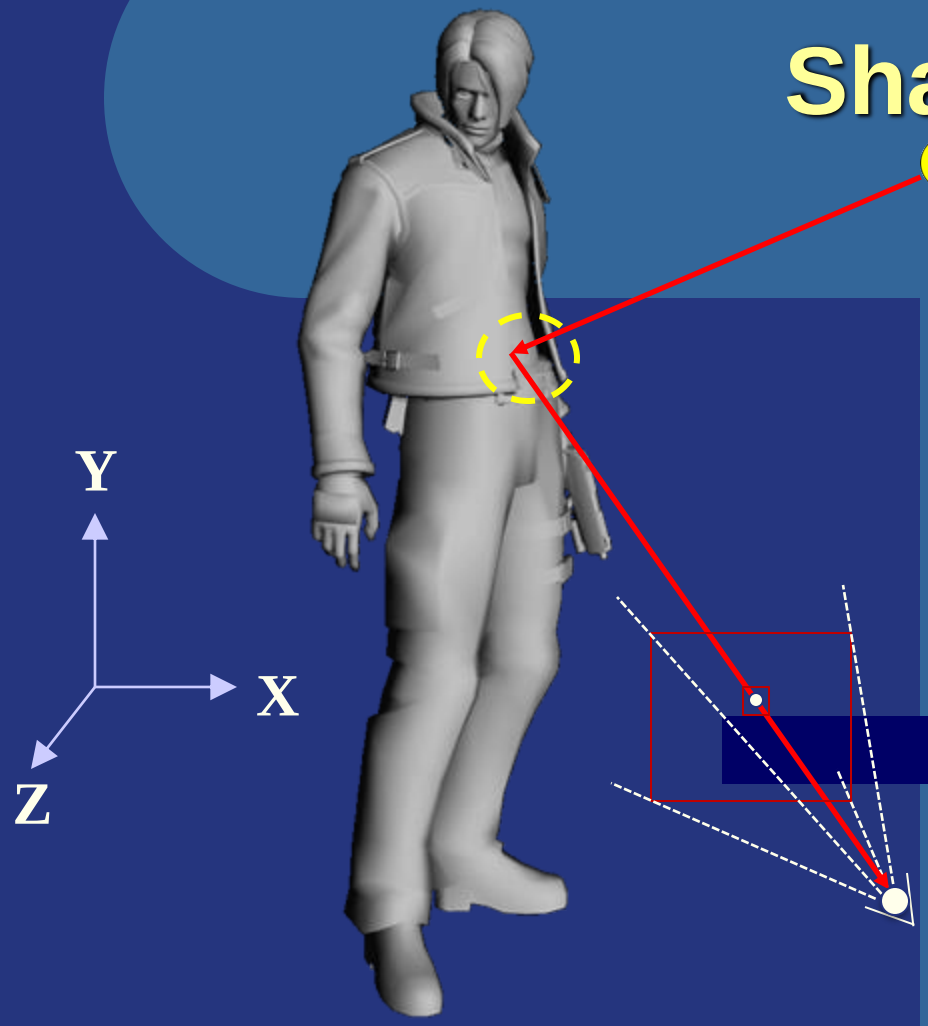
Shading



Slides by Ulf Assarsson and Tomas Akenine-Möller

Department of Computer Engineering

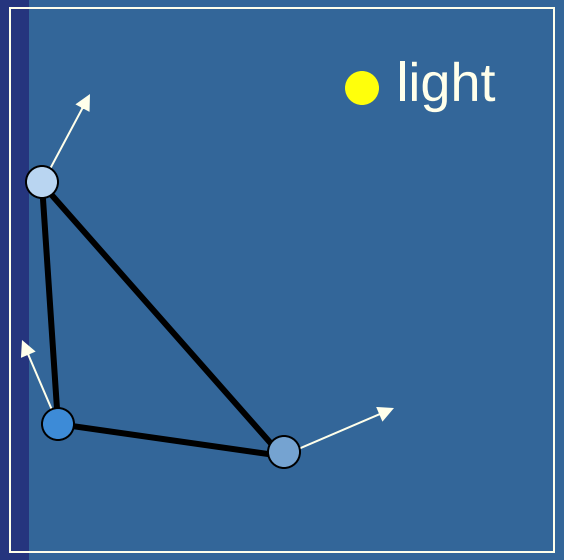
Chalmers University of Technology

Overview of today's lecture

- First, a simple most basic real-time lighting model
  - Shading parts: ambient, diffuse, specular, emission.
    - It is also OpenGL's old fixed pipeline lighting model
- Physically-based shading (PBS)
  - Metalness (vs dielectric) in percent,
  - Fresnel: F_0. (”reflection color”, base reflectance)
  - Specularity: shininess or roughness,
  - Base color: $\boldsymbol{c}_{base}$
- Fog
- Gamma correction
- Transparency and alpha

Lighting and Shading

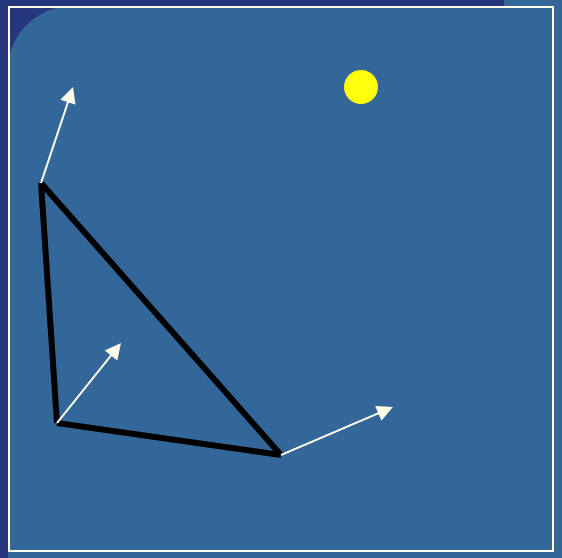
Typically done in the fragment shader.



Lighting computation

Full shading

A basic lighting model

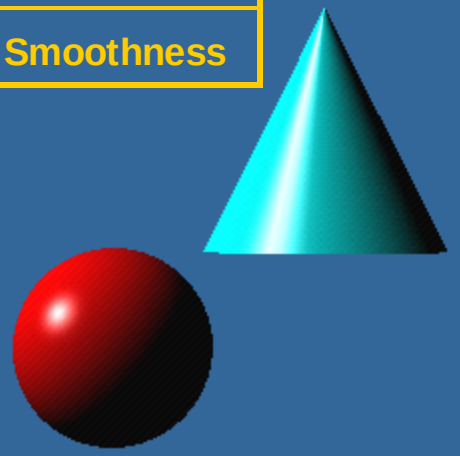


Light: (r,g,b)

DIFFUSE	Base color
SPECULAR	Highlight Color
AMBIENT	Low-light Color
EMISSION	Glow Color
SHININESS	Surface Smoothness

Material:

- Ambient (r,g,b,a)
- Diffuse (r,g,b,a)
- Specular (r,g,b,a)
- Emission (r,g,b,a) =”self-glowing color”



The ambient/diffuse/specular/emission lighting contribution model

- The most basic real-time model:
- Light interacts with material and change color at bounces:

$$\mathbf{outColor}_{rgb} \sim \mathbf{material}_{rgb} \text{ Ä } \mathbf{lightColor}_{rgb}$$

$\otimes$ is here component-wise mult. – not cross product.

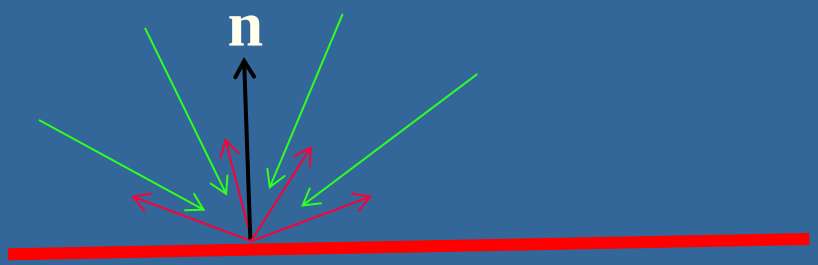
- **Ambient** light: incoming homogeneous background light from all directions (view-independent and light-position- independent color)

$$\mathbf{i}_{amb} = \mathbf{m}_{amb}\, \mathbf{l}_{amb}$$

We assume homogeneous background light

i.e., $(i_r, i_g, i_b) = (m_r, m_g, m_b)\ (l_r, l_g, l_b) = (m_r l_r, m_g l_g, m_b l_b)$

Ambient



The ambient/diffuse/specular/emission model

- The most basic real-time model:
- Light interacts with material and change color at bounces:

$$\mathbf{outColor}_{rgb} \sim \mathbf{material}_{rgb} \text{ Ä } \mathbf{lightColor}_{rgb}$$

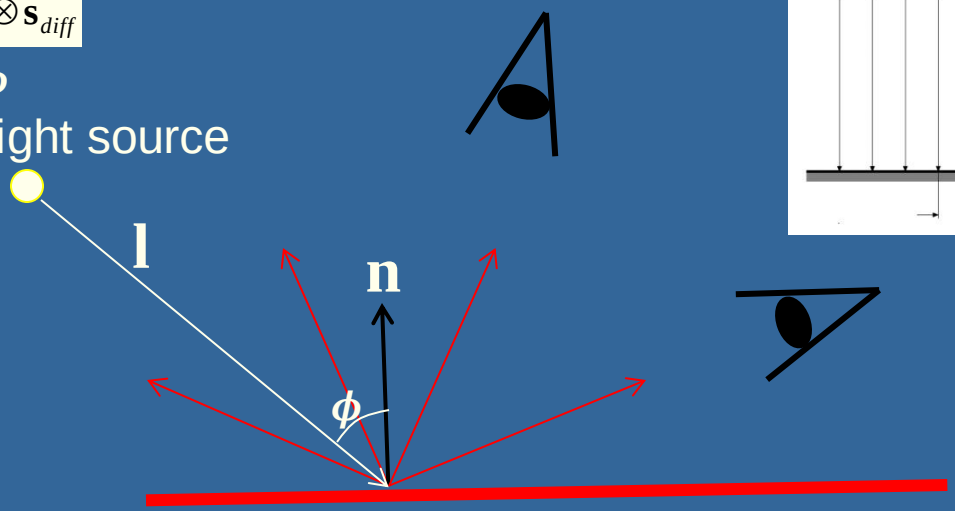
- Ambient light: incoming homogeneous background light from all directions (view-independent and light-position- independent color)
- **Diffuse** light: from light source, bouncing equally into **all** directions (view independent) due to surface being very **rough** on microscopic level

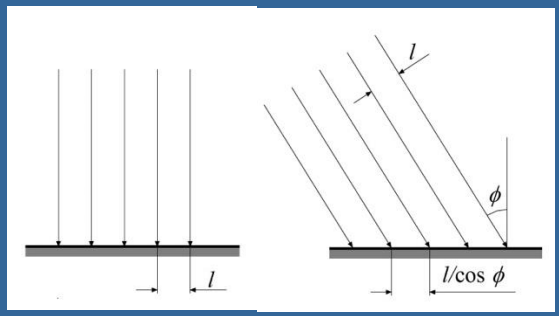
Just scale light intensity with incoming angle

$$\mathbf{i}_{diff} = (\mathbf{n} \cdot \mathbf{l})\mathbf{m}_{diff} \otimes \mathbf{s}_{diff}$$

$$(n \cdot l) = cos\ \phi$$

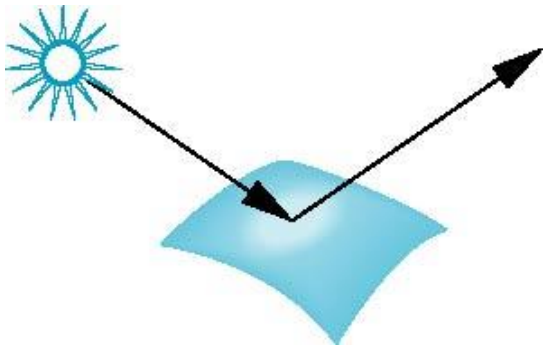
Amb + Diff



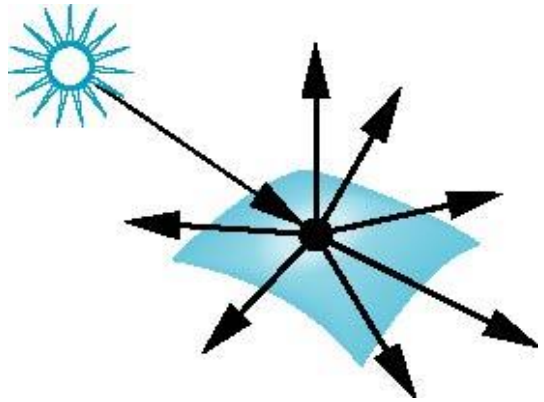


A 100% diffuse material is called a ”Lambertian” Surface

- A perfectly diffuse reflector
- Light scattered equally in all directions



Highly reflective surface (specular)



Fully diffuse surface (Lambertian)

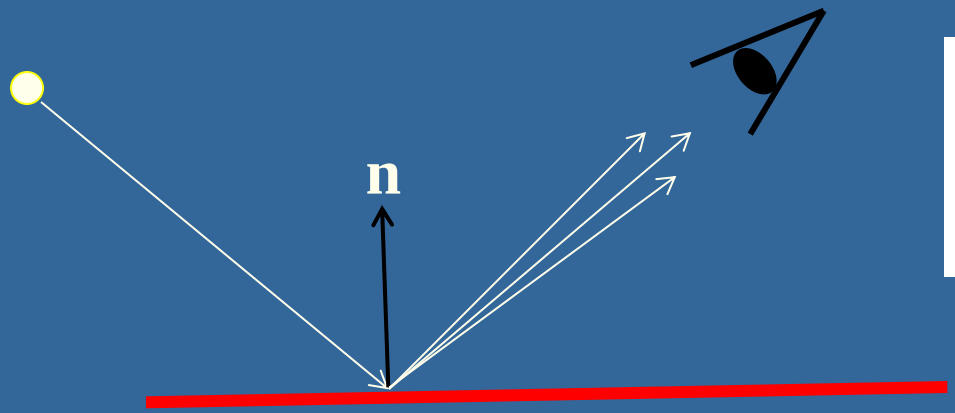
The ambient/diffuse/specular/emission model

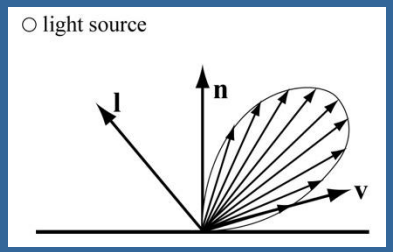
- The most basic real-time model:
- Light interacts with material and change color at bounces:

$$\mathbf{outColor}_{rgb} \sim \mathbf{material}_{rgb} \text{ Ä } \mathbf{lightColor}_{rgb}$$

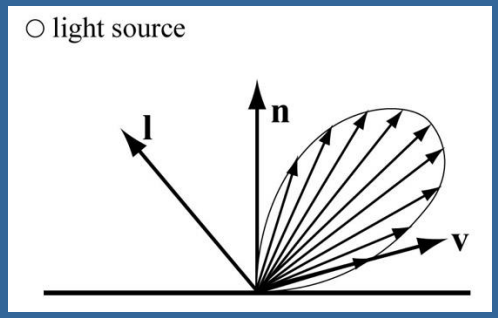
- Ambient light: incoming homogeneous background light from all directions (view-independent and light-position- independent color)
- Diffuse light: from light source, bouncing equally into **all** directions (view independent) due to surface being very **rough** on microscopic level
- **Specular** light: the part that spreads mostly in the reflection direction (often same color as light source)

Amb + Diff + Spec



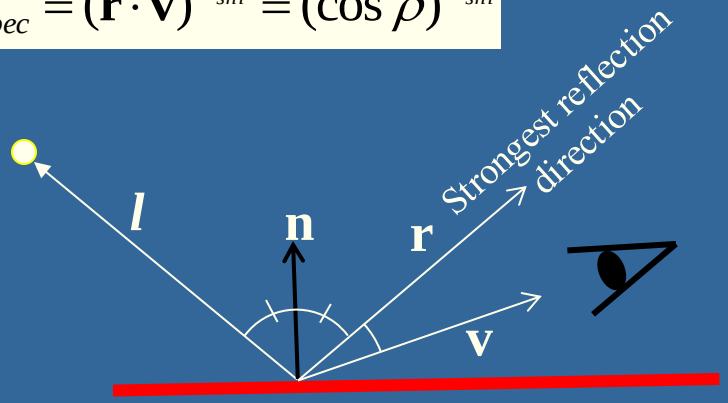


Specular: Phong's model



- Phong's specular highlight model
- Scales the light that reflects along **v**, i.e., towards eye, by angle from main reflection direction:

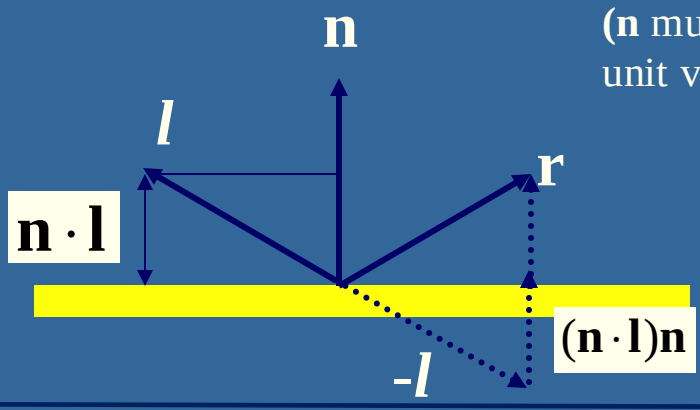
$$i_{spec} = (\mathbf{r}\cdot\mathbf{v})^{m_{shi}} = (\cos\rho)^{m_{shi}}$$



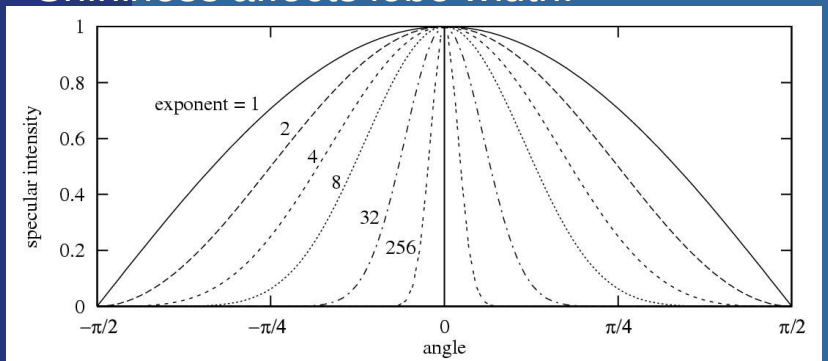
How to compute **r**:

$$\mathbf{r} = -\mathbf{l} + 2(\mathbf{n}\cdot\mathbf{l})\mathbf{n}$$

(**n** must be unit vector)



Shininess affects lobe width:



Full computation:

$$\mathbf{i}_{spec} = \max(0,(\mathbf{r}\cdot\mathbf{v}))^{m_{shi}}\,\mathbf{m}_{spec} \otimes \mathbf{s}_{spec}$$

max() – due to not wanting negative light.
Also check that cam is on same surface side as light source.

Next slide: Blinns highlight formula: $(n\cdot h)^m$

Specular: Blinn's model

Blinn proposed replacing **v**·**r** by **n**·**h**, where

h = (**l**+**v**)/|**l** + **v**|

h is halfway between **l** and **v**

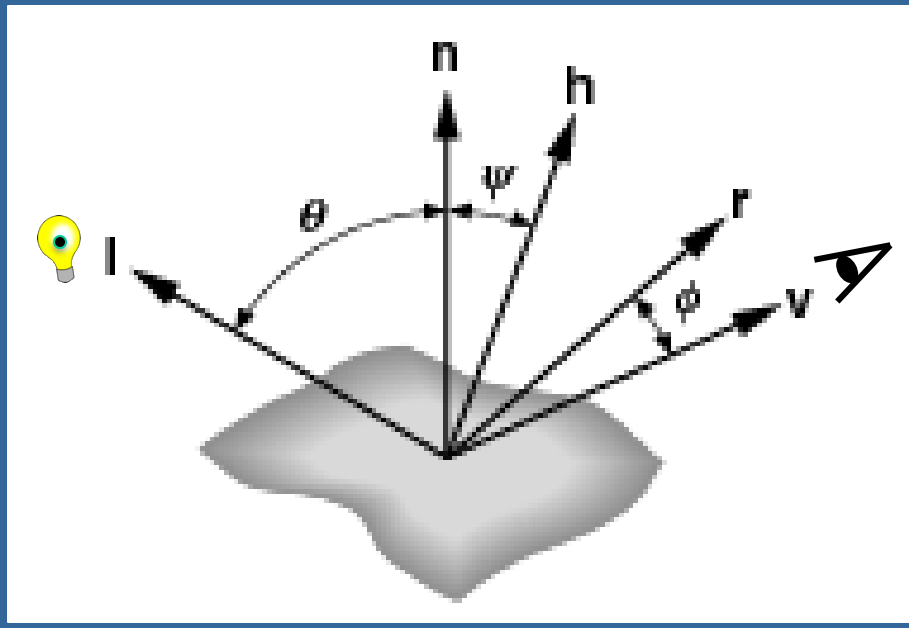
If **n**, **l**, and **v** are coplanar:

$$\psi = \phi/2$$

Must then adjust exponent

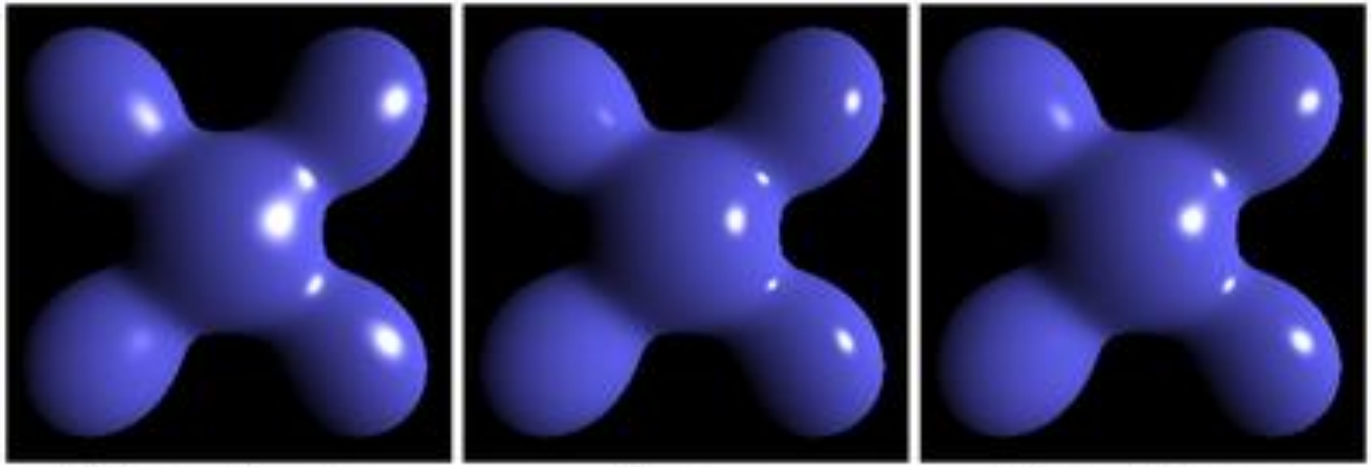
so that

$(\mathbf{n}\cdot\mathbf{h})^{e'} \approx (\mathbf{r}\cdot\mathbf{v})^{e}$, $(e' \approx 4e)$



If the surface is rough, there is a probability distribution of the microscopic normals **n**. This means that the intensity of the reflection is decided by how many percent of the microscopic normals are aligned with **h**. And that probability often scales with how close **h** is to the macroscopic surface normal **n**.

$$\mathbf{i}_{spec} = \max(0, (\mathbf{h}\cdot\mathbf{n})^{m_{shi}})\mathbf{m}_{spec} \otimes \mathbf{s}_{spec}$$



Blinn

Phong

Blinn
(higher exponent)

$(n \cdot h)^s$ $(r \cdot v)^s$ $(n \cdot h)^{4s}$

The ambient/diffuse/specular/emission model

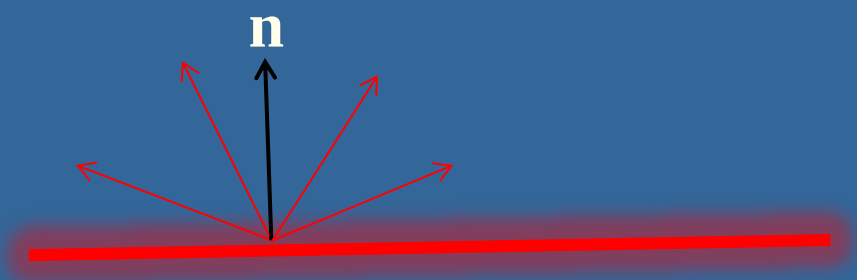
- The most basic real-time model:
- Light interacts with material and change color at bounces:

$$\mathbf{outColor}_{rgb} \sim \mathbf{material}_{rgb} \text{ Ä } \mathbf{lightColor}_{rgb}$$

- Ambient light: incoming homogeneous background light from all directions (view-independent and light-position- independent color)
- Diffuse light: from light source, bouncing equally into **all** directions (view independent) due to surface being very **rough** on microscopic level
- Specular light: the part that spreads mostly in the reflection direction (often same color as light source)
- **Emission**: self-glowing surface

Amb + Diff + Spec + Em

$$\mathbf{i}_{em} = \mathbf{m}_{emission}$$



The ambient/diffuse/specular/emission model

- Summary of formulas:

Ambient: $\mathbf{i}_{amb} = \mathbf{m}_{amb}\,\mathbf{l}_{amb}$ "color result from homogeneous background light"

Diffuse: $(\boldsymbol{n}\cdot\boldsymbol{l})\,\mathbf{m}_{diff}\,\mathbf{l}_{diff}$ "Scale illumination by surface's angle to light source"

Specular:

- Phong: $(\boldsymbol{r}\cdot\boldsymbol{v})^{shi}\,\mathbf{m}_{spec}\,\mathbf{l}_{spec}$
- Blinn: $(\boldsymbol{h}\cdot\boldsymbol{n})^{shi}\,\mathbf{m}_{spec}\,\mathbf{l}_{spec}$

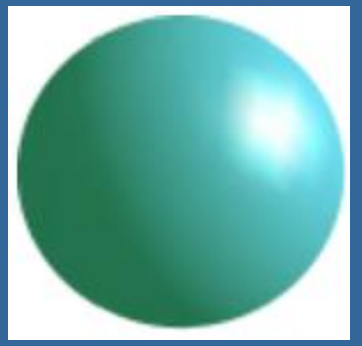
"strength of highlight based on viewing angle from main reflection direction"

Emission: $\mathbf{m}_{emission}$ "Self-glowing color"

Ambient

Amb + Diff

Amb + Diff + Spec



Amb + Diff + Spec + Em

Physically-based Shading (PBS)

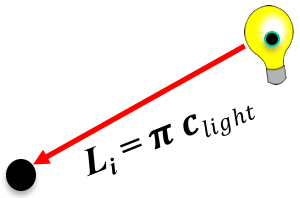
Physically-based Shading (PBS)

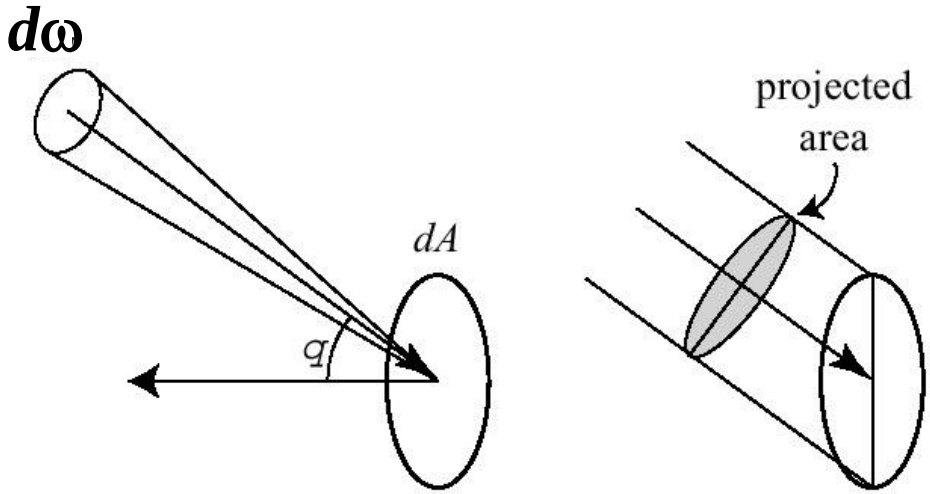
Radiance

- In graphics, we typically use rgb-colors, $\boldsymbol{c} = (c_r, c_g, c_b)$, and mean the intensity or ***radiance*** for the red, green, and blue light.
- Radiance, L : a radiometric term. What we store in a pixel is the radiance towards the eye through that pixel: a tripplet $\boldsymbol{L} = (L_r, L_g, L_b)$
  - Radiance = the amount of electromagnetic radiation leaving or arriving at a point on a surface (per unit solid angle per unit projected area)
- Radiance is ”power per unit projected area per unit solid angle”

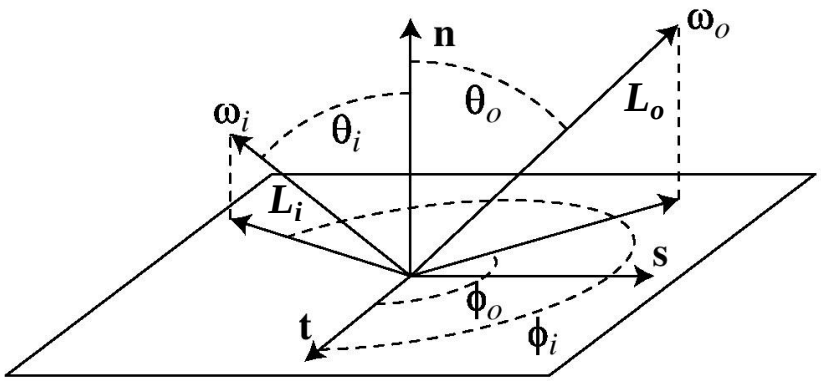
Radiance from a specific *direction* uses differentials, where the cone of the solid angle becomes an infinitesmally thin ray.

Hence, in graphics we often *sloppily* talk about the radiance from an incoming direction to a surface point.

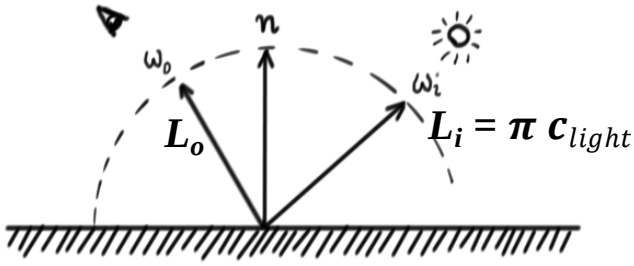




BRDF



- BRDF = Bidirectional Reflection Distribution Function
- Is a material description, $f(\omega_i, \omega_o)$
- What the BRDF *describes*: how much of the incoming radiance $\boldsymbol{L}_i$ from a given direction $\boldsymbol{\omega}_i$ that will leave in a given outgoing direction $\boldsymbol{\omega}_o$.



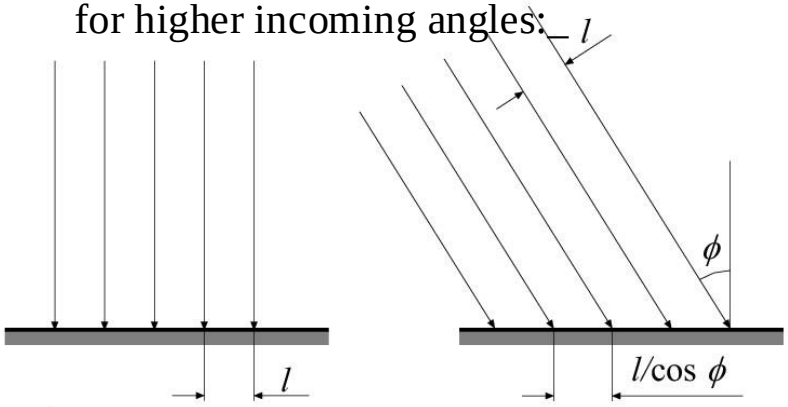
How to compute color, i.e outgoing radiance $\boldsymbol{L}_o$ from a point light:

$$\boldsymbol{L}_o(\boldsymbol{\omega}_o) = f(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\boldsymbol{L}_i(\omega_i)(\boldsymbol{n} \cdot \boldsymbol{\omega}_i)$$

$$\boldsymbol{L}_o(\boldsymbol{\omega}_o) = f(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\,\pi \boldsymbol{c}_{light}(\boldsymbol{n} \cdot \boldsymbol{\omega}_i)$$

where π comes from that the definition of radiance uses differentials $d\omega_i$ and integrates a cosine factor $(\boldsymbol{n} \cdot \boldsymbol{\omega}_i)$ for the hemisphere. The brdf, *f()*, contains a division by π, which cancel out π.

The cosines, $(\boldsymbol{n} \cdot \boldsymbol{\omega}_i)$, comes from decreased incoming intensity for higher incoming angles:



A fully diffuse (Lambertian) brdf can be written as:

$$f(\omega_i, \omega_o) = \frac{\boldsymbol{c}_{diff}}{\pi}$$

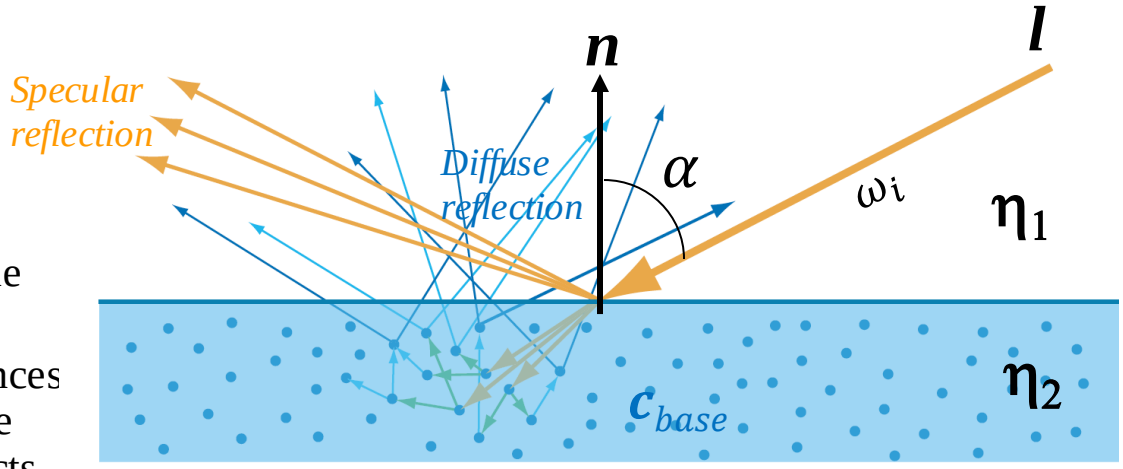
=>

diffuse color: $\boldsymbol{L}_o(\omega_o) = \boldsymbol{c}_{diff}\,\boldsymbol{c}_{light}(n \cdot \omega_i)$

Materials – the basic model

Material-light model:

- Some amount of incoming light from direction ω_i :
  - reflects to various outgoing directions (yellow), causing the ***specular*** reflection.
  - refracts into the material, bounces around, gets color tinted by the atoms or molecules, and refracts out as a fully ***diffuse*** reflection (blue). This coloring is modeled by the material's base color (albedo), $\boldsymbol{c}_{base}$.



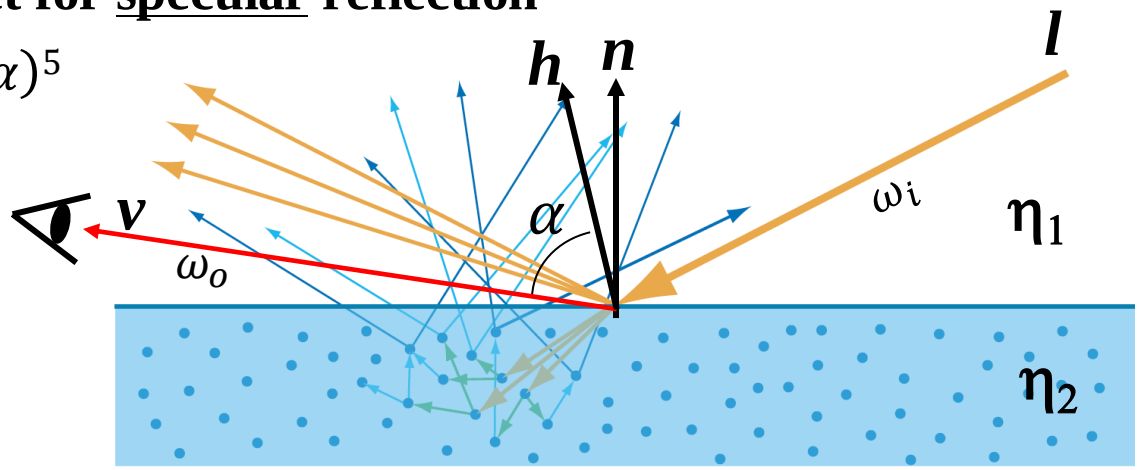
- The Fresnell equations describe how much of the incoming light that reflects or refracts. They depend on the relative refraction index $\eta = \eta_1/\eta_2$ (which is wavelength dependent) and the incoming angle, α, to the surface.

- The amount of light that reflects as **diffuse** is equal to the amount that **refracts** into the material (minus total absorption but then that is often baked into $\boldsymbol{c}_{base}$).
- The Fresnel equations accurately models this % of refraction, but we like to use a faster approximation called *Schlick's approximation* and that models the **specular** reflection. Let's call it F.
- Hence, amount of diffuse reflection = % of refraction = (1-F).
- So, how do we model F?

Materials – Schlick's approximation

Approximates the Fresnel effect for <u>specular</u> reflection

$$\boldsymbol{F}(\alpha) \approx \boldsymbol{F}_0 + (1 - \boldsymbol{F}_0)(1 - \cos\alpha)^5$$



where α is half the angle between incoming direction, $\boldsymbol{l}$, and outgoing direction, $\boldsymbol{v}$ (often called ω_i and ω_o).

Cos α can be computed as:

- $\cos\alpha = \boldsymbol{h} \cdot \boldsymbol{v}$ (see image),

where half vector $\boldsymbol{h} = \frac{\boldsymbol{l}+\boldsymbol{v}}{\|\boldsymbol{l}+\boldsymbol{v}\|}$

i.e., vector half between $\boldsymbol{l}$ and $\boldsymbol{v}$.

(Rationale: it is only those microfacets with normal $\boldsymbol{h}$, that will reflect from $\boldsymbol{l}$ along $\boldsymbol{v}$ – see later slide.)
Hence, Schlick's approximation can be written as:

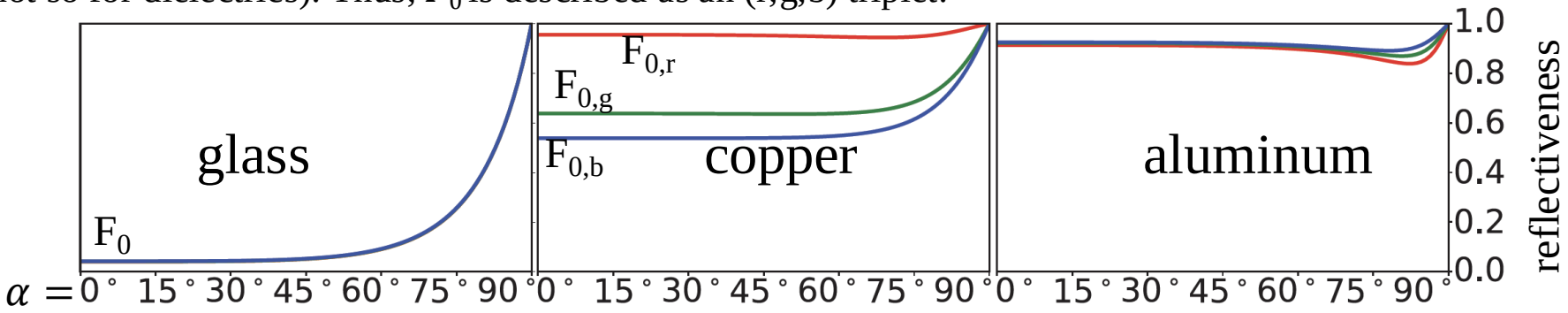
$$\boldsymbol{F}(\boldsymbol{h},\boldsymbol{v}) \approx \boldsymbol{F}_0 + (1 - \boldsymbol{F}_0)\big(1 - (\boldsymbol{h} \cdot \boldsymbol{v})\big)^5$$

where $F_0 = \left(\frac{\eta_1-\eta_2}{\eta_1+\eta_2}\right)^2$, i.e., the reflectance at $\alpha = 0$ degrees

Vice versa: $\eta = \frac{1+\sqrt{F_0}}{1-\sqrt{F_0}}$

Also clamp $\cos\alpha$, i.e., $\boldsymbol{h} \cdot \boldsymbol{v}$ to [0,1].

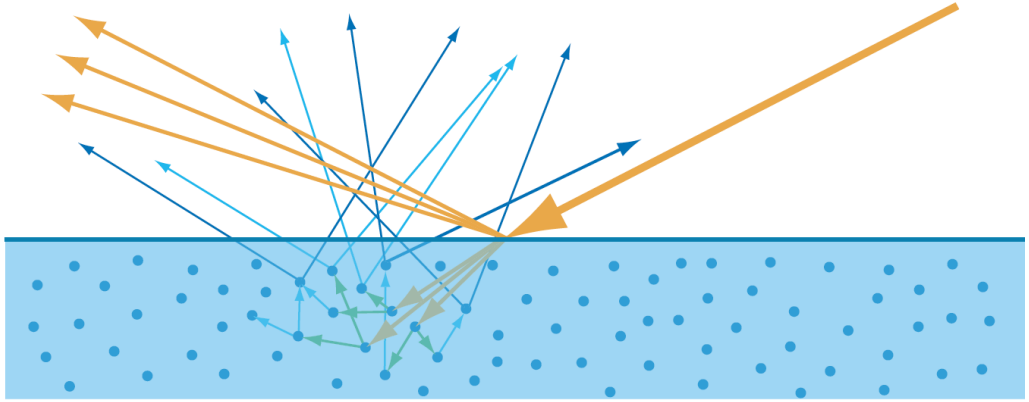
The refraction indices are wavelength dependent, so F_0 is also wavelength dependent (highly for metals, not so for dielectrics). Thus, $\boldsymbol{F}_0$ is described as an (r,g,b) triplet.



Materials – dielectrics vs metals

Materials:

- **Dielectrics:**
  - Ex: glass, plastic, water, wood, stone, concrete, hair, leather, skin,
  - The **glossy** reflection has the light's color.
  - The **diffuse** reflection is colored by the material
- **Metals:** has **only** reflection, no refraction (so no diffuse component)



Dielectric	Linear	Texture	Color	Notes
Water	0.02	39		
Living tissue	0.02–0.04	39–56		Watery tissues are toward the lower bound, dry ones are higher
Skin	0.028	47		
Eyes	0.025	44		Dry cornea (tears have a similar value to water)
Hair	0.046	61		

Metal	Linear	Texture	Color
Titanium	0.542,0.497,0.449	194,187,179	
Chromium	0.549,0.556,0.554	196,197,196	
Iron	0.562,0.565,0.578	198,198,200	
Nickel	0.660,0.609,0.526	212,205,192	
Platinum	0.673,0.637,0.585	214,209,201	
Copper	0.955,0.638,0.538	250,209,194	
Palladium	0.733,0.697,0.652	222,217,211	
Mercury	0.781,0.780,0.778	229,228,228	
Brass (C260)	0.910,0.778,0.423	245,228,174	
Zinc	0.664,0.824,0.850	213,234,237	
Gold	1.000,0.782,0.344	255,229,158	
Aluminum	0.913,0.922,0.924	245,246,246	
Silver	0.972,0.960,0.915	252,250,245	

F_0 values p:322-323.

<u>Example of the typical PBS material parameters:</u>

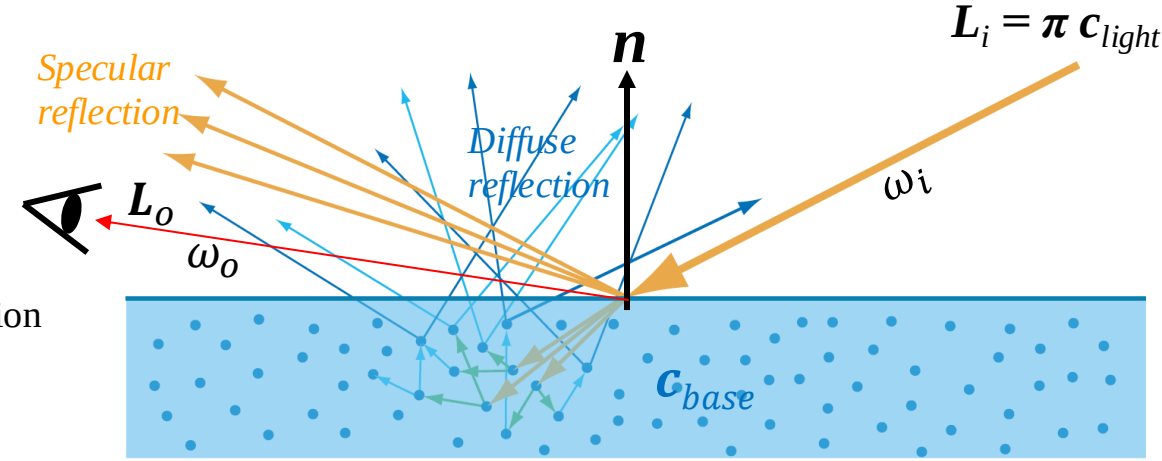
- Metalness (vs dielectric). In percent.
  - Allows layered mtrls, e.g., metal with lacquer layer
- Roughness (in [0,1]) or Shininess (in [0,∞])
- Fresnel $\mathbf{F}_0$. p:322-323.
- Base_color: $\boldsymbol{c}_{base}$

Basic Physically-Based Shading model

Putting it together…

Parameters:

- ***Metalness:*** vs dielectric, in %.
- ***Fresnel*** $\mathbf{F}_0$: base reflectance, in %.
- ***Roughness***: in [0,1]
- ***Base color***: c_{base} i.e., color tint by absorption



Formulas:

Radiance from point light: $\boldsymbol{L_i = \pi\, c_{light}}$ *1/r^2

Radiance from directional light: $\boldsymbol{L_i = \pi\, c_{light}}$

Fresnell effect: $\boldsymbol{F(h,v) \approx F_0 + (1 - F_0)\big(1 - (h \cdot v)\big)^5}$

diffuse_brdf = $\frac{c_{base}}{\pi}$

metal_brdf = $\frac{G(\omega_i,\omega_o)D(\omega_h)F(\omega_h,\omega_o)}{4|n\cdot\omega_o||n\cdot\omega_i|}$ * c_{base} — metal reflection is colored by material

dielectric_brdf = $\frac{G(\omega_i,\omega_o)D(\omega_h)F(\omega_h,\omega_o)}{4|n\cdot\omega_o||n\cdot\omega_i|}$ * **vec3(1) + (1-F) diffuse_brdf** — but not so for dielectrics

Specular reflection *Diffuse reflection*

Geometry factor, Distribution factor, Fresnel

$$\frac{G(\omega_i,\omega_o)D(\omega_h)F(\omega_h,\omega_o)}{4|n\cdot\omega_o||n\cdot\omega_i|}$$

Specular reflection weight for glossy (semi-rough) surface w.r.t incoming light direction, ω_i, outgoing direction, ω_o, roughness, and surface normal n.
See next slides…

tot_brdf = metalness * **metal_brdf** + (1 - metalness) * **dielectric_brdf**

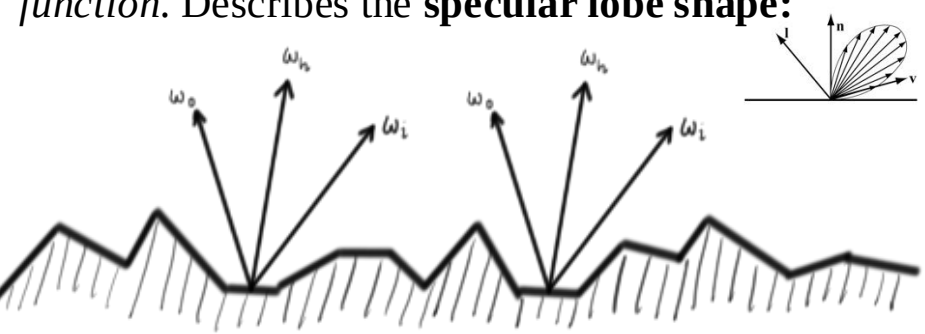
TOTAL: $\boldsymbol{L_o(\omega_o) = \sum_{i=1}^{\#lights}(\text{tot\_brdf})\,(L_i)\,(n \cdot \omega_i)}$

For a good and deep explanation, see for instance "Advanced Computer Graphics Materials" by Matthias Teschner:
https://cg.informatik.uni-freiburg.de/course_notes/graphics2_02_materials.pdf

Basic Physically-Based Shading model

Specular reflection weight for semi-rough) surfaces: $\frac{G(\omega_i,\omega_o)D(\omega_h)F(\omega_h,\omega_o)}{4|n\cdot\omega_o||n\cdot\omega_i|}$

D() is *Distribution* function, a.k.a. *microfacet distribution function*, a.k.a. *normal distribution function*. Describes the **specular lobe shape:**



Only the microfacets whose normal is ω_h will reflect in direction ω_o. $D(\omega_h)$ gives us the density of such facets.

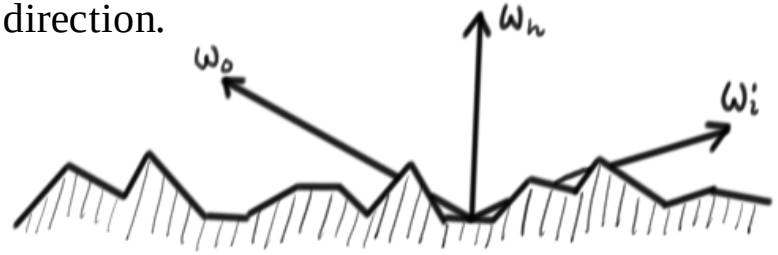
E.g., the *normalized Blinn-Phong* function used in lab 4:

$\omega_h = \texttt{normalize(}\ \omega_i + \omega_o\ \texttt{)}$

$s = \texttt{material\_shininess}$

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

G() is Geometry function. Describes **specular self-shadowing,** i.e., fraction of microfacets visible (non-occluded) from both incoming and outgoing direction.



When ω_o or ω_i are at grazing angles, radiance might be blocked by other microfacets. This is what $G(\omega_i, \omega_o)$ models.

There are various masking-shadowing functions, e.g.:

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

Clamp to [0,1] and avoid denominators ≈ 0

Is explained by Erik… see his online videos 1 2 3 4.

The GGX Specular Reflection Formula

Very popular model for D and G.

- GGX stands for Trowbridge-Reitz GGX, where GGX refers to the generalized gaussian distribution used for microfacet normal distributions.
- GGX better simulates how light reflects off rougher surfaces, especially at oblique angles, compared to older models like the Beckmann distribution. The GGX distribution produces sharp reflections for smoother surfaces and blurry reflections for rougher surfaces. Found in Unreal Engine, Unity, and in the *Disney BRDF* - a cornerstone of modern PBR workflows.

Normal Distribution Function D() for GGX

$$D(\dots) = \frac{\alpha^2}{\pi\big((\boldsymbol{n}\cdot\boldsymbol{h})^2(\alpha^2-1)+1\big)^2}$$

α = roughness.

Geometry function G() – Smith GGX variant:

$$G(\dots) = G_1(\boldsymbol{n},\boldsymbol{v})G_1(\boldsymbol{n},\boldsymbol{l})$$

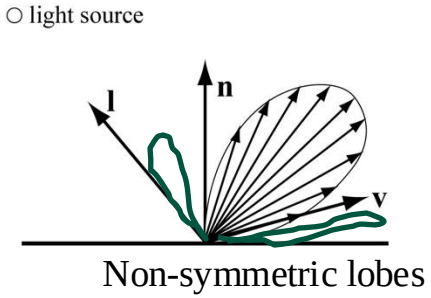
$$G_1(\boldsymbol{n},\boldsymbol{\omega}) = \frac{2(\boldsymbol{n}\cdot\boldsymbol{\omega})}{(\boldsymbol{n}\cdot\boldsymbol{\omega}) + \sqrt{\alpha^2+(1-\alpha^2)(\boldsymbol{n}\cdot\boldsymbol{\omega})^2}}$$

Final Expression

Putting it all together, the GGX-based specular term is:

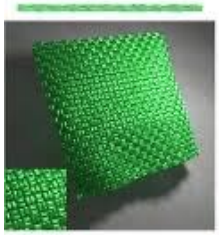
$$I_{\text{specular}} = \frac{\frac{\alpha^2}{\pi\left((N\cdot H)^2(\alpha^2-1)+1\right)^2}\cdot\left(F_0+(1-F_0)(1-H\cdot V)^5\right)\cdot\frac{4\cdot(N\cdot V)(N\cdot L)}{\left((N\cdot V)+\sqrt{\alpha^2+(1-\alpha^2)(N\cdot V)^2}\right)\left((N\cdot L)+\sqrt{\alpha^2+(1-\alpha^2)(N\cdot L)^2}\right)}}{4\cdot(N\cdot V)\cdot(N\cdot L)}$$

Extra… (bonus)



Non-symmetric lobes

- Anisotropic Normal Distribution Functions – update D() in the microfacet model - see p343
- Multibounce surface reflections – p:346
- Subsurface Scattering: p:347 – modify the Diffuse brdf and the Fresnell factor.
- Cloth brdf:s – p:356

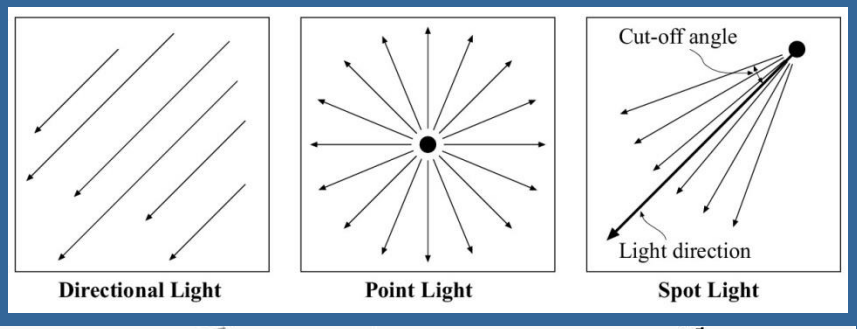


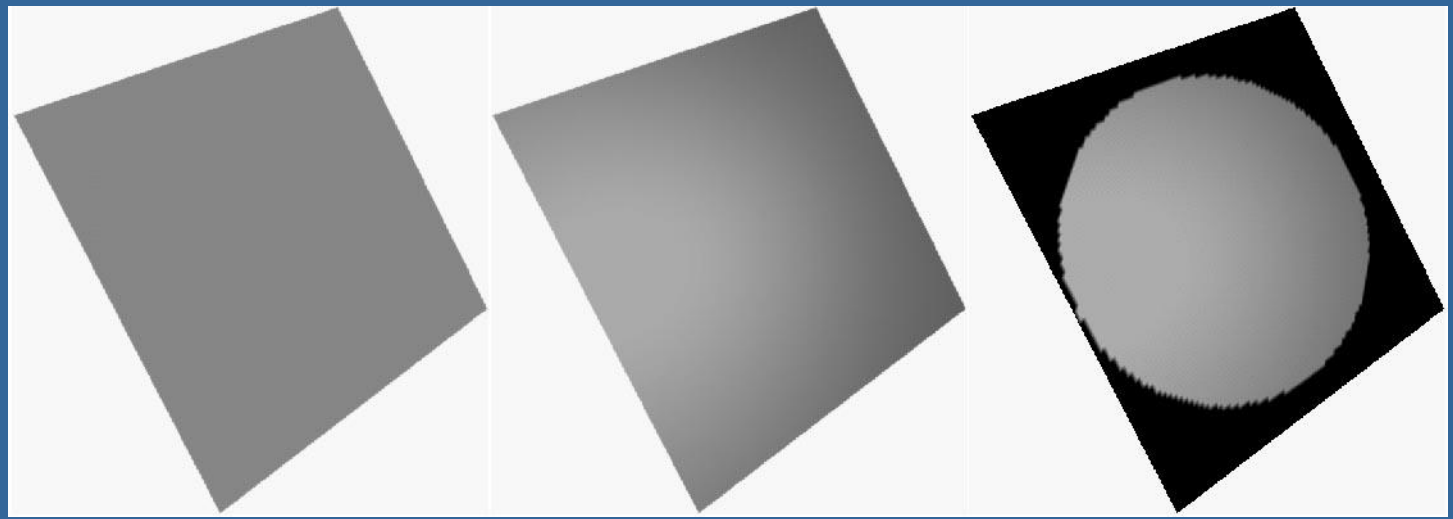
- Light falloff: page 111, Unreal, Frostbite + CryEngine
  - Distance falloff function / windowing function: Just Cause 2
- Lambertian brdf = diffuse color = albedo, and terms are often used interchangeably.
- Some light source types: point lights, area lights,
  - Incoming light from the surrounding can also be captured by environment maps.

Environment maps (reflection maps)

Additions to the lighting equation

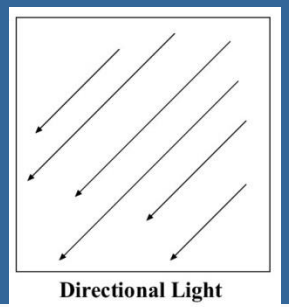
- Accounting for distance: $1/(a+bt+ct^2)$
- Several lights: just sum their respective contributions
- Different light types:

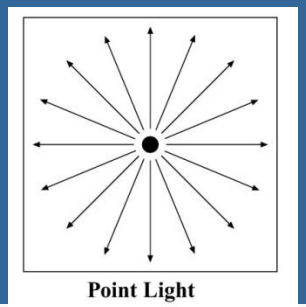


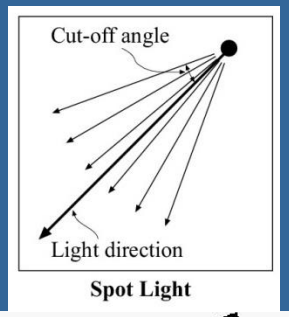


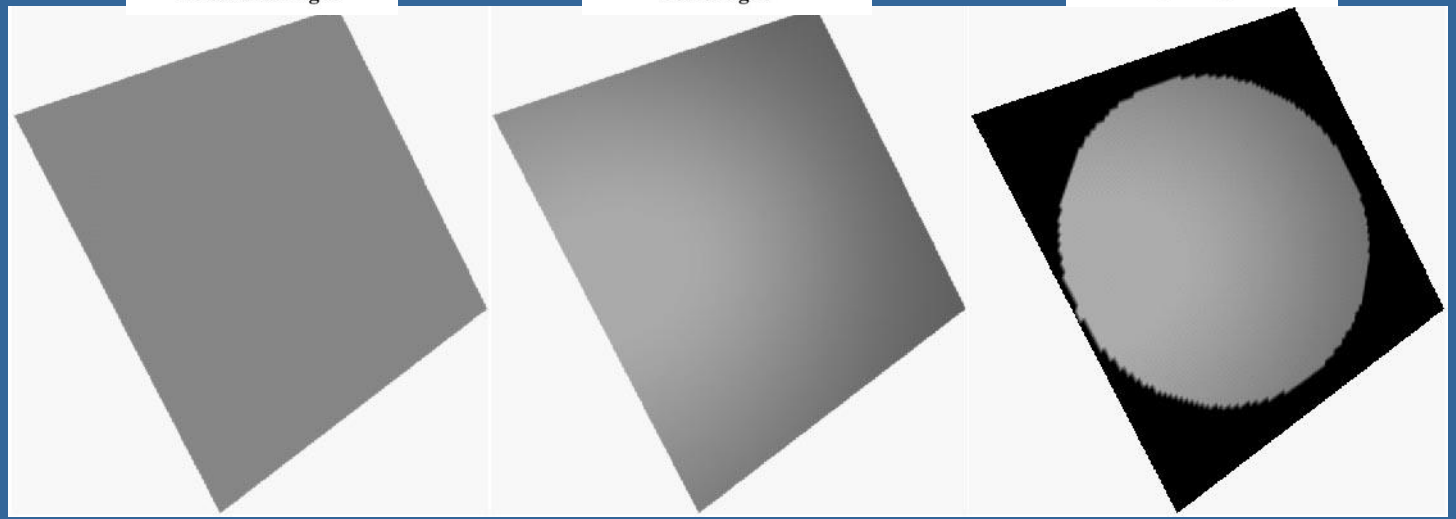
Additions to the lighting equation

- Accounting for distance: $1/(a+bt+ct^2)$
- For several lights: just sum their respective contributions
- Some different light types:





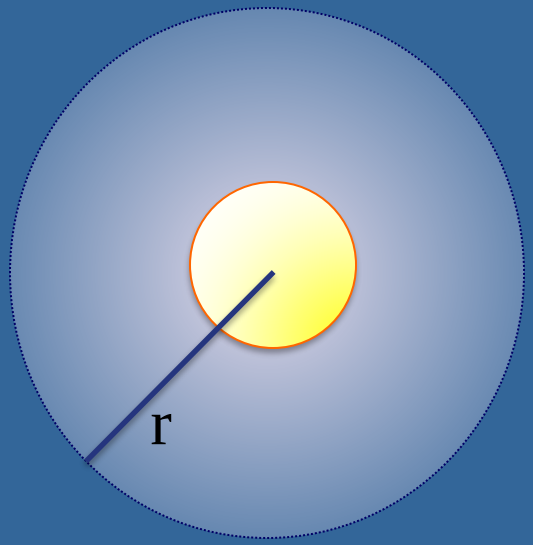




Clarification on accounting for distance

- Energy is emitted at equal proportions in all directions from a spherical radiator. Due to energy conservation, the intensity is proportional to the **spherical area** at distance r from the light center.
- $A = 4\pi r^2$
- Thus, the intensity scales $\sim 1/r^2$

- For efficiency, we often cap or limit how far the light source will affect the environment.
  - Hence, we often want to fade its intensity to zero at some finite distance r.

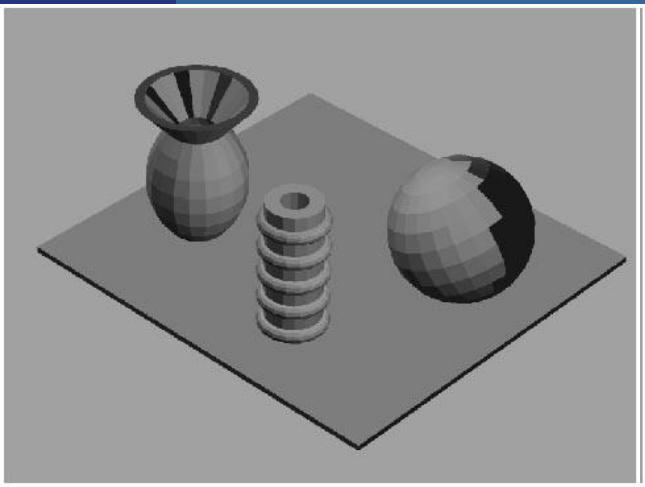
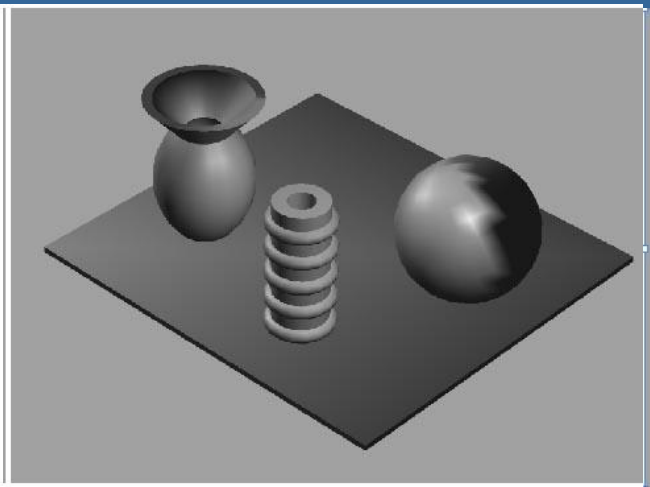
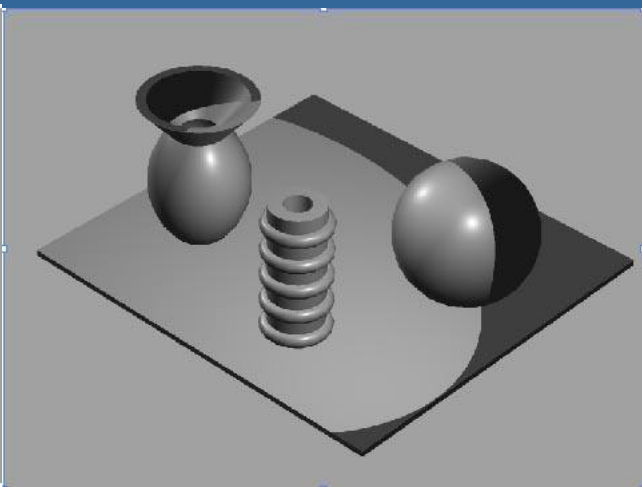


Shading

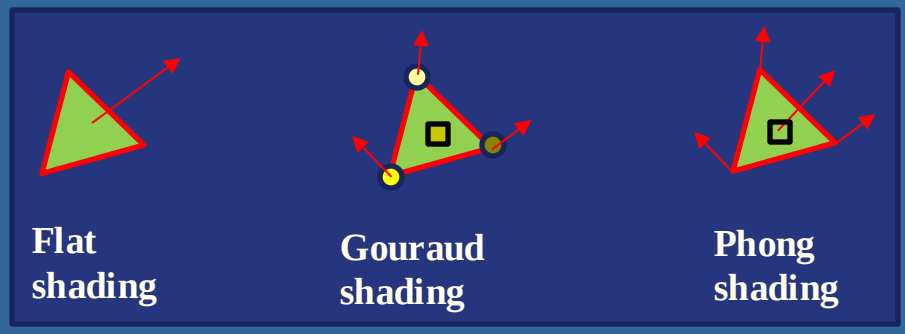
- Shading: compute the fragment's final color contribution to the pixel.
- Three types of shading

  regarding how often it is computed per triangle:

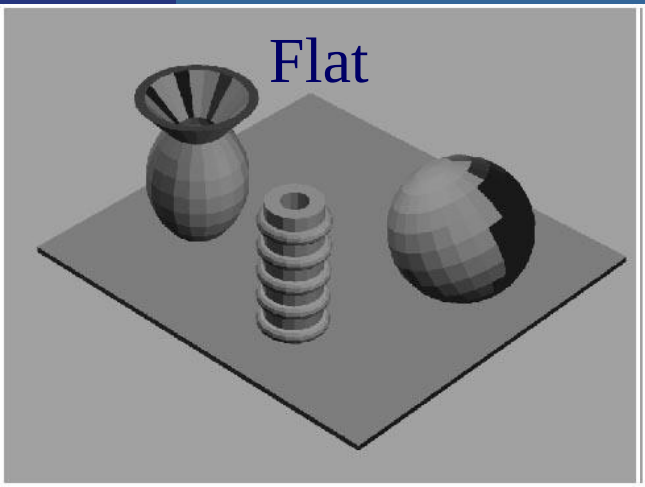
  - Flat shading: once per triangle
  - Goraud shading: once per vertex
  - Phong shading: once per pixel (standard today)

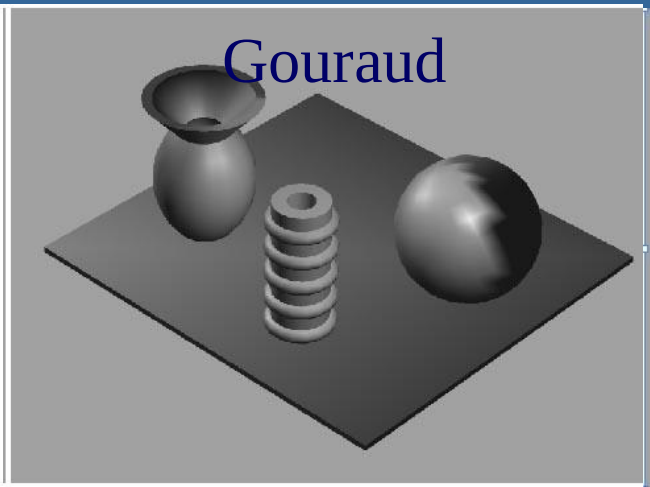




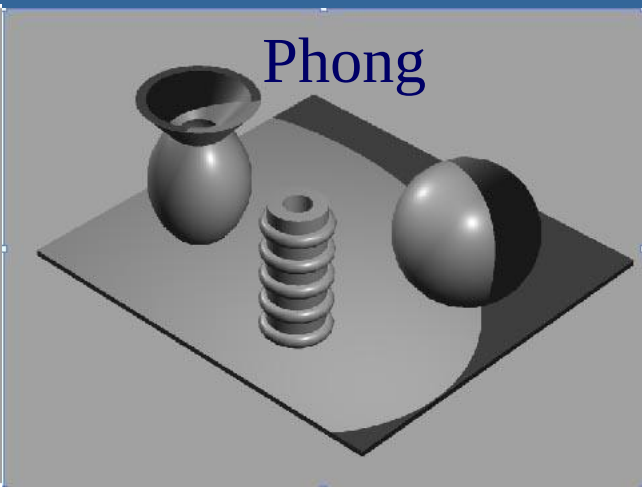
Shading



- Flat, Goraud, and Phong shading:
  - Flat shading: one normal per triangle. Lighting computed once **per triangle**.
  - Gouraud shading: the lighting is computed **per vertex** and for each pixel, the **color is interpolated** from the colors at the vertices.
  - Phong Shading: the lighting is computed **per pixel**. The **normal is interpolated** per pixel from the normals defined at the vertices, and full lighting is computed per pixel using this normal. This is of course more expensive but looks better.







Transparency and alpha

- Transparency
  - Very simple in real-time contexts
- The tool: alpha blending (mix two colors)
- Alpha (α) is the forth color component (r,g,b,α)
  - e.g., of the material for a triangle
  - Represents the opacity
  - 1.0 is totally opaque
  - 0.0 is totally transparent
- The over operator:

$$\mathbf{c}_o = \alpha \mathbf{c}_s + (1-\alpha)\mathbf{c}_d$$

Color already in the frame buffer at the corresponding position

Rendered object

Transparency

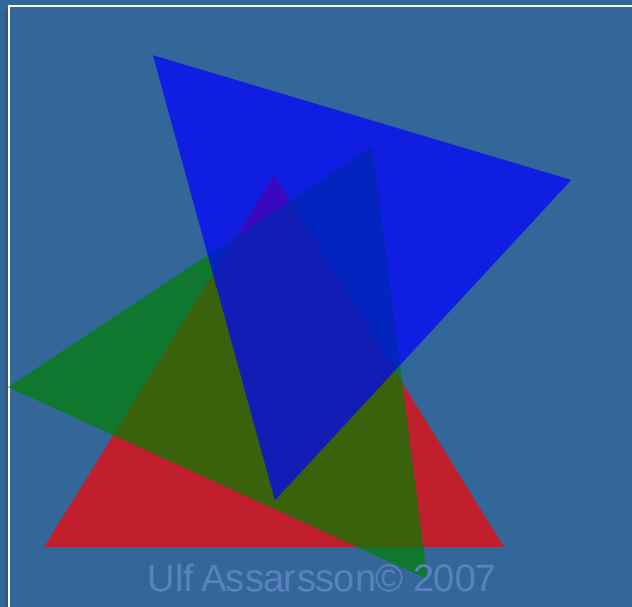
$$\mathbf{c}_o = \alpha \mathbf{c}_s + (1-\alpha)\mathbf{c}_d$$

Rendered fragment

Background

- Need to sort the transparent objects
  - Render back to front (blending is order dep.)
    - See next slide…
- Exist different blending modes
- Can store RGBα in textures as well

So the texels with α=0.0 do not not hide the objects behind

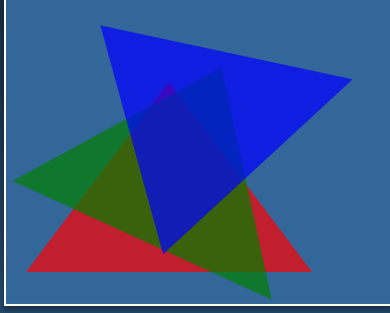


Transparency

- Need to sort the transparent objects
  - **First, render all non-transparent triangles as usual.**
  - **Then, sort all transparent triangles and render them back-to-front with blending enabled.**
    - **The reason for sorting is that the blending operation (i.e., over operator) is order dependent.**

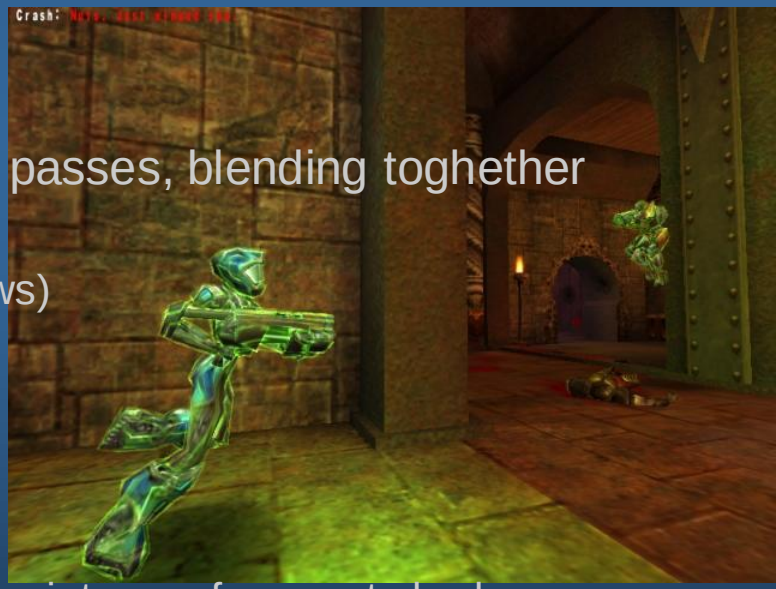
If we have high frame-to-frame coherency regarding the objects to be sorted per frame, then Bubble-sort (or Insertion sort) are really good! (superior to Quicksort).
Because, they have expected runtime of resorting already almost sorted input in O(n) instead of O(n log n), where n is number of elements.

Blending



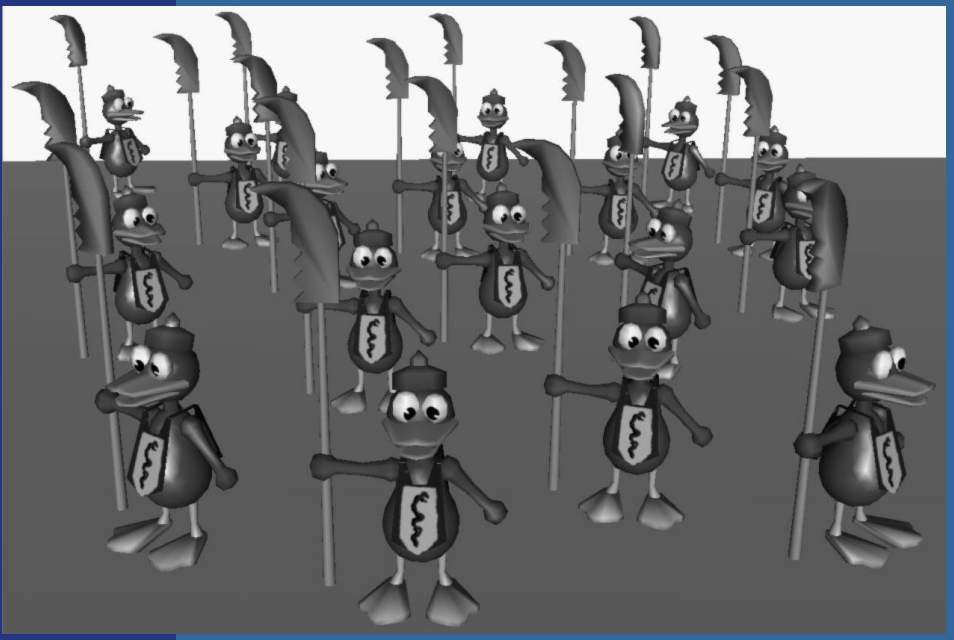
- Used for
  - Transparency
    - **glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)**
  - Effects (shadows, reflections)
  - (Complex materials)
    - Quake3 used up to 10 rendering passes, blending toghether contributions such as:
      - Diffuse lighting (for hard shadows)
      - Bump maps
      - Base texture
      - Specular and emissive lighting
      - Volumetric/atmospheric effects
    - Today, we can merge all those passes into one fragment shader.
  - Enable with **glEnable(GL_BLEND)**

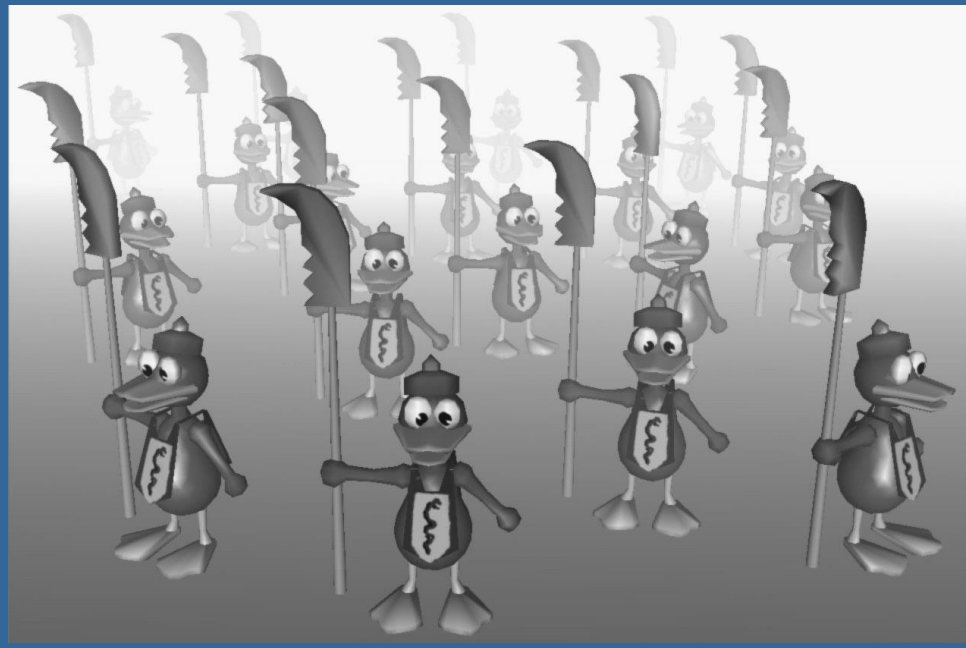
$$\mathbf{c}_o = \alpha \mathbf{c}_s + (1-\alpha)\mathbf{c}_d$$

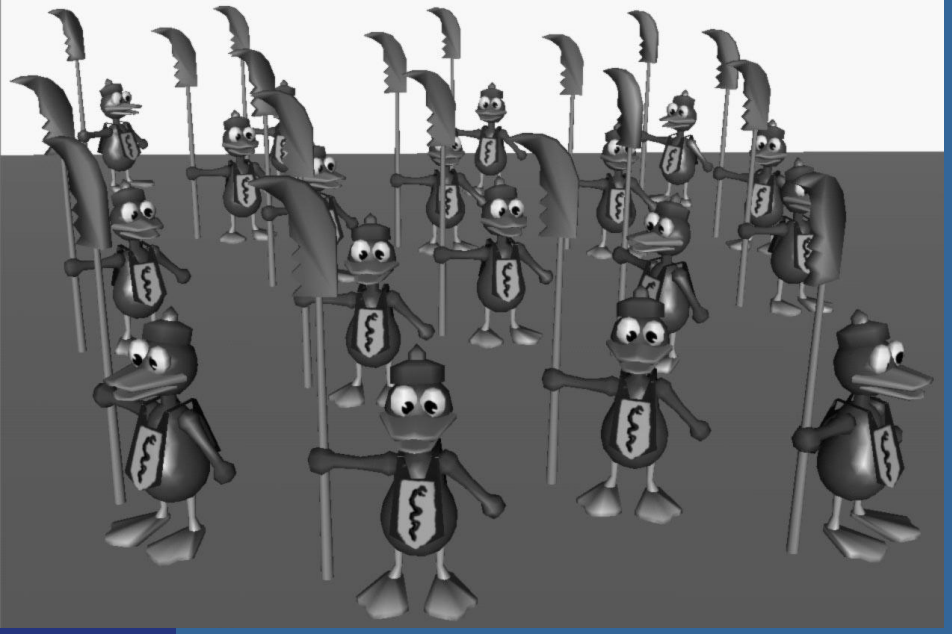


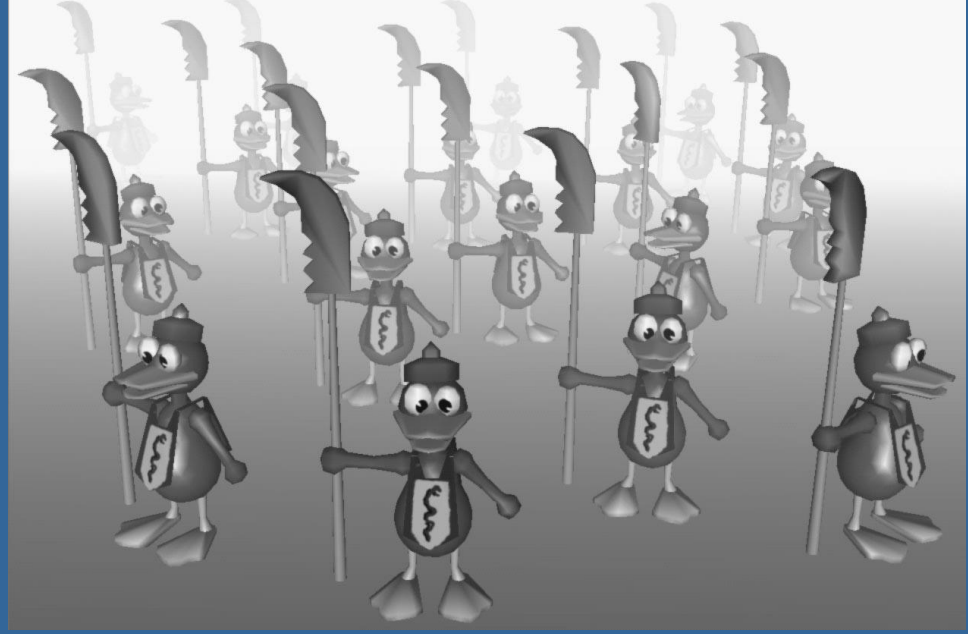
Fog

- Simple atmospheric effect
  - A little better realism
  - Help in determining distances









- Color of fog: $\mathbf{c}_f$ color of surface: $\mathbf{c}_s$

$$\mathbf{c}_p = f\mathbf{c}_s + (1 - f)\mathbf{c}_f \qquad f \in [0,1]$$

- How to compute f ?
- E.g.: linear, exponential
- Linear:

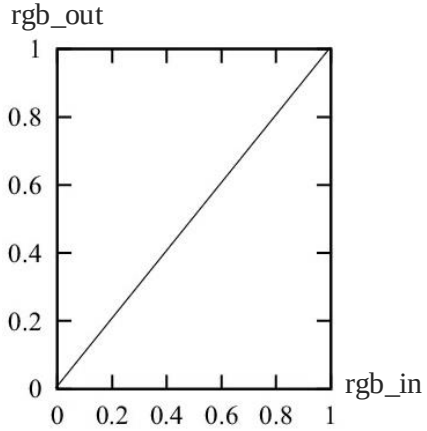
$$f = \frac{z_{end} - z_p}{z_{end} - z_{start}}$$

Program it yourself in the fragment shader.

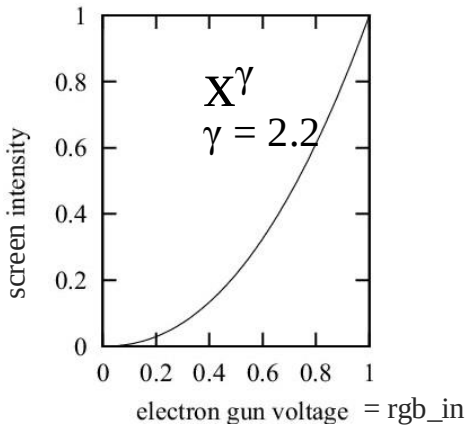
(Old OpenGL – just set OpenGL parameters and turn it on)

Fog in up-direction

Gamma correction



We compute rgb color intensities in linear space from [0,1]

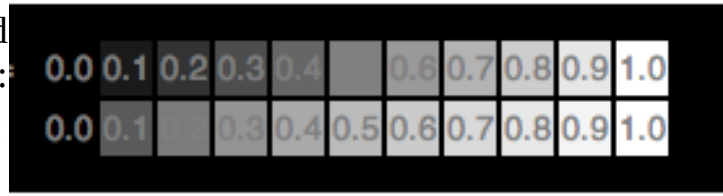


However, CRT-monitor output is exponential. Gives more precision for darker regions. Very Good! But we want linear output. Else, our images will be too dark.

Exponential distribution better for humans. Our eyes have logarithmic sensitivity and monitors have limited brightness

x^γ is perceived as linear:

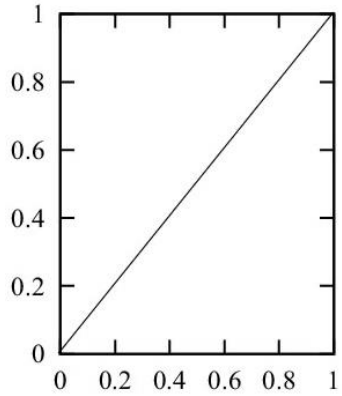
Actually linear:



Intensities: x^γ vs linear

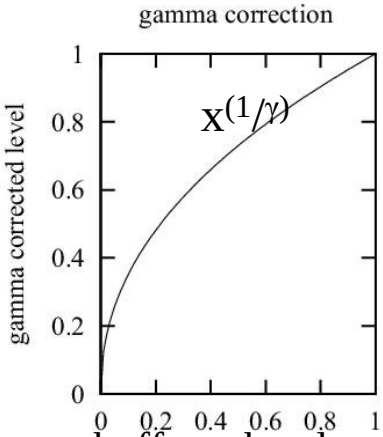
Textures: often stored in gamma space for better distributed precision.

So, store color intensities with more precision for darker colors: i.e., convert color to $x^{(1/\gamma)}$ before storing in 8- bits in the frame buffer. Conversion to $x^{(1/\gamma)}$ is called gamma correction.

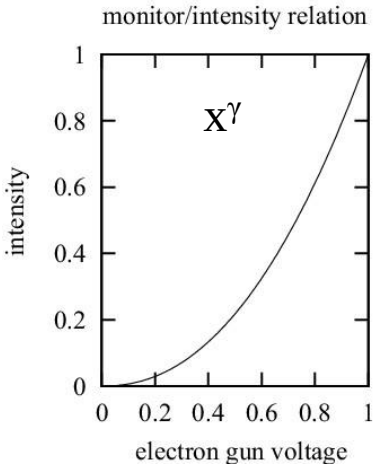


Shader rgb colors

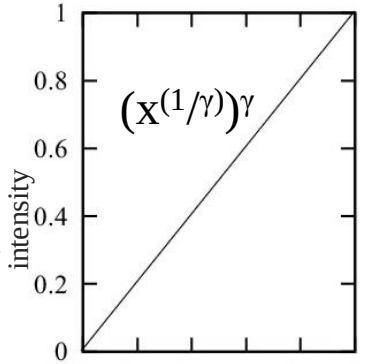
$x^{(1/\gamma)}$



Frame buffer rgb colors.
"Dark pixels are made brighter"

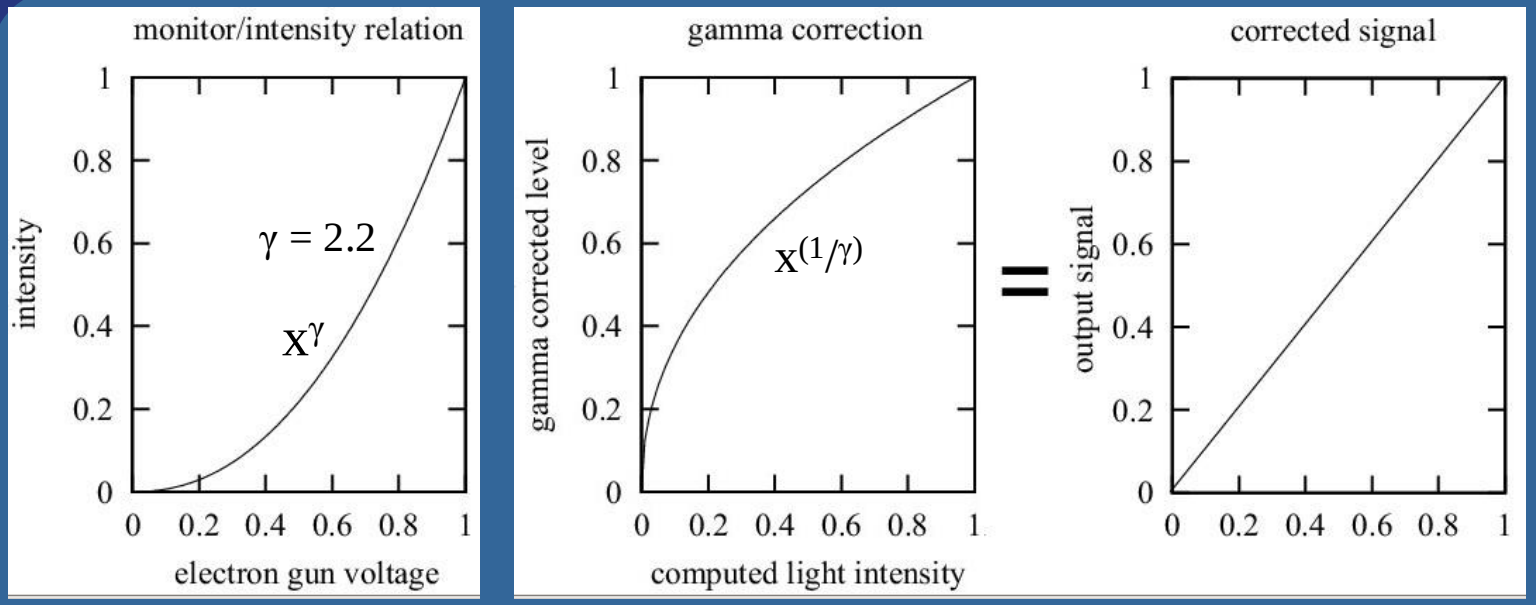


Displayed by CRT



Linear output again, but redistributed precision.

Gamma correction



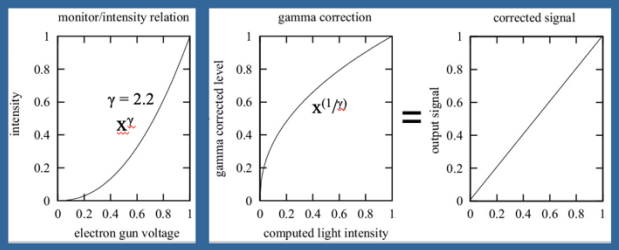
- If input to gun is 0.5, then you don't get 0.5 as output in intensity
- Instead, gamma correct that signal: gives linear relationship

Gamma correction

$$I = a(V + \varepsilon)^{\gamma}$$

- *I*=intensity on screen
- *V*=input voltage (electron gun)
- *a*, ε, *and* γ *are* constants for each system
- Common gamma values: 2.2-2.6
- Assuming ε=0, gamma correction is:

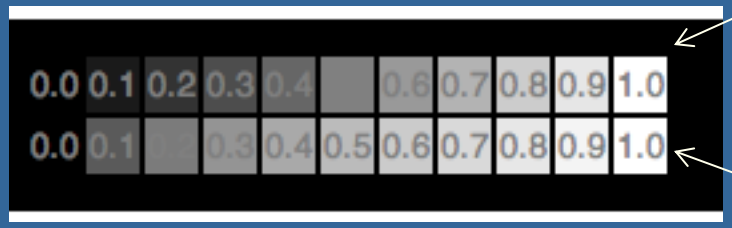
$$c = c_i^{(1/\gamma)}$$



Gamma correction

- Reasons for wanting gamma correction (standard is 2.2):

1. Screen has non-linear color intensity
   - We want linear output for correctness.
   - But, today, screens can be made with linear output, so non-linearity is more for backwards compatibility and better 8-bit color precision.
2. Also gives more efficient color space (when compressing intensity from 32-bit floats to 8-bits). Thus, often desired when storing images (color buffer, textures) in 8 bits rgb.



Gamma of 2.2. Better distribution for humans. Perceived as linear.

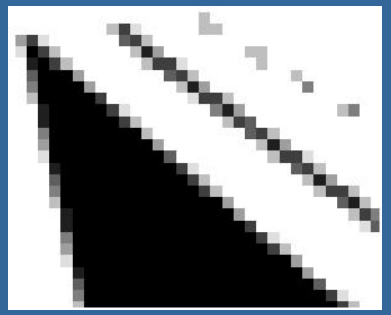
Truly linear intensity increase.

A linear intensity output (bottom) has a large jump in perceived brightness between the intensity values 0.0 and 0.1, while the steps at the higher end of the scale are hardly perceptible.
A nonlinearly-increasing intensity (upper), will show much more even steps in perceived brightness.

Important on Gamma correction

- Give two reasons for gamma correction:
  - screen output is non-linear so we need gamma to counter that.
  - Textures/images and color buffer can be stored with better precision (for human eye) for low-intensity regions.
  - Antialiasing:

Aliasing



Antialiasing

Gamma correction is required for correct half-tones.

Important on Gamma correction

- Give two reasons for gamma correction:
  - screen output is non-linear so we need gamma to counter that.
  - Textures/images can be stored with better precision (for human eye) for low-intensity regions.

Lecture 3.1 Shading - summary

"How to shade your fragments"

- The fragment color is the surface's *radiance* at that point.
- The *radiance*, L_{Out}, can be split into:

Ambient contribution
Assume homogeneous background light everywhere

$L_o = \mathbf{m}_{r,g,b}\, \mathbf{l}_{rgb}$,

where $\mathbf{m}_{r,g,b}$ is surface color and $\mathbf{l}_{rgb}$ is background-light color.

Emission contribution
Self-glowing material

L_o += $\mathbf{m}_{r,g,b}$, where $\mathbf{m}_{r,g,b}$ is the glow color or radiance.

Diffuse contribution
For material part that is fully rough (Lambertian), scale incoming light with angle to surface normal:

L_o += $\mathbf{m}_{r,g,b}\, \mathbf{l}_{rgb}\, (\boldsymbol{n} \cdot \boldsymbol{l})$,

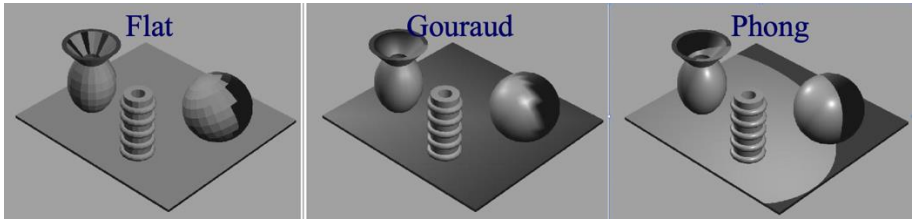
where $\mathbf{l}_{rgb}$ is light-source color and $\boldsymbol{l}$ is direction to the light.

Specular contribution
For material part that is glossy (=semi-specular): Scale light-source reflection with view angle from light's main reflection direction:
Phong: L_o += $\mathbf{m}_{r,g,b}\, \mathbf{l}_{rgb}\, (\boldsymbol{r} \cdot \boldsymbol{v})^{\,shi}$, where $\boldsymbol{r}$ is refl. direction, $\boldsymbol{v}$ = view dir.
Blinn: L_o += $\mathbf{m}_{r,g,b}\, \mathbf{l}_{rgb}\, (\boldsymbol{h} \cdot \boldsymbol{n})^{4\,shi}$, where $\boldsymbol{h}$ is half vector of $\boldsymbol{l}$ and $\boldsymbol{v}$.

PBS (Physically-based Shading):

$$L_o \mathrel{+}= \mathbf{m}_{r,g,b}\, \mathbf{l}_{rgb} \frac{G(\boldsymbol{l},\boldsymbol{v})D(\boldsymbol{h})F(\boldsymbol{h},\boldsymbol{v})}{4|\boldsymbol{n}\cdot\boldsymbol{v}||\boldsymbol{n}\cdot\boldsymbol{l}|}\,(\boldsymbol{n}\cdot\boldsymbol{l})$$

Either way, $\mathbf{m}_{r,g,b}$ is white (1,1,1) for dielectrics and mtrl col for metals.



- Flat, Gouraud, and Phong shading

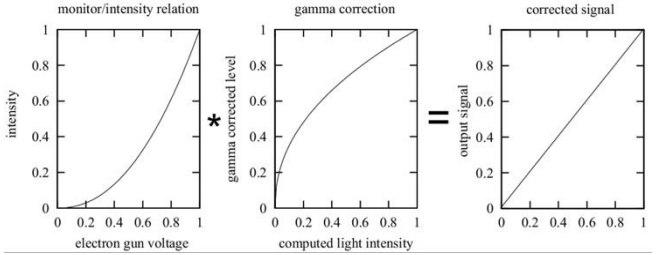
- Transparency

  Render transparent triangles in back-to-front order, with blending.

  $\mathbf{c}_o = \alpha \mathbf{c}_s + (1-\alpha)\mathbf{c}_d$, where $\mathbf{c}_s$ is fragment color, $\mathbf{c}_d$ is pixel color in frame buffer.

  glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)

- Gamma correction:



What is important

- Amb-, diff-, spec-, emission model + formulas
  - But not for PBS. E.g., for specular refl, then only:
    - Phong's + Blinn's highlight model:
      - Phong: $(\boldsymbol{r} \cdot \boldsymbol{v})^s$
      - *Blinn:* $(\boldsymbol{h} \cdot \boldsymbol{n})^s$ *, halfvector* **h** = (**l**+**v**)/|**l** + **v**|
- Flat-, Gouraud- and Phong shading
- Fog – just understand how we can use blending to do it
- Transparency:
  - Draw transparent triangles back-to-front.
  - Use blending with this over operator: $\mathbf{c}_o = \alpha \mathbf{c}_s + (1-\alpha)\mathbf{c}_d$
- Two reasons for wanting gamma correction