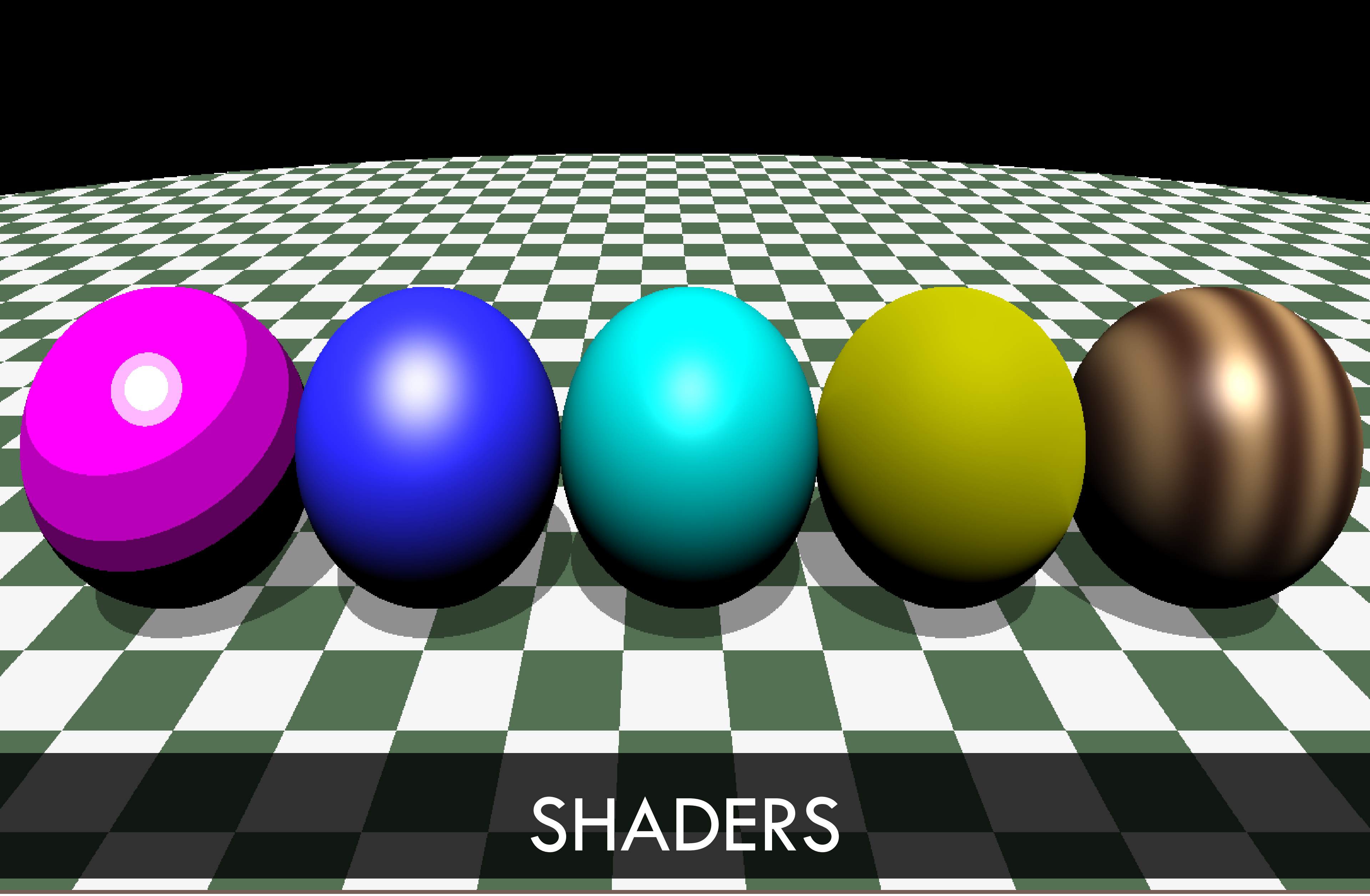


SHADERS

SEMINAR 5

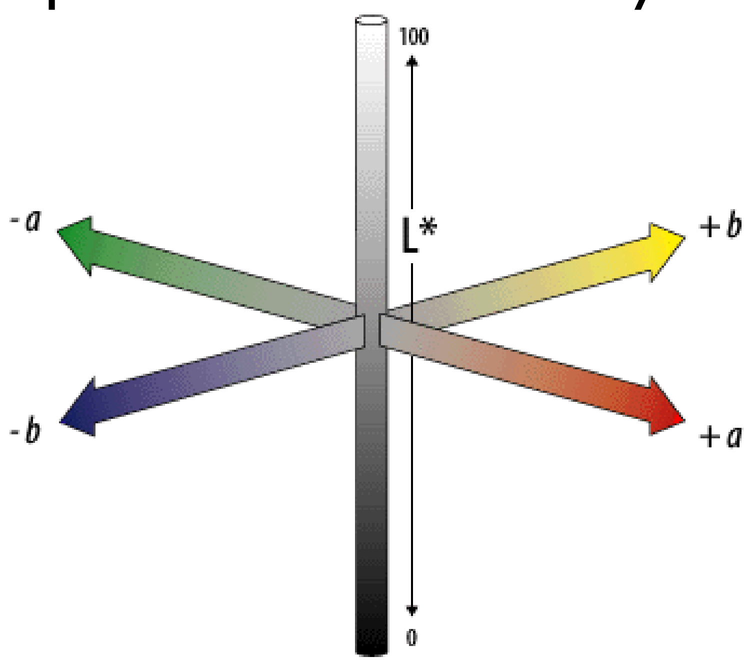
Computer Graphics 2

Experiment Discussion

- Color of sample
- Uniformity of color on sample
- Specular reflections on sample
- Similar specular reflections from both measurements

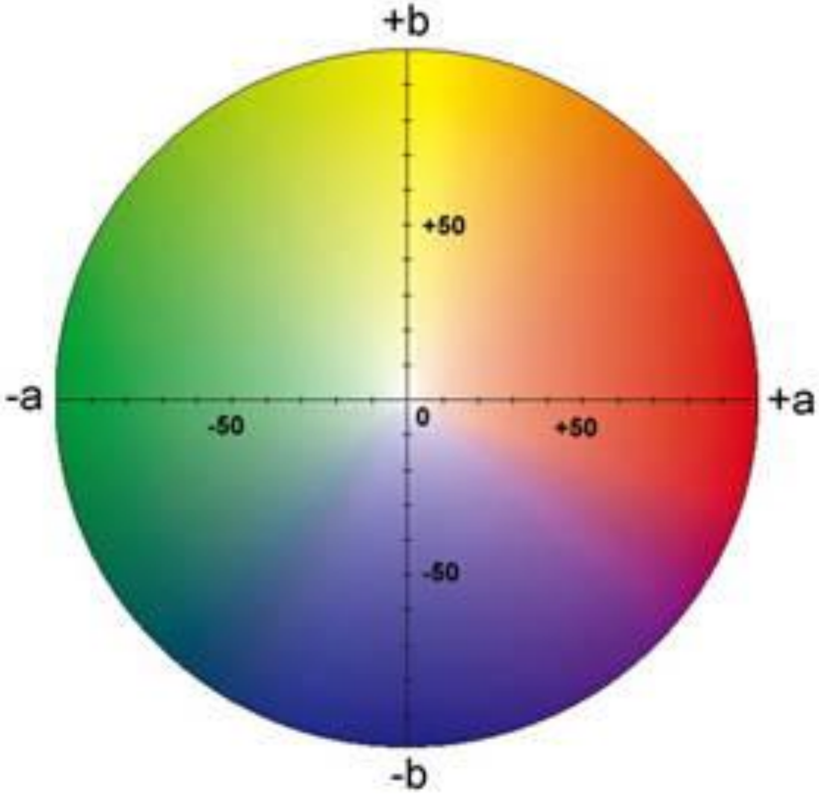
CIE L*a*b*

- Includes all perceivable colors
- Perceptually uniform
- L - lightness, close match to human perception
- a – color component from green to red
- b – color component from blue to yellow

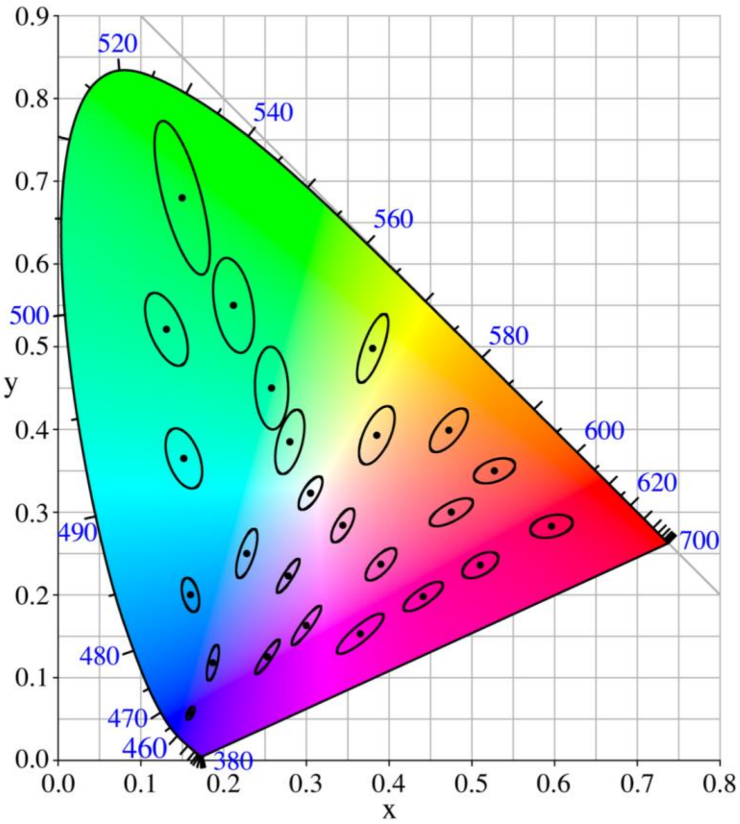


CIELab vs. RGB

CIELab

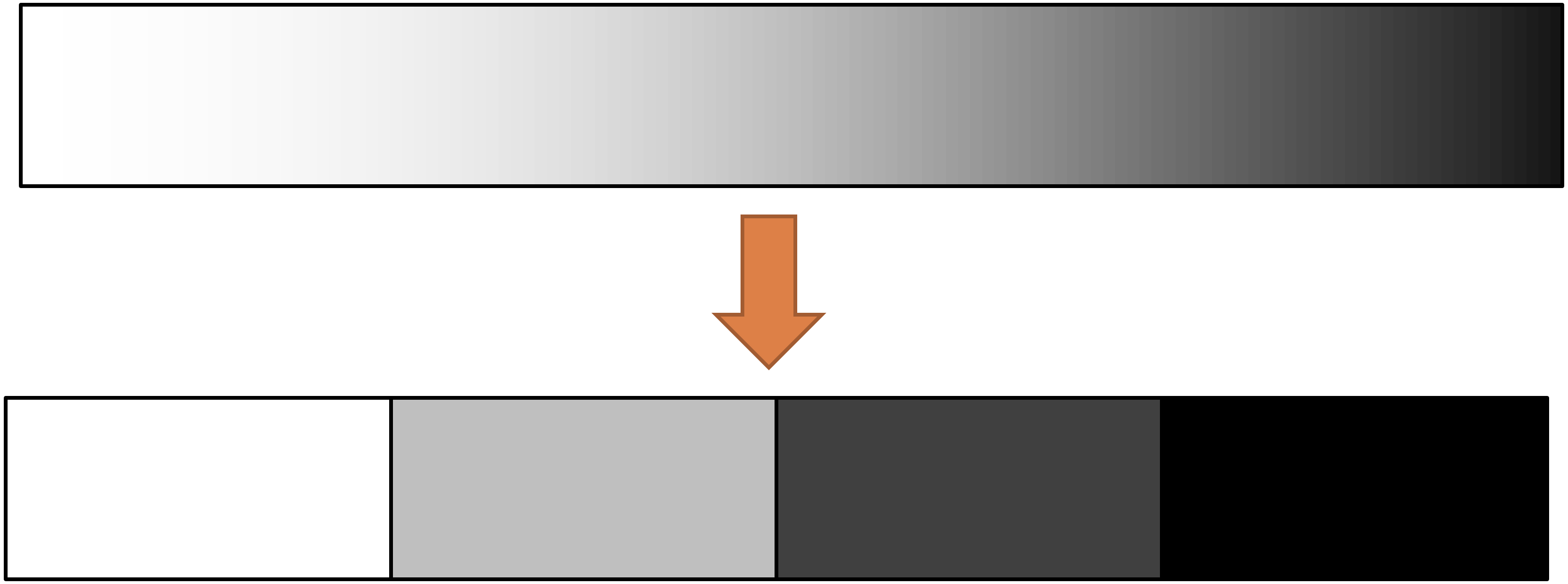


RGB



Toon Shader

- Discretize diffuse and specular factor
  - ~4 intensity values for diffuse factor
  - ~3 intensity values for specular factor



Cook Torrance Shader

- Surfaces are composed of microfacets:
  - Reflect incoming light
  - Multiple facets rendered in single pixel
  - Rough surface = slope varies greatly
  - Smooth surface = similarly oriented microfacets
- Focuses on specular reflection

$$specularColor = (\boldsymbol{n} \cdot \boldsymbol{l}) * specular * (SunColor \wedge MaterialColor)$$

Where: $$specular = \frac{F_\lambda(\theta) * D * G}{\pi(\boldsymbol{n} \cdot \boldsymbol{l})(\boldsymbol{n} \cdot \boldsymbol{v})}$$

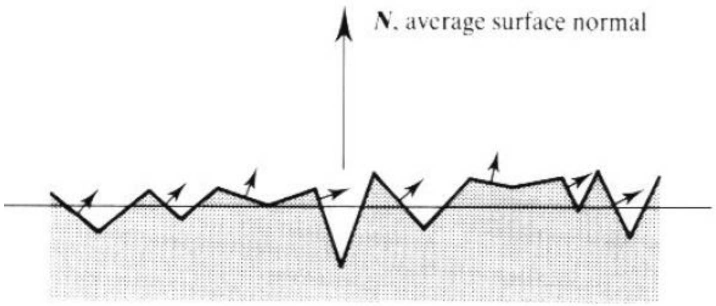
$F_\lambda(\theta)$ Fresnel

D distribution of microfacets

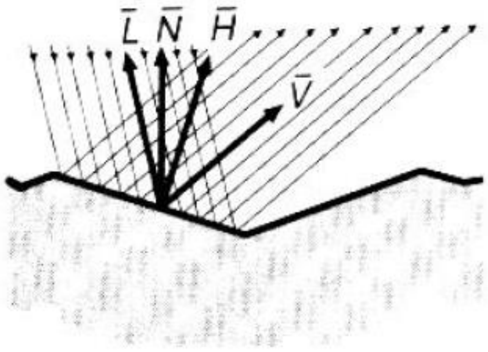
G geometric attenuation

Microfacet Motivation

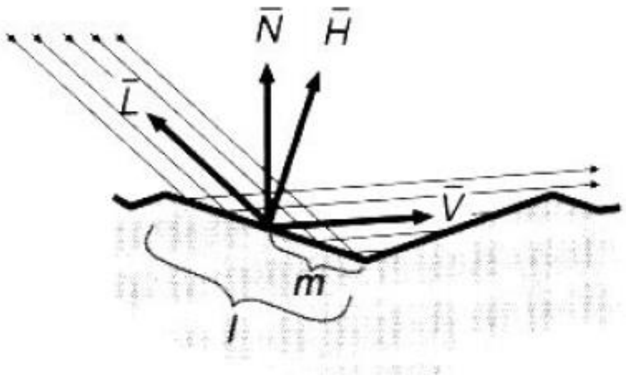
Surface composed by microfacets:



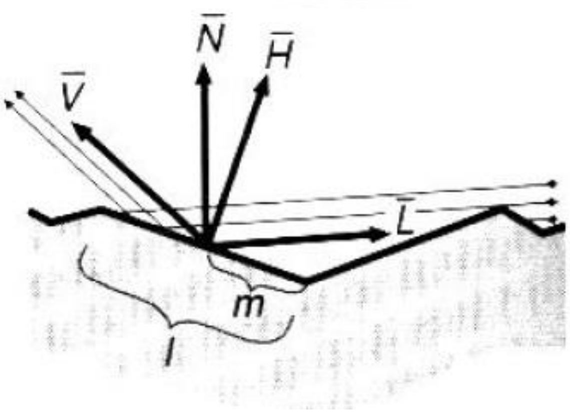
Illuminated microfacet:



Masking of reflected light:



Shadowing of incoming light:



Geometric Attenuation

- Microfacets block incoming light
- Value from [0, 1] which represents remaining light
- Microfacets are assumed to be V-shaped grooves
- There are three cases, final factor is minimal value

The light is reflected without interference: $G_a = 1$

Light is blocked after reflection: $G_b = \frac{2(\boldsymbol{n} \cdot \boldsymbol{h})(\boldsymbol{n} \cdot \boldsymbol{v})}{\boldsymbol{v} \cdot \boldsymbol{h}}$

Light is blocked before reaching next microfacet: $G_c = \frac{2(\boldsymbol{n} \cdot \boldsymbol{h})(\boldsymbol{n} \cdot \boldsymbol{l})}{\boldsymbol{l} \cdot \boldsymbol{h}}$

Final attenuation factor: $G = \min(G_a, G_b, G_c)$

Roughness – Backmann distribution

- Defines fraction of microfacets oriented the same way as half vector **h**
  - On smooth surfaces all light is close to specular reflection
  - On rough surfaces the light is more distributed
- Can be calculated with e.g. Beckmanns distribution

$$D = \frac{1}{\pi m^2 cos^4\alpha} e^{-\left(\frac{\tan\alpha}{m}\right)^2} = \frac{1}{\pi m^2 cos^4\alpha} e^{\left(\frac{(\boldsymbol{n}\cdot\boldsymbol{h})^2-1}{m^2(\boldsymbol{n}\cdot\boldsymbol{h})^2}\right)}$$

Where: m is material roughness

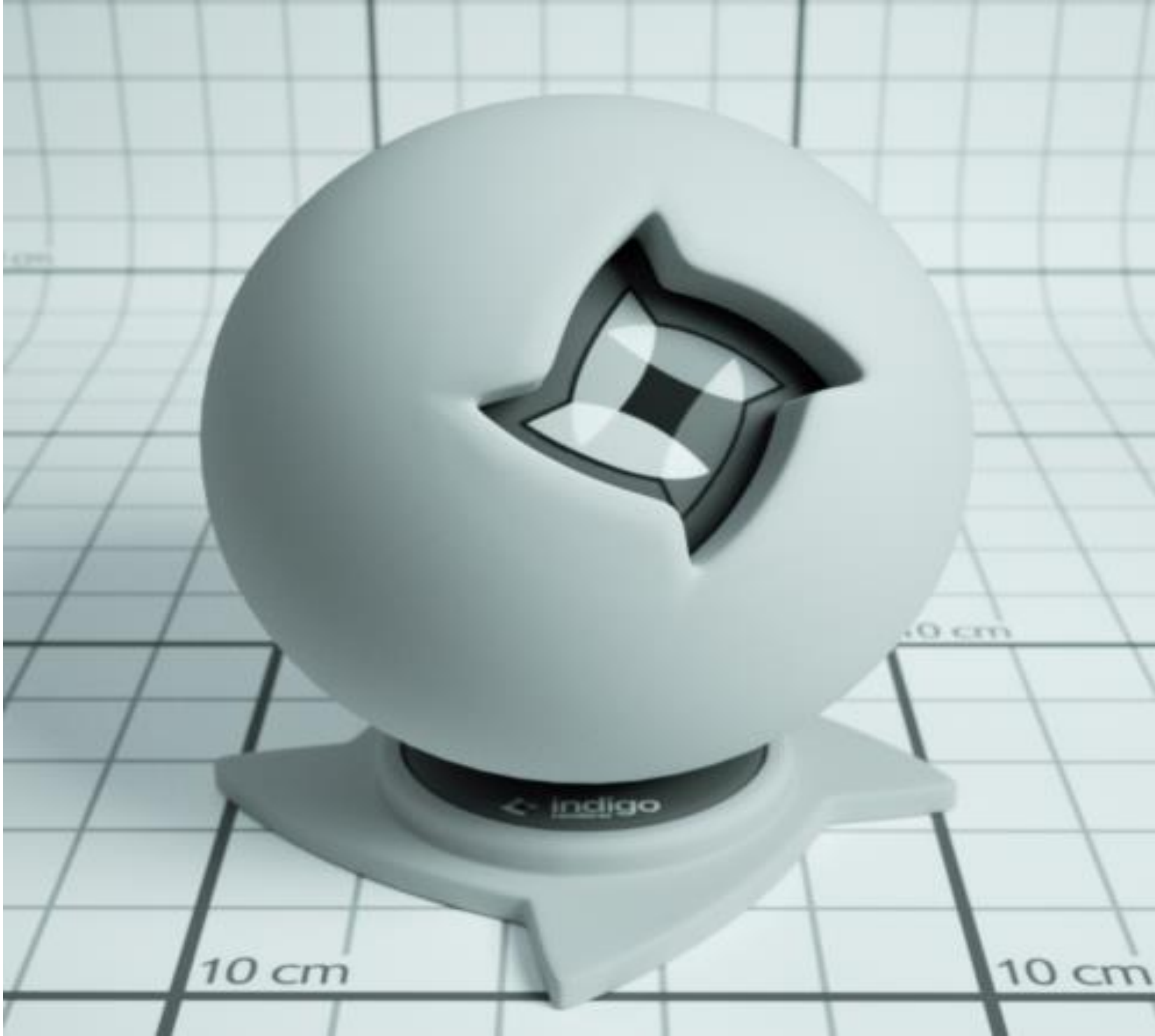
Fresnel – Schlick approximation

- Defines what fraction of incoming light is reflect and transmitted
- Schlick approximation is used, due to complexity of original formula

$$F_\lambda(\theta) = f_\lambda + (1 - f_\lambda)(1 - \theta)^5$$

Where: f_λ reflectance at normal distance

$\theta = \boldsymbol{h} \cdot \boldsymbol{v}$ angle between half and view vectors


indigo
10 cm
10 cm

Oren Nyar Shader

- Lambertian model inappropriate for many materials
- Surfaces can be modeled by microfacets
- Camera projects several facets into one pixel
- Takes into account masking, shadowing, interreflections
- Takes a single parameter the roughness of a surface
- More info in original paper:
  - http://www1.cs.columbia.edu/CAVE/publications/pdfs/Oren_SIGGRAPH94.pdf

Oren Nyar Shader - Formulas

n = normal **l** = light direction **v** = view direction **e** = eye direction

$$\alpha = \max(\measuredangle \boldsymbol{nv}, \measuredangle \boldsymbol{nl})$$

$$\beta = \min(\measuredangle \boldsymbol{nv}, \measuredangle \boldsymbol{nl})$$

$$A = 1 - 0.5 \frac{roughness^2}{roughness^2 + 0.57}$$

$$B = 0.45 \frac{roughness^2}{roughness^2 + 0.09}$$

$$C = \sin\alpha * \tan\beta$$

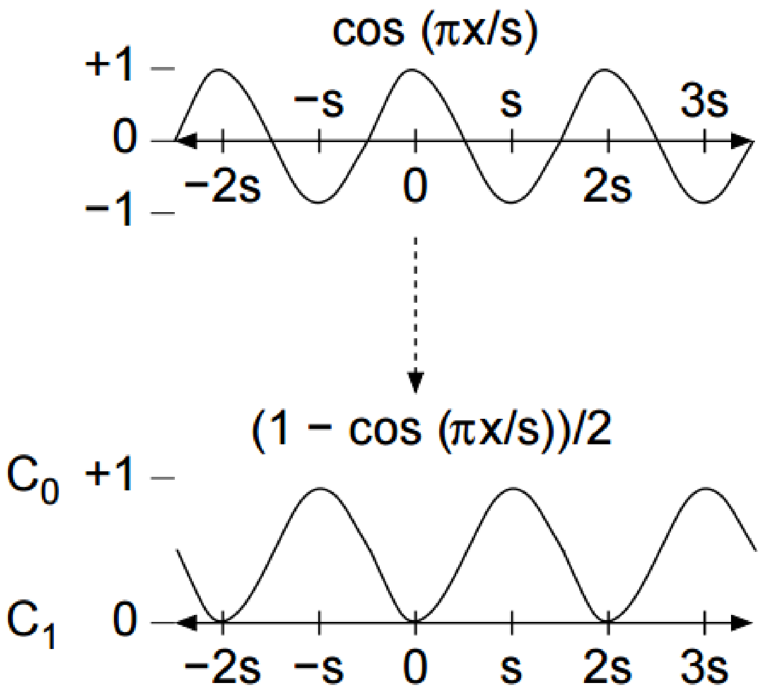
$$\gamma = (\boldsymbol{e} - \boldsymbol{n}(\boldsymbol{e} \cdot \boldsymbol{n})) \cdot (\boldsymbol{l} - \boldsymbol{n}(\boldsymbol{l} \cdot \boldsymbol{n}))$$

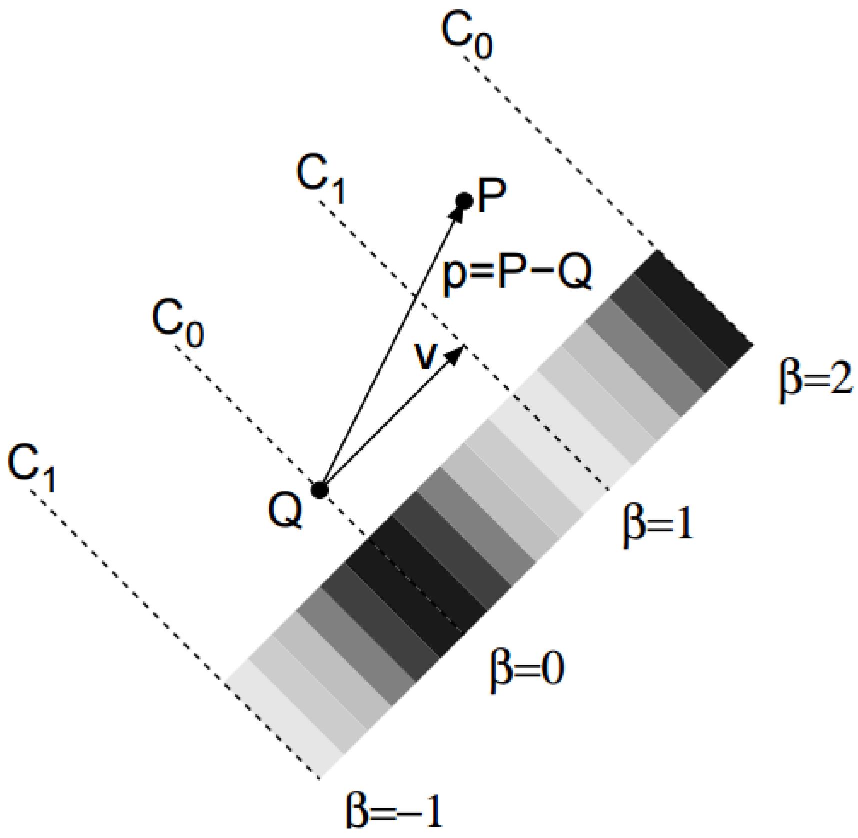
$$L_1 = \max(0, \boldsymbol{n} \cdot \boldsymbol{l}) * (A + B * \max(0, \gamma\ ) * C)$$

Gradient Shader (1)

- Creates cosinusiodal wave
- Project vector from origin to point onto gradient direction
- Calculate cosinus of gradient value
- Transform cosinus from [-1, 1] to [0, 1] to get alpha
- Use alfa blending between two shaders S0 and S1

Gradient Shader (2)





Questions?