Rendering

Basic Shading

CS 415: Game Development

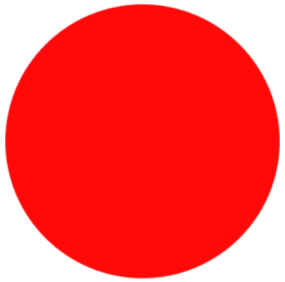
Professor Eric Shaffer

ILLINOIS

Shading

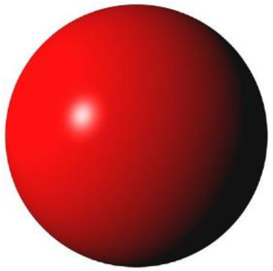
Shading refers to the process of determining the color for a pixel

Shading is one of the key elements of 3D photorealistic rendering…or any kind rendering



Flat shading

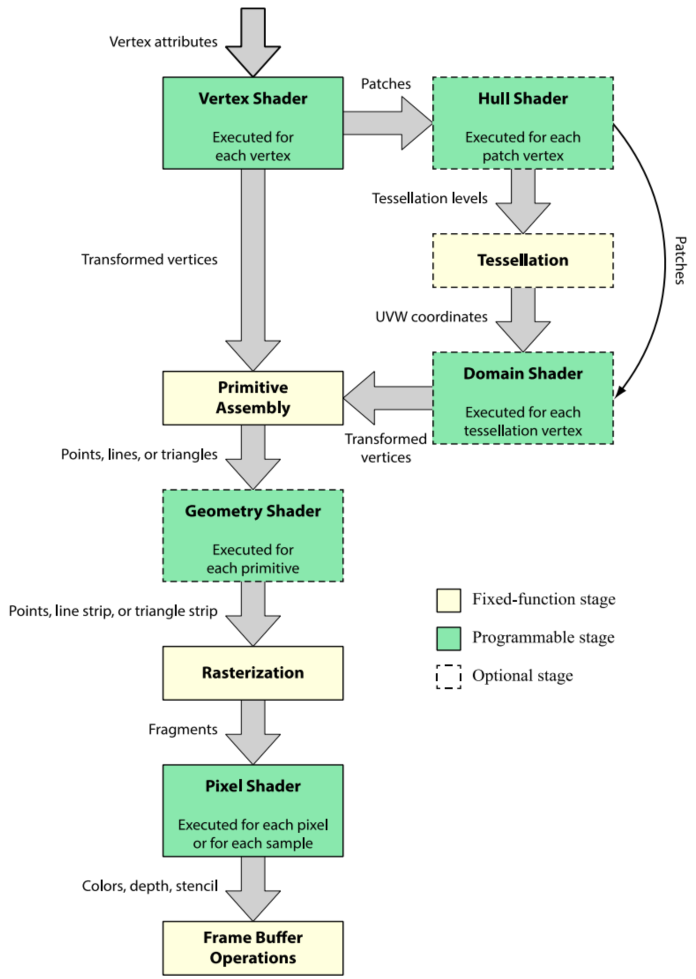
- Same color over entire surface of a sphere
- Looks 2D



Shading with Phong reflection model

- Varies color with geometry and light position
- Looks 3D

Programmable Shaders



Modern GPUs introduced the idea of shader programs

The word shader has come to mean any of a number of programmable GPU Units

Architecture on left is just one example

Not all engines make use of all these shader types

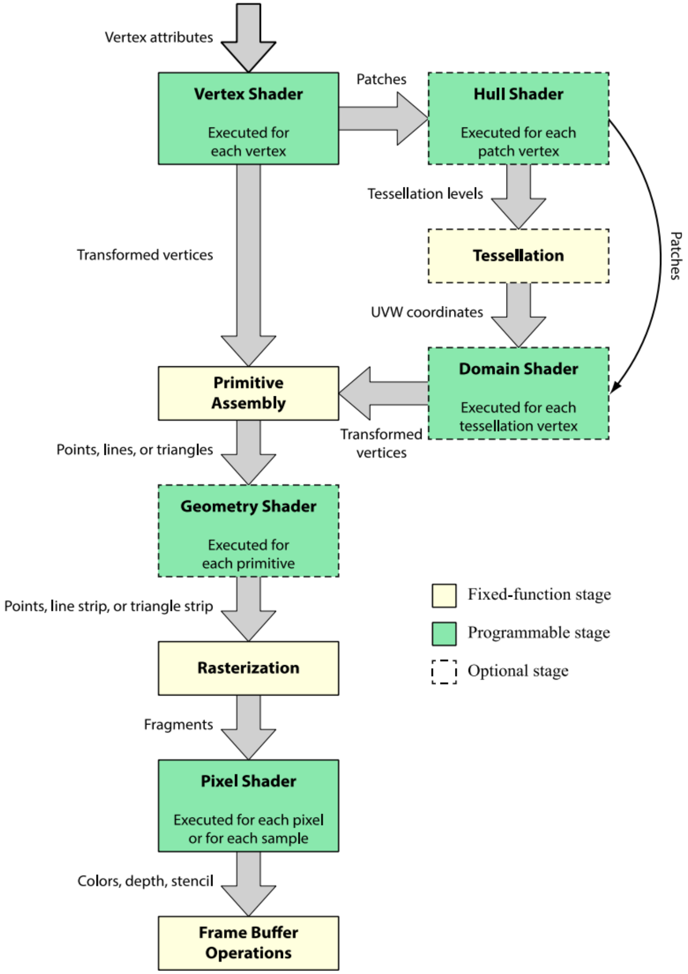
Not all rendering APIs support all these shader types

There are other shader types as well

- Tessellation Shader
- Mesh Shaders
- Ray Tracing Shaders
- Compute Shaders

Since shaders are programmable, many get used for purposes beyond their initial design

Programmable Shaders



Vertex Shader

- Each vertex in the scene geometry gets processed by a vertex shader program
- Usually just geometric transformations related to animation and projection, etc.
- Program is identical running across many cores of GPU vertices stream through

Pixel Shader (or Fragment Shader)

- Runs for each pixel produced by rasterization
- Usually just generates a color for that pixel ← ***this is actual shading***
- Program is identical running across many cores of GPU vertices stream through

The word **fragment** is used to describe a pixel that is produced during rasterization but isn't necessarily the final pixel value you will see on screen (maybe hidden by a surface in front of it, etc.).

Modeling Light Reflection: The Simplest Shader

Most of the time shading means modeling the reflection of light off a surface

What the simplest mathematical model?

Represent the amount of incoming light and the amount of that light reflected by the surface

- (r_i, g_i, b_i) is the color of the incoming light with each channel in the range [0,1]
- (r_k, g_k, b_k) is the reflectance of the material with each channel in the range [0,1]

0 indicates total absorption
1 indicates total reflectance

The shader would then compute a component-wise product:

$$(r_i, g_i, b_i) \times (r_k, g_k, b_k) = (r_i r_k, g_i g_k, b_i b_k)$$

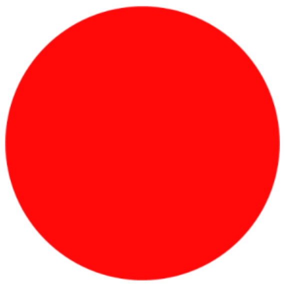
Modeling Light Reflection: The Simplest Shader

Represent the amount of incoming light and the amount of that light reflected by the surface

- (r_i, g_i, b_i) is the color of the incoming light with each channel in the range [0,1]
- (r_k, g_k, b_k) is the reflectance of the material with each channel in the range [0,1]

The shader would then compute a component-wise product:

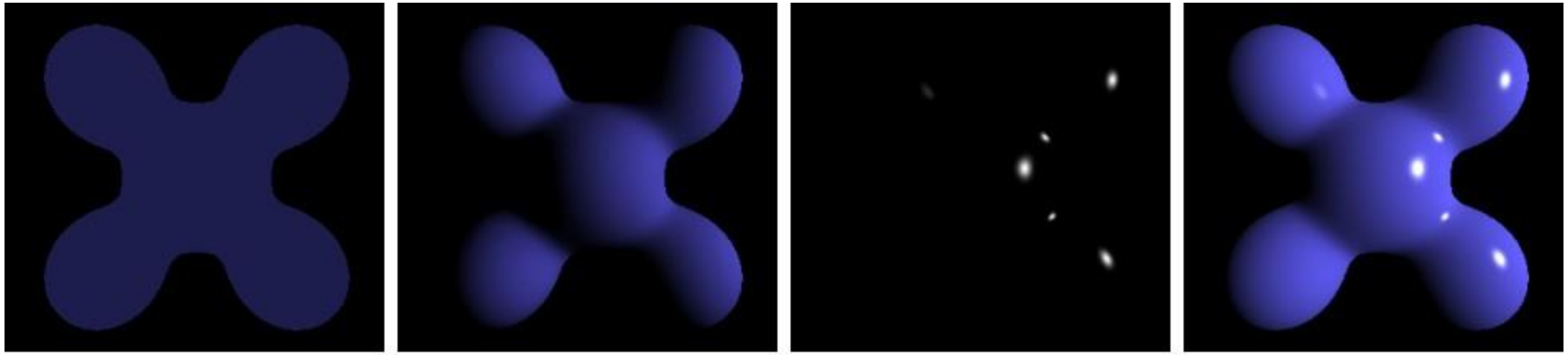
$$(r_i, g_i, b_i) \times (r_k, g_k, b_k) = (r_i r_k, g_i g_k, b_i b_k)$$



Flat shading

- Same color over entire surface of a sphere
- Looks 2D

The Second Simplest Shader: Phong Reflection



Ambient + Diffuse + Specular = Phong Reflection

$$I_{\text{p}} = k_{\text{a}} i_{\text{a}} + \sum_{m \,\in\, \text{lights}} \left(k_{\text{d}} (\hat{L}_m \cdot \hat{N}) i_{m,\text{d}} + k_{\text{s}} (\hat{R}_m \cdot \hat{V})^{\alpha} i_{m,\text{s}} \right)$$

Well…there's a lot of symbols but the ideas and math are simple

Bui Tuong Phong

- December 14, 1942 – July 1975
- Born in Hanoi
- Earned his PhD in 2 years at the University of Utah (1973)
  - Worked with Professor Ivan Sutherland
  - Dissertation work was the Phong reflectance model
  - Also produced model and realistic image of a VW bug

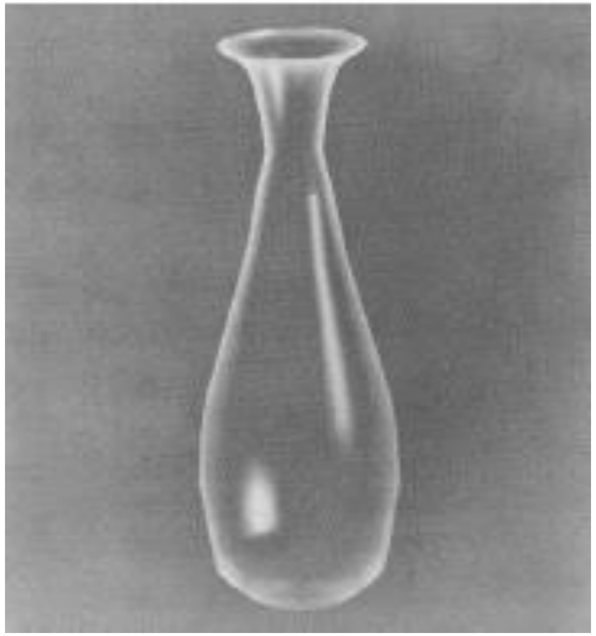
Graphics and Image Processing

W. Newman Editor

Illumination for Computer Generated Pictures

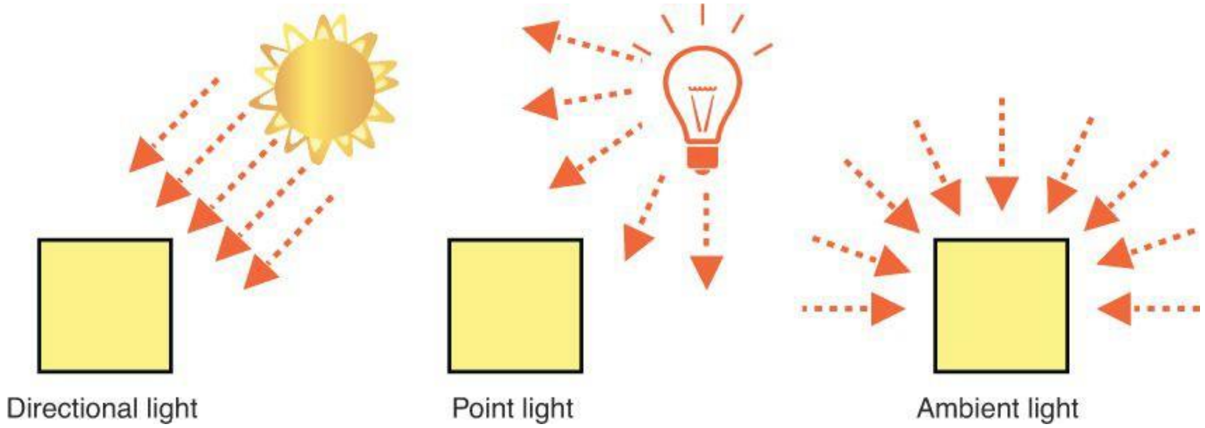
Bui Tuong Phong
University of Utah

Fig. 9. Improved shading, applied to the example of Figure 2.



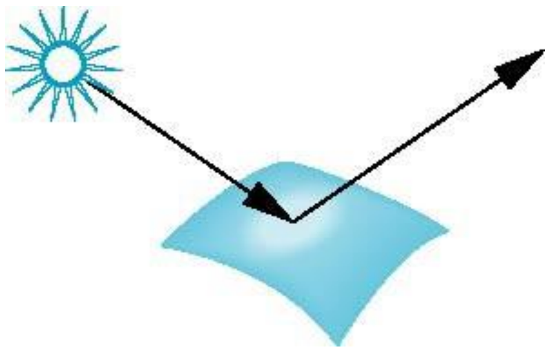
Simple Light Source Models

- Point source
  - Model with position and color
- Directional source
  - Distant source = infinite distance away (parallel)
- Ambient light
  - Same amount of light everywhere in scene
  - Models indirect light from reflecting surfaces

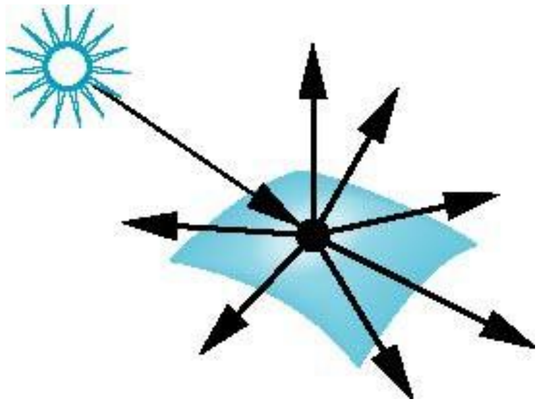


Surface Types

- Consider light traveling along a specific ray
- The smoother a surface, the more reflected light is concentrated in a single direction
  - Perfect mirror reflects perfectly in a single direction
- A very rough surface scatters light in all directions



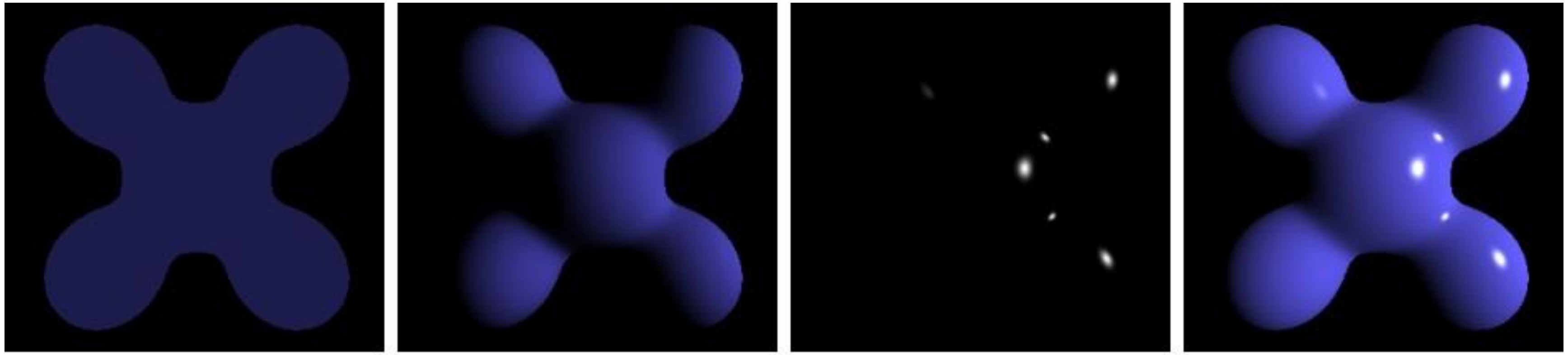
smooth surface = specular



rough surface = diffuse

ILLINOIS

The Phong Reflection Model: Term by Term



Ambient + Diffuse + Specular = Phong Reflection

$$I_{\mathrm{p}} = k_{\mathrm{a}} i_{\mathrm{a}} + \sum_{m \in \text{lights}} \left(k_{\mathrm{d}} (\hat{L}_m \cdot \hat{N}) i_{m,\mathrm{d}} + k_{\mathrm{s}} (\hat{R}_m \cdot \hat{V})^{\alpha} i_{m,\mathrm{s}} \right)$$

Ambient Light

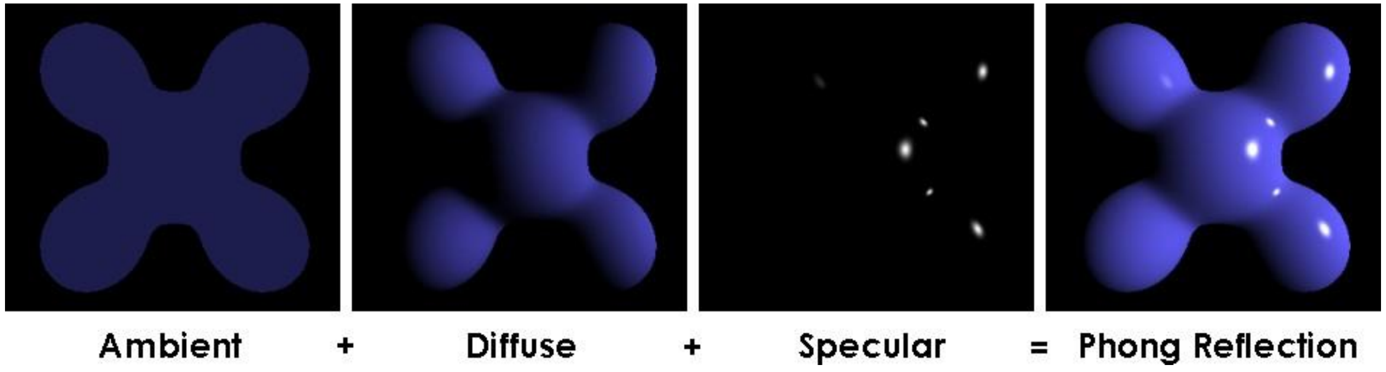
Result of multiple interactions between light sources and surfaces

- Kind of a hack meant to account for indirect light

Add k_a I_a to diffuse and specular terms

reflection intensity of ambient light

Remember that ki multiplications are component-wise multiplications of rgb values
$(k_r, k_g, k_b)(i_r, i_g, i_b) = (k_r i_r, k_g i_g, k_b i_b)$

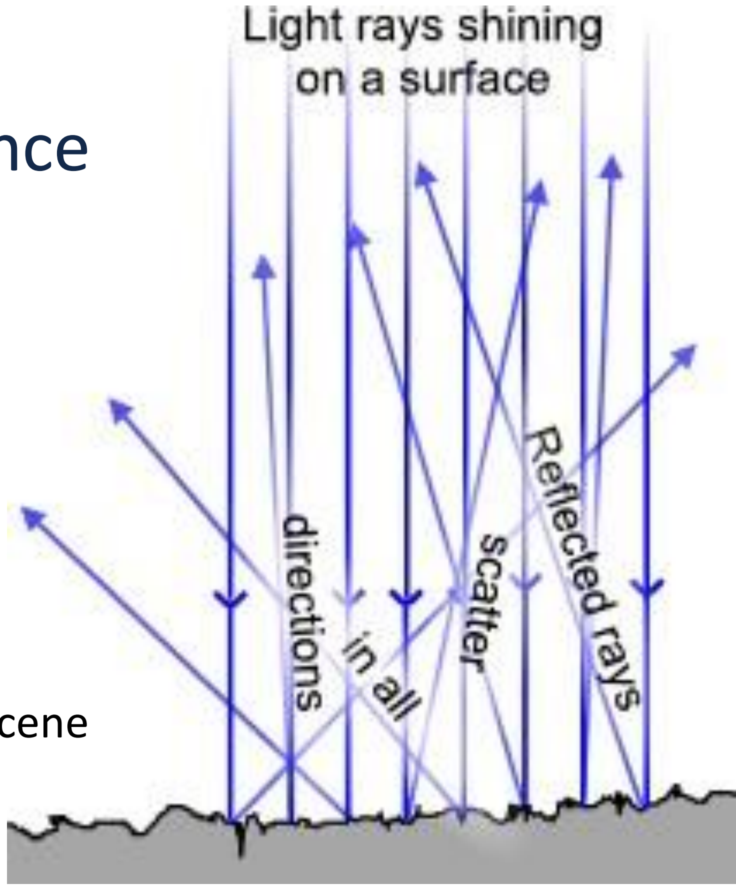


ILLINOIS

Modeling a Lambertian Surface – Diffuse Reflection

- Perfectly diffuse reflector
- Light scattered equally in all directions
- Amount of light reflected is affected by the angle of incidence
  - reflected light proportional to ***cosine of angle between L and N***
  - if vectors normalized

$$\cos(\theta) = \mathrm{L} \cdot \mathrm{N}$$



$$k_{\mathrm{d}}(\hat{L}_m \cdot \hat{N}) i_{m,\mathrm{d}}$$

The diffuse term in Phong reflection

k_d is the diffuse reflectance of the surface
$i_{m,d}$ is the diffuse color of m of n lights in the scene

ILLINOIS

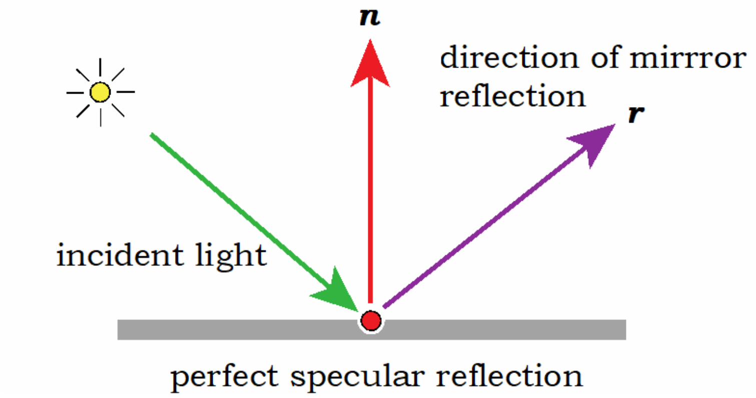
Diffuse Reflection Example

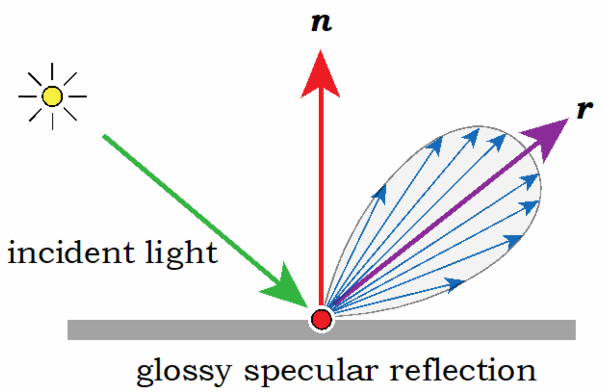
On the left, a barrel is illuminated by ambient light only. On the right, the barrel is illuminated by ambient light and one direct light source, the direction to which increases by 30 degrees to the right in each image. Lambert's cosine law causes the shading to darken as the angle between the surface normal and the direction to the light increases.

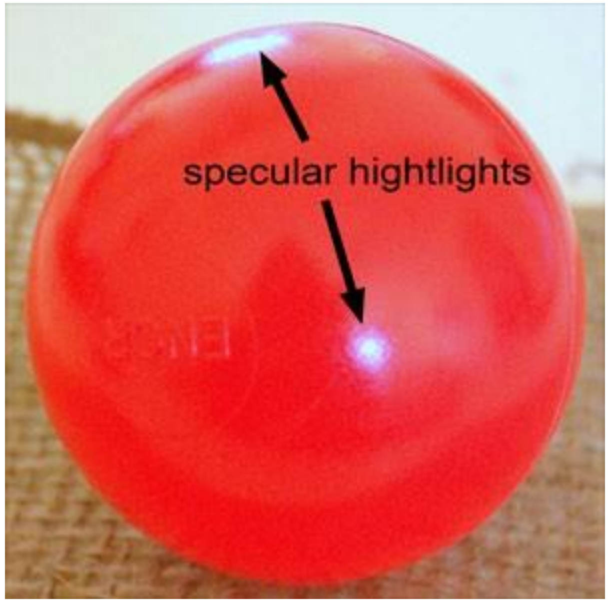
Lengyel, Eric. Foundations of Game Engine Development, Volume 2: Rendering (p. 115).

Specular Reflection

- Perfect specular reflection
  - Light is reflected in the single direction r
  - …the mirror reflection direction
- Glossy specular reflection
  - Scattering clustered around mirror reflection direction







Specular Reflection

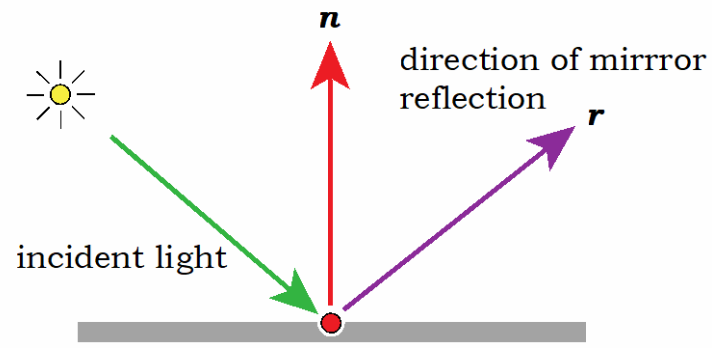
$$k_s(\hat{R}_m \cdot \hat{V})^\alpha i_{m,s}$$

The specular term in Phong reflection

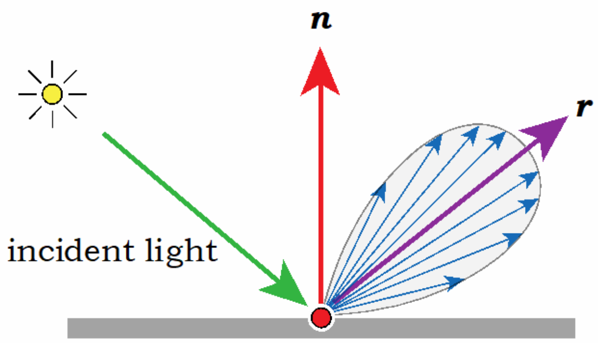
k_s is the specular reflectance of the surface
$i_{m,s}$ is the specular color of light m
R_m is the reflection vector for light m
V is the view vector from pixel to camera
α is the shininess coefficient

Specular reflection will be most powerful in the model when the view and reflection directions are closely aligned

$$(R_m \cdot V) = \cos(\phi)$$



perfect specular reflection

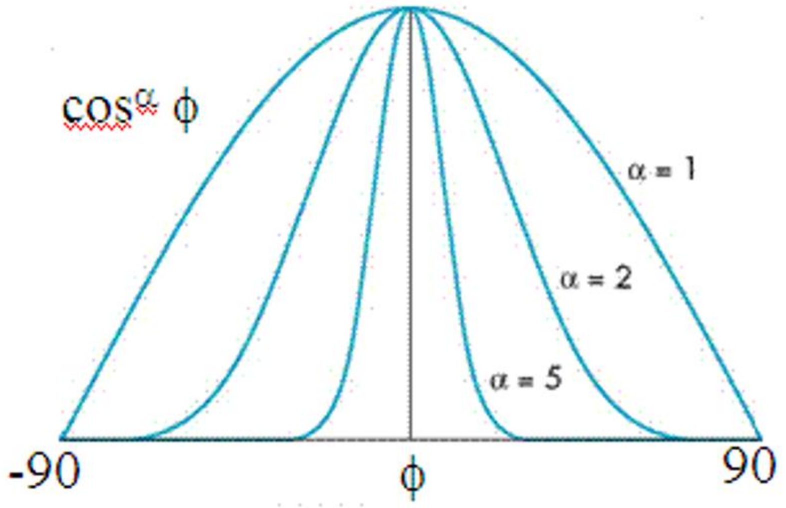


glossy specular reflection

ILLINOIS

Specular Reflection

$$k_s(\hat{R}_m \cdot \hat{V})^{\alpha} i_{m,s}$$



Reflectance determined by

- Alignment of view vector with mirror reflection vector
- Shininess coefficient

High coefficient means smoother look

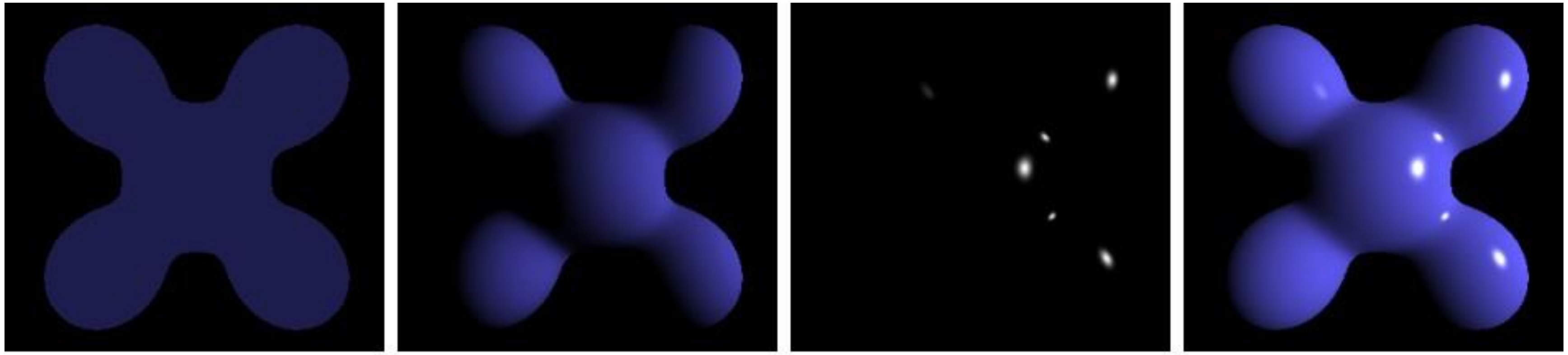
- Maybe 100 for metal
- Maybe 10 for plastic

Specular Reflection Example

On the left, a sci-fi weapon is rendered with diffuse shading only. On the right, specular shading is added with specular powers of 10, 50, and 200 from left to right.

Lengyel, Eric. Foundations of Game Engine Development, Volume 2: Rendering (p. 117).

Phong Reflection Model: Computed on Each Pixel



Ambient + Diffuse + Specular = Phong Reflection

$$I_{\mathrm{p}} = k_{\mathrm{a}} i_{\mathrm{a}} + \sum_{m \in \text{lights}} \left(k_{\mathrm{d}} (\hat{L}_m \cdot \hat{N}) i_{m,\mathrm{d}} + k_{\mathrm{s}} (\hat{R}_m \cdot \hat{V})^{\alpha} i_{m,\mathrm{s}} \right)$$

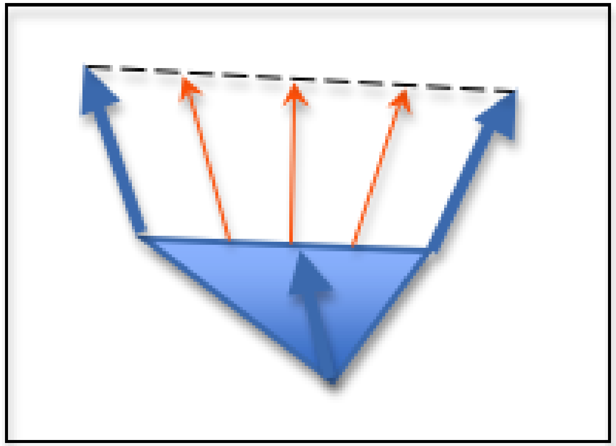
What data do we need at each pixel and how does it get there?

Pixel Data

One approach: pixel data is linearly interpolated from vertex data

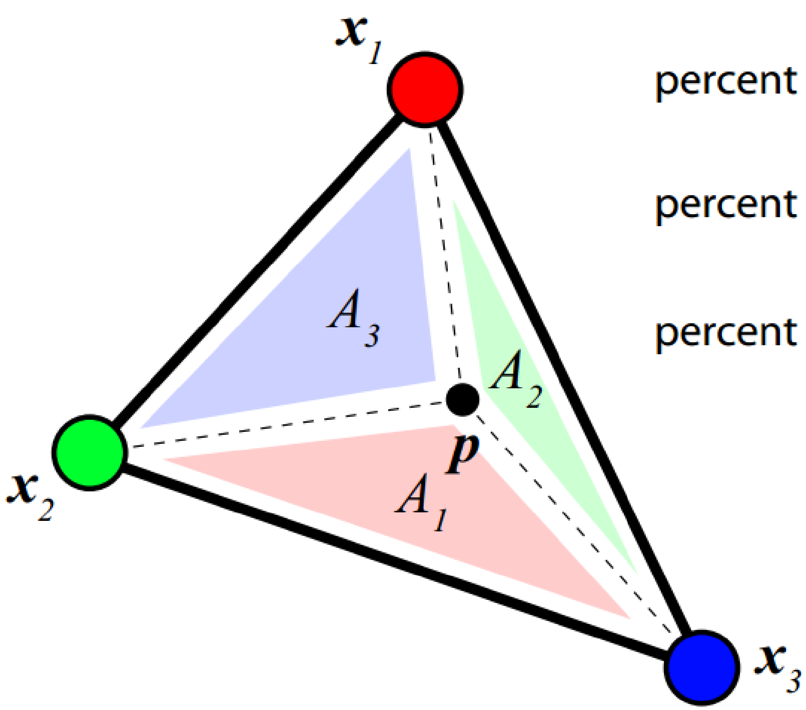
Each vertex has some attributes:

- Normal vector
- Position
- Texture coordinates



Compute a weighted average of each vertex attribute at each pixel

Barycentric Interpolation

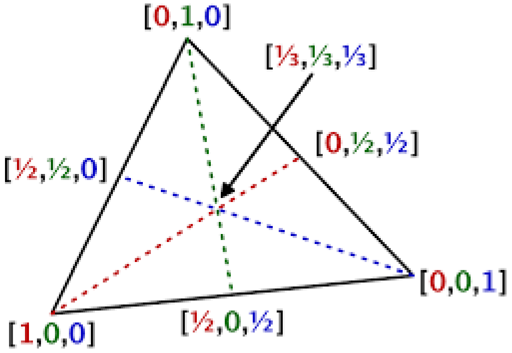


percent red $= \frac{A_1}{A} = \lambda_1$

percent green $= \frac{A_2}{A} = \lambda_2$

percent blue $= \frac{A_3}{A} = \lambda_3$

"barycentric coordinates"



Value at $\boldsymbol{p}$:

$$(A_1\boldsymbol{x}_1 + A_2\boldsymbol{x}_2 + A_3\boldsymbol{x}_3)/A$$

$$\Sigma_i \lambda_i = 1$$

Interpolated normal at point p would be $N_p = \lambda_1 N_1 + \lambda_2 N_2 + \lambda_3 N_3$

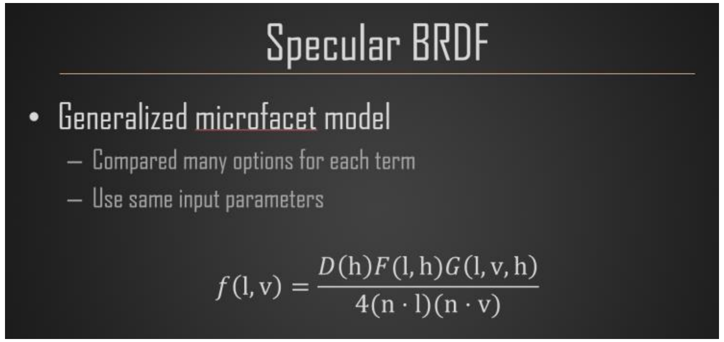
Physically Based Shading

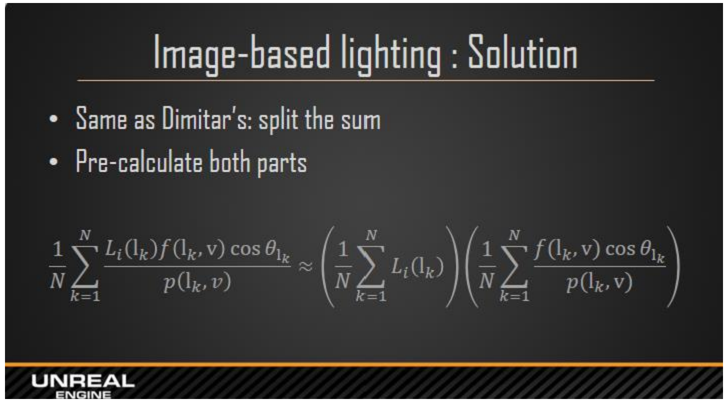
The Phong reflection model is non-physical…just sort of looks mostly right

- Does not accurately model physics of light reflection

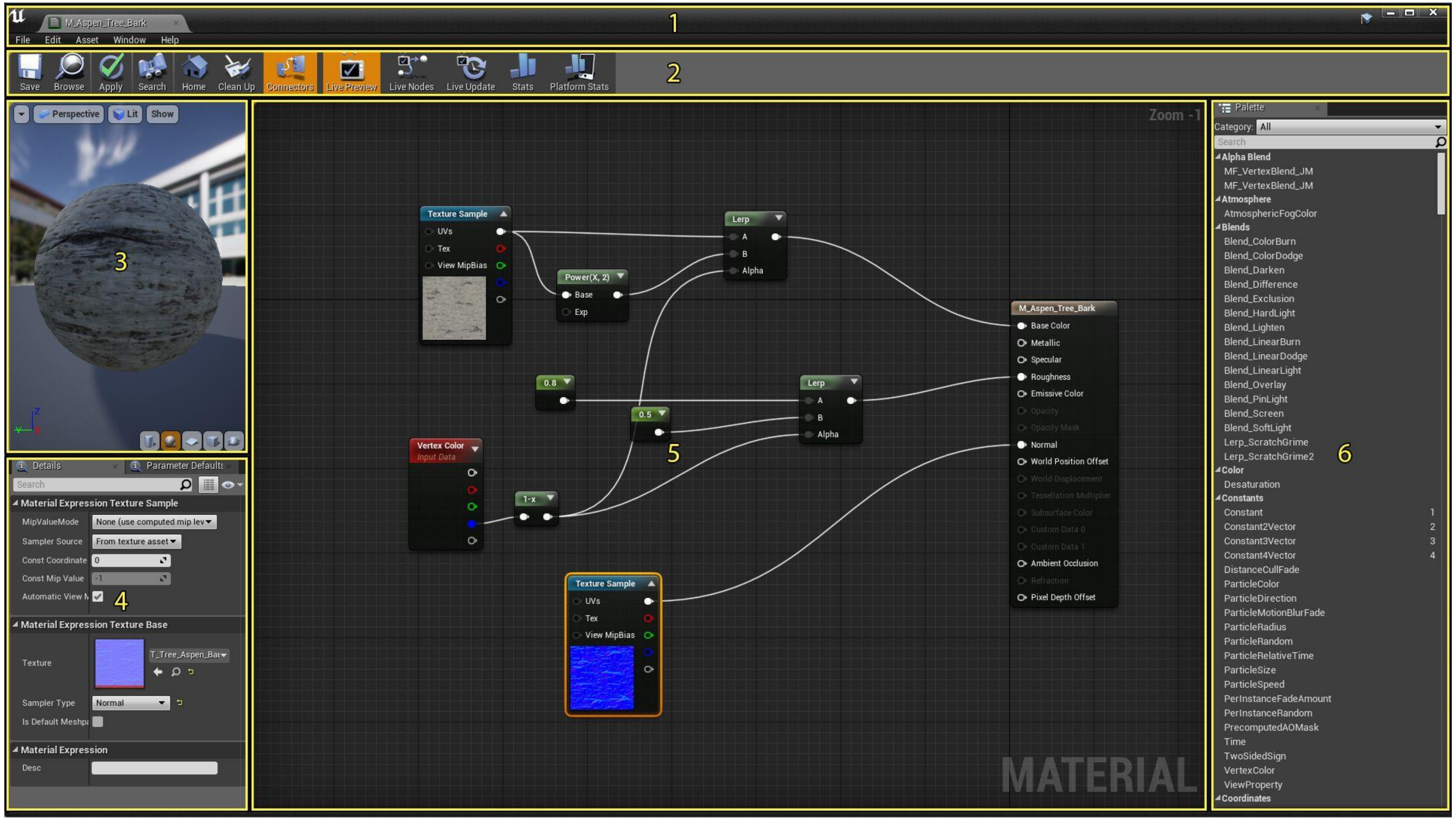
Modern shading has moved on to more physically based models

Much closer to photorealism





Unreal Engine Material Editor



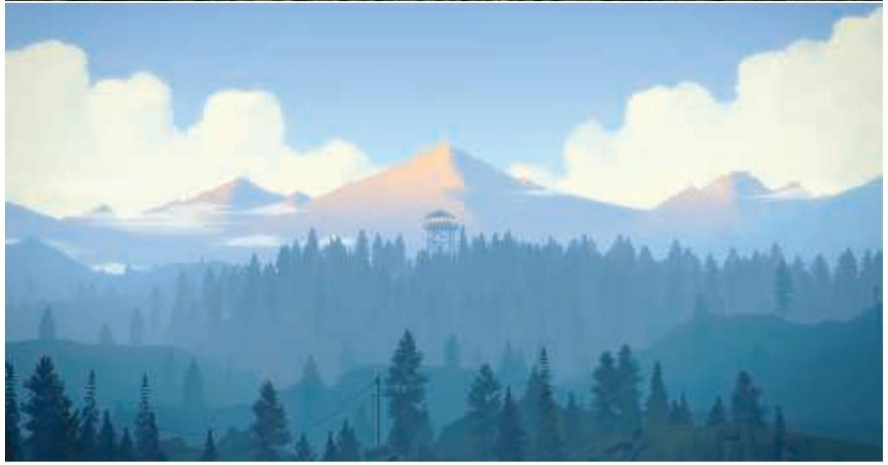
Modern material models are more complex than the Phong model

But…it’s still just some not-too-complicated math to generate a color….

Power of Programmable Shaders

Can render scenes in a procedurally defined art style for a given game

Unreal Engine 5 rendering



Scene from *Firewatch*