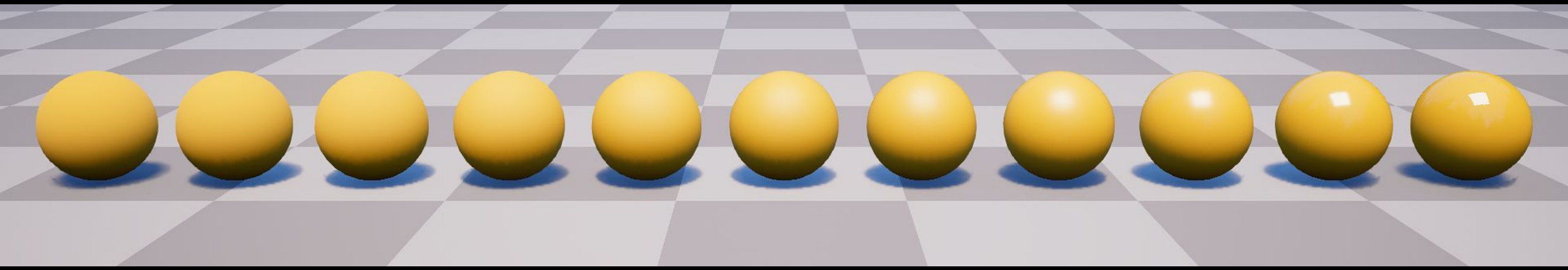
PHOTOGRAPHY & RECORDING PROHIBITED



A Journey Through Implementing Multiscattering BRDFs & Area Lights

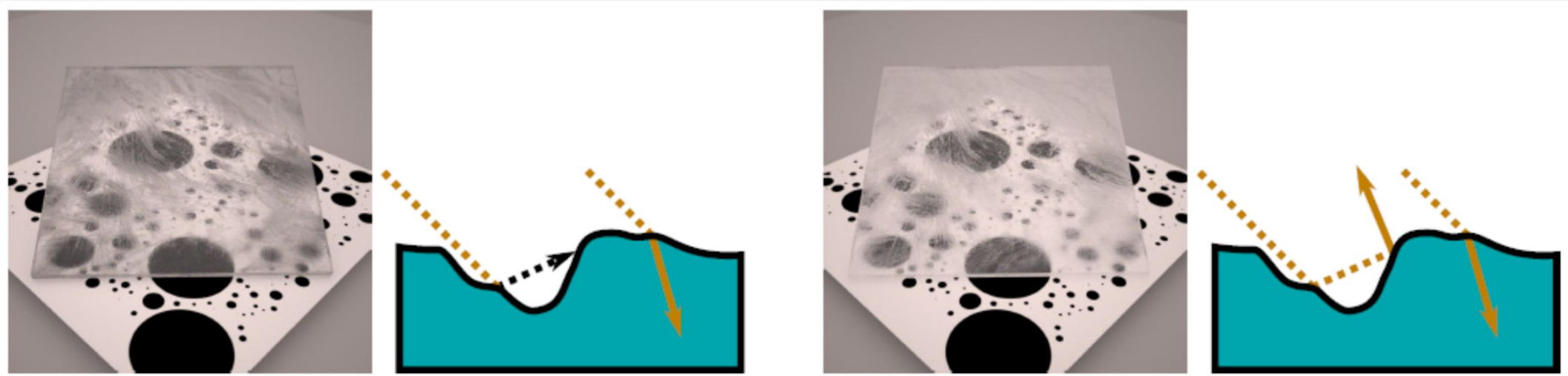
Stephen McAuley


ZU-5225

Multiple-Scattering Microfacet BSDFs with the Smith Model

Eric Heitz, Johannes Hanika, Eugene d'Eon and Carsten Dachsbacher

ACM SIGGRAPH 2016



[Heitz16a]

Revisiting Physically Based Shading at Imageworks

Christopher Kulla & Alejandro Conty

SIGGRAPH 2017

[Kulla17]

If the energy reflected for a given BRDF *f* and a viewing direction is:

$$E(\mu_o) = \int_0^{2\pi} \int_0^1 f(\mu_o, \mu_i, \phi) \mu_i \mathbf{d}\mu_i \mathbf{d}\phi$$

Then find a multiscattering BRDF f_{ms} such that energy is preserved:

$$\int_0^{2\pi} \int_0^1 (f(\mu_o, \mu_i, \phi) + \mathbf{f_{ms}}(\mu_o, \mu_i, \phi)) \mu_i \mathbf{d}\mu_i \mathbf{d}\phi = 1$$

The following BRDF fits that equation:

$$f_{\mathbf{ms}}(\mu_o, \mu_i) = \frac{(1 - E(\mu_o))(1 - E(\mu_i))}{\pi(1 - E_{\mathbf{avg}})}$$

Where:

$$E(\mu_o) = \int_0^{2\pi} \int_0^1 f(\mu_o, \mu_i, \phi)\mu_i \mathbf{d}\mu_i \mathbf{d}\phi$$

$$E_{\mathbf{avg}} = 2 \int_0^1 E(\mu)\mu \mathbf{d}\mu$$

In fact, that only holds for 100% reflective microfacets. Energy *is* lost between each bounce. Sum the loss and scale f_{ms} by:

$$\frac{F_{\mathbf{avg}}^2 E_{\mathbf{avg}}}{1 - F_{\mathbf{avg}}(1 - E_{\mathbf{avg}})}$$

Where:

$$F_{\mathbf{avg}} = 2 \int_0^1 F(\mu)\mu \mathbf{d}\mu$$

Need to Calculate:

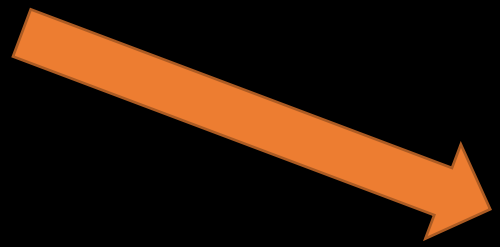
For a given roughness

- $1-E(\mu)$
- E_{avg}
- F_{avg}

```
std::fstream isotropicEnergyFile;
isotropicEnergyFile.open("isotropic_reflected_energy.csv", std::ios_base::out);

for (int j = 0; j < NumSmoothnessSamples; ++j)
{
    float s = float(j) / float(NumSmoothnessSamples - 1);
    float a = GGXAlphaFromSmoothness(s);

    for (int i = 0; i < NumDirectionSamples; ++i)
    {
        float mu = float(i) / float(NumDirectionSamples - 1);

        float isotropicReflectedEnergy = IsotropicReflectedEnergy(mu, a);

        char outputLine[128];
        sprintf_s(outputLine, sizeof(outputLine), "%12.10f, ", 1.0f - isotropicReflectedEnergy);

        isotropicEnergyFile.write(outputLine, strlen(outputLine));
    }

    isotropicEnergyFile.write("\n", 1);
}

isotropicEnergyFile.close();
```

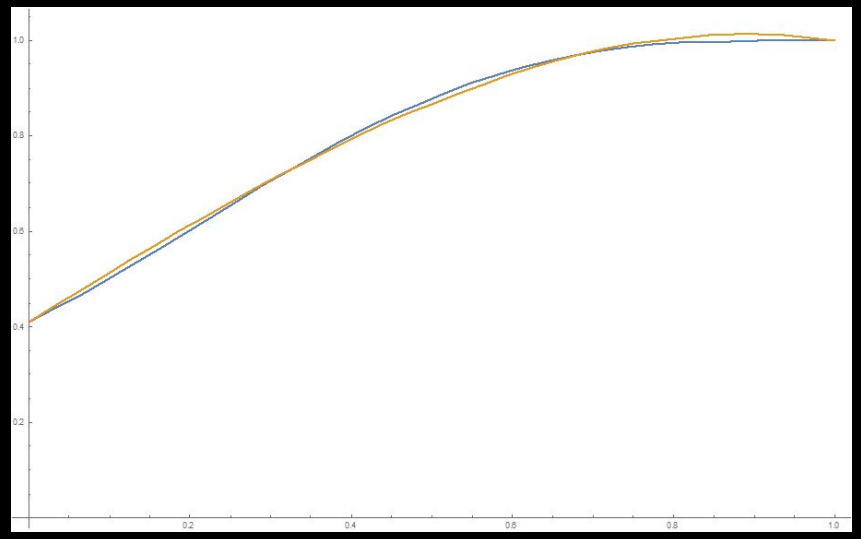
1-E(μ)

```
float AverageEnergy(float alpha)
{
    static const int NumBRDFSamples = 16384;

    float averageEnergy = 0.0f;
    for (int i = 0; i < NumBRDFSamples; ++i)
    {
        float u, v;
        GetQuasiRandomSequence(i, u, v);

        float3 h = UniformSampleHemisphere(u, v);

        float isotropicReflectedEnergy = IsotropicReflectedEnergy(h.z, alpha);
        averageEnergy += isotropicReflectedEnergy * h.z;
    }

    averageEnergy *= 2.0f;
    averageEnergy /= NumBRDFSamples;

    return averageEnergy;
}
```



	A	B
1	0	0.499810815
2	0.032258064	0.524018645
3	0.064516127	0.549035847
4	0.096774191	0.574677348
5	0.129032254	0.600818157
6	0.161290318	0.627211094
7	0.193548381	0.653858841
8	0.225806445	0.6804263
9	0.258064508	0.706650496
10	0.290322572	0.732294083
11	0.322580636	0.757151365
12	0.354838699	0.78101027
13	0.387096763	0.803810895
14	0.419354826	0.825328469
15	0.45161289	0.84570241
16	0.483870953	0.864568889
17	0.516129017	0.88201797
18	0.54838711	0.89817065
19	0.580645144	0.912830234
20	0.612903237	0.926163256
21	0.645161271	0.938052058
22	0.677419365	0.948895156
23	0.709677398	0.958968341
24	0.741935492	0.967251003
25	0.774193525	0.974402309
26	0.806451619	0.980794311
27	0.838709652	0.986319542
28	0.870967746	0.990950704
29	0.90322578	0.994659841
30	0.935483873	0.997420549
31	0.967741907	0.999197424
32	1	0.999991953

E_{avg}



$1 - E(\mu)$

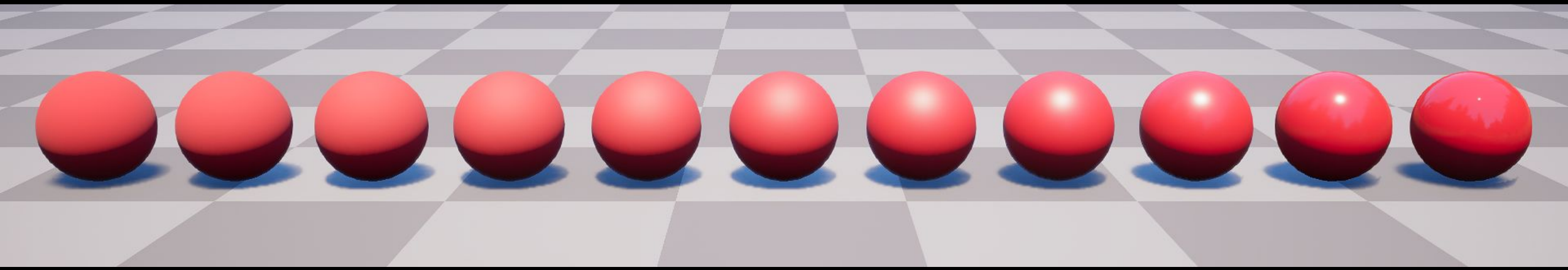
E_{avg}

```
float AverageEnergy(float smoothness)
{
    float r = -0.0761947f - 0.383026f * smoothness;
          r = 1.04997f + smoothness * r;
          r = 0.409255f + smoothness * r;
    return min(0.999f, r);
}
```

F_{avg}

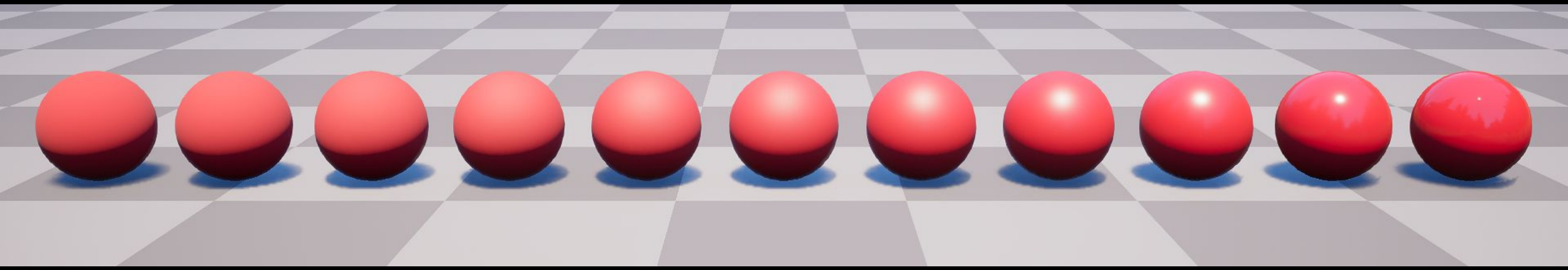
```
float3 AverageFresnel(float3 specularColor)
{
    return specularColor + (1.0f - specularColor) * (1.0f / 21.0f);
}
```

Dielectrics



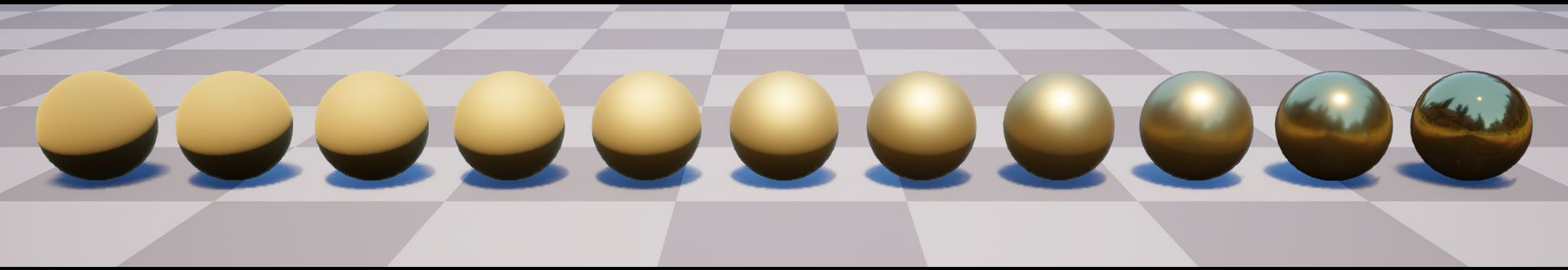
Multiscattering Specular Off

Dielectrics



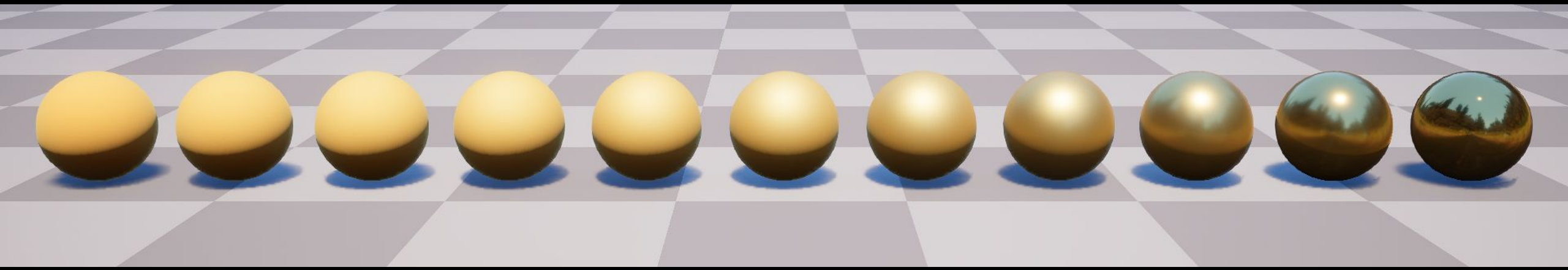
Multiscattering Specular On

Metals

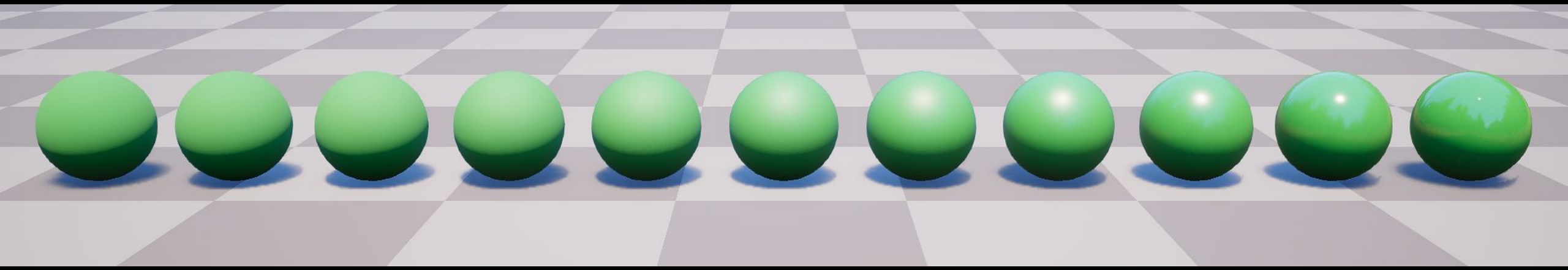


Multiscattering Specular Off

Metals



Multiscattering Specular On



Multiscattering Diffuse

Goals:

Improvements to Lambertian diffuse

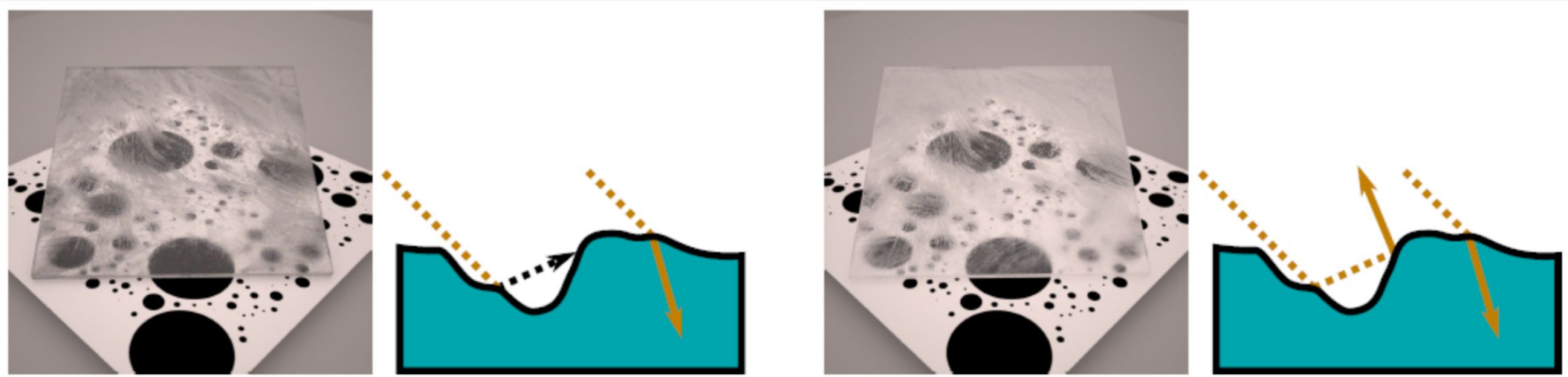
1. Multiscattering is taken into account
2. Diffuse reacts to surface roughness
3. Diffuse depends on the distribution of normals
4. Diffuse and specular are energy conserving

Material Advances in Call of Duty: WWII [Chan18]

Multiple-Scattering Microfacet BSDFs with the Smith Model

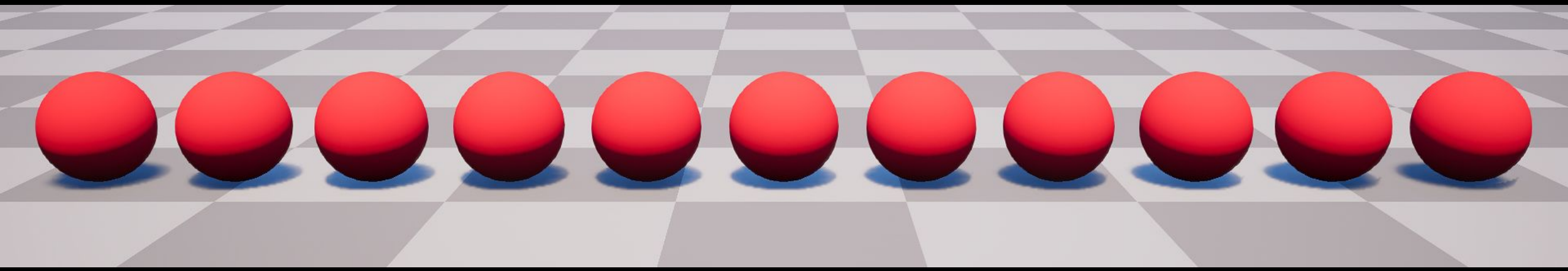
Eric Heitz, Johannes Hanika, Eugene d'Eon and Carsten Dachsbacher

ACM SIGGRAPH 2016

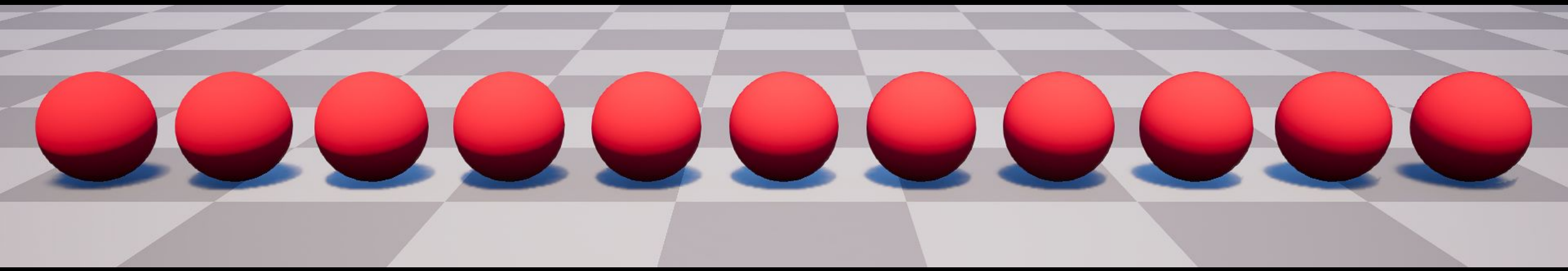


```
float MultiScatteringDiffuseBRDF(float lDotH, float nDotL, float nDotV,
                                 float nDotH, float smoothness)
{
    // Burley to CoD gloss reparametrization
    //   CoD   :  alpha2 = 2 / (1 + 2^(18g))
    //   Burley:  alpha2 = (1 - g)^4
    float g = saturate(0.18455f * log(2.0f / pow(1.0f - smoothness, 4.0f) - 1.0f));

    float f0 = lDotH + pow(1.0f – lDotH, 5.0f);
    float f1 = (1.0f - 0.75f * pow(1.0f – nDotL, 5.0f))
             * (1.0f - 0.75f * pow(1.0f – nDotV, 5.0f));
    float t = saturate(2.2f * g - 0.5f);
    float fd = f0 + (f1 - f0) * t;
    float fb = ((34.5f * g - 59.0f) * g + 24.5f) * lDotH
             * exp2(-max(73.2f * g - 21.2f, 8.9f) * sqrt(nDotH));

    return fd + fb;
}
```

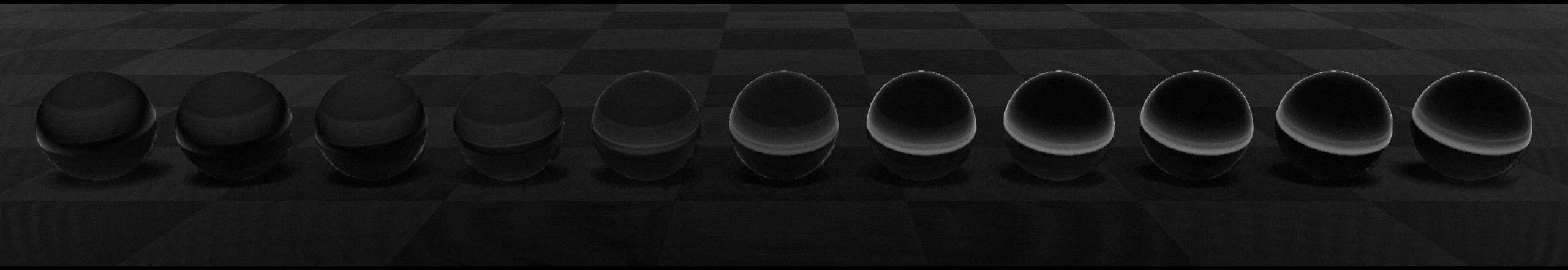
```
float MultiScatteringDiffuseBRDF(float lDotH, float nDotL, float nDotV,
                                 float nDotH, float smoothness)
{
    // Burley to CoD gloss reparametrization
    //   CoD    :  alpha2 = 2 / (1 + 2^(18g))
    //   Burley:  alpha2 = (1 - g)^4
    float g = saturate(0.18455f * log(2.0f / pow(1.0f - smoothness, 4.0f) - 1.0f));

    float f0 = lDotH + pow5(1.0f - lDotH);
    float f1 = (1.0f - 0.75f * pow5(1.0f - nDotL))
             * (1.0f - 0.75f * pow5(1.0f - nDotV));
    float t = saturate(2.2f * g - 0.5f);
    float fd = f0 + (f1 - f0) * t;
    float fb = ((34.5f * g - 59.0f) * g + 24.5f) * lDotH
             * exp2(-max(73.2f * g - 21.2f, 8.9f) * sqrt(nDotH));

    return fd + fb;
}
```



Lambert Diffuse



Multiscattering Diffuse



Difference x8

“Are we there yet?”

Surface Type	Diffuse BRDF	Specular BRDF
Skin	Pre-Integrated Subsurface Scattering (Lambert)	GGX
Hair	Lambert	Modified Marschner
Car Paint	Lambert	Two GGX lobes (Top layer and bottom layer)
Cloth	Lambert	Ashikhmin Cloth + Disney Sheen
Translucent	Two wrapped Lambert lobes (Front and back)	GGX
Default	Lambert	GGX

Light Type	Diffuse Evaluation	Specular Evaluation
Direct	Analytic	Analytic
Indirect	Spherical Harmonics	Screen-Space Reflections Pre-Integrated Cube Maps Pre-Integrated BRDF

Problems:

1. No multiscattering indirect specular
2. No multiscattering specular on hair
3. No multiscattering indirect diffuse
4. No multiscattering diffuse on skin
5. No multiscattering wrapped diffuse

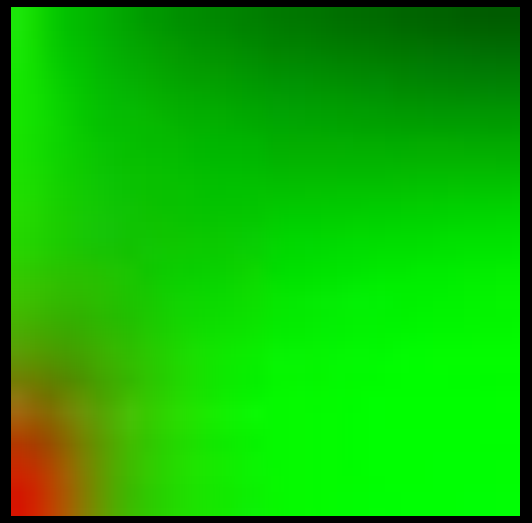
Problems:

1. No multiscattering indirect specular
2. No multiscattering wrapped diffuse
3. No multiscattering diffuse on skin
4. No multiscattering indirect diffuse
5. No multiscattering specular on hair

$$L_o(\omega_o) = \int_\Omega L_i(\omega_i) \frac{F(\omega_i, \omega_o) D(\omega_i, \omega_o) G(\omega_i, \omega_o)}{4 \cos\theta_i \cos\theta_o} \mathbf{d}\Omega$$

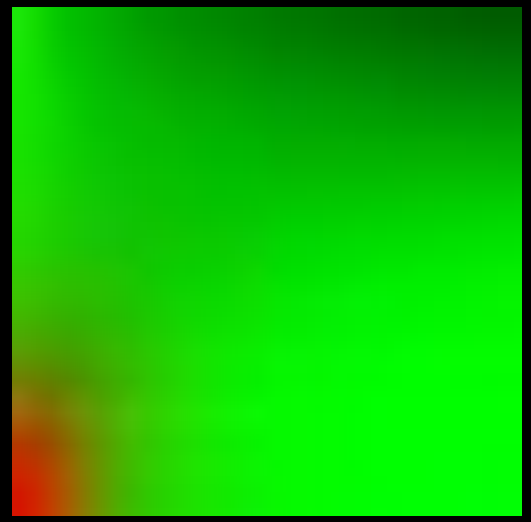
Indirect specular

$$L_o(\omega_o) \simeq \int_\Omega L_i(\omega_i) D(\omega_i, \omega_o) \mathbf{d}\Omega \int_\Omega \frac{F(\omega_i, \omega_o) D(\omega_i, \omega_o) G(\omega_i, \omega_o)}{4 \cos\theta_i \cos\theta_o} \mathbf{d}\Omega$$



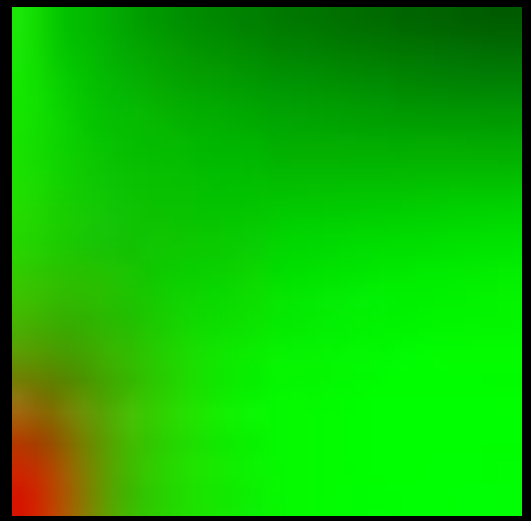
Split-Sum Approximation [Karis13]

$$f_{\mathbf{EnvBRDF}}(\omega_o) = \int_\Omega \frac{F(\omega_i, \omega_o) D(\omega_i, \omega_o) G(\omega_i, \omega_o)}{4 \cos\theta_i \cos\theta_o} \mathbf{d}\Omega$$



Environment Map BRDF

$$f_{\mathbf{EnvBRDF}}(\omega_o) = \int_\Omega (1 - \cos\omega_o)^5 \frac{D(\omega_i, \omega_o) G(\omega_i, \omega_o)}{4 \cos\theta_i \cos\theta_o} \mathbf{d}\Omega + F_0 \int_\Omega (1 - (1 - \cos\omega_o)^5) \frac{D(\omega_i, \omega_o) G(\omega_i, \omega_o)}{4 \cos\theta_i \cos\theta_o} \mathbf{d}\Omega$$

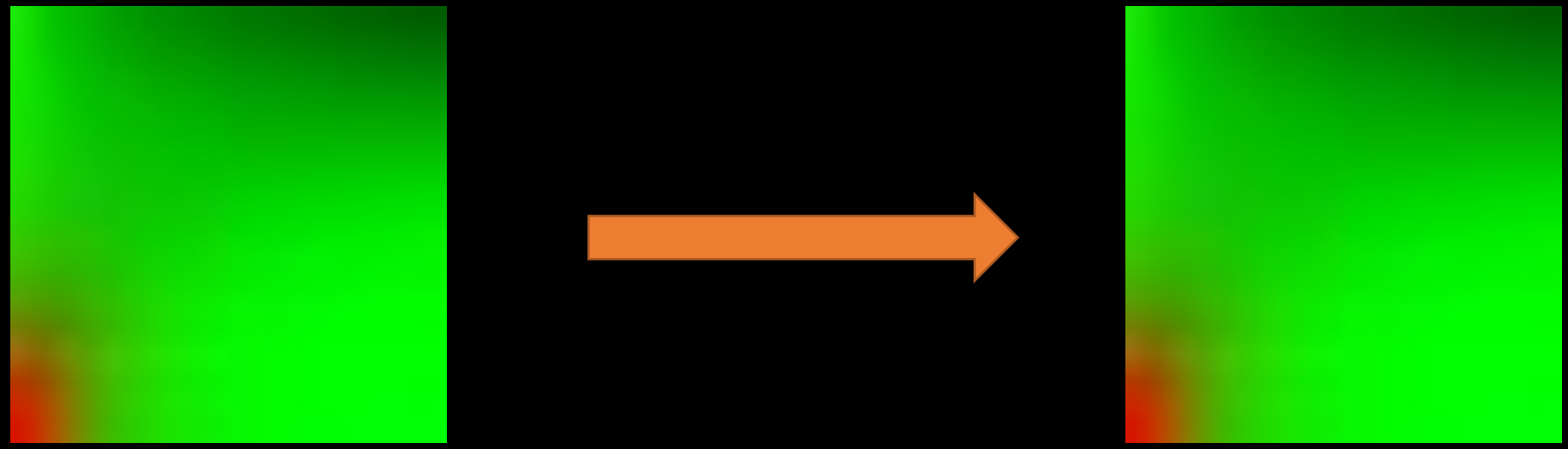


Environment Map BRDF

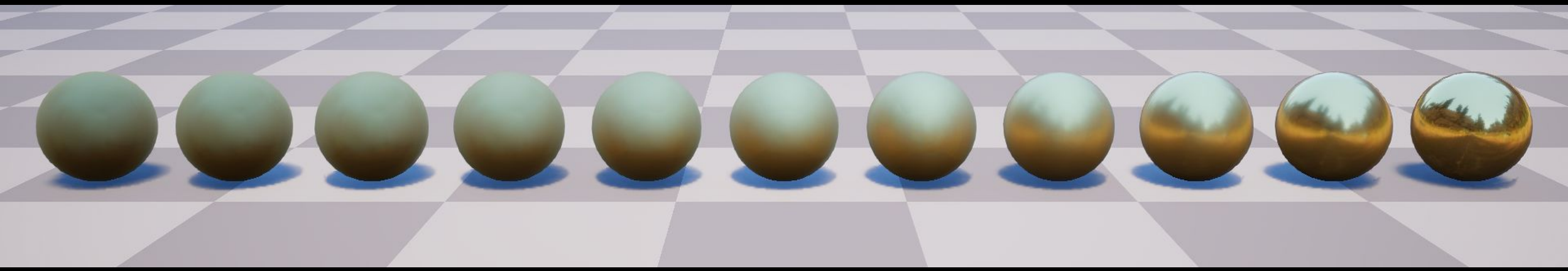
Multiscattering Environment Map BRDF

Multiscattering Environment Map BRDF

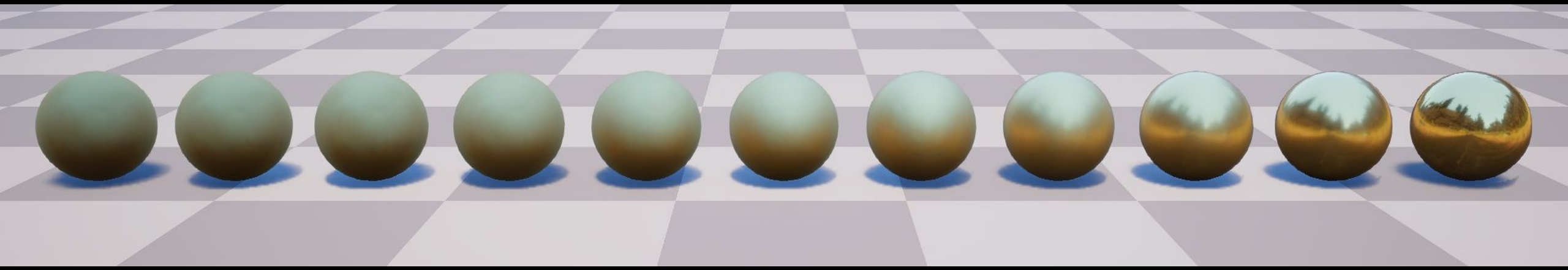
Multiscattering Environment Map BRDF



Multiscattering Environment Map BRDF



Environment Map BRDF

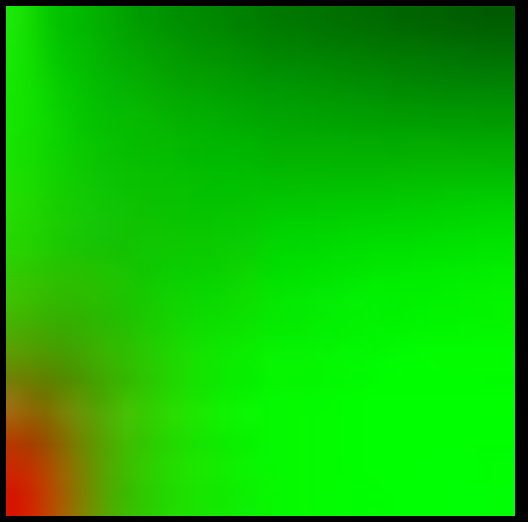


Multiscattering Environment Map BRDF

[FdezAgüera19]

The paper observes that the single-scattering energy is in fact the sum of the red and green channels in our environment BRDF:

$$E(\mu_o) = \int_0^{2\pi} \int_0^1 f(\mu_o, \mu_i, \phi) \mu_i \mathbf{d}\mu_i \mathbf{d}\phi$$



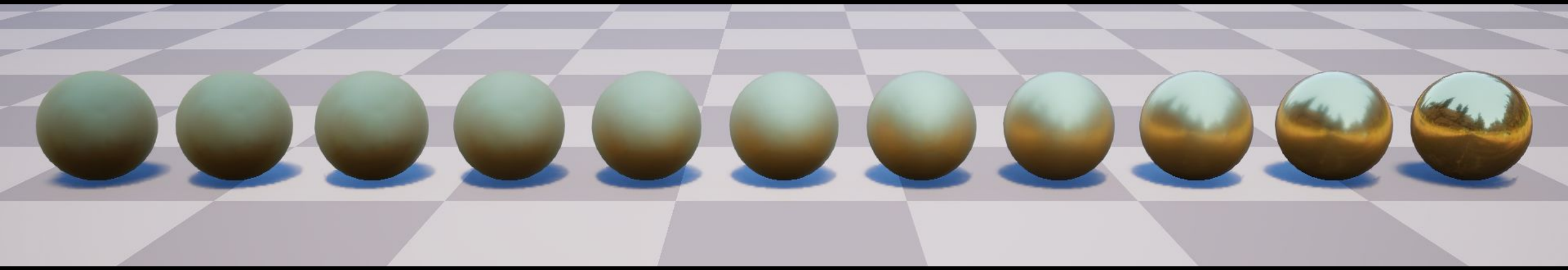
It also observes that E_{avg} can be approximated as $E(\mu)$

Given that F_{avg} can be calculated analytically, and multiply-scattered light is diffuse, we get the following formula:

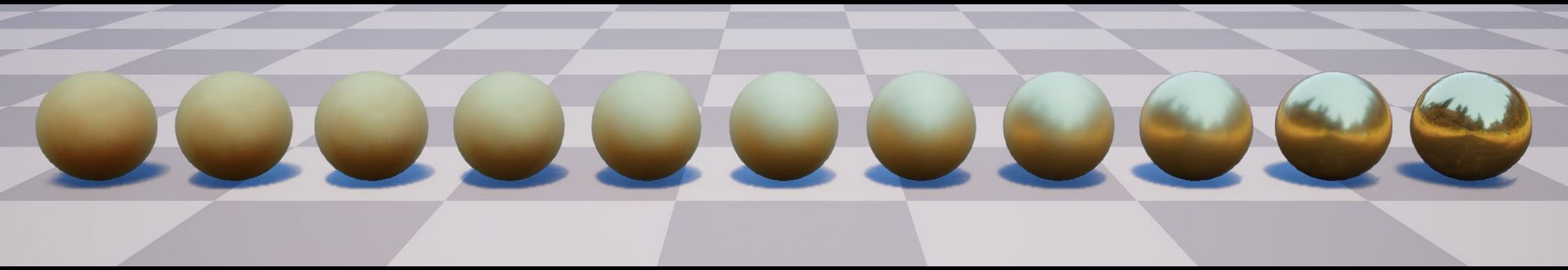
```
float2 FssEss = envBRDF.x + F0 * envBRDF.y;
float  Ess    = envBRDF.x + envBRDF.y;
float  Ems    = 1.0f - Ess;
float  Favg   = F0 + (1.0f / 21.0f) * (1.0f - F0);
float  Fms    = FssEss * Favg / (1.0f - Favg * (1.0f - Ess));
float  Lss    = FssEss * radiance;
float  Lms    = Fms * Ems * irradiance;

return Lss + Lms;
```

Given that F_{avg} can be calculated analytically, and multiply-scattered light is diffuse, we get the following formula:

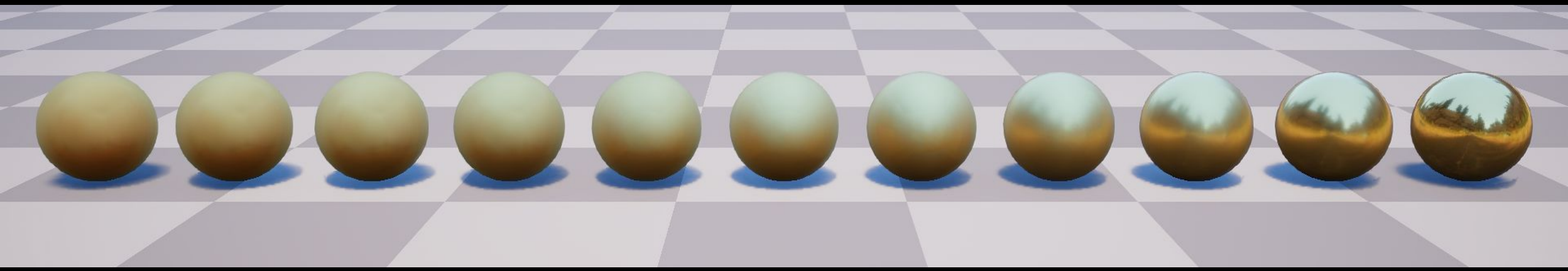
```
float2 FssEss = envBRDF.x + F0 * envBRDF.y;
float  Ess    = envBRDF.x + envBRDF.y;
float  Ems    = 1.0f - Ess;
float  Favg   = F0 + (1.0f / 21.0f) * (1.0f - F0);
float  Fms    = FssEss * Favg / (1.0f - Favg * (1.0f - Ess));
float  Lss    = FssEss * radiance;
float  Lms    = Fms * Ems * radiance;

return Lss + Lms;
```



Single Scattering



Fdez-Agüera



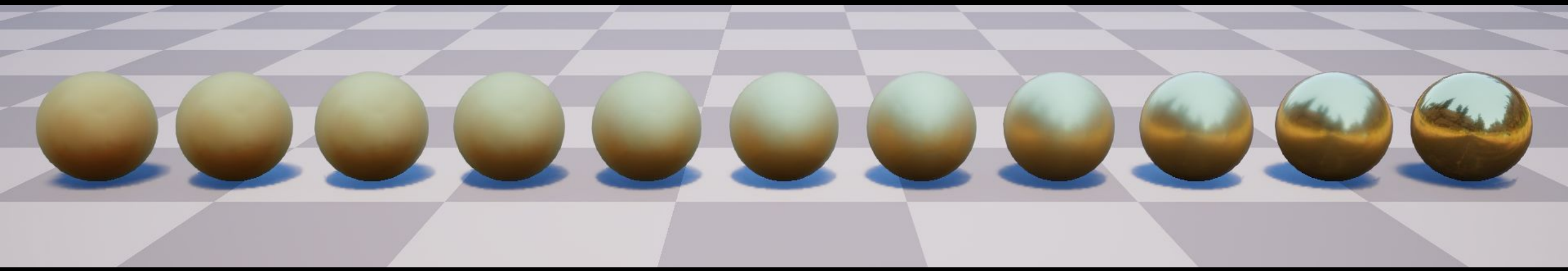
Approximation to Fdez-Agüera

This gives a multiscattering formula for the environment BRDF as:

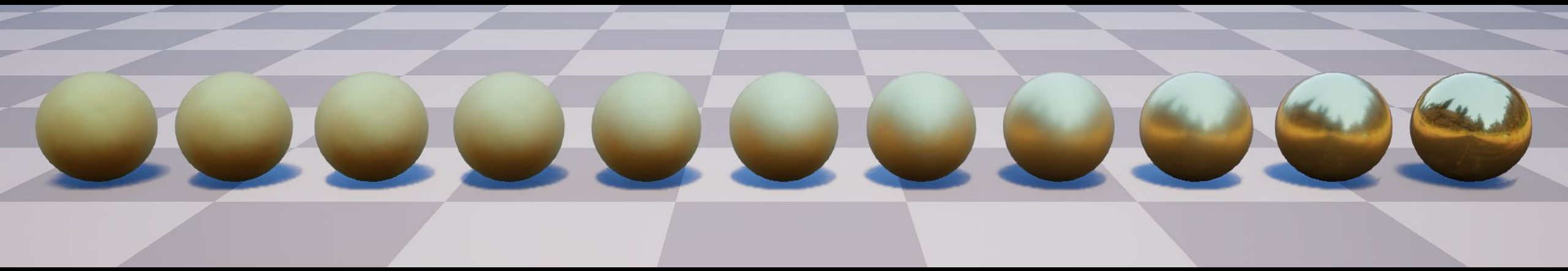
$$f_{\mathbf{ms}}(\mu_o, \mu_i) = \frac{F_{\mathbf{avg}}(1 - E(\mu))}{1 - F_{\mathbf{avg}}(1 - E(\mu))} f_{\mathbf{ss}}(\mu_o, \mu_i)$$

$$f_{\mathbf{ms}}(\mu_o, \mu_i) = \frac{F_0(1 - E(\mu_o))}{E(\mu_o)} f_{\mathbf{ss}}(\mu_o, \mu_i)$$

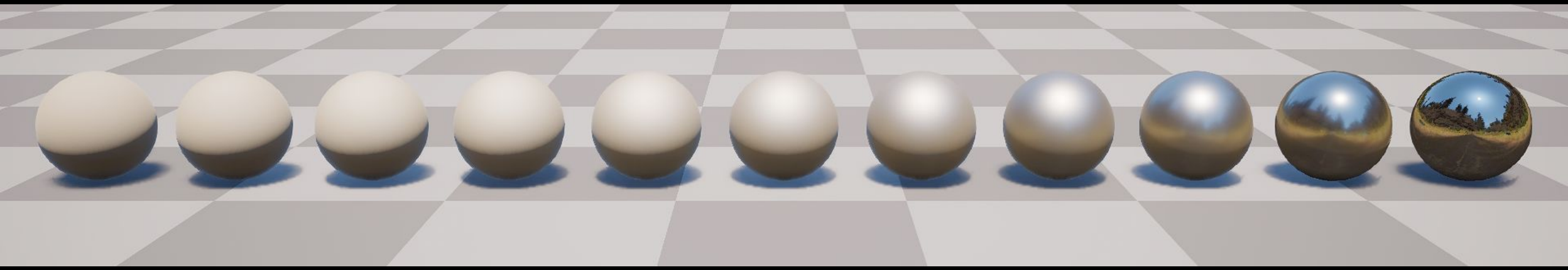
[Turquin19]



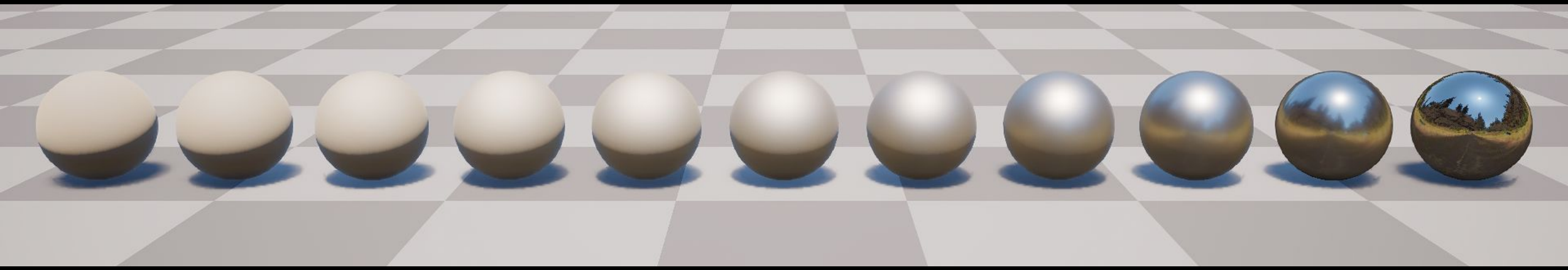
Approximation to Fdez-Agüera



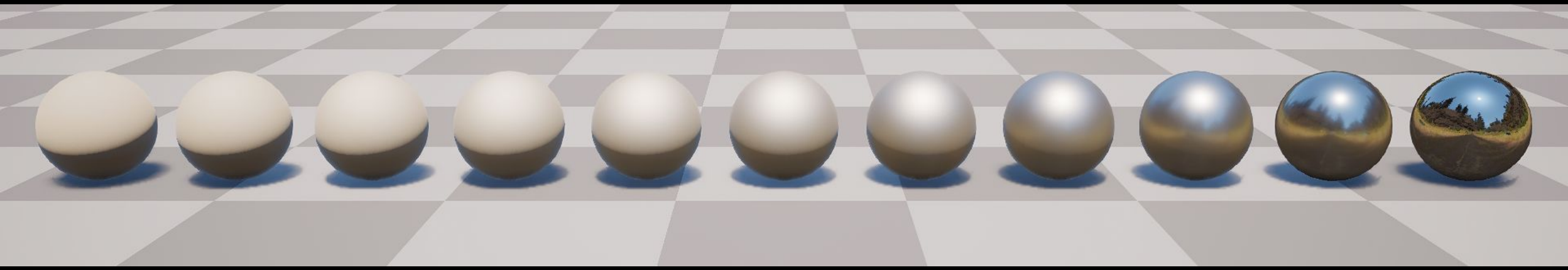
Turquin



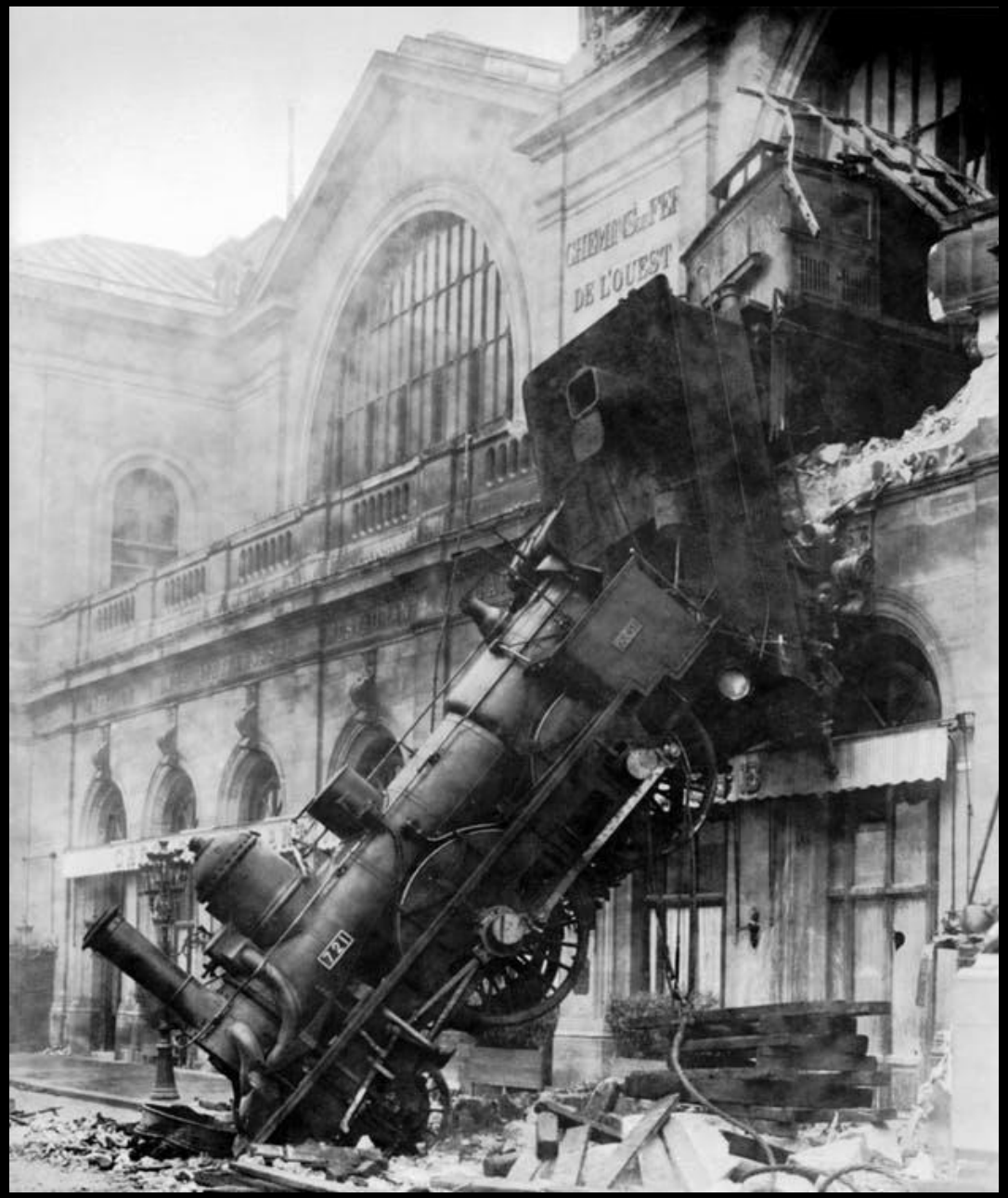
Multiscattered Specular [Kulla17]



Approximation to Fdez-Agüera



Turquin


CHEMINS DE FER
DE L'OUEST

$$F_{\text{avg}} = 2\pi \int_0^{\pi/2} (F_0 + (1 - F_0)(1 - \cos(\theta))^5) \sin\theta \cos\theta d\theta, \tag{10}$$

$$F_{\text{avg}} = F_0 + \frac{\pi}{21}(1 - F_0). \tag{11}$$

JCGT
@JCGT_announce

Following

Update posted today to Fdez-Agüera, A Multiple-Scattering Microfacet Model for Real-Time Image Based Lighting. Primarily fixes some factor-of-pi errors.
jcgt.org/published/0008…

5:34 PM - 1 Feb 2019

28 Retweets **48** Likes

28 48

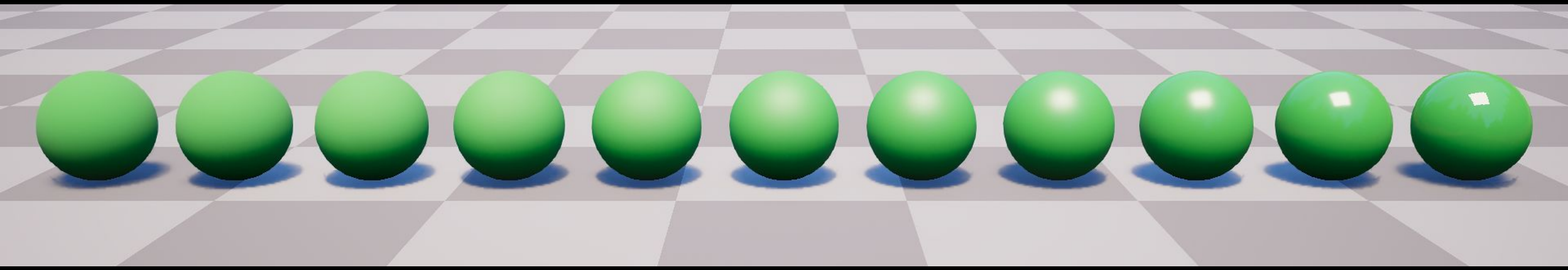
Problems:

1. No multiscattering indirect specular
2. No multiscattering specular on hair
3. No multiscattering indirect diffuse
4. No multiscattering diffuse on skin
5. No multiscattering wrapped diffuse


Important!
To do:

Problems:

1. No multiscattering indirect specular
2. No multiscattering specular on hair
3. No multiscattering indirect diffuse
4. No multiscattering diffuse on skin
5. No multiscattering wrapped diffuse



Area Lights

Goals:

1. Improve cinematic lighting:
   - Softer light falloffs

2. More realistic specular response:
   - Broader, more visible highlights
   - Artists authoring smoother materials

Real-Time Polygonal-Light Shading with Linearly Transformed Cosines

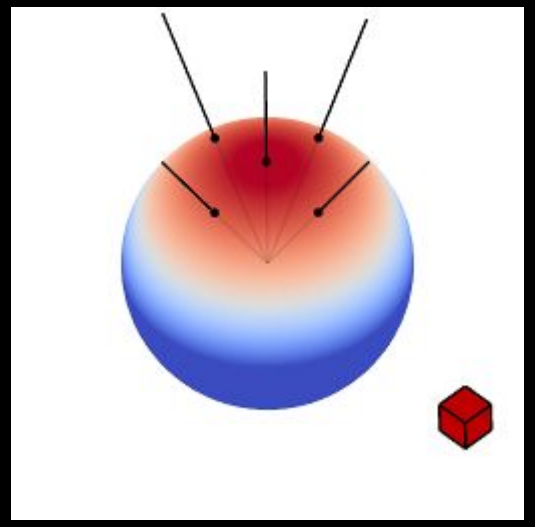
Eric Heitz, Jonathan Dupuy, Stephen Hill and David Neubelt

ACM SIGGRAPH 2016

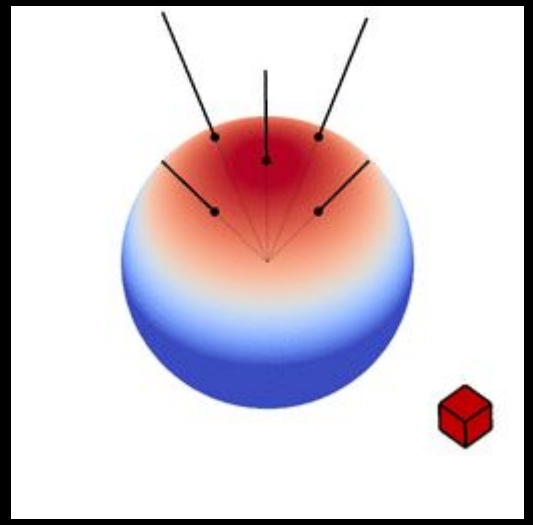
[Heitz16b]

Why LTCs?

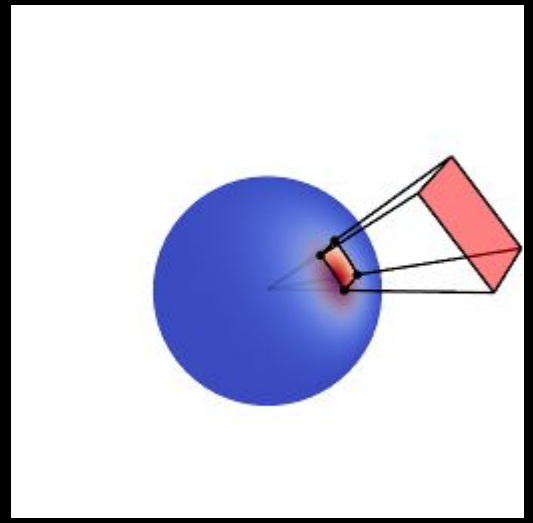
1. Fast to implement
2. Full source code and demos available
3. Flexibility in performance and light types
4. Performant and robust



A clamped cosine distribution can be analytically integrated over polygonal shapes



We can linearly transform this distribution to approximate BRDFs



Integrating a polygon over an LTC becomes integrating a polygon over the analytically solvable clamped cosine distribution

Q. LTCs always integrate to 1, but what about the actual magnitude of the BRDF?

A. Integrate the BRDF and store the magnitude in a LUT.

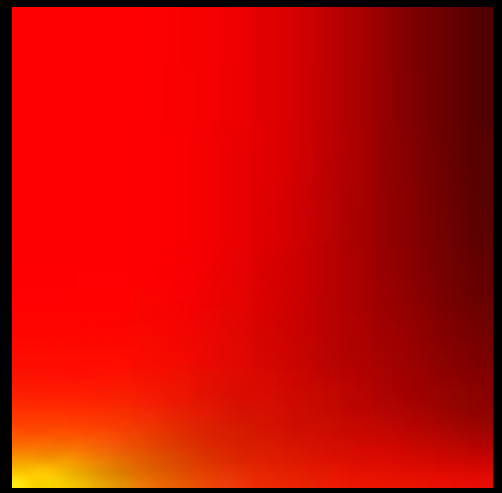
$$\int_{\Omega} F(\omega_i, \omega_o)\rho(\omega_i, \omega_o)\cos\theta_i \mathbf{d}\omega_i$$

Separate out Fresnel so we can take F_0 into account.

[Hill16]

Apply Schlick's approximation to Fresnel to get two components:

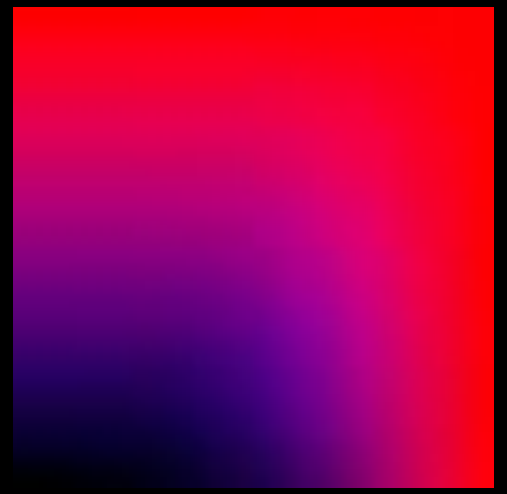
$$F_0 \int_\Omega \rho(\omega_i, \omega_o) \cos\theta_i \mathbf{d}\omega_i + (1 - F_0) \int_\Omega (1 - \mathbf{h} \cdot \mathbf{v})^5 \rho(\omega_i, \omega_o) \cos\theta_i \mathbf{d}\omega_i$$



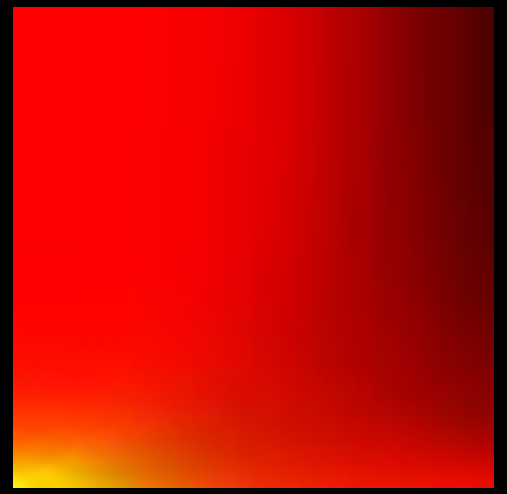
[Hill2016]

Implementation:

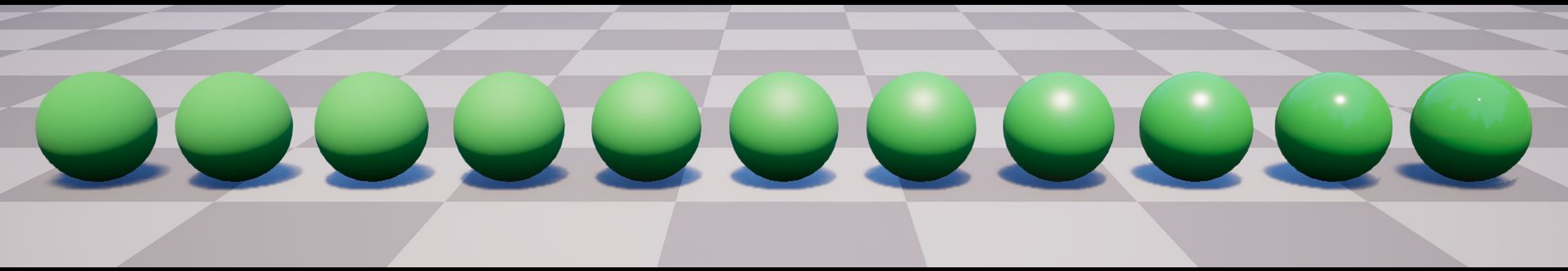
1. Offline, generate look up tables:
   - Inverse matrix transform
   - Magnitude and Fresnel
2. In the shader:
   - Calculate area light coordinates
   - Apply inverse transform
   - Integrate polygon over sphere with a clamped cosine distribution
   - Scale by BRDF magnitude and Fresnel



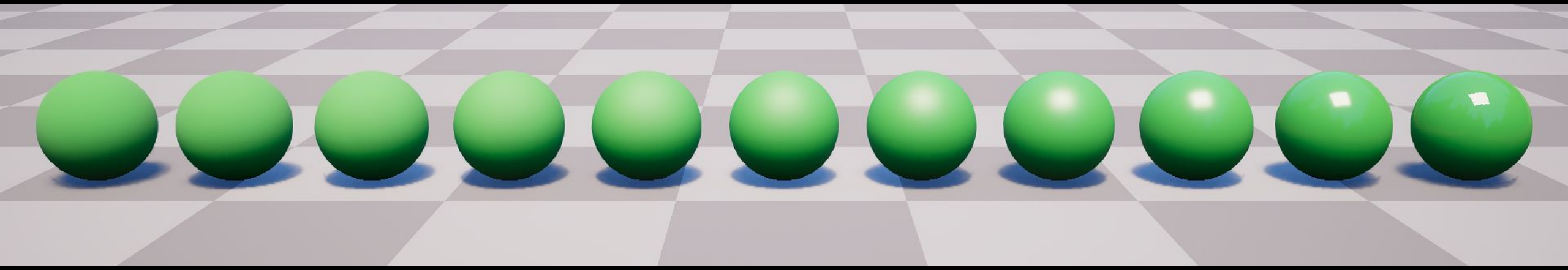
Inverse Matrix LUT



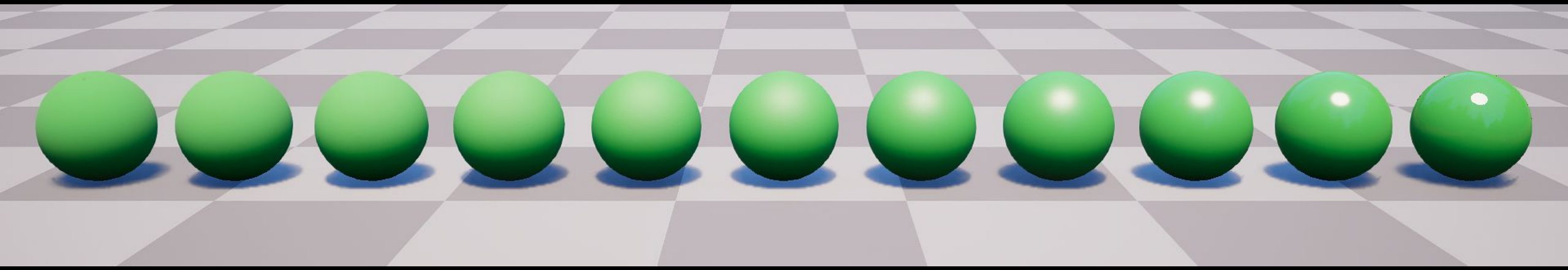
Magnitude and Fresnel LUT



Point Light



Quad Light



Disk Light

“Are we there yet?”

Surface Type	Diffuse BRDF	Specular BRDF
Skin	Pre-Integrated Subsurface Scattering (Lambert)	GGX
Hair	Lambert	Modified Marschner
Car Paint	Lambert	Two GGX lobes (Top layer and bottom layer)
Cloth	Lambert	Ashikhmin Cloth + Disney Sheen
Translucent	Two wrapped Lambert lobes (Front and back)	GGX
Default	Lambert	GGX

Surface Type	Diffuse BRDF	Specular BRDF
Skin	Pre-Integrated Subsurface Scattering (Lambert)	GGX + Multiscattering Lobe
Hair	Multiscattering Diffuse	Modified Marschner + ???
Car Paint	Multiscattering Diffuse	Two GGX + Multiscattering Lobes (Top layer and bottom layer)
Cloth	Multiscattering Diffuse	Ashikhmin Cloth + Disney Sheen
Translucent	Two wrapped Lambert lobes (Front and back)	GGX + Multiscattering Lobe
Default	Multiscattering Diffuse	GGX + Multiscattering Lobe

Light Type	Diffuse Evaluation	Specular Evaluation
Area	Analytic with Pre-Integrated BRDF	Analytic with Pre-Integrated BRDF
Direct	Analytic	Analytic
Indirect	Spherical Harmonics	Screen-Space Reflections Pre-Integrated Cube Maps Pre-Integrated BRDF

Problems:

1. No implementation for cloth
2. No implementation for multiscattering diffuse
3. No implementation for multiscattering specular
4. How to combine LTCs with wrapped diffuse
5. How to combine LTCs with pre-integrated skin scattering
6. No implementation of Marschner approximation for hair

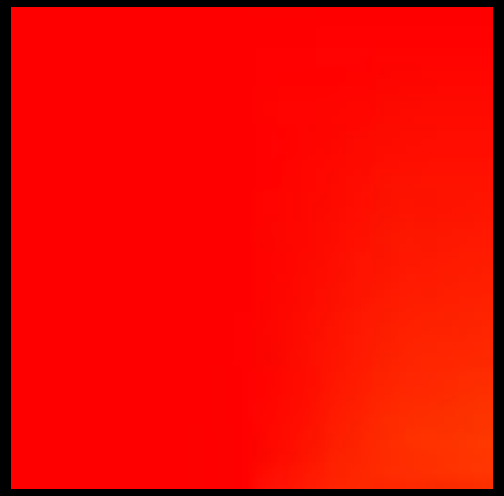
Source code available!

https://github.com/selfshadow/ltc_code

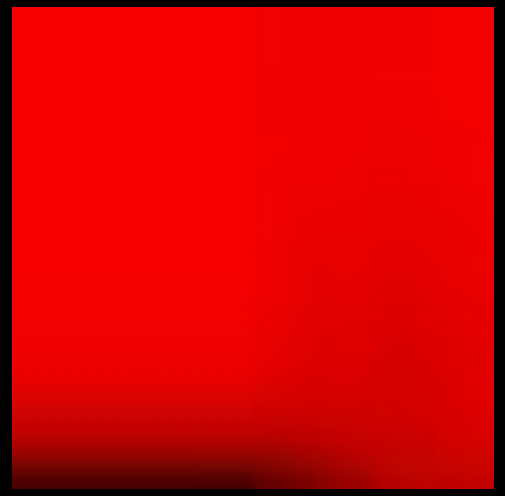
Problems:

1. No implementation for cloth
2. No implementation for multiscattering diffuse
3. No implementation for multiscattering specular
4. How to combine LTCs with wrapped diffuse
5. How to combine LTCs with pre-integrated skin scattering
6. No implementation of Marschner approximation for hair

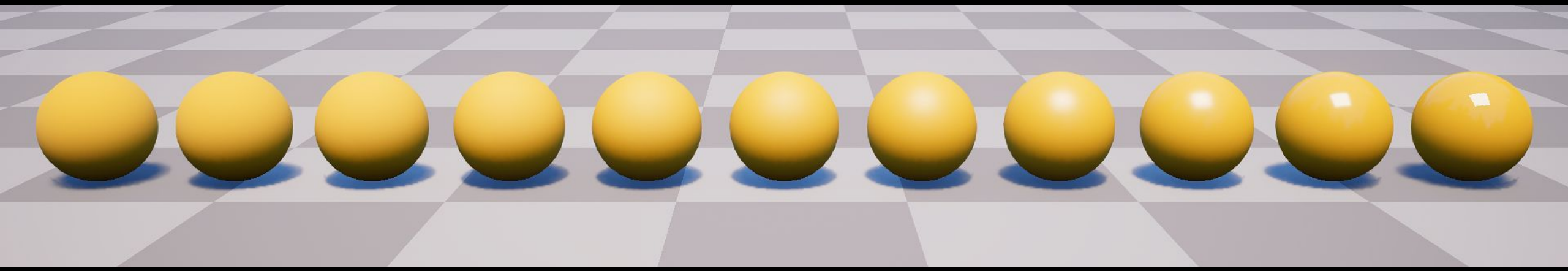
Inverse Matrix LUT



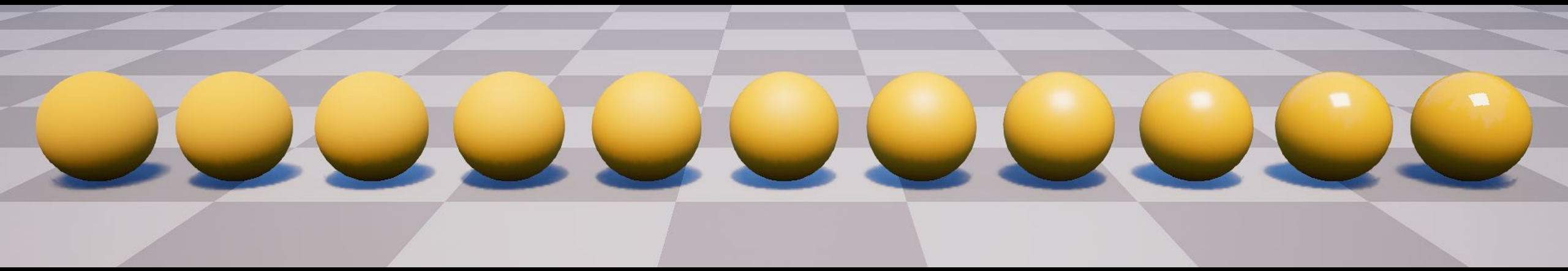
Magnitude and Fresnel LUT



LTCs for Multiscattering Diffuse

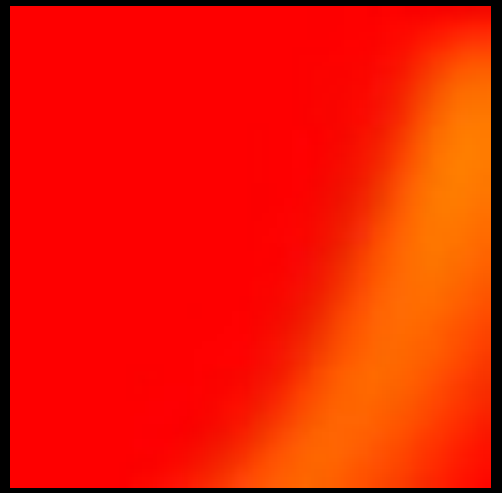


Lambert Diffuse

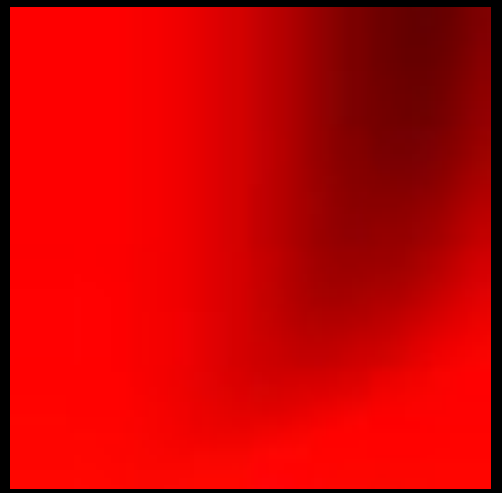


Multiscattering Diffuse

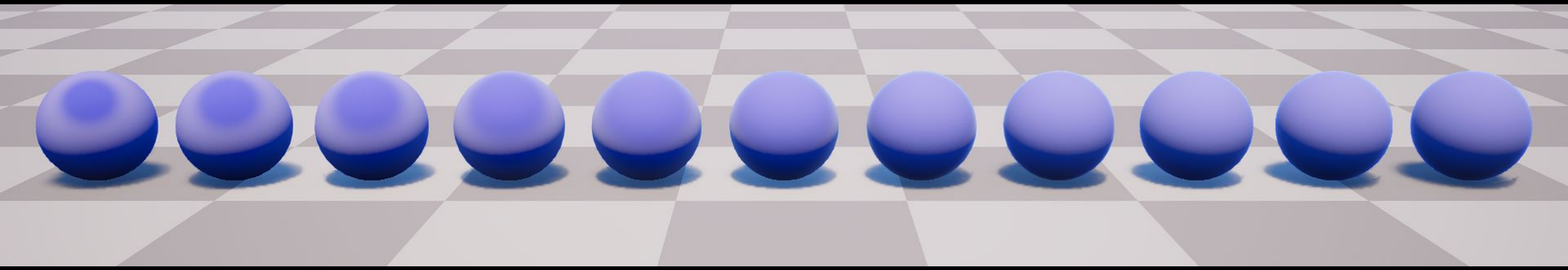
Inverse Matrix LUT



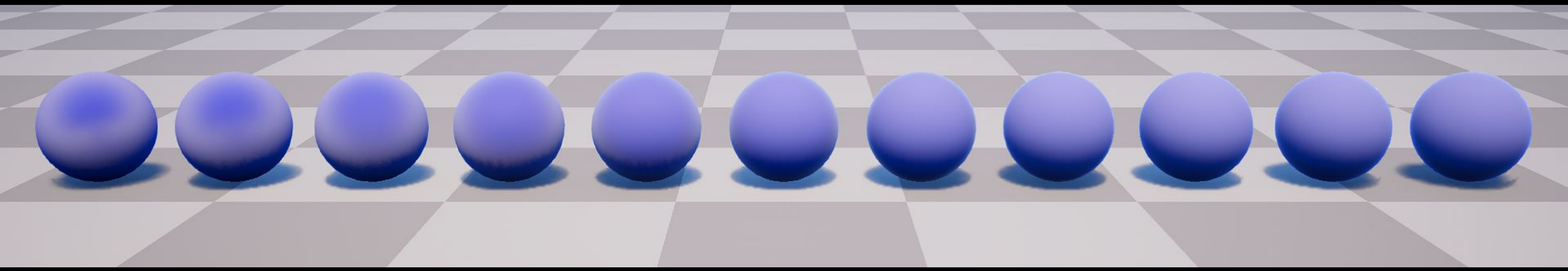
Magnitude and Fresnel LUT



LTCs for Ashikhmin Cloth [Ashikhmin00]



Ashikhmin Cloth: Point Light

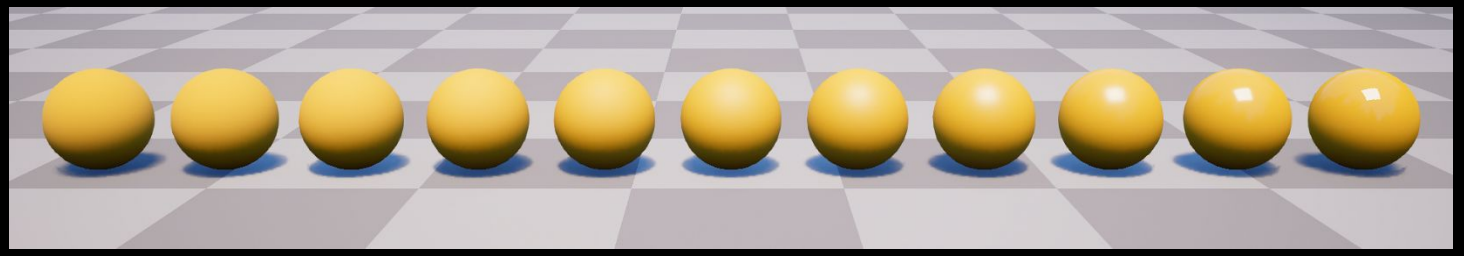


Ashikhmin Cloth: Area Light

$$f_{\text{sheen}}(\theta) = (1 - \cos\theta)^5$$

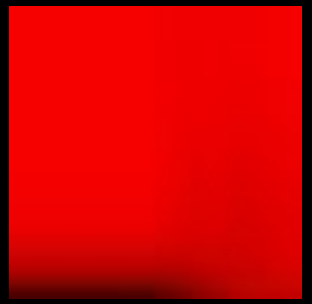
Disney Sheen [Burley12]

Lambert Diffuse

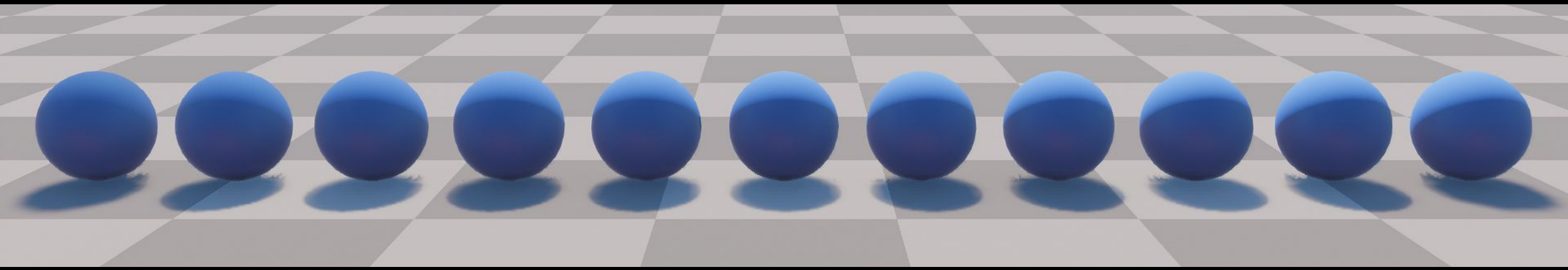


X

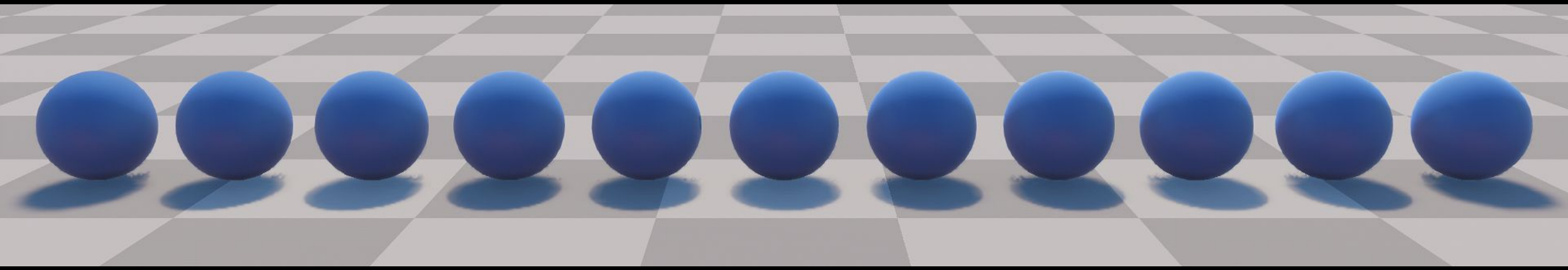
$$f_{\text{sheen}}(\theta) \simeq \text{LTC}_{\text{MagFresnel,MSDiffuse}}(\theta, 0.5)$$



Magnitude and Fresnel LUT for Multiscattering Diffuse



Disney Sheen: Point Light



Disney Sheen: Area Light

Problems:

1. No implementation for cloth
2. No implementation for multiscattering diffuse
3. No implementation for multiscattering specular
4. How to combine LTCs with wrapped diffuse
5. How to combine LTCs with pre-integrated skin scattering
6. No implementation of Marschner approximation for hair

LTC magnitude and Fresnel relies on a linear dependency on F_0:

$$F_0 \int_\Omega \rho(\omega_i, \omega_o) \cos\theta_i \mathbf{d}\omega_i + (1 - F_0) \int_\Omega (1 - \mathbf{h} \cdot \mathbf{v})^5 \rho(\omega_i, \omega_o) \cos\theta_i \mathbf{d}\omega_i$$

But our multiscattering BRDF has a non-linear dependency:

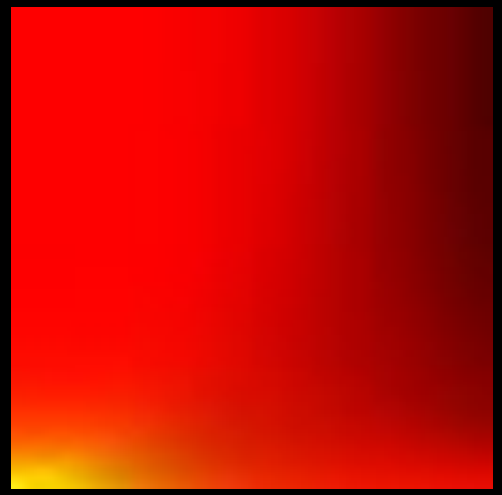
$$f_{\mathbf{ms}}(\mu_o, \mu_i) = \frac{F_{\mathbf{avg}}^2 E_{\mathbf{avg}}}{1 - F_{\mathbf{avg}}(1 - E_{\mathbf{avg}})} \frac{(1 - E(\mu_o))(1 - E(\mu_i))}{\pi(1 - E_{\mathbf{avg}})}$$

[FdezAgüera19]

We have a formula for a multiscattering BRDF:

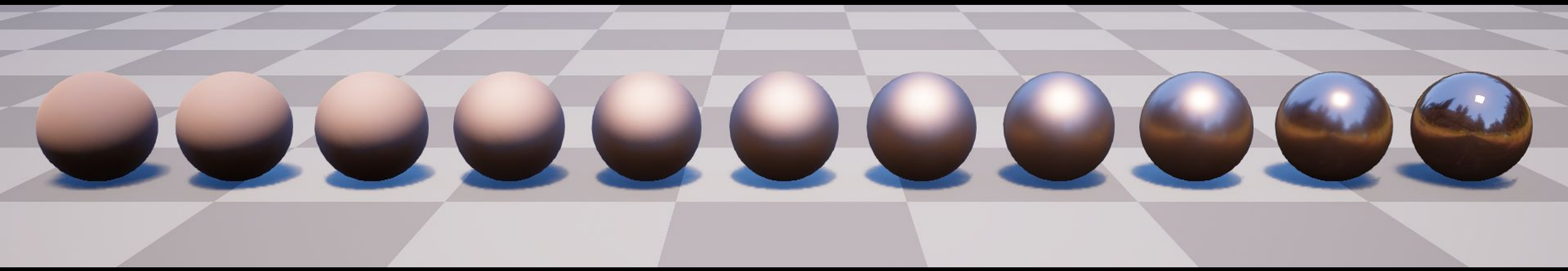
$$f_{\mathbf{ms}}(\mu_o, \mu_i) = \frac{F_{\mathbf{avg}}(1 - E(\mu))}{1 - F_{\mathbf{avg}}(1 - E(\mu))} f_{\mathbf{ss}}(\mu_o, \mu_i)$$

E(μ) is the red channel in our magnitude and Fresnel LUT:

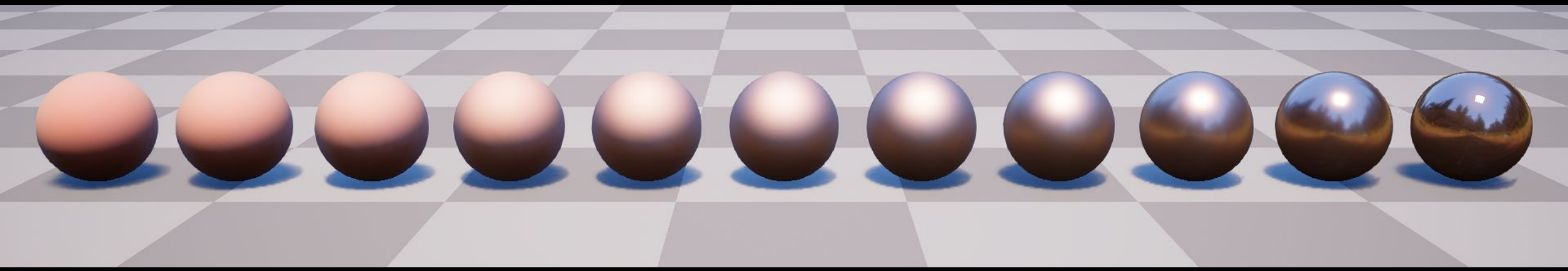


$$f_{\mathbf{ms}}(\mu_o, \mu_i) = \frac{F_0(1 - E(\mu_o))}{E(\mu_o)} f_{\mathbf{ss}}(\mu_o, \mu_i)$$

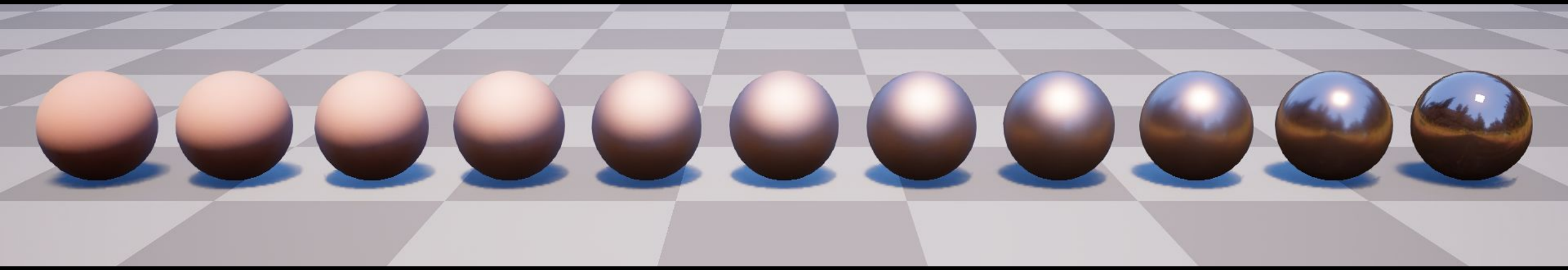
[Turquin19]



Single Scattering Specular



Approximation to Fdez-Agüera

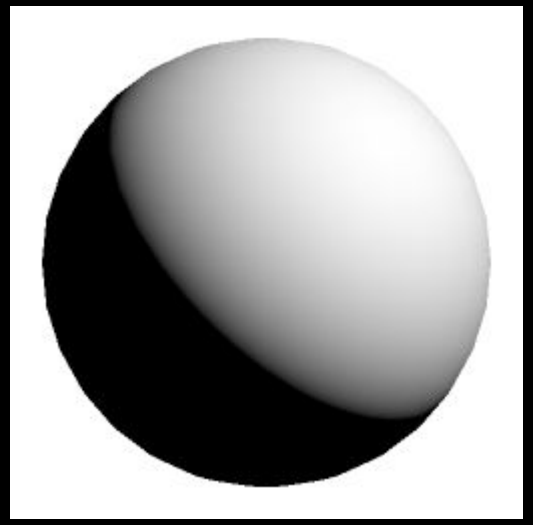


Turquin

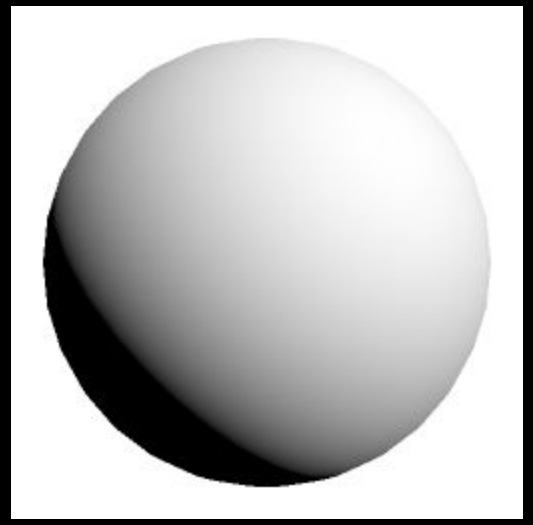
Problems:

1. No implementation for cloth
2. No implementation for multiscattering diffuse
3. No implementation for multiscattering specular
4. How to combine LTCs with wrapped diffuse
5. How to combine LTCs with pre-integrated skin scattering
6. No implementation of Marschner approximation for hair

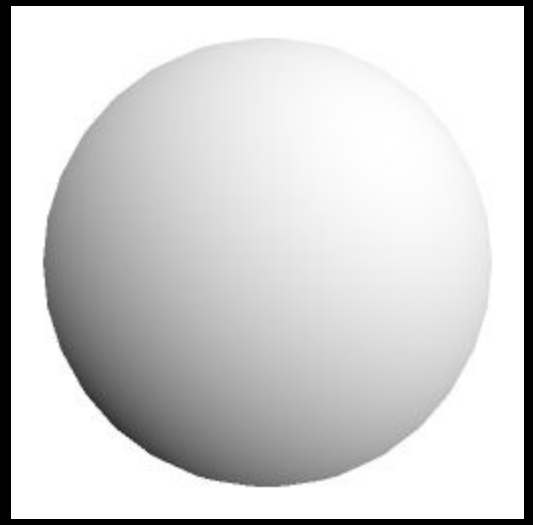
Wrapped Lambertian Diffuse: $\frac{1}{\pi}\frac{\cos\theta + w}{1 + w}$



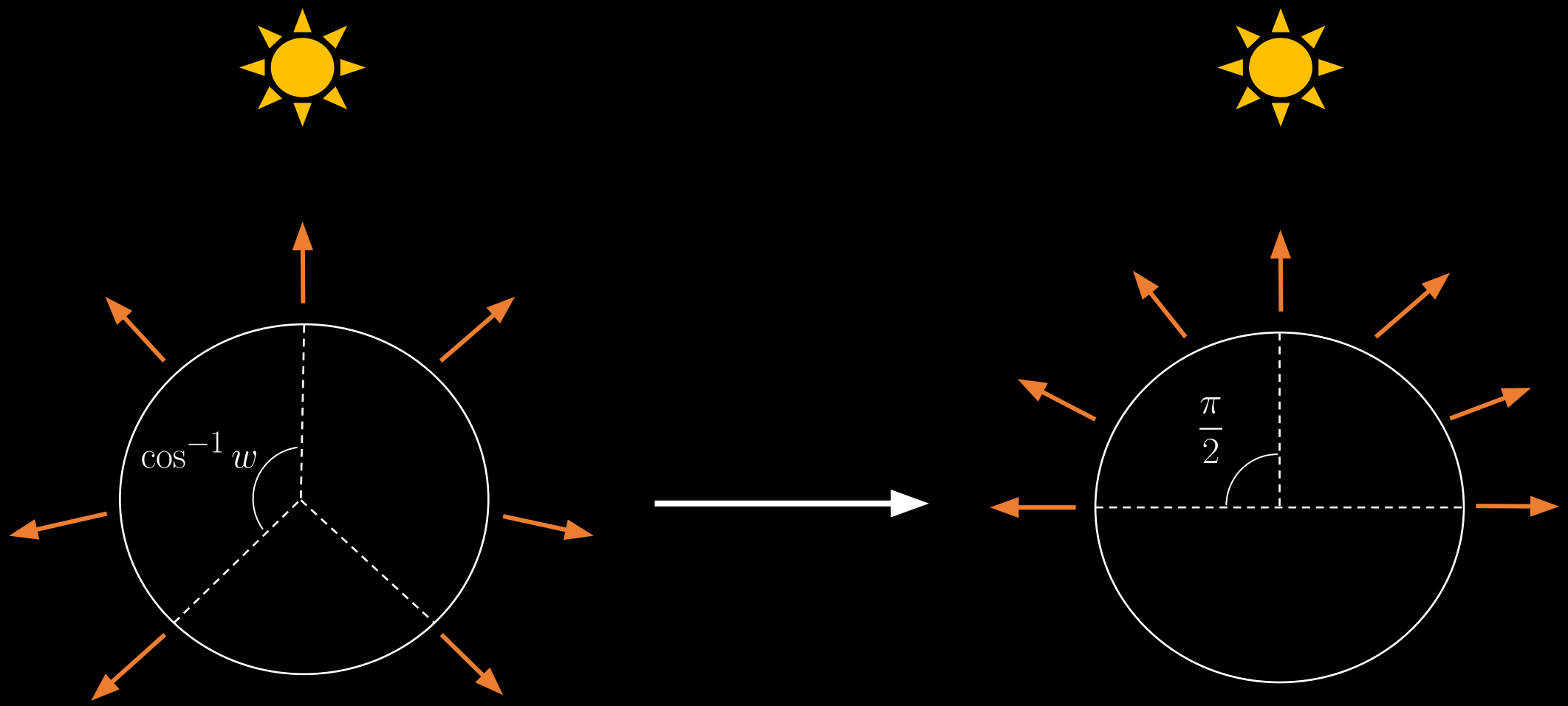
w = 0



w = 0.5



w = 1.0



Rotate the normal towards the light

Axis: $$\mathbf{v} = \mathbf{n} \times \mathbf{l}$$

Angle: $$\phi = \mathbf{lerp}(\cos^{-1} w - \frac{\pi}{2}, 0, \frac{\cos\theta + w}{1 + w})$$

Axis-angle rotation

Axis: $$\mathbf{v} = \mathbf{n} \times \mathbf{l}$$

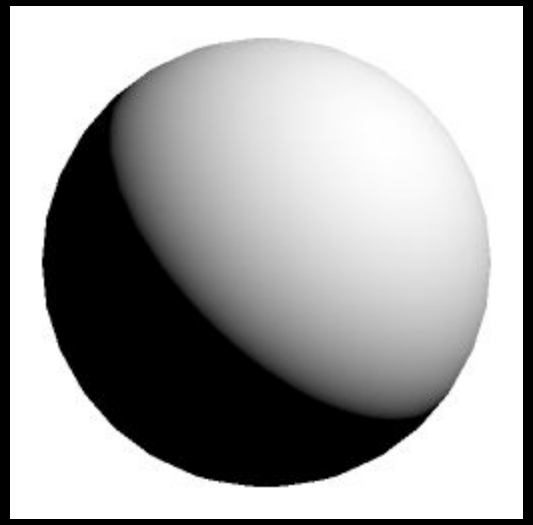
Sin Angle: $$\sin\theta = \mathbf{lerp}(w, 0, \frac{\cos\theta + w}{1 + w})$$

Axis-angle rotation

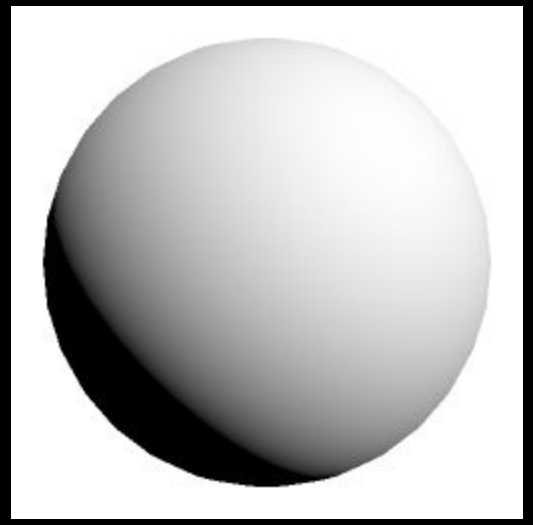
```
float3 WrappedNormal(const float3 N, const float3 L, const float w)
{
    float  cosTheta = dot(N, L);
    float  wrappedCosTheta = saturate((cosTheta + w) / (1 + w));
    float  sinMaximumAngleChange = w;
    float  sinMinimumAngleChange = 0.0f;
    float  sinPhi = lerp(sinMaximumAngleChange, sinMinimumAngleChange, wrappedCosTheta);
    float  cosPhi = sqrt(1.0f - sinPhi * sinPhi);
    return normalize(cosPhi * N + sinPhi * cross(cross(N, L), N));
}
```

Axis-angle rotation

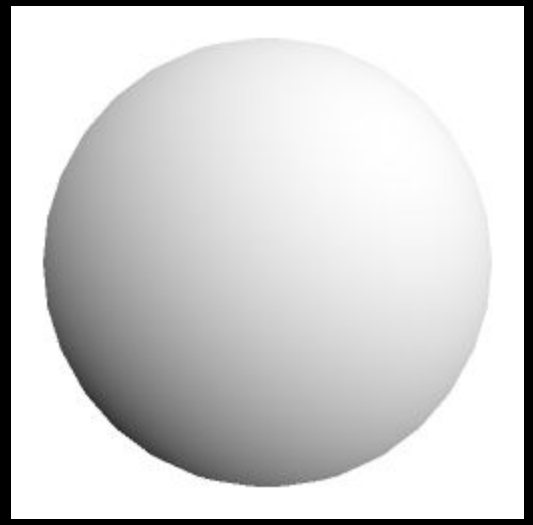
Wrapping the lighting around the sphere adds energy:



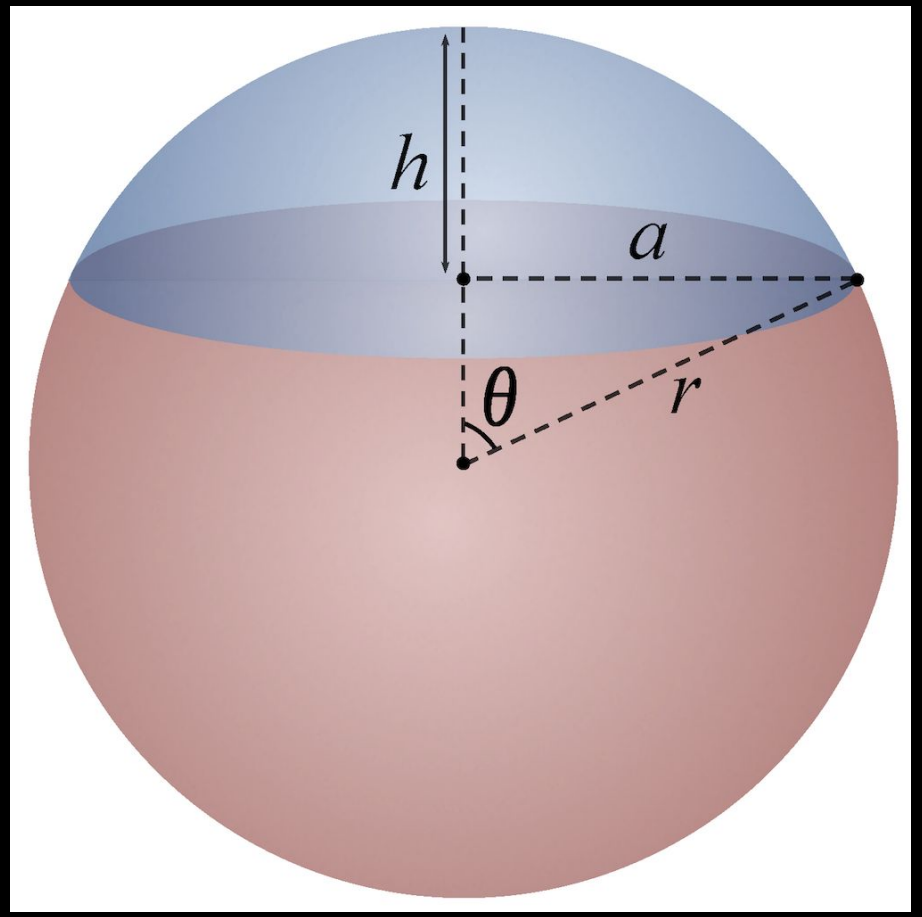
w = 0



w = 0.5



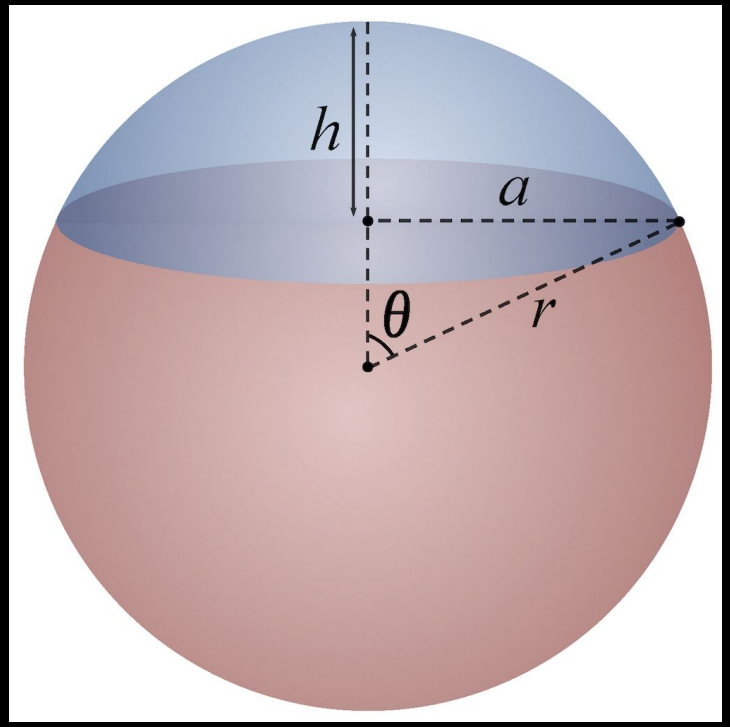
w = 1.0



$$A = 2\pi r(1 - \cos\theta)$$

Surface area of spherical cap

$$\cos\theta = -w$$



$$A = 2\pi(1 + w)$$

Surface area of the extent of wrapped lighting on a unit sphere

Surface area of hemisphere:

$$H = 2\pi$$

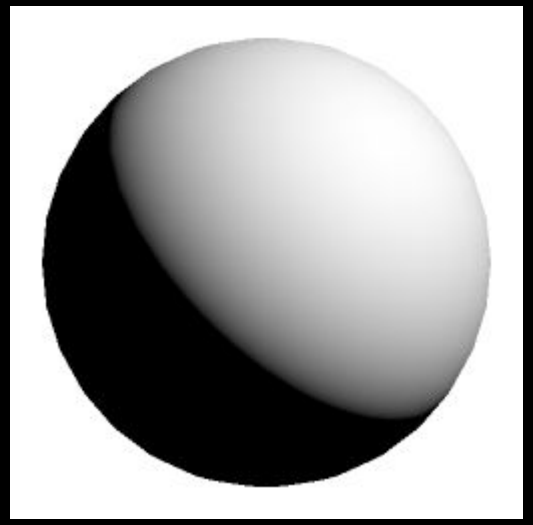
Surface area of the extent of wrapped lighting on a unit sphere

$$A = 2\pi(1 + w)$$

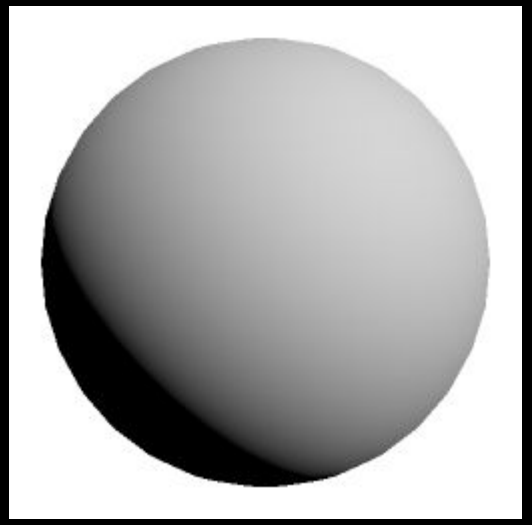
Normalisation factor:

$$\frac{A}{H} = \frac{1}{1 + w}$$

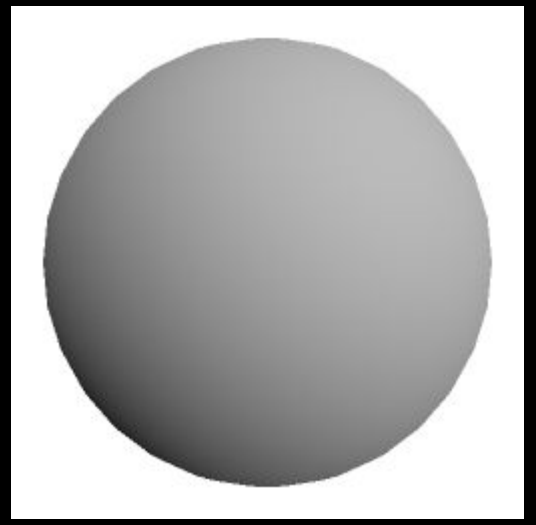
Normalised lighting:



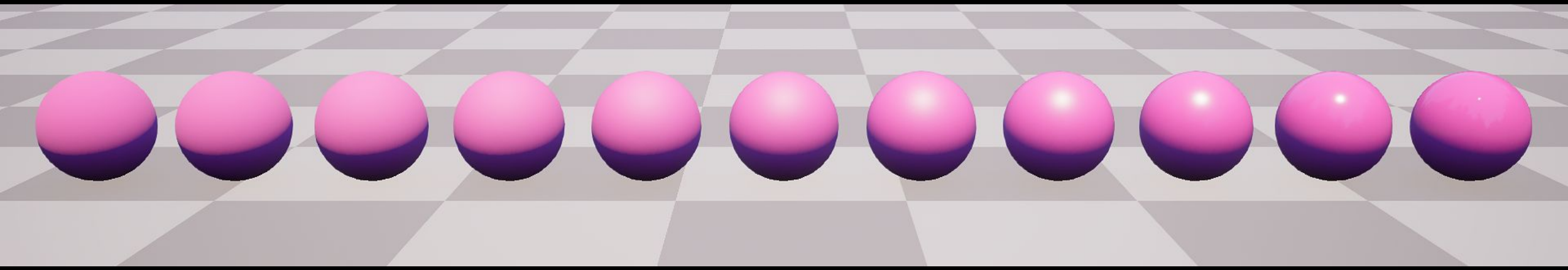
w = 0



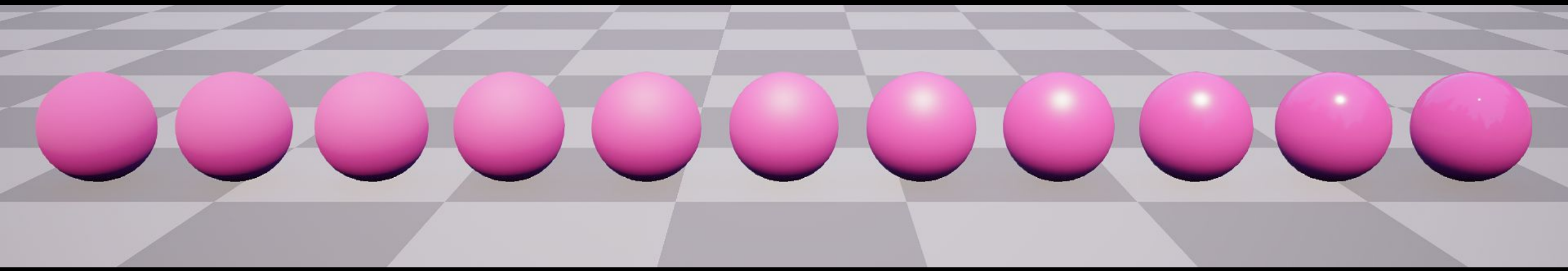
w = 0.5



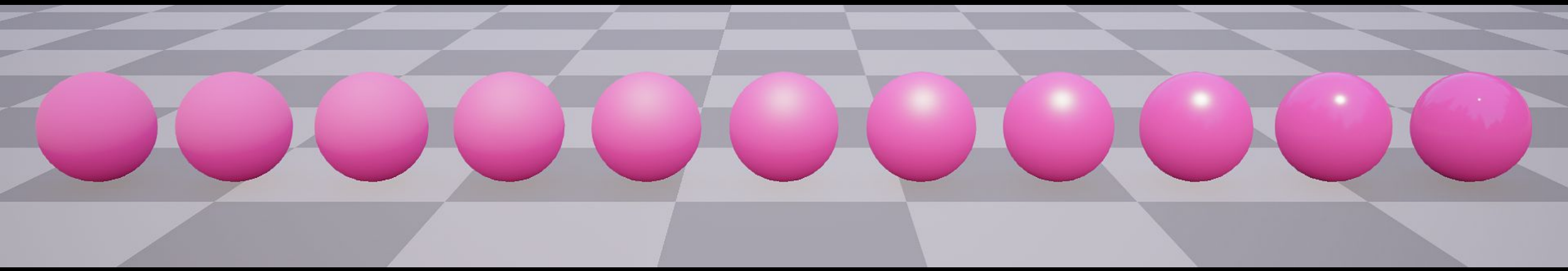
w = 1.0



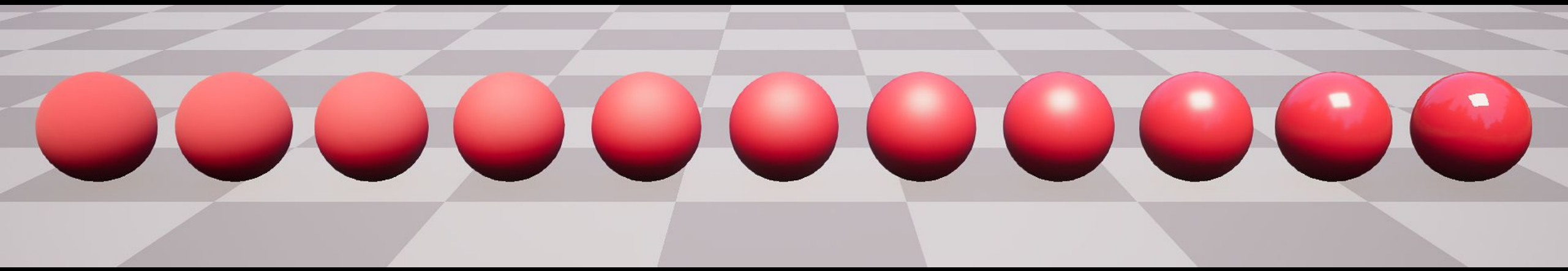
Multiscattering Diffuse, w = 0.0



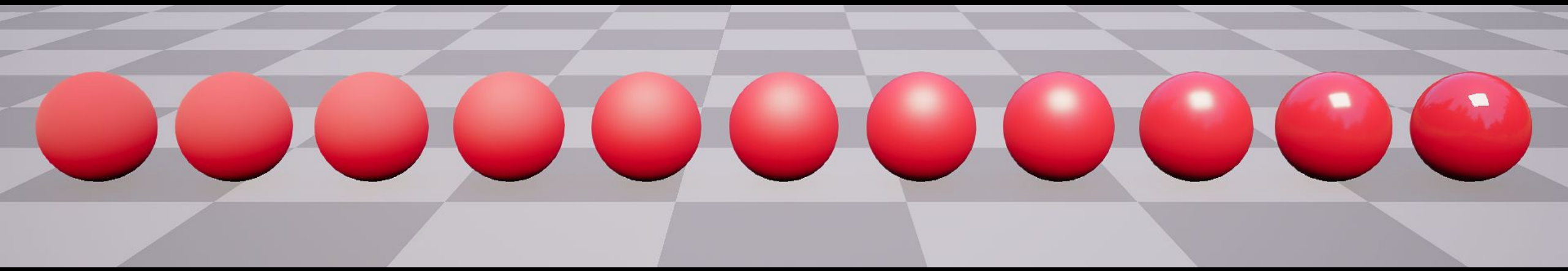
Multiscattering Diffuse, w = 0.5



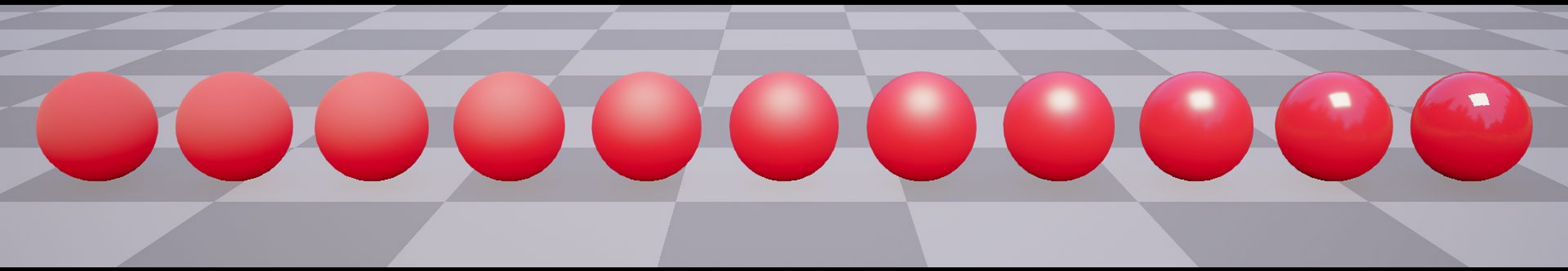
Multiscattering Diffuse, w = 1.0



Area Light, w = 0.0



Area Light, w = 0.5



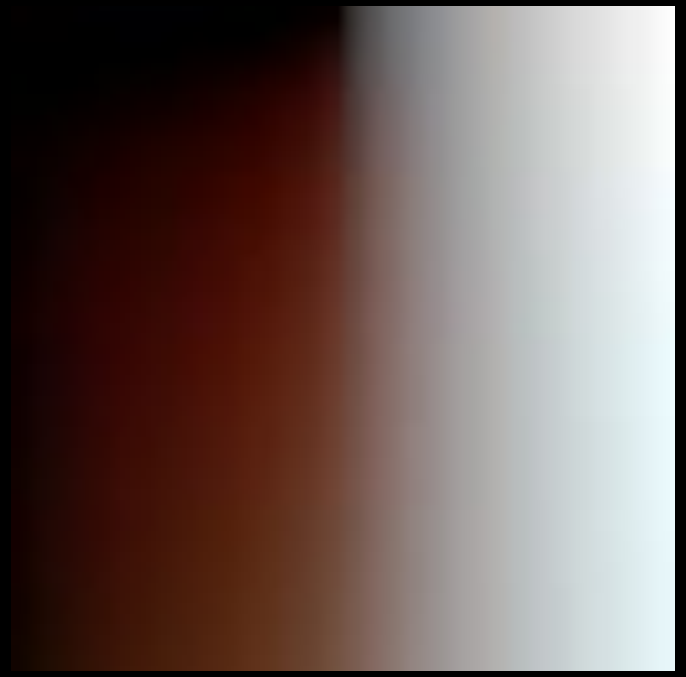
Area Light, w = 1.0

Problems:

1. No implementation for cloth
2. No implementation for multiscattering diffuse
3. No implementation for multiscattering specular
4. How to combine LTCs with wrapped diffuse
5. How to combine LTCs with pre-integrated skin scattering
6. No implementation of Marschner approximation for hair

Pre-integrated scattering:

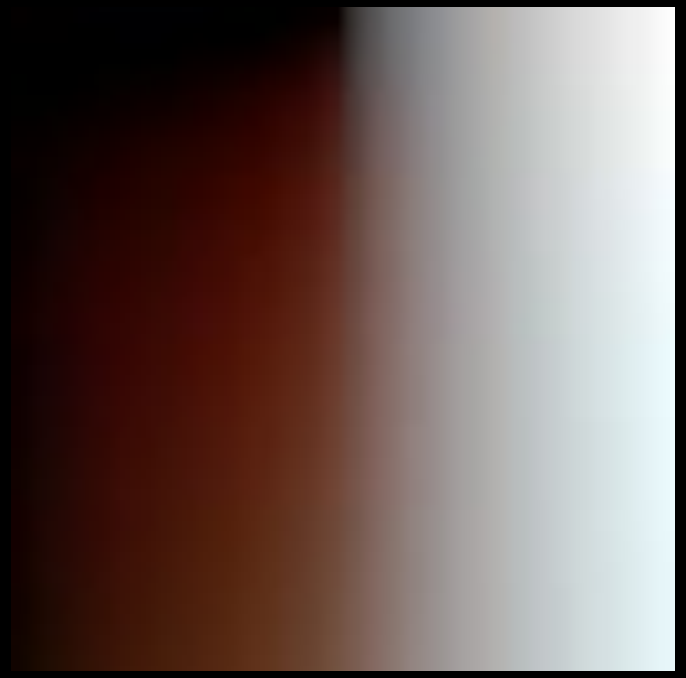
Integrate Lambert diffuse over spheres of different curvatures



[Penner11]

Pre-integrated scattering for multiscattering diffuse BRDF:

4D LUT required – adding view angle and roughness



[Penner11]

Separable Screen-Space Subsurface Scattering

[Jimenez12]


Important!
To do:

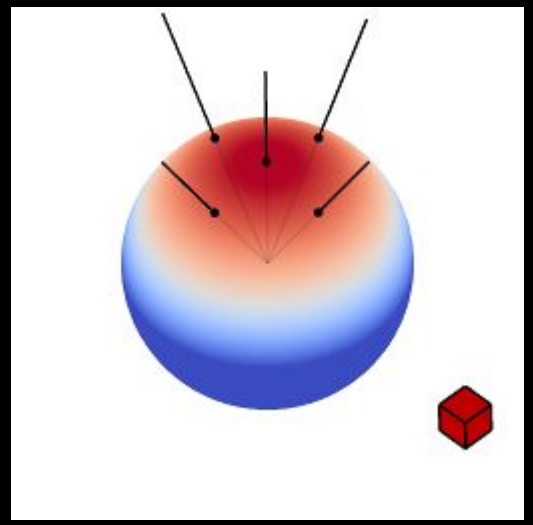
Problems:

1. No implementation for cloth
2. No implementation for multiscattering diffuse
3. No implementation for multiscattering specular
4. How to combine LTCs with wrapped diffuse
5. How to combine LTCs with pre-integrated skin scattering
6. No implementation of Marschner approximation for hair

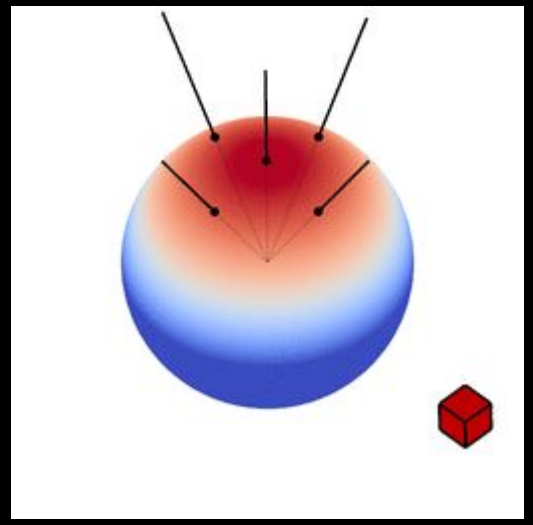

Important!
To do:

Remember this problem?

1. No multiscattering indirect specular
2. No multiscattering specular on hair
3. No multiscattering indirect diffuse
4. No multiscattering diffuse on skin
5. No multiscattering wrapped diffuse



We have a spherical harmonic projection for a clamped cosine distribution…



…and we have a mapping from an LTC to a clamped cosine distribution.

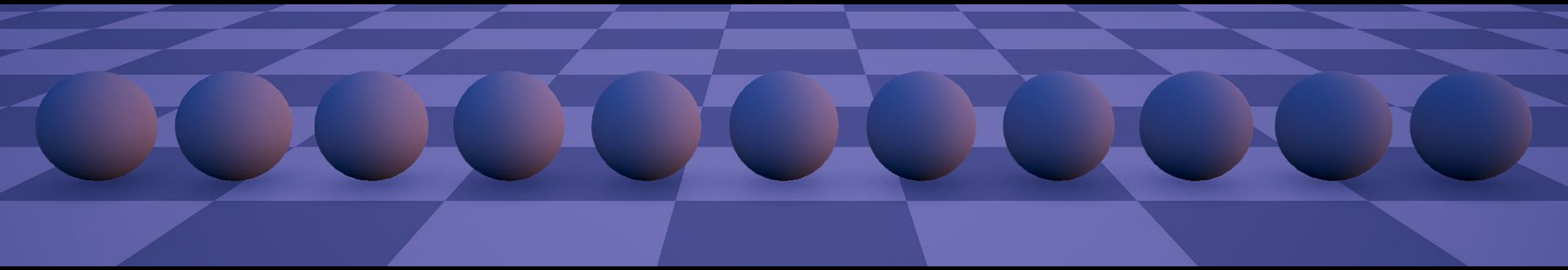
LTCs for Spherical Harmonics

1. Treat SH bands as “area light source”
2. Rotate SH bands by LTC inverse matrix
3. Evaluate SH with cosine lobe
4. Scale result by BRDF magnitude

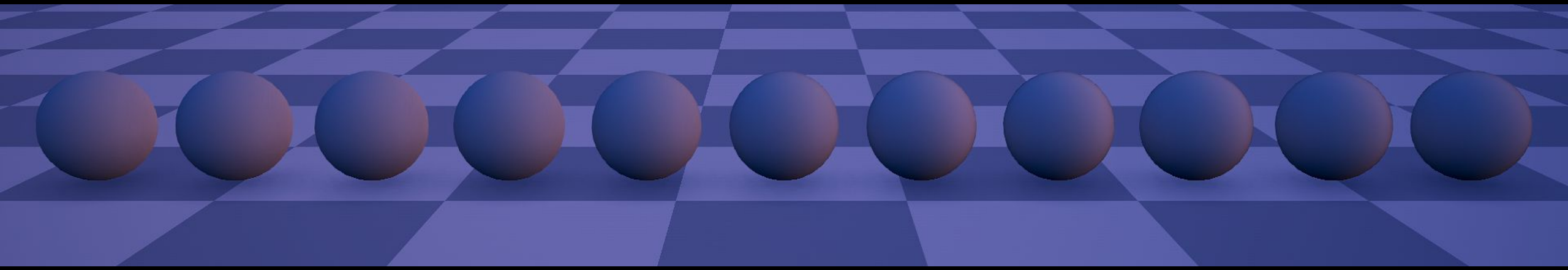
LTCs for Spherical Harmonics

1. Treat SH bands as “area light source”
2. Rotate cosine lobe by LTC inverse matrix
3. Evaluate SH with cosine lobe
4. Scale result by BRDF magnitude

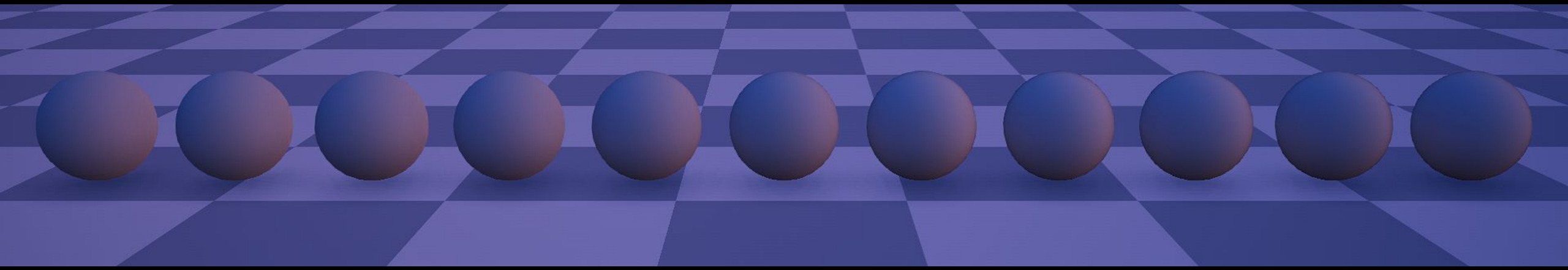
```
float2 uv = LTCTextureCoordinates(nDotV, smoothness);
float4 t1 = LTCMSDiffuseInvMatrixTexture.SampleLevel(Clamp, uv, 0);
float2 t2 = LTCMSDiffuseMagFresnelTexture.SampleLevel(Clamp, uv 0).rg;

// construct inverse matrix, mapping from BRDF to clamped cosine distribution
float3x3 Minv = LTCInverseMatrix(t1);

// construct orthonormal basis around N
float3 T1, T2;
T1 = normalize(V - N * dot(V, N));
T2 = normalize(cross(N, T1));

float3x3 R = float3x3(T1, T2, N);

float3 cosineLobeNormal = mul(float3(0.0f, 0.0f, 1.0f), mul(Minv, R));

SH3Coeffs shCosineLobe3 = SH3EvalCosineLobe(cosineLobeNormal);

// evaluate SH and scale by the BRDF magnitude
return SH3DotClamped(shR, shG, shB, shCosineLobe3) * t2.x;
```



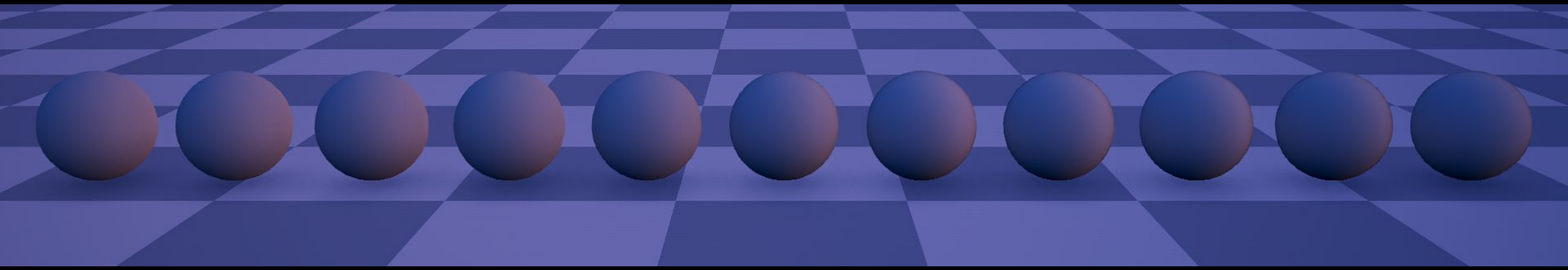
Lambert Indirect Diffuse



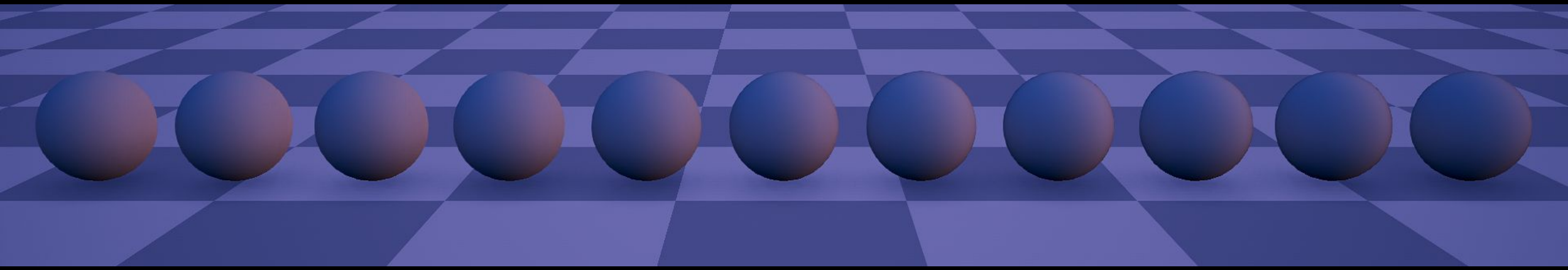
Multiscattering BRDF Indirect Diffuse



Ground
Truth*



Multiscattering BRDF Indirect Diffuse



Scaling by BRDF Magnitude Only

Problems Solved:

1. Multiscattering diffuse
   - Direct, indirect and wrapped lighting
2. Multiscattering specular
   - Direct and indirect lighting
3. Area lights
   - Multiscattering diffuse and specular
   - Cloth
   - Wrapped lighting

Problems Remaining:

1. Hair
   - Area lighting
   - Multiscattering diffuse and specular
2. Skin
   - Diffuse area lighting
   - Multiscattering diffuse

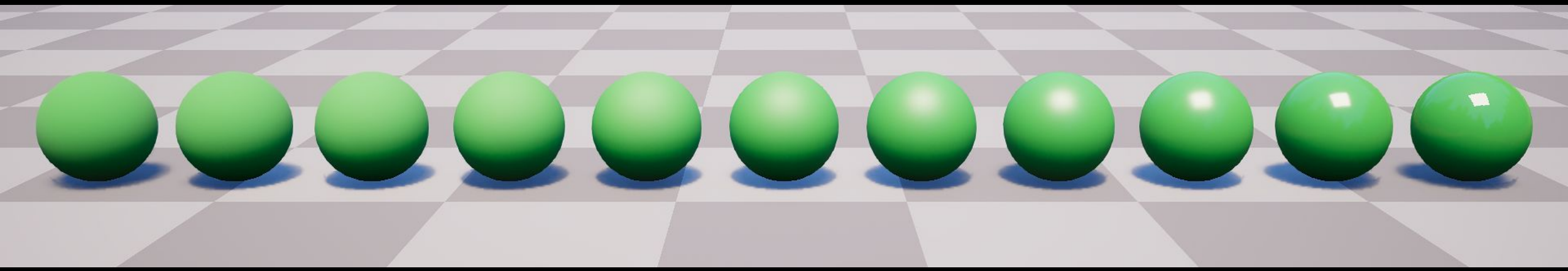
What have we learned?

Takeaway #1: Source code is invaluable

Takeaway #2: Separate insight from implementation

Takeaway #3: Build on successful existing work

Takeaway #4: Never underestimate implementation time



Epilogue: Implementation Details for Area Lights

Goals:

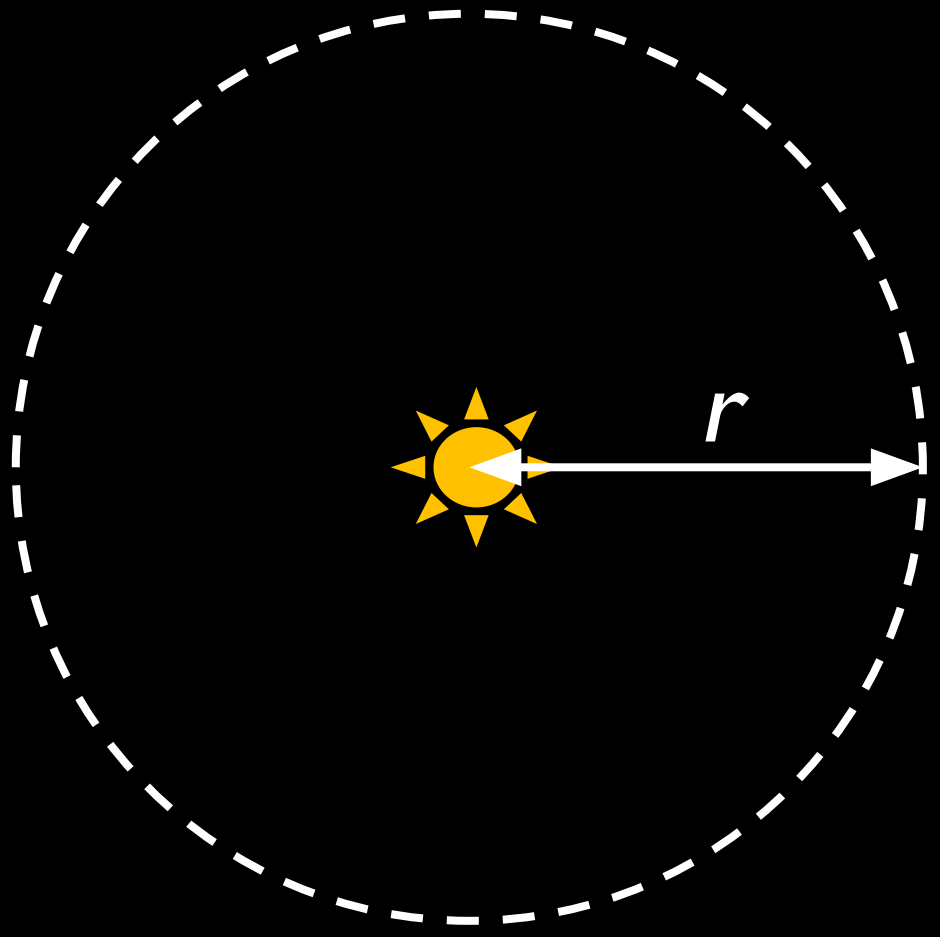
1. Control over performance
   - Radius falloff
   - Cone angle falloff

2. Fall back to point lights
   - Omni or spot light

Goals:

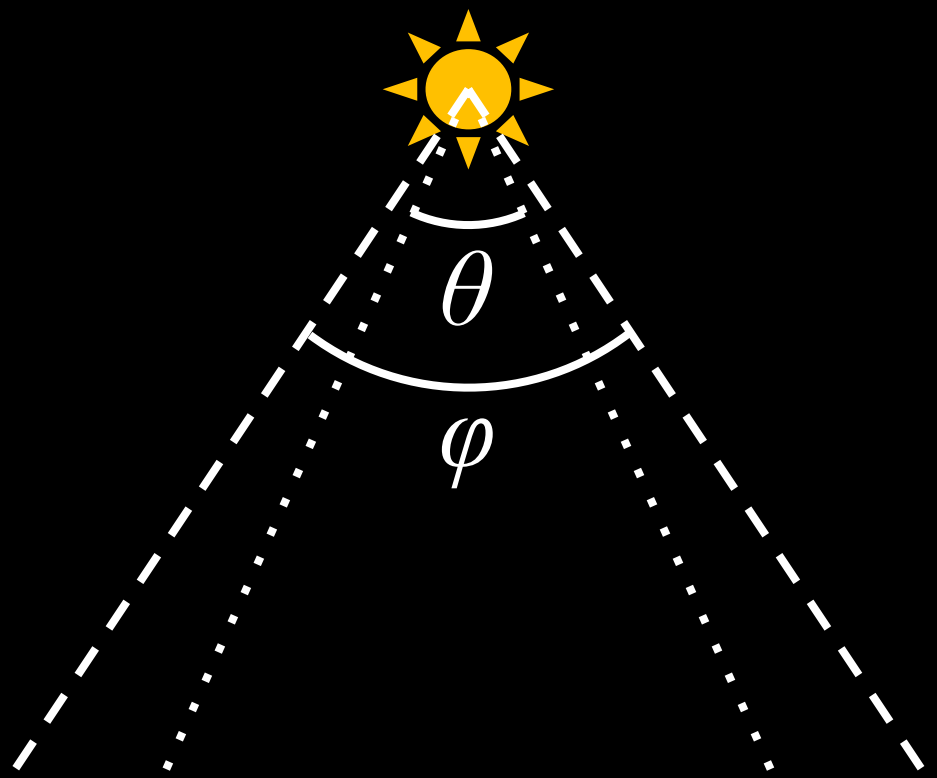
1. Control over performance
   - Radius falloff
   - Cone angle falloff

2. Fall back to point lights
   - Omni or spot light



$$\mathbf{saturate}(1 - \frac{d^4}{r^4})^2$$

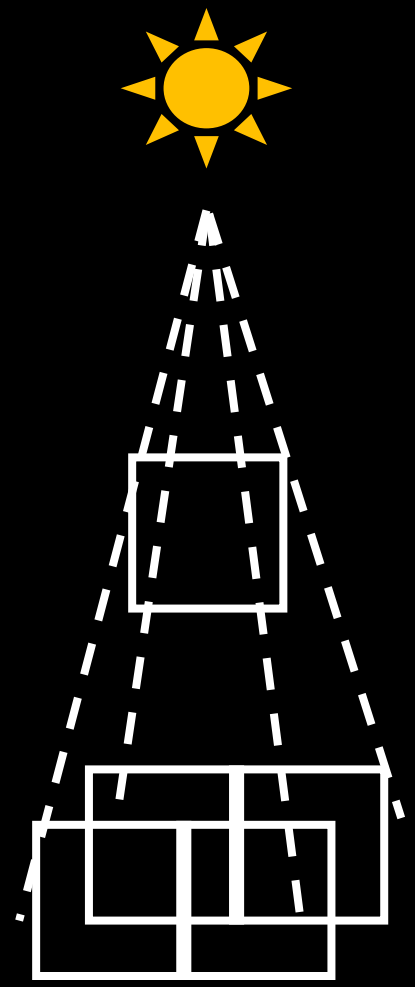
Windowing Function [Karis13]



θ = *inner angle*
φ = *outer angle*

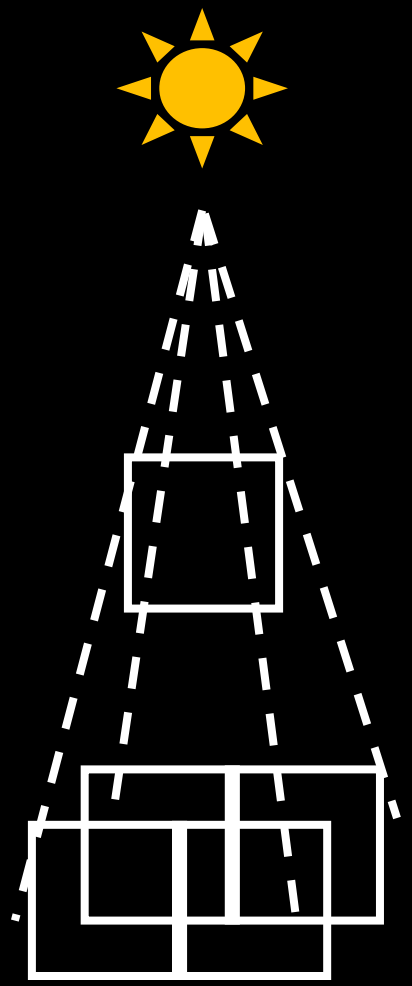
$$\mathbf{saturate}\left(\frac{\cos\frac{1}{2}\alpha - \cos\frac{1}{2}\phi}{\cos\frac{1}{2}\theta - \cos\frac{1}{2}\phi}\right)$$

Cone Falloff



$$\frac{1}{r^2}$$

Inverse Square Law



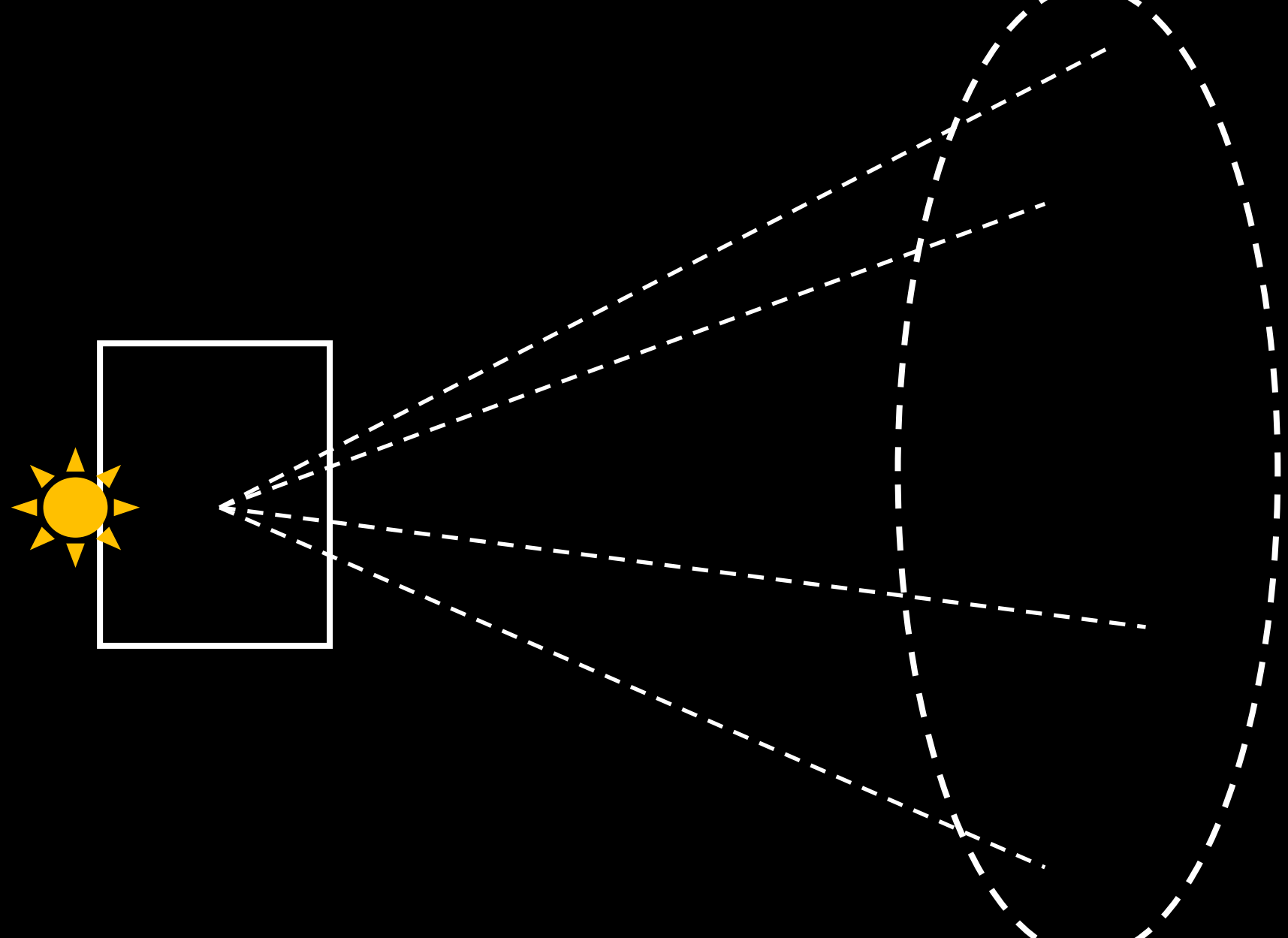
$$\frac{1}{r^2}$$

Accounted for in integration over the hemisphere

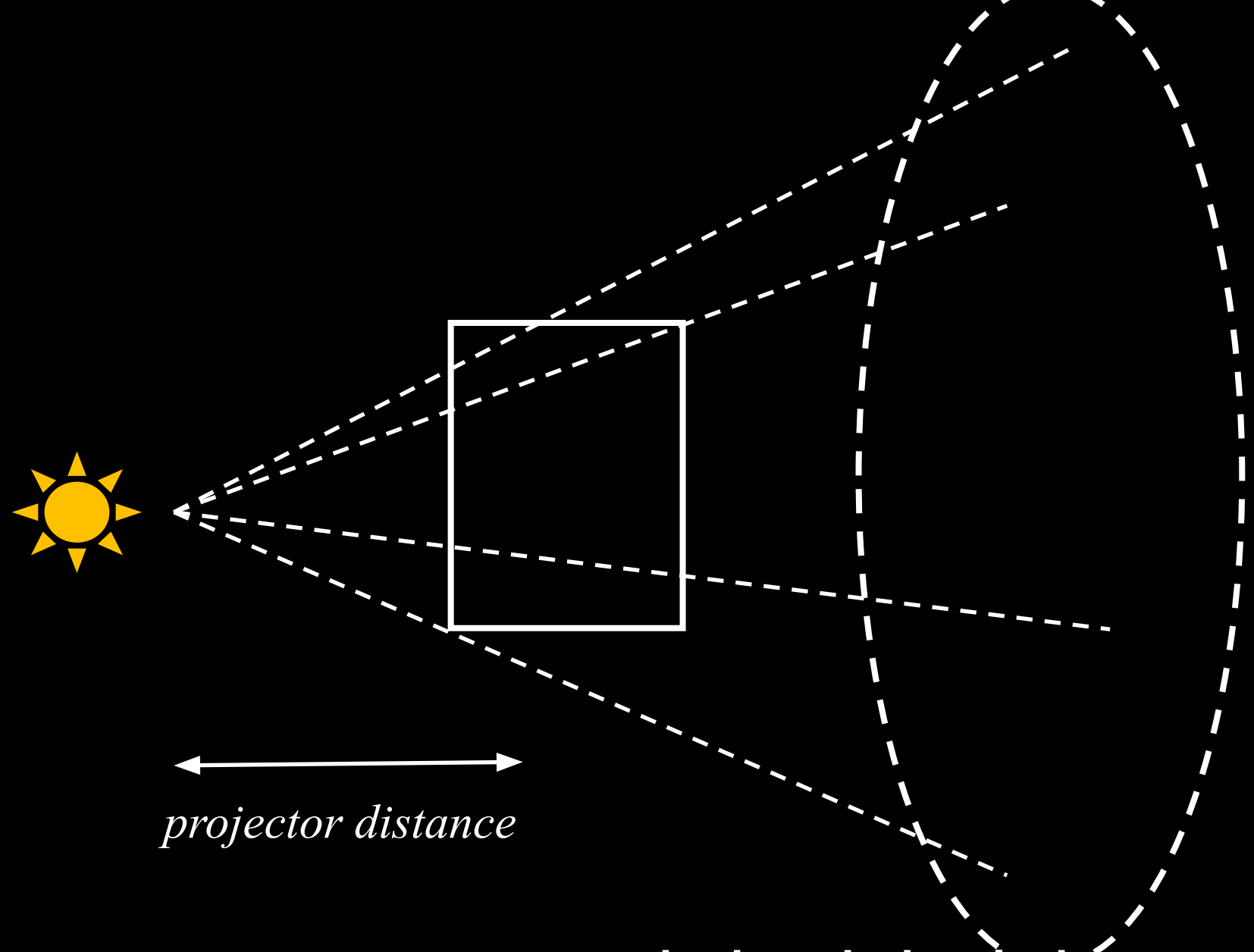
Goals:

1. Control over performance
   - Radius falloff
   - Cone angle falloff

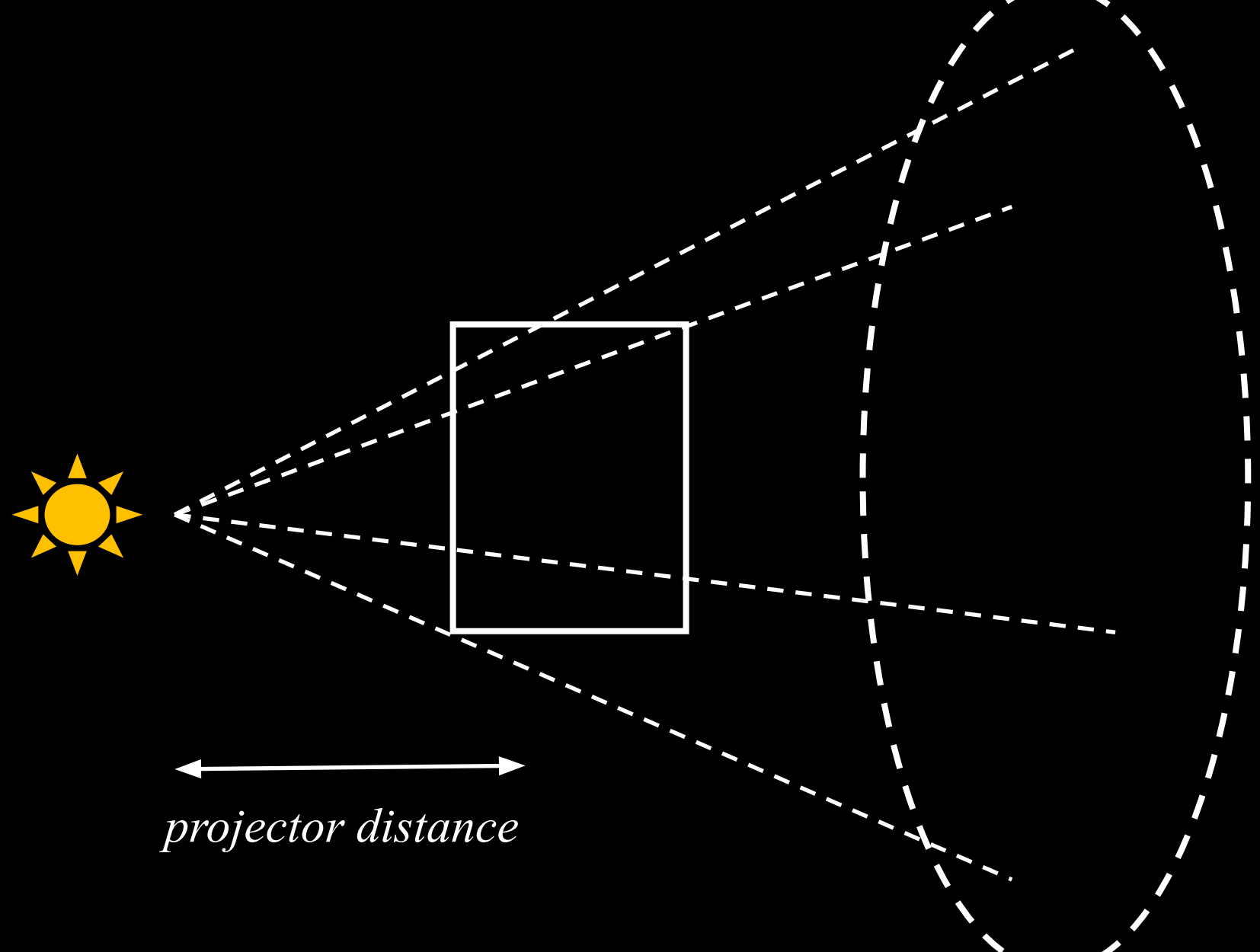
2. Fall back to point lights
   - Omni or spot light



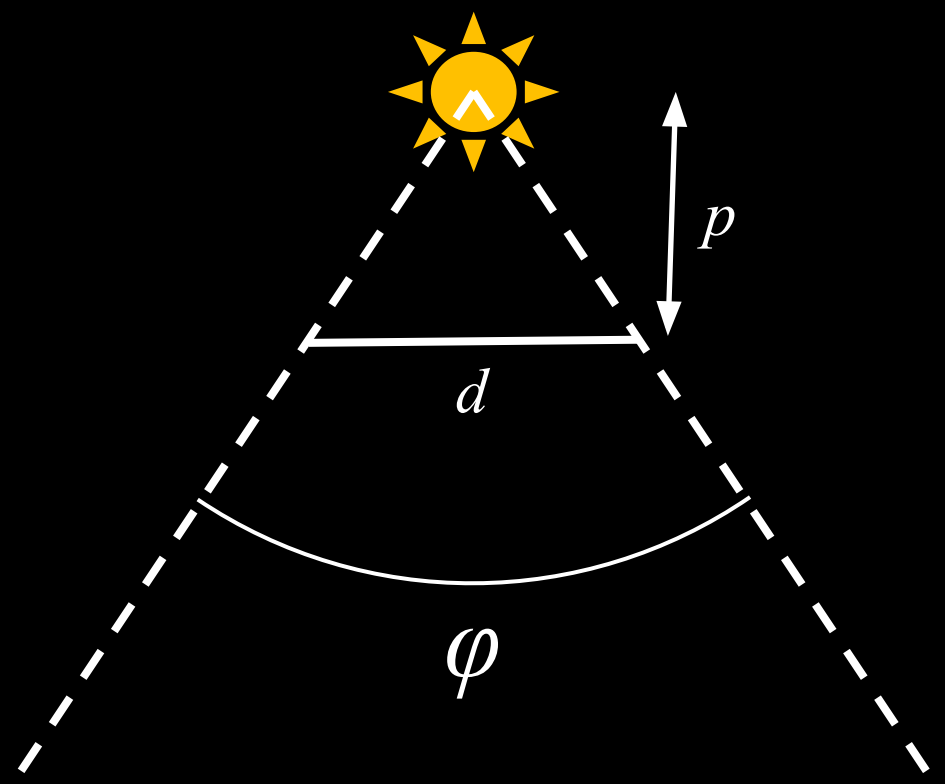
Quad light falls back to spot light



Move projector position behind the light source



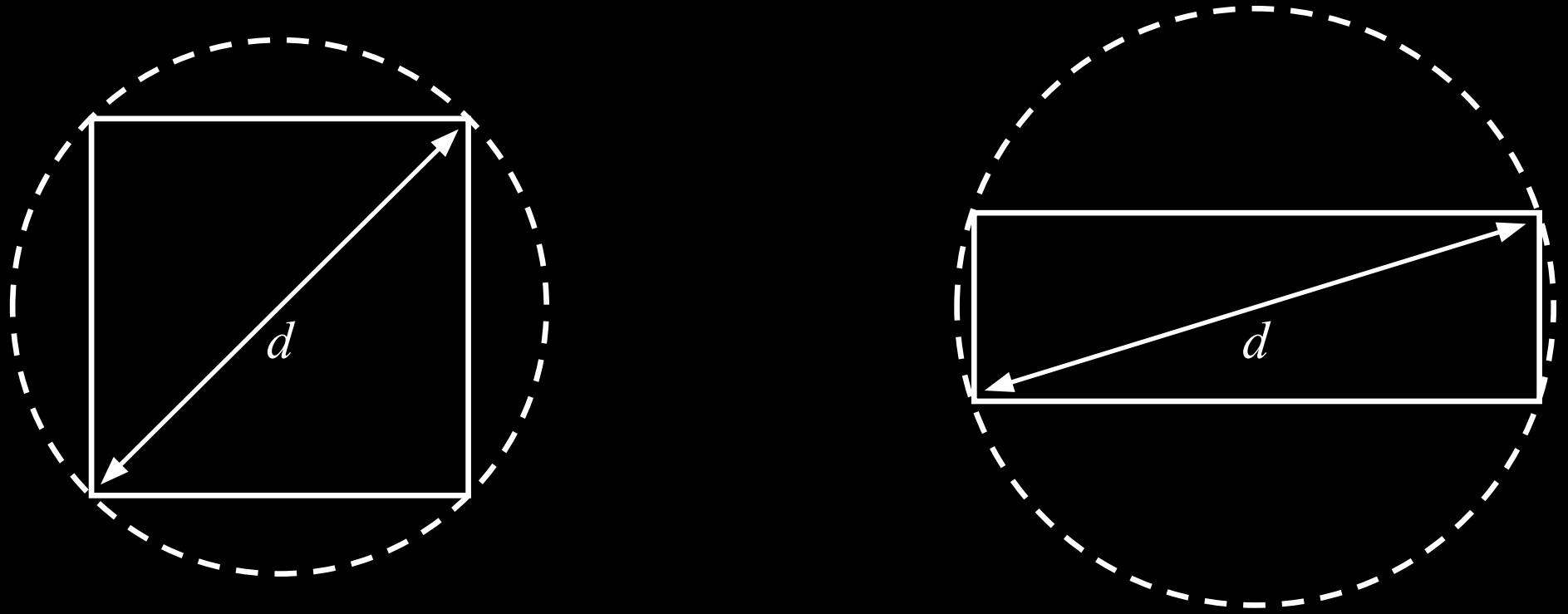
Use cone apex for cone angle fall off for spot and quad light



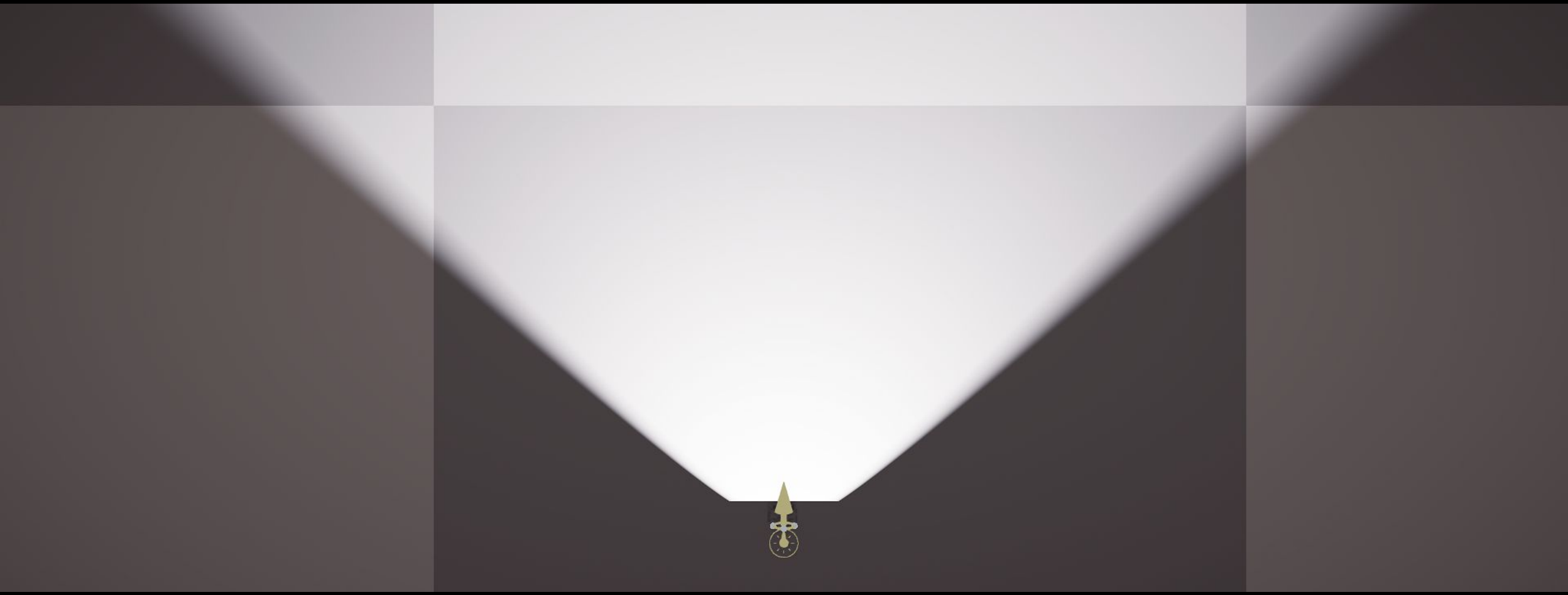
$$p = \frac{\frac{1}{2}d}{\tan\frac{1}{2}\phi}$$

Calculate the projector distance p from outer angle and light source diameter

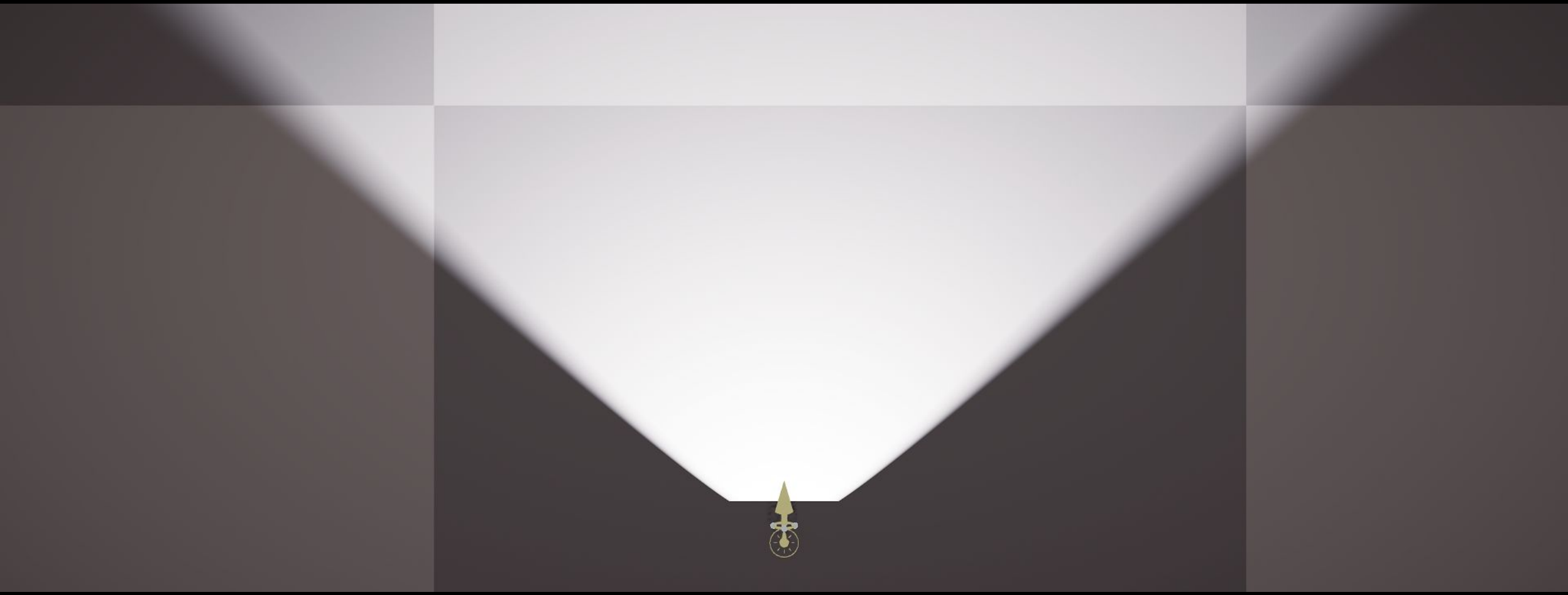
Cone fits around quad light source



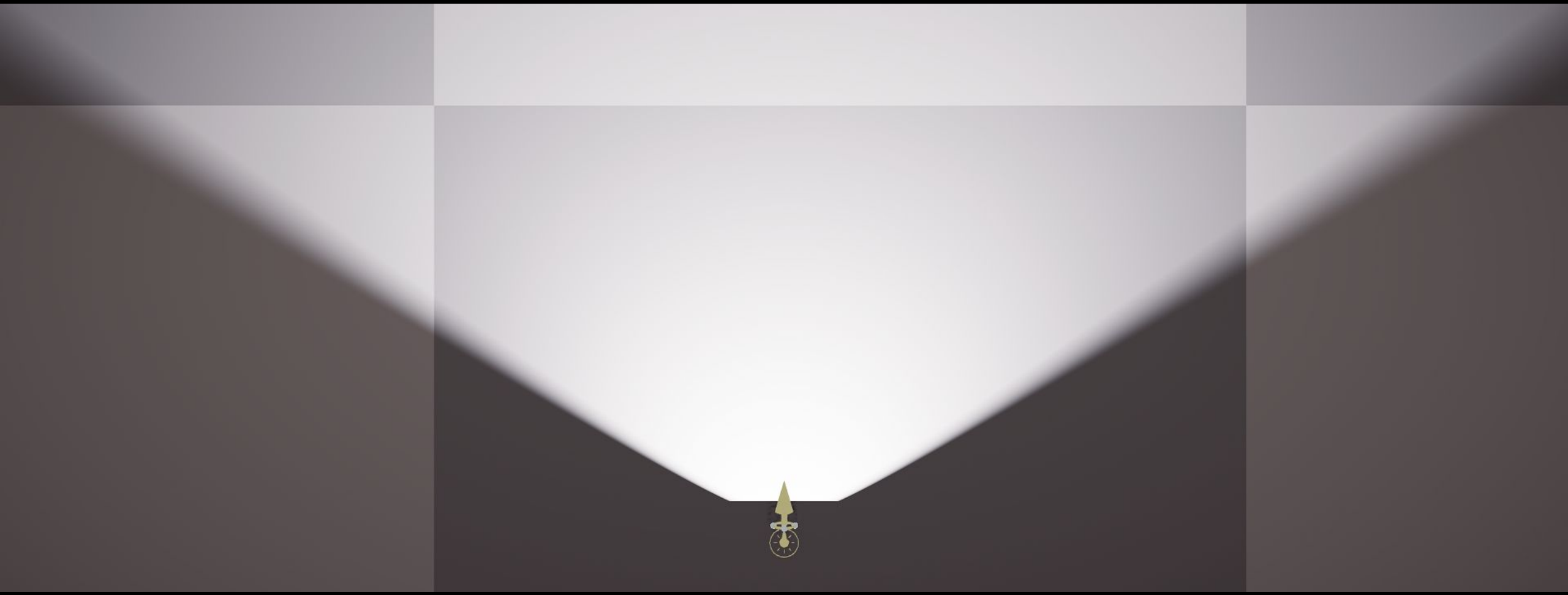
Worse fit for non-square light source shapes



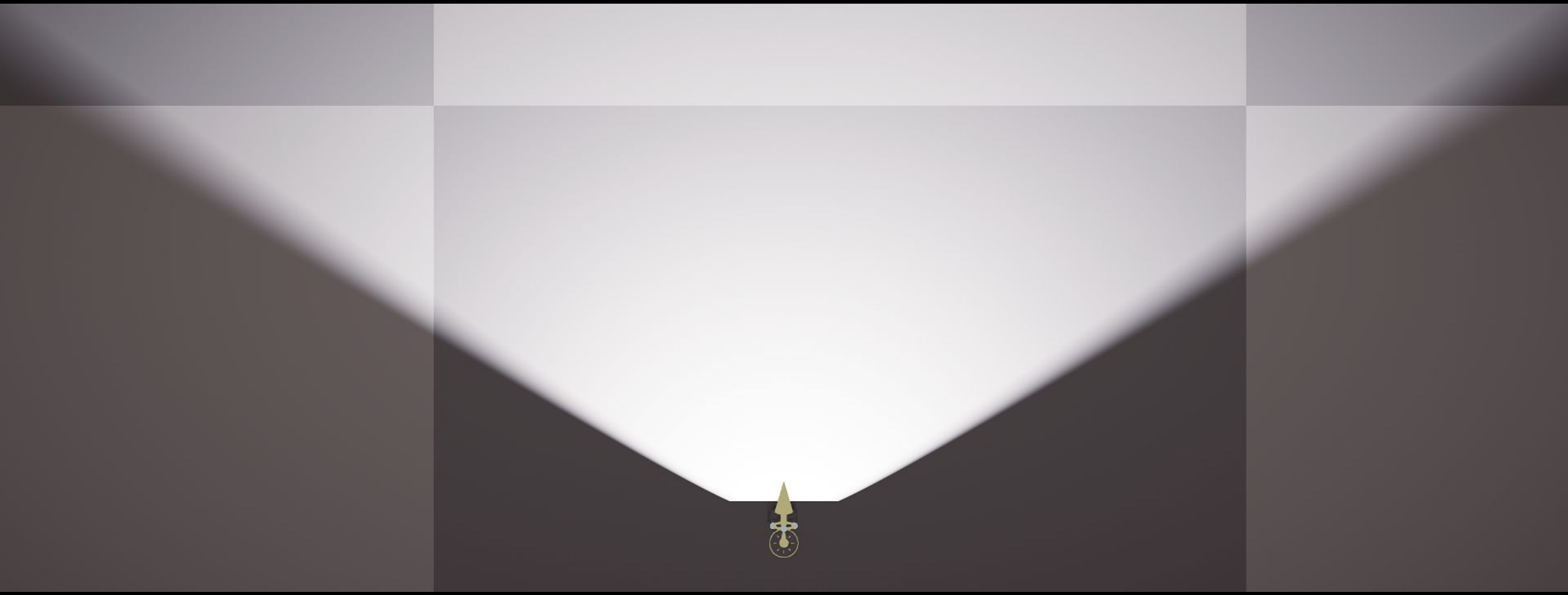
Quad light, 100° outer angle, 90° inner angle



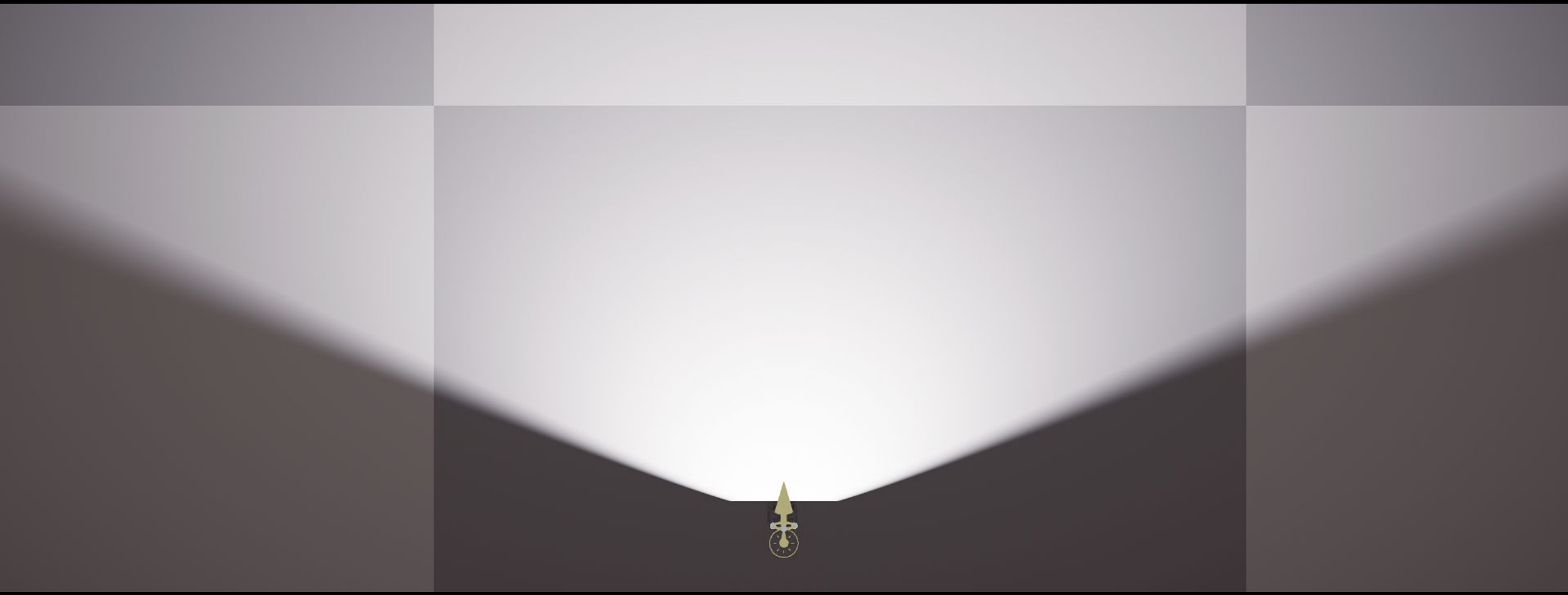
Fall back, 100° outer angle, 90° inner angle



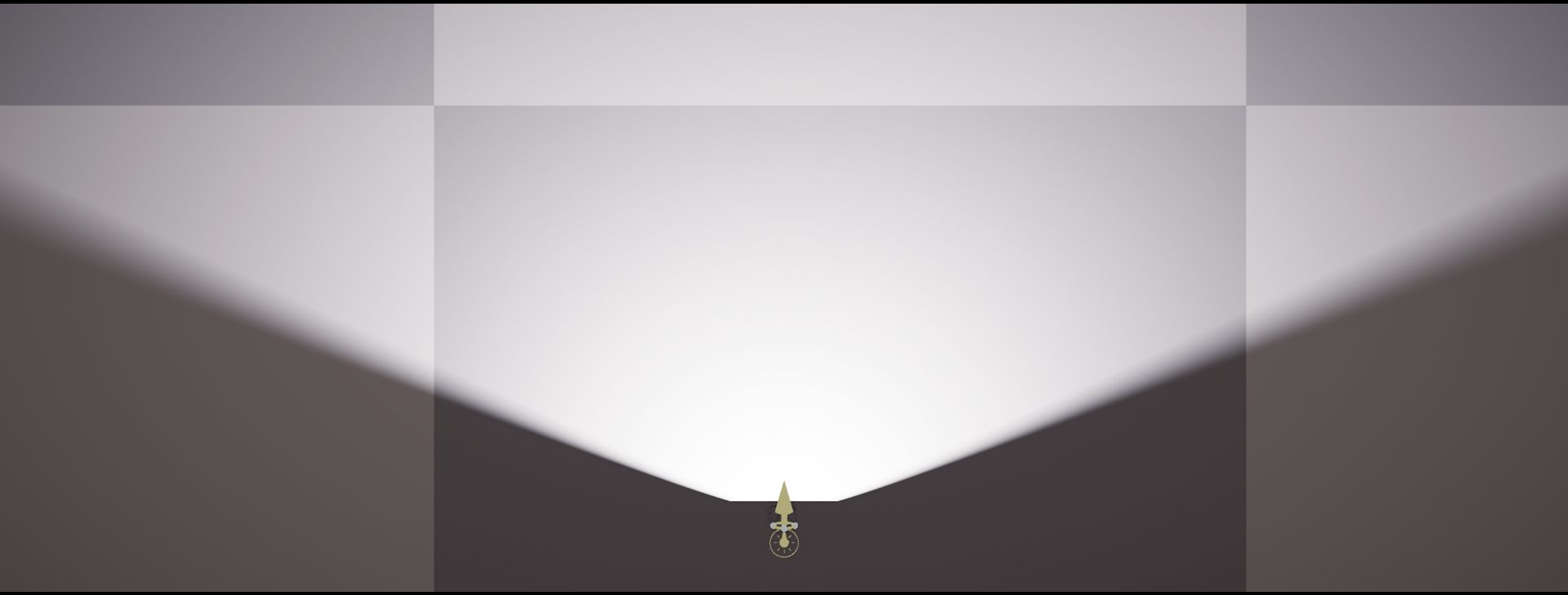
Quad light, 120° outer angle, 110° inner angle



Fall back, 120° outer angle, 110° inner angle



Quad light, 140° outer angle, 130° inner angle



Fall back, 140° outer angle, 130° inner angle


Thank You!

References

[Ashikhmin00] *A Microfaced-based BRDF Generator*, Michael Ashikhmin et. al., SIGGRAPH 2000

[Burley12] *Physically Based Shading at Disney*, Brent Burley, SIGGRAPH 2012

[Chan18] *Material Advances in Call of Duty: WWII*, Danny Chan, SIGGRAPH 2018

[FdezAgüera19] *A Multiple-Scattering Microfacet Model for Real-Time Image-based Lighting*, Carmelo J. Fdez-Agüera, JCGT Vol. 8, No. 1

[Heitz16a] *Multiple-Scattering Microfacet BSDFs with the Smith Model*, Eric Heitz et. al., SIGGRAPH 2016

[Heitz16b] *Real-Time Polygonal-Light Shading with Linearly Transformed Cosines*, Eric Heitz et. al., SIGGRAPH 2016

[Hill16] *LTC Fresnel Approximation*, Stephen Hill, Tech Report, https://blog.selfshadow.com/publications/s2016-advances/s2016_ltc_fresnel.pdf

[Jimenez12] *Separable Subsurface Scattering and Photorealistic Eyes Rendering*, Jorge Jimenez, SIGGRAPH 2012

[Karis13] *Real Shading in Unreal Engine 4*, Brian Karis, SIGGRAPH 2013

[Kulla17] *Revisiting Physically Based Shading at Imageworks*, Christopher Kulla & Alejandro Conty, SIGGRAPH 2017

[Lagarde14] *Moving Frostbite to PBR*, Sébastien Lagarde & Charles De Rousiers, SIGGRAPH 2014

[Penner11] *Pre-Integrated Skin Shading*, Eric Penner, SIGGRAPH 2011

[Turquin19] *Practical Multiple Scattering Compensation for Microfacet Models*, Emmanuel Turquin, Tech Report, http://blog.selfshadow.com/publications/turquin/ms_comp_final.pdf