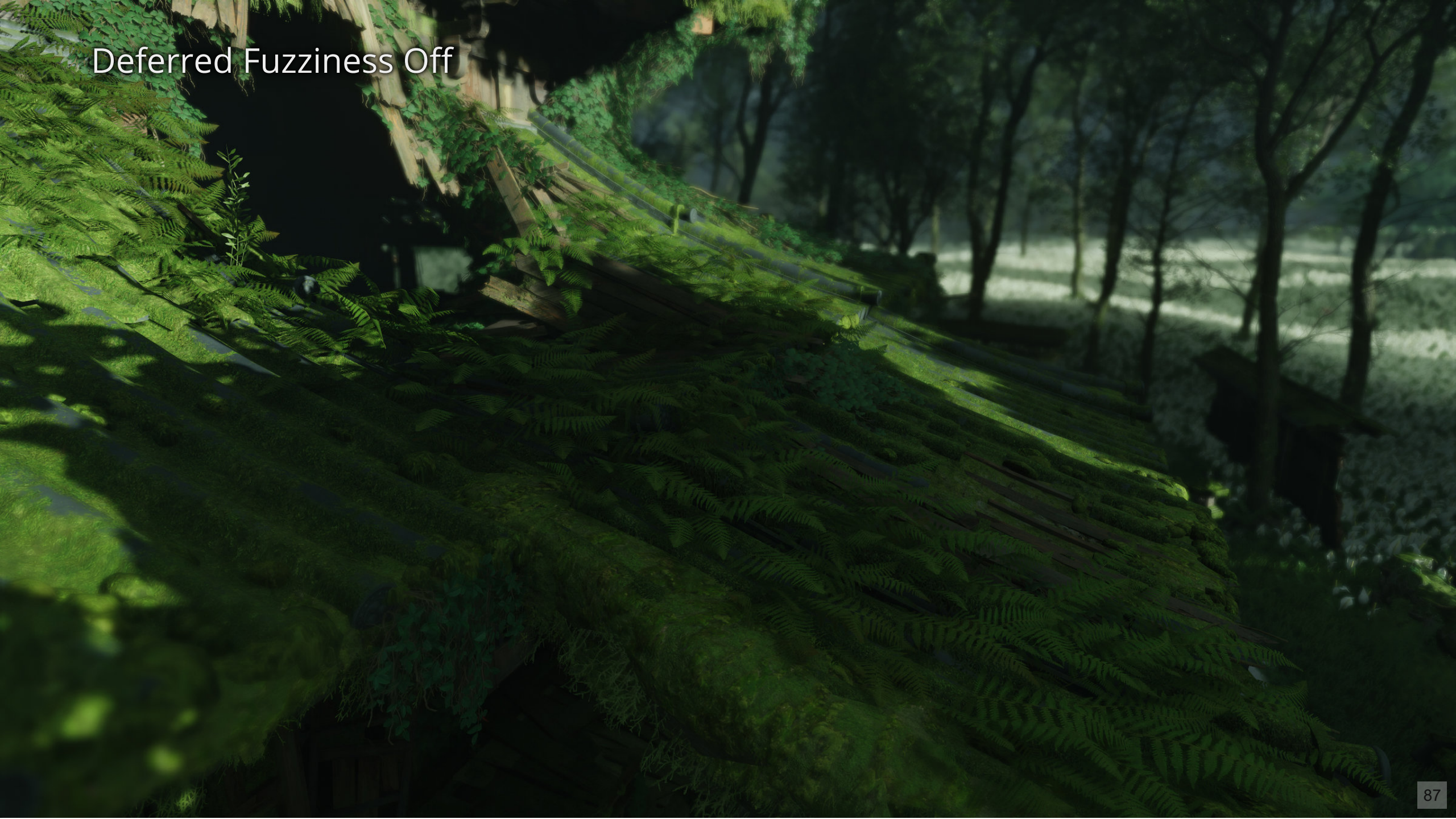
Vary Spread $[0.0, 1.0]$

Vary Fiber Tilt θ_t $[0.0, 90.0^\circ]$

Deferred Fuzziness On

Deferred Fuzziness Off


Deferred Fuzziness On


Deferred Fuzziness Off

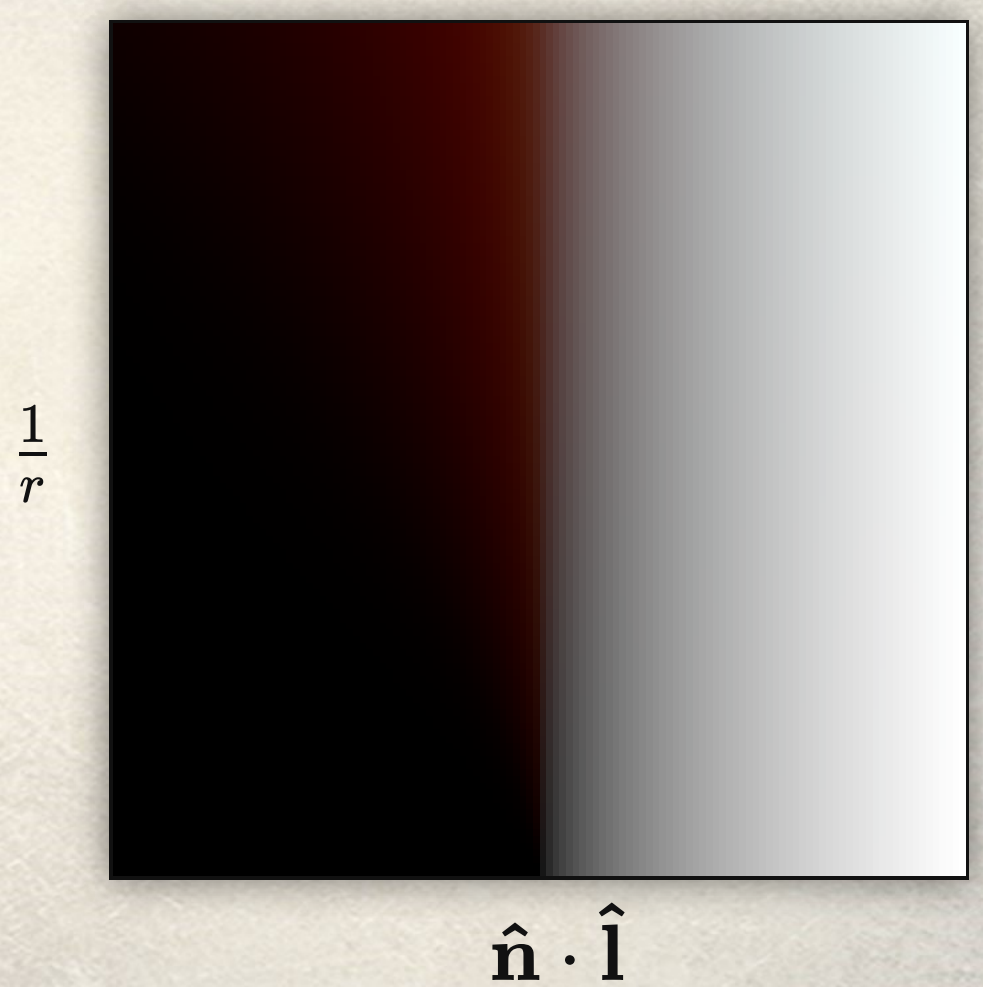
Anisotropic Fuzz | Summary

- New anisotropic fuzz BRDF
  - Able to reproduce the appearance of materials like crushed velvet
    - Under direct and indirect light
  - Inexpensive enough to be used on PlayStation 4 hardware


Skin Shading

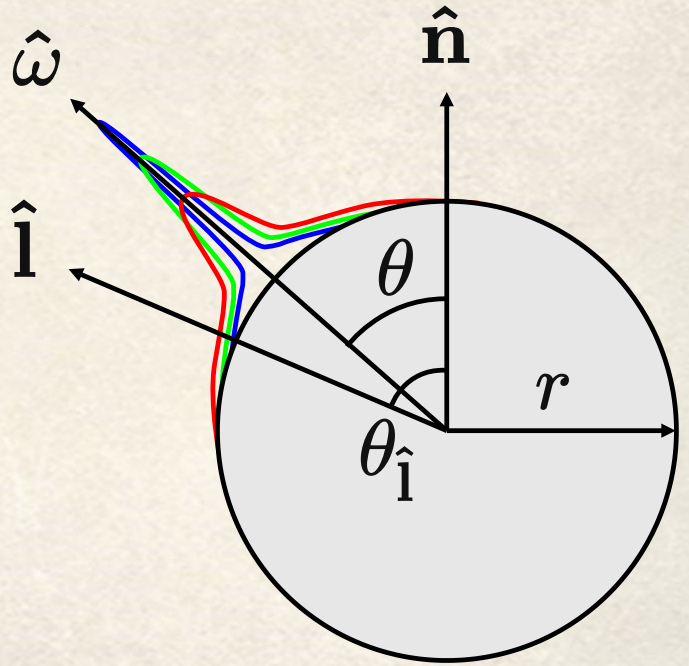
Skin Shading

- Using the *Pre-Integrated Skin Shading* technique (Penner and Borshukov, 2011)
- Uses LUTs to compute subsurface scattering based on $\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}$ and curvature $\kappa = \frac{1}{r}$

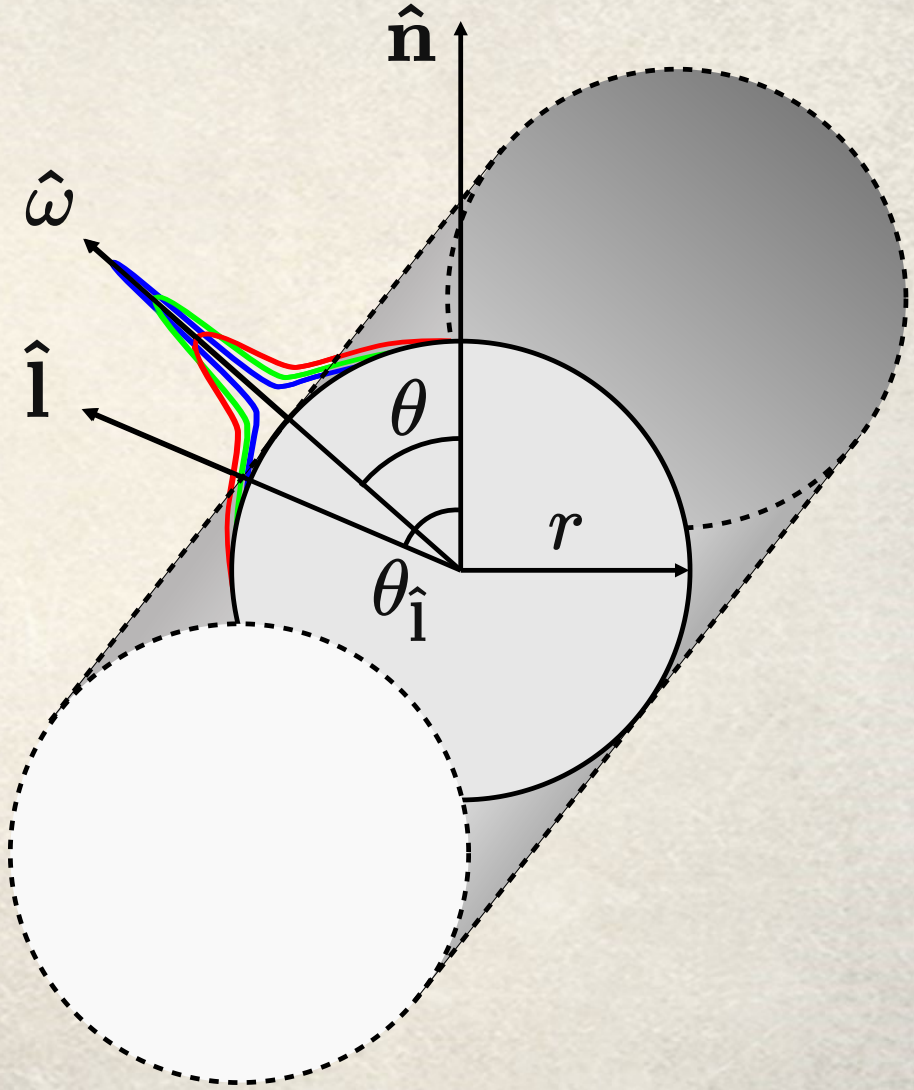


- The LUT is computed by performing integration on a “ring”:

$$D\left(r, \cos\theta_{\hat{\mathbf{l}}}\right) = \frac{\int_{-\pi}^{\pi} \cos\left(\theta_{\hat{\mathbf{l}}} - \theta\right) R\left(2r\sin\left(\frac{\theta}{2}\right)\right) \mathrm{d}\theta}{\int_{-\pi}^{\pi} R\left(2r\sin\left(\frac{\theta}{2}\right)\right) \mathrm{d}\theta}$$

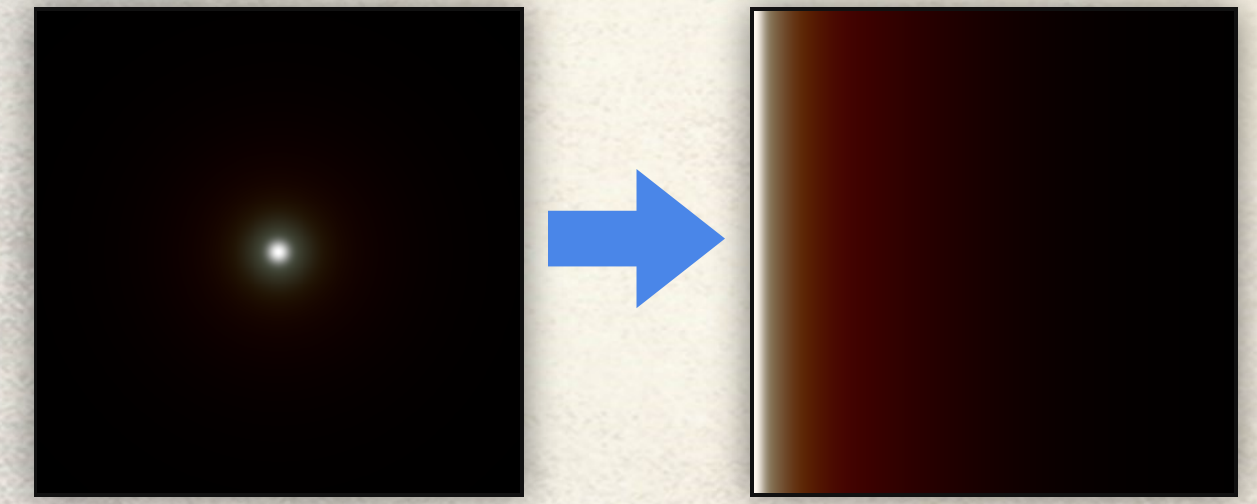


- But it's better to think about it as being on a cylinder instead.
- Two things to notice:
  1. We want the curvature in the direction of the light
  2. We need to take care about which scattering profile we're using



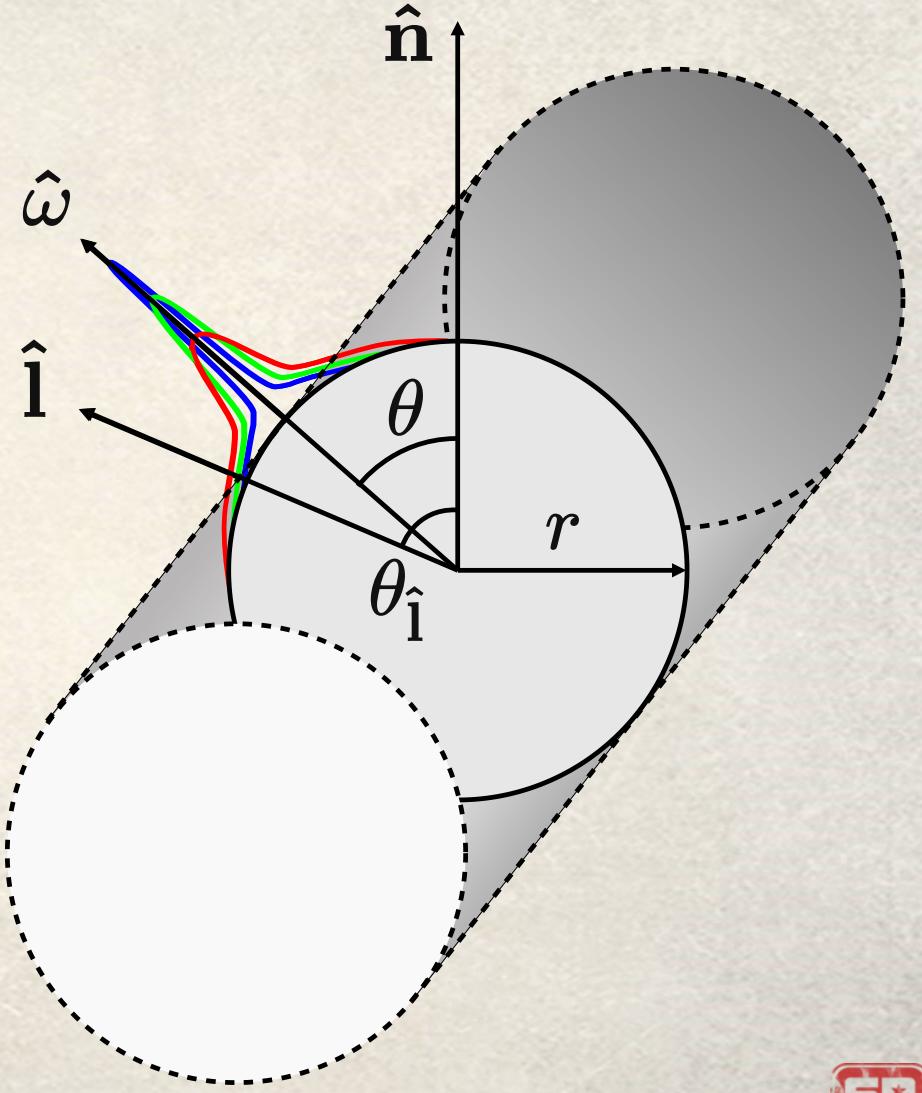
$$D\left(r, \cos\theta_{\hat{\mathbf{l}}}\right) = \frac{\int_{-\pi}^{\pi} \cos\left(\theta_{\hat{\mathbf{l}}} - \theta\right) R\left(2r\sin\left(\frac{\theta}{2}\right)\right) \mathrm{d}\theta}{\int_{-\pi}^{\pi} R\left(2r\sin\left(\frac{\theta}{2}\right)\right) \mathrm{d}\theta}$$

$$D\left(r, \cos\theta_{\hat{\mathrm{l}}}\right) = \frac{\int_{-\pi}^{\pi} \cos\left(\theta_{\hat{\mathrm{l}}} - \theta\right) R\left(2r\sin\left(\frac{\theta}{2}\right)\right) \mathrm{d}\theta}{\int_{-\pi}^{\pi} R\left(2r\sin\left(\frac{\theta}{2}\right)\right) \mathrm{d}\theta}$$



radial

linear



- We want the *directional* curvature, not just the mean curvature
- Can calculate directional curvature using the curvature tensor

$$\mathbb{II} = \begin{bmatrix} \hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2 \end{bmatrix} \begin{bmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2 \end{bmatrix}^T$$

  - $\hat{\mathbf{d}}_\mathbf{i}$: Principal directions
  - κ_i: Principal curvatures
- The curvature in direction $\hat{\mathbf{l}}$ is found by

$$\kappa_{\hat{\mathbf{l}}} = \hat{\mathbf{l}}^T \mathbb{II} \hat{\mathbf{l}}$$

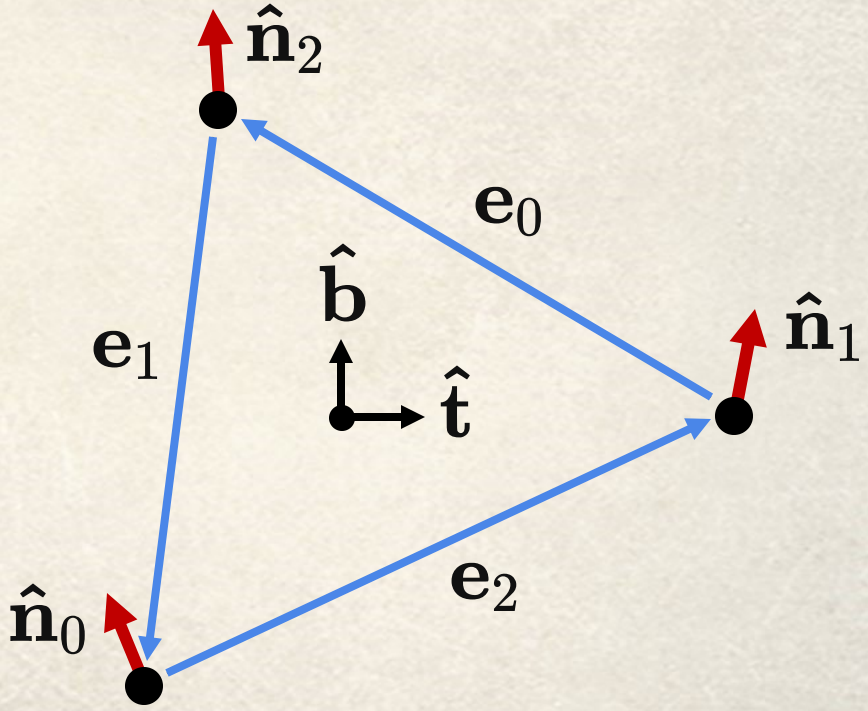
- The curvature tensor $\mathbb{II}$ is a symmetric 2x2 matrix
- Store $\mathbb{II}$ + mean curvature in 4-byte vertex channel
  - Ambient lighting uses mean curvature
- Calculate tangent-space curvature tensors as a pre-process
  - Used algorithm described in *Estimating Curvature and Their Derivatives on Triangle Meshes* (Rusinkiewicz, 2004)
- Blurred resulting curvature slightly
  - Smooths out very high curvature at isolated vertices
  - Used an equal blend of two Gaussians ($\sigma_1 = 0.8\ \mathrm{cm}, \sigma_2 = 0.23\ \mathrm{cm}$)
  - Clamped concave curvature to zero for blur

- Calculate $\mathbb{I}$ for each face using least squares with the following constraints:

$$\mathbb{I}\begin{bmatrix}\mathbf{e}_0\cdot\hat{\mathbf{t}}\\ \mathbf{e}_0\cdot\hat{\mathbf{b}}\end{bmatrix}=\begin{bmatrix}(\hat{\mathbf{n}}_2-\hat{\mathbf{n}}_1)\cdot\hat{\mathbf{t}}\\ (\hat{\mathbf{n}}_2-\hat{\mathbf{n}}_1)\cdot\hat{\mathbf{b}}\end{bmatrix}$$

$$\mathbb{I}\begin{bmatrix}\mathbf{e}_1\cdot\hat{\mathbf{t}}\\ \mathbf{e}_1\cdot\hat{\mathbf{b}}\end{bmatrix}=\begin{bmatrix}(\hat{\mathbf{n}}_0-\hat{\mathbf{n}}_2)\cdot\hat{\mathbf{t}}\\ (\hat{\mathbf{n}}_0-\hat{\mathbf{n}}_2)\cdot\hat{\mathbf{b}}\end{bmatrix}$$

$$\mathbb{I}\begin{bmatrix}\mathbf{e}_2\cdot\hat{\mathbf{t}}\\ \mathbf{e}_2\cdot\hat{\mathbf{b}}\end{bmatrix}=\begin{bmatrix}(\hat{\mathbf{n}}_1-\hat{\mathbf{n}}_0)\cdot\hat{\mathbf{t}}\\ (\hat{\mathbf{n}}_1-\hat{\mathbf{n}}_0)\cdot\hat{\mathbf{b}}\end{bmatrix}$$



- Combine face $\mathbb{I}$ on vertices using “Voronoi area” weighting
  - See paper for details

- Comparison of:
  - Zero curvature
  - Mean curvature
  - Directional curvature


Zero Curvature

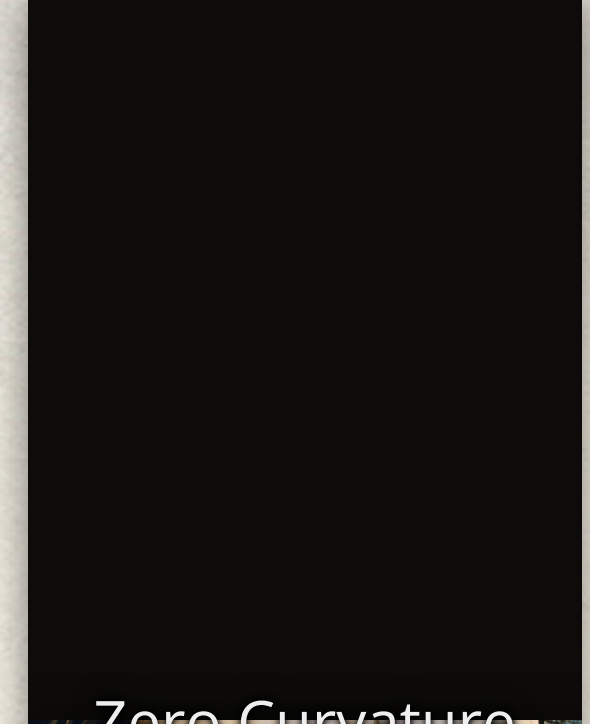

Mean Curvature

Directional Curvature

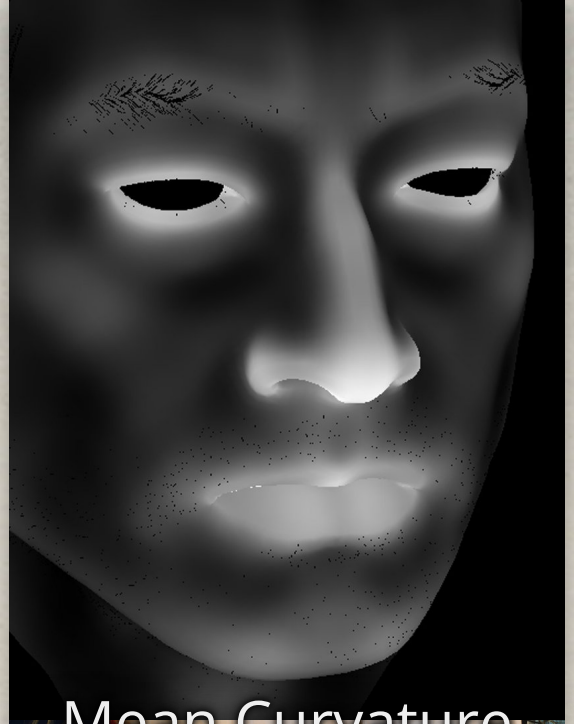

Zero Curvature


Mean Curvature

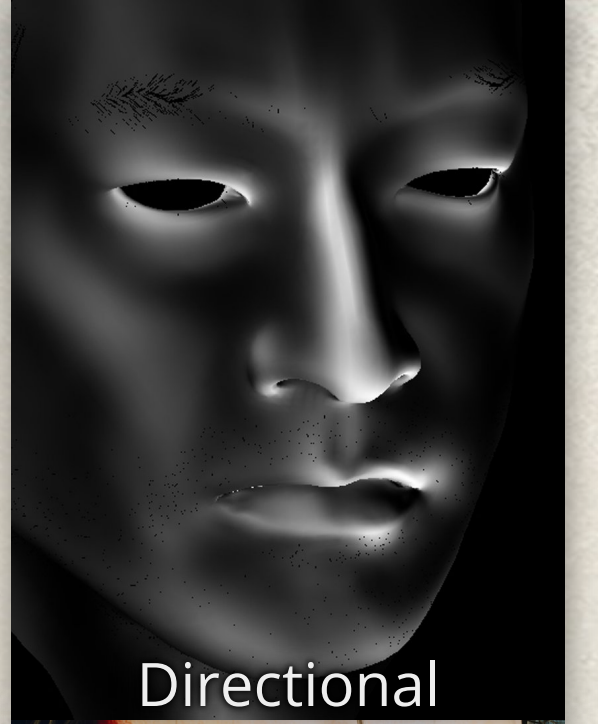

Directional Curvature



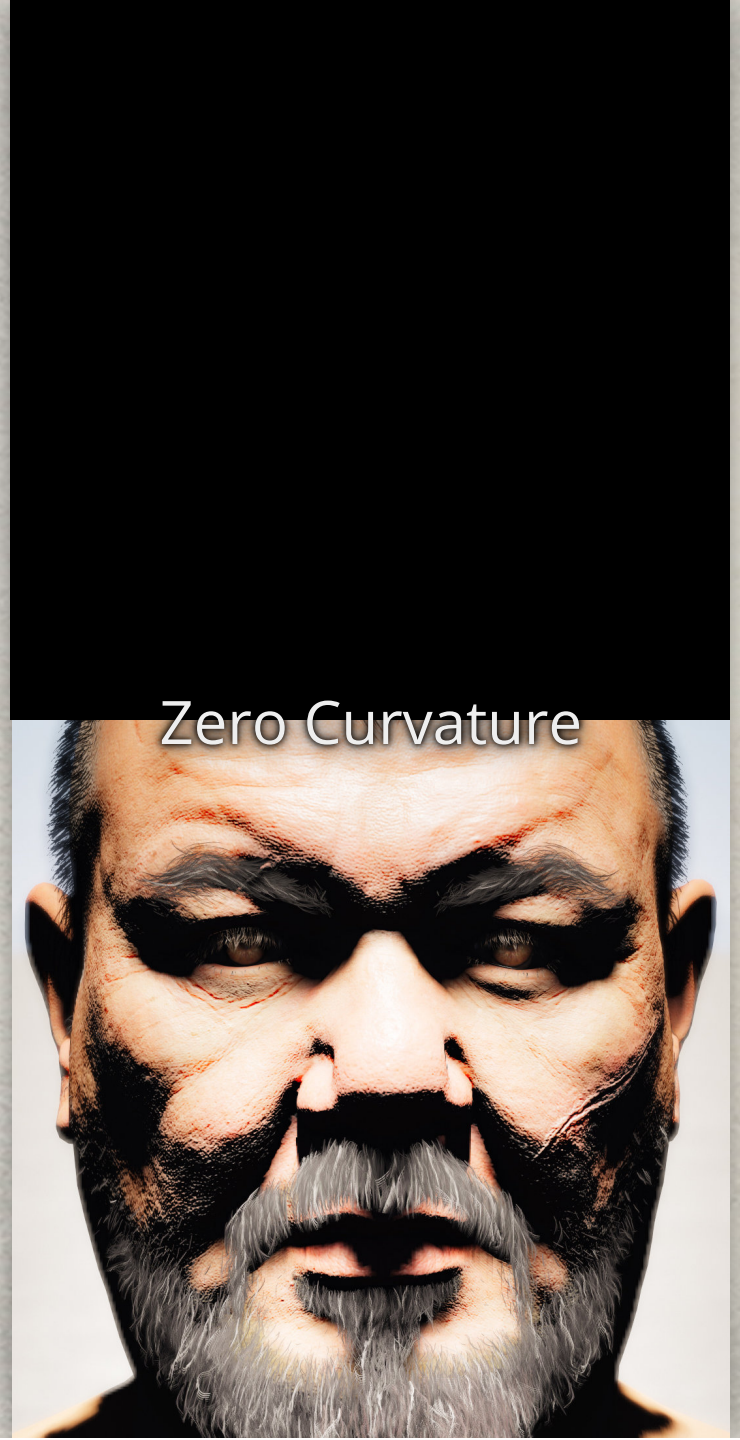
Zero Curvature



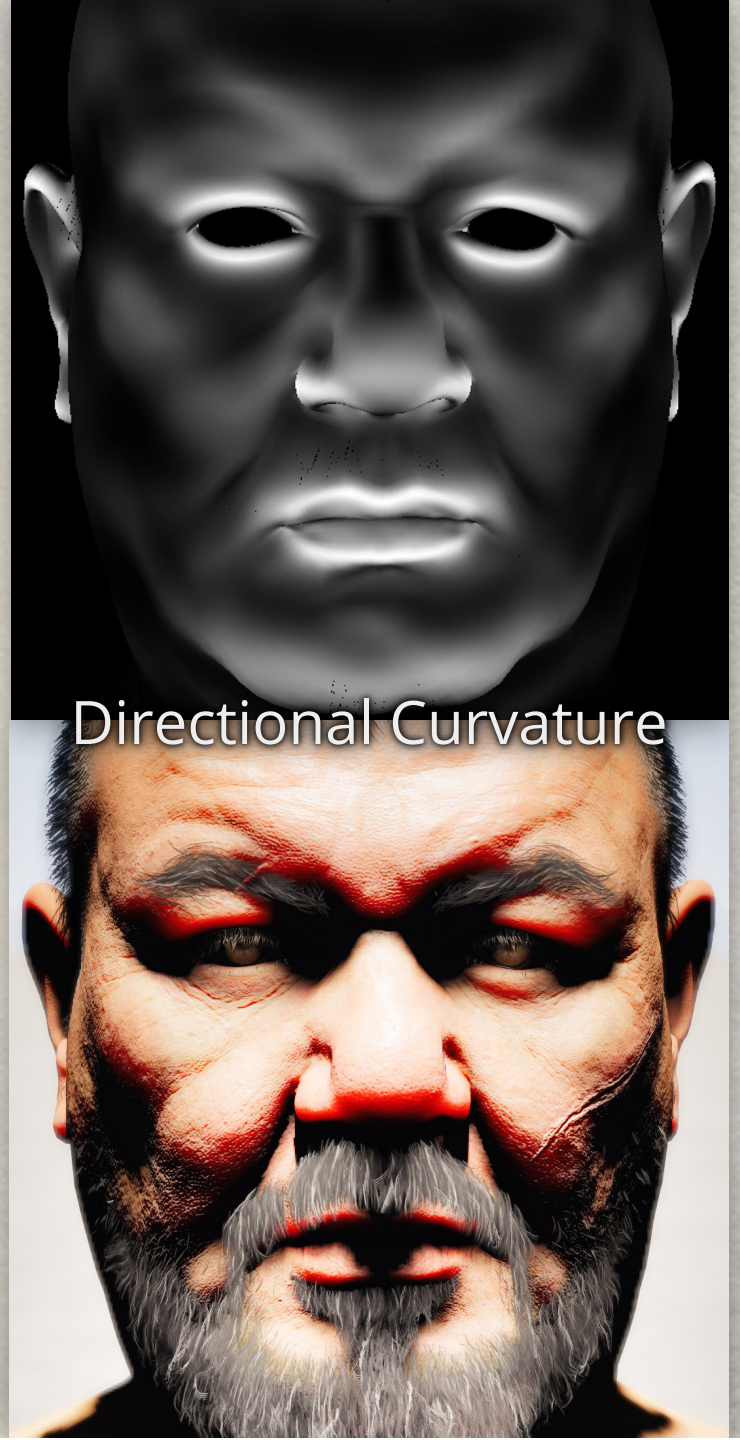
Mean Curvature



Directional Curvature


Zero Curvature


Mean Curvature


Directional Curvature

```
1  float CurvatureFromLight(
2      float3 tangent,
3      float3 bitangent,
4      float3 curvTensor,
5      float3 lightDir)
6  {
7      // Project light vector into tangent plane
8
9      float2 lightDirProj = float2(dot(lightDir, tangent), dot(lightDir, bitangent));
10
11     // NOTE (jasminp) We should normalize lightDirProj here in order to correctly
12     //    calculate curvature in the light direction projected to the tangent plane.
13     //    However, it makes no perceptible difference, since the skin LUT does not vary
14     //    much with curvature when N.L is large.
15
16     float curvature = curvTensor.x * GSquare(lightDirProj.x) +
17                       2.0f * curvTensor.y * lightDirProj.x * lightDirProj.y +
18                       curvTensor.z * GSquare(lightDirProj.y);
19
20     return curvature;
21 }
```

- New way of thinking about pre-integrated skin shading
  - Use linear scattering profile/**cylindrical** integration for punctual light LUT generation
  - Radial profile with **spherical** integration for SH lighting LUT
  - Importance of directional curvature for improved accuracy

Detail Maps

Bonus Slide

Detail Maps

- Goals:
  - Reduce texture memory usage by separating low- and high-frequency content
  - Use library of tileable physically based materials for high frequency
  - Intuitive authoring of low frequency (wear, grime, etc.)
  - Combine in an efficient way at runtime
- For normal maps, *Reoriented Normal Mapping* (Barré-Brisebois and Hill, 2012) works well
- Want something that works well for other maps too (albedo, specular, gloss, AO, etc.)

Bonus Slide

Detail Maps | Traditional Approach

- Traditionally, tiled detail maps are used to add high-frequency detail on top of low-frequency maps
  - Using some kind of blending operator (overlay, additive, etc.)
- No guarantee that the result makes sense
  - Physically
  - Matching desired appearance
  - Needs interactive tuning
- Unclear how to generate from scanned materials
  - Usually needs preprocessing
- Combining two compressed textures
  - Potential for increased artifacts

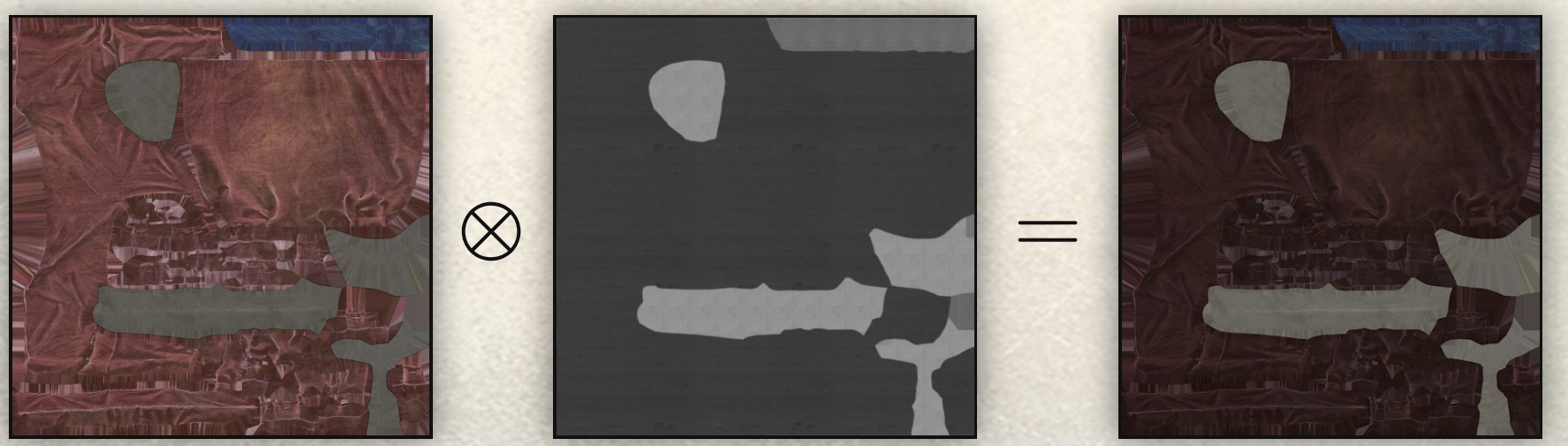
Bonus Slide

Detail Maps | Our Approach

- Start with tiled materials and paint low-frequency content on top
- Synthesize a map such that:

$$\text{Synthesized Map} \otimes \text{Tiled Detail Layers} = \text{Target Map}$$

$$s \otimes d = t$$



Bonus Slide

Detail Maps | Workflow

- Use Substance Designer and Substance Painter for authoring
  - Added SD nodes that match our shader UV and HSV transforms
- Set up tiled materials (up to 3 combined with masks)
- Paint on top to generate a target map
  - This is what artists export for use in our shaders
- At asset compile time:
  - Process target + tiled layers and masks
  - Generate a new synthesized map

Bonus Slide

- What blend operator $\otimes$ should we use?
- Need to ensure that a solution exists for s in

$$s \otimes d = t \quad \forall d, t \in [0, 1]$$

  - s: Synthesized map
  - d: Detail map
  - t: Authored target map
- Excludes hard light, soft light, screen, multiply, etc.
- Overlay works fairly well:

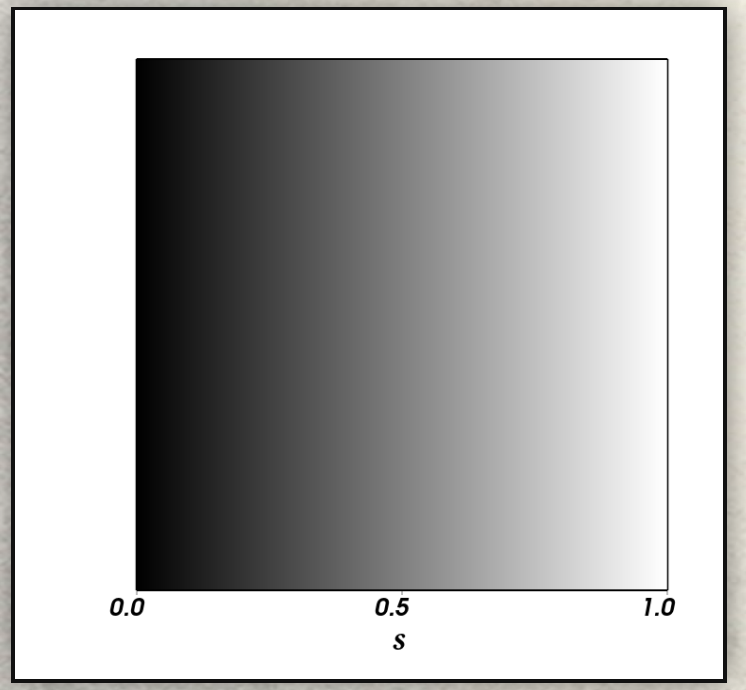
$$f_{overlay}(s, d) = \begin{cases} 2sd, & s < 0.5 \\ 1 - 2(1 - s)(1 - d), & s \geq 0.5 \end{cases}$$

Bonus Slide

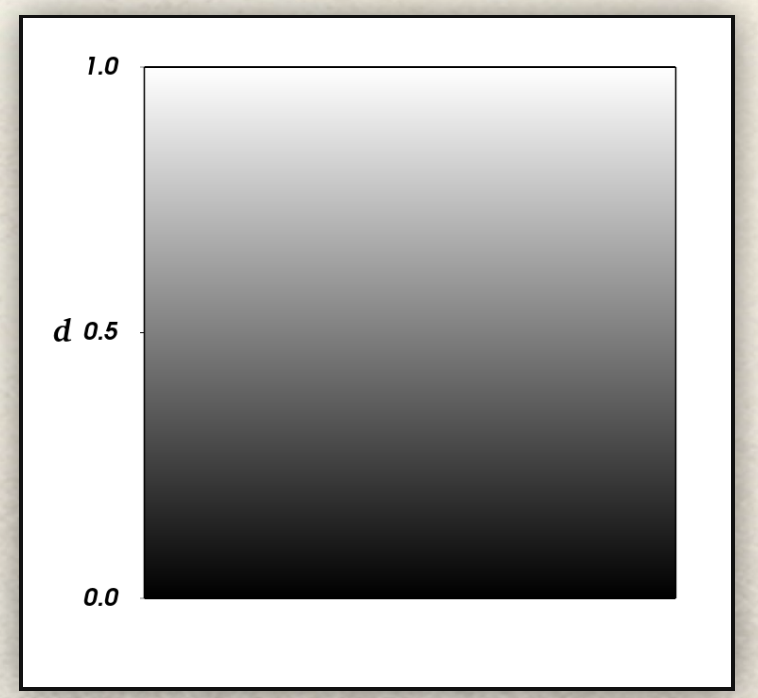
Detail Maps | Blending

$$f_{overlay}(s,d) = \begin{cases} 2sd, & s < 0.5 \\ 1 - 2(1-s)(1-d), & s \geq 0.5 \end{cases}$$

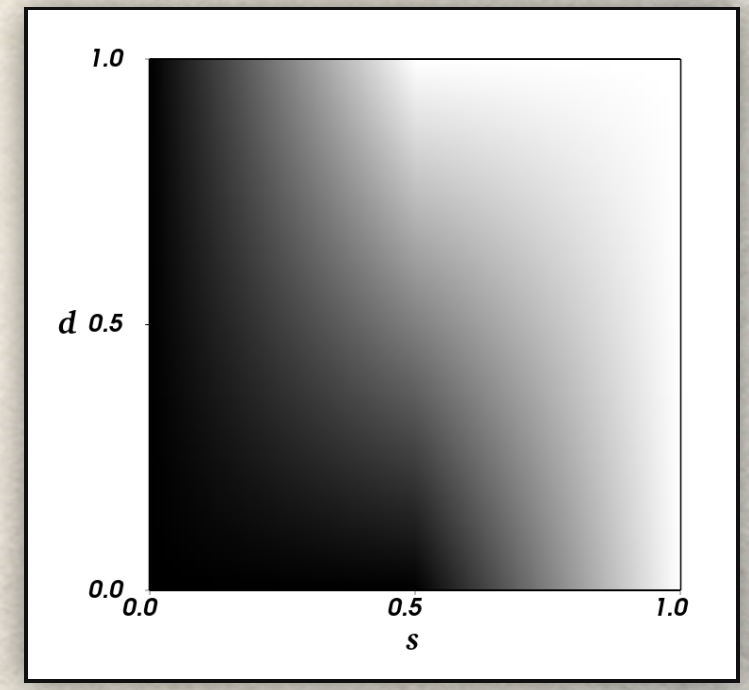
Synthesized



Detail

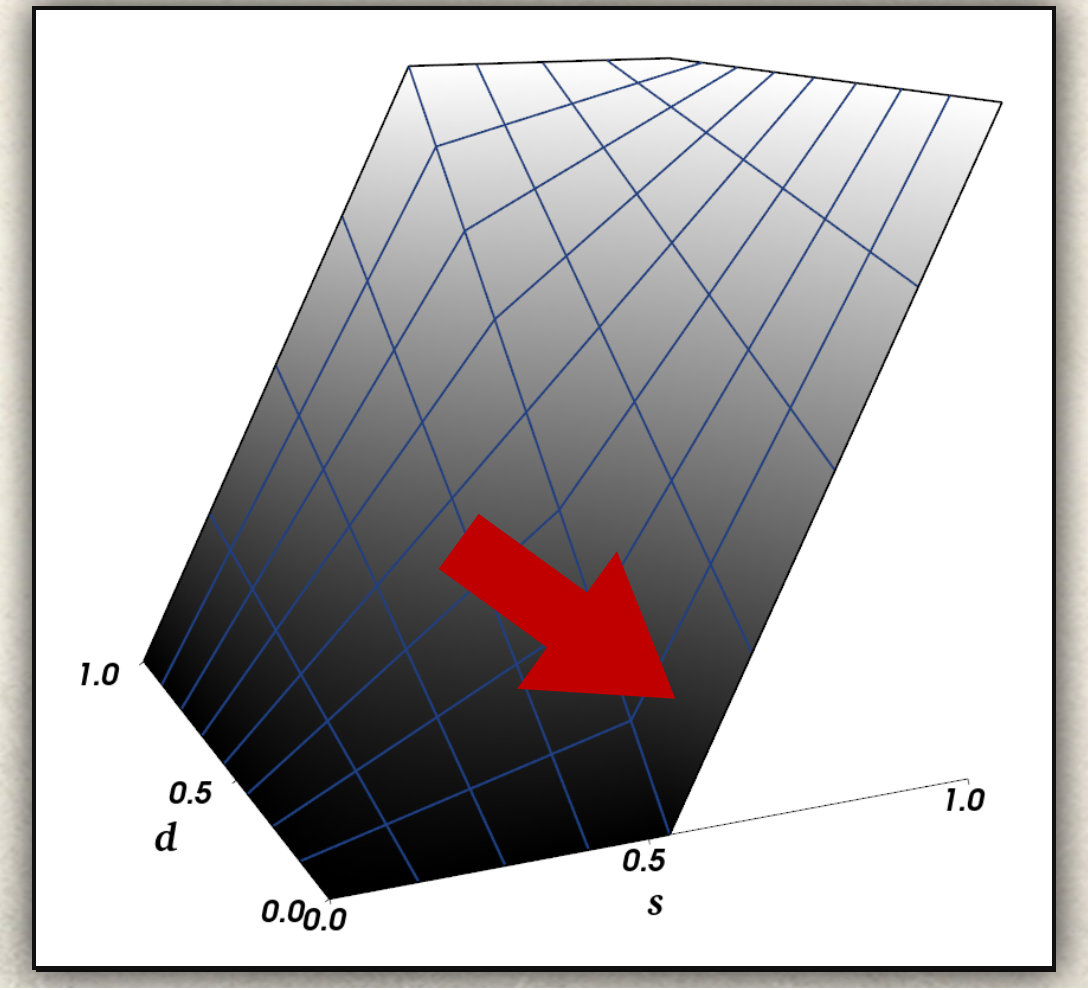


Target



Bonus Slide

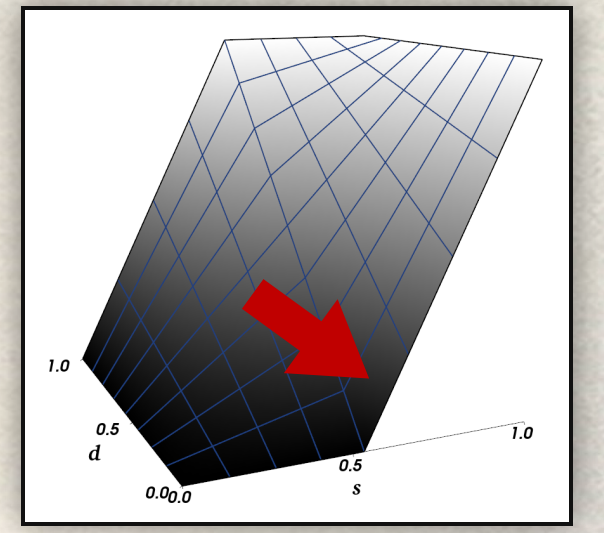
$$f_{overlay}(s,d) = \begin{cases} 2sd, & s < 0.5 \\ 1 - 2(1-s)(1-d), & s \geq 0.5 \end{cases}$$



Bonus Slide

Detail Maps | Precision

- Precision gets worse as detail d approaches 0 and 1
  - Relative precision especially bad as d approaches 0
- Noticeable quality degradation when using BC1
- Wanted to increase precision when target t is near d
  - Especially when brightening dark tiled layers
  - Led us to modify the blending equation:

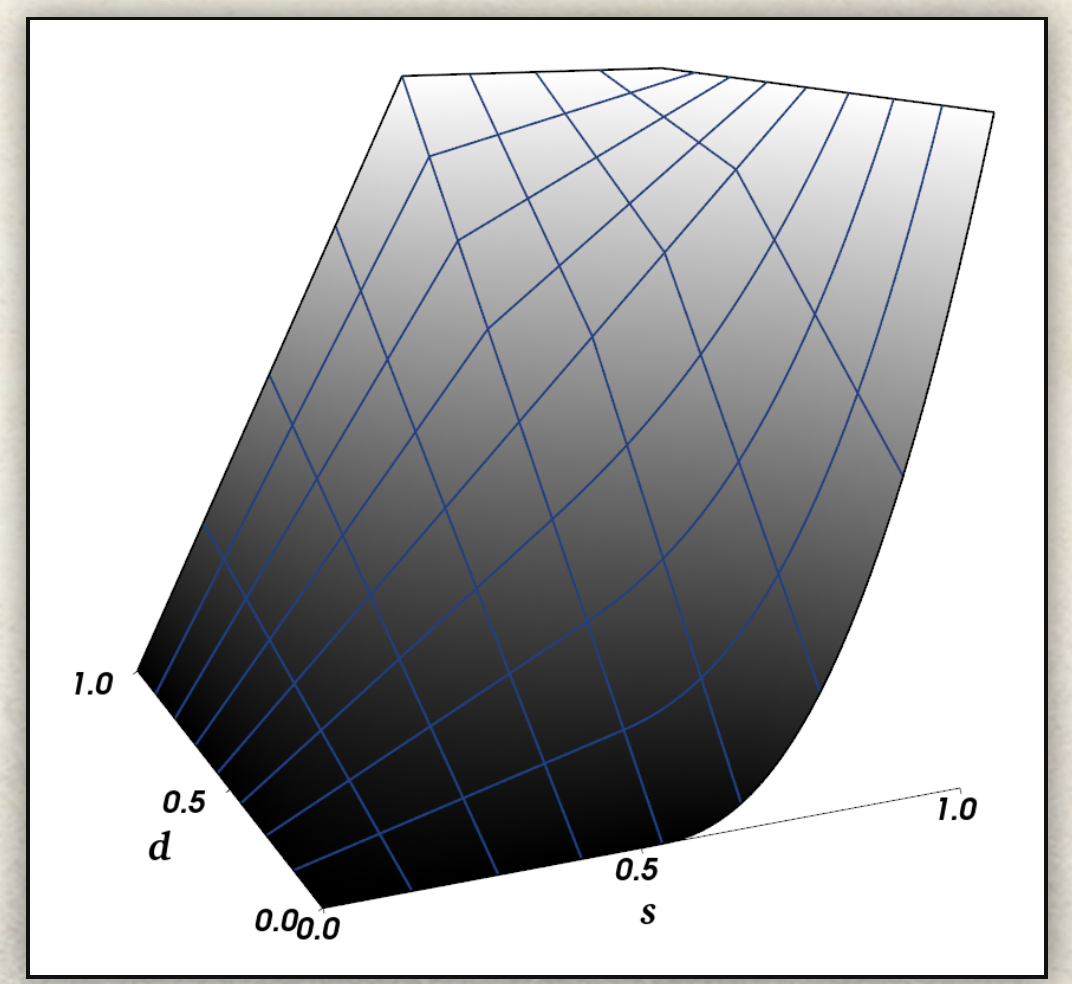


$$f_{os}(s,d) = \begin{cases} 2sd, & s < 0.5 \\ 1 - 2(1-s)(1-d), & s \geq 0.5 \text{ and } d \geq 0.5 \\ \text{lerp}\left(2sd, s, (2s-1)^2\right), & s \geq 0.5 \text{ and } d < 0.5 \end{cases}$$

Bonus Slide

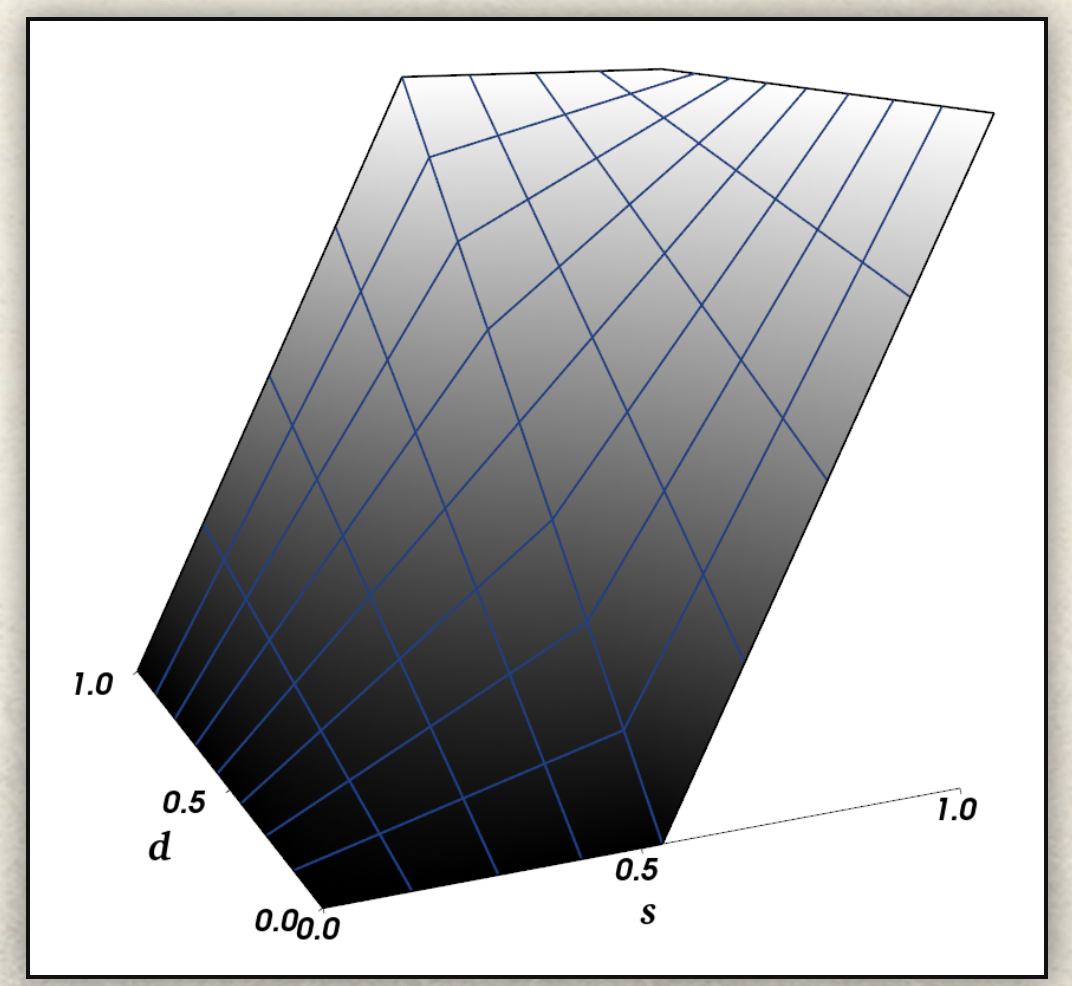
Detail Maps | Precision

$$f_{os}(s,d) = \begin{cases} 2sd, & s < 0.5 \\ 1 - 2(1-s)(1-d), & s \geq 0.5 \text{ and } d \geq 0.5 \\ \text{lerp}\left(2sd, s, (2s-1)^2\right), & s \geq 0.5 \text{ and } d < 0.5 \end{cases}$$



Bonus Slide

$$f_{overlay}(s,d) = \begin{cases} 2sd, & s < 0.5 \\ 1 - 2(1-s)(1-d), & s \geq 0.5 \end{cases}$$



Bonus Slide

Detail Maps | Precision

Bonus Slide

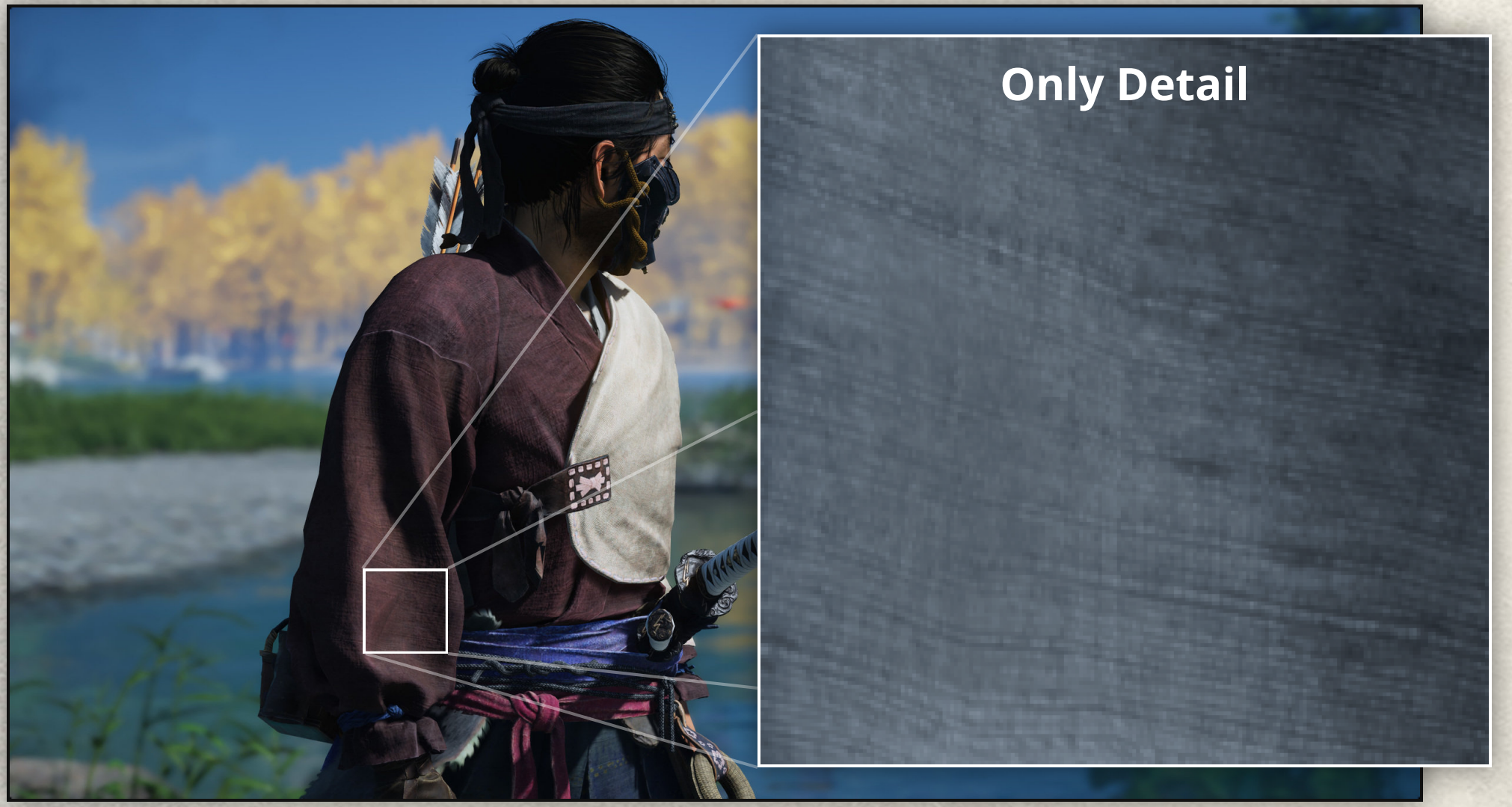
Detail Maps | Precision

Bonus Slide

Detail Maps | Precision

Bonus Slide

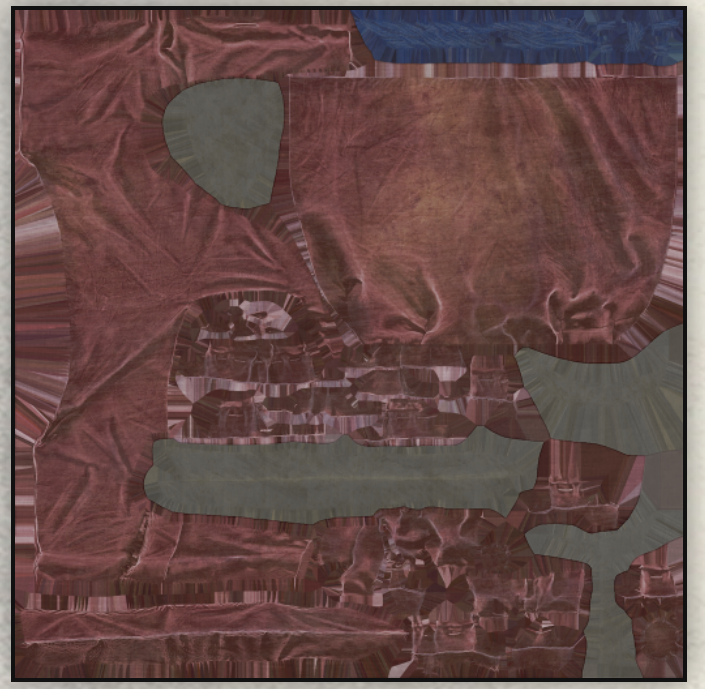
Detail Maps | Precision



Bonus Slide

Detail Maps | Precision

Synthesized



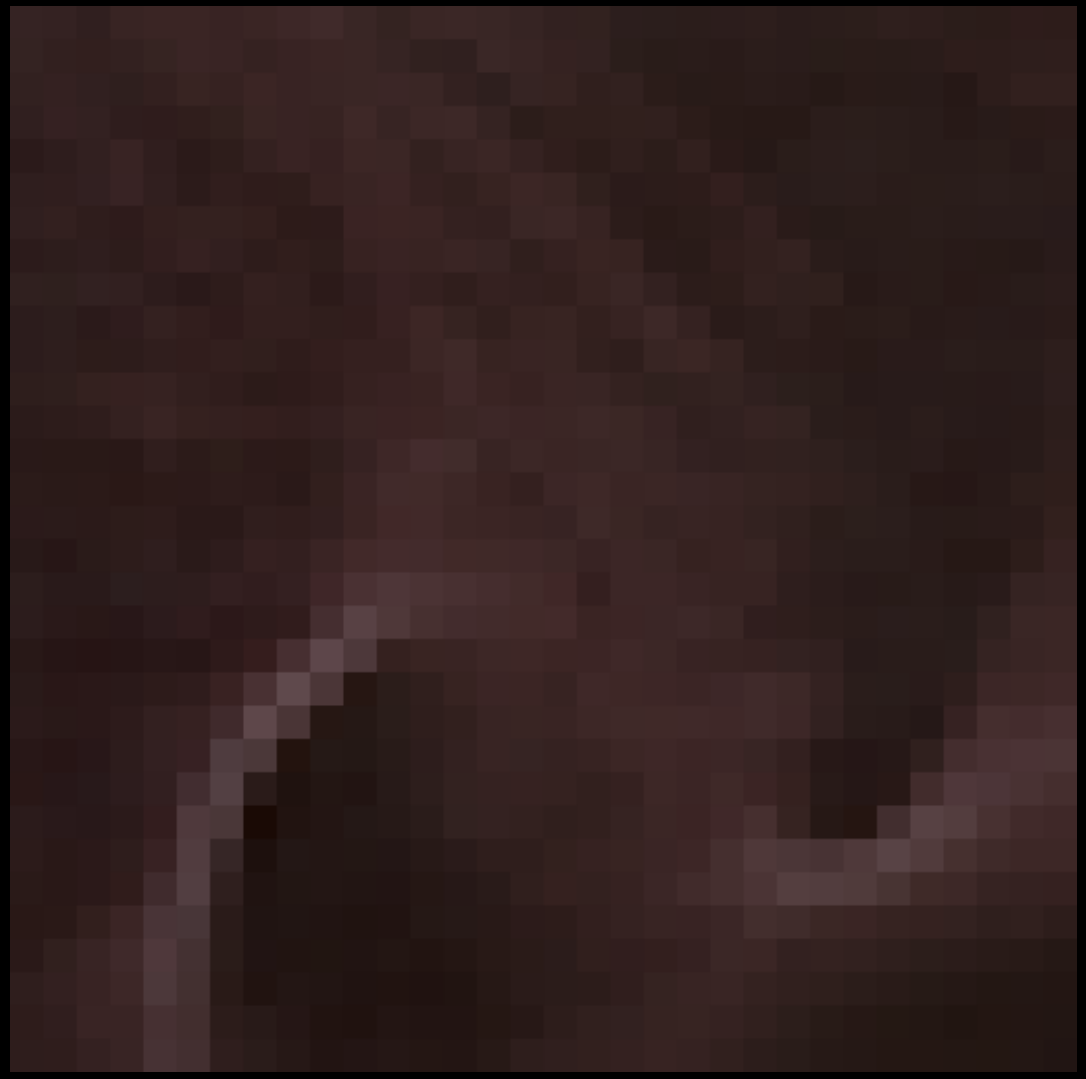
Detail

Target

Bonus Slide

Detail Maps| Precision

Target

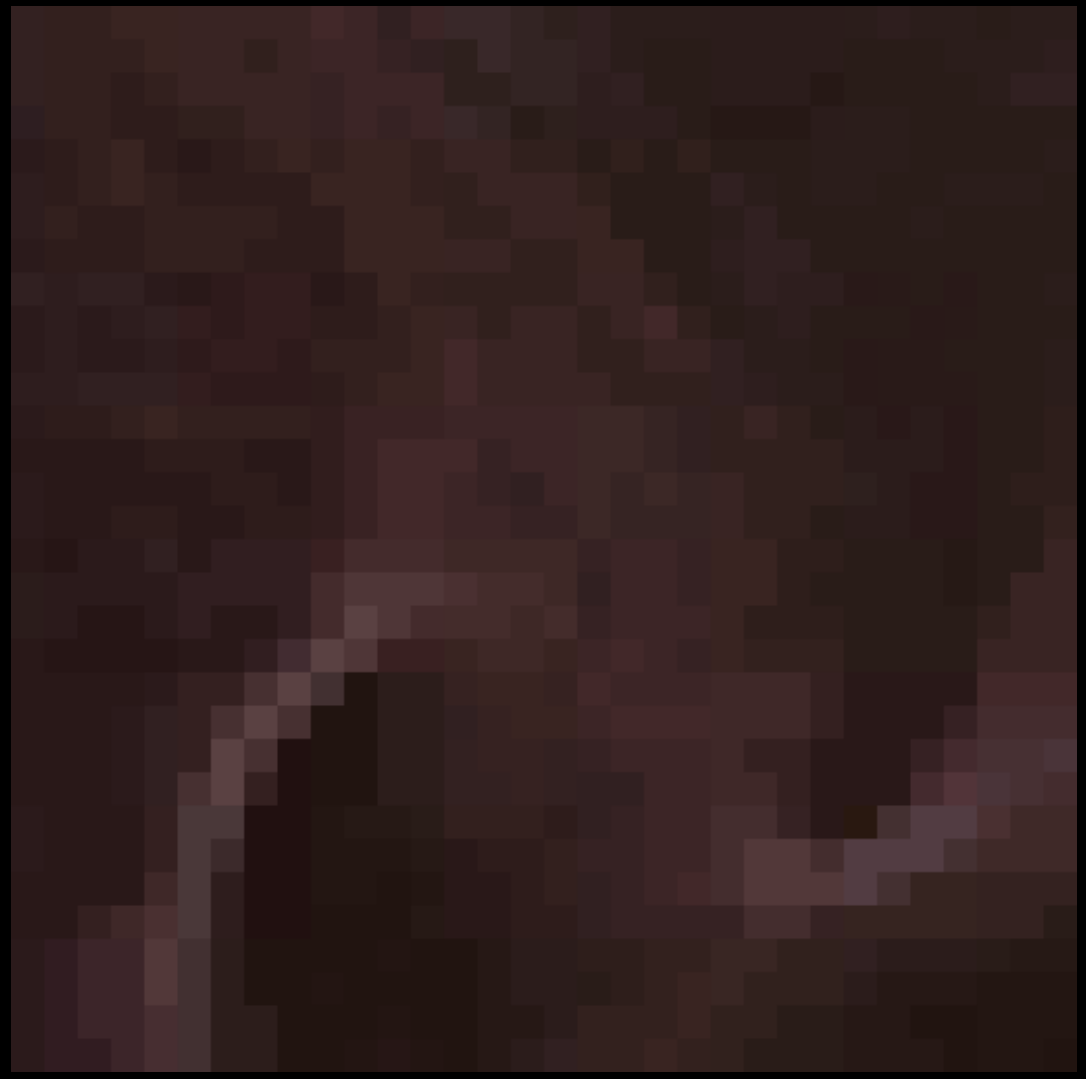


Bonus Slide

Detail Maps| Precision

Target BC1

RMSE: 1.95

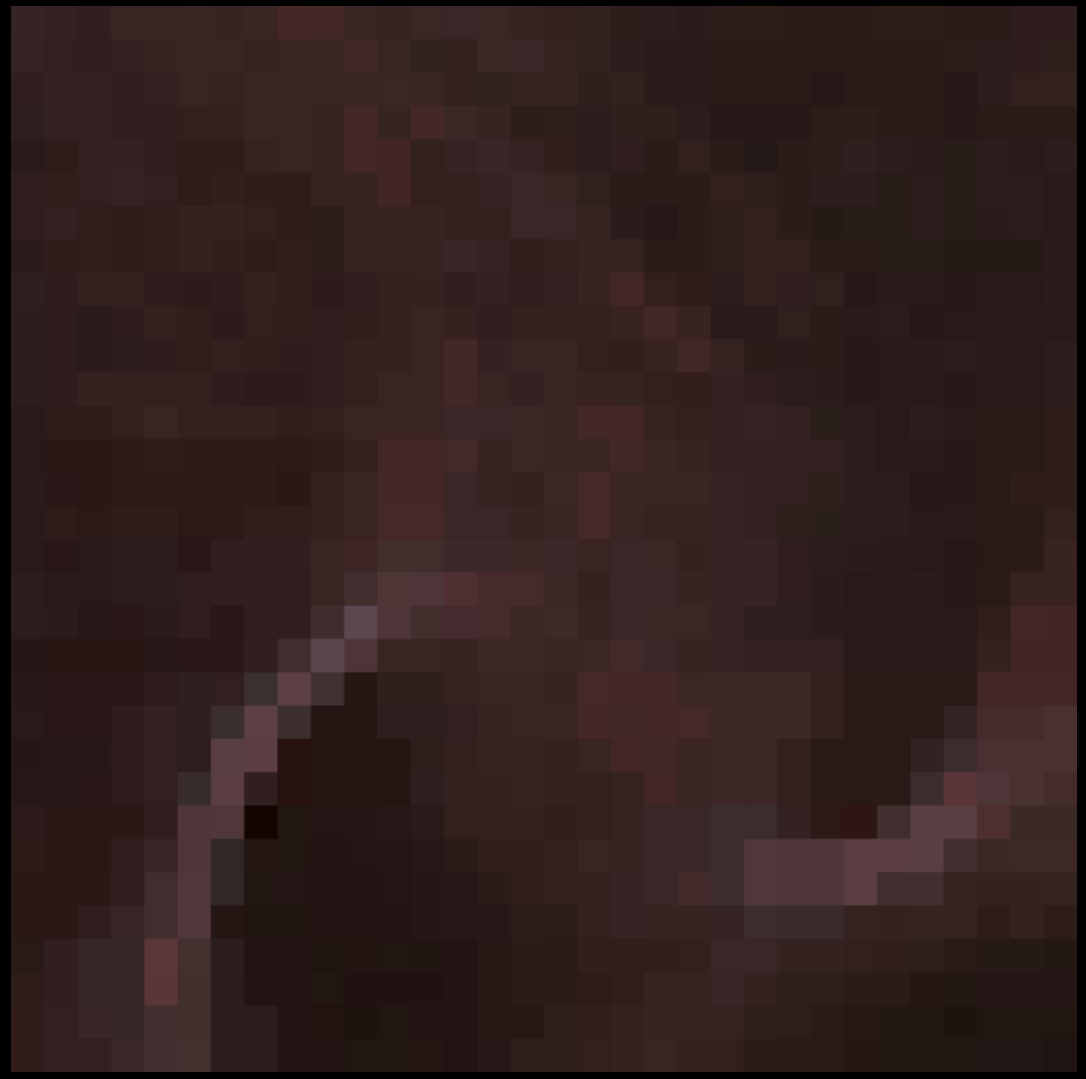


Bonus Slide

Detail Maps| Precision

Standard Overlay Blend

RMSE: 1.74

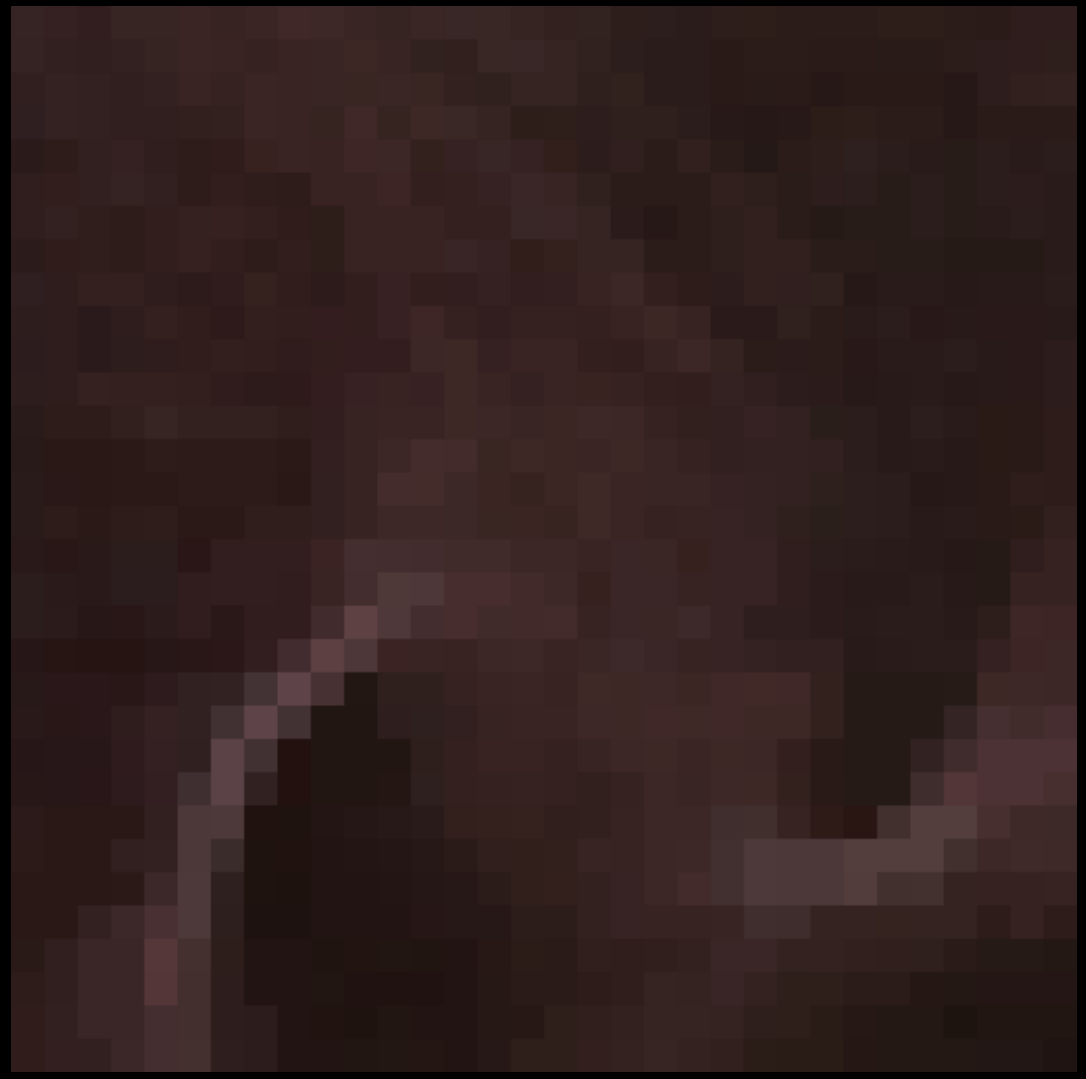


Bonus Slide

Detail Maps| Precision

Smooth Overlay Blend (Ours)

RMSE: 1.60

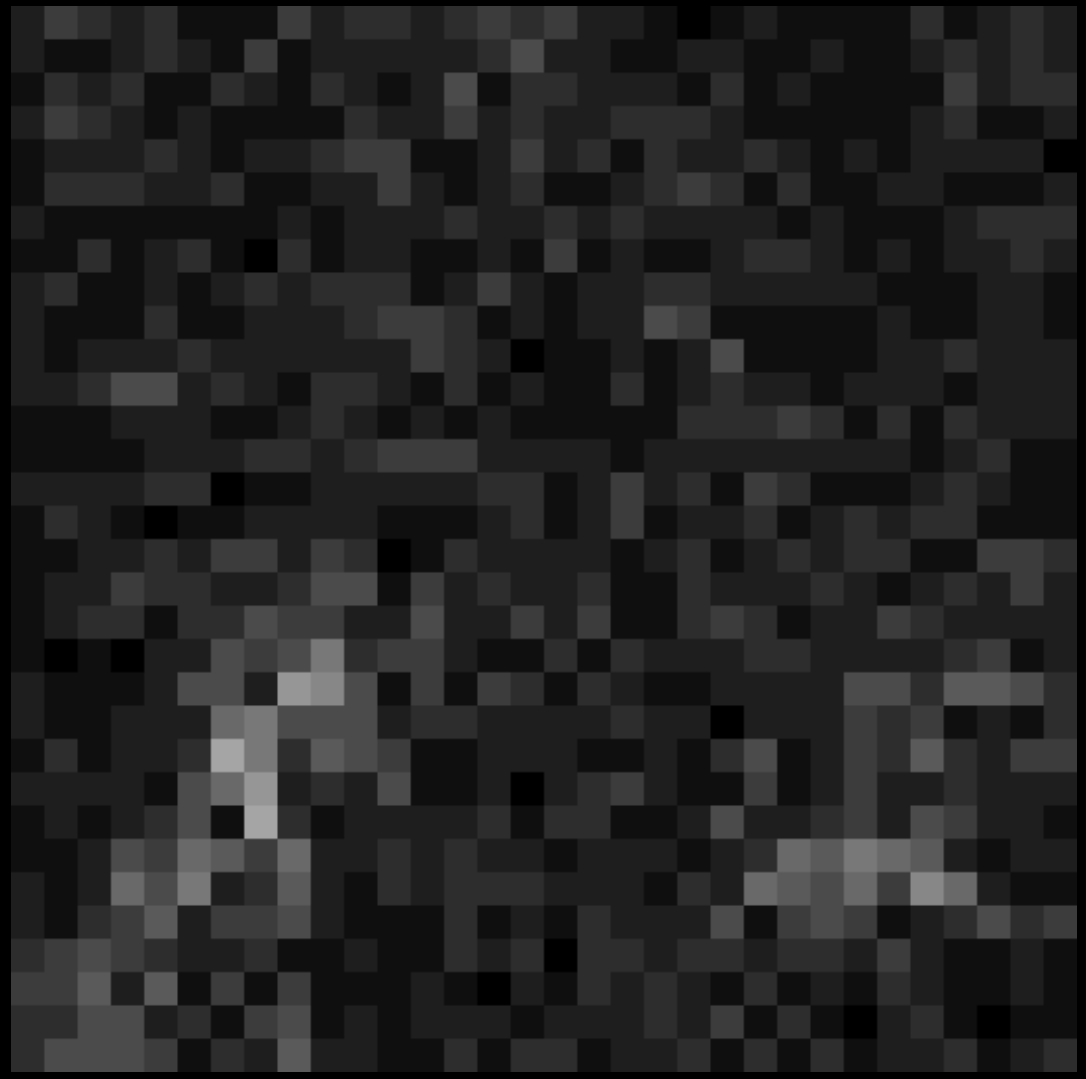


Bonus Slide

Detail Maps| Precision

Target BC1 Error

RMSE: 1.95

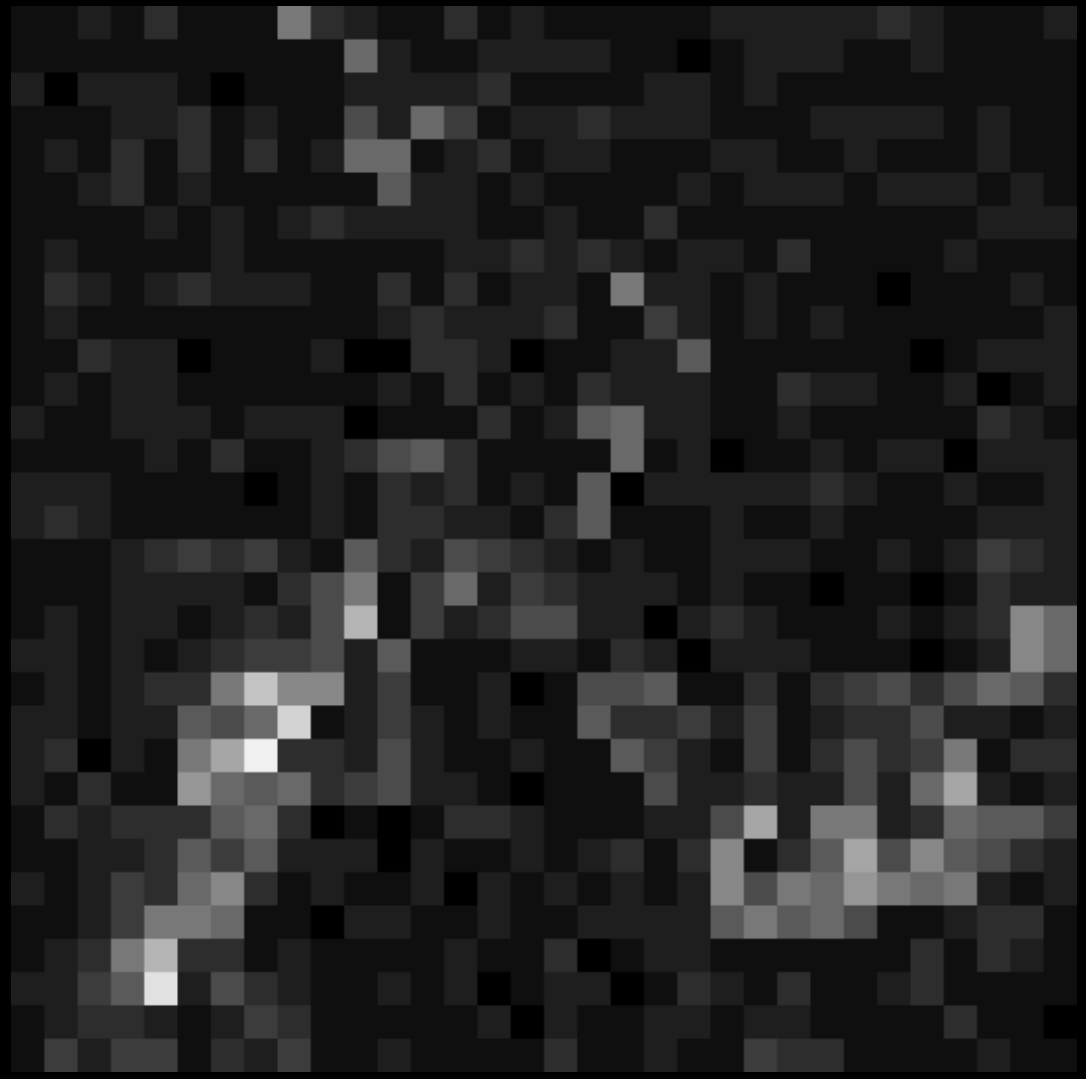


Bonus Slide

Detail Maps| Precision

Standard Overlay Blend Error

RMSE: 1.74

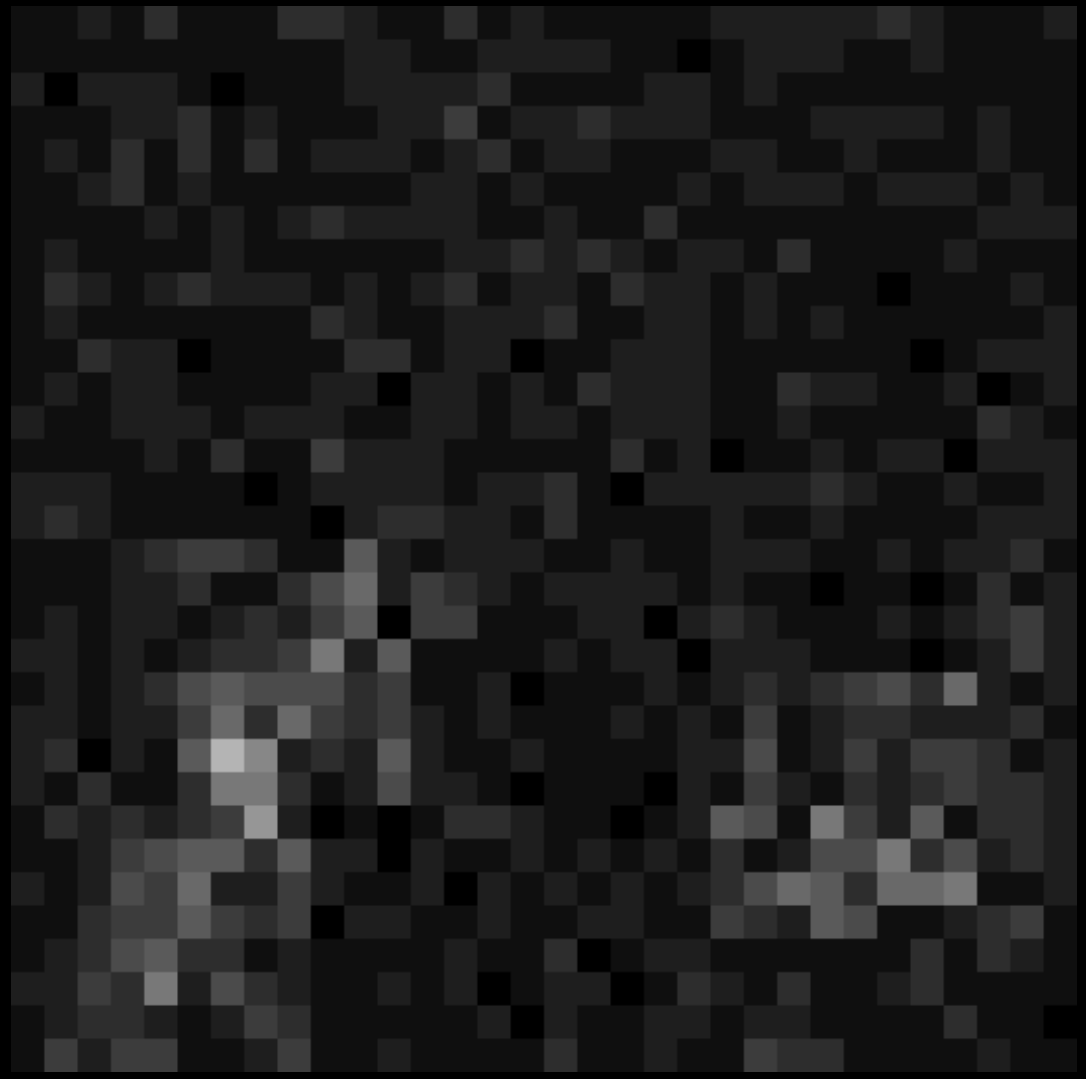


Bonus Slide

Detail Maps| Precision

Smooth Overlay Blend Error (Ours)

RMSE: 1.60



Bonus Slide

Detail Maps | Compression

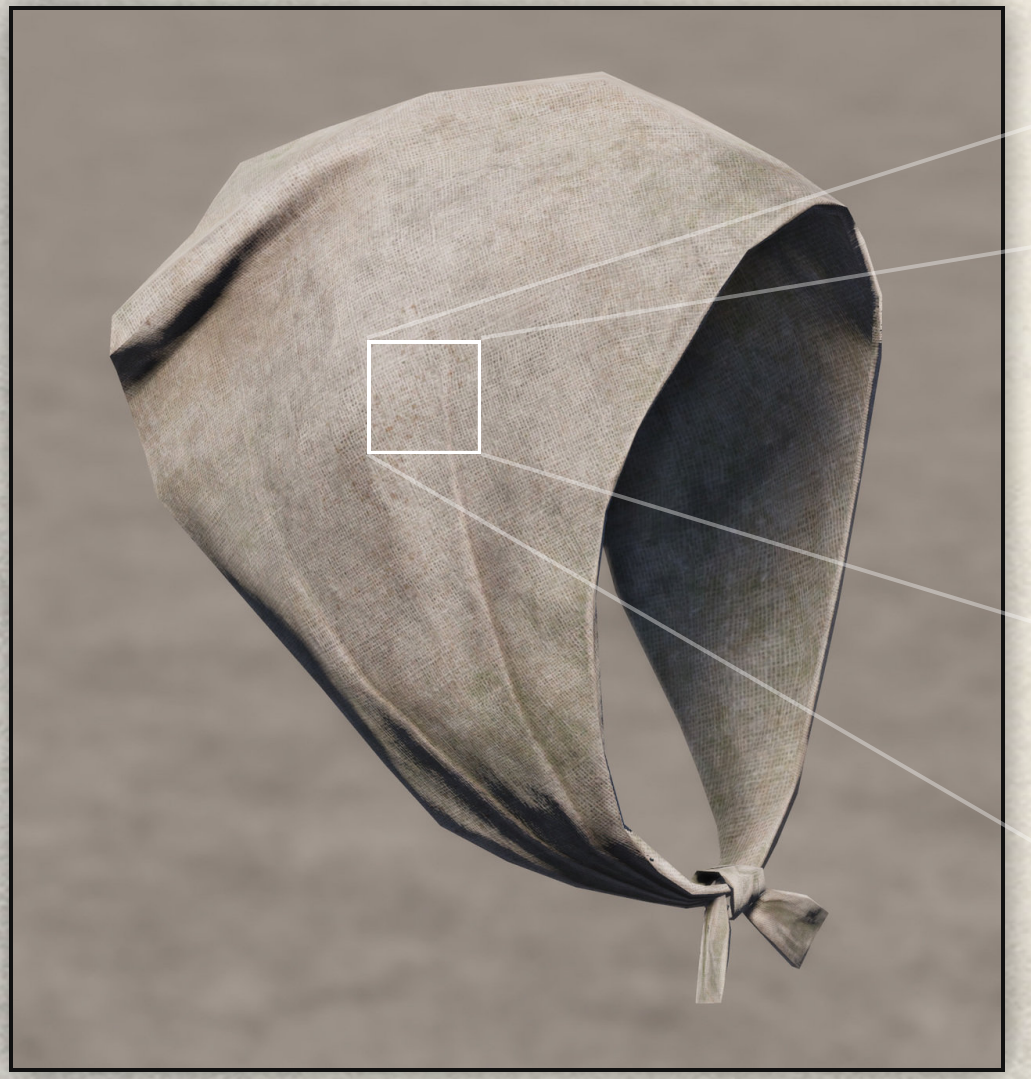
- Blending two sets of compressed textures
  - Exacerbates compression artifacts
- Reduce by synthesizing texture against *compressed* tiled maps
  - Except for top mip which is magnified against multiple detail mips

Bonus Slide

Detail Maps | Compression

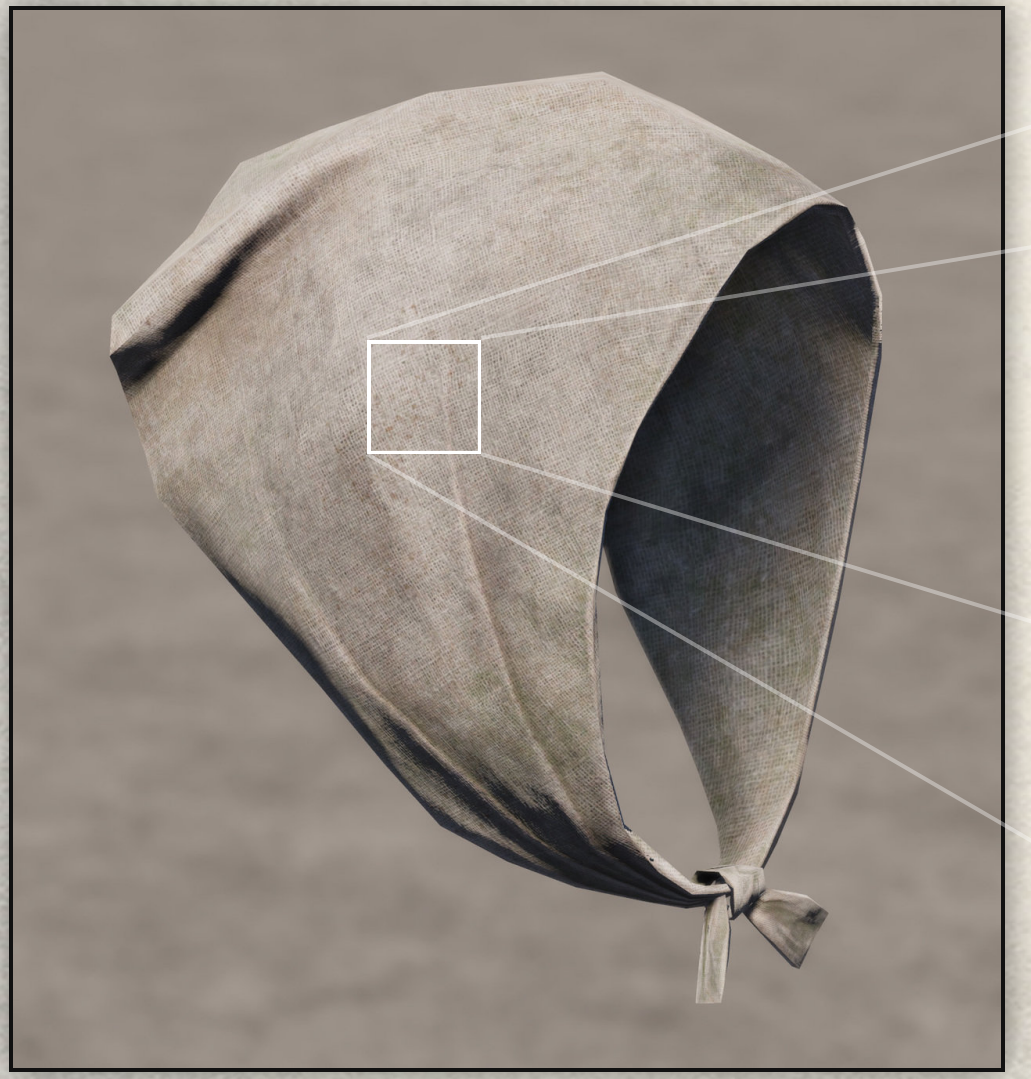
Bonus Slide

Detail Maps | Compression



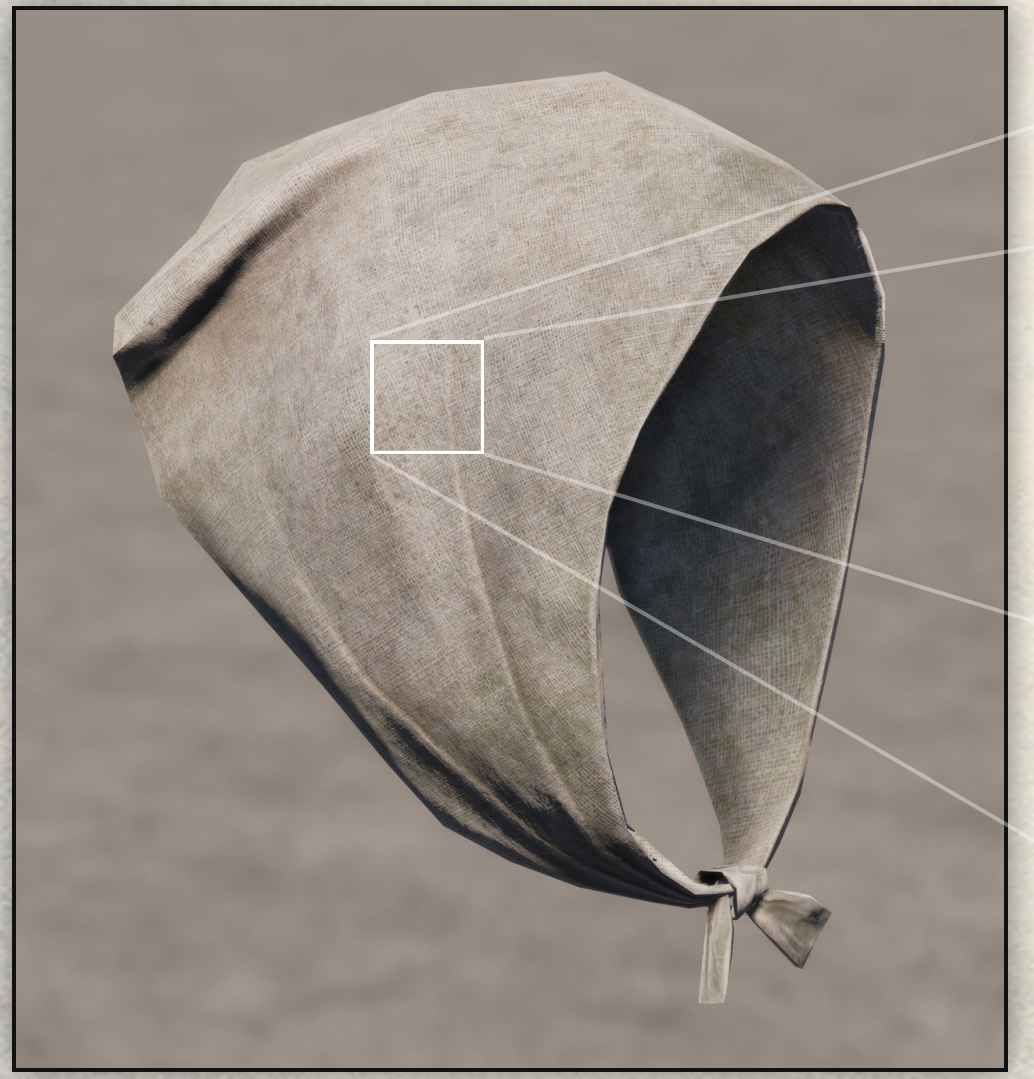
Bonus Slide

Detail Maps | Compression



Bonus Slide

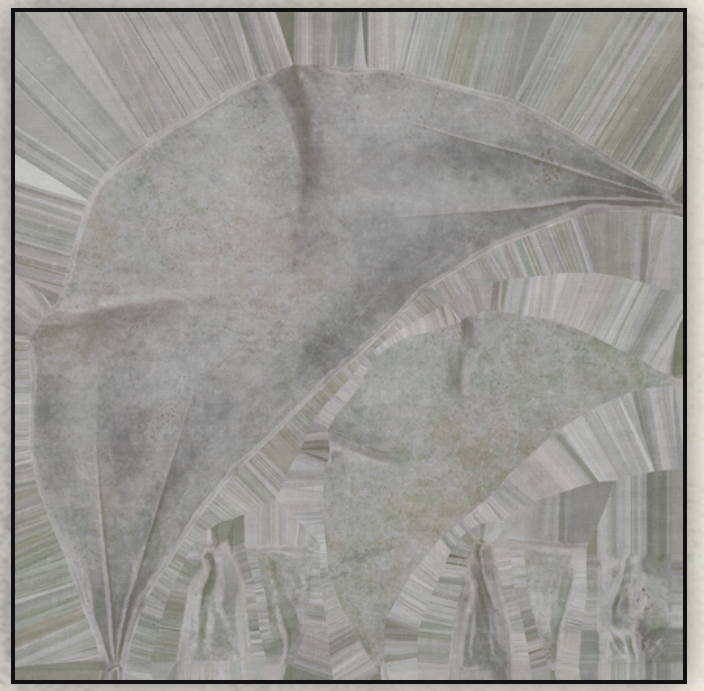
Detail Maps | Compression



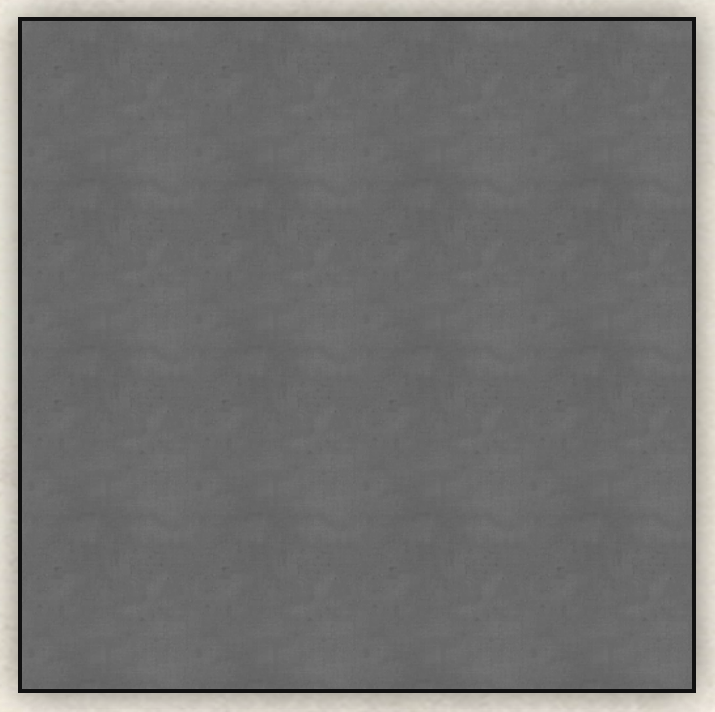
Bonus Slide

Detail Maps | Compression

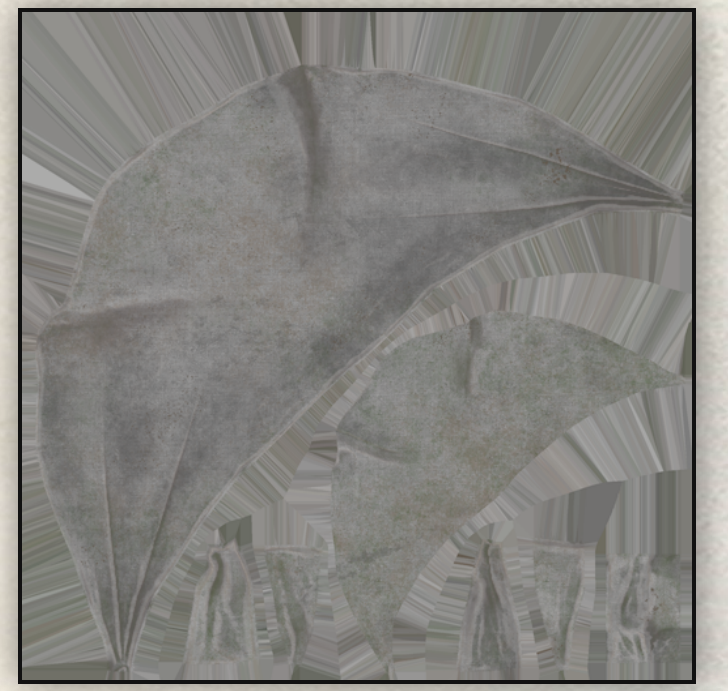
Synthesized



Detail



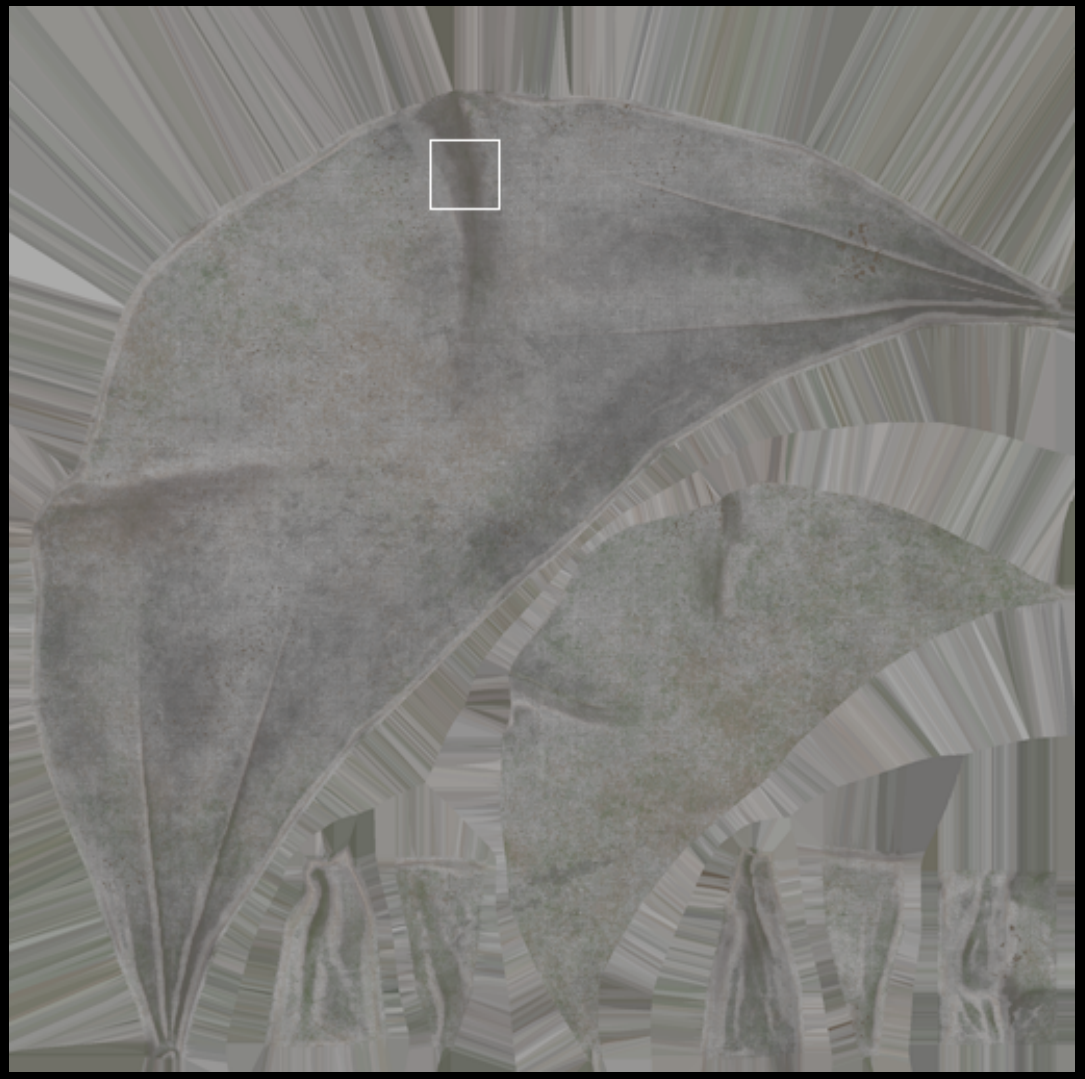
Target

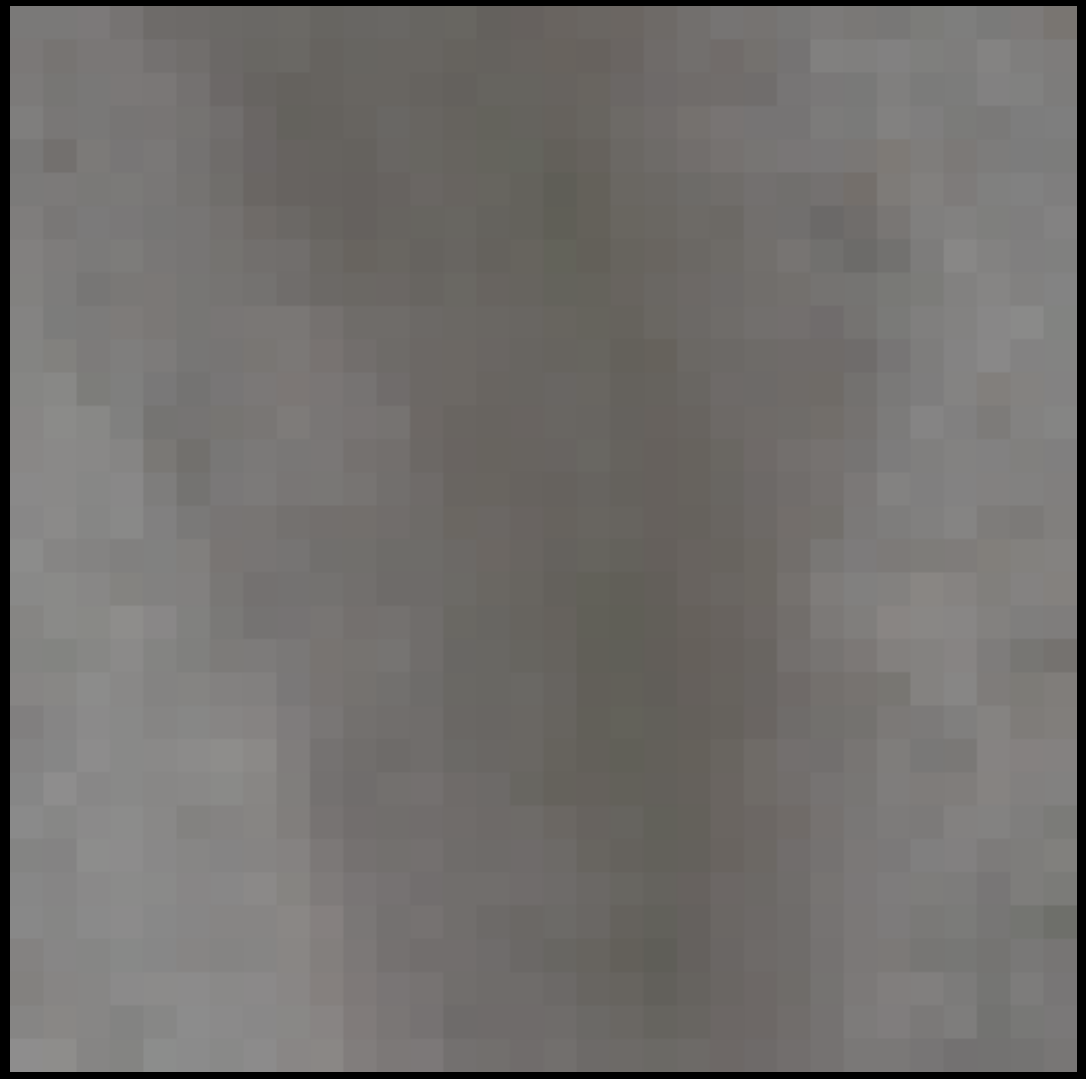


Bonus Slide

Detail Maps| Compression

Target

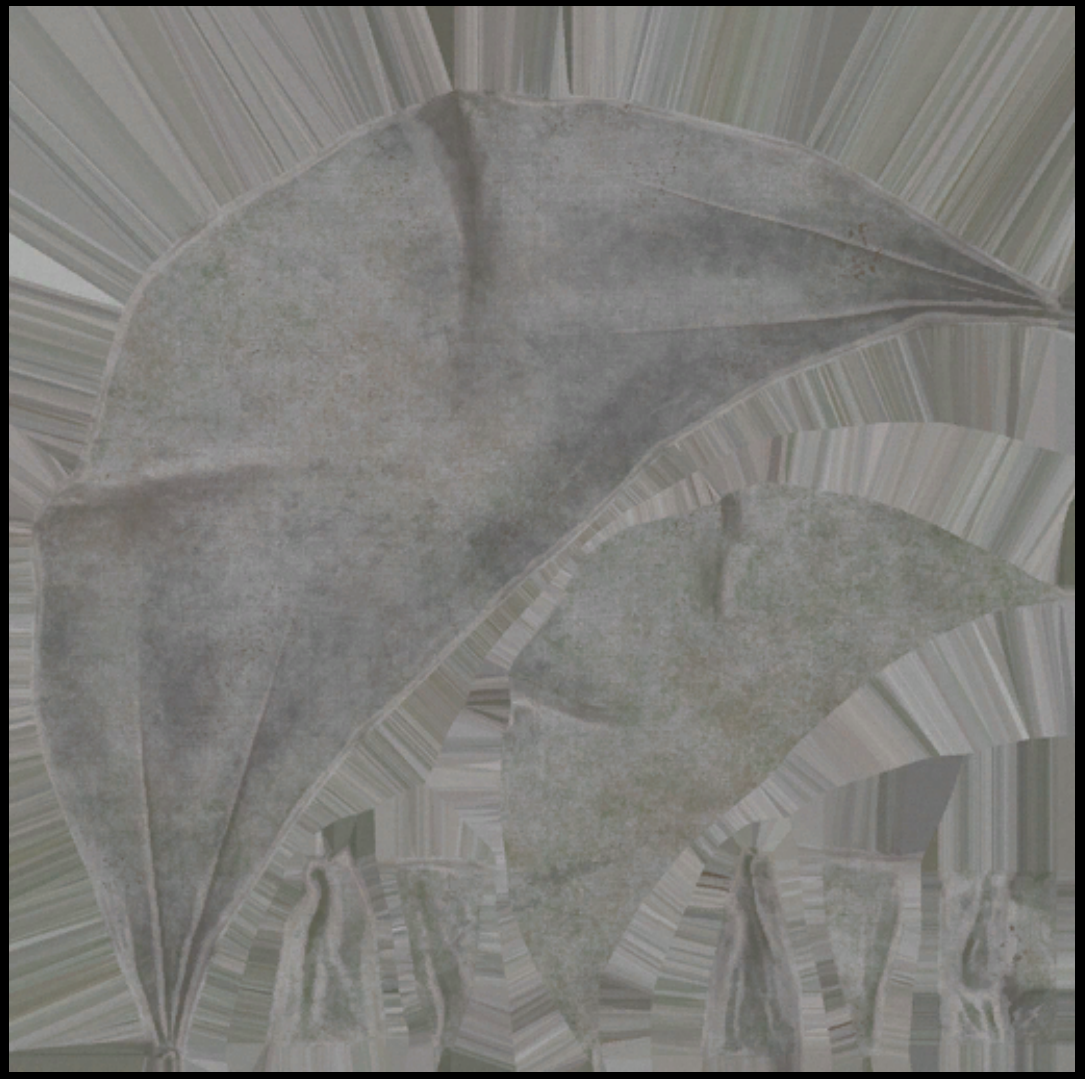




Bonus Slide

Detail Maps| Compression

Blend without Compression Compensation RMSE: 2.43



Bonus Slide

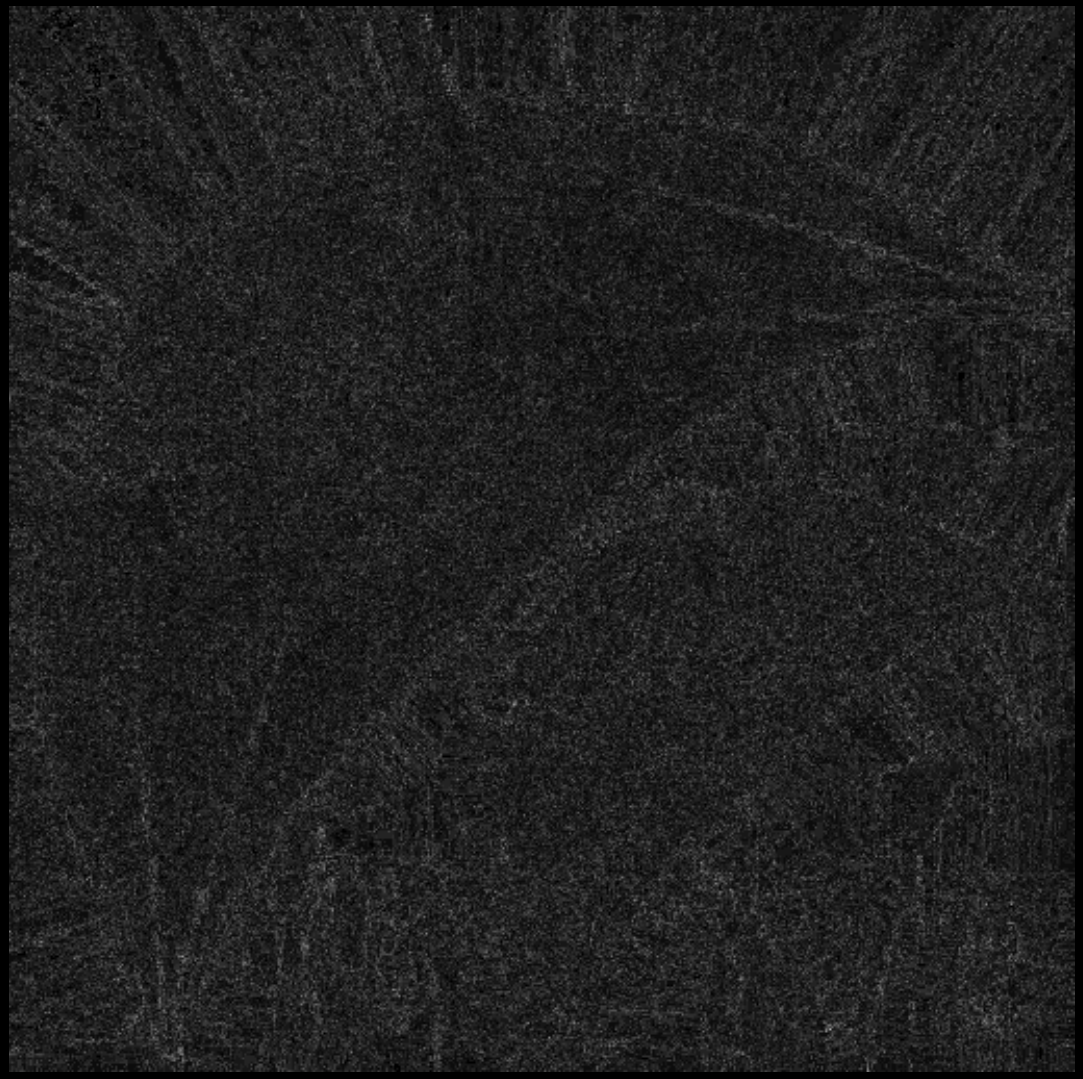
Detail Maps| Compression

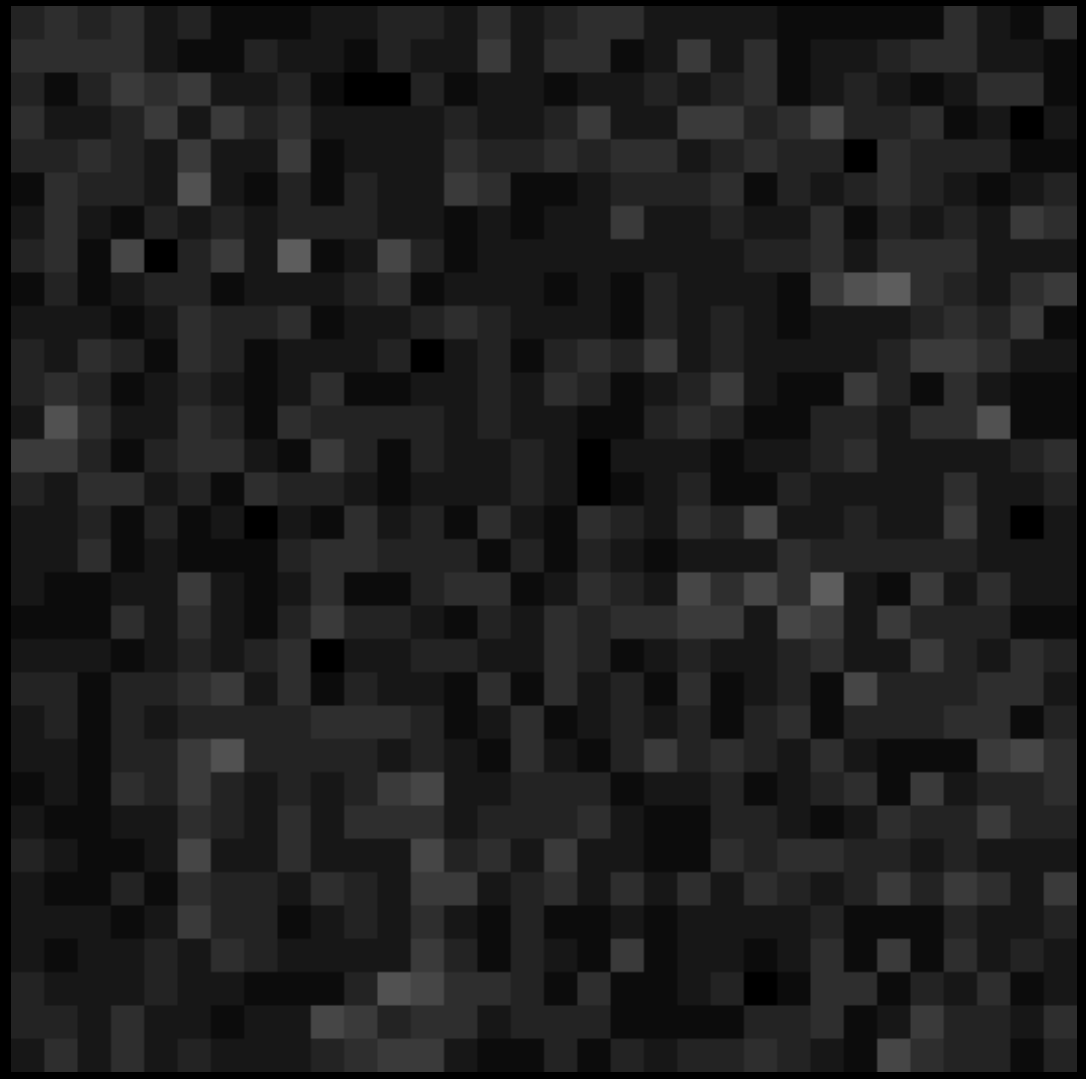
Blend with Compression Compensation (Ours) RMSE: 2.00

Bonus Slide

Detail Maps| Compression

Error without Compression Compensation RMSE: 2.43

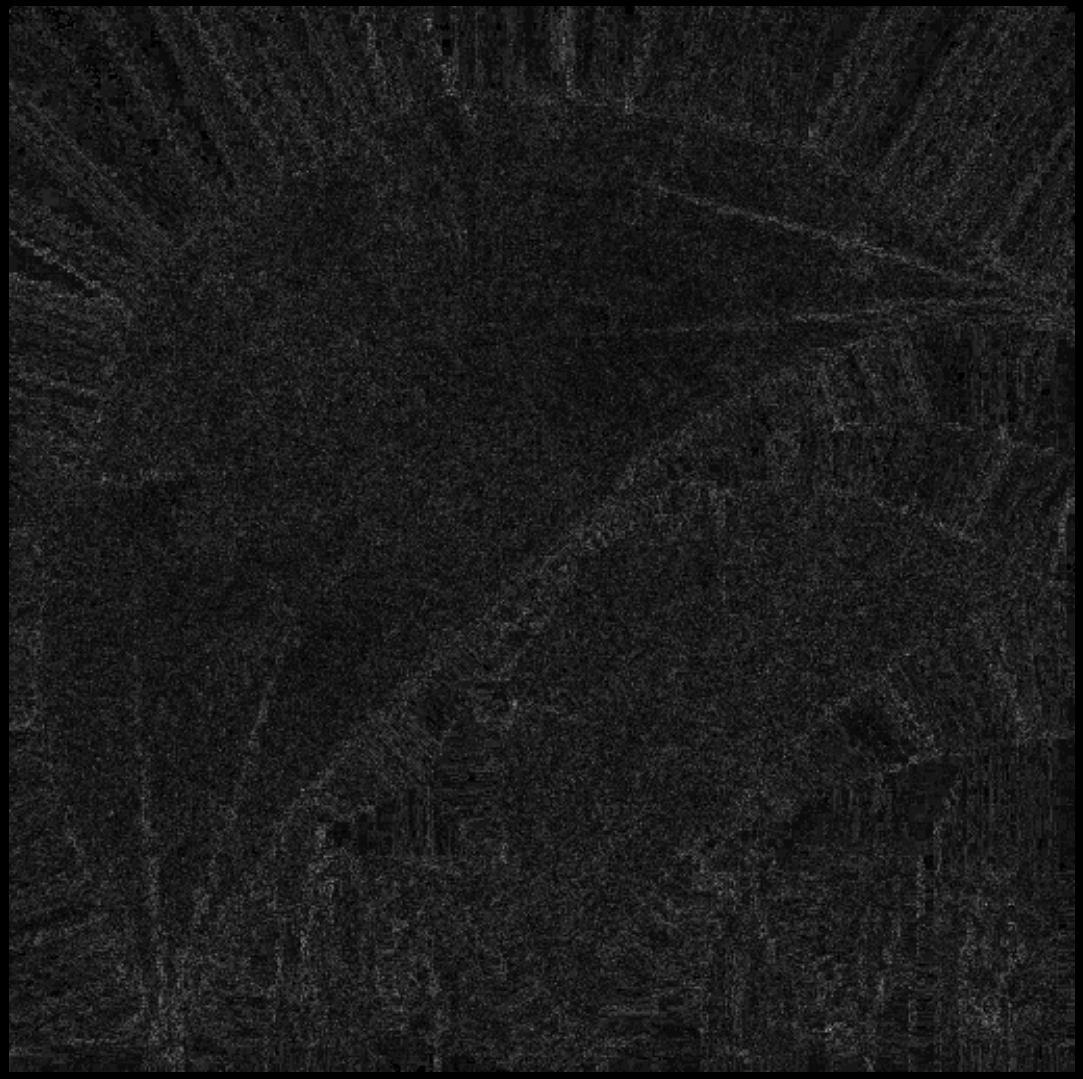


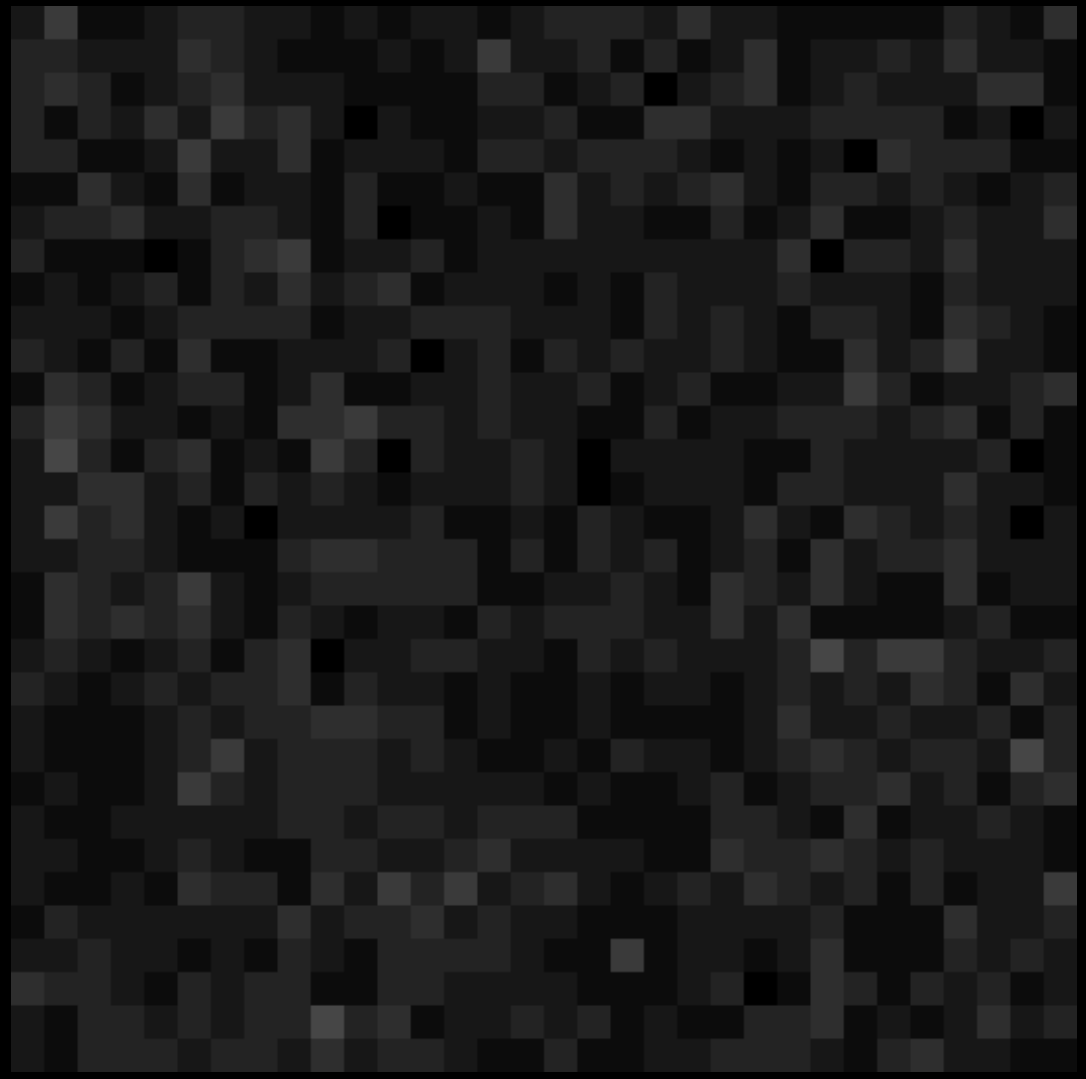


Bonus Slide

Detail Maps| Compression

Error with Compression Compensation (Ours) RMSE: 2.00





Bonus Slide

Detail Maps | Statistics

- Analyzed second mipmap of 1743 material color textures in game
- Average compressed target RMSE: **5.67**
- Average detail-blended RMSE: **5.81**
  - **0.14 (2.5%) higher** than with no detail
- Average RMSE using standard overlay blend: **5.89**
  - **0.09 (1.5%) higher** than ours (ΔRMSE 61% higher)
- Average RMSE without compression compensation: **5.87**
  - **0.06 (1.1%) higher** than ours (ΔRMSE 44% higher)
  - Recall that first mipmap doesn't use compression compensation

Bonus Slide

Detail Maps | Other Blend Modes

- Normal maps blended using Reoriented Normal Mapping (Barré-Brisebois and Hill, 2012)
  - https://blog.selfshadow.com/publications/blending-in-detail/
  - Compute like other maps (by inverting the blend function)
- Anisotropic specular (“aniso”) maps blended with regular alpha blending
  - I.e., “over” operator
  - Alpha is a shader parameter

Bonus Slide

Detail Maps | Summary

- Successful widespread adoption of detail maps for Ghost
- Advantages over traditional approach:
  - Can selectively cancel out lower frequency components of detail
  - More potential texture reuse since detail can be used as material
- Disadvantages:
  - Slightly longer texture processing
  - Slightly higher RMSE on average for color textures (vs no detail)
    - Can be reduced by centering histogram of detail maps
      - I.e., same type of preprocessing used with traditional detail maps

Bonus Slide

Acknowledgements

- My wife and son for their patience while I went from finishing Ghost to working on this presentation
- Sucker Punch rendering programmers:
  - Adrian Bentley
  - Bill Rockenbeck
  - Eric Wohllaib (who implemented deferred fuzziness)
  - Matthew Pohlmann
  - Tom Low
- Drew Harrison, Harold Lamb, Joanna Wang, Omar Aweidah, and Phillip Jenne for their assistance with assets for this talk
- Everybody at Sucker Punch for making everything awesome
- Steve McAuley and Stephen Hill for the insightful feedback

References

[DHI+13] Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. Linear efficient antialiased displacement and reflectance mapping. *ACM Trans. Graph.*, 32(6), November 2013.

[HDCD15] Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. The SGGX microflake distribution. *ACM Trans. Graph.*, 34(4), July 2015.

[KK89] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph.*, 23(3):271–280, July 1989.

[KP03] Jan Koenderink and Sylvia Pont. The secret of velvety skin. *Machine Vision and Applications*, 14(4):260–268, 2003.

[McA13] Steve McAuley. Extension to energy-conserving wrapped diffuse. http://blog.stevemcauley.com/2013/01/30/extension-to-energy-conserving-wrapped-diffuse/, January 2013.

[MHH+12] Stephen McAuley, Stephen Hill, Naty Hoffman, Yoshiharu Gotanda, Brian Smits, Brent Burley, and Adam Martinez. Practical physically-based shading in film and game production. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, New York, NY, USA, 2012. Association for Computing Machinery.

[OB10] Marc Olano and Dan Baker. Lean mapping. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, page 181–188, New York, NY, USA, 2010. Association for Computing Machinery.

[PB11] Eric Penner and George Borshukov. Pre-integrated skin shading. In Wolfgang Engel, editor, *GPU Pro 2*, chapter 1. A K Peters/CRC Press, New York, 1st edition, 2011.

[PH10] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*, pages 583–587. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[RH01a] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 497–500, New York, NY, USA, 2001. ACM.

[RH01b] Ravi Ramamoorthi and Pat Hanrahan. On the relationship between Radiance and Irradiance: Determining the illumination from images of a convex Lambertian object. *Journal of the Optical Society of America*, 18(10):2448–2459, Oct 2001.

[Rus04] Szymon Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission*, September 2004.

[RV11] P. Ramachandran and G. Varoquaux. Mayavi: 3D Visualization of Scientific Data. *Computing in Science & Engineering*, 13(2):40–51, 2011.


Thank You!