Computer Graphics: Lighting

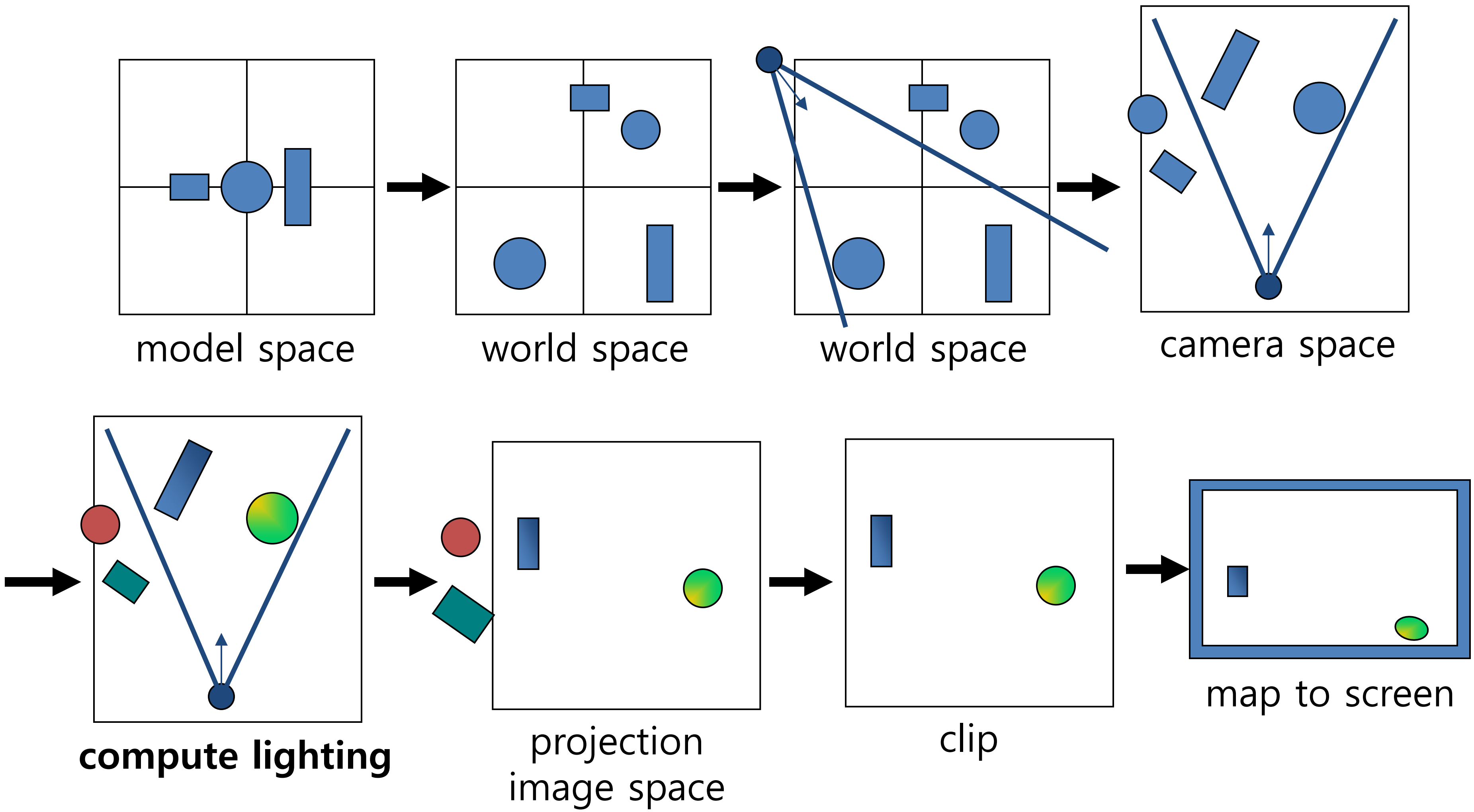
Dept. of Game Software
Yejin Kim

Overview

- Lighting Interaction
- Illumination Models
  - Global/Local illumination
- Lighting Sources
- Shading
- Tutorials

Rendering Pipeline

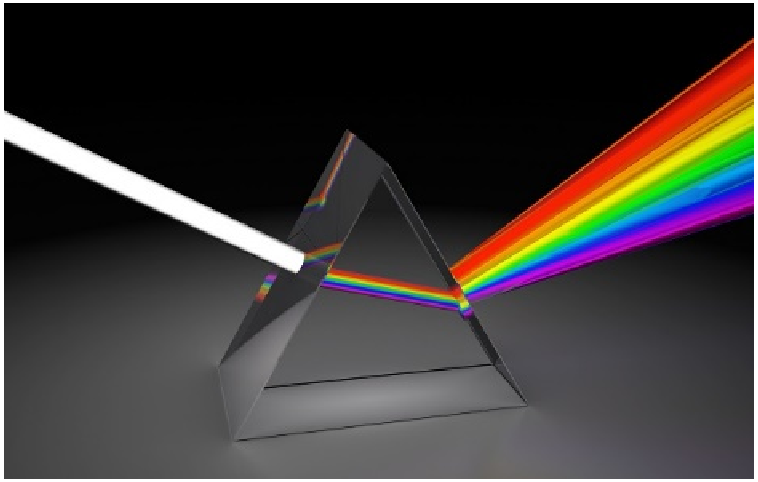
- Geometry stages: Summary



Lighting Interaction

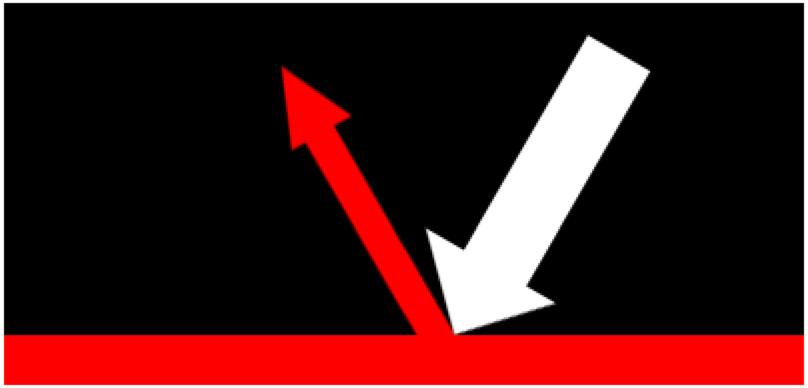
- Rendered image
  - A result of complex interaction between light and objects

- Physical interactions of light
  - Absorption(흡수)
  - Reflection(반사)
  - Transmission & refraction(전달 & 굴절)
  - Diffraction(회절)



Lighting Interaction

- Absorption
  - Light가 surface에 흡수되는 성질
  - 물리적으로 light의 photon(광자) energy가 흡수



Why does red look red?

Lighting Interaction

- Reflection
  - Light가 surface에 반사되는 성질

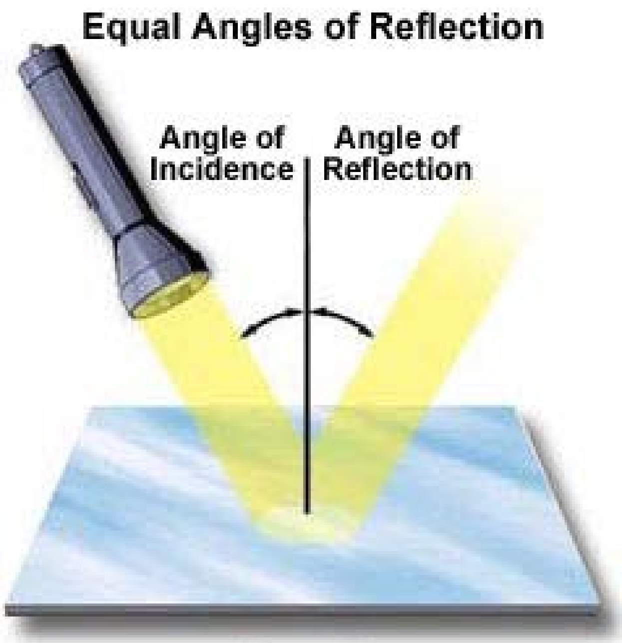


Figure 1

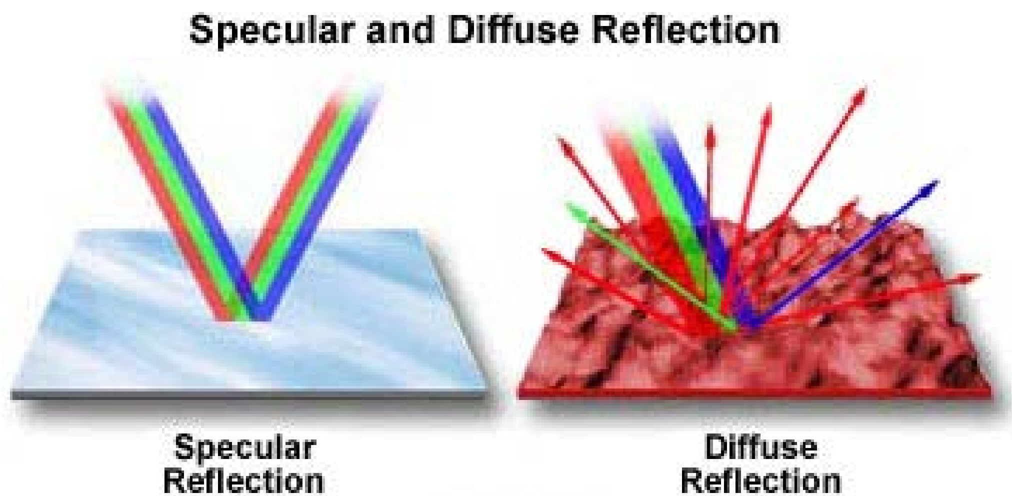
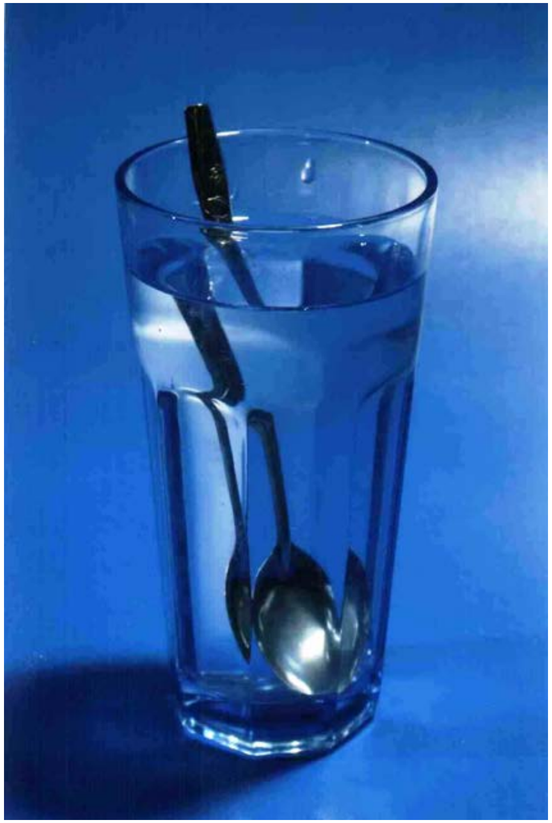


Figure 2

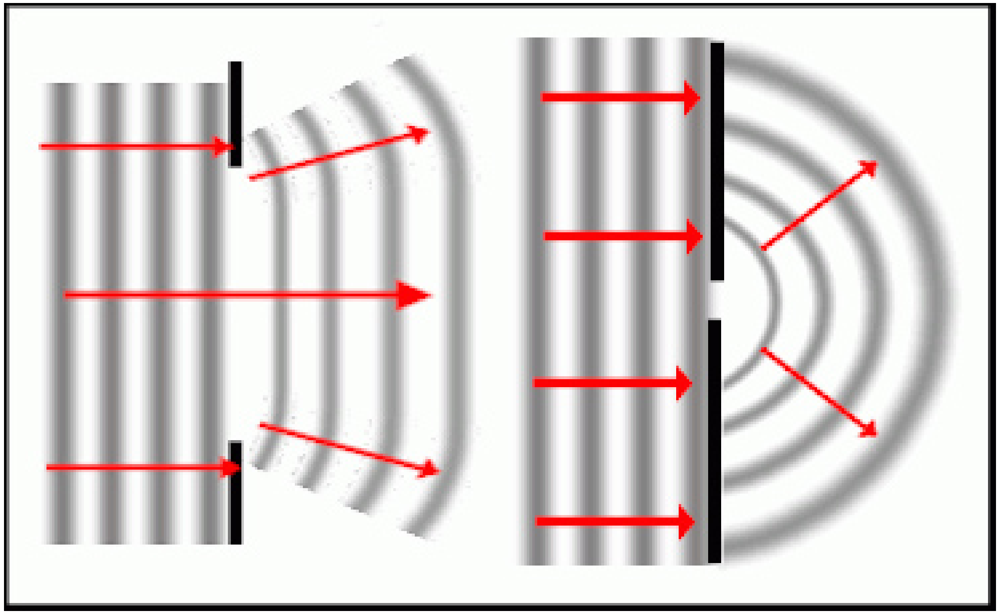
Lighting Interaction

- Transmission & Refraction
  - Light가 object를 통과하는 성질
  - 부분적으로 absorption 또는 refraction 됨



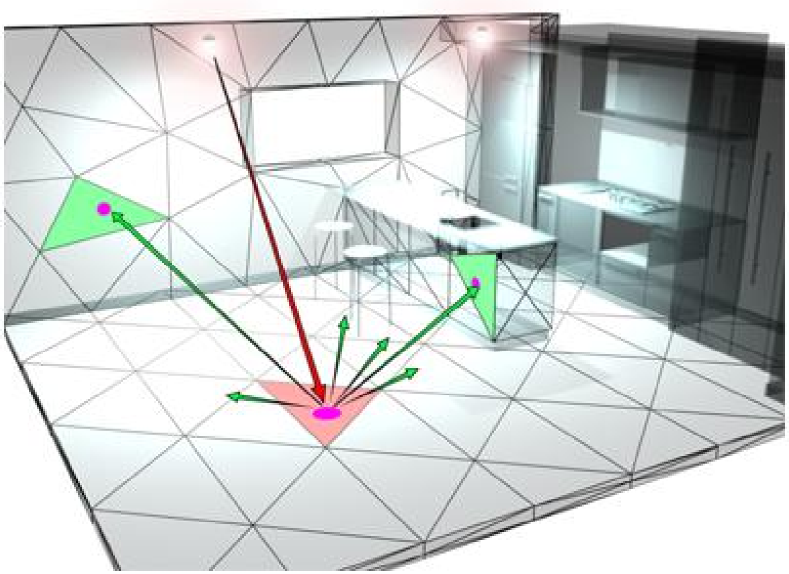
Lighting Interaction

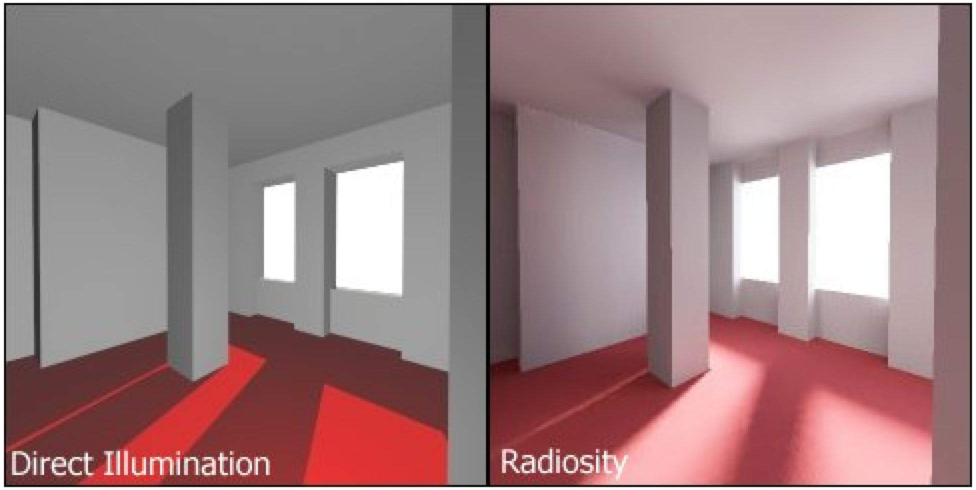
- Diffraction
  - Light가 object의 가장자리를 부딪혀 bending(휘는) 성질
  - 부딪치는 공간에 따라 bending 정도가 다름



Illumination Models

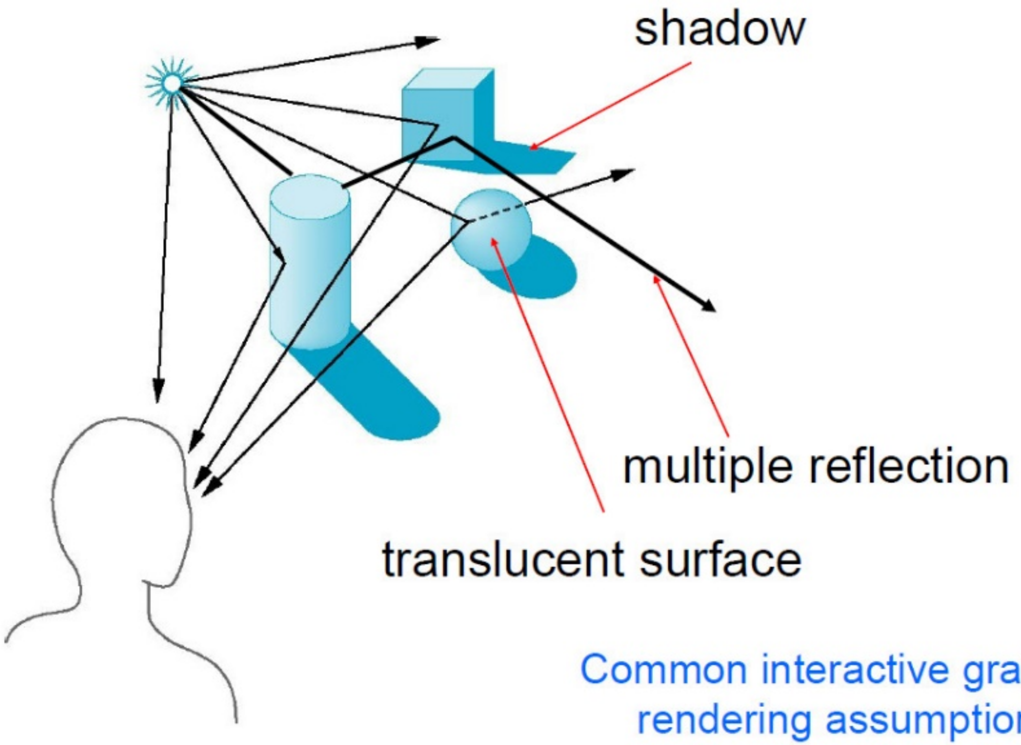
- Illumination(Lighting) model
  - Determine the color and brightness of the surface point by simulating the interaction of light with surface attributes and lighting parameters
  - Global illumination: Accurate, high-quality, slow
  - Local(Direct) illumination: Approximate, low-quality, fast





Global Illumination

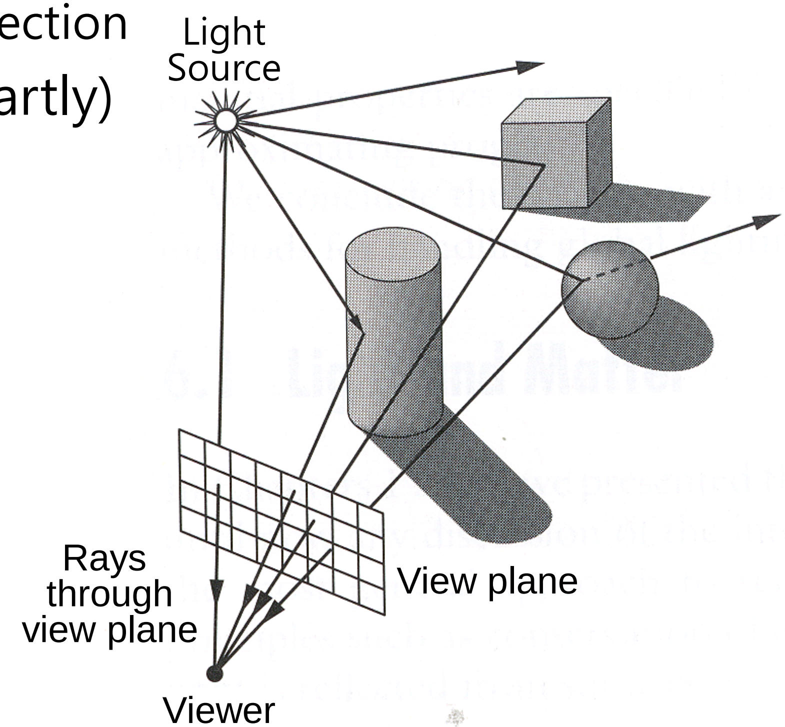
- Global illumination model
  - Consider the interaction of light from all surfaces (or sources) in the scene
  - **Ray tracing**, radiosity, Monte Carlo methods, etc.



Common interactive graphics rendering assumption:
only model local lighting effects

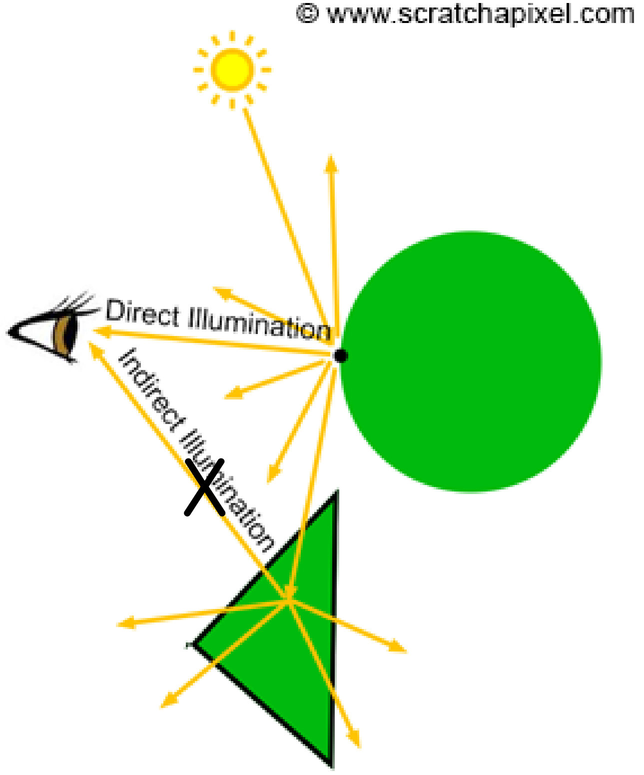
Global Illumination

- Ray tracing
  - Trace the path of light as pixels in an view plane and simulate the effects of its encounters with scene objects
  - Forward: trace rays from the light sources to the viewer
    - Only a tiny fraction of rays will reach the image → extremely slow
  - Backward: trace rays from the viewer into the scene
    - Each pixel gets light from just one direction
  - Bi-directional: backward + forward (partly)



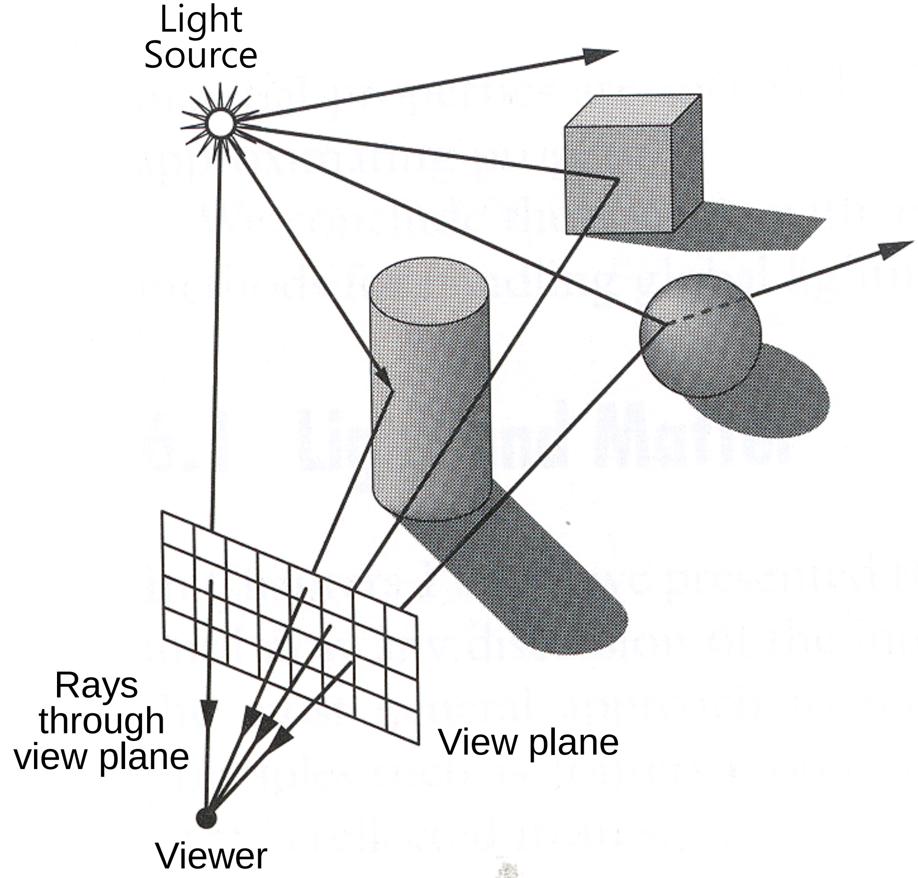
Local Illumination

- Local(Direct) illumination model
  - Only consider the light that directly hits a surface and is reflected to the viewer
  - **Ray casting, Phong reflection**(Phong illumination, Phong lighting), Blinn-Phong reflection, polygon shading, etc.



Local Illumination

- Visible surface determination
  - The color of each pixel on the view plane depends on the radiance emanating from visible surfaces
  - Simplest method: **Ray casting**



Local Illumination

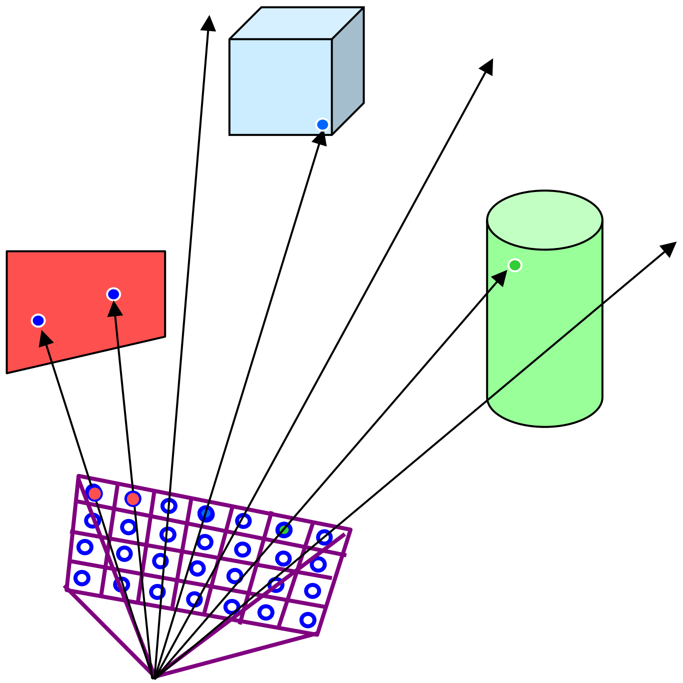
- Ray Casting
  - For each sample,

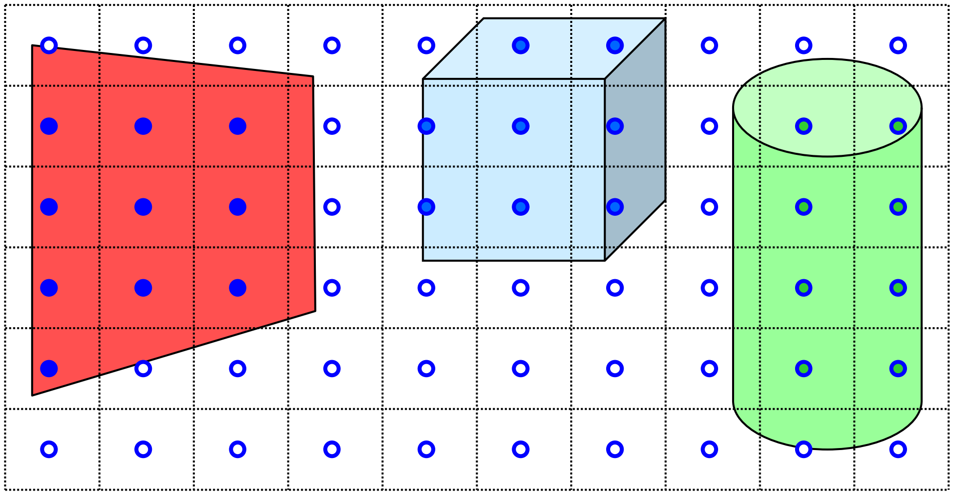
    Construct ray from eye position through view plane

    Find first surface intersected by ray through pixel

    Compute color of sample based on surface radiance

  → Visual surface detection

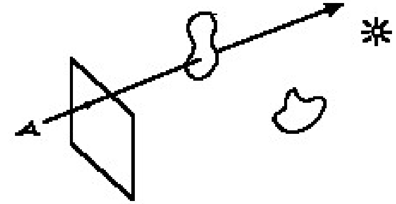




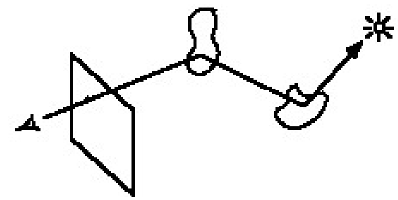
Local Illumination

- Ray casting vs Ray tracing
  - Performance
  - Realism
  - Visual effects

1st path = ray casting



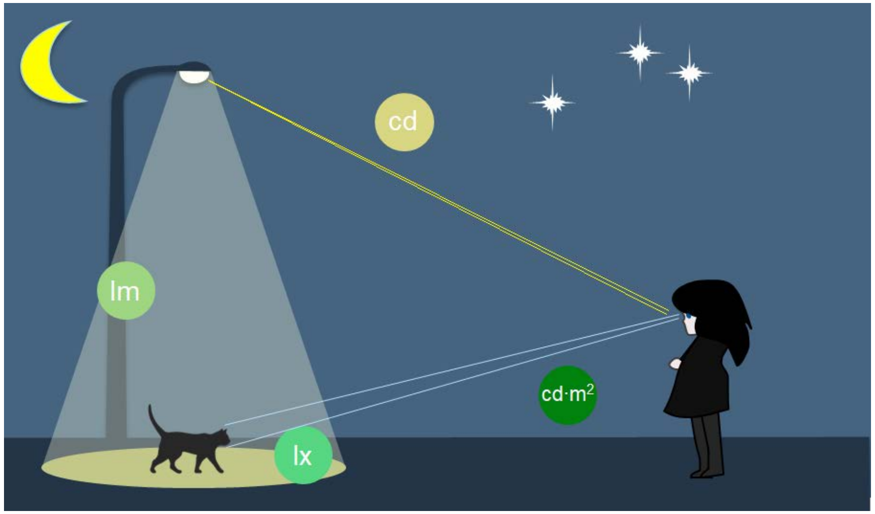
recursive = ray tracing



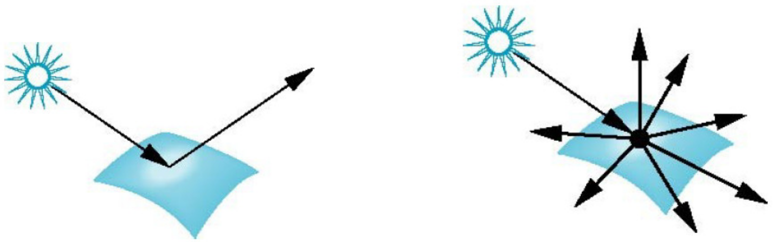
Rasterized 3D graphics imagery   Ray traced 3D graphics imagery ©Siliconarts

Local Illumination

- Lighting parameters
  - Light source emission
  - **Surface reflectance**
  - Atmospheric attenuation
  - Camera response



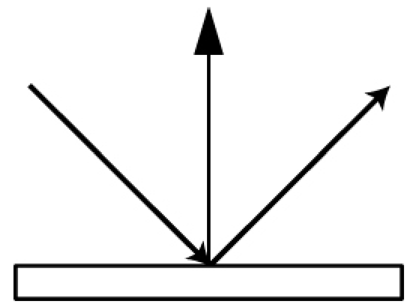
- Surface reflectance
  - The amount of light reflected by the surface of an object
  - Depending on the surface types of an object
    - Smooth surfaces reflect like mirrors
    - Rough surfaces scatter light
    - Generally, smooth + rough

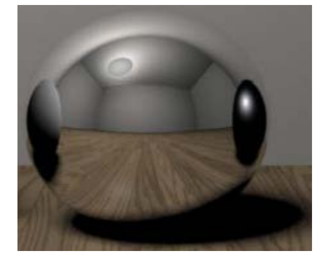


Smooth reflective surface **Rough diffuse surface**

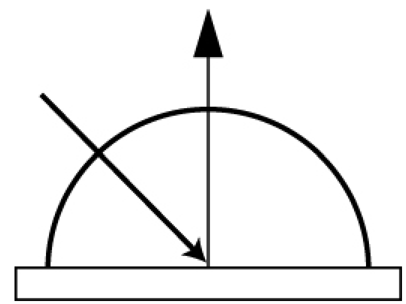
Local Illumination

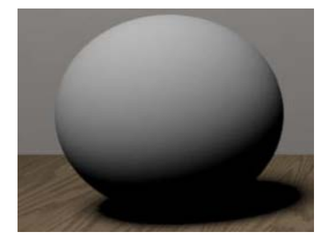
- Surface reflectance: Reflection types
  - Mirror reflection
    - Ideal reflection
    - Reflection law



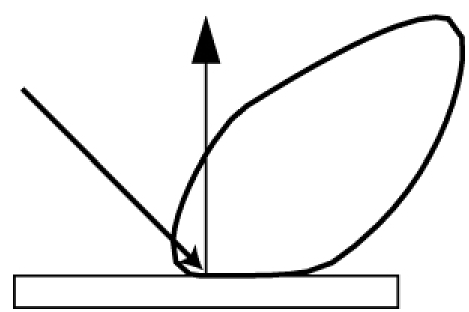


  - Diffuse reflection
    - Matte, flat finish
    - Lambert's law



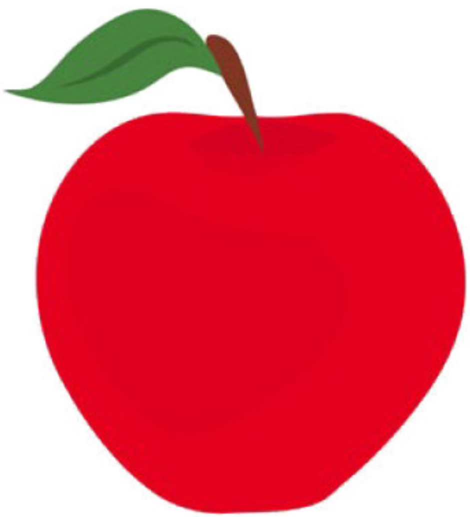


  - Specular reflection
    - Highlights and glossy
    - Micro-facet model



Local Illumination

- Basic concept: What makes an apple red?



Child's view:
"an apple is red"

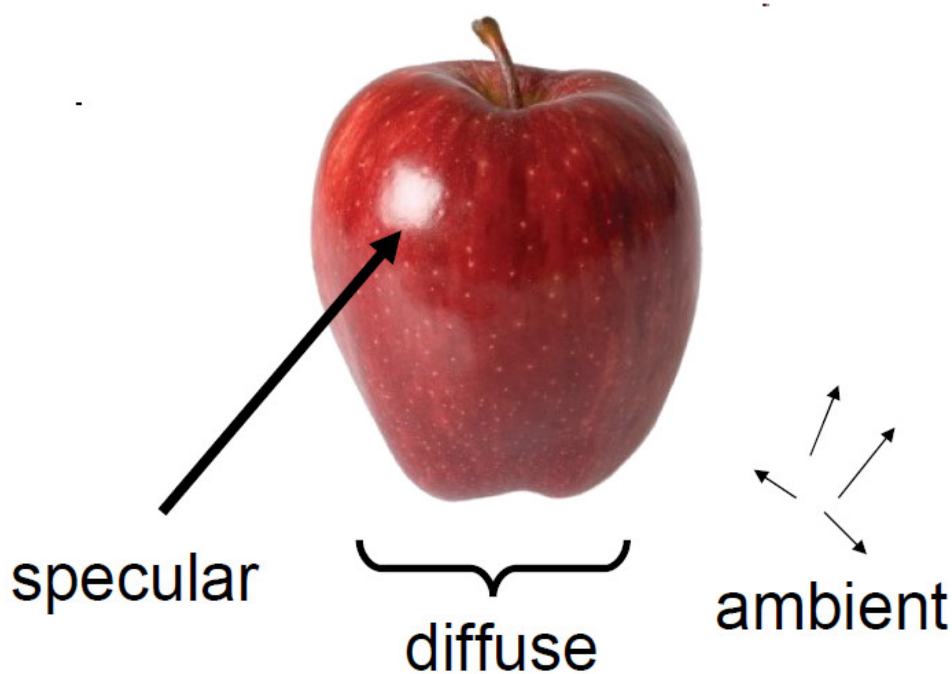


Image synthesis view:
"light, surface, and material interact to reflect light perceived as color, modeled via simplifying assumptions"

Local Illumination

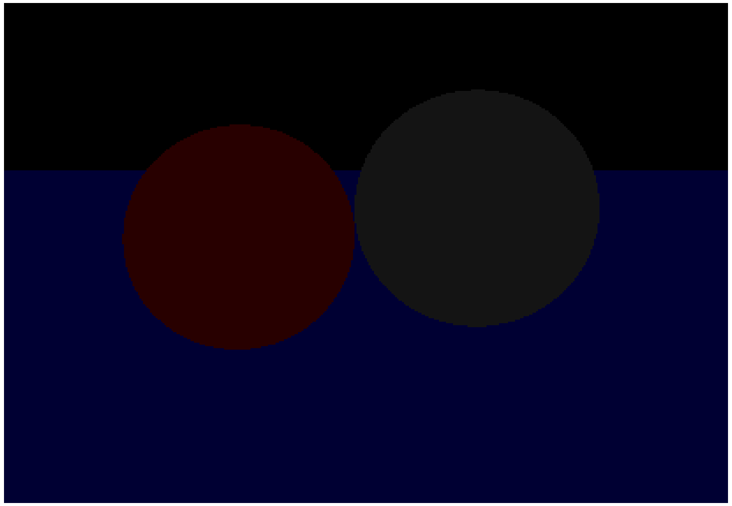
- Ambient light
  - Indirect light that is the result from the light reflecting off other surfaces in the scene

$$I_A = k_a I_a$$

  I_a = intensity of ambient light

  k_a= ambient reflection coefficient, $0 < k_a < 1$

  - 3 equations for each color: red, green, blue
  - Independent of surface direction and light source position
  - Each light source has I_a
  - Determines color of shadows



Local Illumination

- Diffuse light
  - Reflects light from the surface equally in all directions

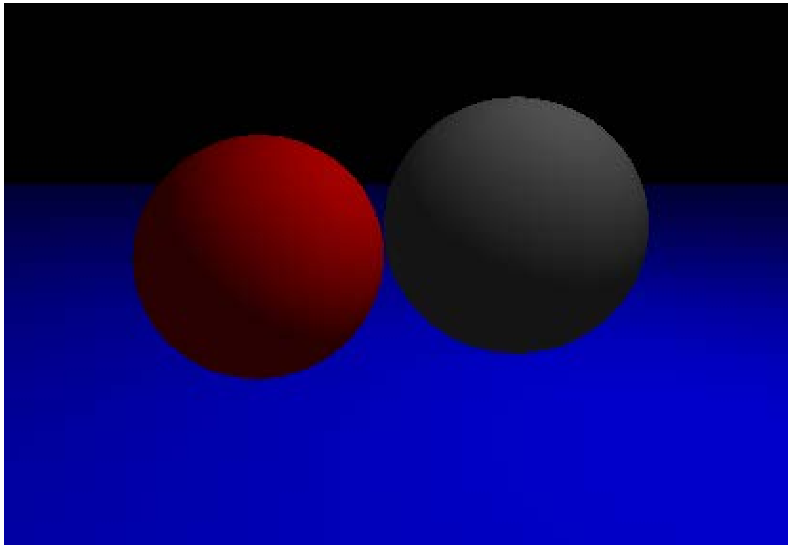
$$I_D = I_d k_d \cos(\theta) = I_d k_d (L \cdot N)$$

I_d = intensity of light source

k_d = diffuse reflection coefficient, $0 < k_d < 1$

θ = angle between normal and direction to light

  - Lambertian reflection (Lambertian surfaces)
  - Independent of viewer position



Local Illumination

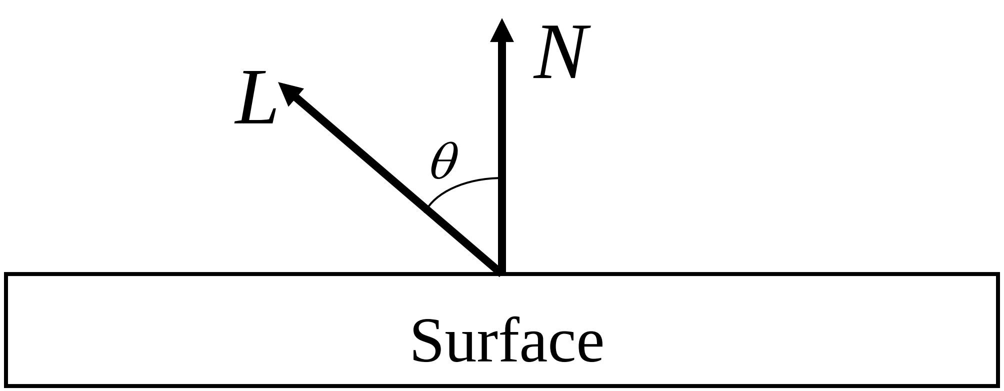
- Lambert's law
  - How much light the surface receives from a light source depends on the angle between its angle and the vector from the surface point to the light (light vector)
  - Brightness of surface is same regardless of angle of view

$$\mathbf{cos}(\boldsymbol{\theta}) = \boldsymbol{max}\,(\boldsymbol{L} \cdot \boldsymbol{N}, \mathbf{0})$$

L = direction to light source

N = surface normal

θ= angle between N and L



Local Illumination

- Specular light
  - Reflect light from the surface to certain directions

$$\boldsymbol{I = I_s k_s \cos^n(\alpha) = I_s k_s (R \cdot E)^n}$$

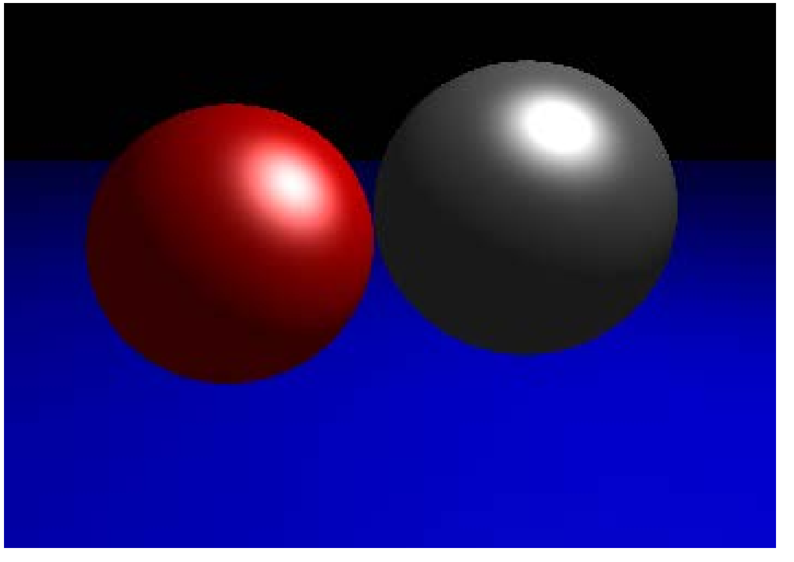
I_s = intensity of light source

k_s = specular reflection coefficient

θ = angle between reflected light, *R*, and direction to eye, *E*

n = specular coefficient

  - Reflection's law
  - Dependent of viewer position
  - Bright spots on objects around *R*



Local Illumination

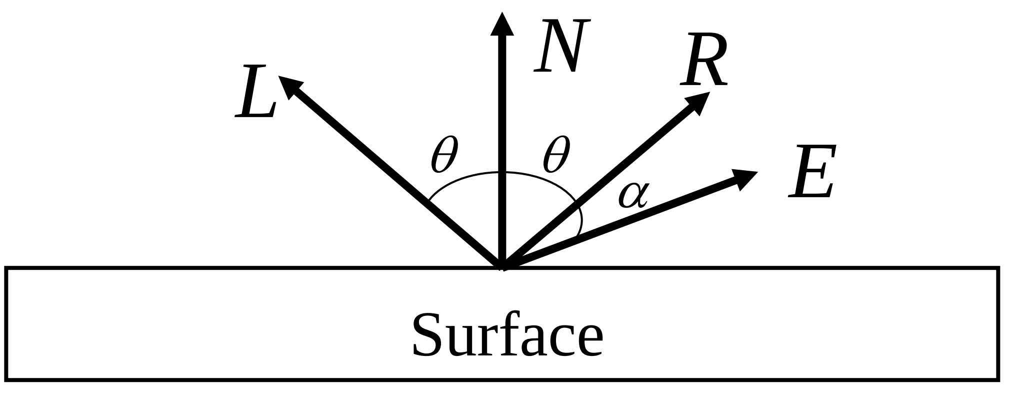
- Reflection's law
  - Reflection light depends on the viewer's position

$$\cos(\alpha) = max(R \cdot E, 0)$$

R = reflected light

E = direction to eye (viewer)

α = angle between R and E



Local Illumination

- Specular light
  - Calculating reflected vector, *R*

$P = N \cos\theta \; /L/ \;\; /N/$ projection of *L* onto *N*

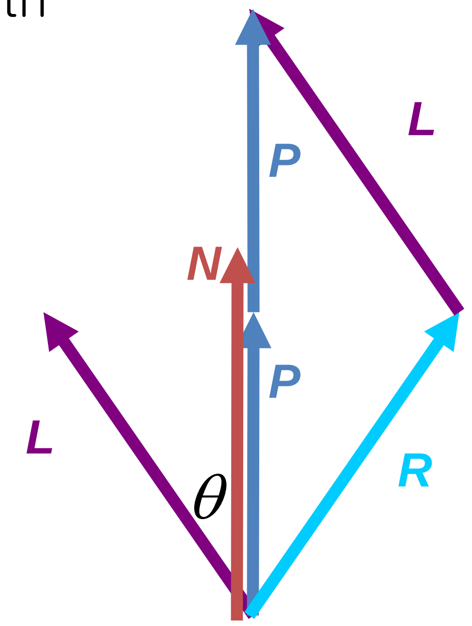
$P = N \cos\theta$ *L*, *N* are unit length

$P = N\,(\,N \cdot L\,)$

$2\,P = R + L$

$2\,P - L = R$

$2\,(N\,(\,N \cdot L\,)) - L = R$



Local Illumination

- Specular light

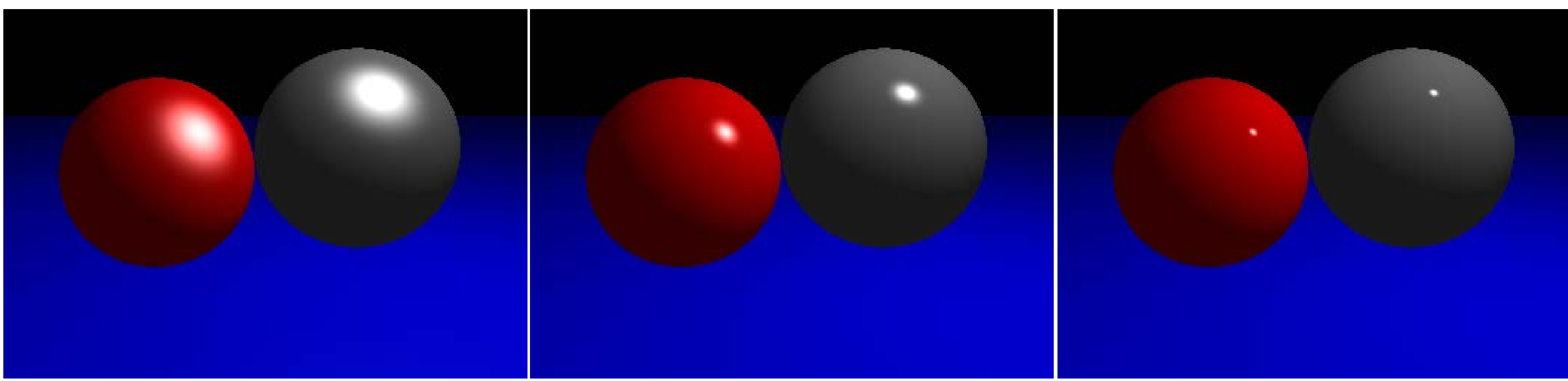
$$\boldsymbol{I = I_s k_s \cos^n(\alpha) = I_s k_s (R \cdot E)^n}$$

I_s = intensity of light source

k_s = specular reflection coefficient

θ = angle between reflected light, *R,* and direction to eye, *E*

n = specular coefficient

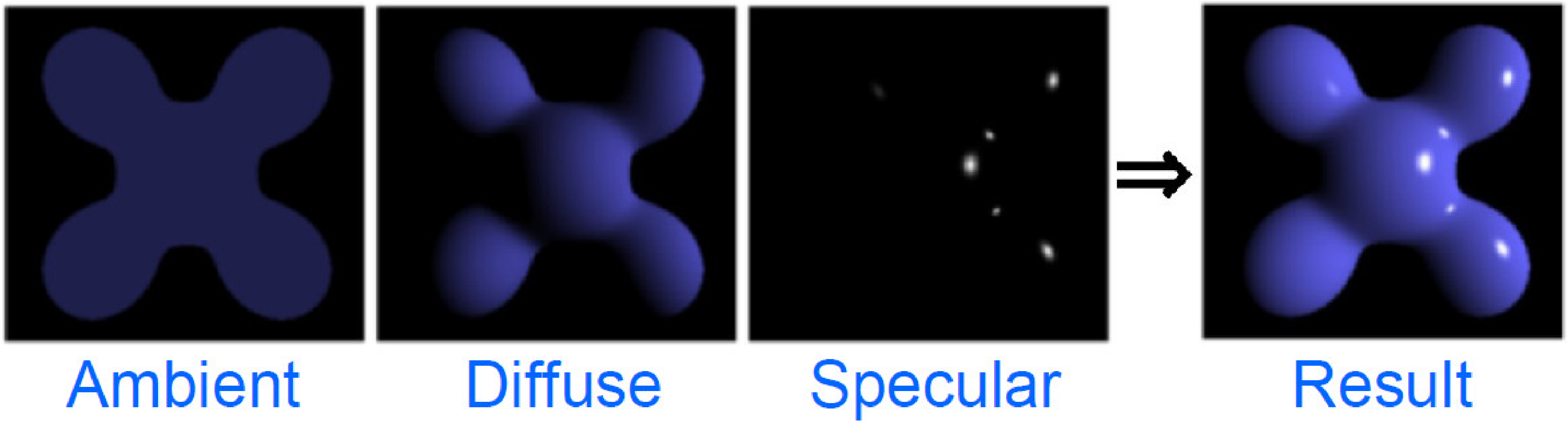


n = 5 *n = 50* *n = 500*

Local Illumination

- Phong reflection model
  - Ambient light: Uniform light caused by surface reflections in scene
  - Diffuse light: Light scattered equally in all directions
  - Specular light: Highlights on shiny surfaces

$$I_{total} = k_a I_a + \sum_{i=1}^{lights} \left(I_d k_d \max(L_i \bullet N, 0) + I_s k_s \max(R_i \bullet E, 0)^n (L_i \bullet N > 0) \right)$$



Local Illumination

- Emissive light
  - Objects that emit light
  - Typically models as an emissive color
  - e.g. incandesce(백열), fluorescence(형광), phosphorescence(인광), burning, etc.

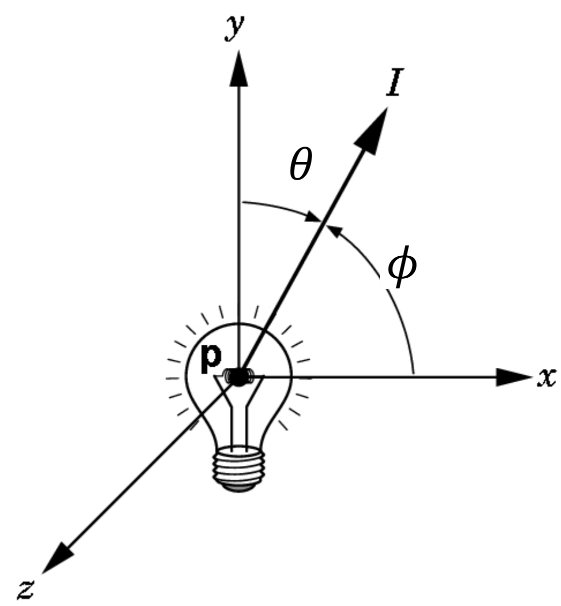



Lighting Sources

- Lighting parameters
  - Emit different amounts of light for each
    - Location: (x, y, z)
    - Direction: (θ, ϕ)
    - Wavelength: (λ)
  - Illumination function: $I(x, y, z, \theta, \phi, \lambda)$ for each color

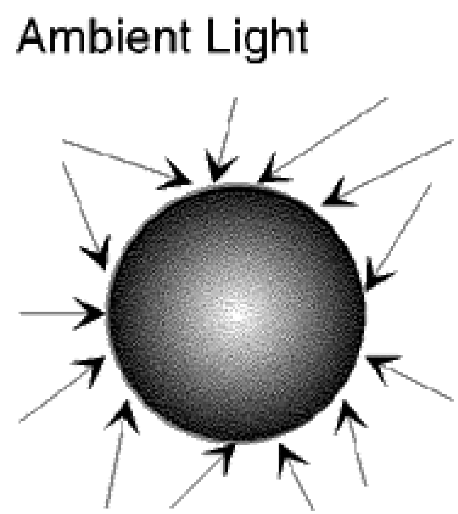
$$I = \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} I_r(x, y, z, \theta, \phi, \lambda) \\ I_g(x, y, z, \theta, \phi, \lambda) \\ I_b(x, y, z, \theta, \phi, \lambda) \end{bmatrix}$$

  - e.g. Ambient, directional, point, spot, etc.



Lighting Sources

- Ambient lights
  - Uniform lighting
  - Simulate the combination of light reflections from various surfaces
  - Intensity doesn't vary with $I(x, y, z, \theta, \phi)$



Lighting Sources

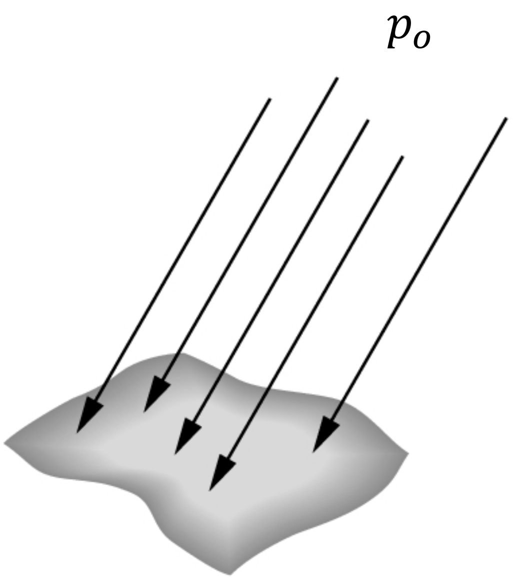
- Directional(Distant) light
  - Emit light in parallel direction without attenuation based on the distance

    $$I(p_o) = [I_r(p_o), I_g(p_o), I_b(p_o)]$$

  - In homogeneous coordinate

    $$p_o = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

  - e.g. Sun



Lighting Sources

- Point light
  - Emit light equally in all direction from a position

    $$I(p_o) = [I_r(p_o), I_g(p_o), I_b(p_o)]$$

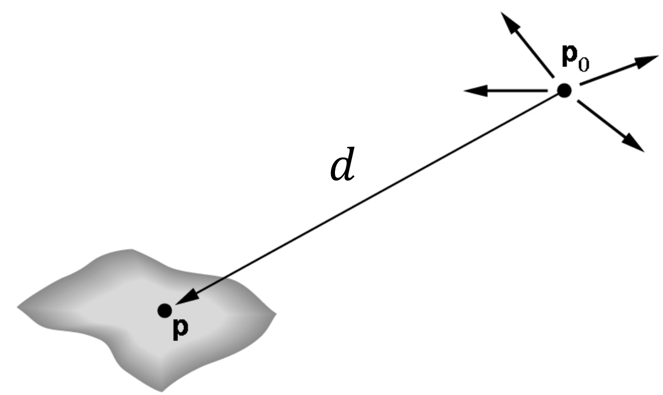
  - Farther objects receive less light from a a position

    $$I(p, p_o) = (\frac{1}{d^2})[I_r(p_o), I_g(p_o), I_b(p_o)]$$

  - In homogeneous coordinate

    $$p_o = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

  - e.g. Light bulb



Lighting Sources

- Spotlight
  - Emit light in a narrow range of angles (cone shape)

    $\mathbf{P}_s$: apex(정점)

    $\mathbf{I}_s$: light direction

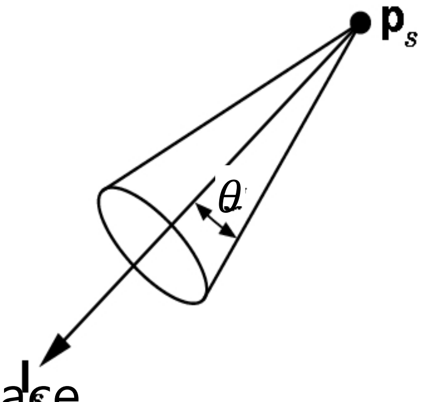
    $\boldsymbol{\theta}$: angle

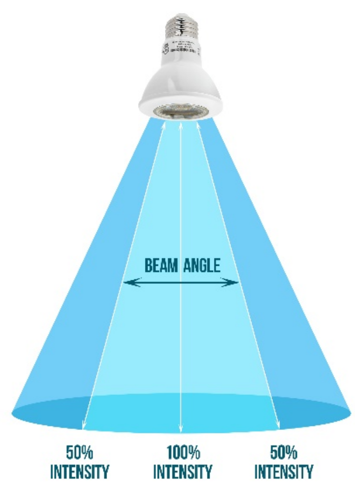
  - More realistic case: a gradual falloff of light

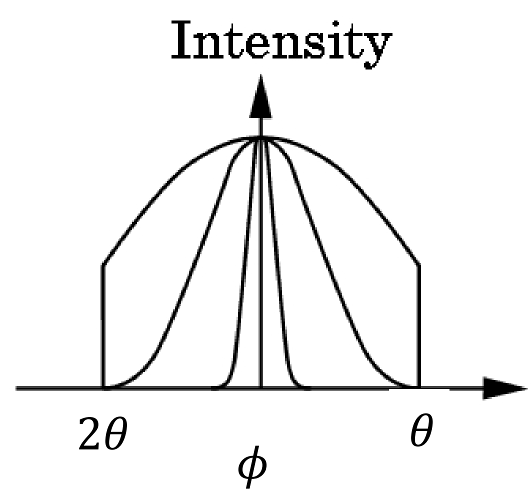
    $\cos^e \phi = (\mathbf{v}_s \cdot \mathbf{I}_s)^e$

    $\mathbf{v}_s$: vector with angle ϕ from $\mathbf{P}_s$ to a point on surface

  - e.g. Flashlight



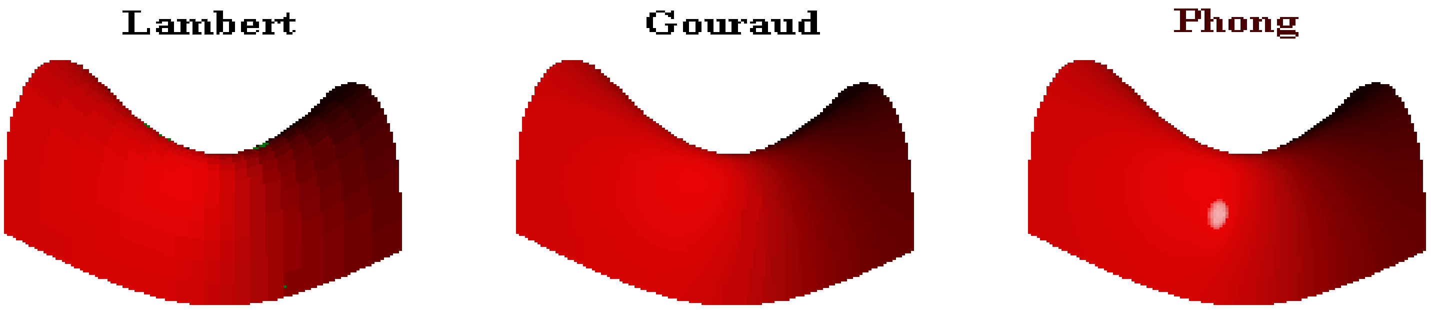




Shading

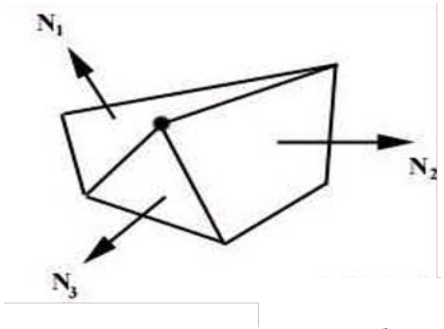
- Shading
  - Apply the illumination models at a set of points and colors the image pixels

- Shading techniques
  - Flat(Lambert) shading
  - Gouraud shading
  - Phong shading

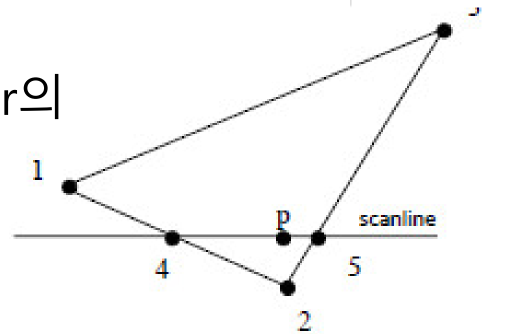


Shading

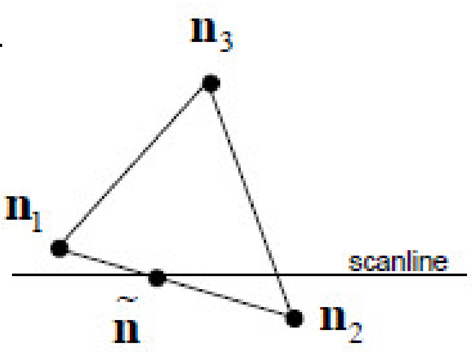
- Flat shading
  - 각 face의 normal vector를 구해 lighting source로부터의 intensity 값을 계산



- Gouraud shading
  - 각 vertex의 normal vector를 주위 face의 normal vector의 평균값으로 구함
  - Interpolation을 사용하여 face의 pixel별 intensity 계산



- Phong shading
  - 각 vertex의 normal vector를 구한 후 interpolation으로 face의 pixel별 normal vector를 계산
  - 각 pixel별로 light source으로부터의 intensity를 계산

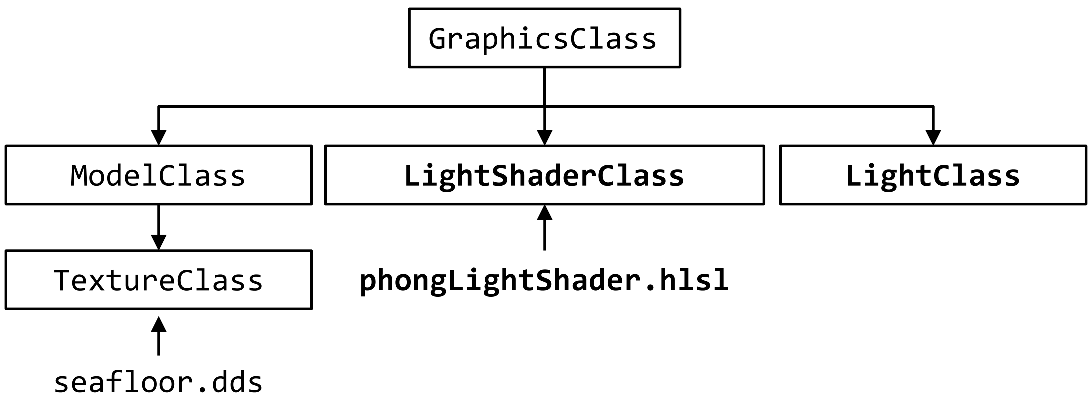


Tutorials

- Phong Lighting
  - Diffuse, Ambient, Specular Lights
- Multiple Point Lights

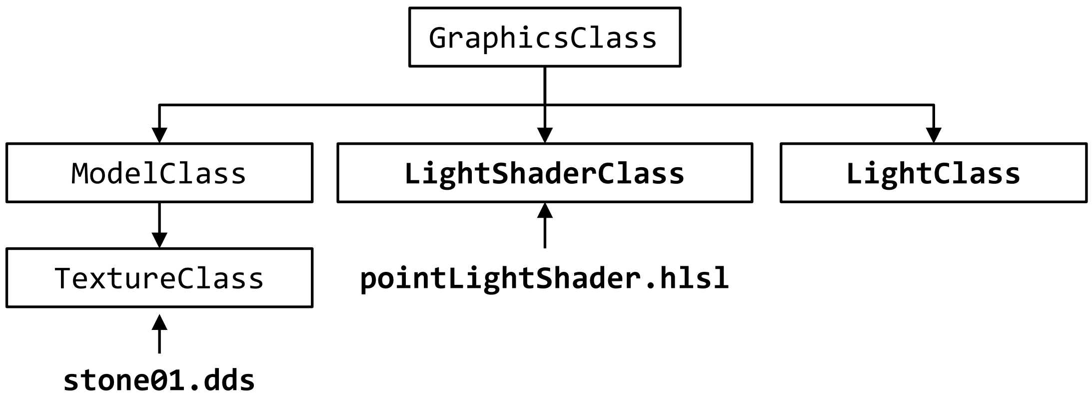
5-1 Phong Lighting

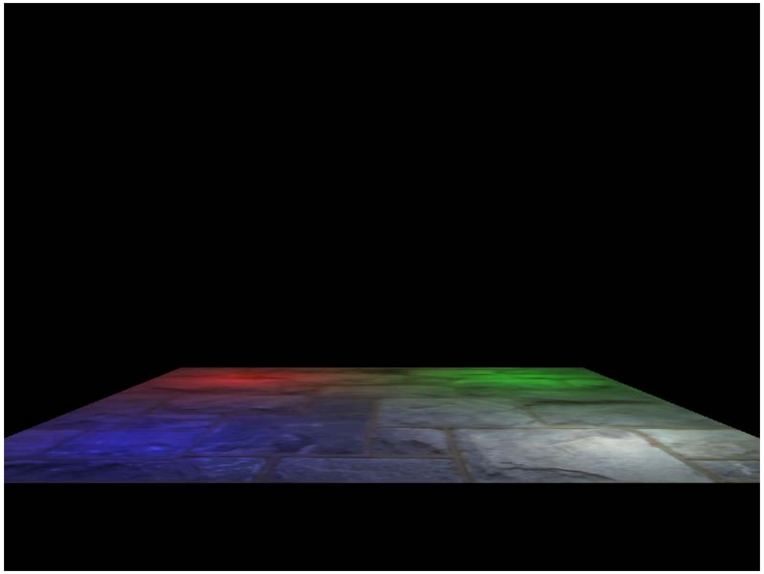
- Adding Phong reflection model to the framework
  - **LightClass**: handles ambient, diffuse, specular lights
  - **LightShaderClass**: render a lighted texture using HLSL



5-2 Multiple Point Lights

- Adding multiple point lights to the framework
  - **LightClass**: handles a point light
  - **LightShaderClass**: render a lighted texture using HLSL





References

- Wikipedia
  - www.wikipedia.org
- Introduction to DirectX 11
  - www.3dgep.com/introduction-to-directx-11
- Raster Tek
  - www.ratertek.com
- Braynzar Soft
  - www.braynzarsoft.net)
- CS 445: Introduction to Computer Graphics *[Aaron Bloomield]*
  - www.cs.virginia.edu/~asb/teaching/cs445-fall06

Q & A