Illumination 101

Pixar Animation Studios

- Ryusuke Villemin
  - MU Lighting System Overview
  - Direct Lighting Solution
- Christophe Hery
  - Luminaire examples
  - Beckmann BRDF

Tuesday, July 23, 13

Tuesday, July 23, 13

BEFORE PHYSICALLY BASED LIGHTING

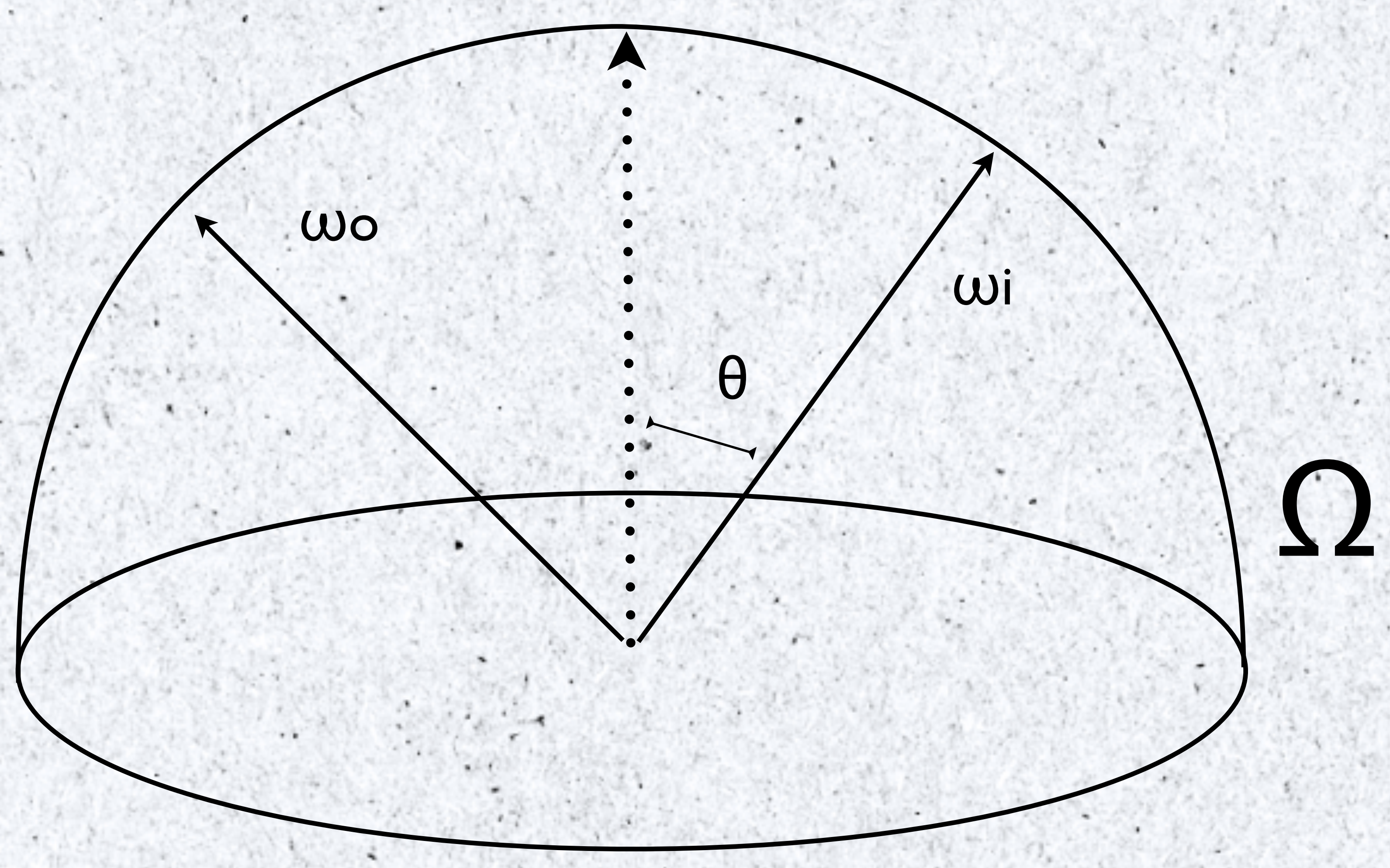
- REYES with prman
  - simple point lights, no energy, no falloff
  - non physical, non normalized BRDFs
  - deep shadow maps
- it was fine because there was no secondary effect

no concept of energy, just color and intensity
light 100, albedo 1 or light 1, albedo 100, same thing!

- Solving the rendering equation

$$L(x,\omega_o) = \int_{\Omega} f(x,\omega_i,\omega_o)L(x,\omega_i)cos(\theta)d\omega$$

- L(x, ωo) : outgoing radiance
- L(x, ωi) : incoming radiance
- f(x, ωi, ωo) : BRDF



recursion means conditions on L and f not to diverge to infinity

- 2 parts working in tandem:
  - physically correct lights emitting energy in the scene
  - physically correct BRDFs bouncing energy in the scene

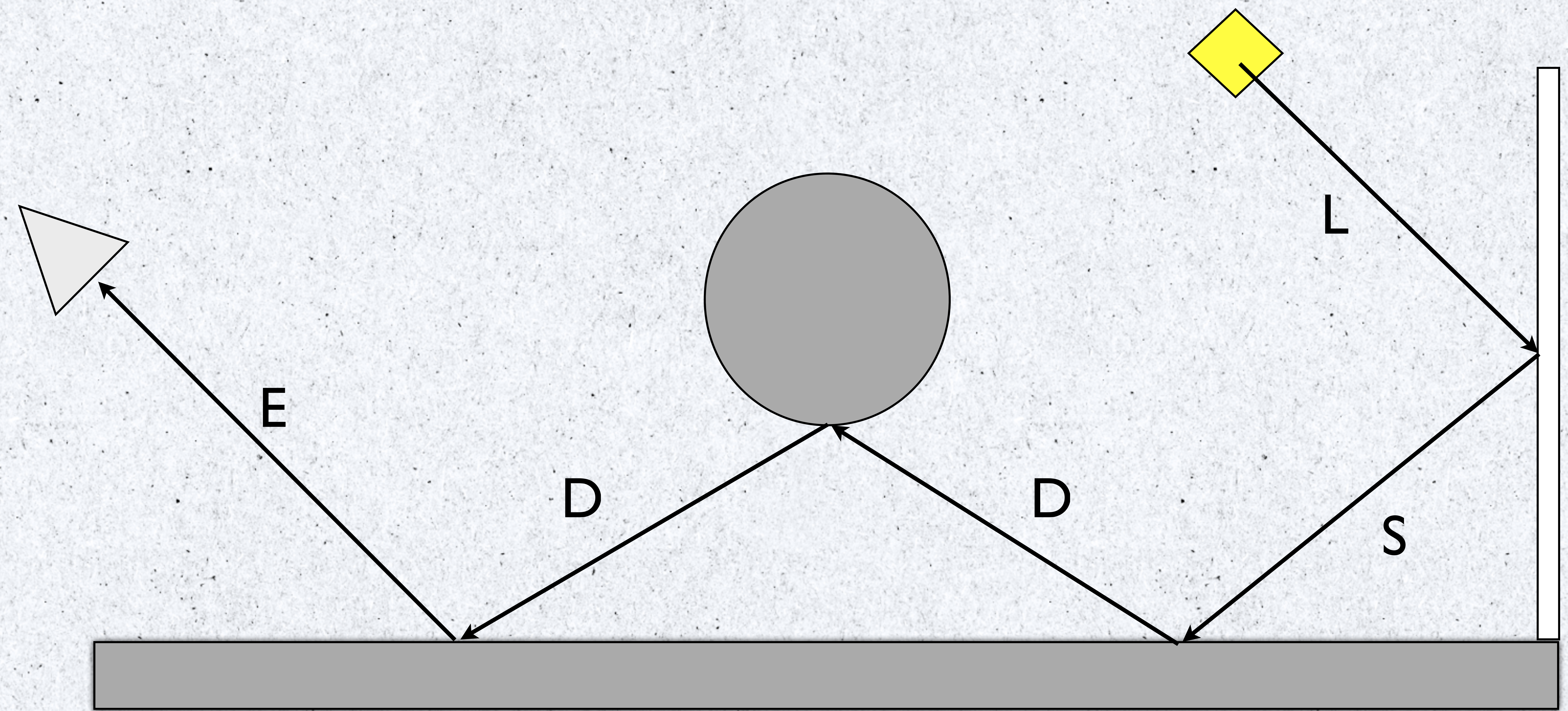
Tuesday, July 23, 13

we can't and shouldn't decouple those 2
keep the same results between a rect area light and a glowing wall with same intensity

- Break down the rendering equation, each part will be solved by different coshaders called integrators

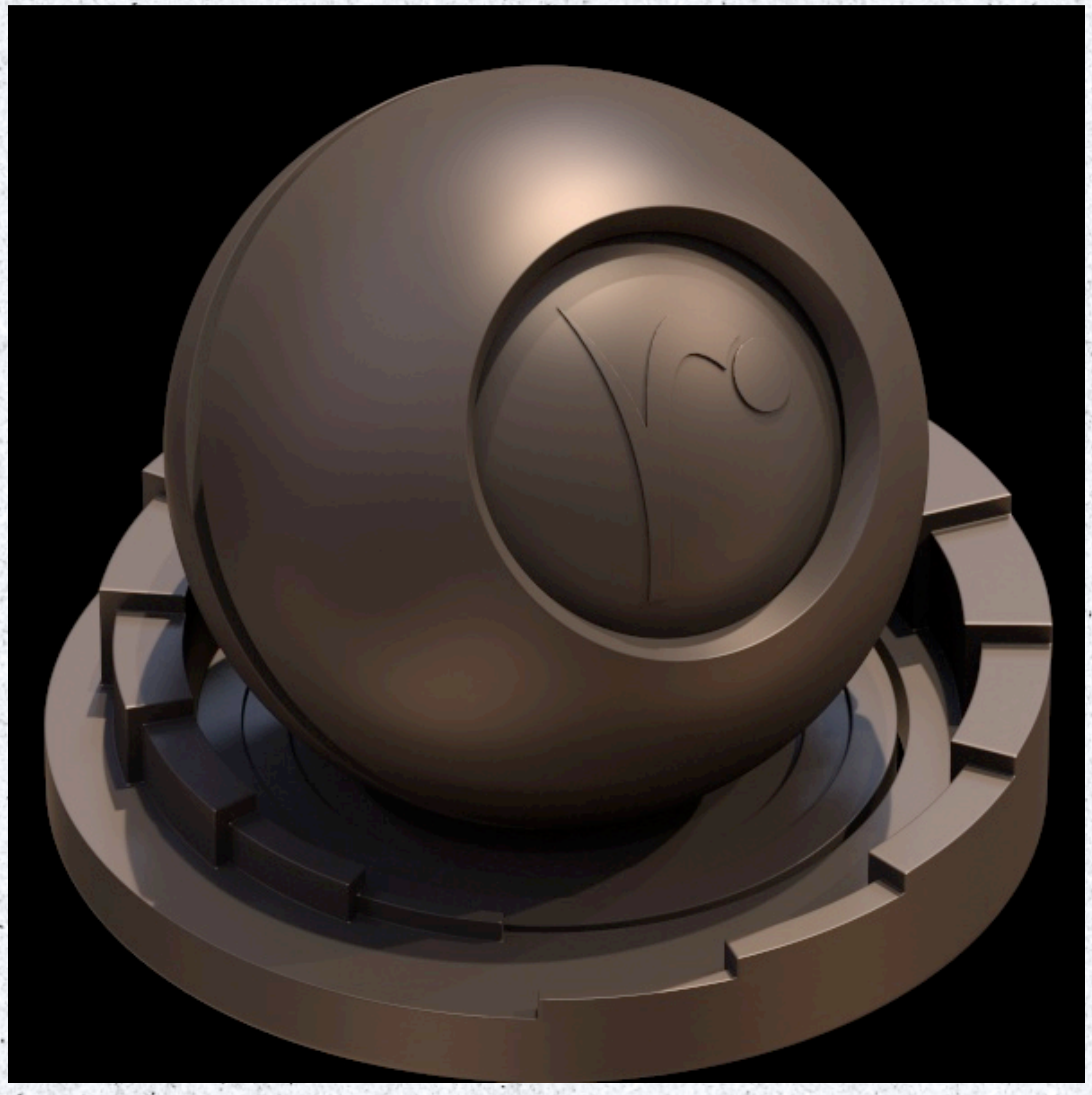
$$L(x,\omega_o) = \int_\Omega f(x,\omega_i,\omega_o) L_{direct}(x,\omega_i) cos(\theta) d\omega$$
$$+ \int_\Omega f_{diffuse}(x,\omega_i,\omega_o) L_{indirect}(x,\omega_i) cos(\theta) d\omega$$
$$+ \int_\Omega f_{specular}(x,\omega_i,\omega_o) L_{indirect}(x,\omega_i) cos(\theta) d\omega$$

direct <-> indirect to get rid of the “recursion”
diffuse spec because we use different method of integration taking advantage of the renderer



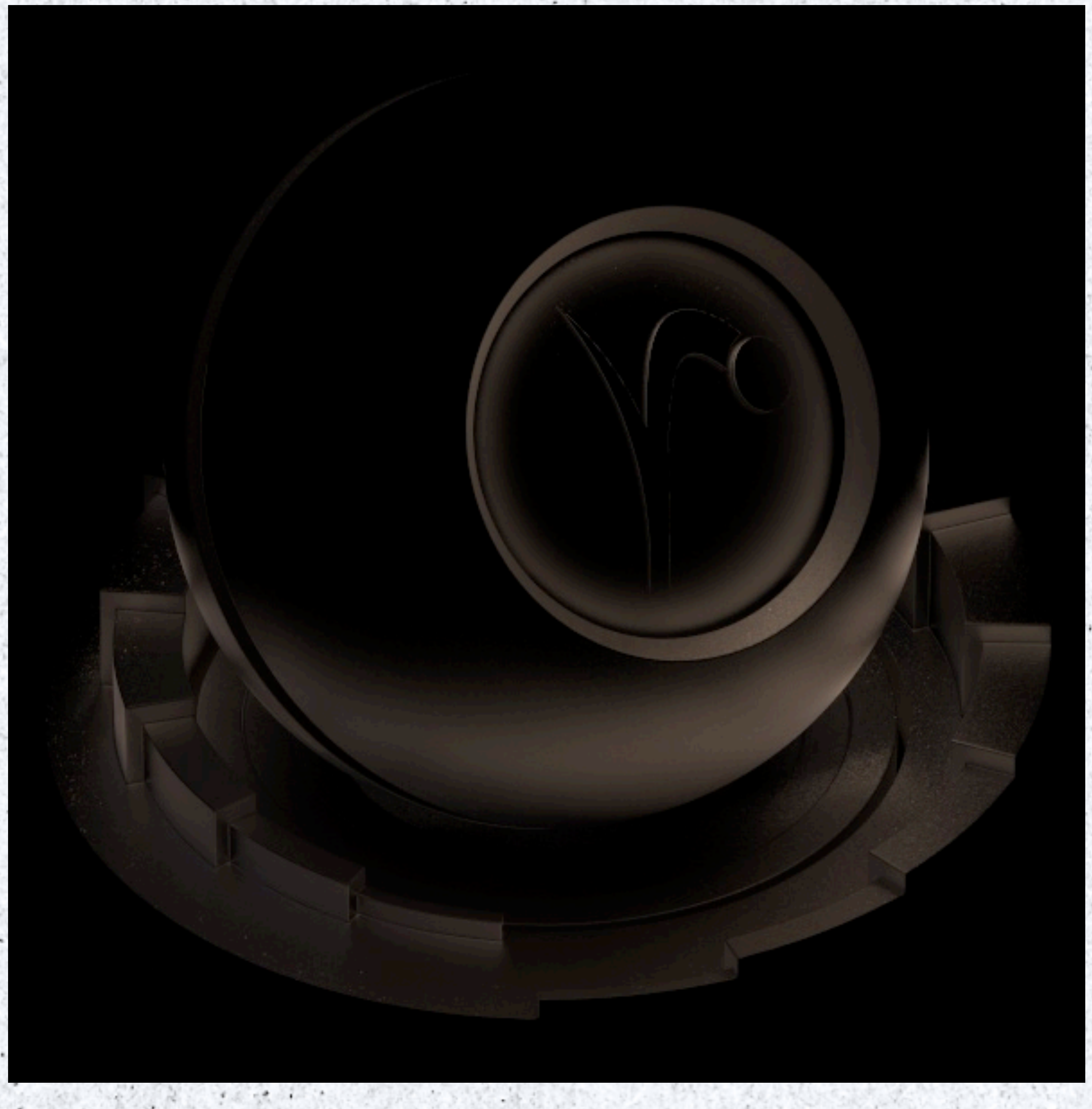
using heckbert's notation

- directLighting integrator solves L{D,S}E path



- indirectLighting integrator solves LDD*E path

- reflection integrator solves LS{S*,D*}E path



- indirectDiffuse integrator uses
  - irradiance caching
  - radiosity cache
  - photon cache
- reflection integrator uses
  - caustic cache
  - radiosity cache (optional, in that case we give up secondary specular effects)

no caustic path
radiosity cache is prman specific

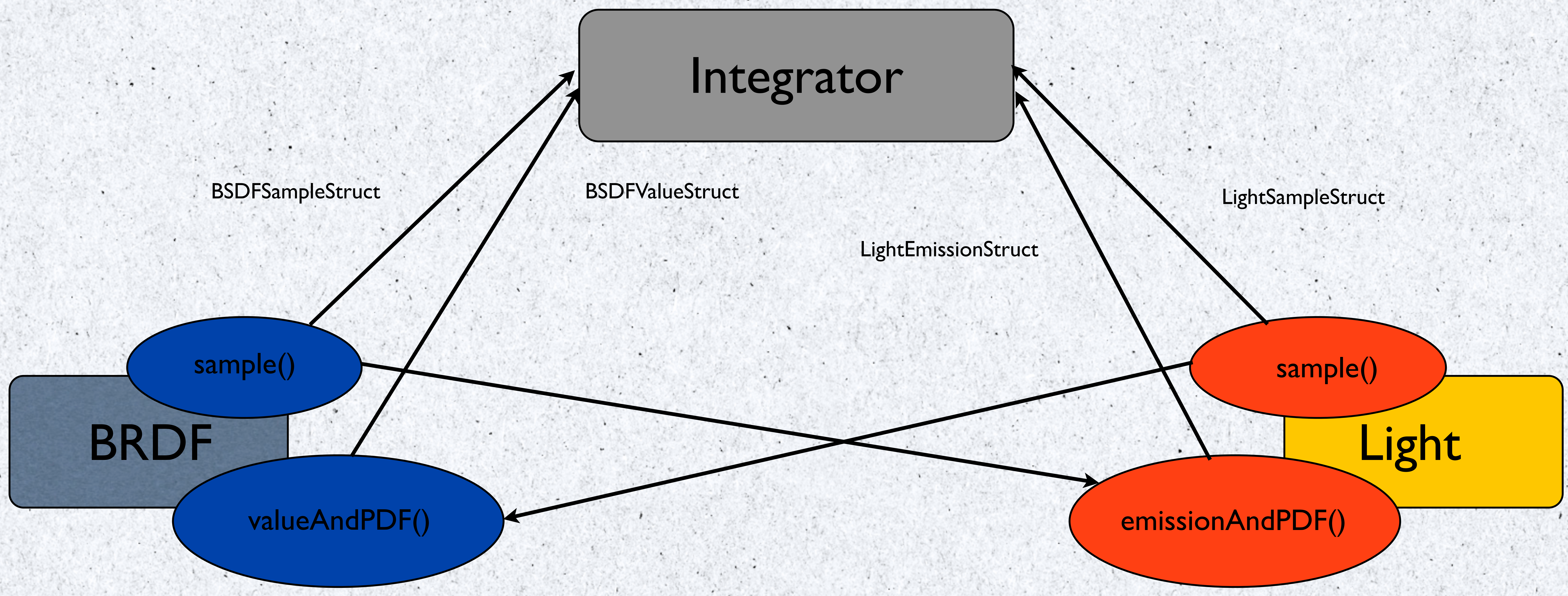
- has 3 main parts
  - Light coshader
  - BRDF coshader
  - Integrator coshader

- Multiple importance sampling
- Filtered importance sampling
- Importance resampling to reduce number of shadow rays
- Control variates

Tuesday, July 23, 13

brute force is good, always work
but we want to speedup the process
principal piece -> MIS

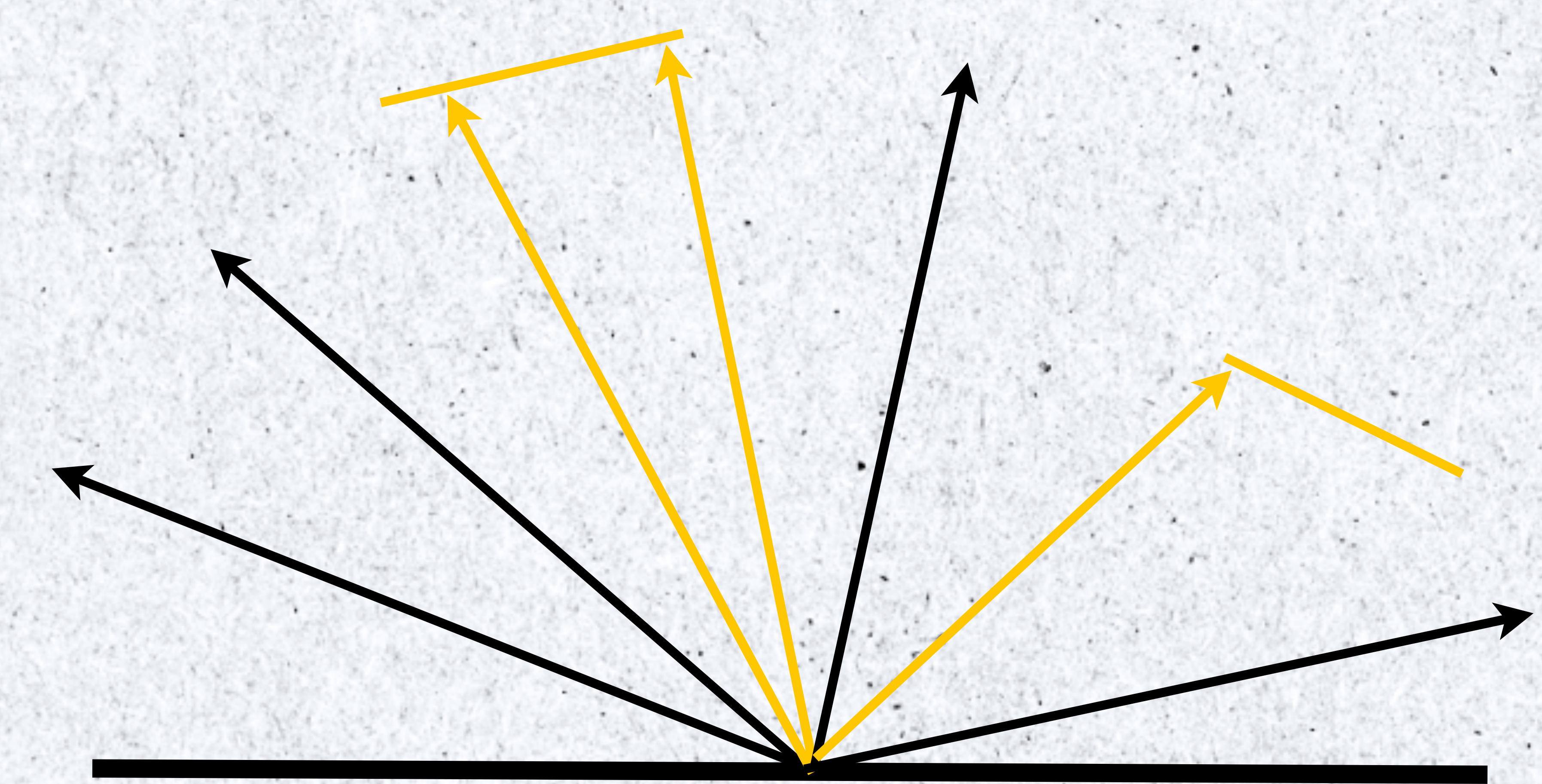
DIRECT LIGHTING INTEGRATOR



- Call sample() for each BRDF coshaders
- Create a pruned BSDFSampleStruct using the arealight BVH
- Pass the original and the pruned BSDFSampleStruct

BEFORE THE LIGHT LOOP: BRDF INTEGRATION

- The original struct contains all directions
- The pruned struct only contains samples hitting a light

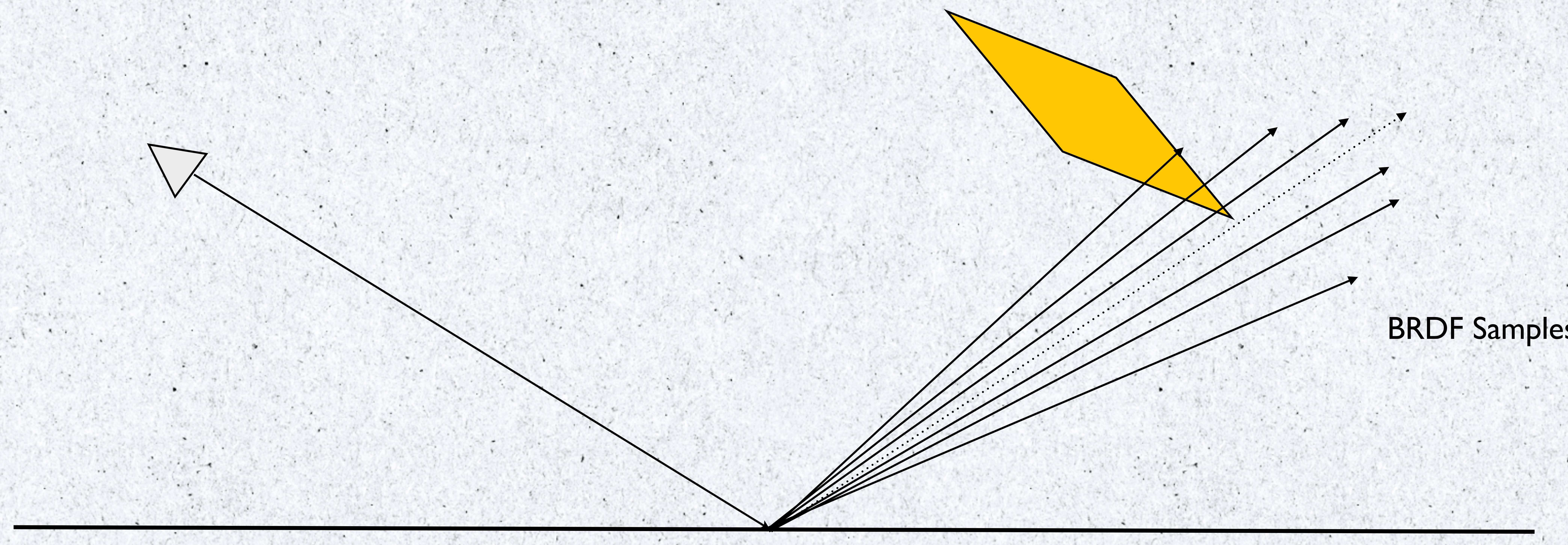


BEFORE THE LIGHT LOOP: BRDF INTEGRATION

```
void integrateBRDF(output BSDFSampleStruct bs, output BSDFSampleStruct bsbvh,
                   output int numSpecLobes, output int numSpecSamples)
{
    numSpecLobes = bsdf->getNumSpecSamples(numLobeSamples);
    if (numSpecLobes) {
     bs = bsdf->sample(facingRatio, w_in);
     numSpecSamples = bs->getNumSamples();

     bsbvh = BVHReduce(P, bs);
    }
}
```

BRDF SAMPLING



brdf samples around the reflection direction

- Call sample() on the light coshader
- Call valueAndPDF() on each BRDF coshader
- Call emissionAndPDF() on the light coshader
- ComputeMIS()
- ComputeShadows()

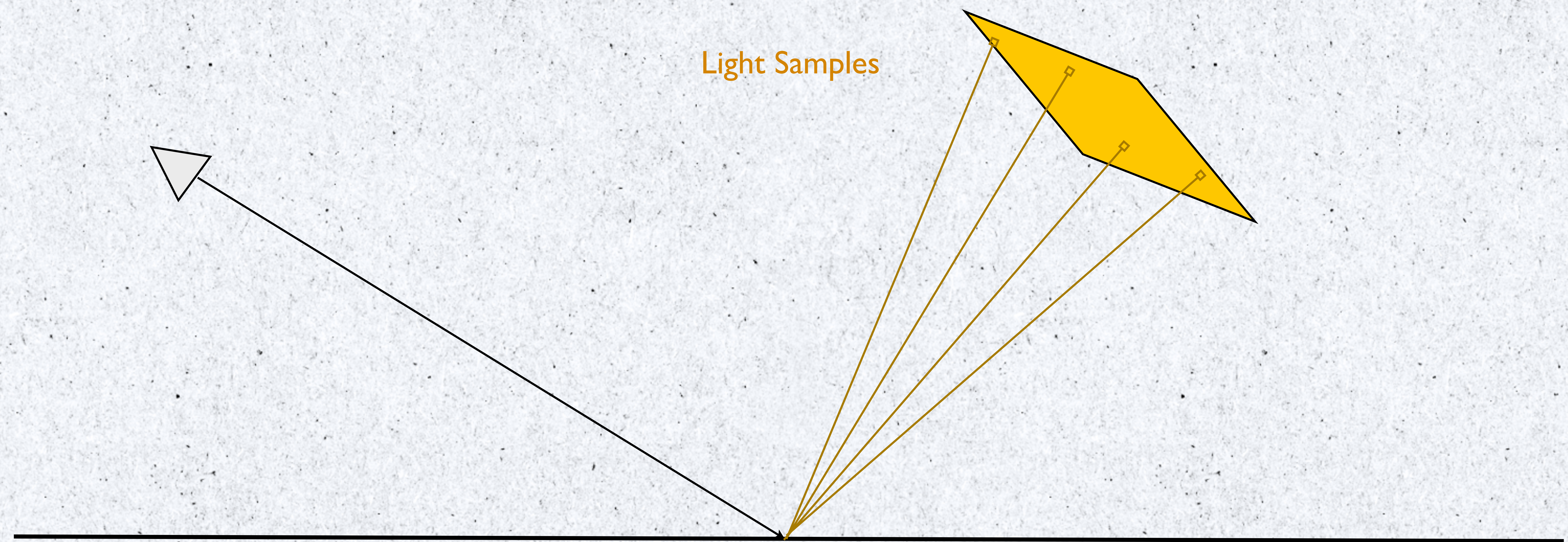
```
void integrateLight(shader li, output color diffPerLight,
                               output color specPerLight)
{
    li->emissionandPDF(bs, lightValues);

    li->sample(ls);
    bsdf->valueandPDF(facingRatio, w_in, w_outs, diffValues);

    computeMIS(ls, diffValues, specValues, bs, lightValues, Cdiff, Cspec, CspecBRDF);

    computeShadows(Cdiff, Cspec, diffPerLight, specPerLight);
    computeBRDFShadows(CspecBRDF, specPerLight);
}
```

LIGHT SAMPLING



light samples on the light

- Sample from lobes using BRDF coshaders
- Sample from lights using light coshaders
- Combine samples using Multiple Importance Sampling

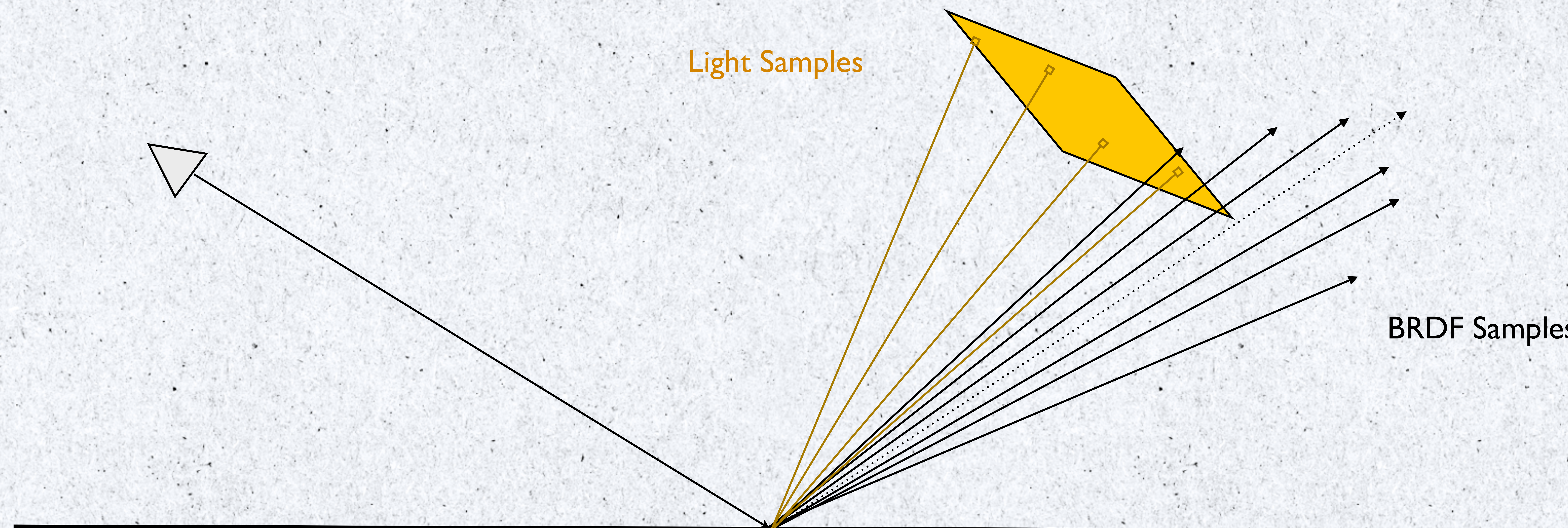
```
color integrate()
{
  integrateBRDF(bs, bsbvh, numSpecLobes, numSpecSamples);

  for (l=0; l<lightCount; l+=1) {
    integrateLight(lights[l], bs, numSpecLobes, numSpecSamples, diff, spec);

      resultDiff += diff;
      resultSpec += spec;
      result += diff+spec;
  }

  return result;
}
```

LIGHT LOOP

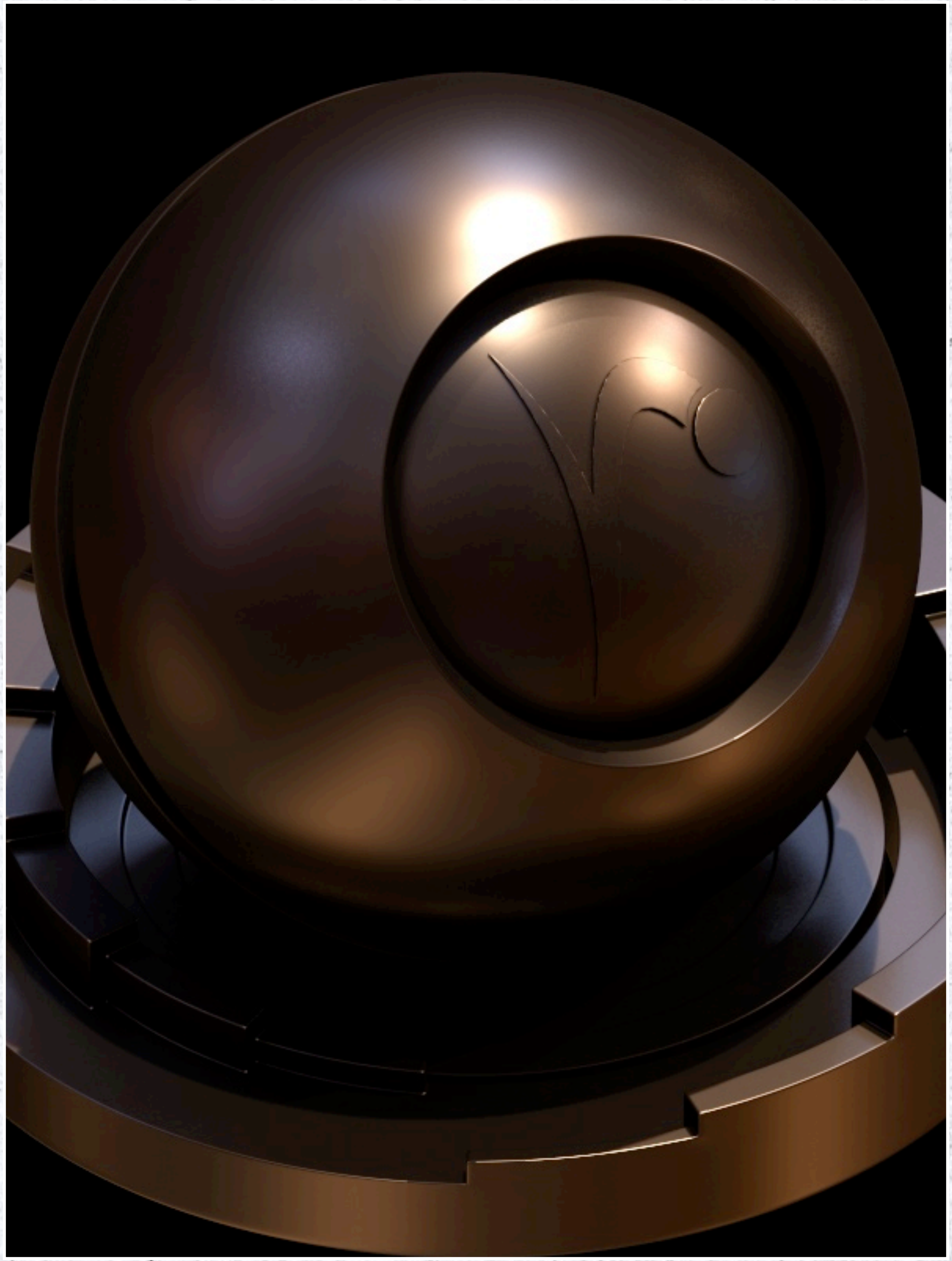


- BRDF sampling only
  - good sampling everywhere except in the highlights where BRDF sampled directions miss bright spots
  - fuzzy shadows

- Light Sampling Only
  - good sampling in the highlights, well defined shadows
  - noise in low brightness region where the sample directions don't match the BRDF peak

MULTIPLE IMPORTANCE SAMPLING

- Best of both worlds!



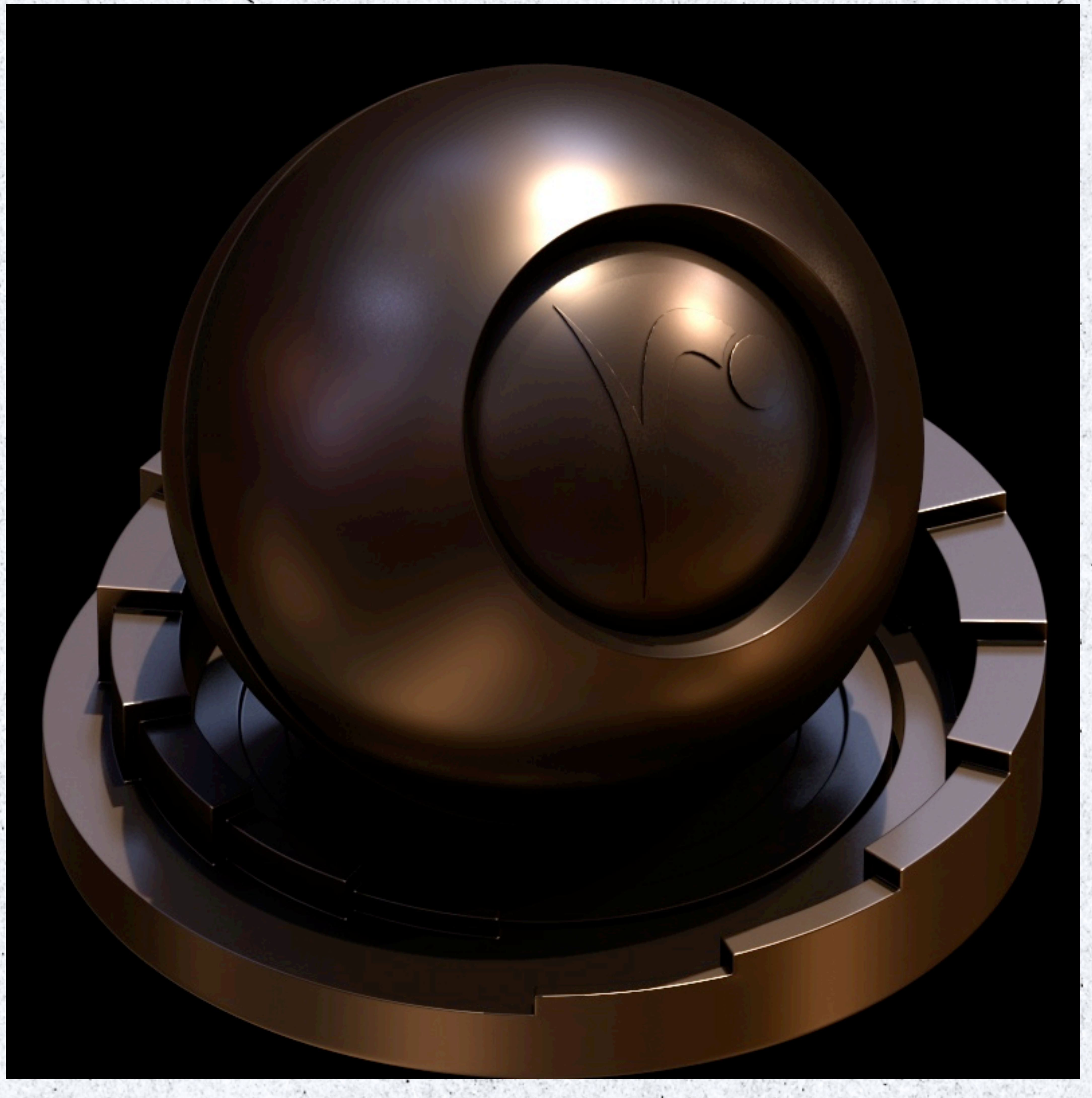
- Happens after MIS
  - can optimize shadow computation using the final weights of each sample (cutoff, russian roulette)
  - can perform a resampling step to reduce the number of shadow rays

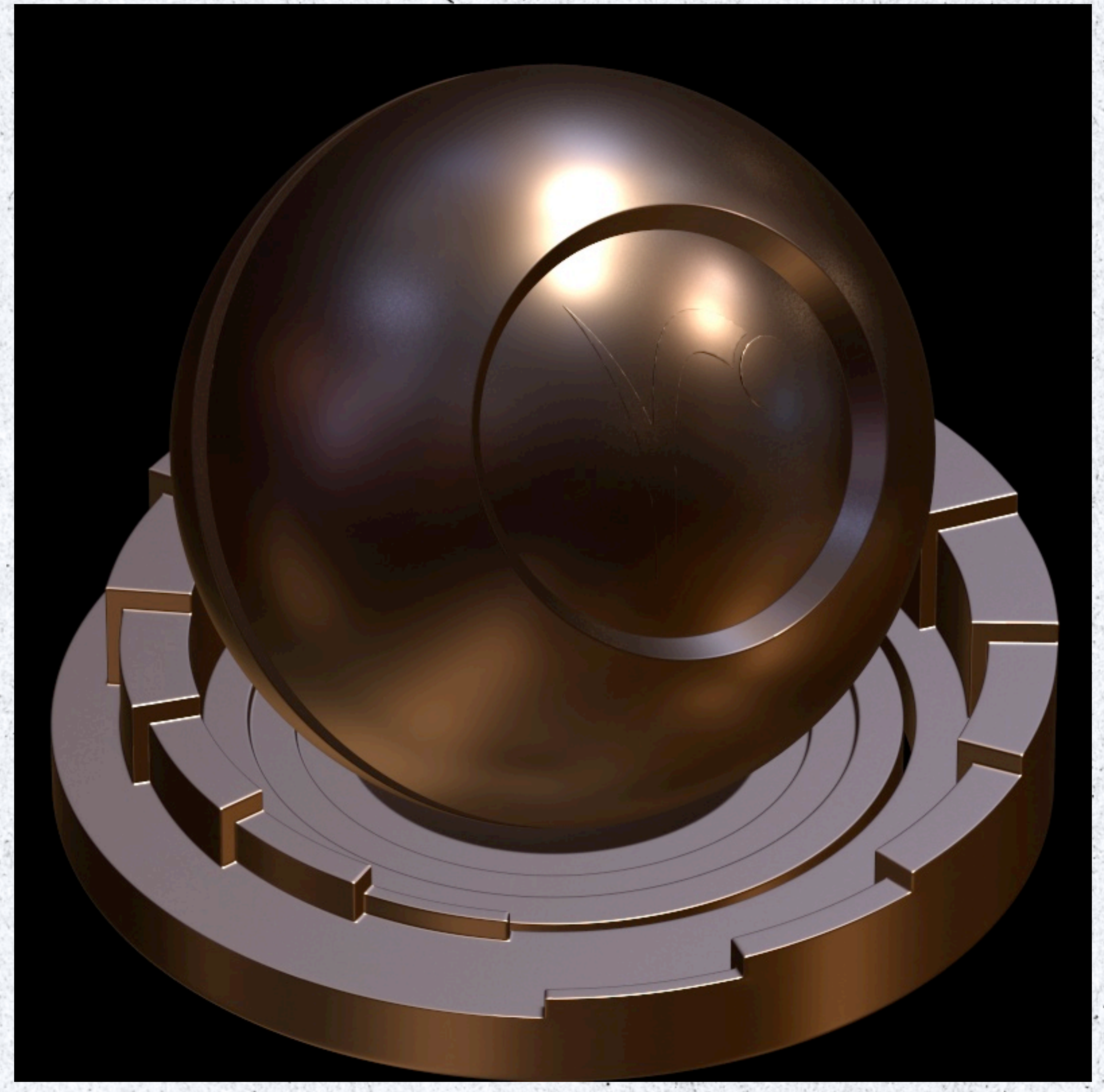
```
void computeLightShadows()
{
    avgVis = AreashadowRays(ls->P, ls->pdf, Lvis);

    for(i=0; i<lightSamples; i+=1) {
        diffPerLight += Lvis[i] * Cdiff[i];
        diffPerLightNoshad += Cdiff[i];
        specPerLight += Lvis[i] * Cspec[i];
    }

    if (li->diffuConvolution(diffConv)) {
        diffConv *= bsdf->albedo();
        diffPerLight += avgVis * (diffConv -diffPerLightNoshad);
        diffPerLightNoShad = diffConv;
    }
}
```

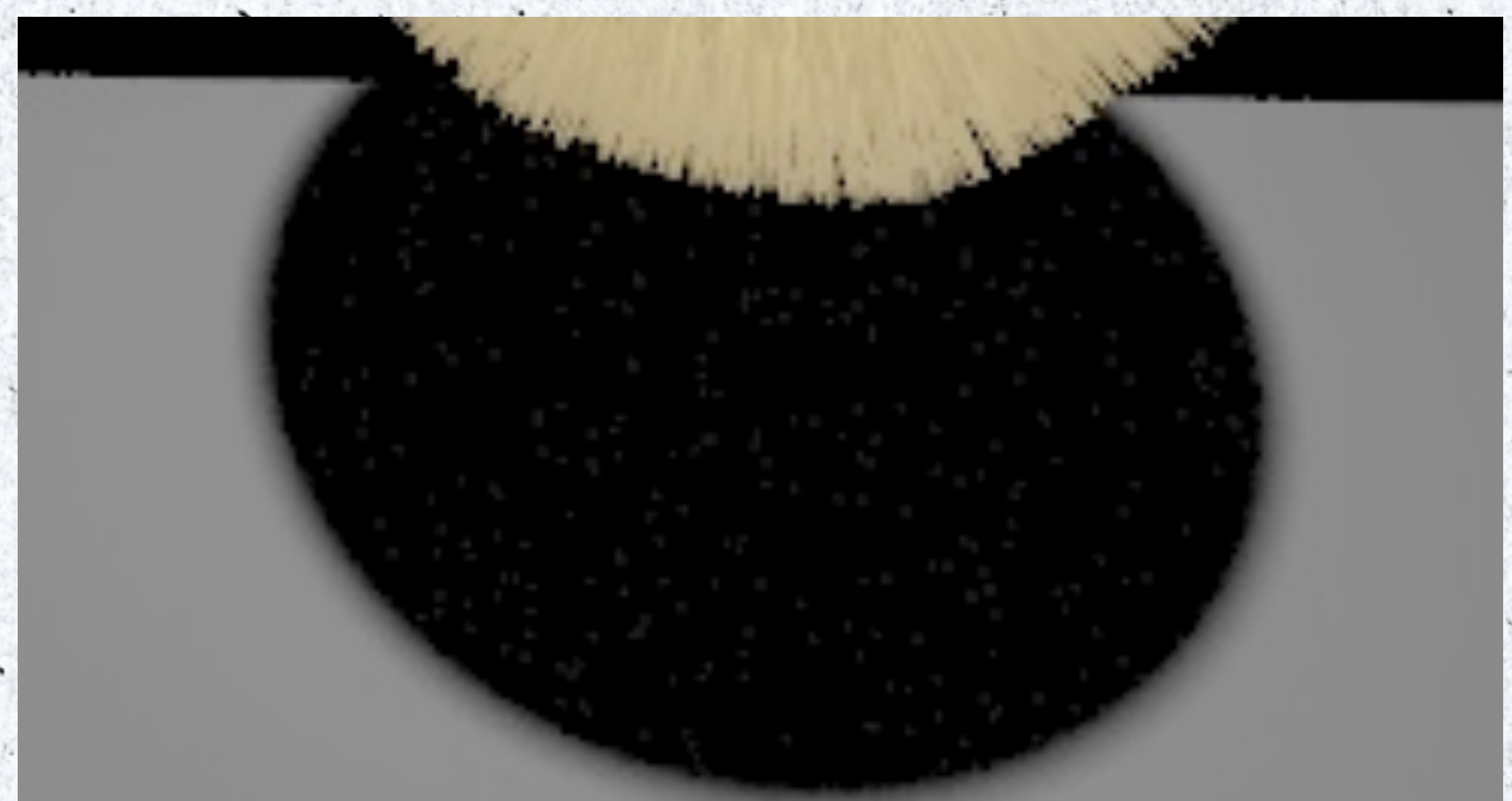
SHADOWING



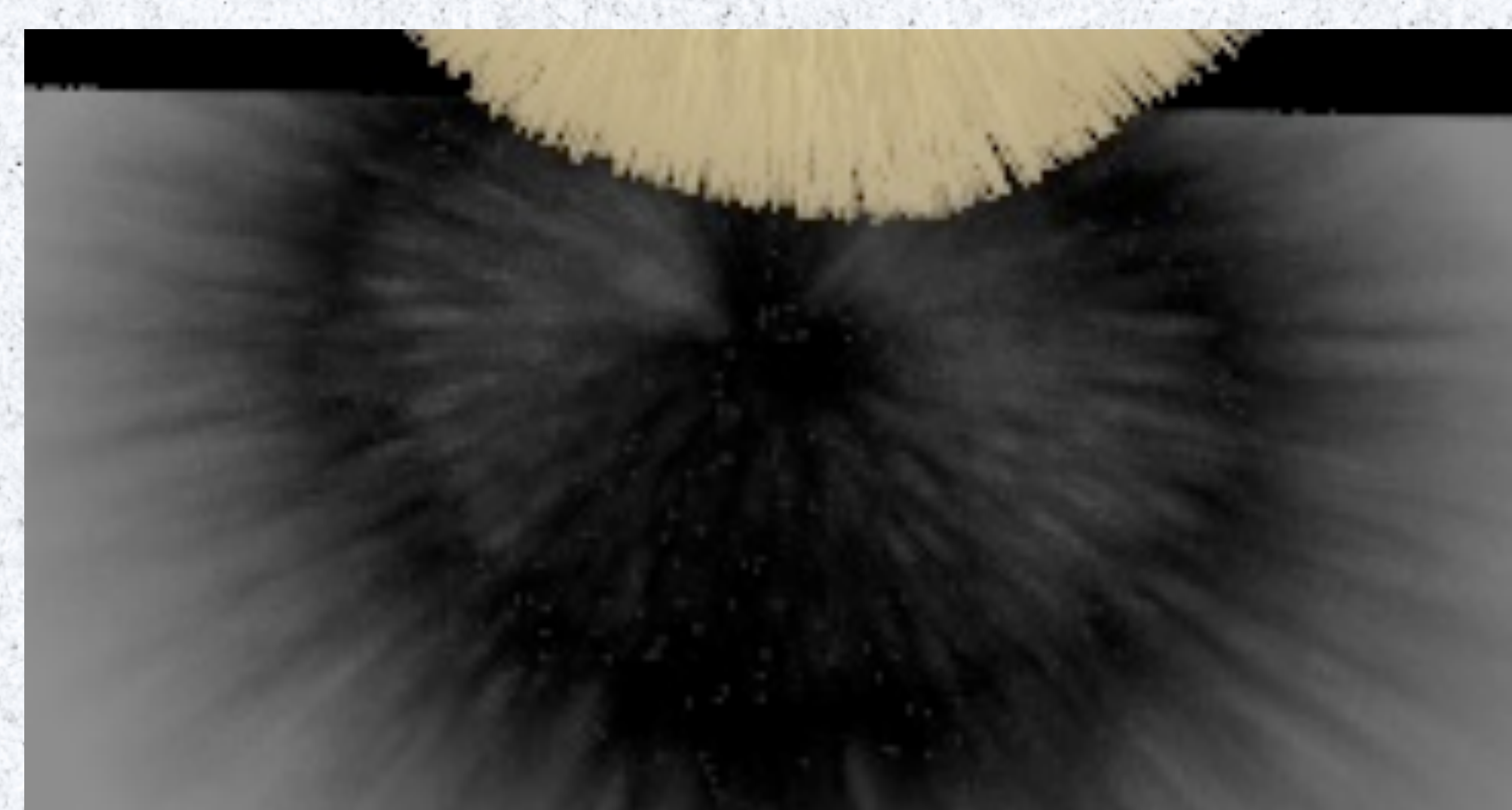


- Can use multiple techniques as long as they don't overlap and create double shadowing
  - shadowmaps for hair
  - SH for large terrain vegetation

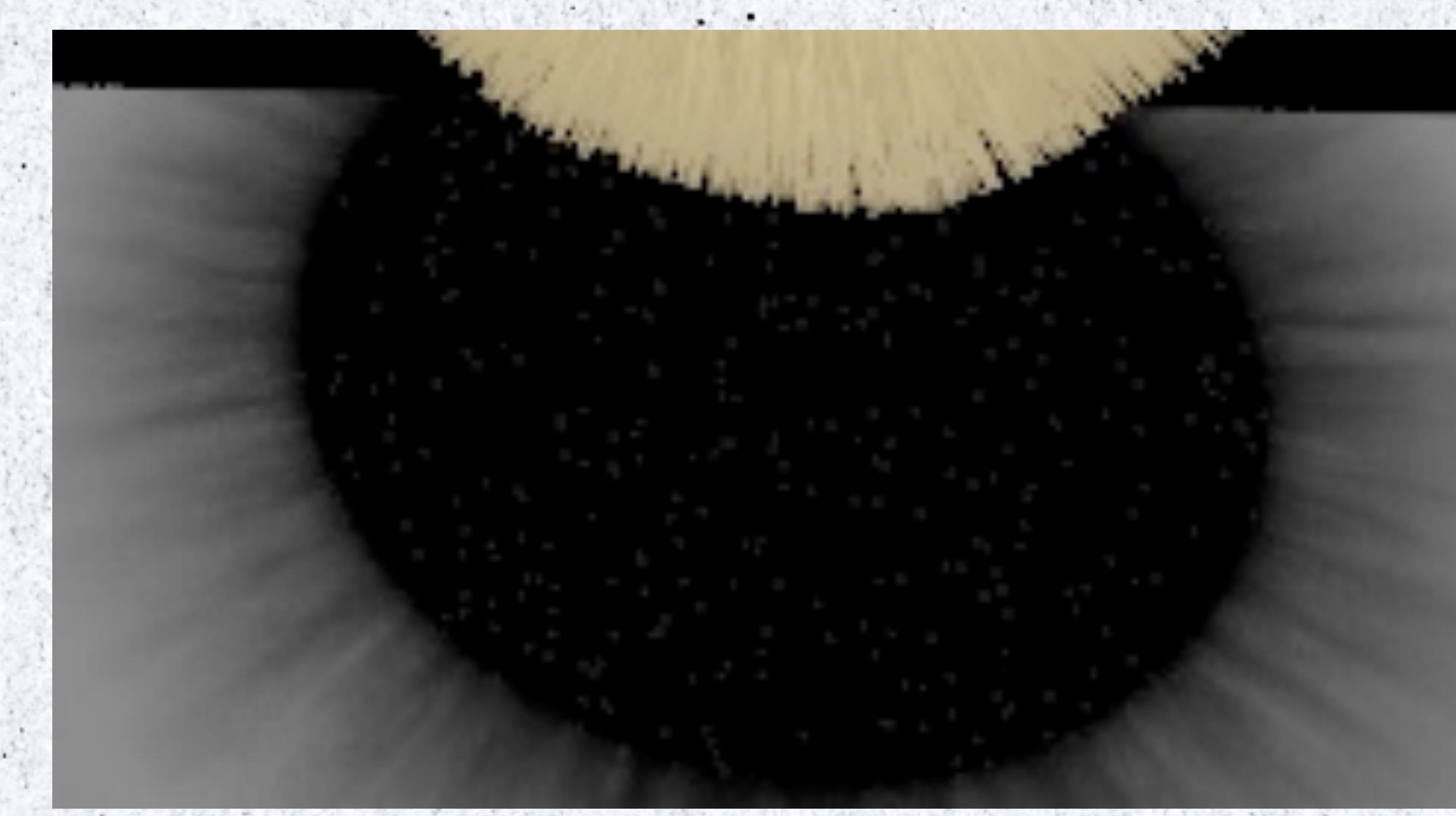
SHADOWING



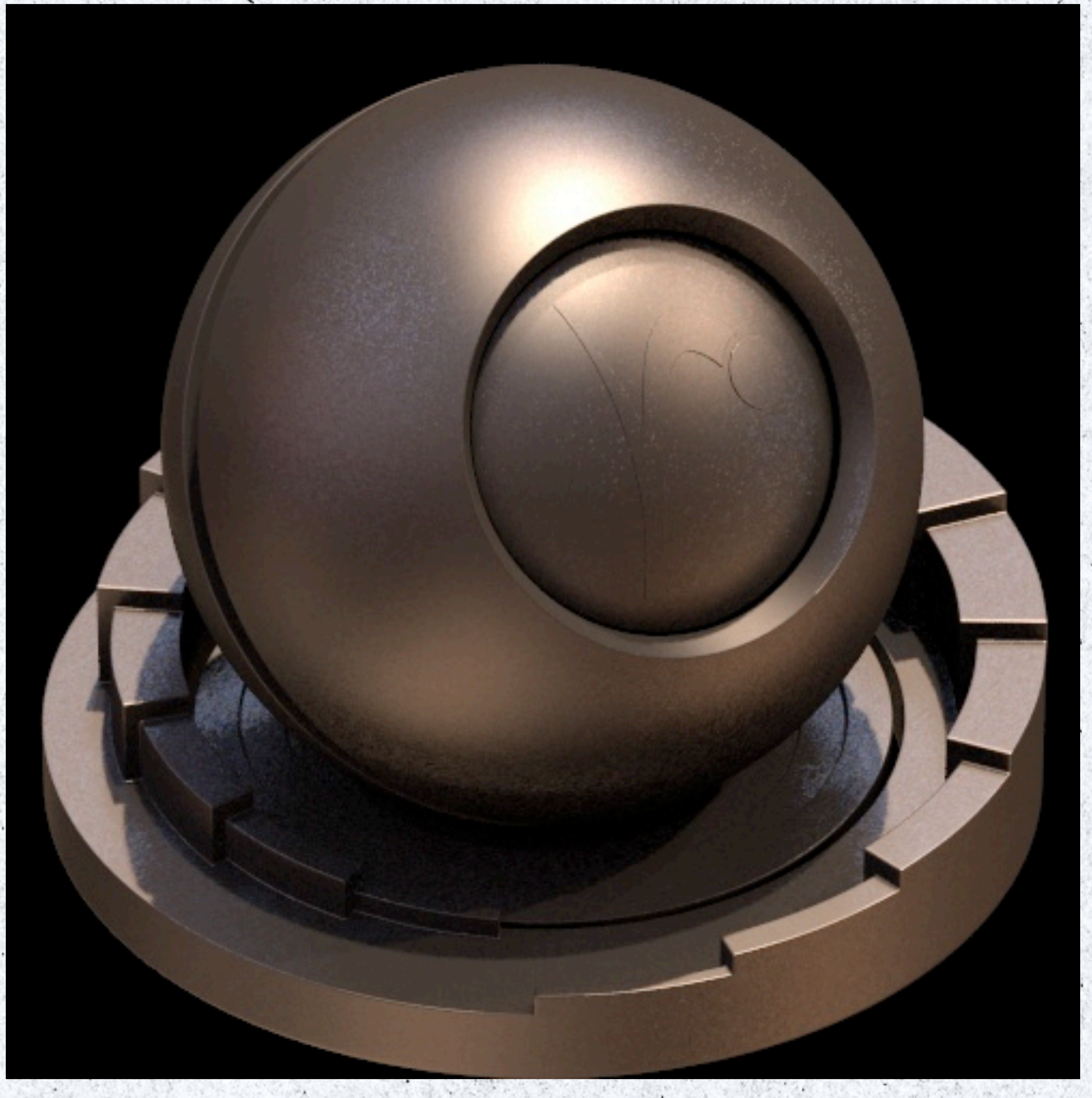
+

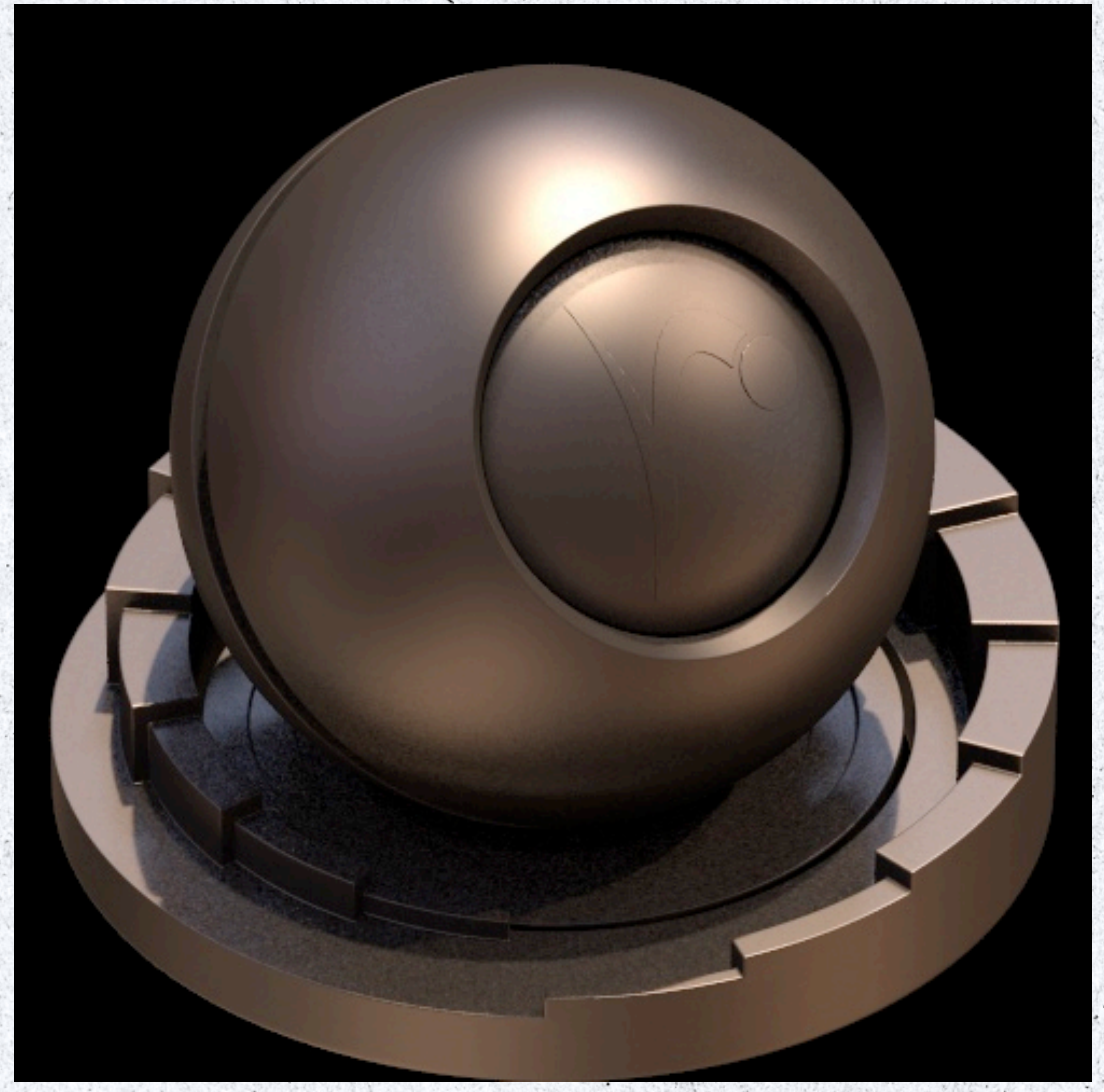


=



RESAMPLING MULTIPLE IMPORTANCE SAMPLING





- Use of average visibility to further reduce variance
  - the lobe must be completely view independent to be computed before rendering
  - the light must be able to provide such answer

Tuesday, July 23, 13

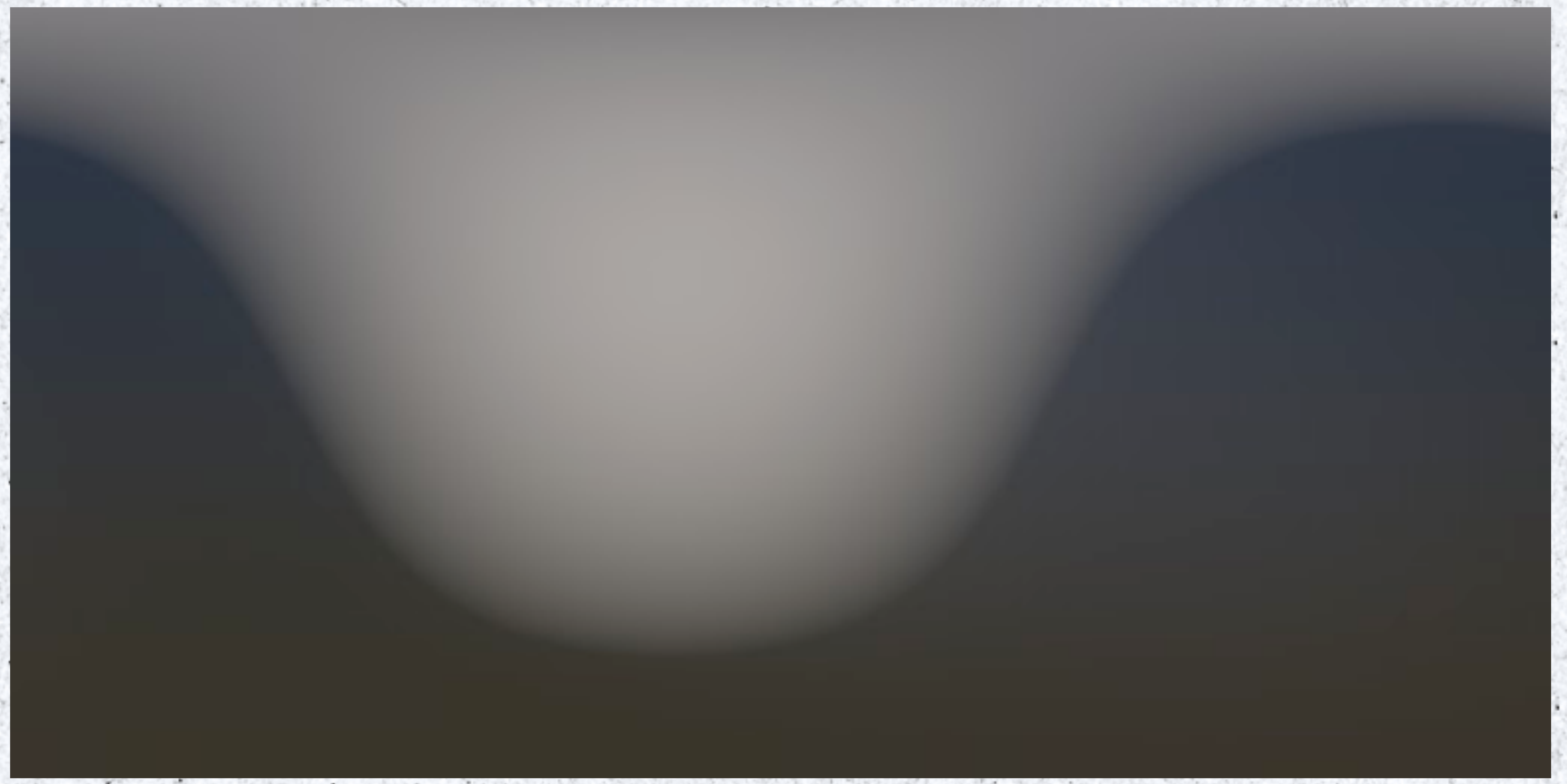
Control variates is a classic Monte Carlo technique.
Here we rely on the lights providing an analytical integral (usually of their diffuse contribution).

- Sphere: sub-hemispherical light source integration
- Rect, Disk: polygonal light source analytical integration
- Dome: image space convolution

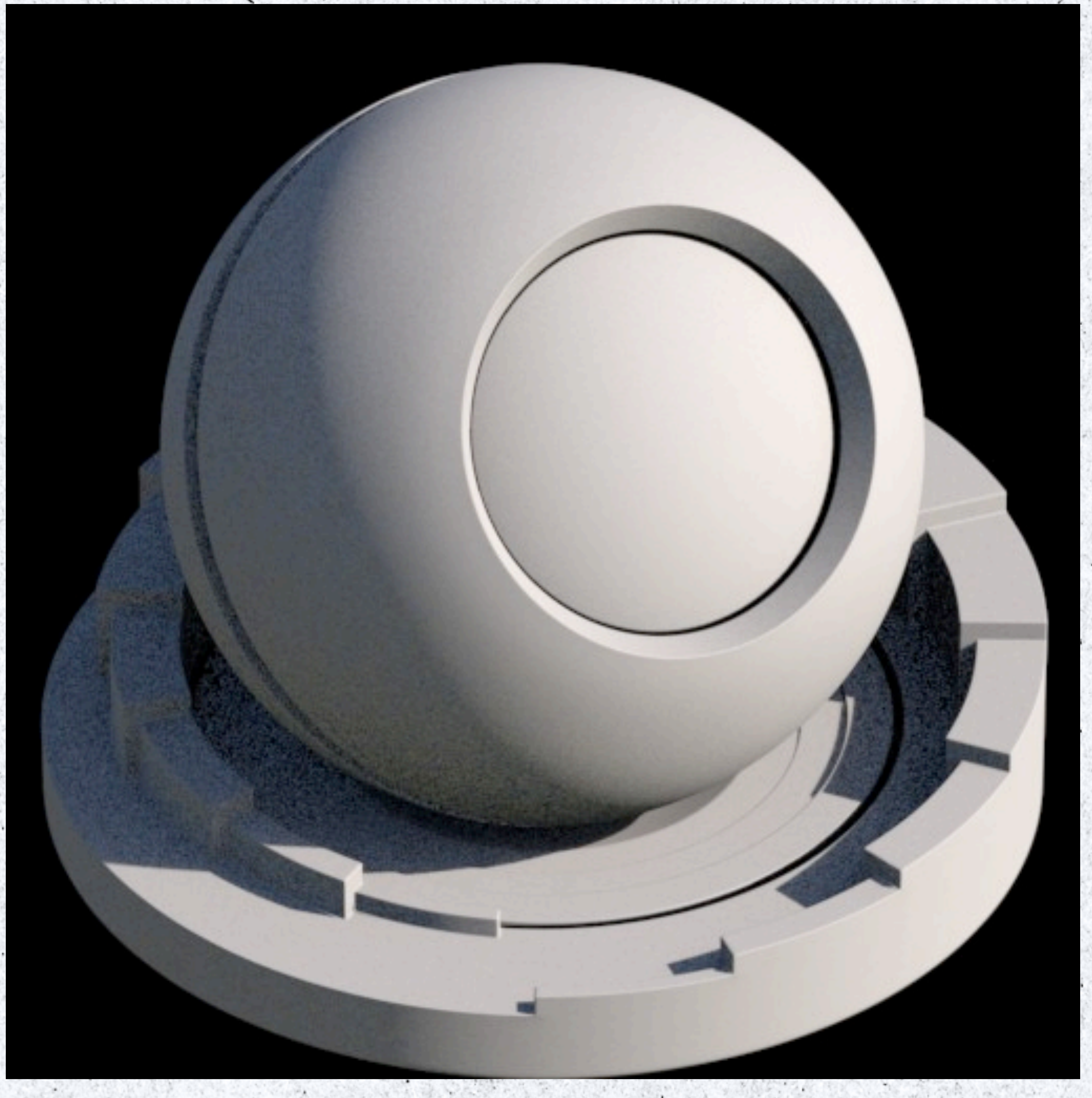
Tuesday, July 23, 13

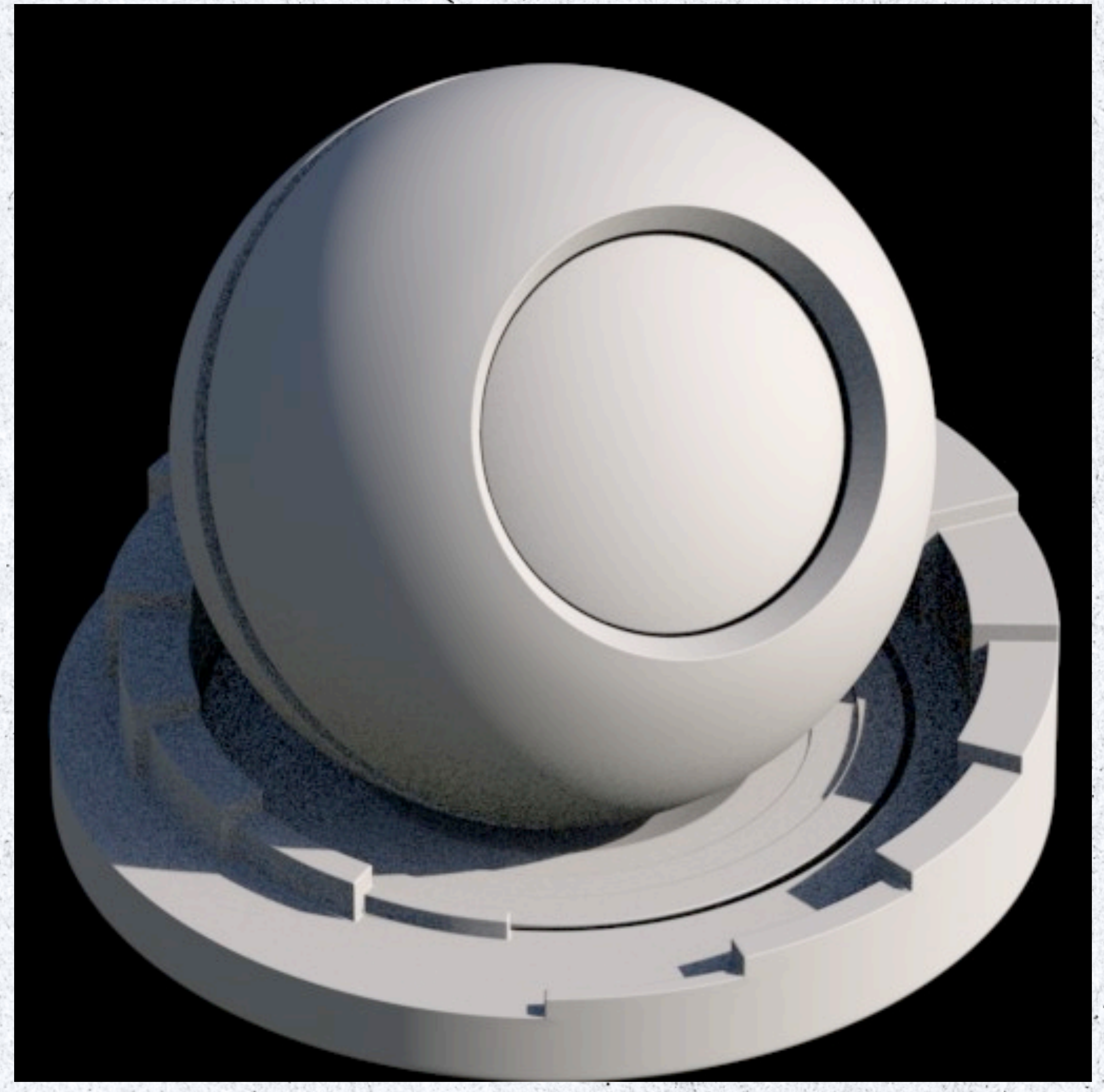
Details and references are in the course notes.

Pre-computed convolution for a dome light texture



CONTROL VARIATES





- try to be energy conserving
- reflections are handled through the BRDFs

NEW MATERIALS... CONSEQUENCES

- no distinction between "planar reflections" and "planar area lights"
- no distinction between "spherical environments" and "dome area lights"
- not only true for specular: diffuse and bounce are also interchangeable

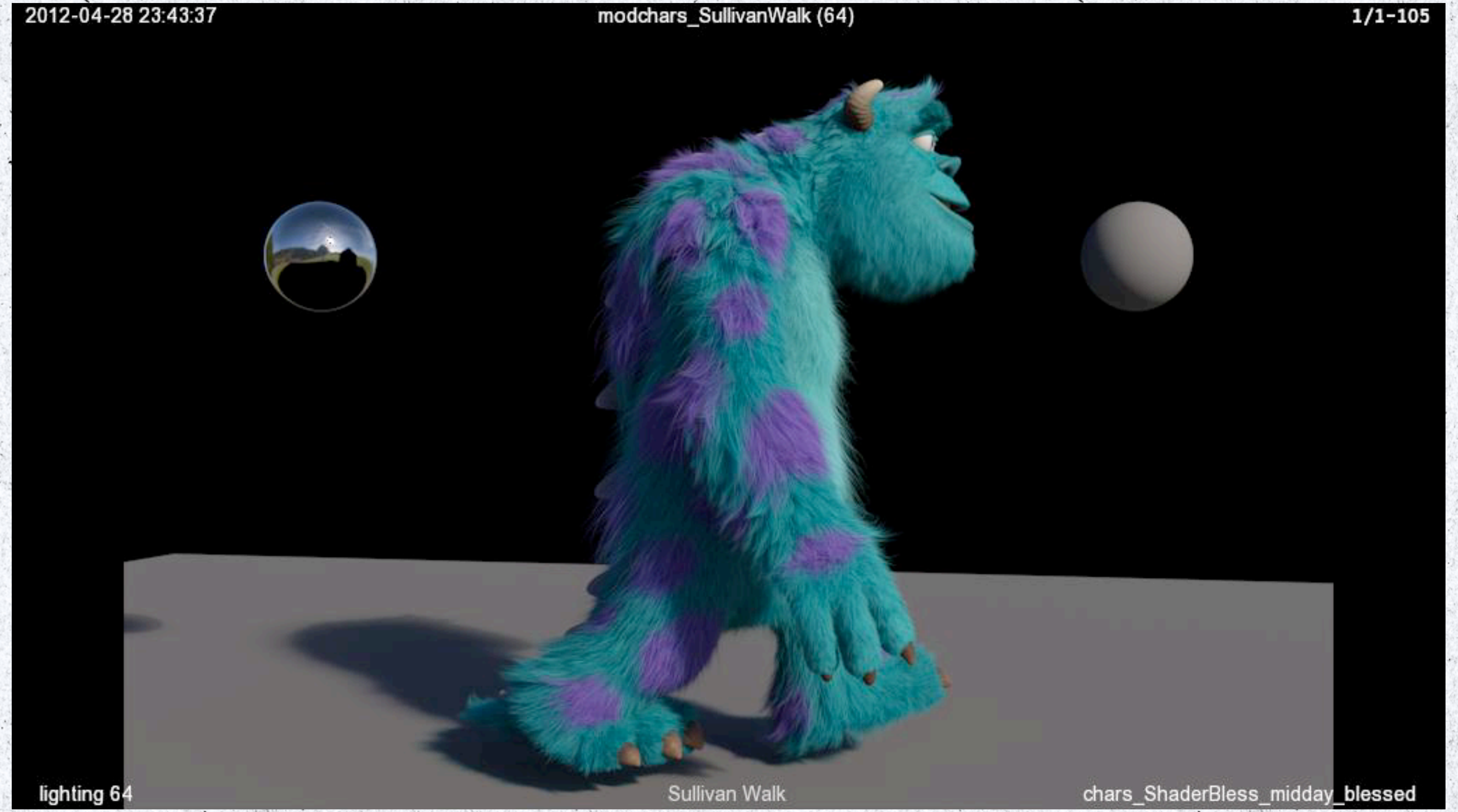
LOOK DEVELOPMENT: IBL1

My friend Arkell Rasiah and I shot a series of IBLs, which we initially used to calibrate materials.

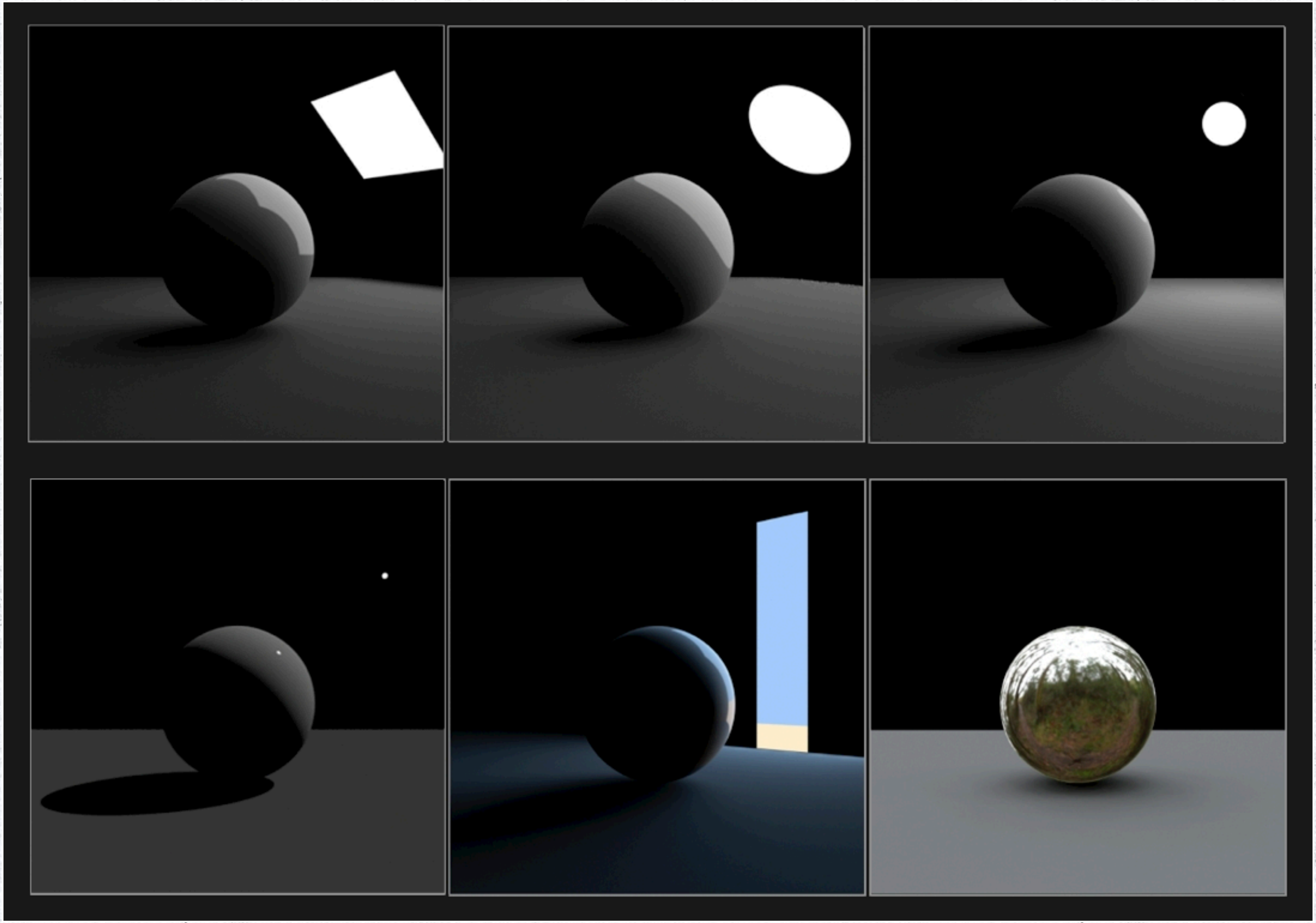
LOOK DEVELOPMENT: IBL2

LOOK DEVELOPMENT: IBL1

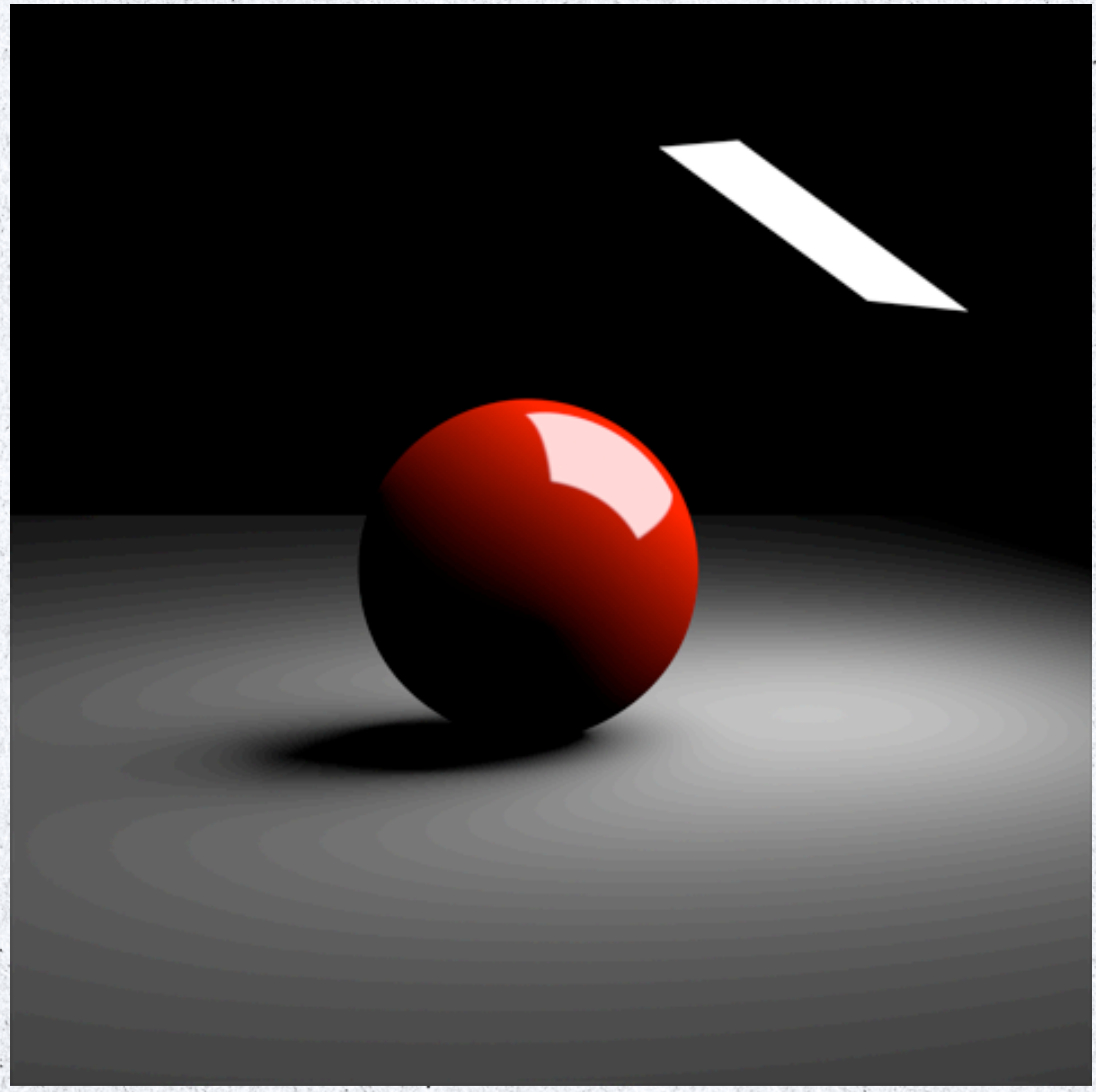
LOOK DEVELOPMENT: IBL2



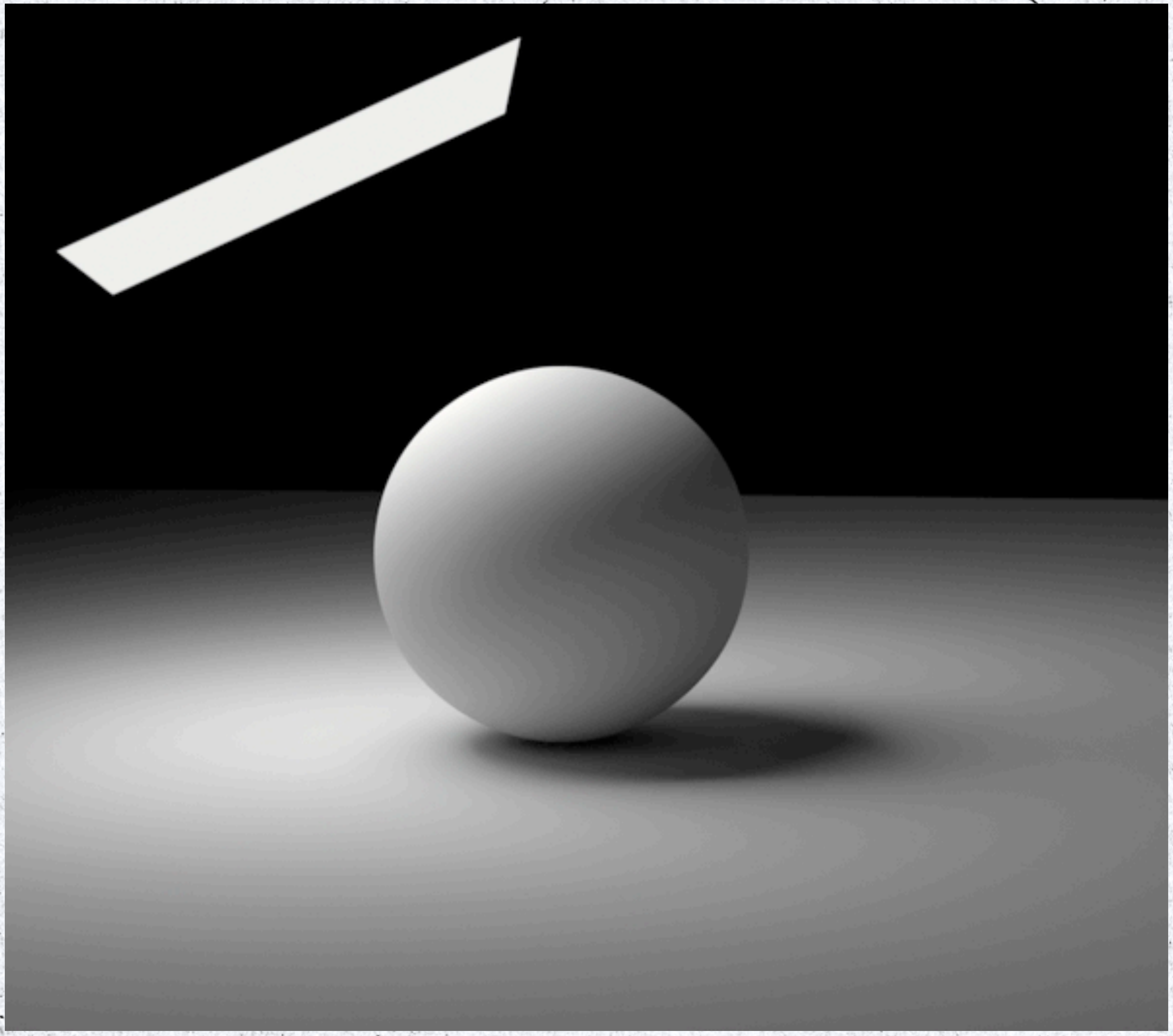
- various shapes: rect, disk, sphere
- infinite dome light (IBL)
- all can be textured, provide diffuse and specular and produce soft shadows
- only one point light: sun



EXAMPLE: RECT AREA LIGHT

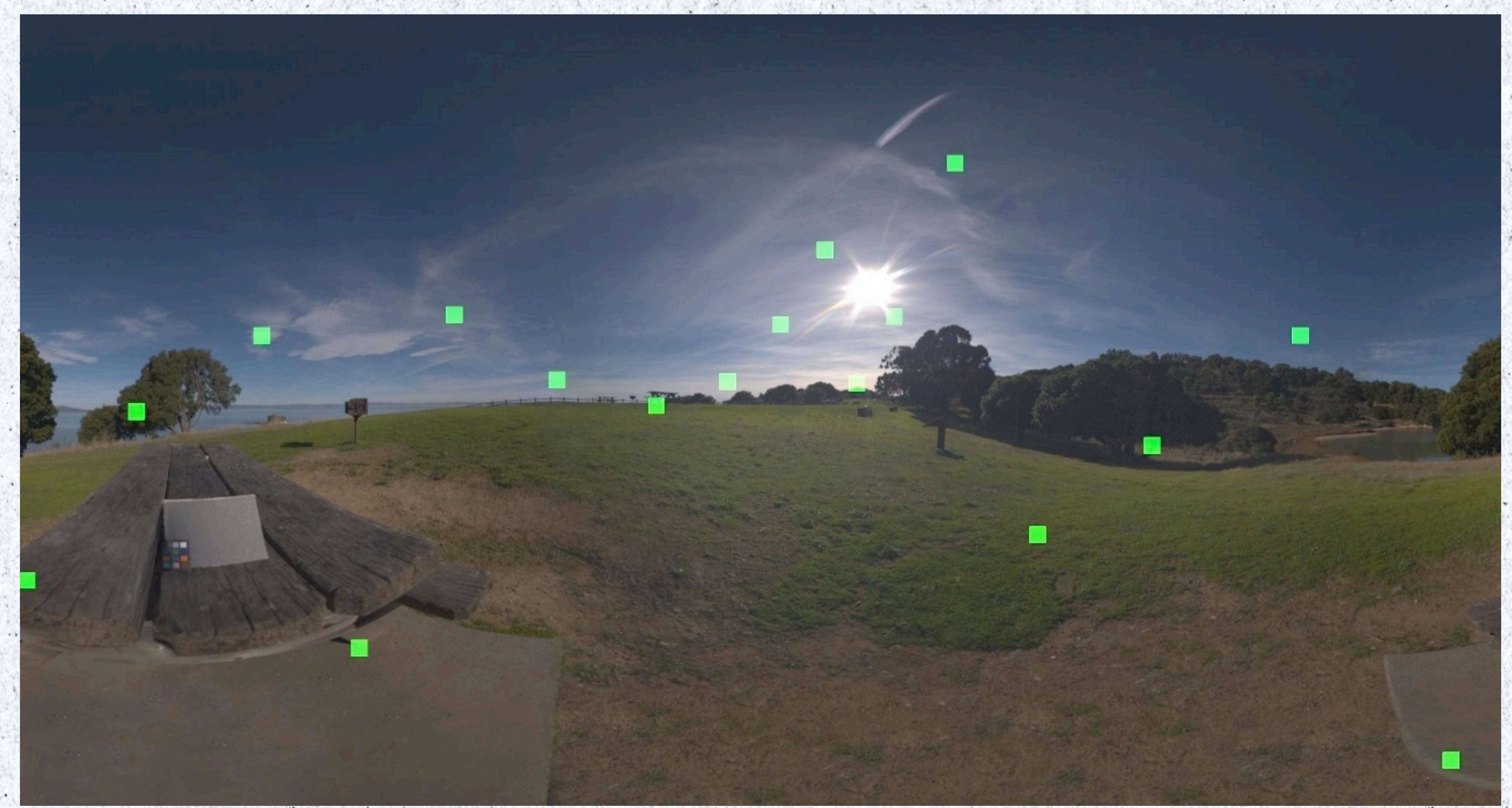


EXAMPLE: RECT AREA LIGHT

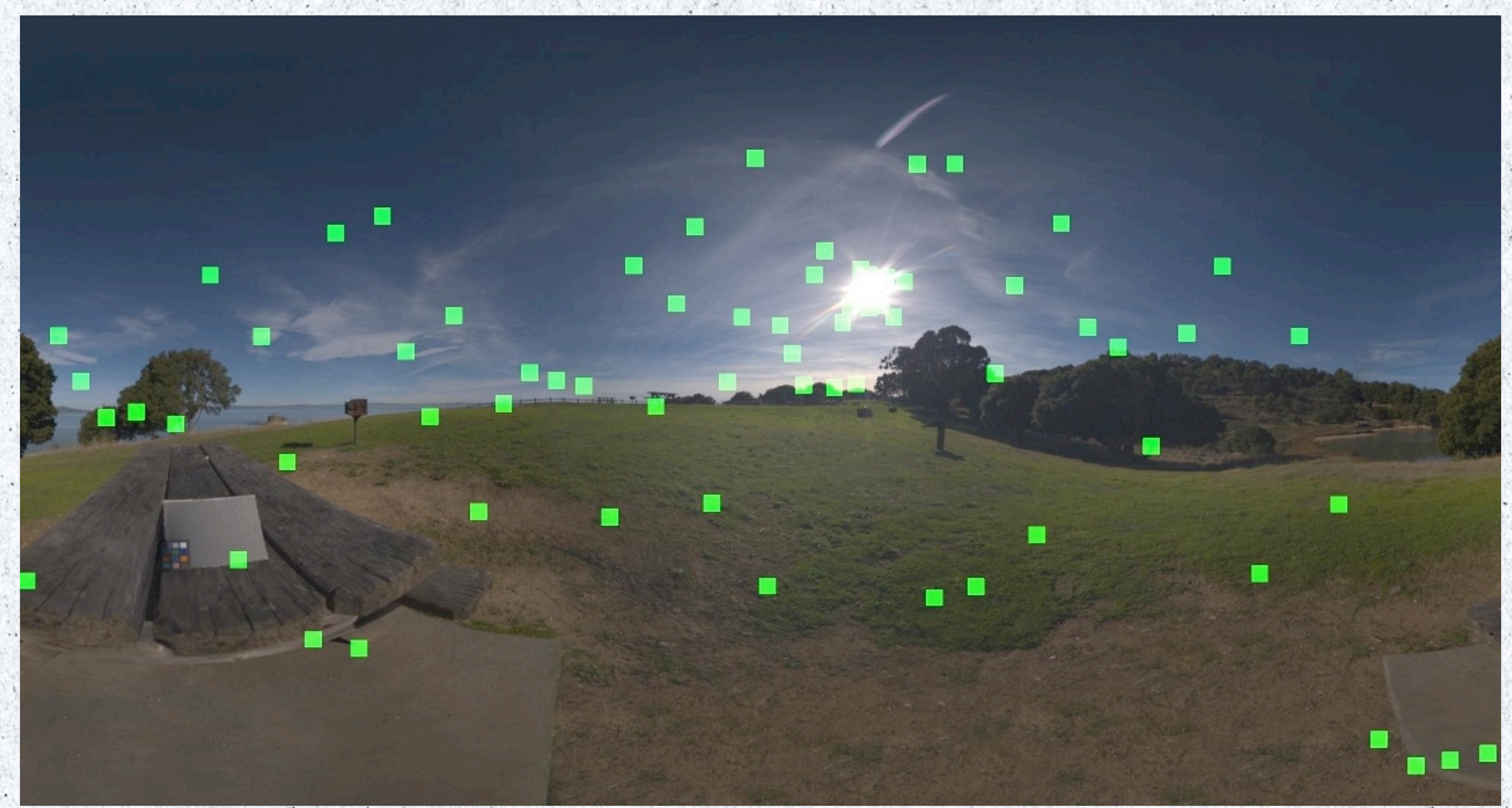


EXAMPLE: DOME LIGHT

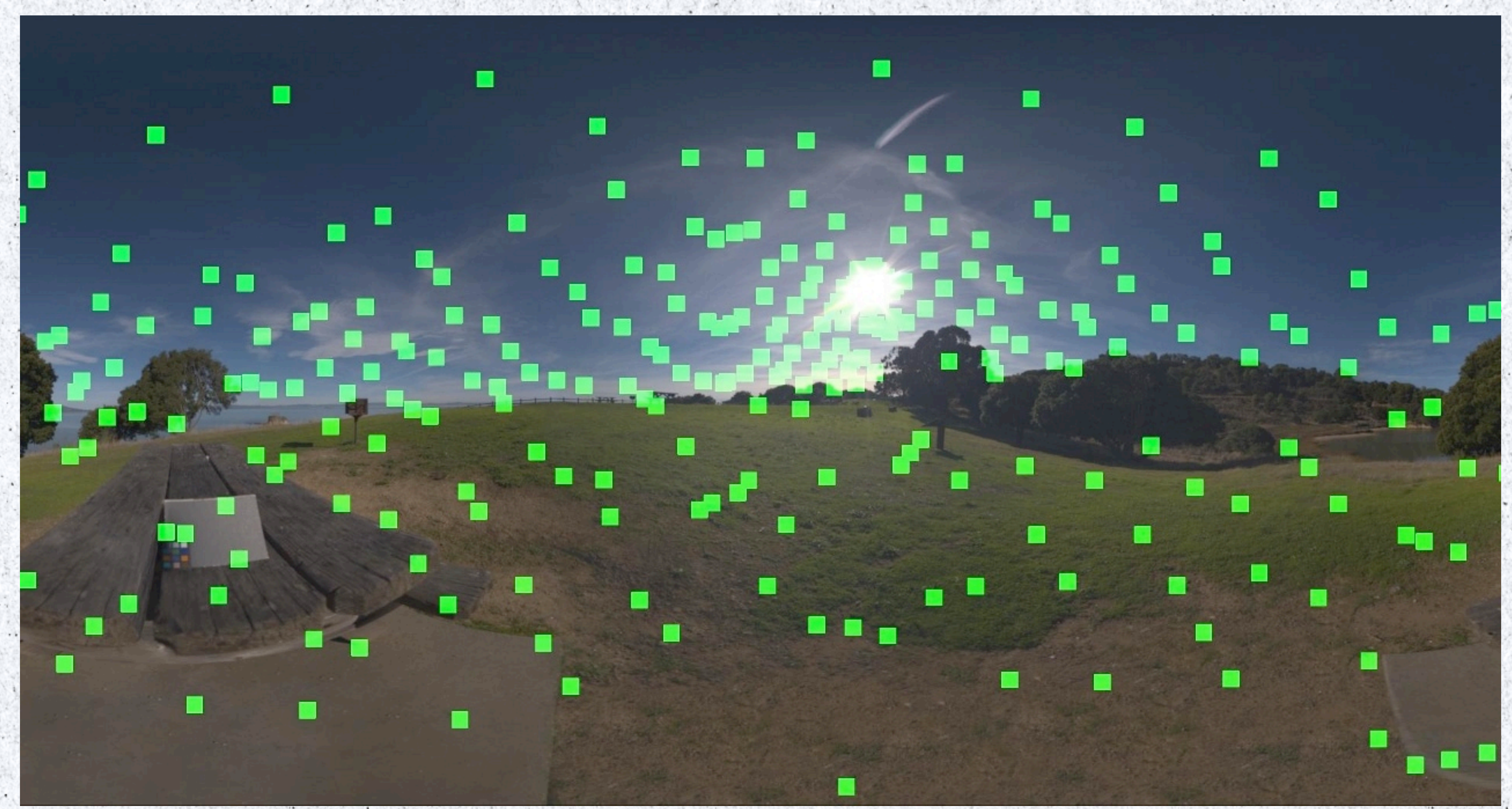
DOME IMPORTANCE SAMPLING



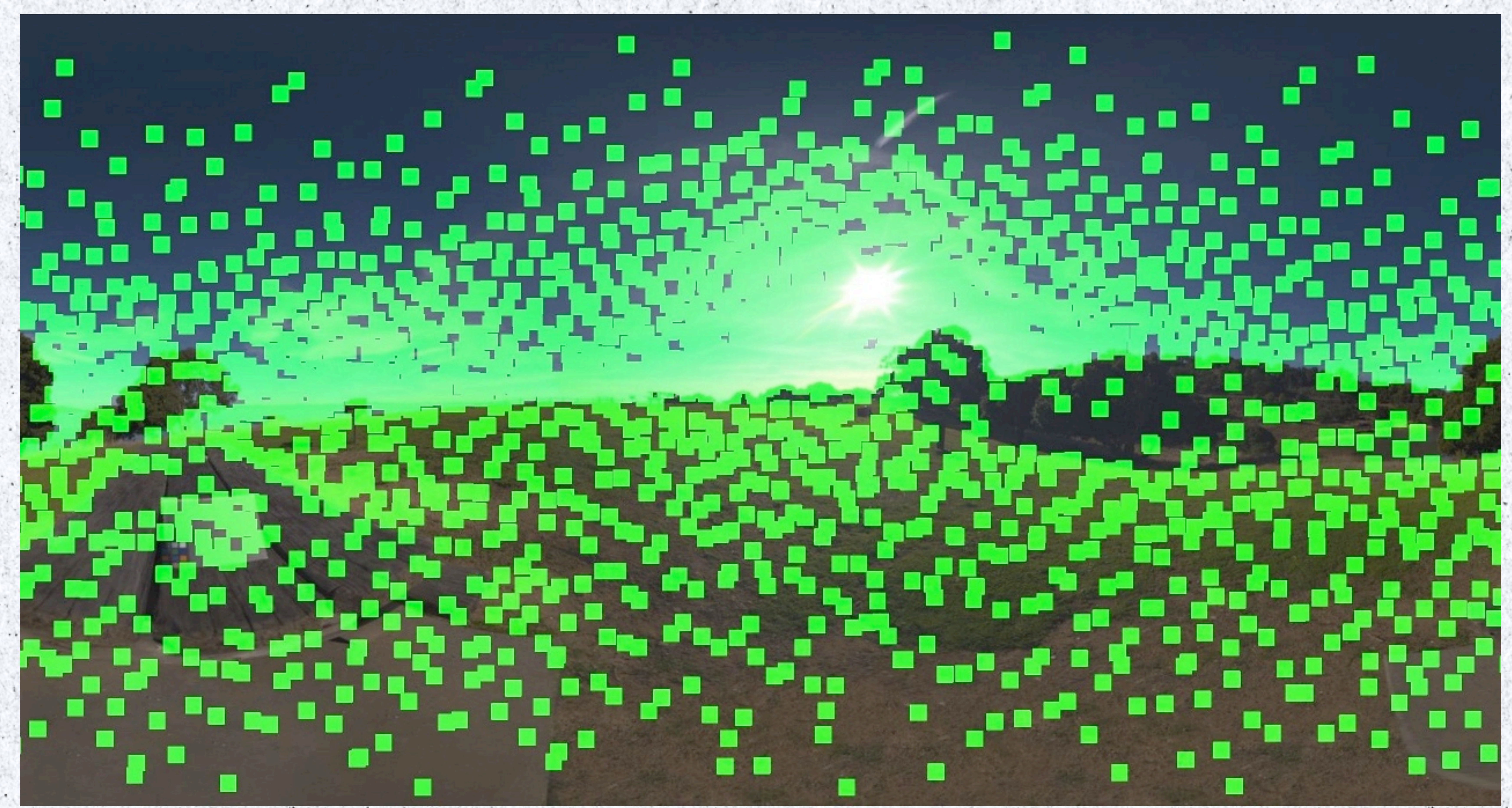
DOME IMPORTANCE SAMPLING



DOME IMPORTANCE SAMPLING



DOME IMPORTANCE SAMPLING




SCHOOL OF SCARING
Sunny

- must be energy conserving (normalized)
- must "substitute" to our trusted old Cook-Torrance
- should be anisotropy aware (brushed metals)
- can be efficiently computed (sampling)

based on Beckmann distribution:

$$D_b(\mathbf{m}) = \frac{e^{-\tan^2(\theta_m)/\alpha^2}}{\pi\,\alpha^2\,\cos^4(\theta_m)} = \frac{e^{\frac{(\mathbf{n}\cdot\mathbf{m})^2-1}{\alpha^2(\mathbf{n}\cdot\mathbf{m})^2}}}{\pi\,\alpha^2\,(\mathbf{n}\cdot\mathbf{m})^4}$$

"OUR" SOLUTION: D-BRDF

$$f_\mu(\mathbf{l}, \mathbf{v}) = \frac{F(\mathbf{l}, \mathbf{h})\, D_b(\mathbf{h})\, (\mathbf{n} \cdot \mathbf{h})}{4\, (\mathbf{n} \cdot \mathbf{l})\, (\mathbf{v} \cdot \mathbf{h})}$$

$$pdf = \frac{D_b(\mathbf{h})\, (\mathbf{n} \cdot \mathbf{h})}{4\, (\mathbf{v} \cdot \mathbf{h})}$$

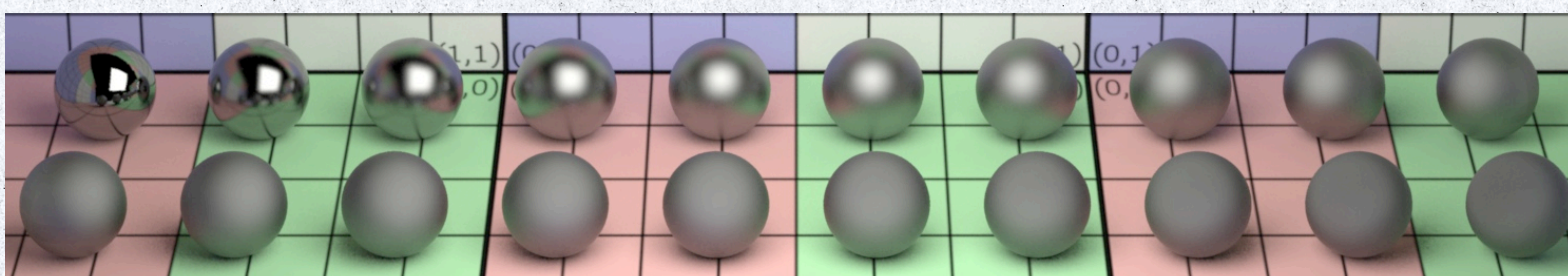
We imposed a strict energy conservation to the model and worked backwards to derive it. The details are in the course notes. We importance sample the main exponential term in the Beckmann distribution, similarly to what Walter proposed, and thus the pdf is the following.

split roughness α into 2 distinct coefficients along 2 surface directions

$$D_b(\mathbf{h}) = \frac{e^{-\frac{\left(\frac{\mathbf{u_1}\cdot\mathbf{h}}{\alpha_1}\right)^2+\left(\frac{\mathbf{u_2}\cdot\mathbf{h}}{\alpha_2}\right)^2}{(\mathbf{n}\cdot\mathbf{h})^2}}}{\pi\,\alpha_1\,\alpha_2\,(\mathbf{n}\cdot\mathbf{h})^4}$$

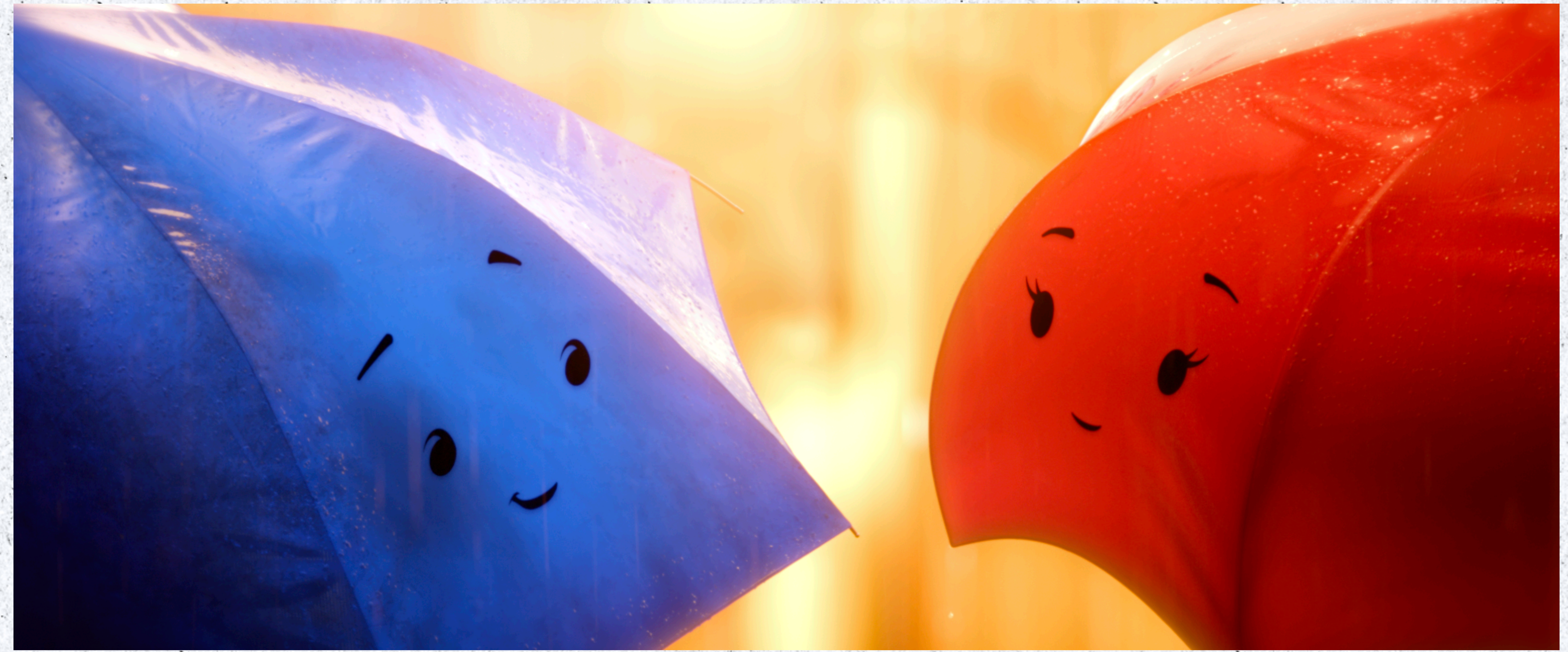
Anisotropy is a very simple extension.

BECKMANN BRDF RESULTS



- Eric Veach's paper: http://graphics.stanford.edu/papers/combine
- D-BRDF: http://www.cs.utah.edu/~premoze/dbrdf/dBRDF.pdf
- Siggraph2012 Monte-Carlo Course: https://sites.google.com/site/qmcrendering/

Like in the story, many people gave us a helping hand in this work.



Tuesday, July 23, 13

So thank you

.. to the folks in the GI team (Bill Reeves, Jean-Claude Kalache, Davide Pesare, Jacob Kunzel, Farhez Rayani and Guido Quaroni), the lighters and shading artists on Monsters University and The Blue Umbrella.