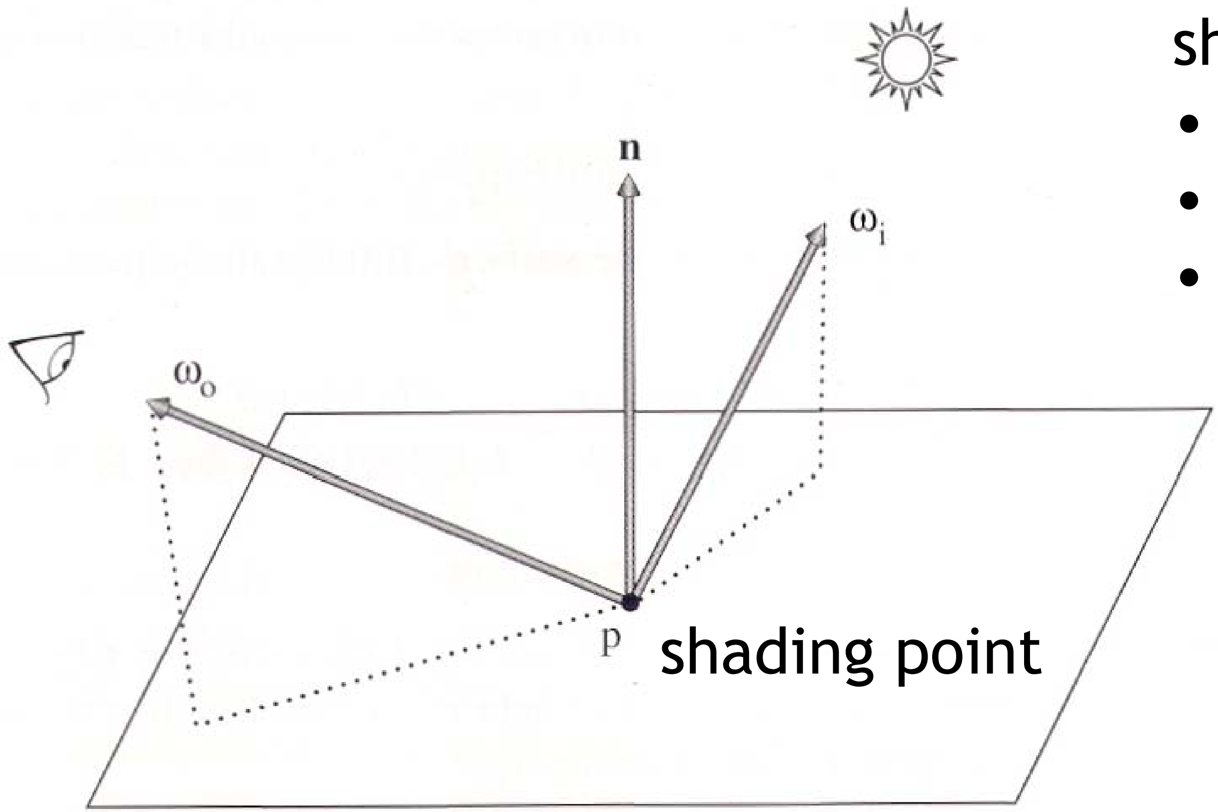
Reflection models

Digital Image Synthesis

Yung-Yu Chuang

with slides by Pat Hanrahan and Matt Pharr

Rendering equation



shading model
- accuracy
- expressiveness
- speed

$$L(\omega_o) = \int_{\Omega} f(\omega_i \rightarrow \omega_o) L(\omega_i) \cos\theta_i \, d\omega_i$$

Taxonomy 1

$$(x, y, t, \theta, \phi, \lambda)_{in} \rightarrow (x, y, t, \theta, \phi, \lambda)_{out}$$

General function = 12D

↓ assume time doesn't matter (no phosphorescence)
assume wavelengths are equal (no fluorescence)

Scattering function = 9D

↓ assume wavelength is discretized or integrated into RGB
(This is a common assumption for computer graphics)

Single-wavelength Scattering function = 8D

$$(x, y, \theta, \phi)_{in} \rightarrow (x, y, \theta, \phi)_{out}$$

Taxonomy 2

$$(x, y, \theta, \phi)_{in} \rightarrow (x, y, \theta, \phi)_{out}$$

Single-wavelength Scattering function = 8D

ignore subsurface scattering (x,y) in = (x,y) out

ignore dependence on position

Bidirectional Texture Function (BTF)
Spatially-varying BRDF (SVBRDF) = 6D

Bidirectional Subsurface Scattering Distribution Function (BSSRDF) = 6D

ignore direction of incident light

ignore dependence on position

ignore subsurface scattering

Light Fields, Surface LFs = 4D

$$(x, y, \theta, \phi)_{out}$$

BRDF = 4D

$$(\theta, \phi)_{in} \rightarrow (\theta, \phi)_{out}$$

assume Lambertian

assume isotropy

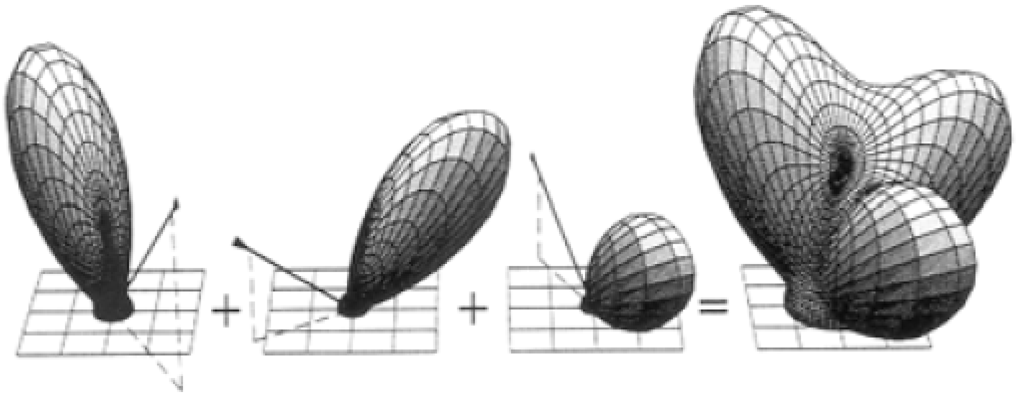
Texture Maps = 2D

$$(x, y)_{out}$$

3D

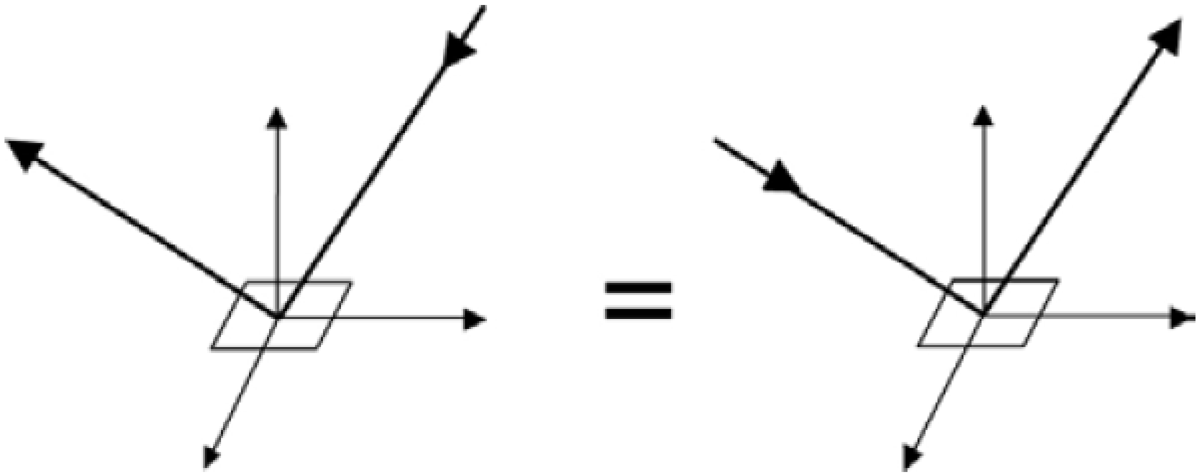
Properties of BRDFs

1. Linear



From Sillion, Arvo, Westin, Greenberg

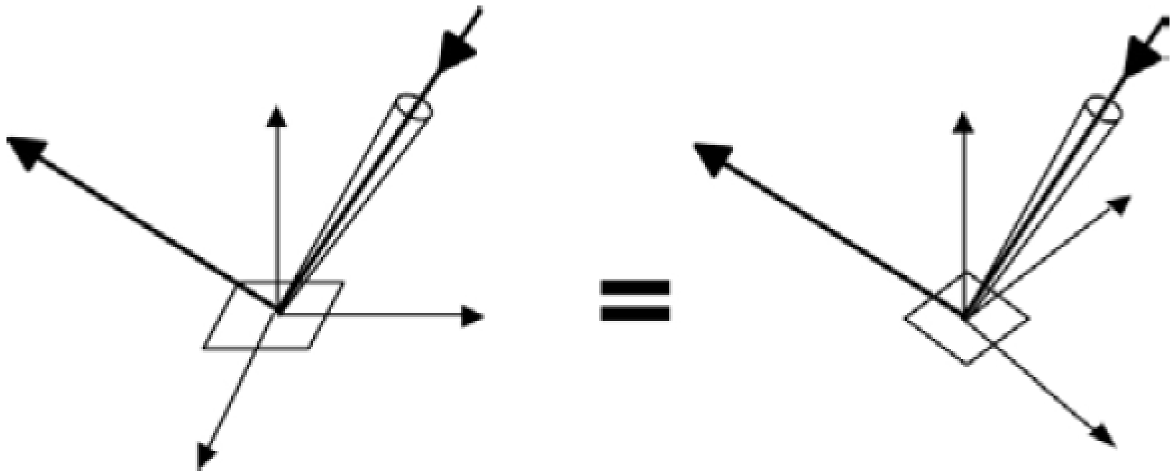
2. Reciprocity principle $f_r(\omega_r \rightarrow \omega_i) = f_r(\omega_i \rightarrow \omega_r)$



Properties of BRDFs

3. Isotropic vs. anisotropic

$$f_r(\theta_i,\varphi_i;\theta_r,\varphi_r) = f_r(\theta_i,\theta_r,\varphi_r-\varphi_i)$$

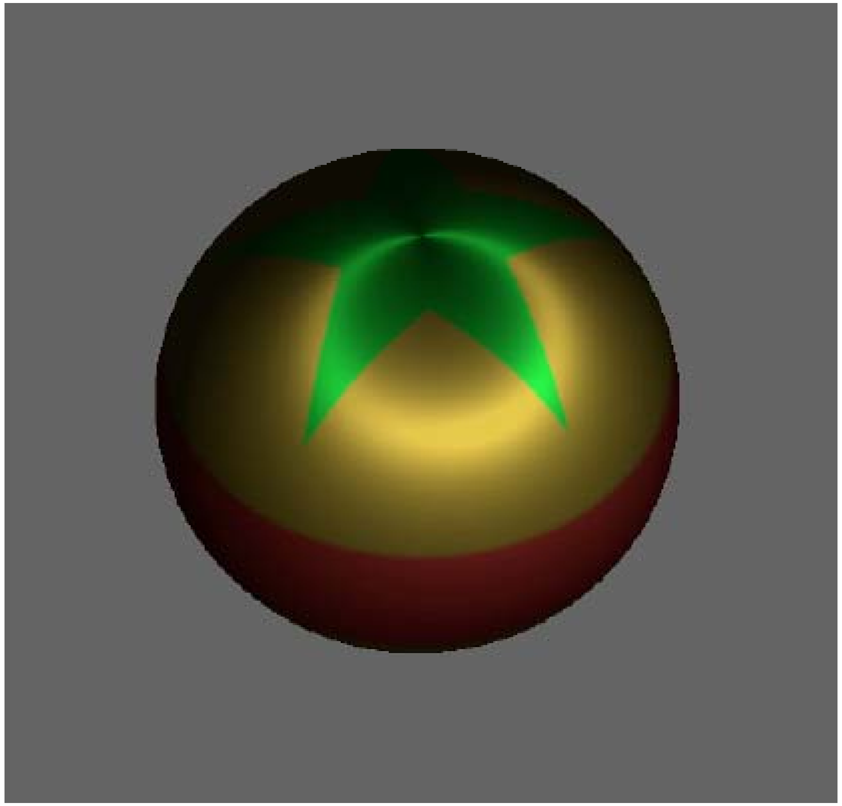


Reciprocity *and* isotropy

$$f_r(\theta_i,\theta_r,\varphi_r-\varphi_i) = f_r(\theta_r,\theta_i,\varphi_i-\varphi_r) = f_r(\theta_i,\theta_r,|\varphi_r-\varphi_i|)$$

4. Energy conservation $\int_\Omega f_r(\omega_o,\omega_i)\cos\theta_i d\omega_i \le 1$

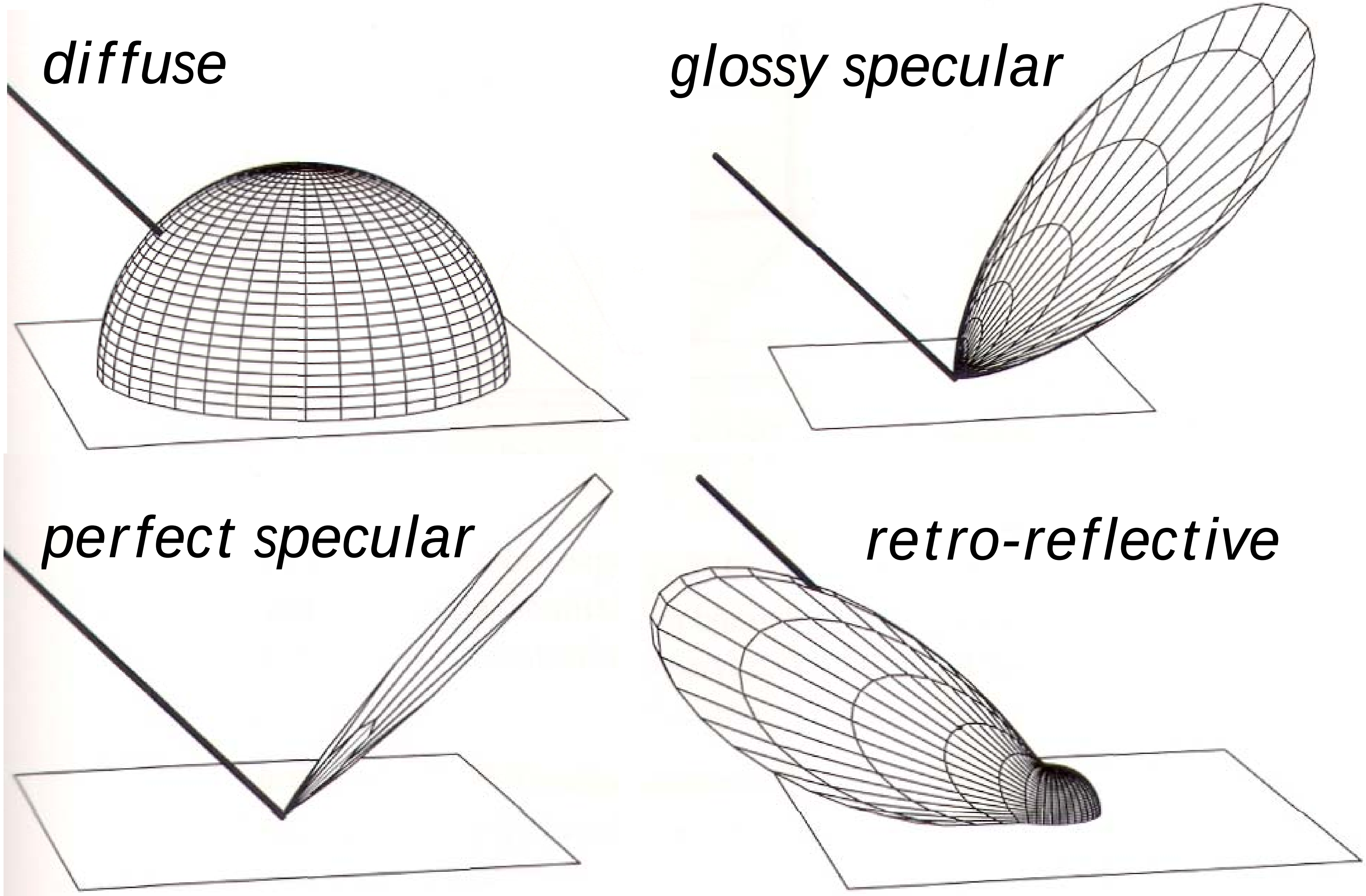
Isotropic and anisotropic



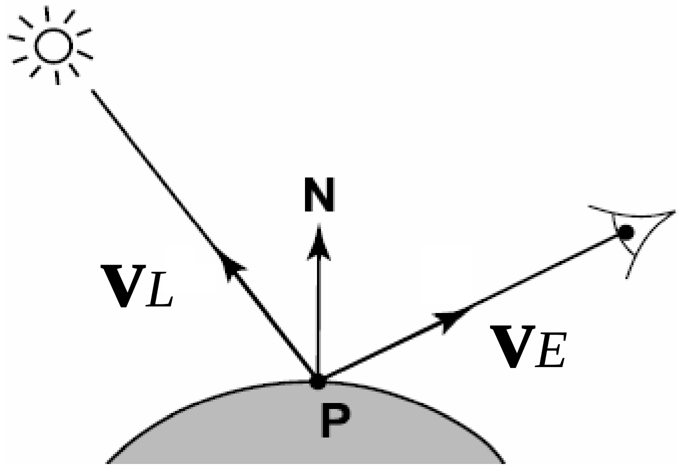
Surface reflection models

- **Measured data:** usually described in tabular form or coefficients of a set of basis functions
- **Phenomenological models**: *qualitative* approach; models with intuitive parameters
- Simulation: simulates light scattering from microgeometry and known reflectance properties
- Physical optics: solve Maxwell's equation
- **Geometric optics**: microfacet models

Reflection categories



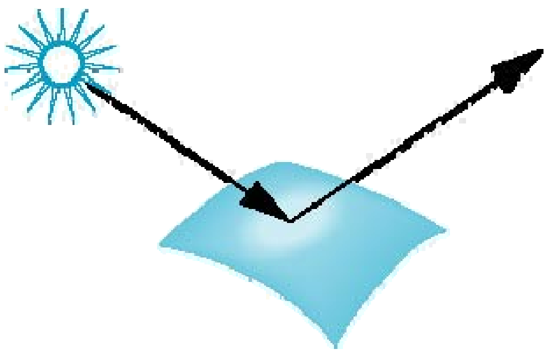
Setup



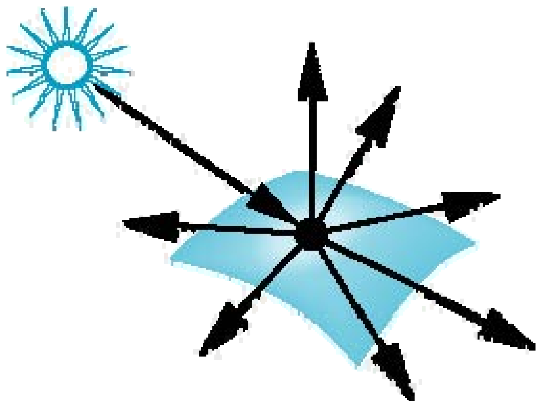
- Point **P** on a surface through a pixel **p**
- Normal **N** at **P**
- Lighting direction $\mathbf{v}_L$
- Viewing direction $\mathbf{v}_E$
- Compute color **L** for pixel **p**

Surface types

- The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflected the light
- A very rough surface scatters light in all directions



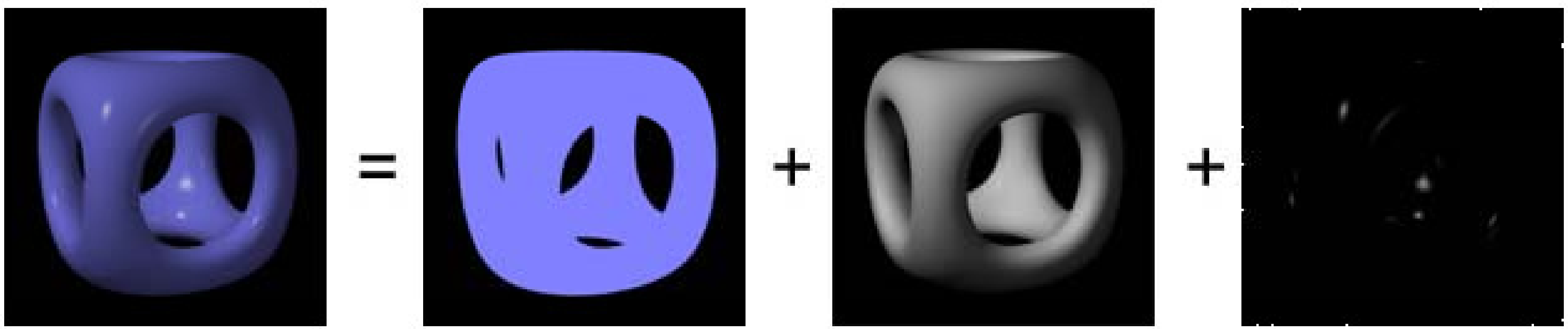
smooth surface



rough surface

Basics of local shading

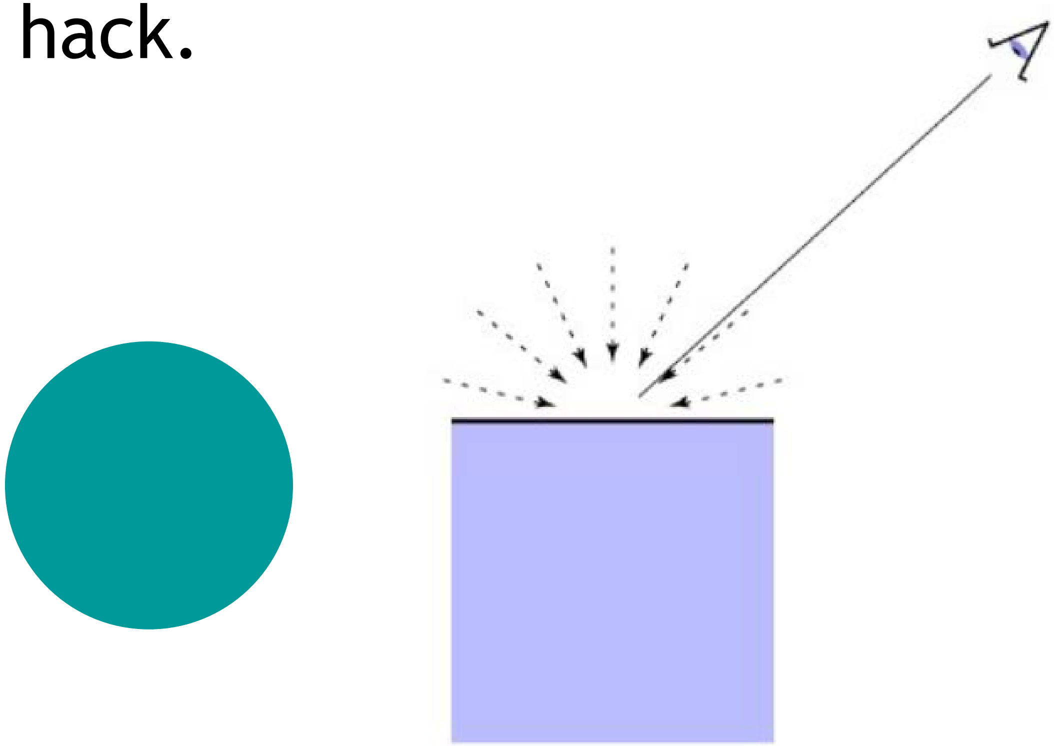
- Diffuse reflection
  - light goes everywhere; colored by object color
- Specular reflection
  - happens only near mirror configuration; usually white
- Ambient reflection
  - constant accounted for other source of illumination



color and ambient   diffuse   specularity

Ambient shading

- add constant color to account for disregarded illumination and fill in black shadows; a cheap hack.



ambient light

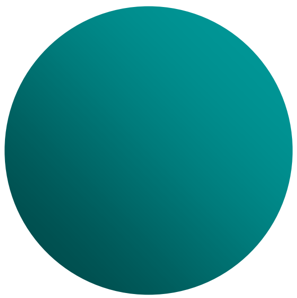
$$L_a = k_a I_a$$

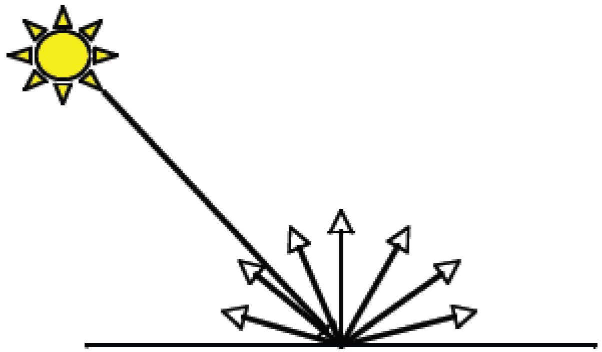
ambient coefficient

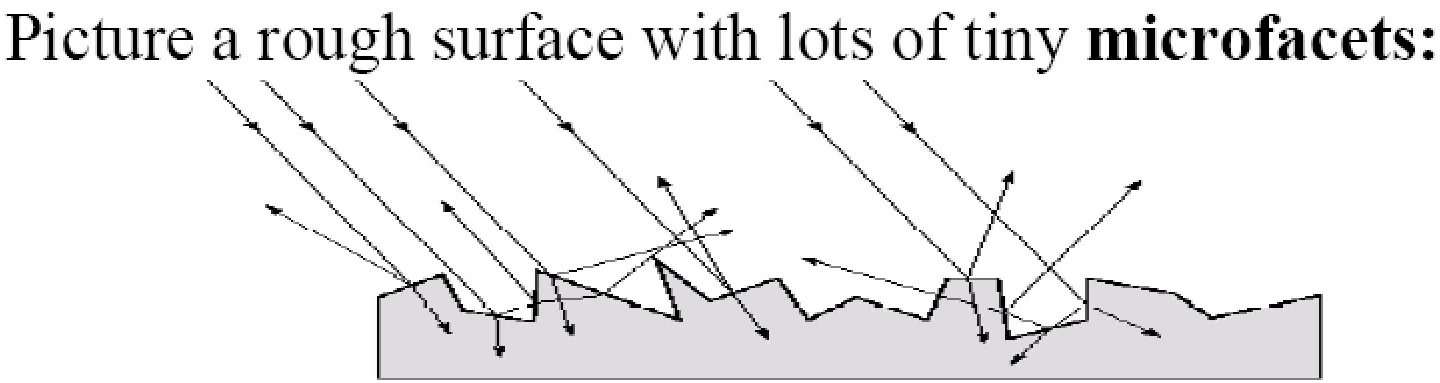
reflected ambient light

Diffuse shading

- Assume light reflects equally in all directions
  - Therefore surface looks same color from all views; "view independent"

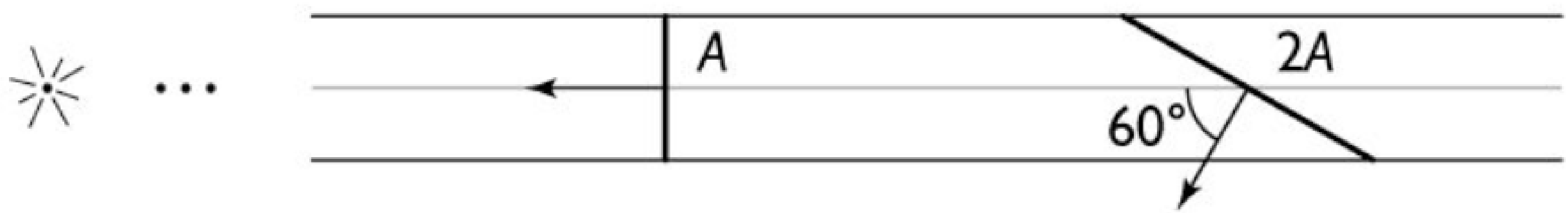






Diffuse shading

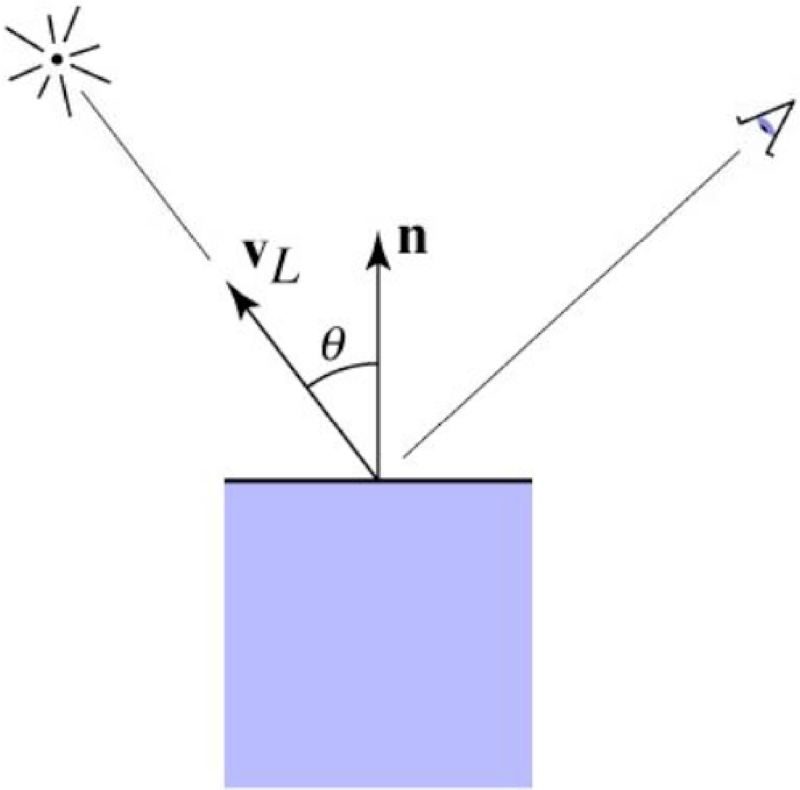
- Illumination on an oblique surface is less than on a normal one (Lambertian cosine law)



  - Generally, illumination falls off as $\cos\theta$

Diffuse shading (Gouraud 1971)

- Applies to *diffuse*, *Lambertian* or *matte* surfaces



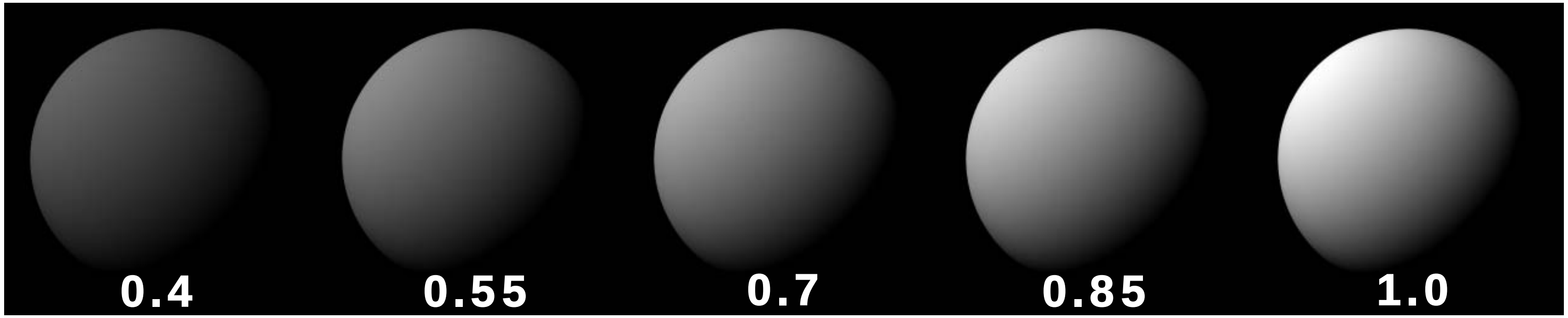
illumination from source

$$L_d = k_d \, I \, \max(0, \mathbf{n} \cdot \mathbf{v}_L)$$

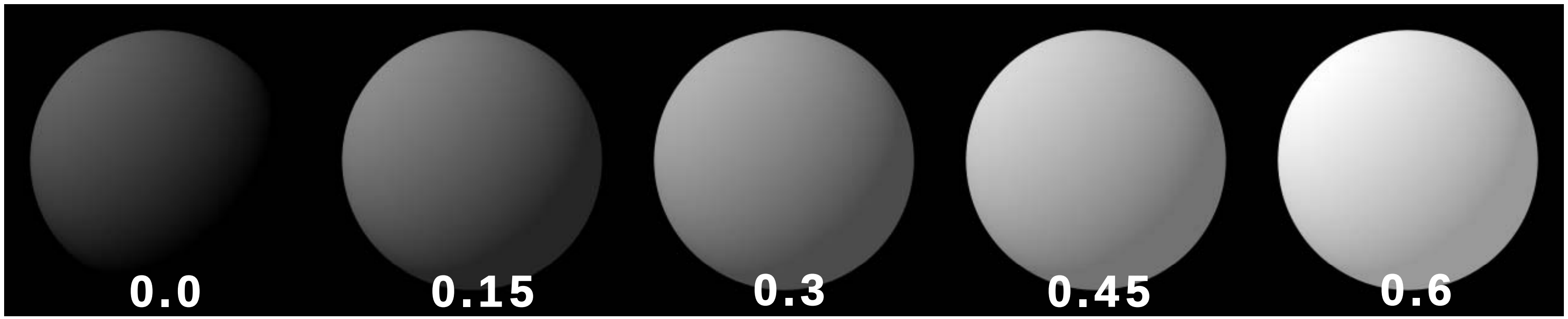
diffusely reflected light

diffuse coefficient (albedo)

Diffuse shading



diffuse-reflection model with different k_d

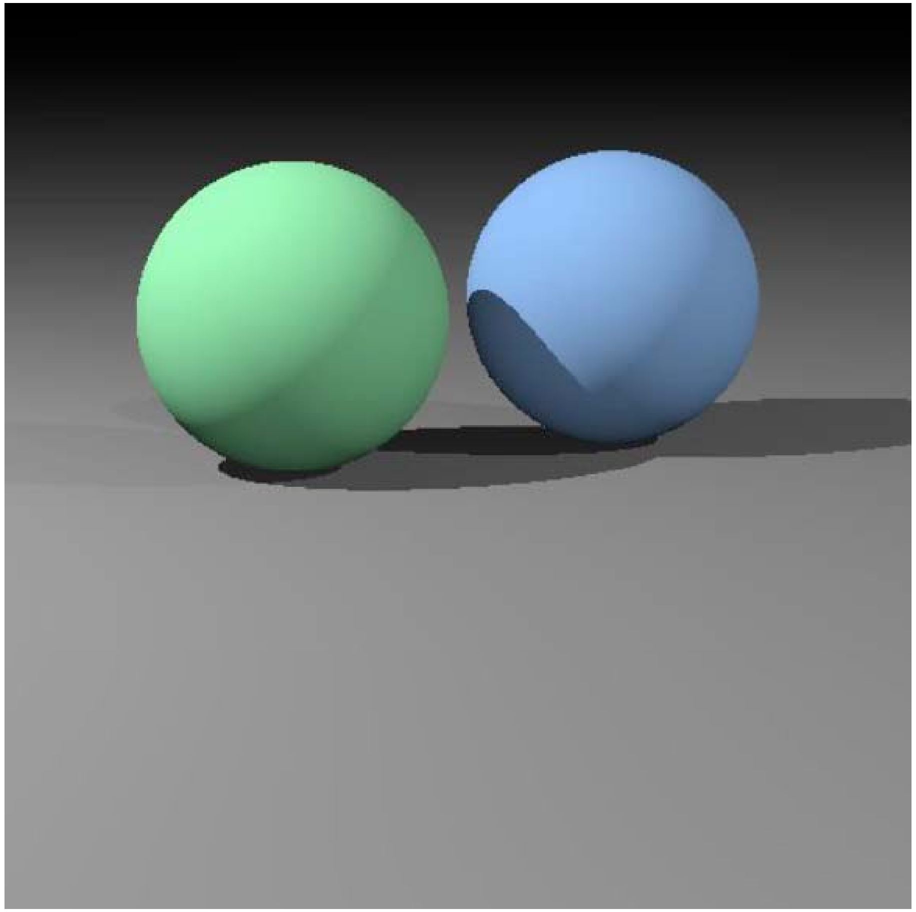


ambient and diffuse-reflection model with different k_a

and $I_a = I_p = 1.0, k_d = 0.4$

Diffuse shading

For color objects, apply the formula for each color channel separately

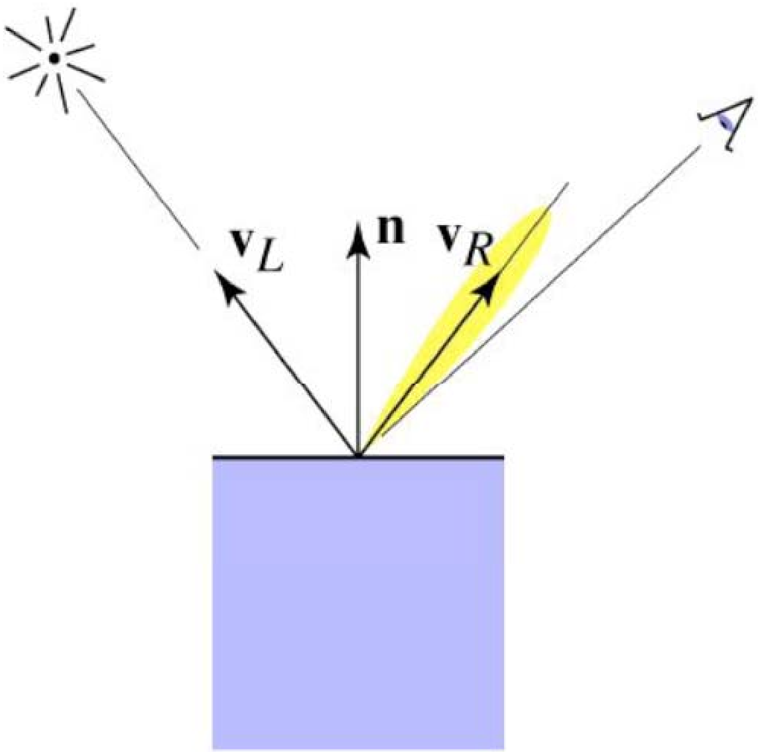


Specular shading

- Some surfaces have highlights, mirror like reflection; view direction dependent; especially for smooth shinny surfaces

Specular shading (Phong 1975)

- Also known as *glossy*, *rough specular* and *directional diffuse* reflection

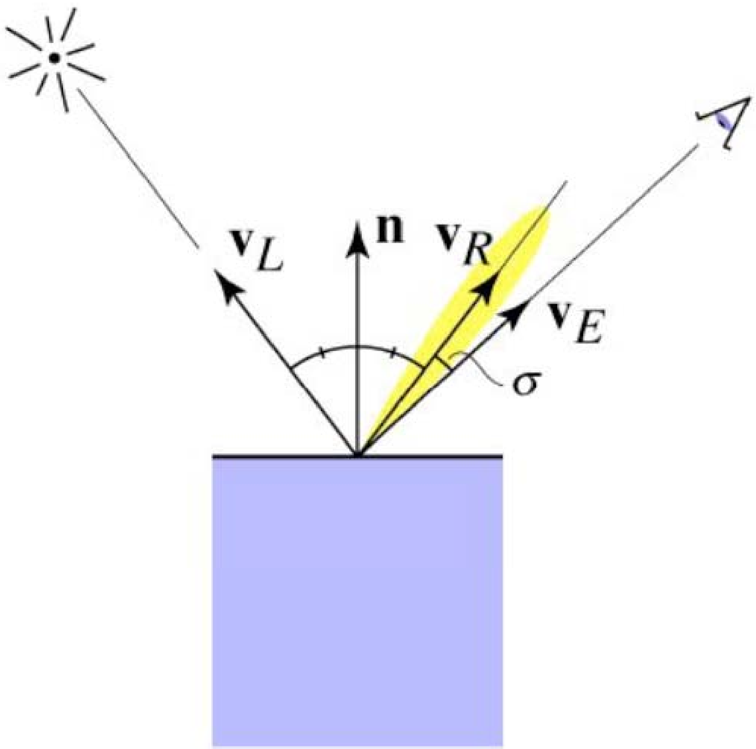


$$\mathbf{v}_R = \mathbf{v}_L + 2((\mathbf{n}\cdot\mathbf{v}_L)\mathbf{n} - \mathbf{v}_L)$$
$$= 2(\mathbf{n}\cdot\mathbf{v}_L)\mathbf{n} - \mathbf{v}_L$$

Bui-Tuong Phong 1942-1975
1971 attend U. Utah
1973 Phd
1975 Stanford faculty

Specular shading

- Fall off gradually from the perfect reflection direction



$$\mathbf{v}_R = \mathbf{v}_L + 2((\mathbf{n}\cdot\mathbf{v}_L)\mathbf{n} - \mathbf{v}_L)$$
$$= 2(\mathbf{n}\cdot\mathbf{v}_L)\mathbf{n} - \mathbf{v}_L$$

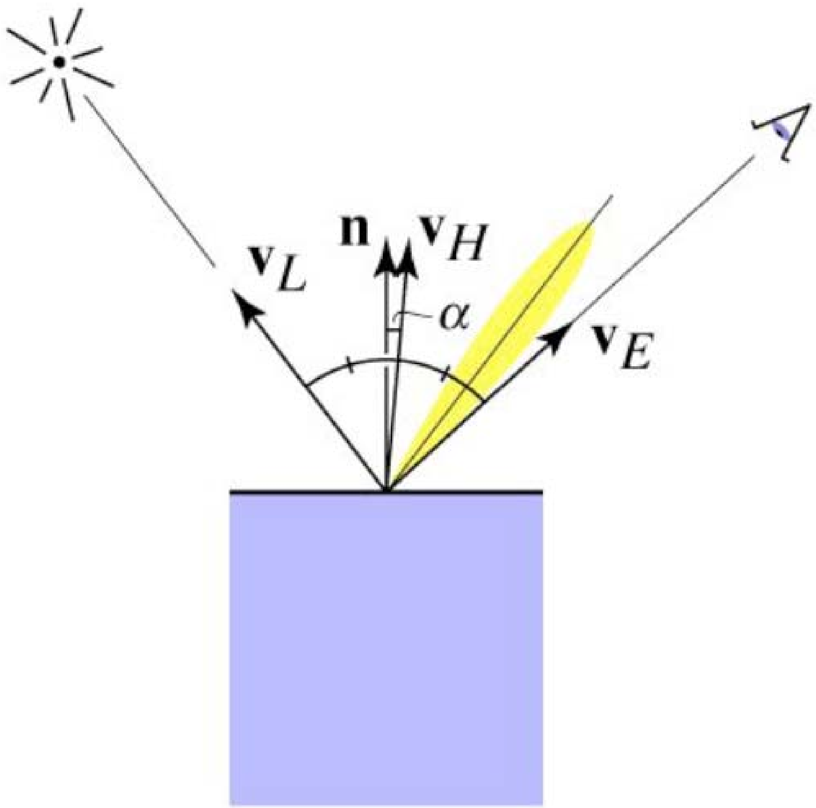
$$L_s = k_s\, I \max(0, \cos\sigma)^n$$
$$= k_s\, I \max(0, \mathbf{v}_E\cdot\mathbf{v}_R)^n$$

L_s: specularly reflected light

k_s: specular coefficient

Phong variant: Blinn-Phong

- Rather than computing reflection directly; just compare to normal bisection property.



$$\mathbf{v}_H = \text{bisector}(\mathbf{v}_L, \mathbf{v}_E) = \frac{(\mathbf{v}_L + \mathbf{v}_E)}{\|\mathbf{v}_L + \mathbf{v}_E\|}$$

$$L_s = k_s\, I \max(0, \cos\alpha)^n = k_s\, I \max(0, \mathbf{n} \cdot \mathbf{v}_H)^n$$

Blinn-Phong

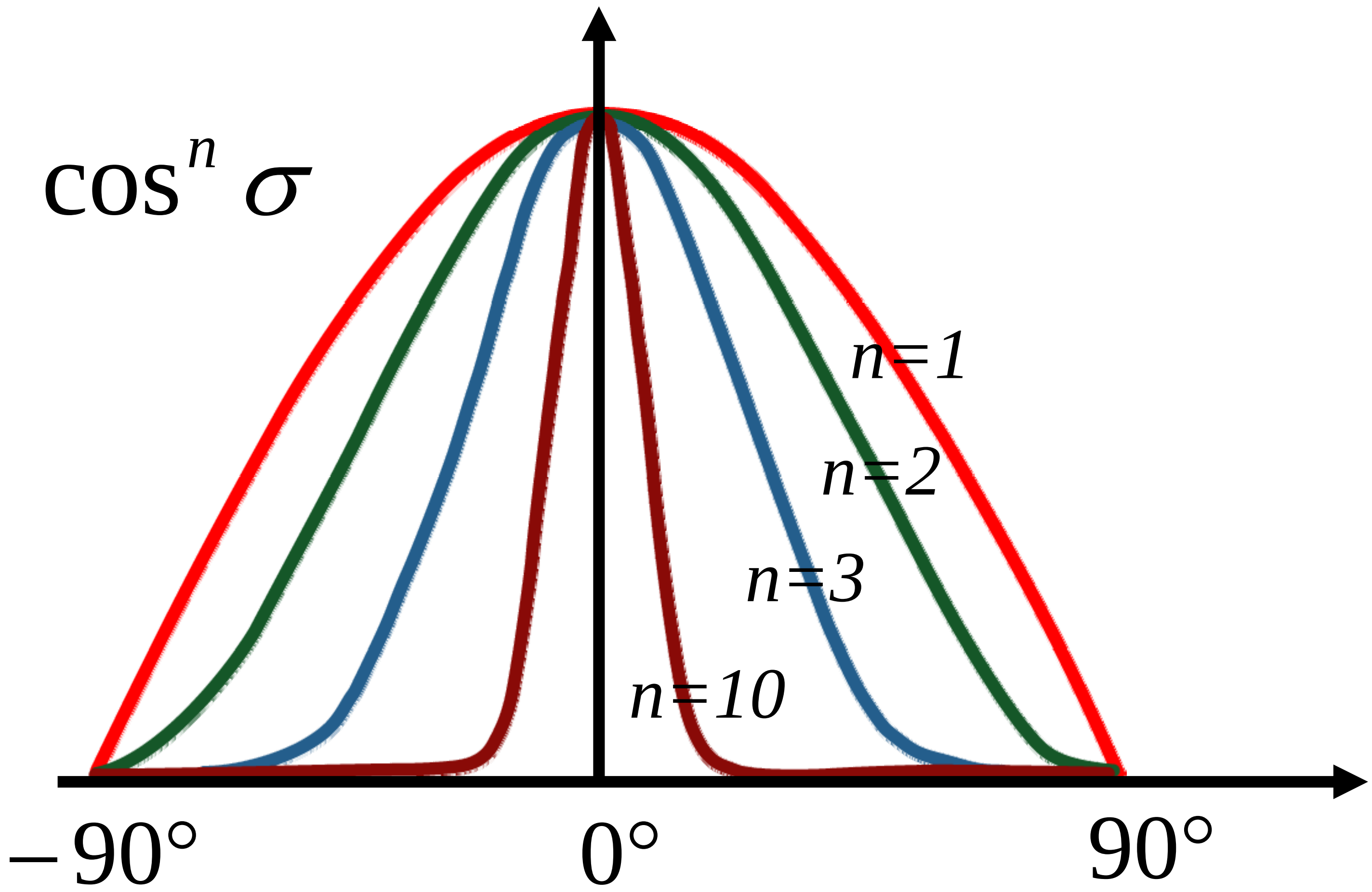
- One can prove that, for small σ

$$\cos^n \sigma = \cos^{4n} \alpha$$

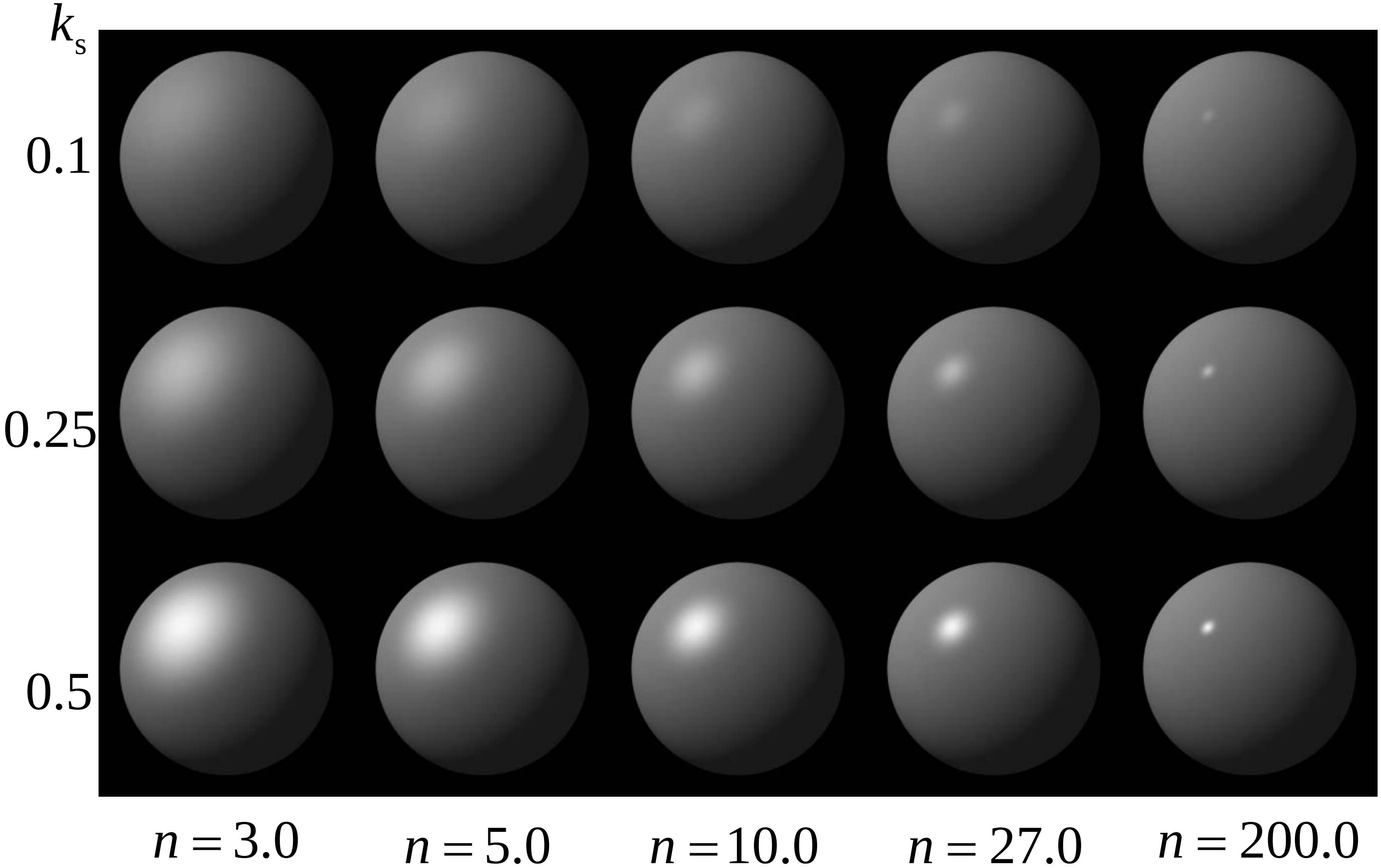
- Blinn-Phong model is
  - Potentially faster (especially for directional light and orthographic projection)
  - More physically-based (closer to Torrance-Sparrow model than Phong model)

Specular shading

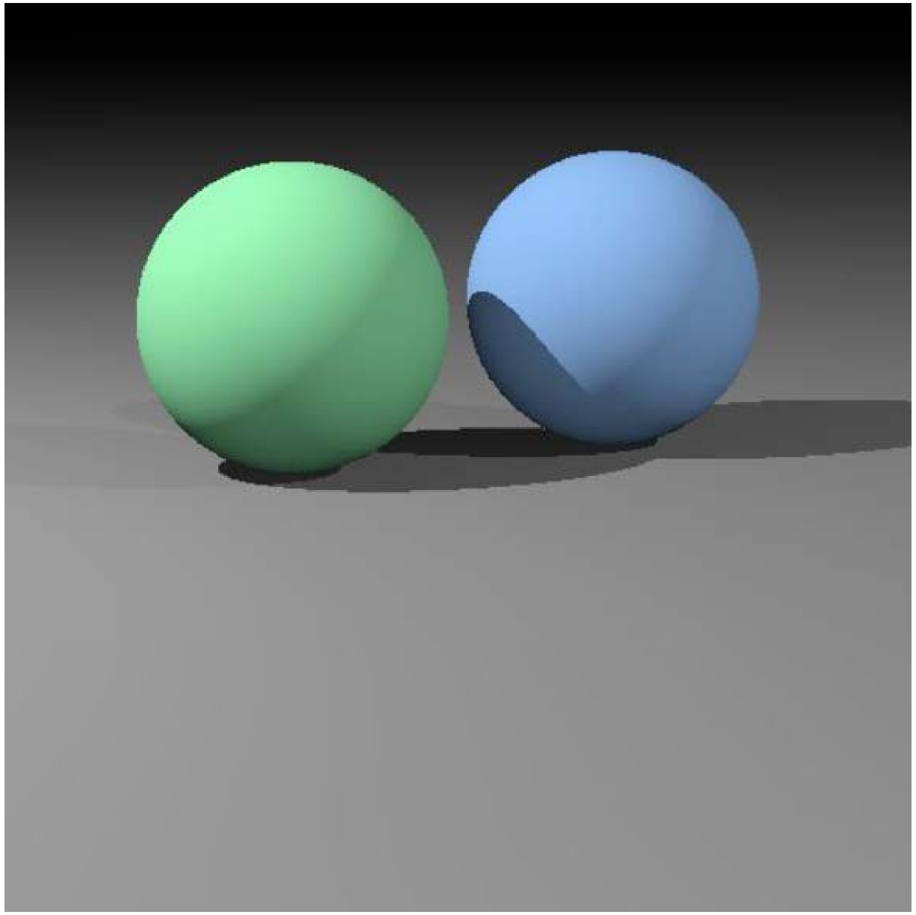
- Increasing n narrows the lobe



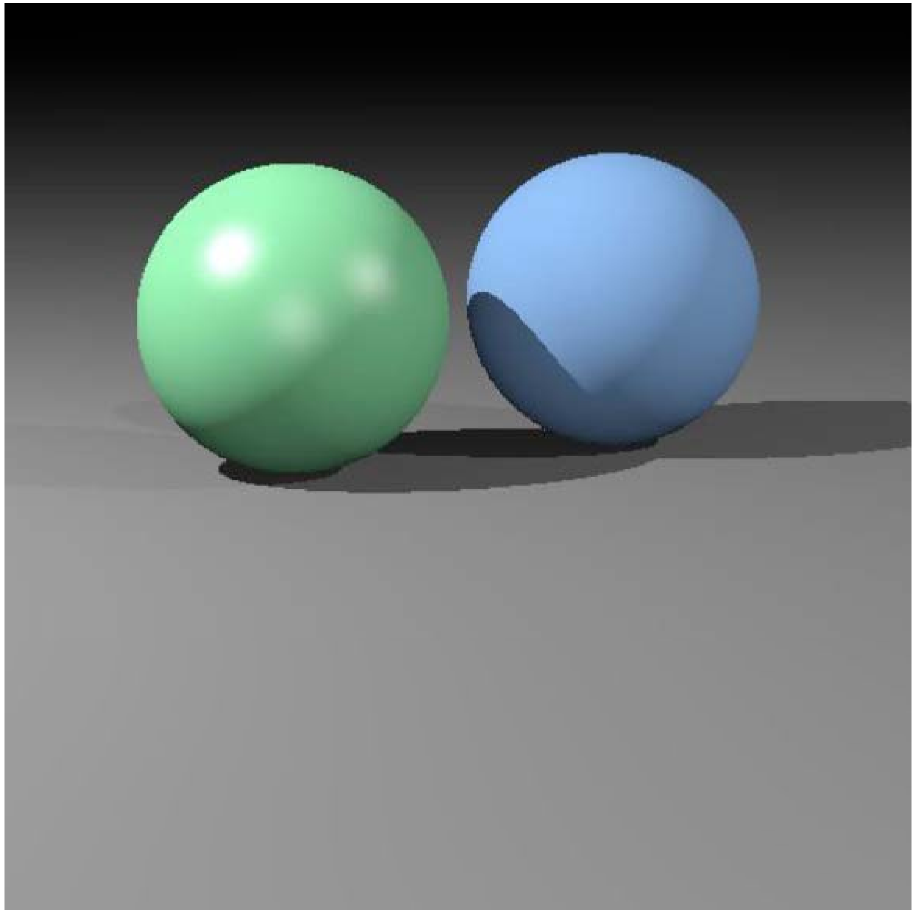
Specular shading



Specular shading



diffuse



diffuse + specular

Put it all together

- Include ambient, diffuse and specular

$$
\begin{aligned}
L &= L_a + L_d + L_s \\
&= k_a I_a + I\left(k_d \max(0, \mathbf{n} \cdot \mathbf{v}_L) + k_s \max(0, \mathbf{n} \cdot \mathbf{v}_H)^n\right)
\end{aligned}
$$

- Sum over many lights

$$
\begin{aligned}
L &= L_a + \sum_i (L_d)_i + (L_s)_i \\
&= k_a I_a + \sum_i I_i \left(k_d \max(0, \mathbf{n} \cdot (\mathbf{v}_L)_i) + k_s \max(0, \mathbf{n} \cdot (\mathbf{v}_H)_i)^n\right)
\end{aligned}
$$

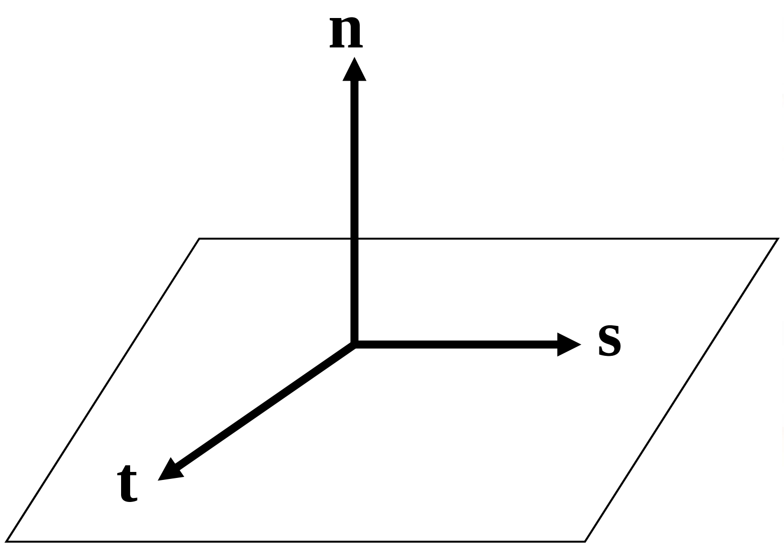
Knoll’s class on local shading

Reflection models

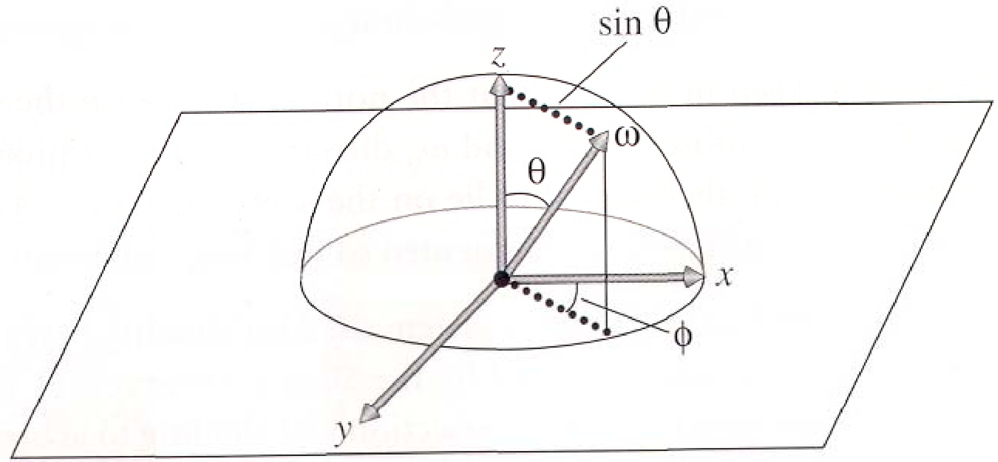
- BRDF/BTDF/BSDF
- Scattering from realistic surfaces is best described as a mixture of multiple BRDFs and BTDFs.
- **`core/reflection.*`**
- Material = BSDF that combines multiple BRDFs and BTDFs. (chap. 9)
- Textures = reflection and transmission properties that vary over the surface. (chap. 10)

Geometric setting

incident and outgoing directions are normalized and outward facing after being transformed into the local frame



local frame



$$\cos\theta = \omega_z, \qquad \sin\theta = \sqrt{1-\omega_z^2}$$

$$\cos\phi = \frac{\omega_x}{\sin\theta}, \quad \sin\phi = \frac{\omega_y}{\sin\theta}$$

BxDF

- **BxDFType**
  - **BSDF_REFLECTION**, **BSDF_TRANSMISSION**
  - **BSDF_DIFFUSE**, **BSDF_GLOSSY** (retro-reflective), **BSDF_SPECULAR**
- `Spectrum f(Vector &wo, Vector &wi)=0;`
- `Spectrum Sample_f(Vector &wo, Vector *wi, float u1, float u2, float *pdf);`

  used to find an incident direction for an outgoing direction; especially useful for reflection with a delta distribution
- `Spectrum rho(Vector &wo, int nSamples, float *samples);`

hemispherical-directional reflectance; computed analytically or by sampling

$$\rho_{hd}(\omega_o) = \int_{\Omega} f_r(p, \omega_o, \omega_i) \,|\cos\theta_i|\, d\omega_i$$

- `Spectrum rho(int nSamples,float *samples1, float *sample2);`

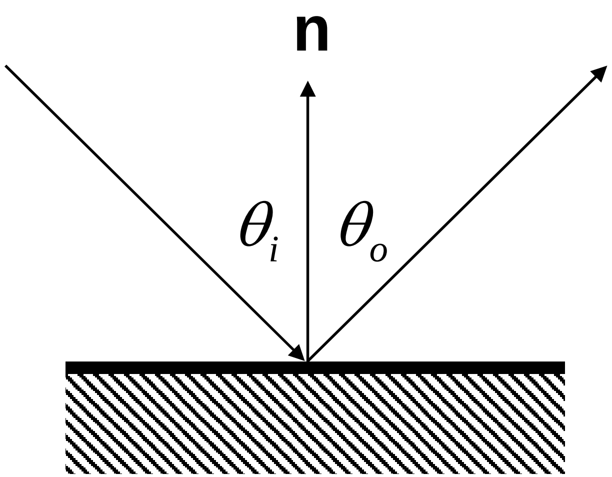
hemispherical-hemispherical reflectance

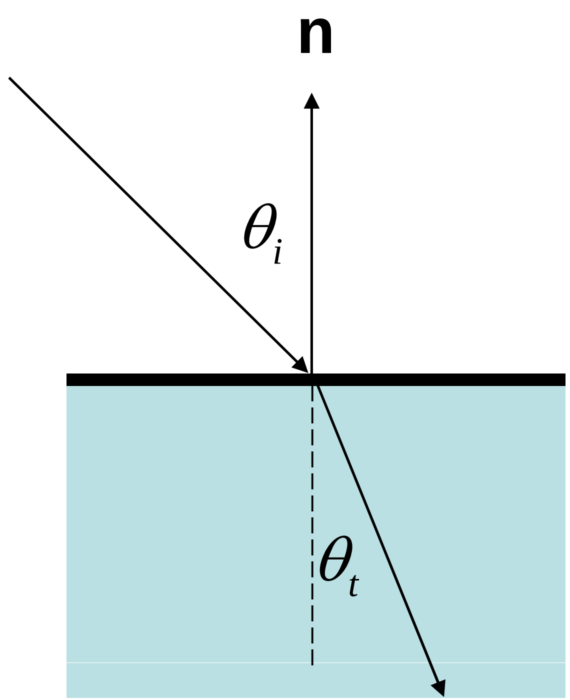
$$\rho_{hh} = \frac{1}{\pi} \int_{\Omega}\int_{\Omega} f_r(p, \omega_o, \omega_i) \,|\cos\theta_i \cos\theta_o|\, d\omega_i d\omega_o$$

Specular reflection and transmission

- Reflection: $\theta_i = \theta_o$
- Transmission: $\eta_i \sin\theta_i = \eta_t \sin\theta_t$ (Snell's law)

index of refraction *dispersion*





Fresnel reflectance

- Reflectivity and transmissiveness: fraction of incoming light that is reflected or transmitted; they are usually view dependent. Hence, the reflectivity is not a constant and should be corrected by *Fresnel equation*
- *Fresnel reflectance*
  for dielectrics

$$r_{\parallel} = \frac{\eta_t \cos\theta_i - \eta_i \cos\theta_t}{\eta_t \cos\theta_i + \eta_i \cos\theta_t}$$

$$r_{\perp} = \frac{\eta_i \cos\theta_i - \eta_t \cos\theta_t}{\eta_i \cos\theta_i + \eta_t \cos\theta_t}$$

- Assume light is unpolarized

$$F_r(\omega_i) = \frac{1}{2}\left(r_{\parallel}^2 + r_{\perp}^2\right)$$

$$F_t(\omega_i) = \left(1 - F_r(\omega_i)\right)$$

Indices of refraction

medium	Index of refraction
Vaccum	1.0
Air at sea level	1.00029
Ice	1.31
Water (20°C)	1.333
Fused quartz	1.46
Glass	1.5~1.6
Sapphire	1.77
Diamond	2.42

Fresnel reflectance

- *Fresnel reflectance* for conductors (no transmission)

index of refraction

absorption coefficient

$$r_{\parallel}^2 = \frac{\left(\eta^2 + k^2\right)\cos^2\theta_i - 2\eta\cos\theta_i + 1}{\left(\eta^2 + k^2\right)\cos^2\theta_i + 2\eta\cos\theta_i + 1}$$

$$r_{\perp}^2 = \frac{\left(\eta^2 + k^2\right) - 2\eta\cos\theta_i + \cos^2\theta_i}{\left(\eta^2 + k^2\right) + 2\eta\cos\theta_i + \cos^2\theta_i}$$

$$F_r(\omega_i) = \frac{1}{2}\left(r_{\parallel}^2 + r_{\perp}^2\right)$$

η and k for a few conductors

Object	η	k
Gold	0.370	2.820
Silver	0.177	3.638
Copper	0.617	2.630
Steel	2.485	3.433

- However, for most conductors, these coefficients are unknown. Approximations are used to find plausible values for these quantities if reflectance at the normal incidence is known.

Fresnel class

```
class Fresnel {
public:
  virtual Spectrum Evaluate(float cosi) const = 0;
};
class FresnelConductor : public Fresnel {
public:
  FresnelConductor(Spectrum &e, Spectrum &kk)
      : eta(e), k(kk) {}
private:
  Spectrum eta, k;
};
class FresnelDielectric : public Fresnel {
public:
  FresnelDielectric(float ei, float et) {
      eta_i = ei; eta_t = et; }
private:
  float eta_i, eta_t;
};
```

Evaluate directly implements Fresnel formula for conductor

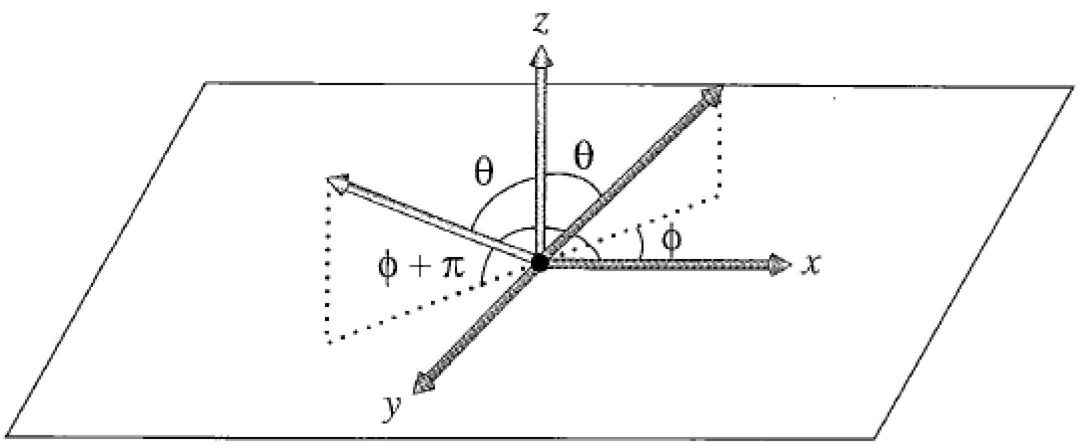
Evaluate directly implements Fresnel formula for dielectric

Specular reflection

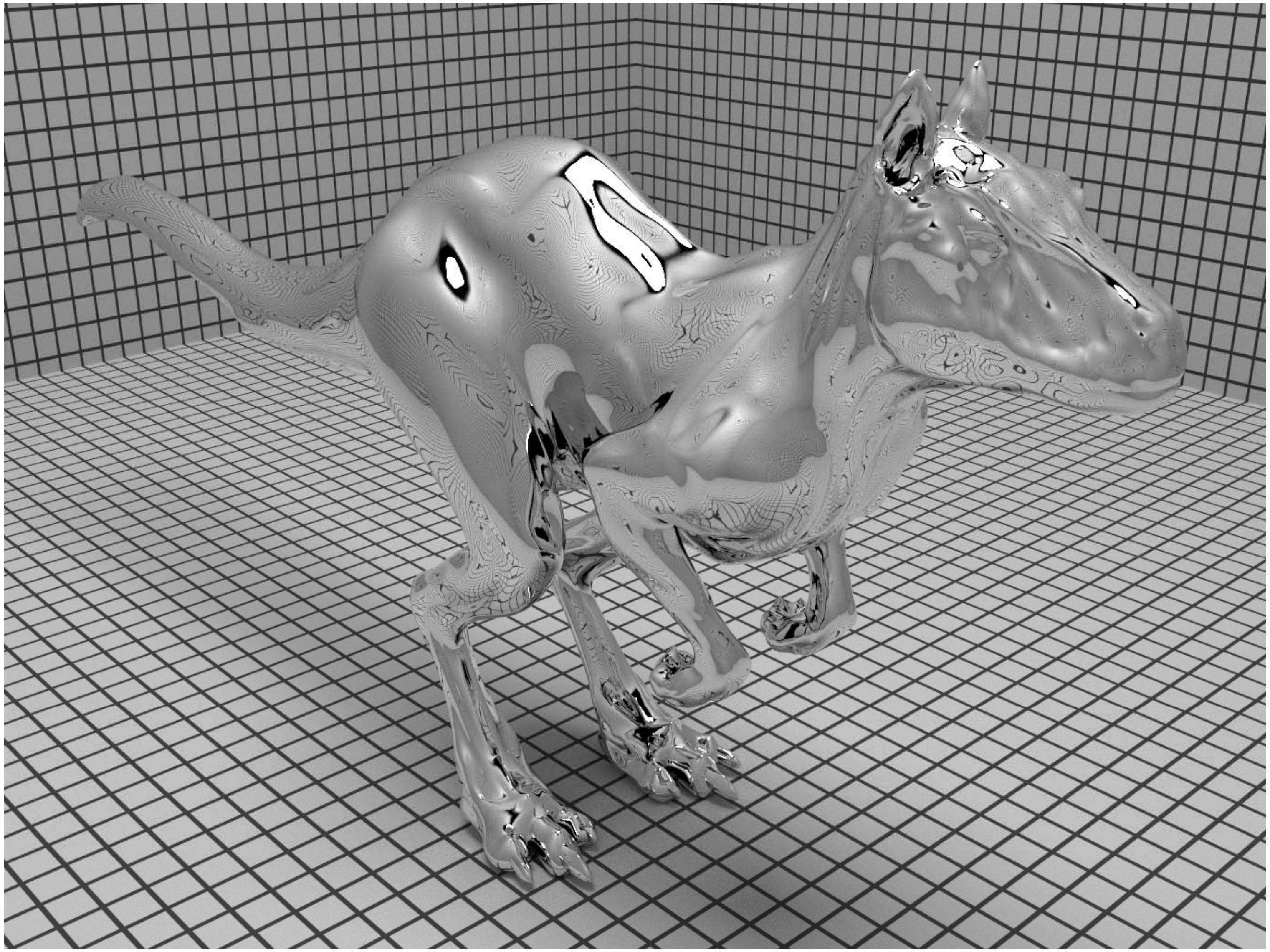
```
class SpecularReflection : public BxDF {
public:
  SpecularReflection(const Spectrum &r, Fresnel *f)
  : BxDF(BxDFType(BSDF_REFLECTION | BSDF_SPECULAR)),
    R(r), fresnel(f) { }
  Spectrum f(const Vector &, const Vector &) const {
    return Spectrum(0.);
  }
  Spectrum Sample_f(const Vector &wo, Vector *wi,
               float u1, float u2, float *pdf) const;
  float Pdf(const Vector &wo, const Vector &wi) const{
    return 0.;
  }
private:
  Spectrum R;
  Fresnel *fresnel;
};
```

Specular reflection

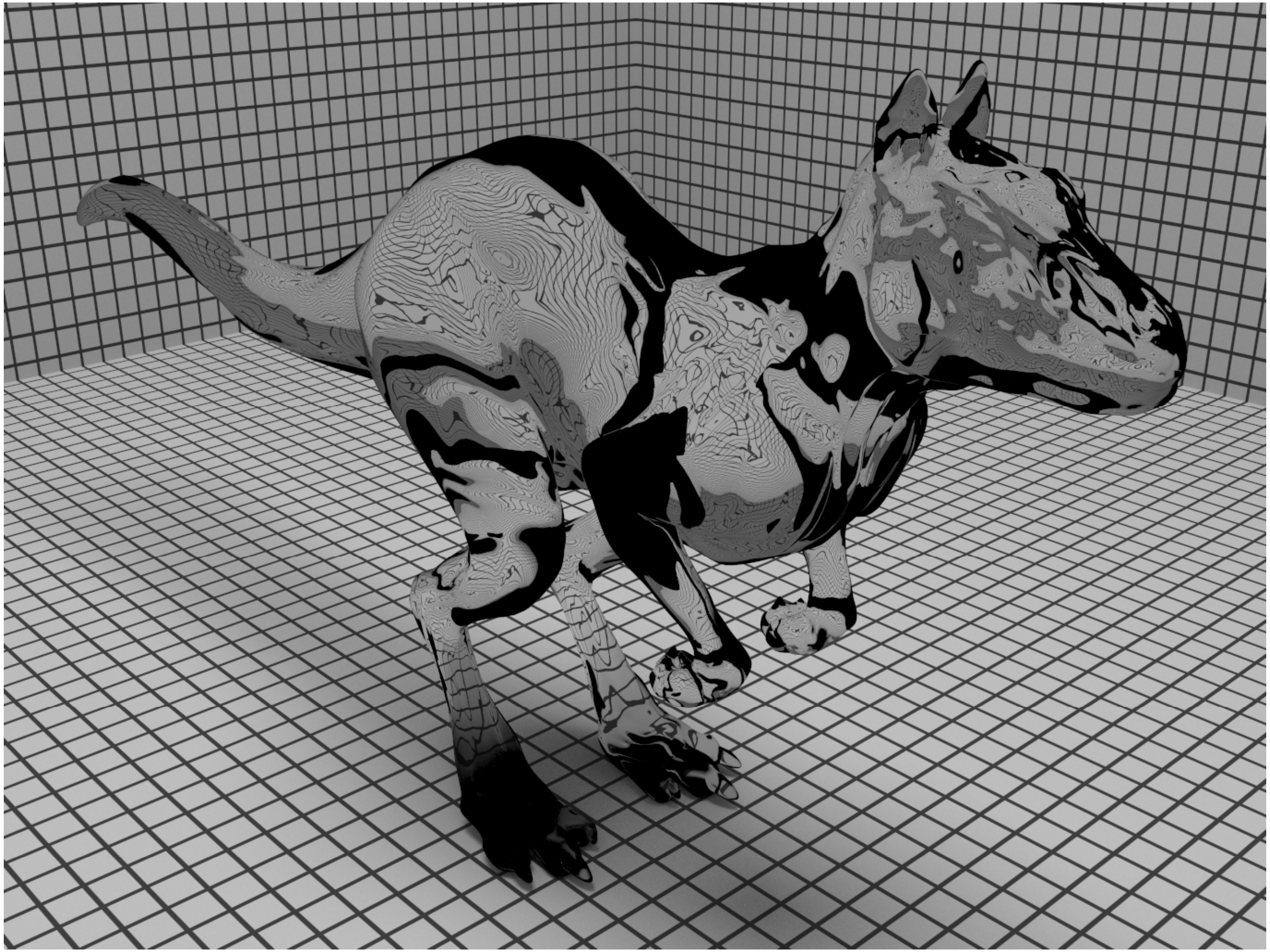
```
Spectrum SpecularReflection::Sample_f(Vector &wo,
  Vector *wi, float u1, float u2, float *pdf) const{
  // Compute perfect specular reflection direction
  *wi = Vector(-wo.x, -wo.y, wo.z);
  *pdf = 1.f;
  return fresnel->Evaluate(CosTheta(wo)) * R /
         fabsf(CosTheta(*wi));
}
```



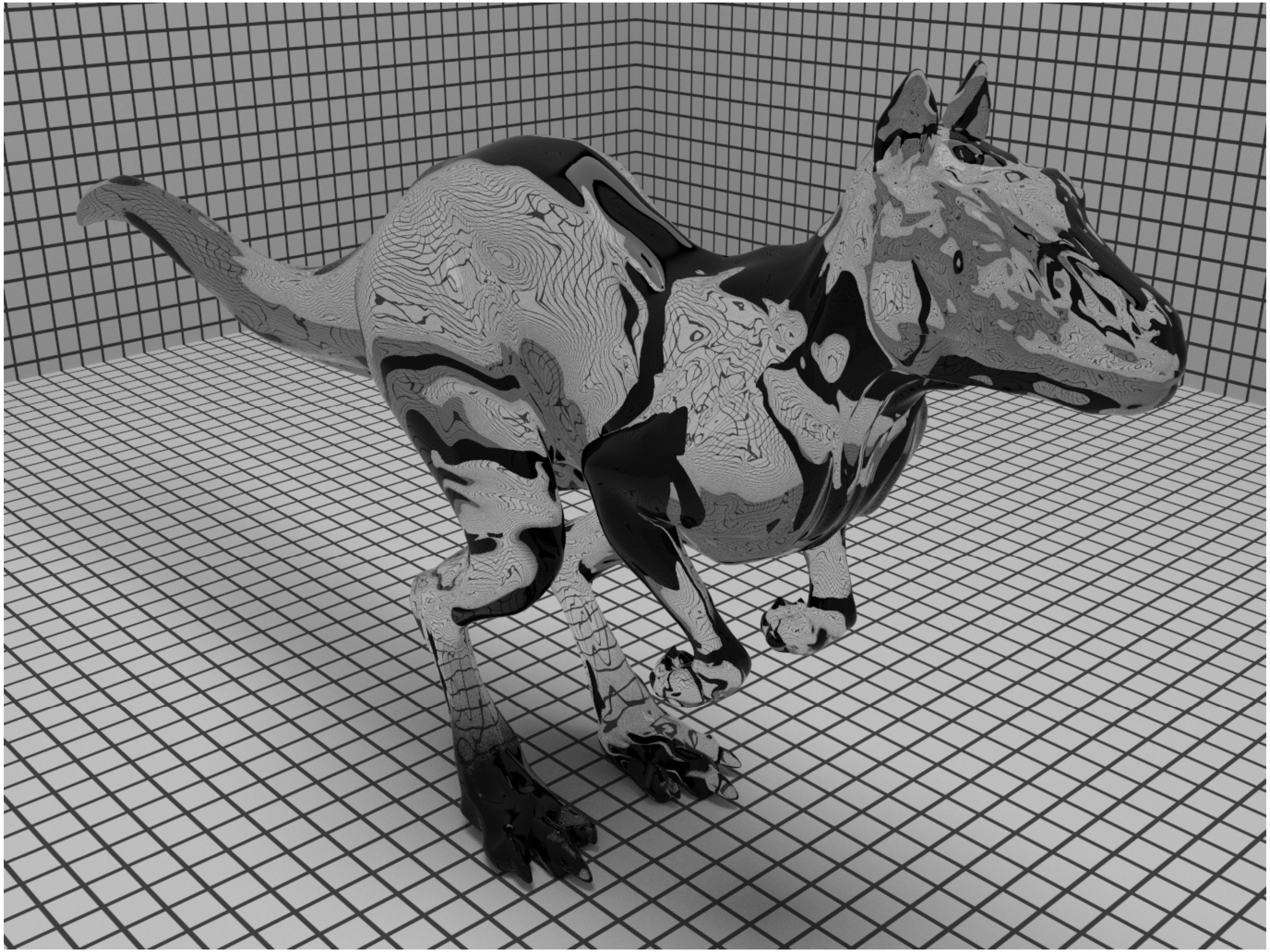
Perfect specular reflection



Perfect specular transmission



Fresnel modulation



Lambertian reflection

- It is not physically feasible, but provides a good approximation to many real-world surfaces.

```
class COREDLL Lambertian : public BxDF {
public:
  Lambertian(Spectrum &reflectance)
  : BxDF(BxDFType(BSDF_REFLECTION | BSDF_DIFFUSE)),
    R(reflectance), RoverPI(reflectance * INV_PI) {}
  Spectrum f(Vector &wo, Vector &wi) {return RoverPI}
  Spectrum rho(Vector &, int, float *) { return R; }
  Spectrum rho(int, float *) { return R; }
private:
  Spectrum R, RoverPI;
};
```

Derivations

$$\rho_{hh} = \frac{1}{\pi} \int_{\Omega} \int_{\Omega} f_r(p, \omega_o, \omega_i) \,|\cos\theta_i \cos\theta_o|\, d\omega_i d\omega_o$$

Derivations

$$\rho_{hh} = \frac{1}{\pi} \int_\Omega \int_\Omega f_r(p, \omega_o, \omega_i) \, | \cos\theta_i \cos\theta_o \, | \, d\omega_i d\omega_o$$

$$R = \frac{1}{\pi} \int_\Omega \int_\Omega c \, | \cos\theta_i \cos\theta_o \, | \, d\omega_i d\omega_o$$

$$R = \frac{c}{\pi} \cdot \int_\Omega \cos\theta_i d\omega_i \cdot \int_\Omega \cos\theta_o d\omega_o = c\pi$$

$$c = \frac{R}{\pi}$$

$$\int_\Omega \cos\theta_i d\omega_i = \int_0^{2\pi} \int_0^{\pi/2} \cos\theta_i \sin\theta_i d\theta_i d\phi_i$$

$$= \int_0^{2\pi} d\phi_i \int_0^{\pi/2} \cos\theta_i \sin\theta_i d\theta_i$$

$$= 2\pi \int_0^{\pi/2} \tfrac{1}{2} \sin(2\theta_i) \tfrac{1}{2} d(2\theta_i)$$

$$= \frac{\pi}{2} \cdot -\cos(2\theta_i) \Big|_0^{\pi/2} = \pi$$

Derivations

$$\rho_{hd}(\omega_o) = \int_{\Omega} f_r(p, \omega_o, \omega_i) |\cos\theta_i| d\omega_i$$

Derivations

$$\rho_{hd}(\omega_o) = \int_{\Omega} f_r(p, \omega_o, \omega_i) \,|\cos\theta_i|\, d\omega_i$$

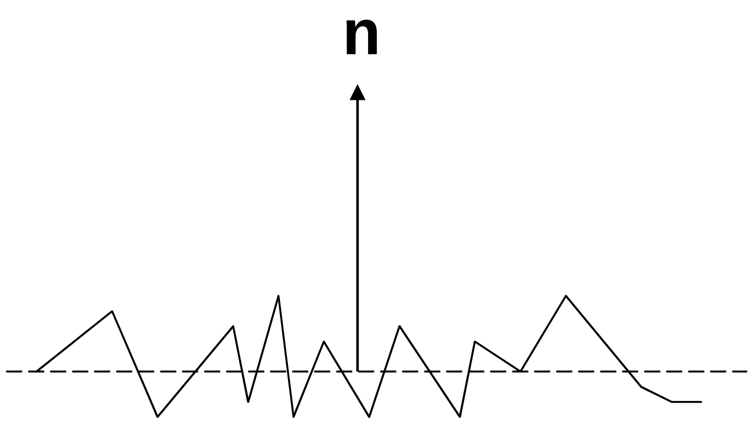
$$= \int_{\Omega} \frac{R}{\pi} \cos\theta_i d\omega_i$$

$$= \frac{R}{\pi} \int_{\Omega} \cos\theta_i d\omega_i$$

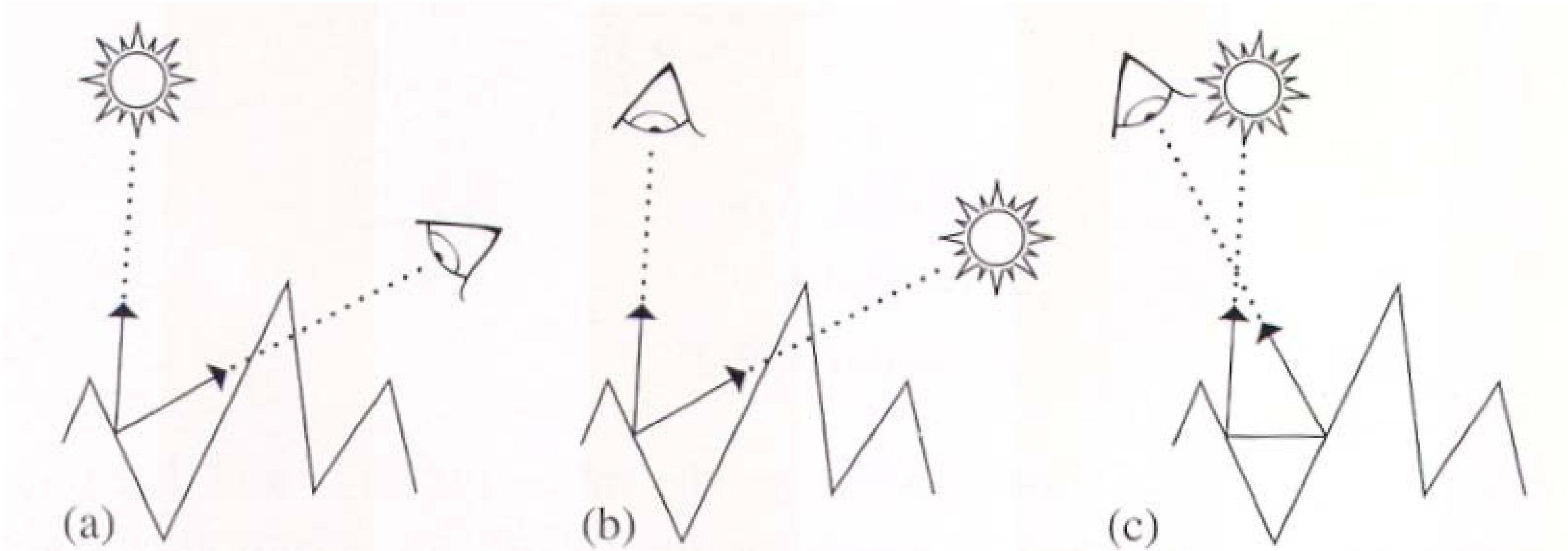
$$= \frac{R}{\pi} \cdot \pi = R$$

Microfacet models

- Rough surfaces can be modeled as a collection of small microfacets. Their aggregate behavior determines the scattering.
- Two components: distribution of microfacets and how light scatters from individual microfacet → closed-form BRDF expression



Important geometric effects to consider



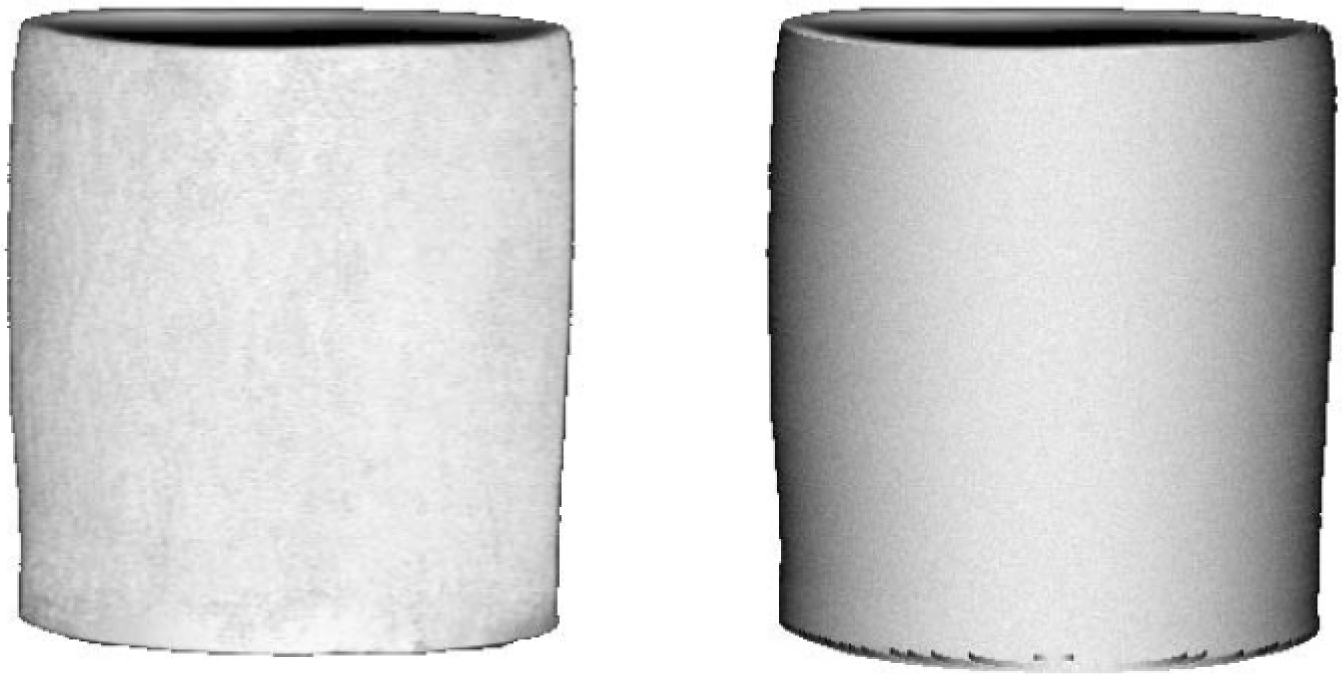
masking shadowing interreflection

Most microfacet models assume that all microfacets make up symmetric V-shaped grooves so that only neighboring microfacet needs to be considered. Particular models consider these effects with varying degrees of accuracy.

Oren-Nayar model

- Many real-world materials such as concrete, sand and cloth are not real Lambertian. Specifically, rough surfaces generally appear brighter as the illumination direction approaches the viewing direction.
- A collection of symmetric V-shaped perfect Lambertian grooves whose orientation angles follow a Gaussian distribution.
- Don’t have a closed-form solution, instead they used an approximation

Oren-Nayar model



(a) Real image (b) Lambertian model

Oren-Nayar model

standard deviation for Gaussian

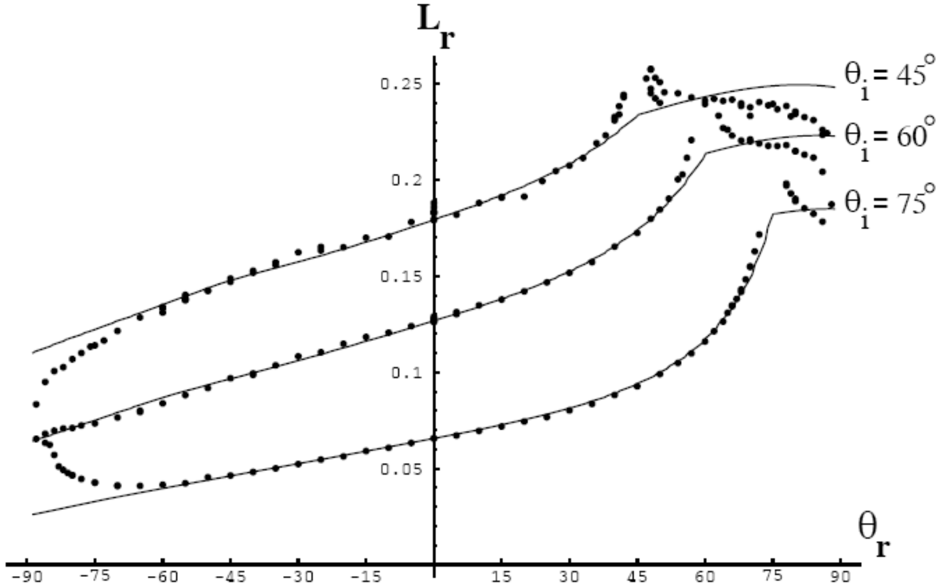
$$A = 1 - \frac{\sigma^2}{2(\sigma^2 + 0.33)}, \quad B = \frac{0.45\sigma^2}{\sigma^2 + 0.09}$$

$$\alpha = \max(\theta_i, \theta_o), \quad \beta = \min(\theta_i, \theta_o)$$

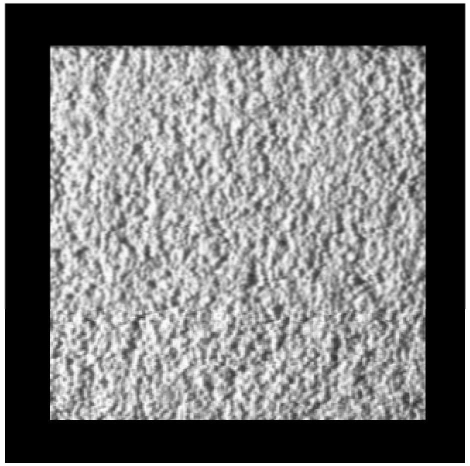
$$f_r(\omega_i, \omega_o) = \frac{\rho}{\pi}(A + B\max(0, \cos(\phi_i - \phi_o))\sin\alpha\tan\beta)$$

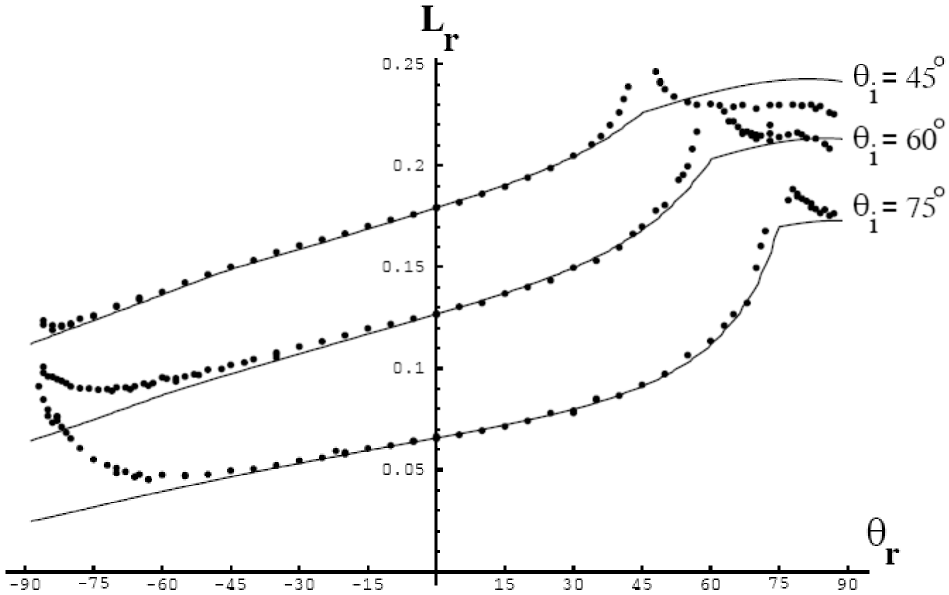
Oren-Nayar model

Sand Paper

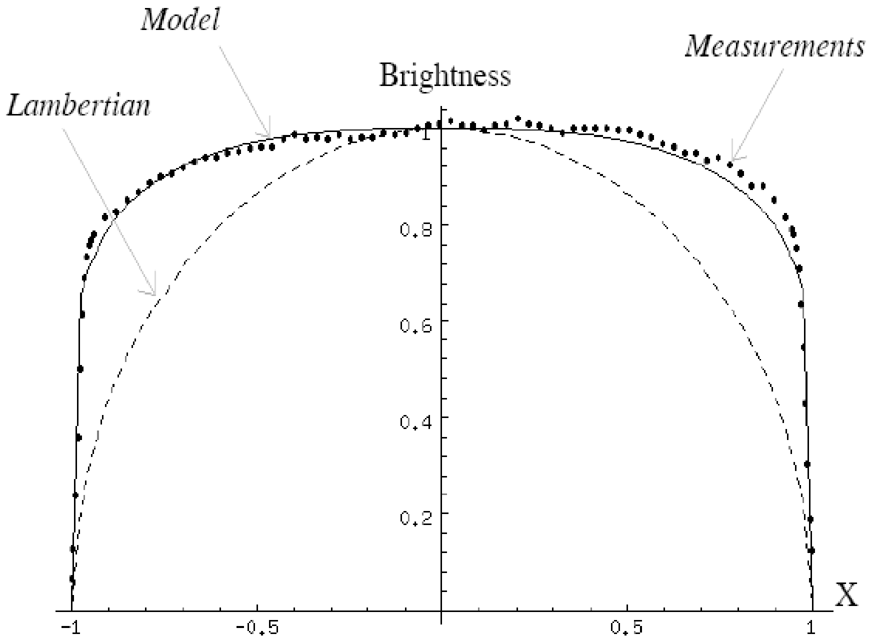


Sand

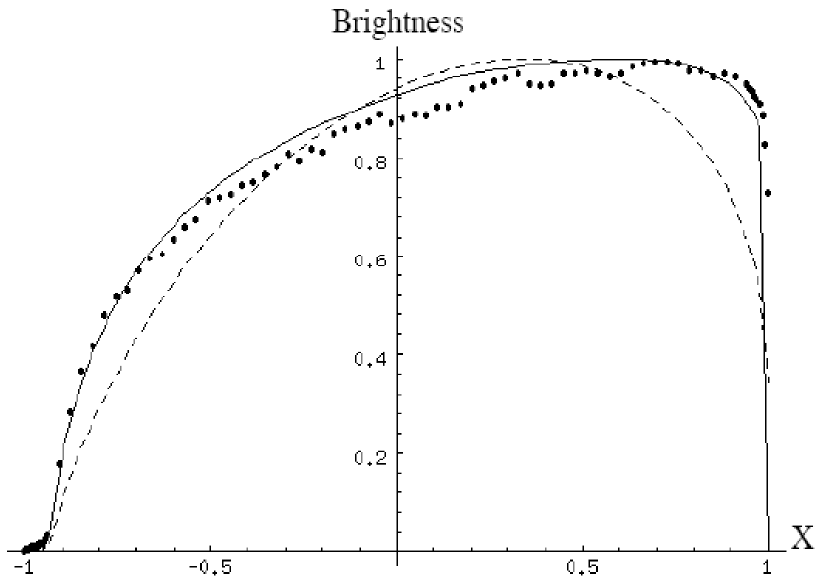




Oren-Nayar model



(a) $\theta_i = 0°$



(b) $\theta_i = 20°$

Oren-Nayar model

(a) Real image (b) Lambertian model (c) Proposed model

Oren-Nayar model

```
class OrenNayar : public BxDF {
public:
  Spectrum f(const Vector &wo, const Vector &wi)const;
  OrenNayar(const Spectrum &reflectance, float sig)
  : BxDF(BxDFType(BSDF_REFLECTION | BSDF_DIFFUSE)),
    R(reflectance) {
    float sigma = Radians(sig);
    float sigma2 = sigma*sigma;
    A = 1.f - (sigma2 / (2.f * (sigma2 + 0.33f)));
    B = 0.45f * sigma2 / (sigma2 + 0.09f);
  }
private:
  Spectrum R;
  float A, B;
};
```

Oren-Nayar model

standard deviation for Gaussian

$$A = 1 - \frac{\sigma^2}{2(\sigma^2 + 0.33)}, \quad B = \frac{0.45\sigma^2}{\sigma^2 + 0.09}$$

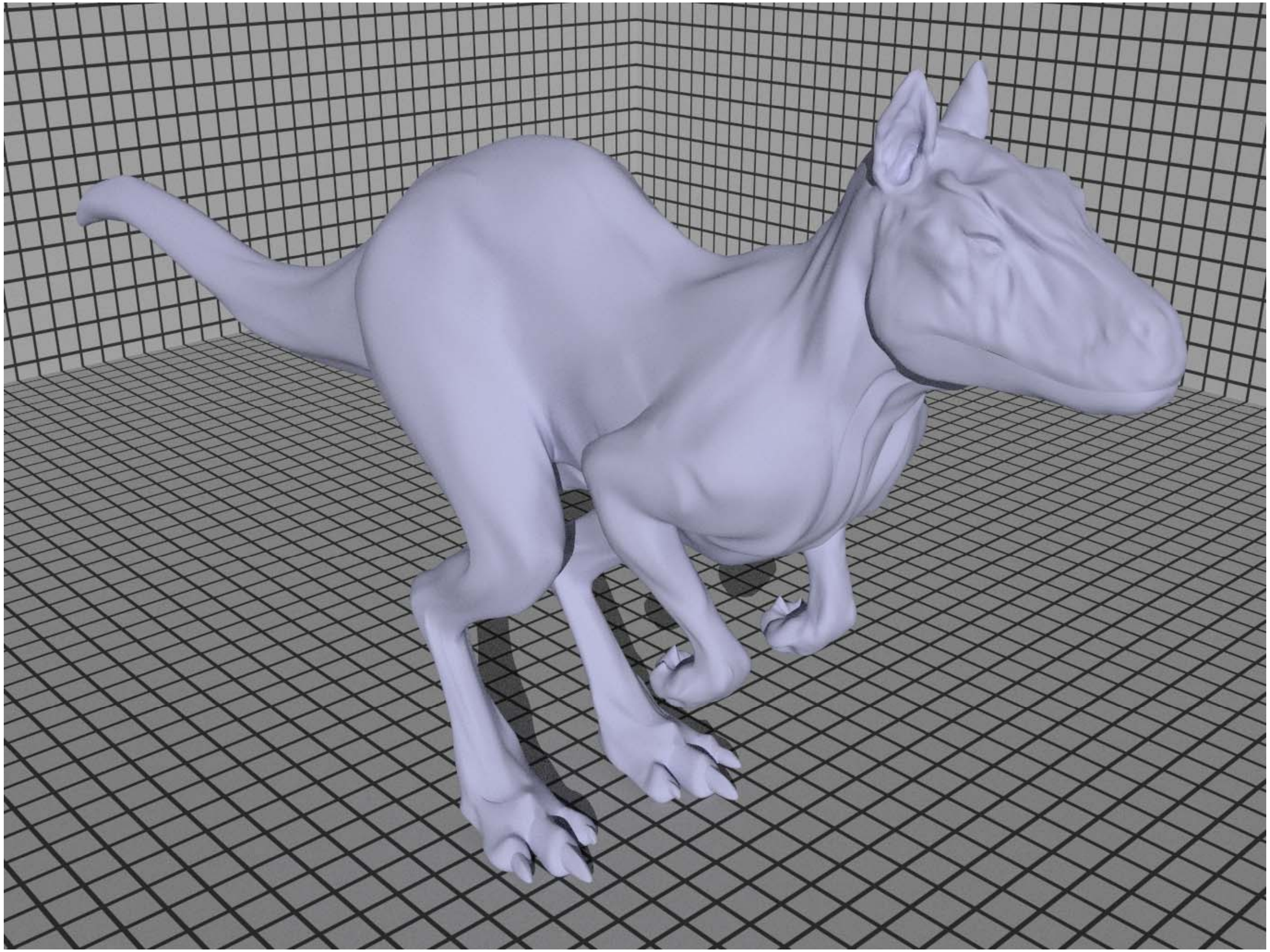
$$\alpha = \max(\theta_i, \theta_o), \quad \beta = \min(\theta_i, \theta_o)$$

$$f_r(\omega_i, \omega_o) = \frac{\rho}{\pi}(A + B\max(0, \cos(\phi_i - \phi_o))\sin\alpha \tan\beta)$$

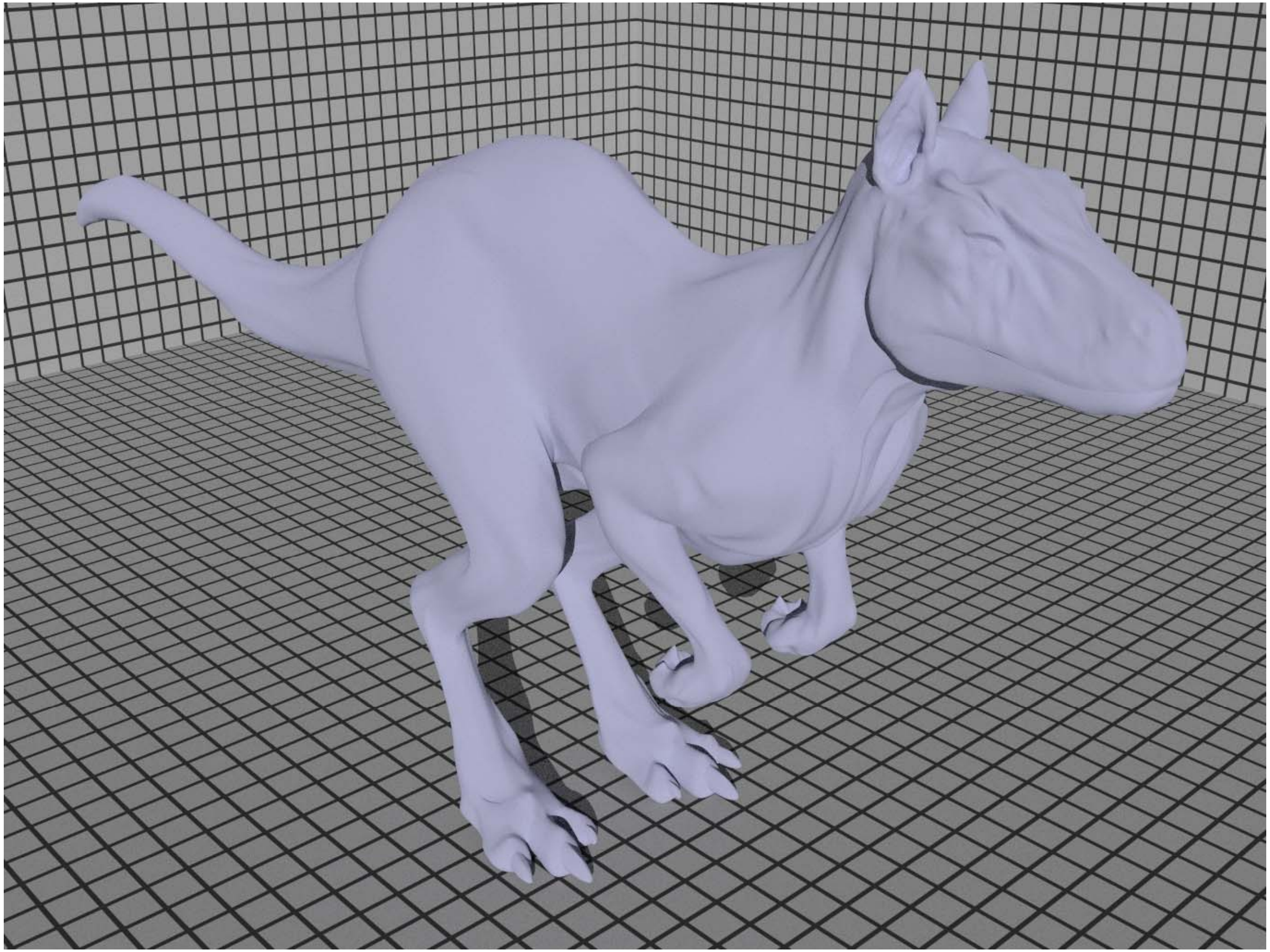
Oren-Nayar model

```
Spectrum OrenNayar::f(Vector &wo, Vector &wi)vconst{
  float sinthetai = SinTheta(wi);
  float sinthetao = SinTheta(wo);
  float sinphii = SinPhi(wi), cosphii = CosPhi(wi);
  float sinphio = SinPhi(wo), cosphio = CosPhi(wo);
  float dcos = cosphii * cosphio + sinphii * sinphio;
  float maxcos = max(0.f, dcos);
  float sinalpha, tanbeta;
  if (fabsf(CosTheta(wi)) > fabsf(CosTheta(wo))) {
    sinalpha = sinthetao;
    tanbeta = sinthetai / fabsf(CosTheta(wi));
  } else {
    sinalpha = sinthetai;
    tanbeta = sinthetao / fabsf(CosTheta(wo));
  }
  return R * INV_PI *
         (A + B * maxcos * sinalpha * tanbeta);
}
```

Lambertian

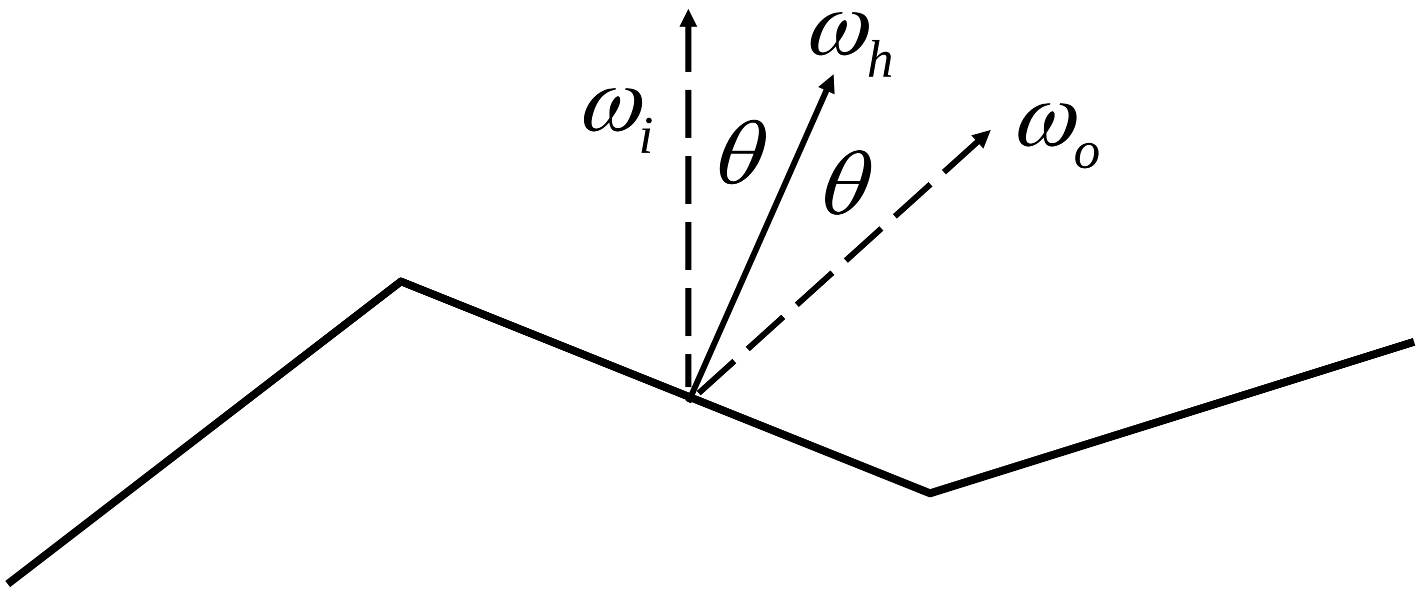


Oren-Nayer model



Torrance-Sparrow model

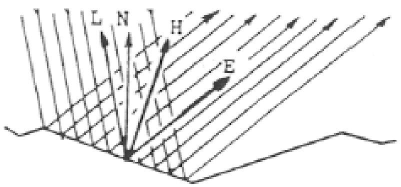
- One of the first microfacet models, designed to model metallic surfaces
- A collection of perfectly smooth mirrored microfacets with distribution $D(\omega_h)$



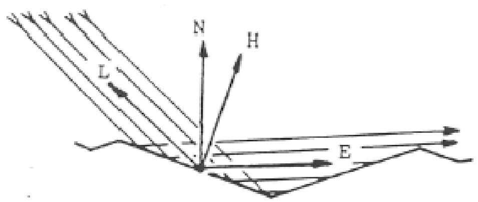
Torrance-Sparrow model

- Microfacet distribution D
- Fresnel reflection F
- Geometric attenuation G

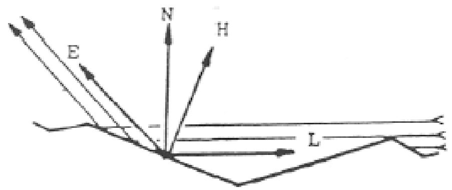
$$f_r(\omega_i \omega_o) = \frac{D(\omega_h)\, G(\omega_i, \omega_o)\, F(\omega_i, \omega_h)}{4 \cos\theta_i \cos\theta_o}$$



$$G = 1$$

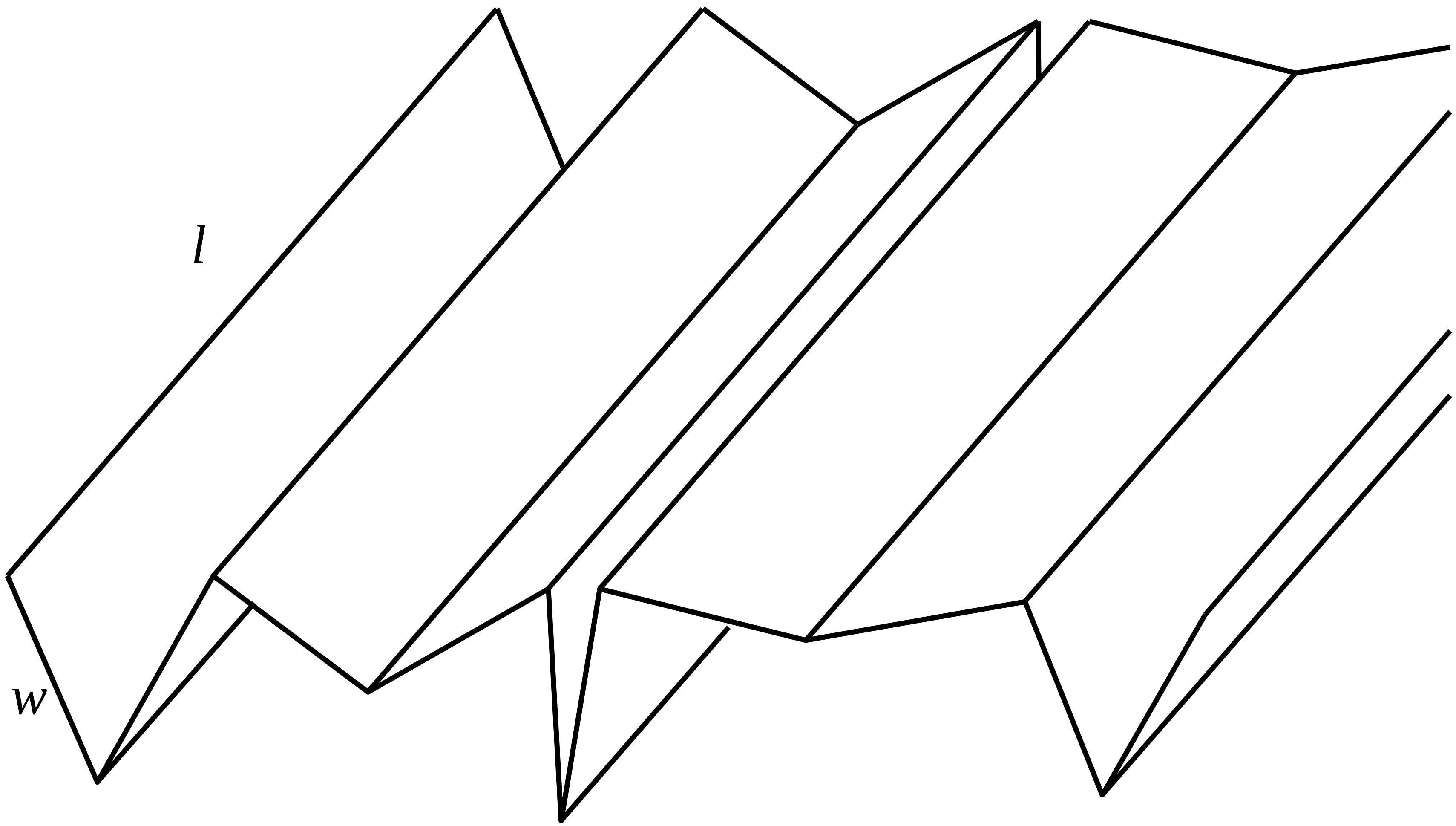


$$G = \frac{2(N \cdot H)(N \cdot \omega_i)}{(H \cdot \omega_i)}$$



$$G = \frac{2(N \cdot H)(N \cdot \omega_o)}{(H \cdot \omega_o)}$$

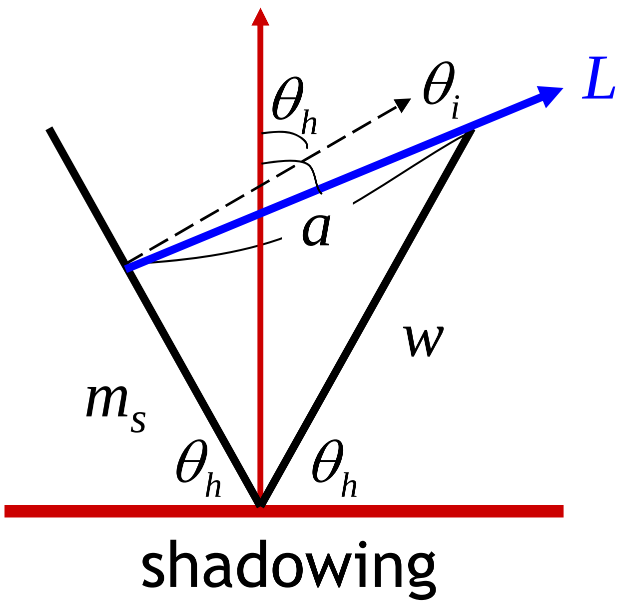
Configuration



Geometry attenuation factor

$$G = \frac{\text{facet area that is both visible and illuminated}}{\text{total facet area}}$$

$$= \frac{1 \cdot \min(w - m_s, w - m_v)}{1 \cdot w} = \min\left(1 - \frac{m_s}{w}, 1 - \frac{m_v}{w}\right)$$



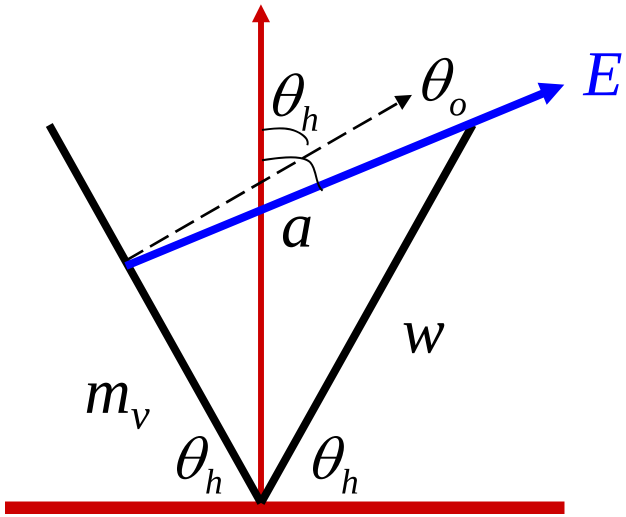
$$a \sin\theta_i = w\cos\theta_h + m_s \cos\theta_h \quad \times\cos\theta_i$$

$$a \cos\theta_i = w\sin\theta_h - m_s \sin\theta_h \quad \times -\sin\theta_i$$

$$\frac{m_s}{w} = -\frac{\cos(\theta_h + \theta_i)}{\cos(\theta_h - \theta_i)}$$

$$1 - \frac{m_s}{w} = \frac{2\cos\theta_h \cos\theta_i}{\cos(\theta_h - \theta_i)}$$

Geometry attenuation factor



$$1 - \frac{m_v}{w} = \frac{2\cos\theta_h \cos\theta_o}{\cos(\theta_h - \theta_o)}$$

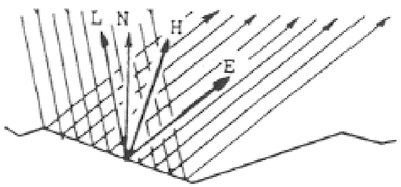
$$G = \min\left(1 - \frac{m_s}{w}, 1 - \frac{m_v}{w}\right) = \min\left(\frac{2\cos\theta_h \cos\theta_i}{\cos(\theta_h - \theta_i)}, \frac{2\cos\theta_h \cos\theta_o}{\cos(\theta_h - \theta_o)}\right)$$

$$G(\omega_o, \omega_i) = \min\left(1, \min\left(\frac{2(n \cdot \omega_h)(n \cdot \omega_i)}{\omega_i \cdot \omega_h}, \frac{2(n \cdot \omega_h)(n \cdot \omega_o)}{\omega_o \cdot \omega_h}\right)\right)$$

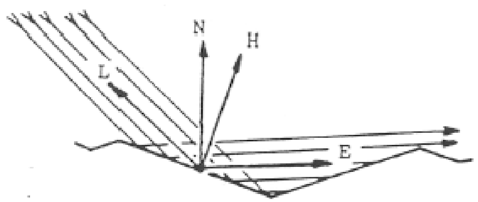
Torrance-Sparrow model

- Microfacet distribution D
- Fresnel reflection F
- Geometric attenuation G

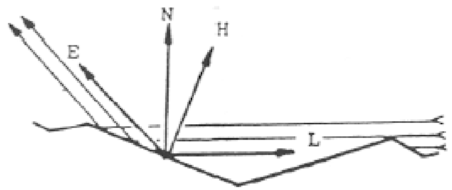
$$f_r(\omega_i \omega_o) = \frac{D(\omega_h)\, G(\omega_i, \omega_o)\, F(\omega_i, \omega_h)}{4 \cos\theta_i \cos\theta_o}$$



$G = 1$



$$G = \frac{2(N \cdot H)(N \cdot \omega_i)}{(H \cdot \omega_i)}$$



$$G = \frac{2(N \cdot H)(N \cdot \omega_o)}{(H \cdot \omega_o)}$$

Microfacet model

```
class MicrofacetDistribution {
public:
  virtual ~MicrofacetDistribution() { }
  virtual float D(const Vector &wh) const=0;
  virtual void Sample_f(const Vector &wo,
    Vector *wi, float u1, float u2,
    float *pdf) const = 0;
  virtual float Pdf(const Vector &wo,
                  const Vector &wi) const = 0;
};
```

Microfacet model

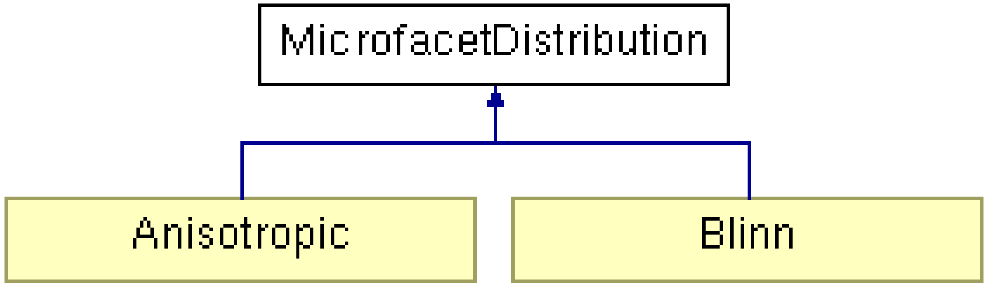
```
class Microfacet : public BxDF {
public:
  Microfacet(const Spectrum &reflectance, Fresnel *f,
             MicrofacetDistribution *d);
  Spectrum f(const Vector &wo,const Vector &wi) const;
  float G(Vector &wo, Vector &wi, Vector &wh) const {
    float NdotWh = fabsf(CosTheta(wh));
    float NdotWo = fabsf(CosTheta(wo));
    float NdotWi = fabsf(CosTheta(wi));
    float WOdotWh = AbsDot(wo, wh);
    return min(1.f, min((2.f*NdotWh*NdotWo/WOdotWh),
                        (2.f*NdotWh*NdotWi/WOdotWh)));
  }
  ...
private:
  Spectrum R;   Fresnel *fresnel;
  MicrofacetDistribution *distribution;
};
```

Microfacet model

```
Spectrum Microfacet::f(Vector &wo, Vector &wi)
{
  float cosThetaO = fabsf(CosTheta(wo));
  float cosThetaI = fabsf(CosTheta(wi));
  if (cosThetaI == 0.f || cosThetaO == 0.f)
    return Spectrum(0.f);
  Vector wh = wi + wo;
  if (wh.x == 0. && wh.y == 0. && wh.z == 0.)
    return Spectrum(0.f);
  wh = Normalize(wh);
  float cosThetaH = Dot(wi, wh);
  Spectrum F = fresnel->Evaluate(cosThetaH);
  return R * distribution->D(wh) * G(wo, wi, wh) * F
         / (4.f * cosThetaI * cosThetaO);
}
```

Microfacet models

- Blinn
- Anisotropic



Blinn microfacet distribution

- Distribution of microfacet normals is modeled by an exponential falloff

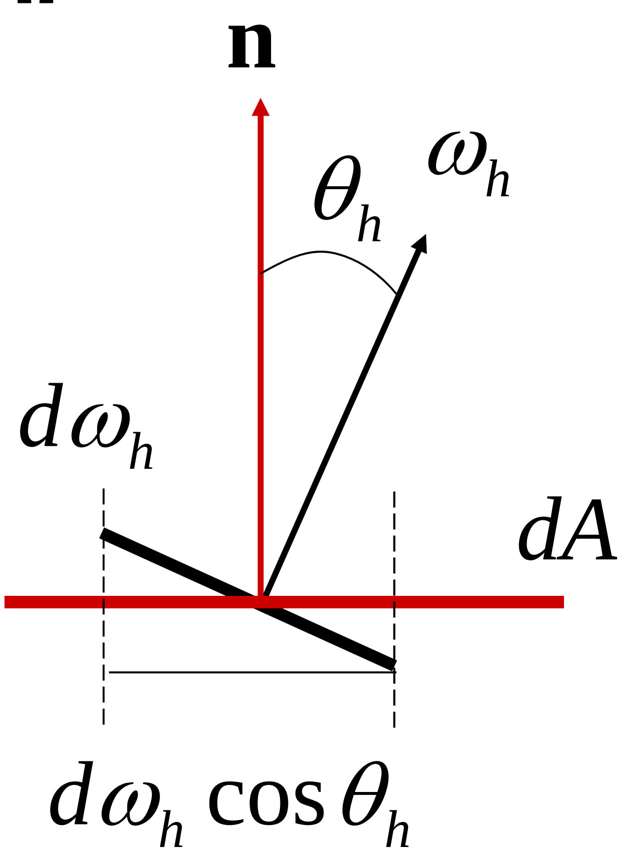
$$D(\omega_h) \propto (\omega_h \cdot n)^e = (\cos\theta_h)^e$$

- For smooth surfaces, this falloff happens very quickly; for rough surfaces, it is more gradual.
- Microfacet distribution must be normalized to ensure that they are physically plausible. The projected area of all microfacet faces over some area dA, the sum should be dA.

$$\int_{\Omega} D(\omega_h)\cos\theta_h d\omega_h = 1$$

Blinn microfacet distribution

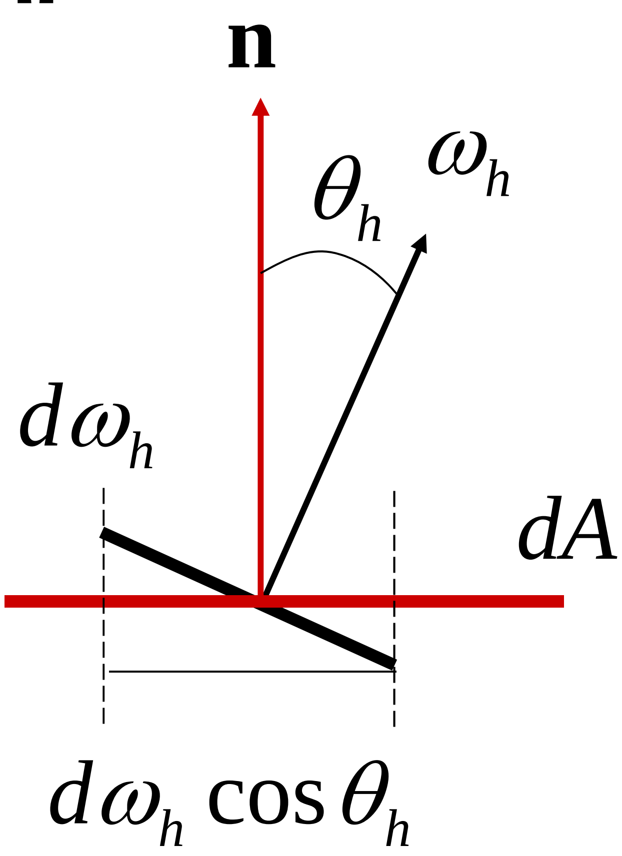
$$\int_{\Omega} D(\omega_h)\cos\theta_h d\omega_h = 1 \qquad \int_{\Omega} c(\omega_h \cdot \mathbf{n})^e \cos\theta_h d\omega_h = 1$$



Blinn microfacet distribution

$$\int_\Omega D(\omega_h)\cos\theta_h d\omega_h = 1$$

$$\int_\Omega c(\omega_h \cdot \mathbf{n})^e \cos\theta_h d\omega_h = 1$$



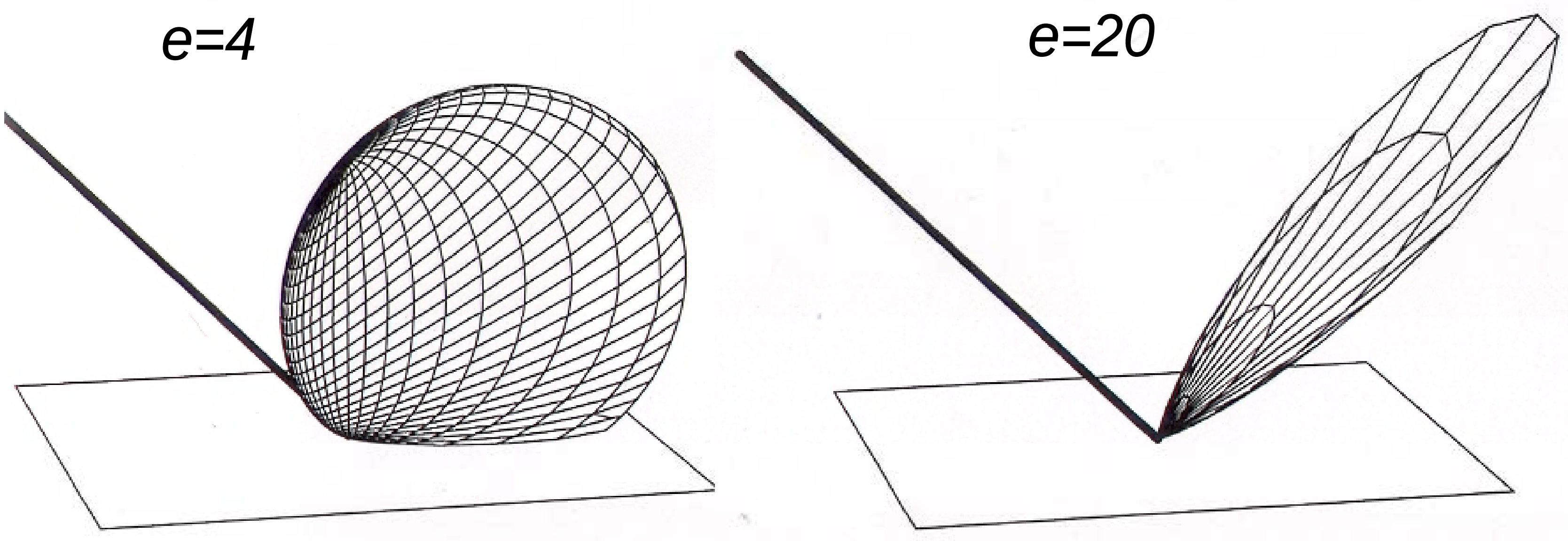
$$\int_0^{2\pi}\int_0^{\frac{\pi}{2}} c(\cos\theta_h)^{e+1}\sin\theta_h d\theta_h d\phi_h = 1$$

$$2\pi c\int_0^{\frac{\pi}{2}}(\cos\theta_h)^{e+1}(-d\cos\theta_h) = 1$$

$$-2\pi c\left.\frac{(\cos\theta_h)^{e+2}}{e+2}\right|_{\cos\theta_h=1}^{\cos\theta_h=0} = 1$$

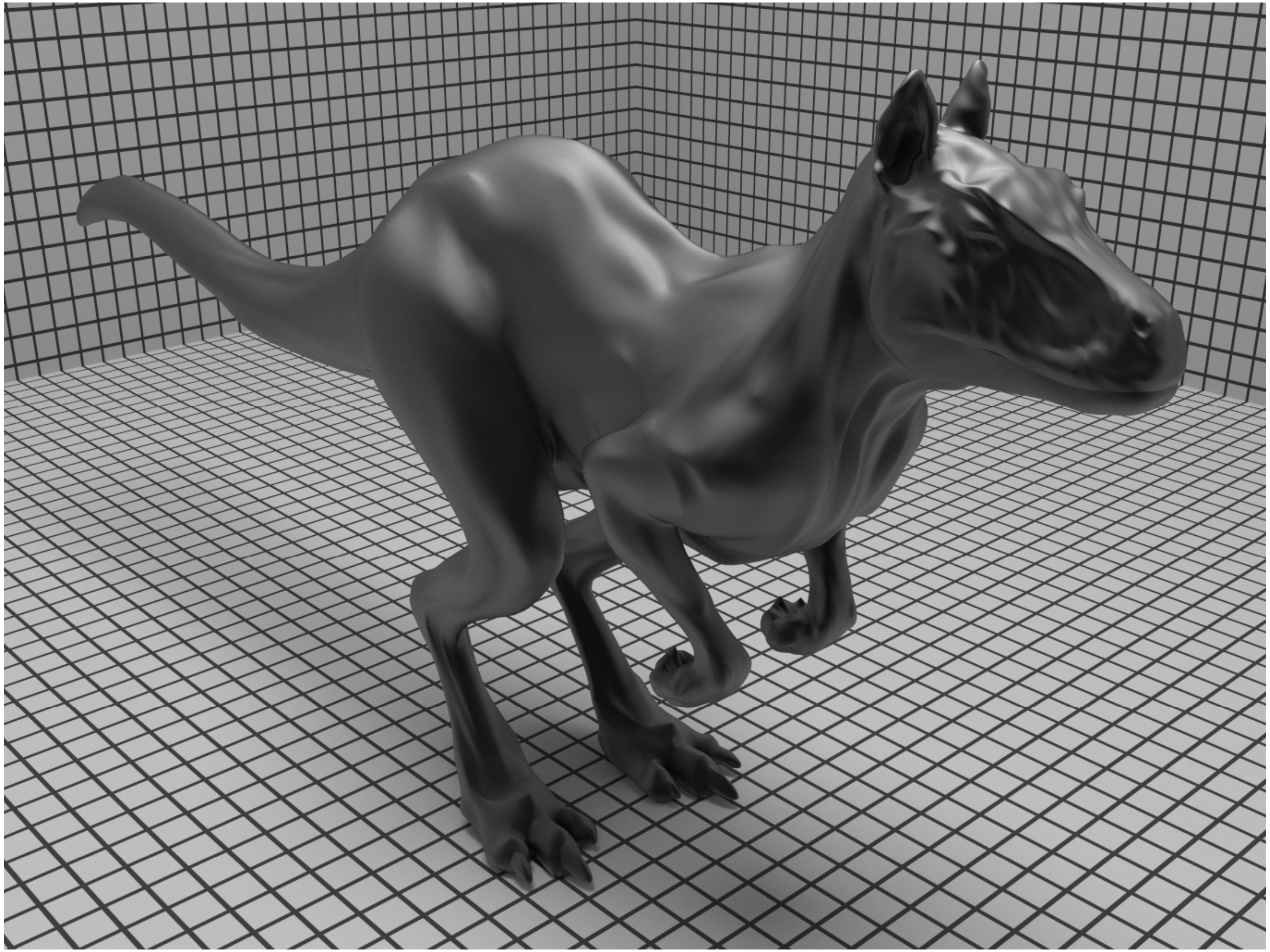
$$c = \frac{e+2}{2\pi} \qquad D(\omega_h) = \frac{e+2}{2\pi}(\omega_h \cdot n)^e$$

Blinn microfacet distribution



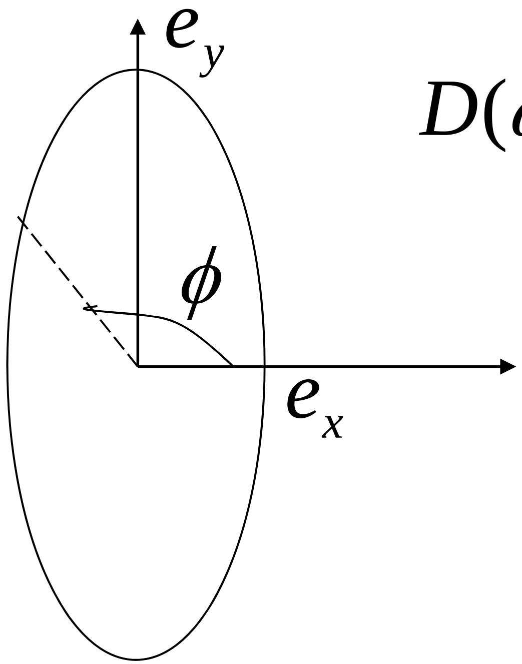

```
class Blinn : public MicrofacetDistribution
{
  ...
  float Blinn::D(const Vector &wh) const {
    float costhetah = fabsf(CosTheta(wh));
    return (exponent+2) * INV_TWOPI *
      powf(max(0.f, costhetah), exponent);
  }
}
```

Torrance-Sparrow with Blinn distribution



Anisotropic microfacet model

- Blinn microfacet model is radially symmetric (only depending on θ_h); hence, it is isotropic.
- Ashikmin and Shirley have developed a microfacet model for anisotropic surfaces



$$D(\omega_h) \propto (\omega_h \cdot n)^{e_x \cos^2 \phi_h + e_y \sin^2 \phi_h}$$

Ashikmin-Shirley model

$$\int_{\Omega} c(\omega_h \cdot \mathbf{n})^{e_x \cos^2 \phi_h + e_y \sin^2 \phi_h} \cos \theta_h \, d\omega_h = 1$$

Ashikmin-Shirley model

$$\int_{\Omega} c(\omega_h \cdot \mathbf{n})^{e_x \cos^2\phi_h + e_y \sin^2\phi_h} \cos\theta_h d\omega_h = 1$$

$$\int_0^{2\pi}\int_0^{\pi/2} c(\cos\theta_h)^{e_x \cos^2\phi_h + e_y \sin^2\phi_h + 1} \sin\theta_h d\theta_h d\phi_h = 1$$

$$c\int_0^{2\pi}\int_0^{\pi/2} (\cos\theta_h)^{e_x \cos^2\phi_h + e_y \sin^2\phi_h + 1} d\cos\theta_h d\phi_h = -1$$

$$c\int_0^{2\pi} \left. \frac{(\cos\theta_h)^{e_x \cos^2\phi_h + e_y \sin^2\phi_h + 2}}{e_x \cos^2\phi_h + e_y \sin^2\phi_h + 2} \right|_1^0 d\phi_h = -1$$

$$c\int_0^{2\pi} \frac{1}{e_x \cos^2\phi_h + e_y \sin^2\phi_h + 2} d\phi_h = 1$$

Ashikmin-Shirley model

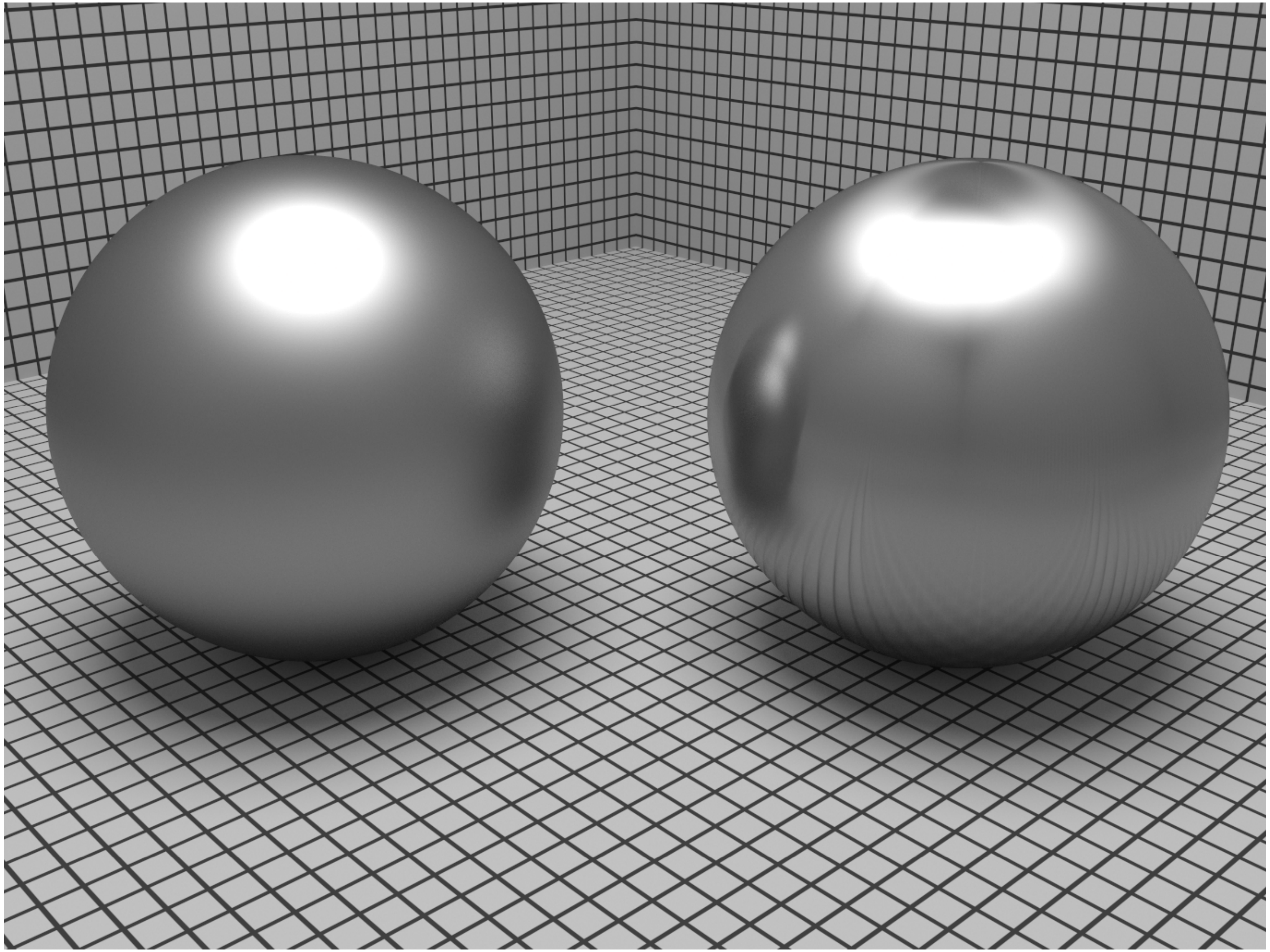
$$c\int_0^{2\pi}\frac{1}{e_x\cos^2\phi_h+e_y\sin^2\phi_h+2}\,d\phi_h=1$$

$$\int\frac{1}{a\cos^2(x)+b\sin^2(x)+2}\,dx=\frac{\tan^{-1}\left(\frac{\sqrt{b+2}\,\tan(x)}{\sqrt{a+2}}\right)}{\sqrt{a+2}\,\sqrt{b+2}}$$

$$c\frac{2\pi}{\sqrt{e_x+2}\sqrt{e_y+2}}=1$$

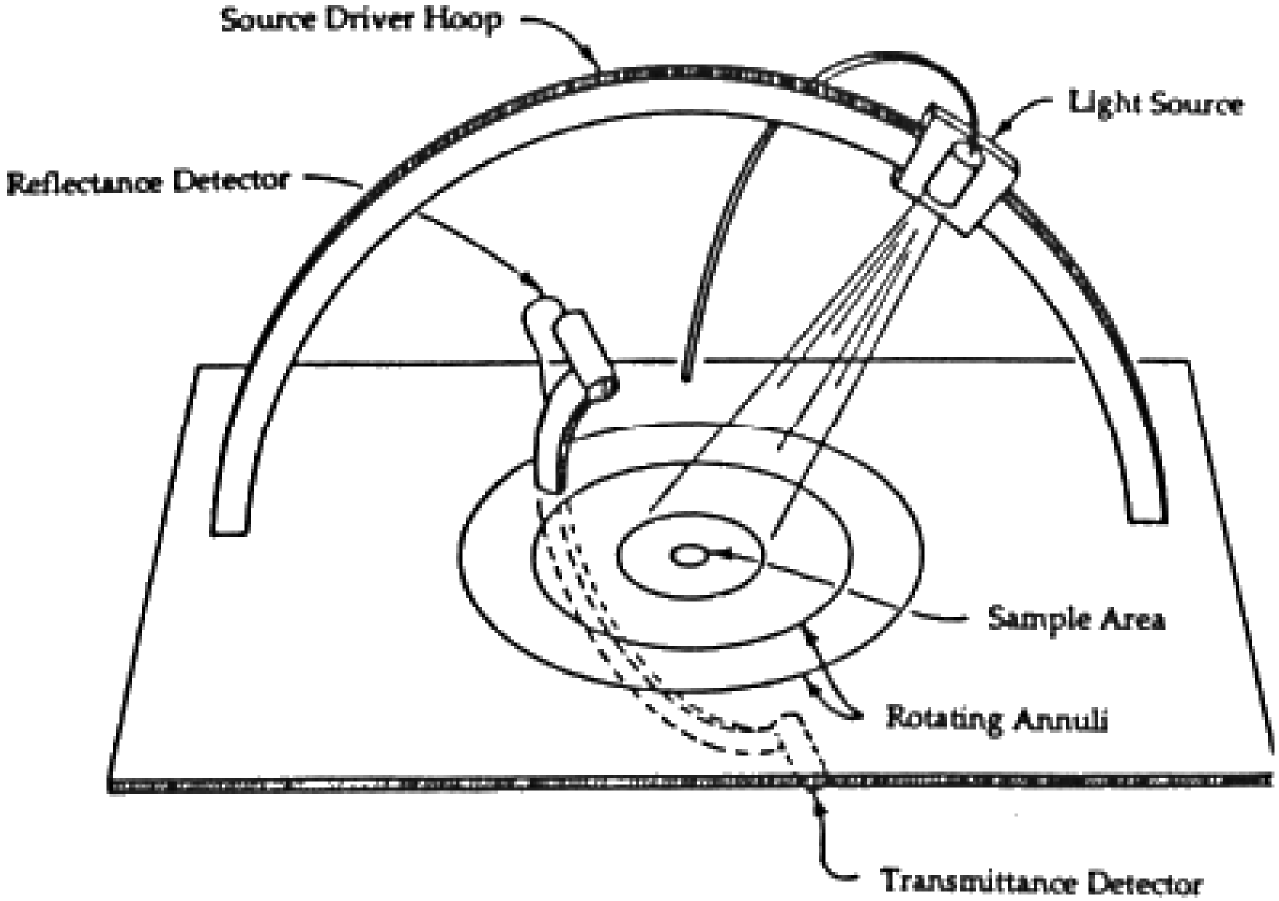
$$D(\omega_h)=\frac{\sqrt{(e_x+2)(e_y+2)}}{2\pi}(\omega_h\cdot n)^{e_x\cos^2\phi_h+e_y\sin^2\phi_h}$$

Anisotropic microfacet model



Measured BRDFs

- An effective approach for realistic materials is to use measured dat. The following device is proposed by Greg Ward in SIGGRAPH 1992



Measured BRDFs

- The measured data can be

1. Fitted into parametric models: compact and memory saving; not flexible enough to capture full complexity of scattering properties
2. Used directly: memory intensive, difficult to adjust, more subjective to measurement noise; high fidelity

- Measured data may come in one of two forms:

1. regularly spaced tabularized data (efficient to look up but could be more difficult to acquire)
2. a large number of irregularly spaced individual samples

- pbrt has support for both: **`RegularHalfangleBRDF`** and **`IrregIsotropicBRDF`**

Irregular isotropic measured BRDF

- Because of isotropy, we could use the mapping

$$m(\theta_i, \phi_i, \theta_o, \phi_o) \rightarrow (\theta_i, \theta_o, \phi_i - \phi_o)$$

- For the following properties, we instead use the mapping

$$m(\theta_i, \phi_i, \theta_o, \phi_o) \rightarrow (\sin\theta_i \sin\theta_o, \Delta\phi / \pi, \cos\theta_i \cos\theta_o)$$

1. Distance between points is meaningful
2. Isotropy is reflected
3. Reciprocity is represented

IrregIsotropicBRDF

```
class IrregIsotropicBRDF : public BxDF {
public:
 IrregIsotropicBRDF(KdTree<IrregIsotropicBRDFSample> *d)
   : BxDF(BxDFType(BSDF_REFLECTION | BSDF_GLOSSY)),
     isoBRDFData(d) { }
 Spectrum f(const Vector &wo, const Vector &wi) const;
private:
 const KdTree<IrregIsotropicBRDFSample> *isoBRDFData;
};
```

IrregIsotropicBRDF

```
Point BRDFRemap(const Vector &wo, const Vector &wi)
{
  float cosi = CosTheta(wi), coso = CosTheta(wo);
  float sini = SinTheta(wi), sino = SinTheta(wo);
  float phii = SphericalPhi(wi),
        phio = SphericalPhi(wo);
  float dphi = phii - phio;
  if (dphi < 0.) dphi += 2.f * M_PI;
  if (dphi > 2.f * M_PI) dphi -= 2.f * M_PI;
  if (dphi > M_PI) dphi = 2.f * M_PI - dphi;

  return Point(sini * sino, dphi / M_PI,
               cosi * coso);
}
```

IrregIsotropicBRDF

```
Spectrum IrregIsotropicBRDF::f(const Vector &wo,
                               const Vector &wi)
{
  Point m = BRDFRemap(wo, wi);
  float lastMaxDist2 = .001f;
  while (true) {
    IrregIsoProc proc;
    float maxDist2 = lastMaxDist2;
    isoBRDFData->Lookup(m, proc, maxDist2);
    if (proc.nFound > 2 || lastMaxDist2 > 1.5f)
      return proc.v.Clamp() / proc.sumWeights;
    lastMaxDist2 *= 2.f;
  }
}
```

callback structure used by kdtree for each sample within the search radius

IrregIsoProc

```
struct IrregIsoProc {
  IrregIsoProc() { sumWeights = 0.f; nFound = 0; }
  void operator()(const Point &p, const
        IrregIsotropicBRDFSample &sample,
        float d2, float &maxDist2)
  {
    float weight = expf(-100.f * d2);
    v += weight * sample.v;
    sumWeights += weight;
    ++nFound;
  }
  Spectrum v;
  float sumWeights;
  int nFound;
};
```

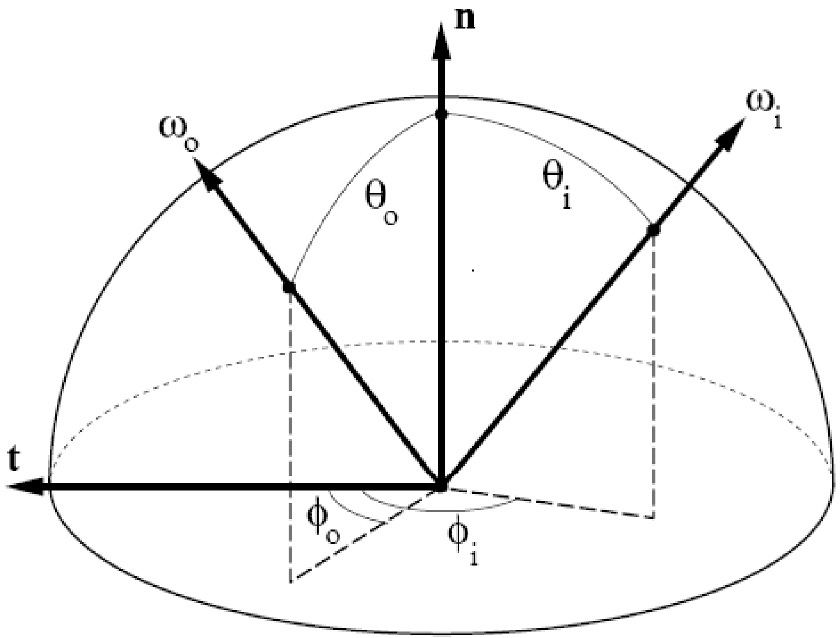
Regular halfangle format

- Supports isotropic measured BRDFs stored in a format used by Matusik et al. (2003).
- Rusunkiewicz proposed to use a mapping based on the half-angle vector and the difference vector, found by applying to ω_i the rotation which rotates the half vector to (0,0,1)

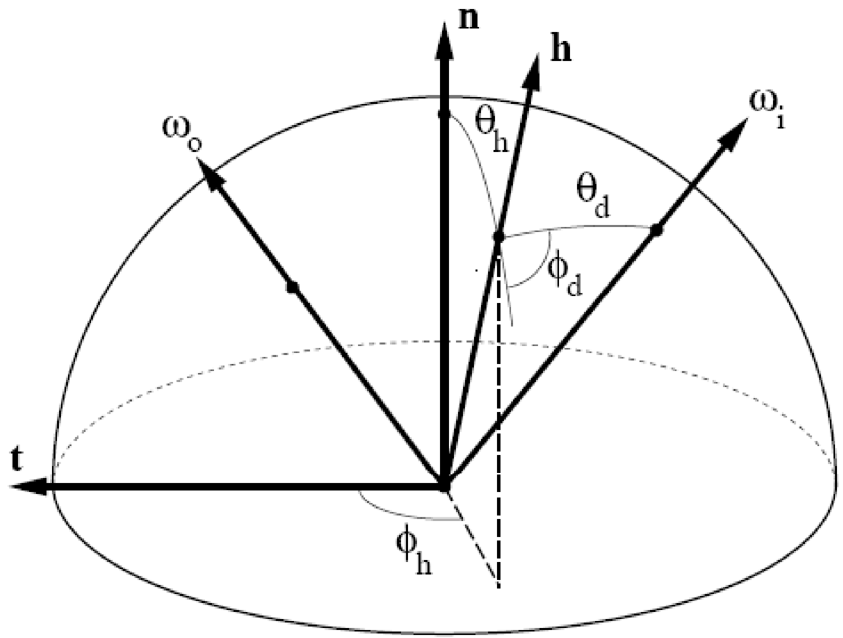
$$m(\theta_i, \phi_i, \theta_o, \phi_o) \rightarrow (\theta_h, \phi_h, \theta_d, \phi_d)$$

- Assuming isotropy, ϕ_h can be dropped and the table is indexed by $(\sqrt{\theta_h}, \theta_d, \phi_d)$. The square root is used to increase sampling for near-zero θ_h because small change there could lead to big function value change

Data representation

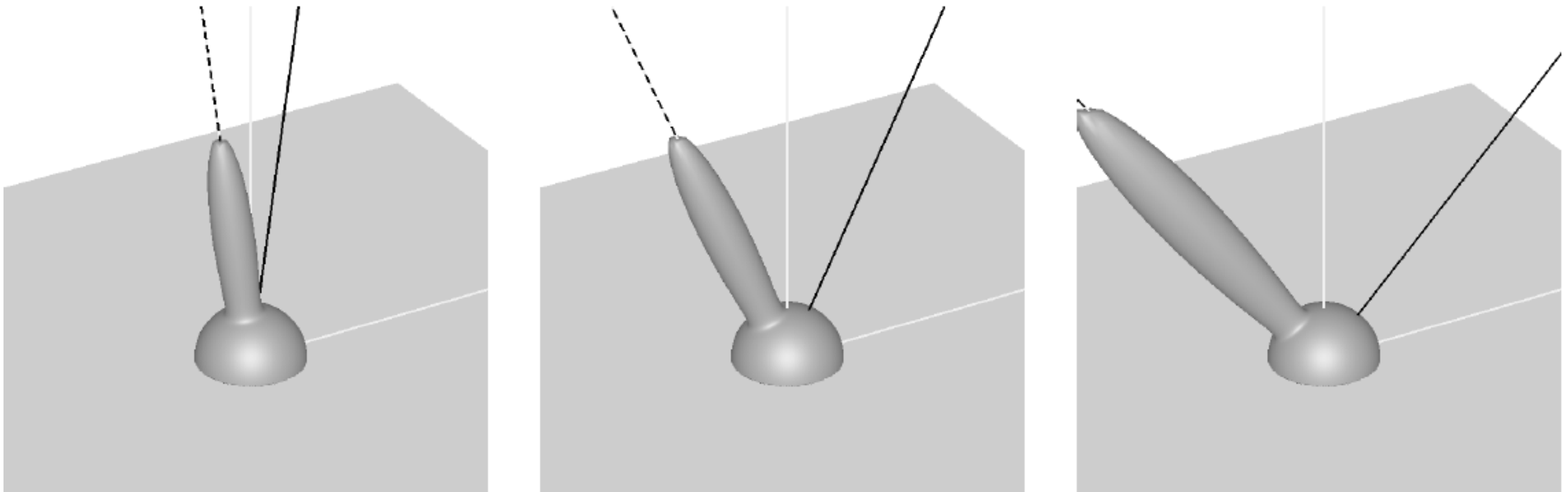


standard coordinate



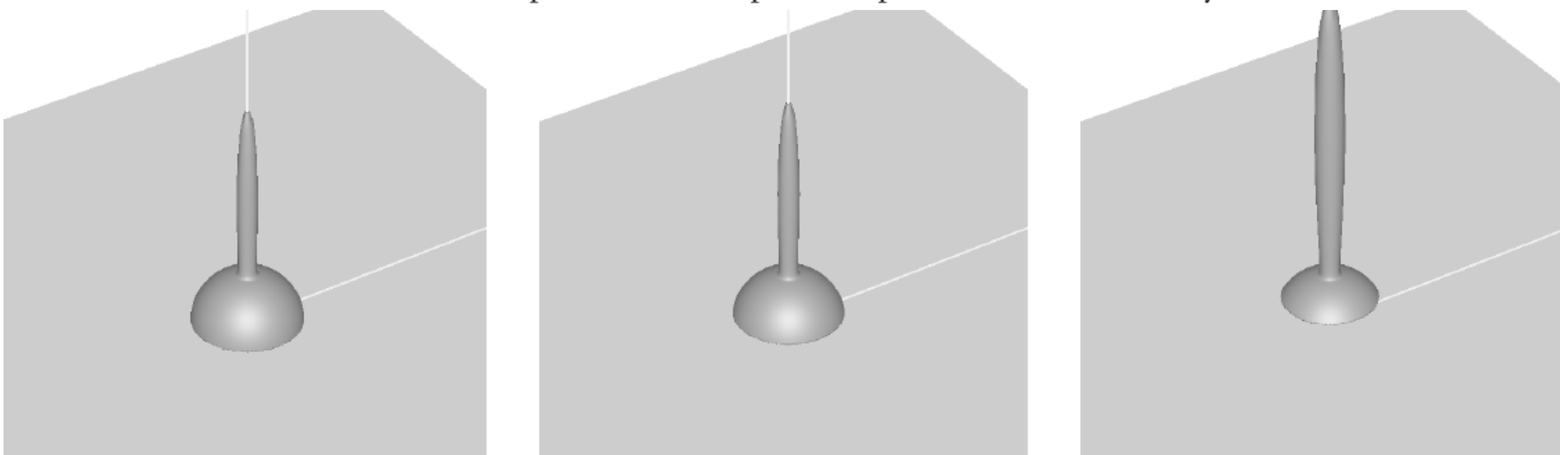
Rusinkiewicz coordinate

Data representation



$\theta_i = 10°$ $\theta_i = 20°$ $\theta_i = 40°$

The Cook-Torrance-Sparrow BRDF seen as a function of (θ_o, ϕ_o), for various values of (θ_i, ϕ_i). Note that the position of the peak in space varies considerably.



$\theta_d = 0°$ $\theta_d = 20°$ $\theta_d = 60°$

RegularHalfangleBRDF

```
class RegularHalfangleBRDF : public BxDF {
public:
  RegularHalfangleBRDF(const float *d,
   uint32_t nth, uint32_t ntd, uint32_t npd)
  :BxDF(BxDFType(BSDF_REFLECTION|BSDF_GLOSSY)),…)
  { }
  Spectrum f(const Vector &wo, const Vector &wi);
private:
  const float *brdf;
  const uint32_t nThetaH, nThetaD, nPhiD;
};
```

RegularHalfangleBRDF

```
Spectrum RegularHalfangleBRDF::f(Vector &wo,
                                 Vector &wi)
{ Vector wh = wi + wo;
  if (wh.x==0.f && wh.y==0.f && wh.z==0.f)
    return Spectrum (0.f);
  wh = Normalize(wh);
  float whTheta = SphericalTheta(wh);
  float whCosPhi = CosPhi(wh),
        whSinPhi = SinPhi(wh);
  float whCosTheta = CosTheta(wh),
        whSinTheta = SinTheta(wh);
  Vector whx(whCosPhi*whCosTheta, whSinPhi*whCosTheta,
             -whSinTheta);
  Vector why(-whSinPhi, whCosPhi, 0);
  Vector wd(Dot(wi, whx), Dot(wi, why), Dot(wi, wh));
```

Rotation matrix to align the halfangle vector to z-axis; whx, why and wh are rows.

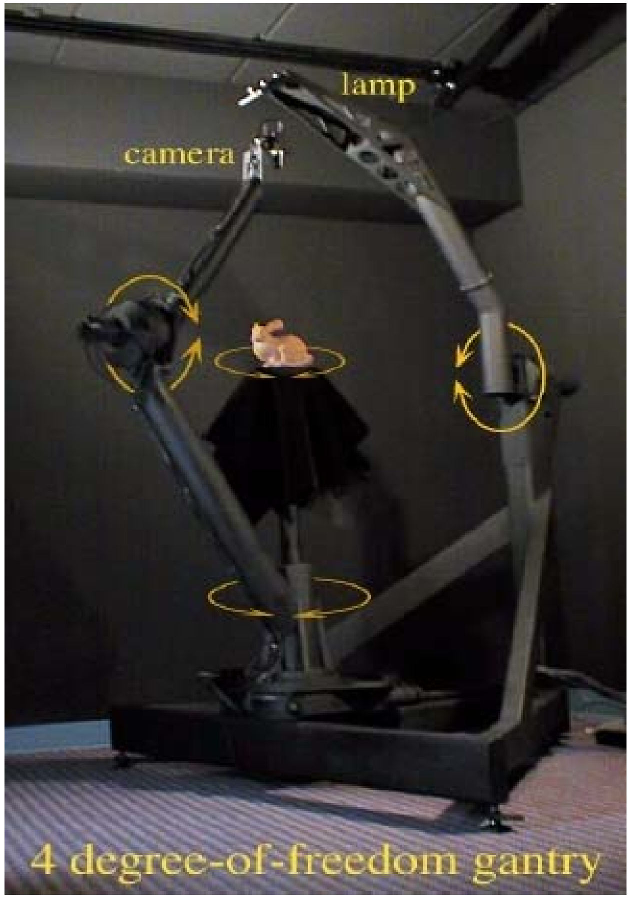
RegularHalfangleBRDF

```
  float wdTheta = SphericalTheta(wd),
        wdPhi = SphericalPhi(wd);
  if (wdPhi > M_PI) wdPhi -= M_PI;

  #define REMAP(V, MAX, COUNT) \
     Clamp(int((V) / (MAX) * (COUNT)), 0, (COUNT)-1)
  int whThetaIndex = REMAP(sqrtf(max(0.f, whTheta /
  (M_PI / 2.f))), 1.f, nThetaH);
  int wdThetaIndex = REMAP(wdTheta,M_PI/2.f,nThetaD);
  int wdPhiIndex = REMAP(wdPhi, M_PI, nPhiD);
#undef REMAP
  int index = wdPhiIndex + nPhiD *
              (wdThetaIndex + whThetaIndex * nThetaD);
  return Spectrum::FromRGB(&brdf[3*index]);
}
```

Lafortune model

An efficient model to fit measured data to a parameterized model with a relatively small number of parameters



modified Phong model

$$f_r(p,\omega_o,\omega_i) = \left(\omega_o \cdot R(\omega_i,\mathbf{n})\right)^e$$

$$= \left(\omega_o \cdot (-\omega_{ix}, -\omega_{iy}, \omega_{iz})\right)^e$$

orientation vector $(o_{i,x}, o_{i,y}, o_{i,z})$

$(-1,-1,+1)$ specular $(1,1,1)$ retro-reflective

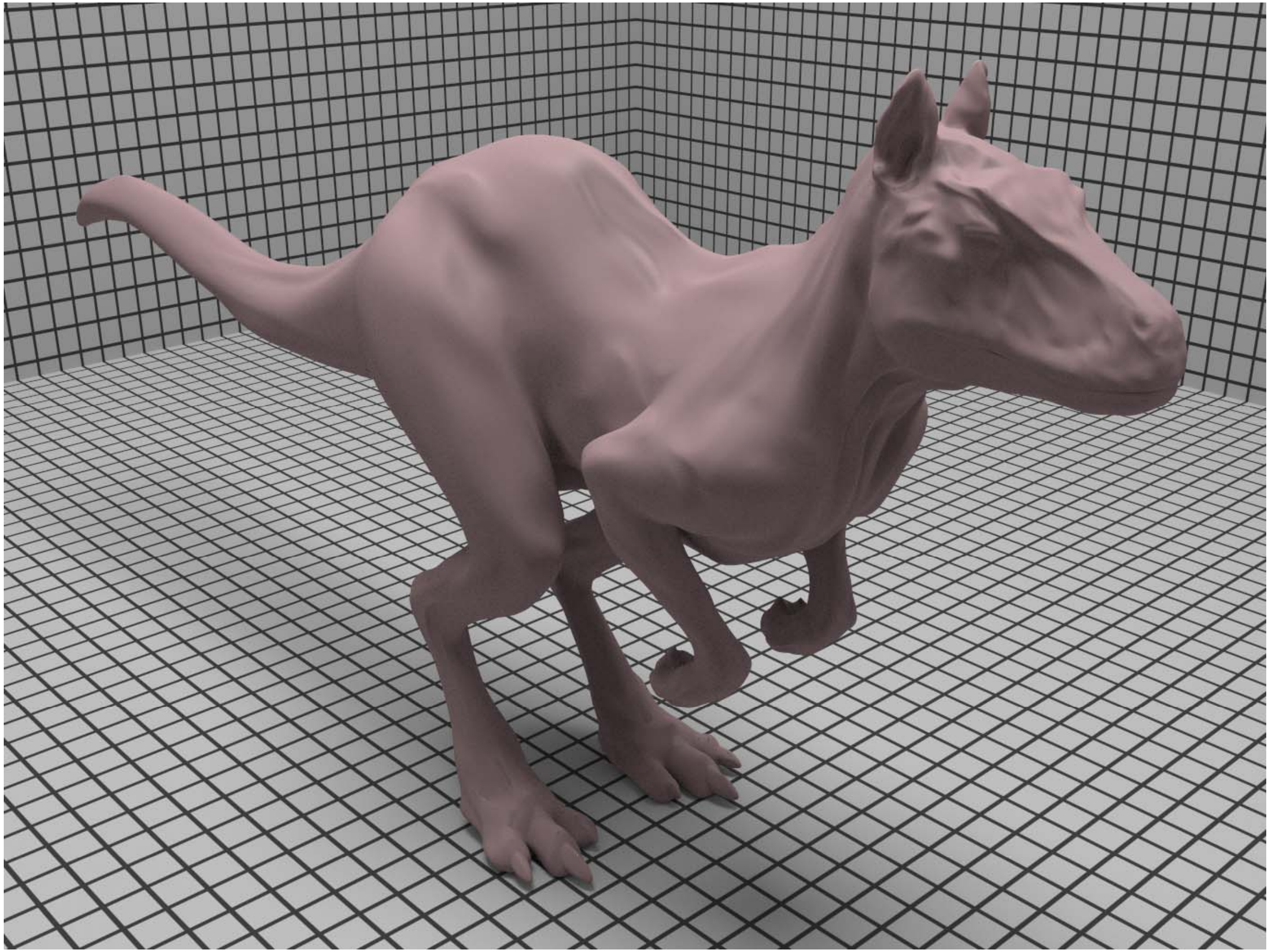
$(-1,-1,+0.5)$ off-specular

Lafortune model

$$f_r(p,\omega_o,\omega_i)$$

$$= \frac{\rho_d}{\pi} + \sum_{i=1}^{n} \left(\omega_o \cdot (\omega_{ix} o_{i,x}, \omega_{iy} o_{i,y}, \omega_{iz} o_{i,z})\right)^{e_i}$$

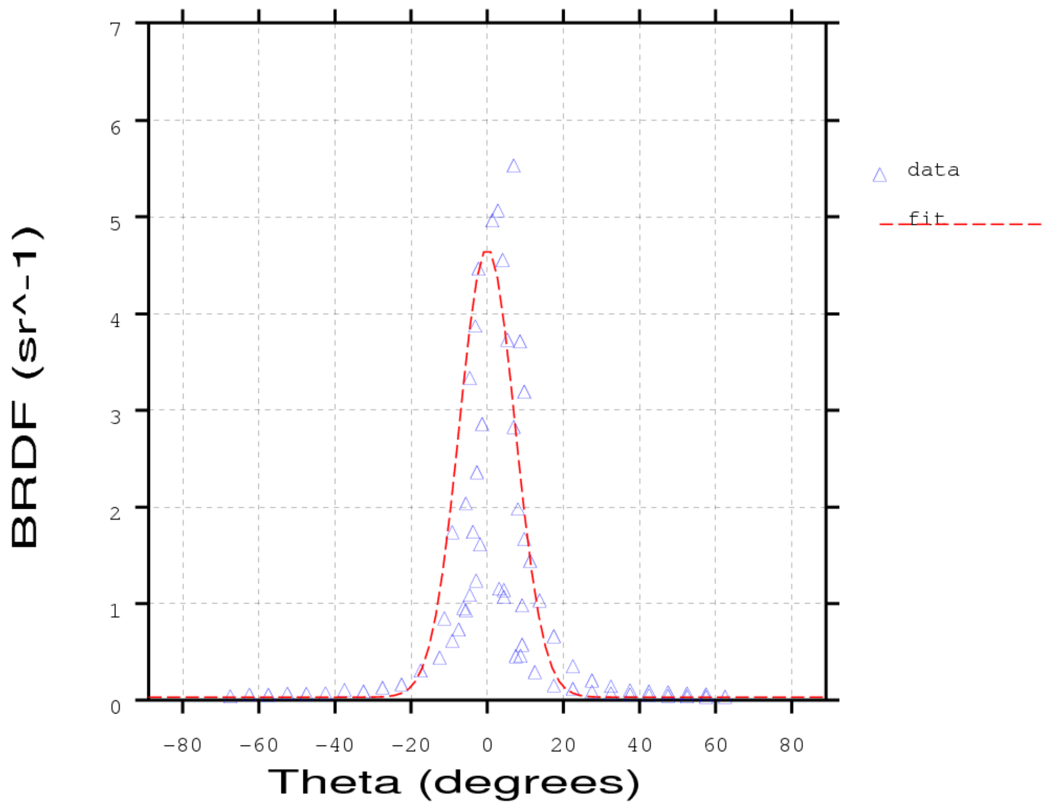
Lafortune model (for a measured clay)



Ward model

$$f(\omega_i,\omega_o)=\frac{\rho_d}{\pi}+\rho_s\frac{1}{4\pi\sigma^2\sqrt{\cos\theta_i\cos\theta_o}}\exp\left[-\frac{\tan^2\theta_h}{\sigma^2}\right]$$

$$f(\omega_i,\omega_o)=\frac{\rho_d}{\pi}+\rho_s\frac{1}{4\pi\sigma_x\sigma_y\sqrt{\cos\theta_i\cos\theta_o}}\exp\left[-\tan^2\theta_h\left(\frac{\cos^2\phi_h}{\sigma_x^2}+\frac{\sin^2\phi_h}{\sigma_y^2}\right)\right]$$



Ward model

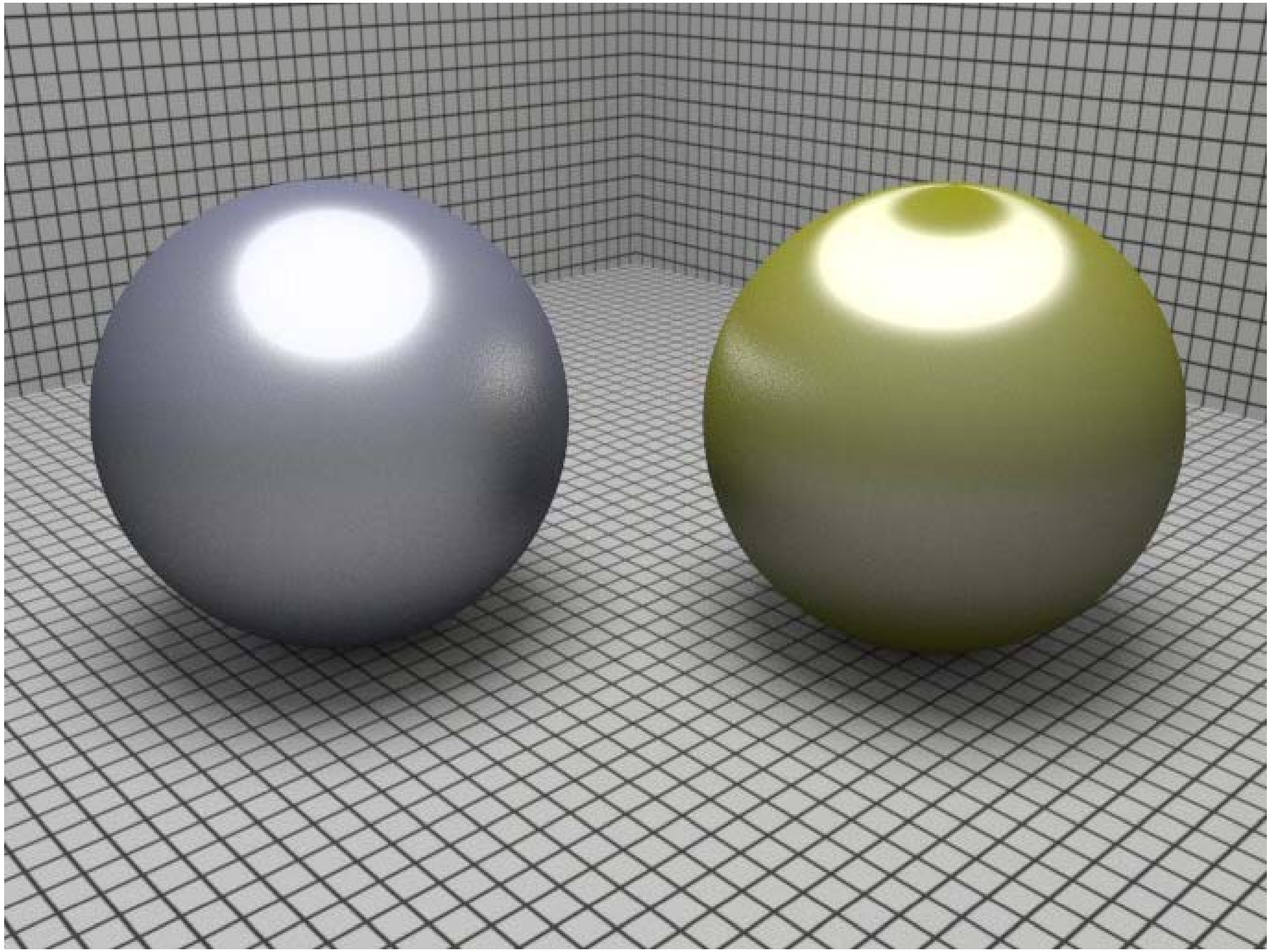
photograph

isotropic

anisotropic

Ward model

Ward model

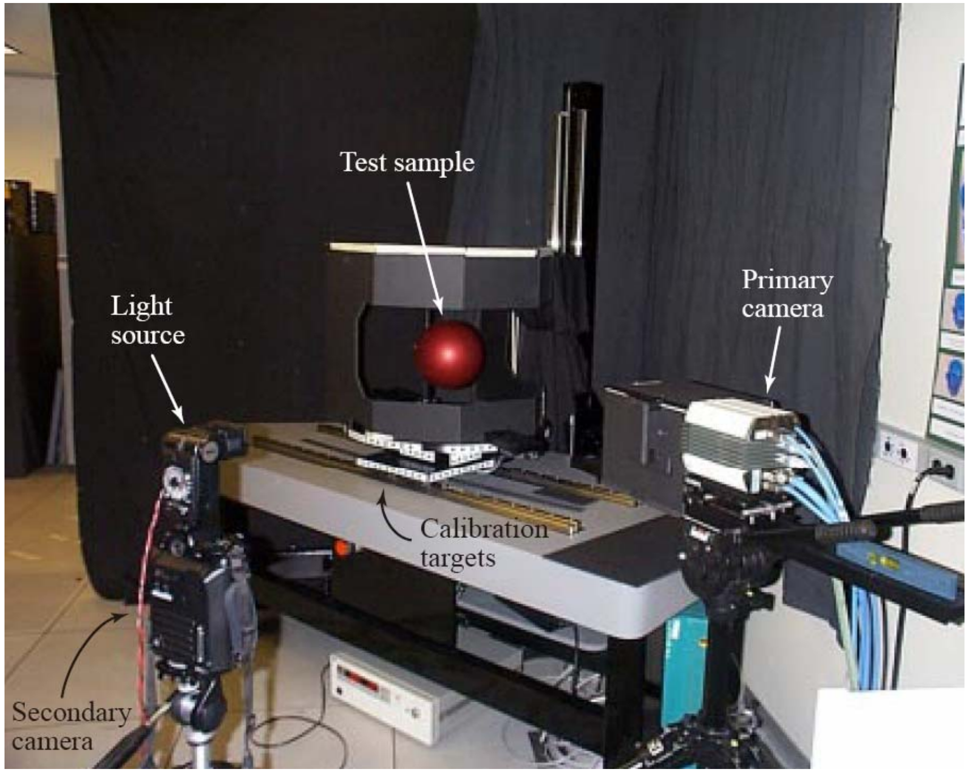


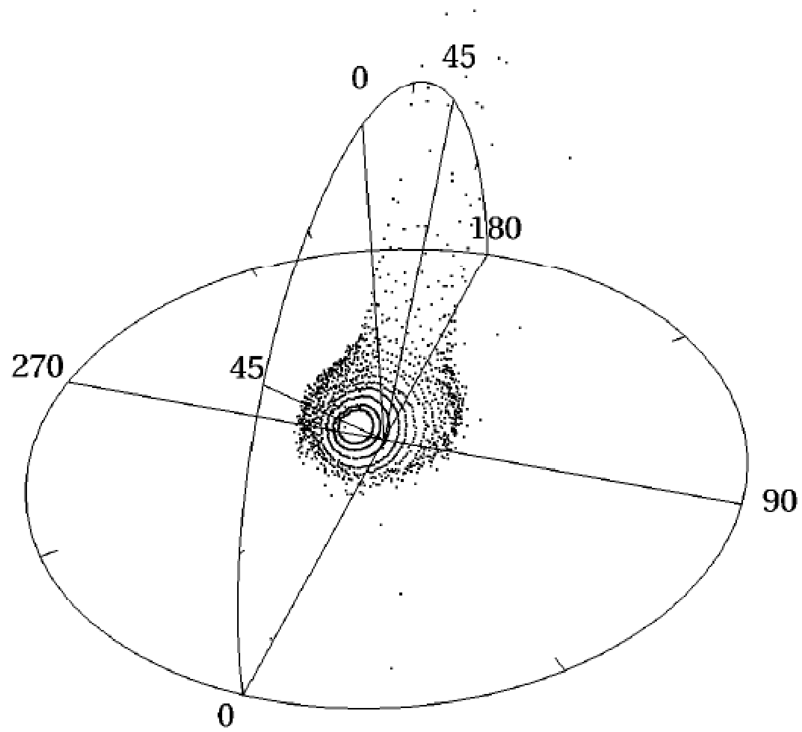
A data-driven reflectance model

- Analytic models
- measure-then-fit
  - approximation: reduce noise but also characteristic of the model
  - non-obvious error metric: often biased to specular
  - difficult optimization: nonlinear; depends on initial guess
- Tabulated BRDF
  - time-consuming
  - not editable
- Data-Driven Reflectance Model by Matusik et. al. in SIGGRAPH 2003

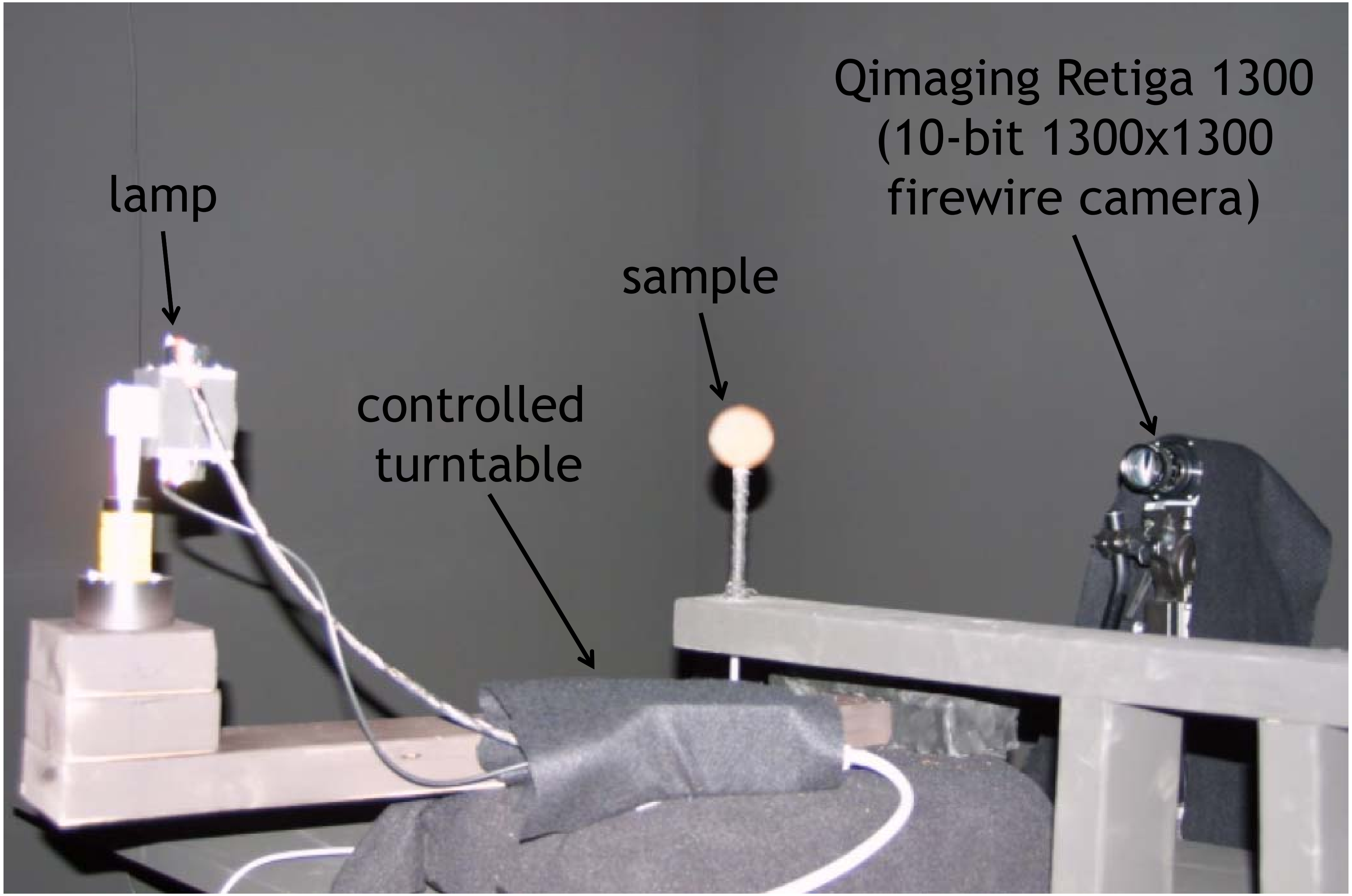
Acquisition

- Requirements: dense samples and wide range of BRDF models
- Inspired by Marschner; requires a spherically homogeneous sample of the material





Acquisition

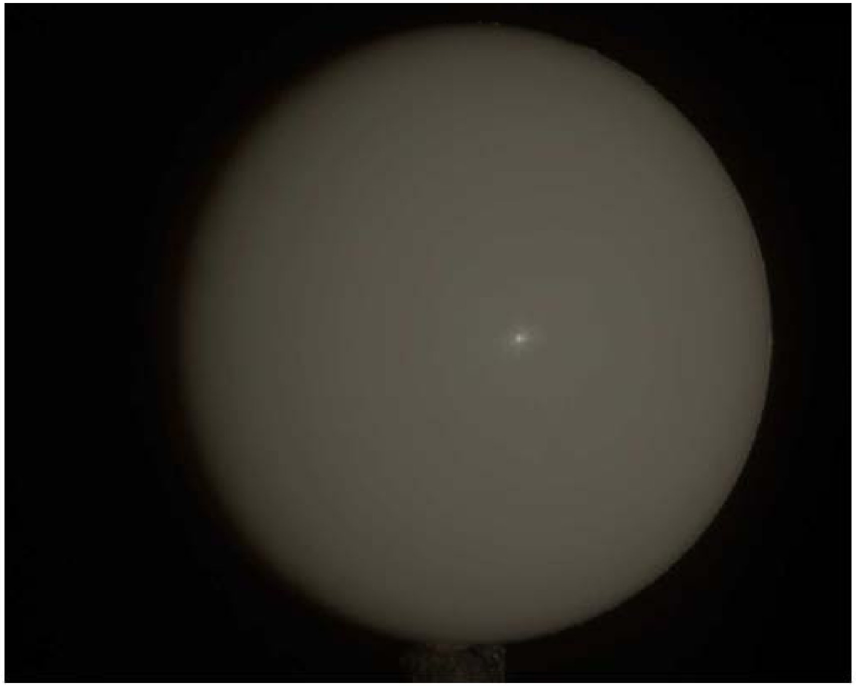


Acquisition

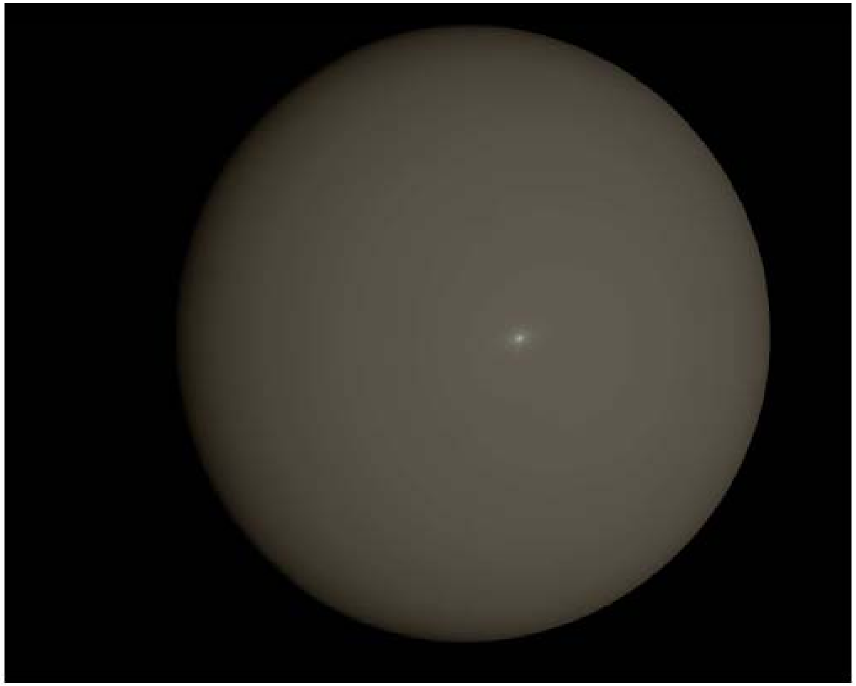
- Fixed calibrated camera; the light moves roughly every 0.5 degree
- It took 3 hours to take a total of 330 HDR images for a sample. (18 10-bit pictures for each HDR; linearly fitted)
- Each pixel gives one BRDF sample

Acquisition

90x90x180=1,458,000 bins (isotropic, reciprocity to reduce 360 to 180)
20~80M samples in total
For each bin; remove top and bottom 25% and then find the average
Reduce systematic error and tolerate spatial material variation

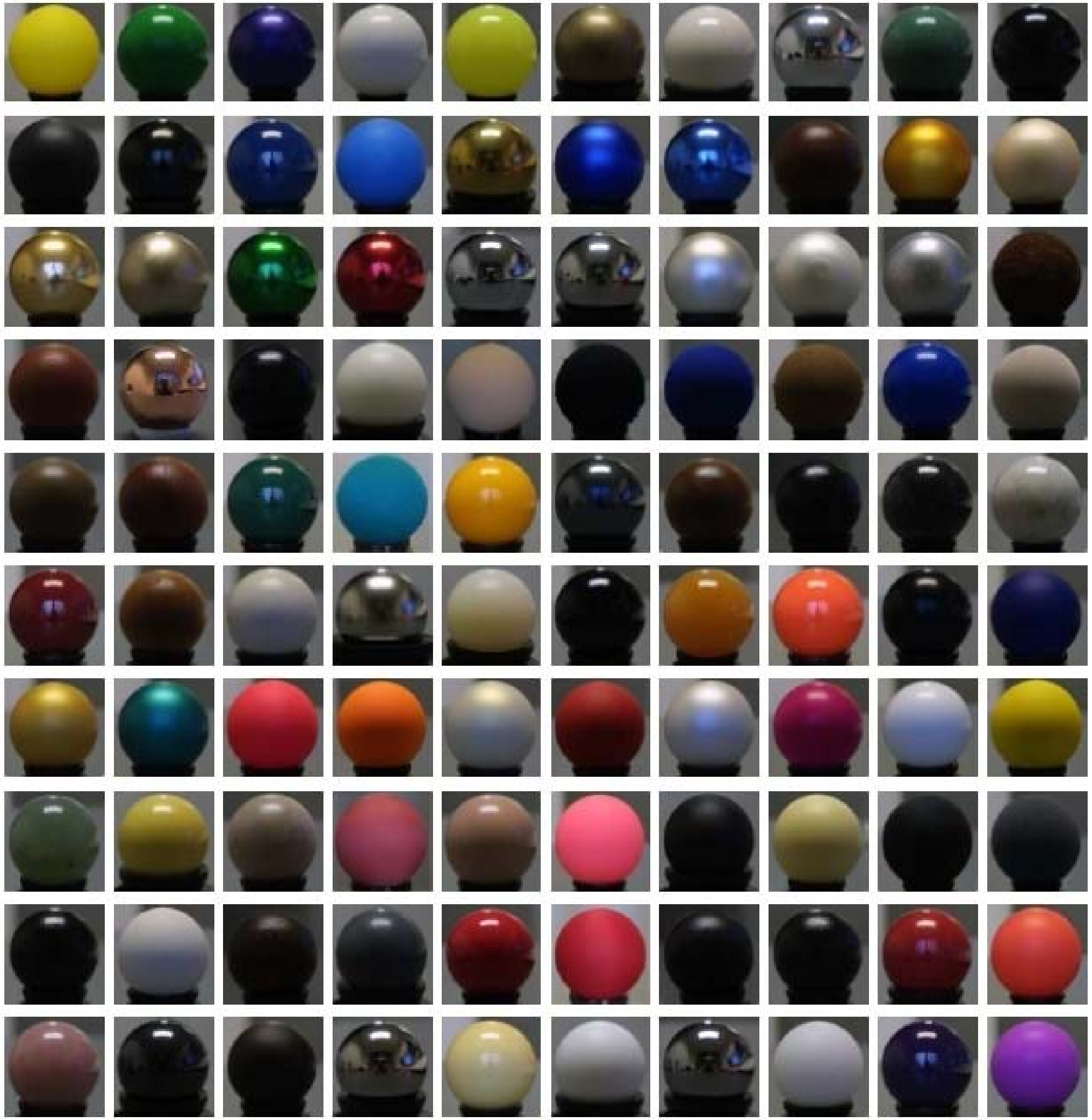


photograph



rendering using tabulated BRDF

Acquisition

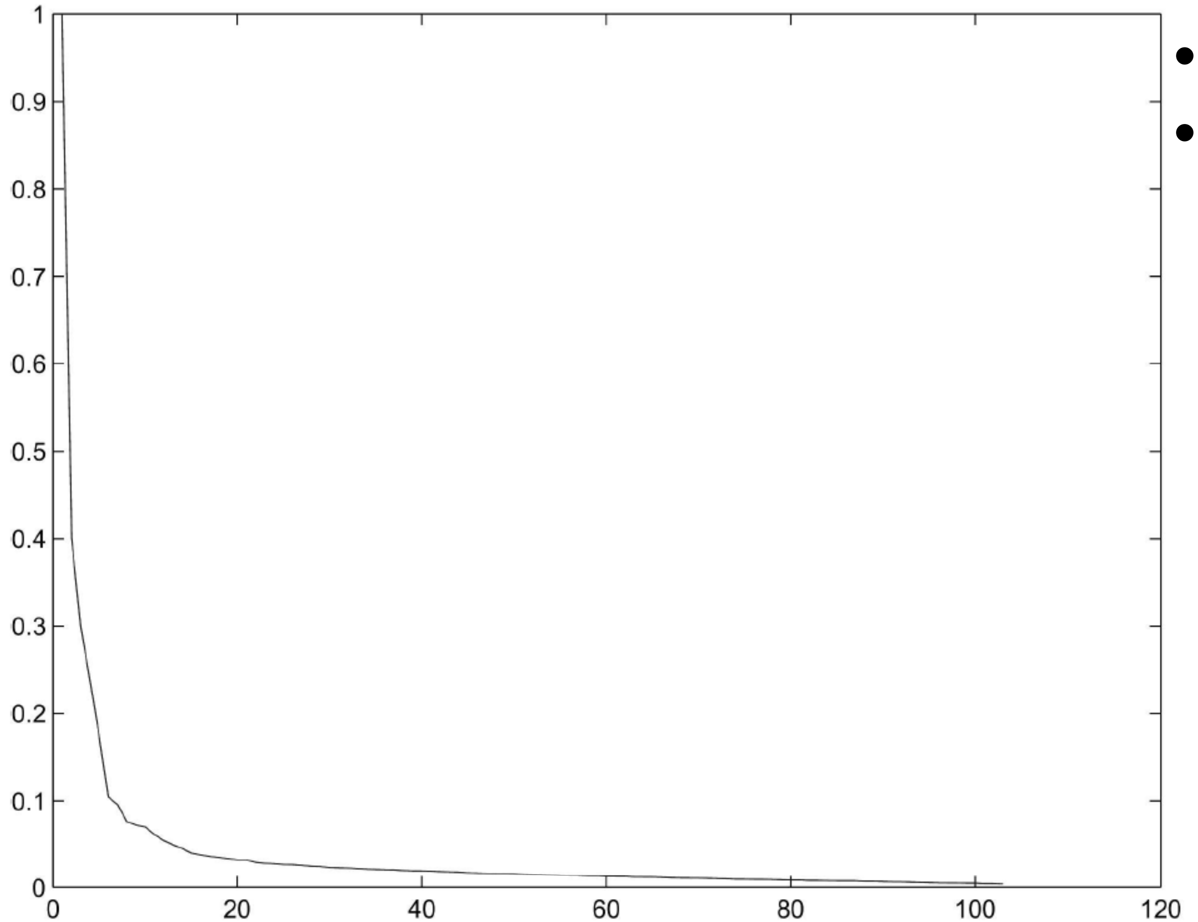


130 materials were scanned; 100 of them shown here

Tabulated BRDF

Linear dimension reduction

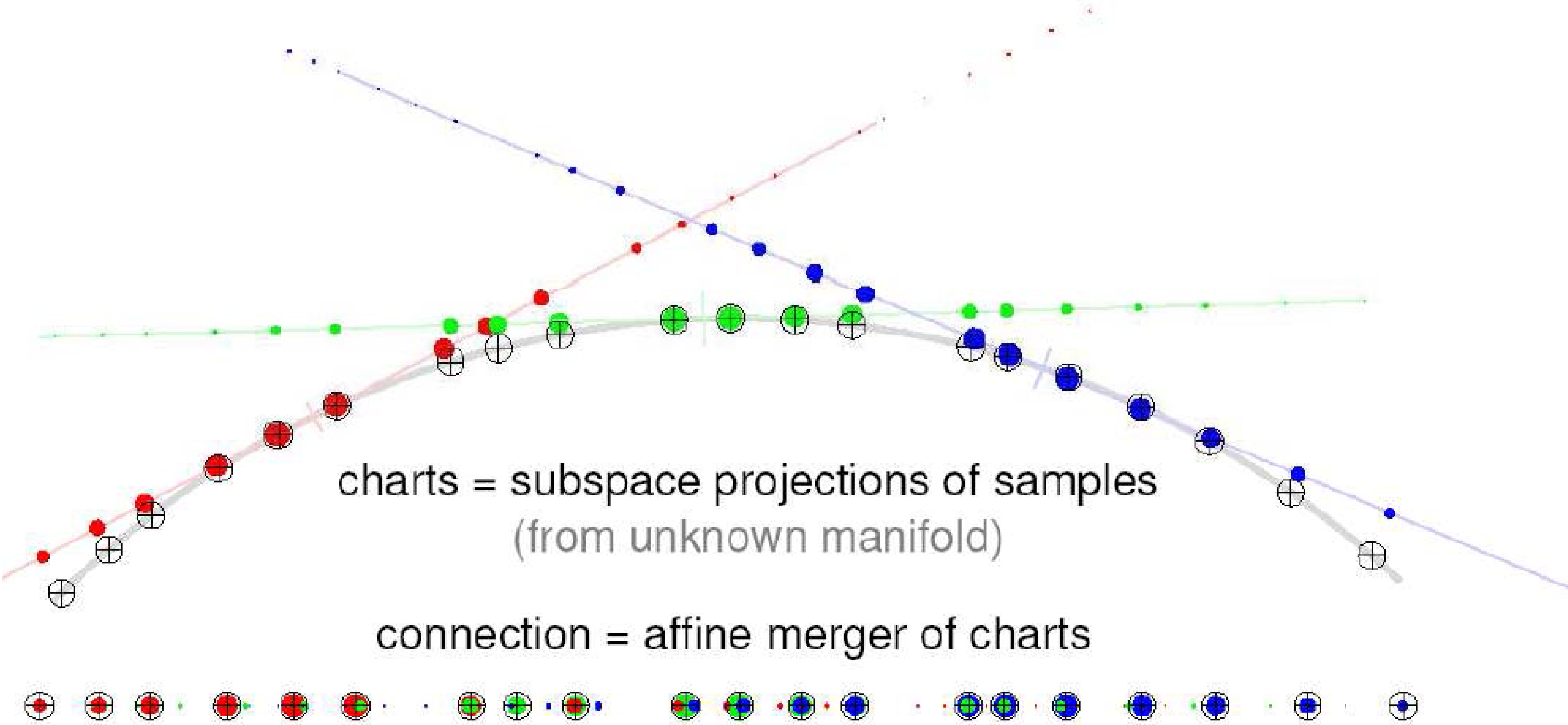
- SVD on the 4,374,000x104 matrix.



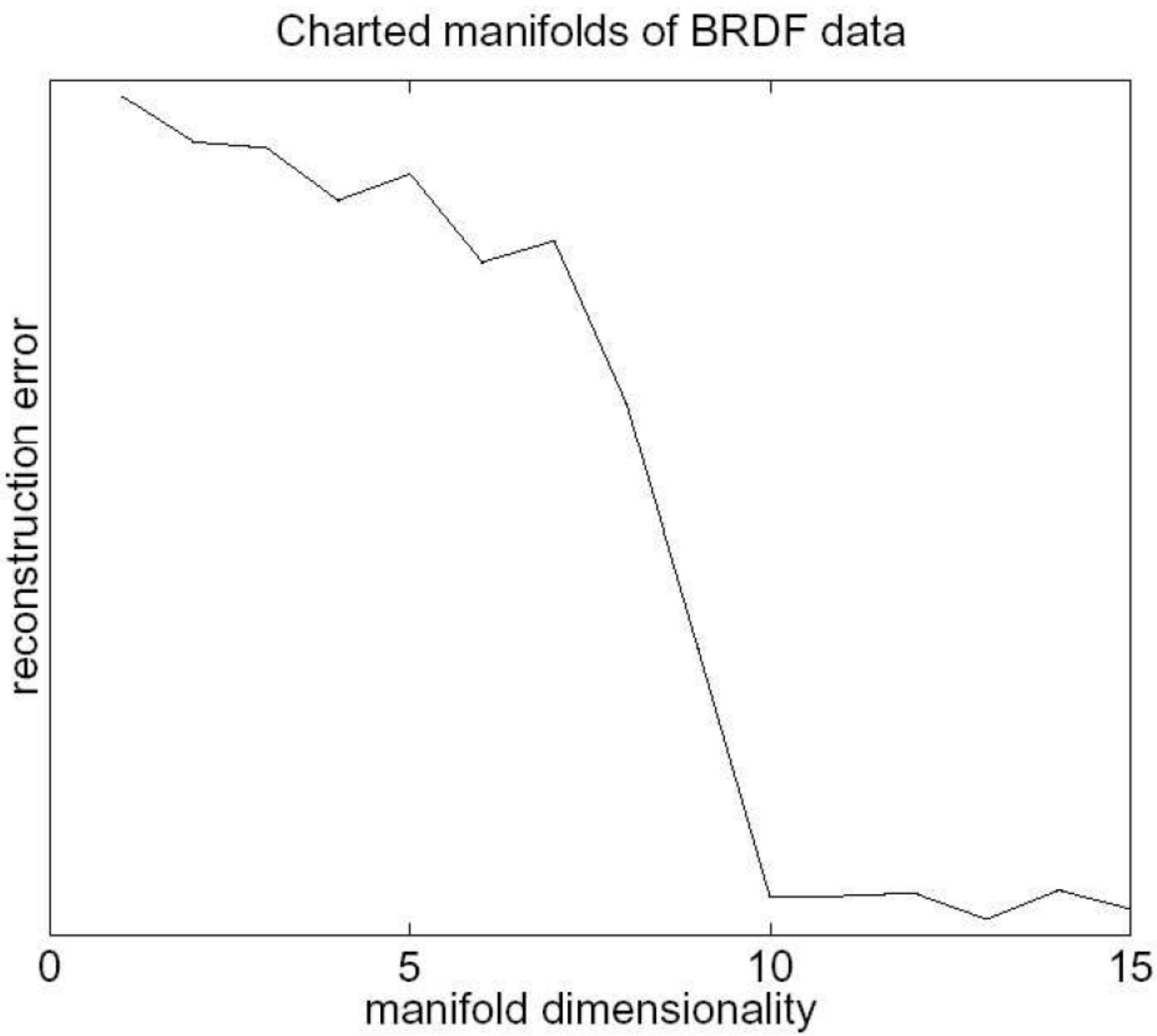
- 45D space
- It spans a space bigger than the space of all possible BRDFs
  1. more parameters than most models
  2. it interpolates invalid BRDF

Nonlinear dimension reduction

- Charting by Matt Brand



Nonlinear dimension reduction



- 10D gives good reconstruction
- Choose to work on 15D

Model construction

- A subject characterized each BRDF by 16 categories as yes, no and unclear: redness, greenness, blueness, specularness, diffuseness, glossiness, metallic-like, plastic-like, roughness, silverness, gold-like, fabric-like, acrylic-like, greasiness, dustiness, rubber-like
- SVM is used to build the model

Results

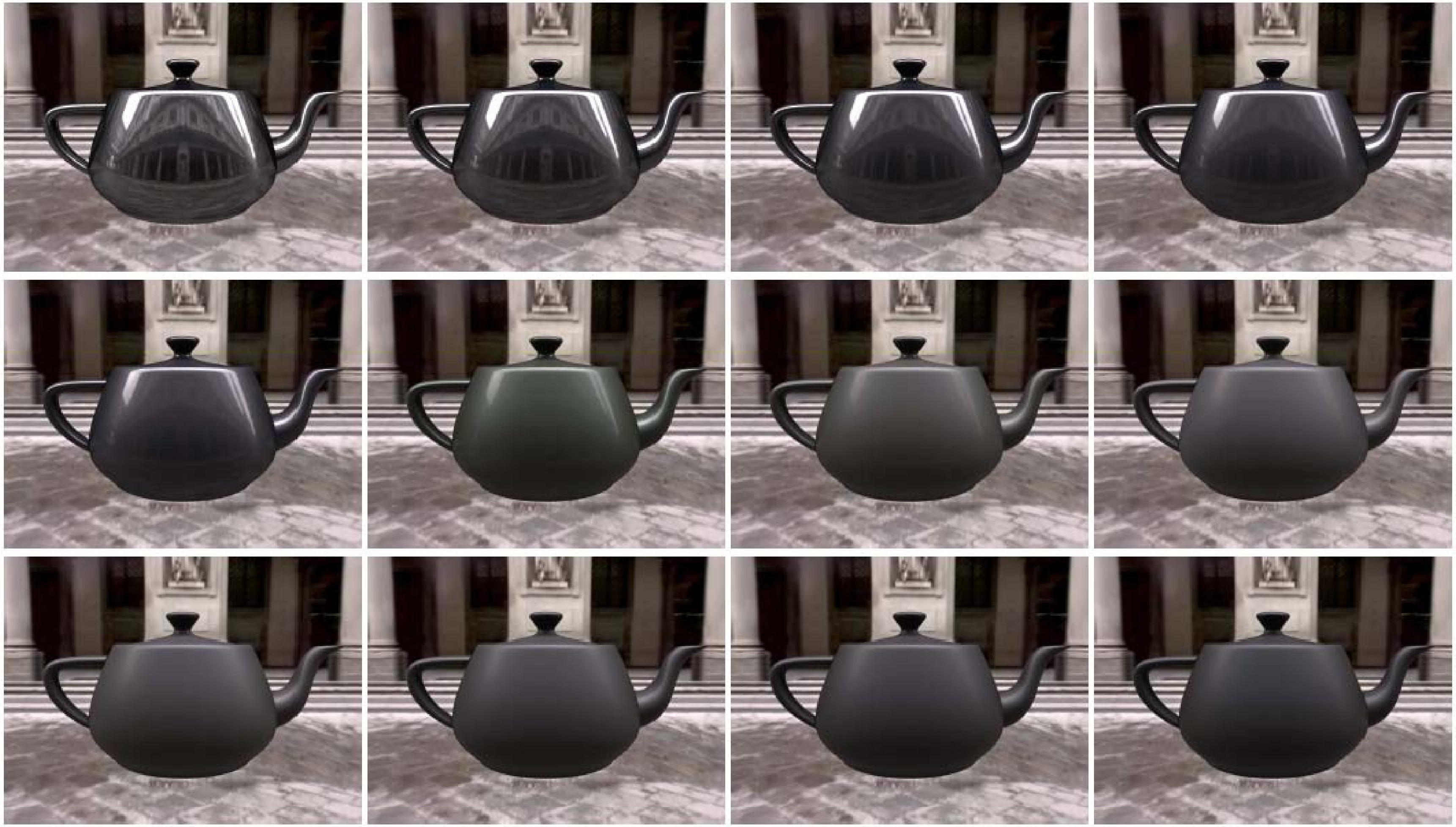

Gold paint
redness
Spec. Gold
Silver-ness
Blue Glossy Paint
Gold-like
Black Matte Plastic
Spec.-ness

Results


Copper
Green Acrylic
Violet Acrylic
Yellow Diffus. Paint
Rough -ness
Blue-ness
Metallic-like
Glossi-ness

Results

Polished steel



Black oxidized