ARLAB

ME/CprE/ComS 557

Computer Graphics and Geometric Modeling

Light and Material

September 22, 2015

Rafael Radkowski

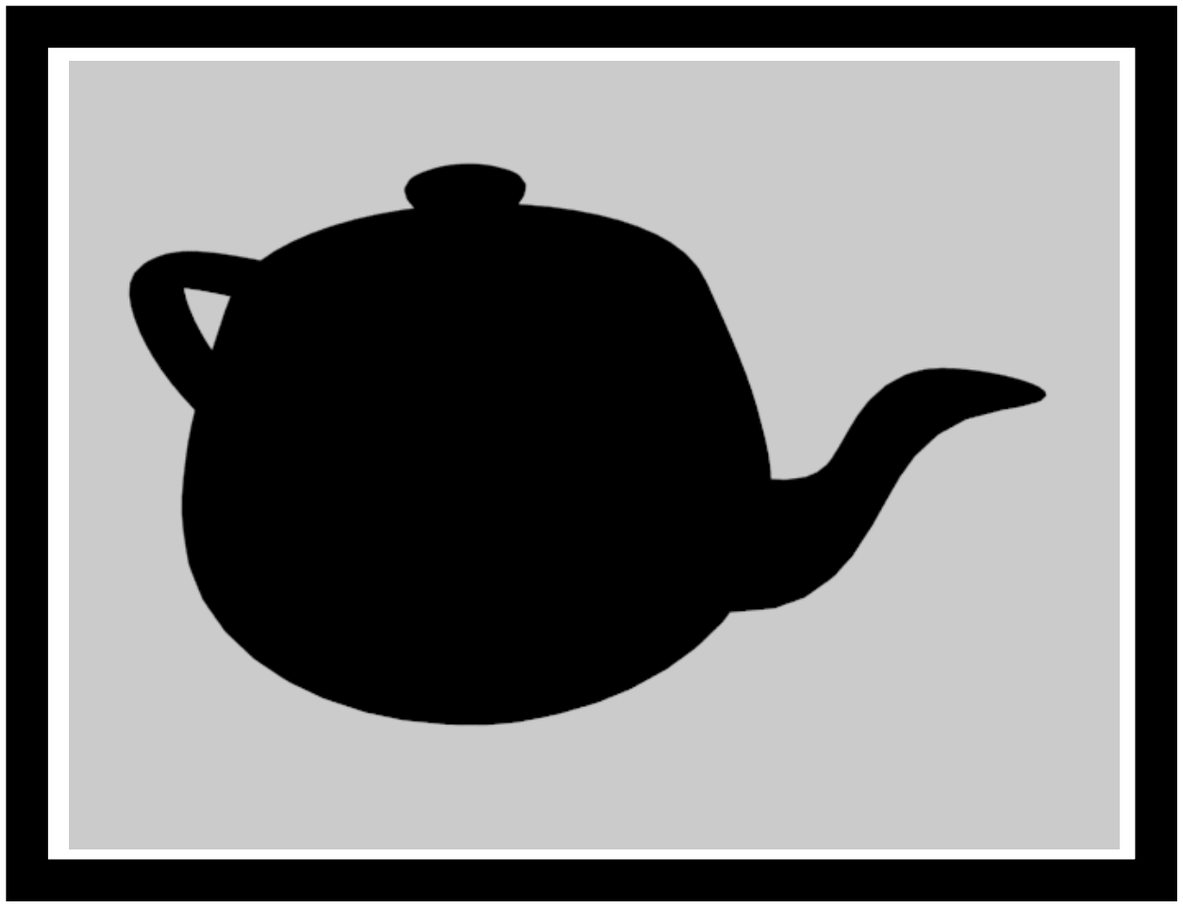
IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

ARLAB

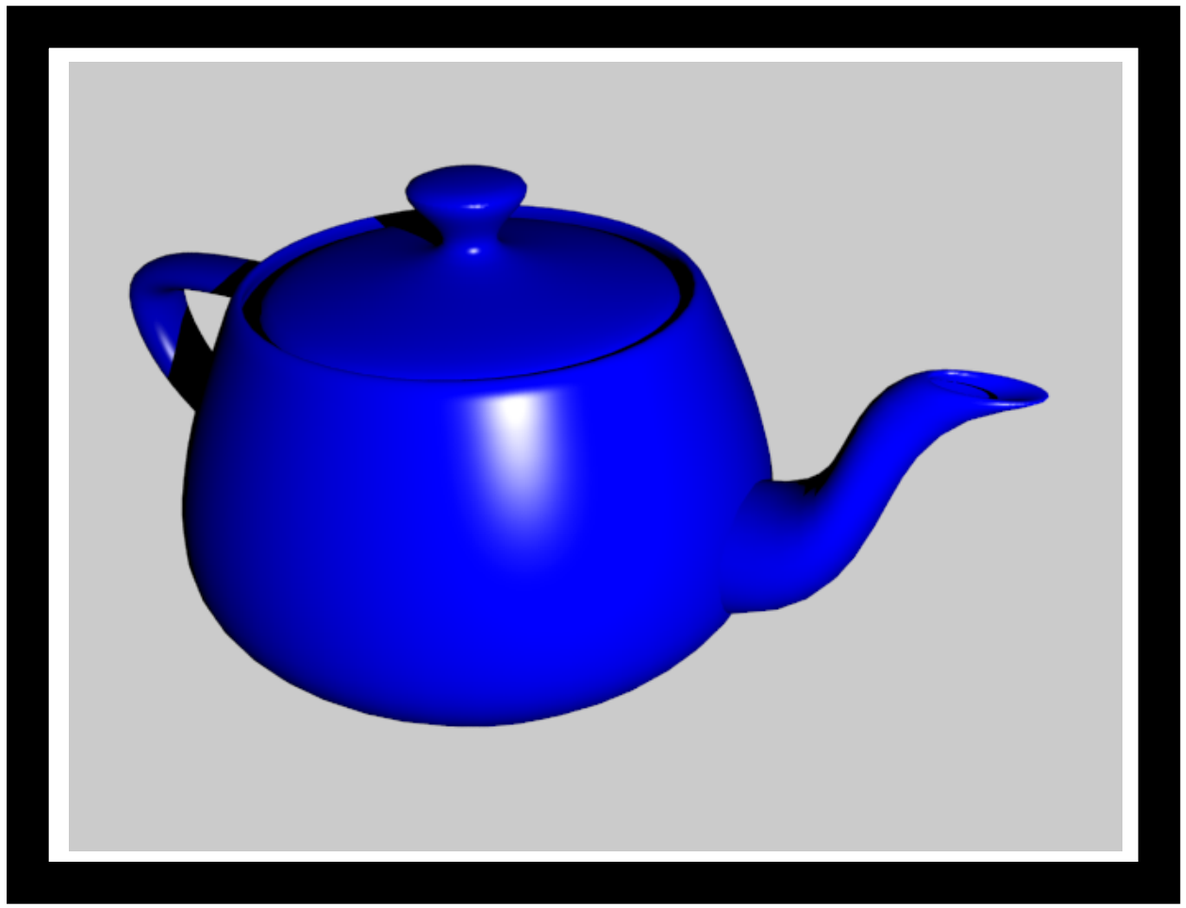
Why do we need light?

Why do we need light?

ARLAB



3D model without light simulation



3D model with light simulation

Content

ARLAB

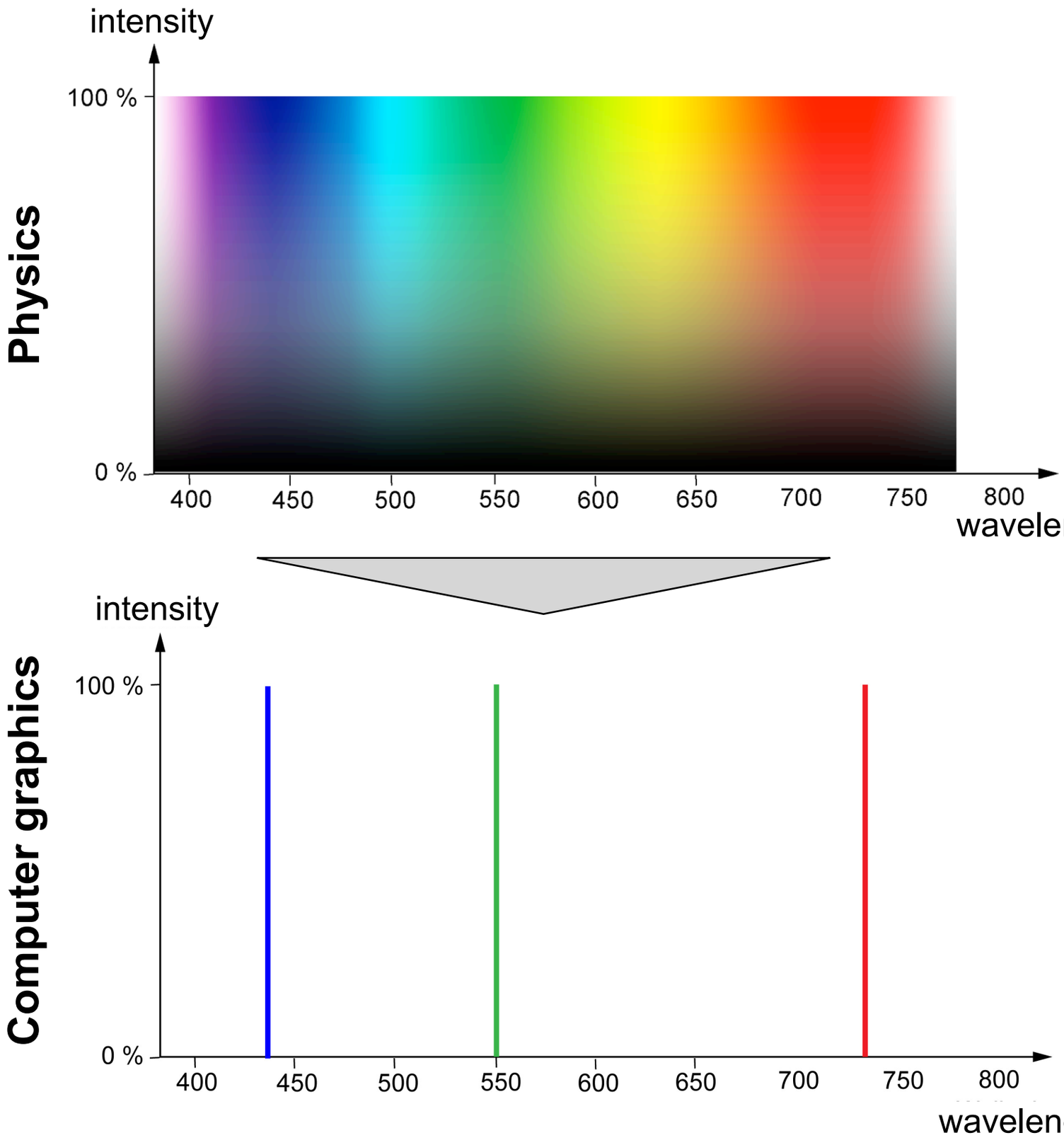
- Basics of Light and Reflections
- Light and Material in OpenGL
- Code Example
- Shading

Learning goals:

- Understand the illumination process in OpenGL
- Be able to create light sources
- Be able to apply materials to objects
- Understand and setup shading

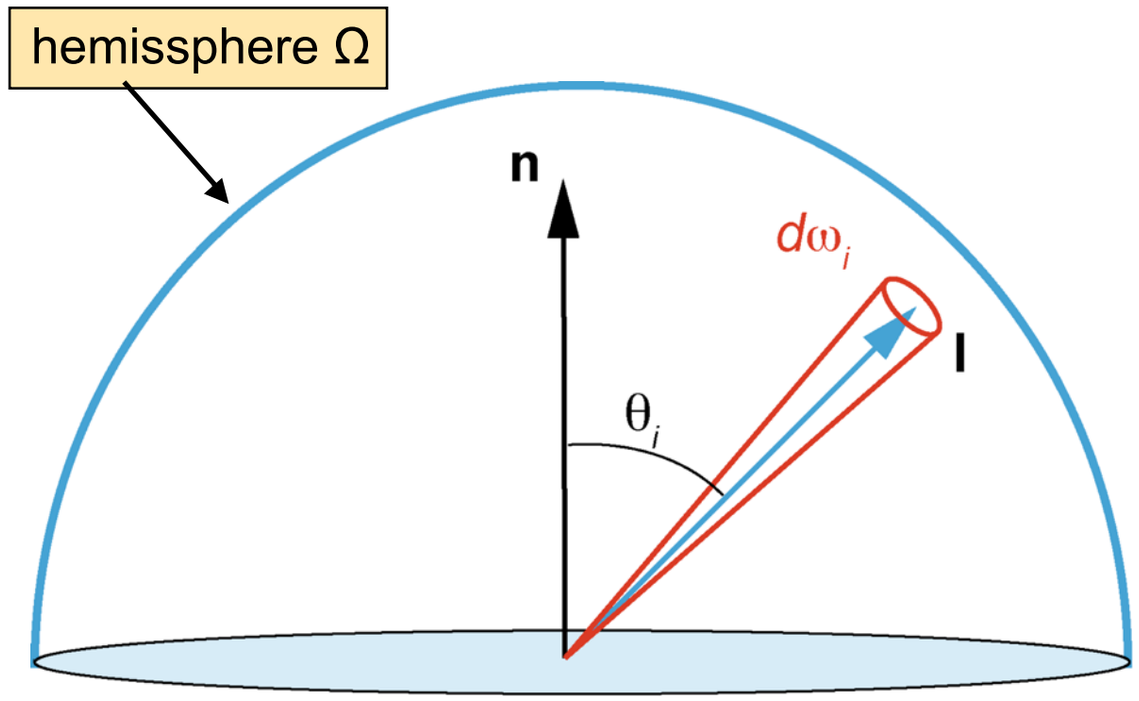
RGB Color

ARLAB



In computer graphics, all colors are simulated combining the three basic colors red, green, and blue:

RGB = [0…1, 0…1, 0...1]



The Rendering Equation

All the reflection models rely on the physics of light, in particular, on the light energy L_o on a surface point o

$$L_o(\mathbf{v}) = \int_\omega f(\mathbf{l}, \mathbf{v}) \otimes L_i(\mathbf{l}) \cdot cos(\theta_i) d\omega$$

This equation cannot be solved!

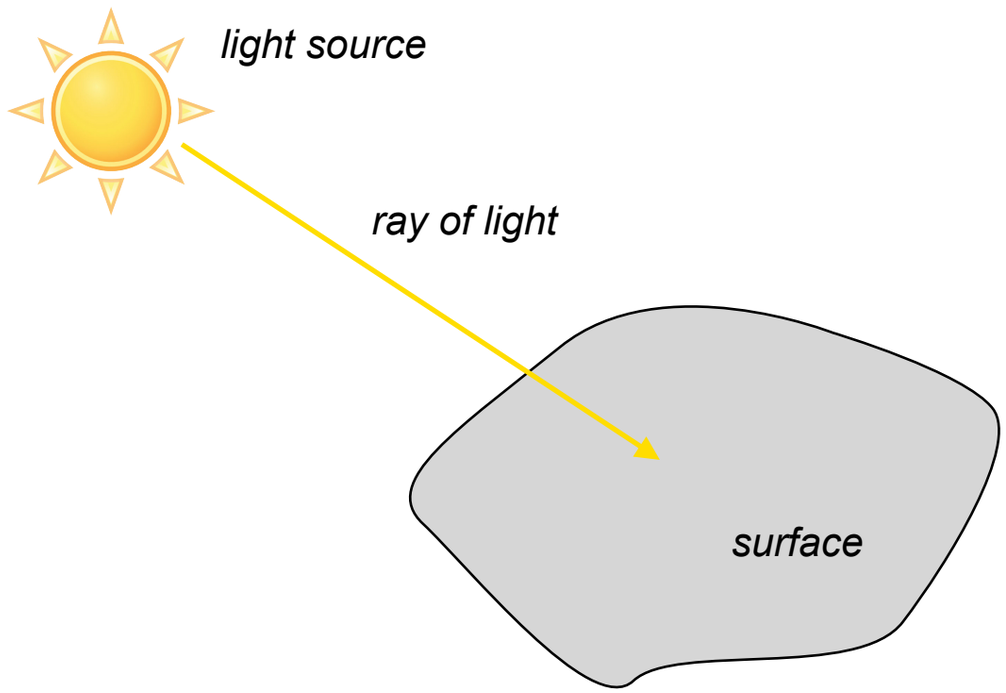
We solve an approximation instead.

$$L_o(\mathbf{v}) \approx f(\mathbf{l}, \mathbf{v}) \otimes E_L \cdot cos(\theta_{i_L})$$

Global vs. Local Lighting

ARLAB

Local Lighting

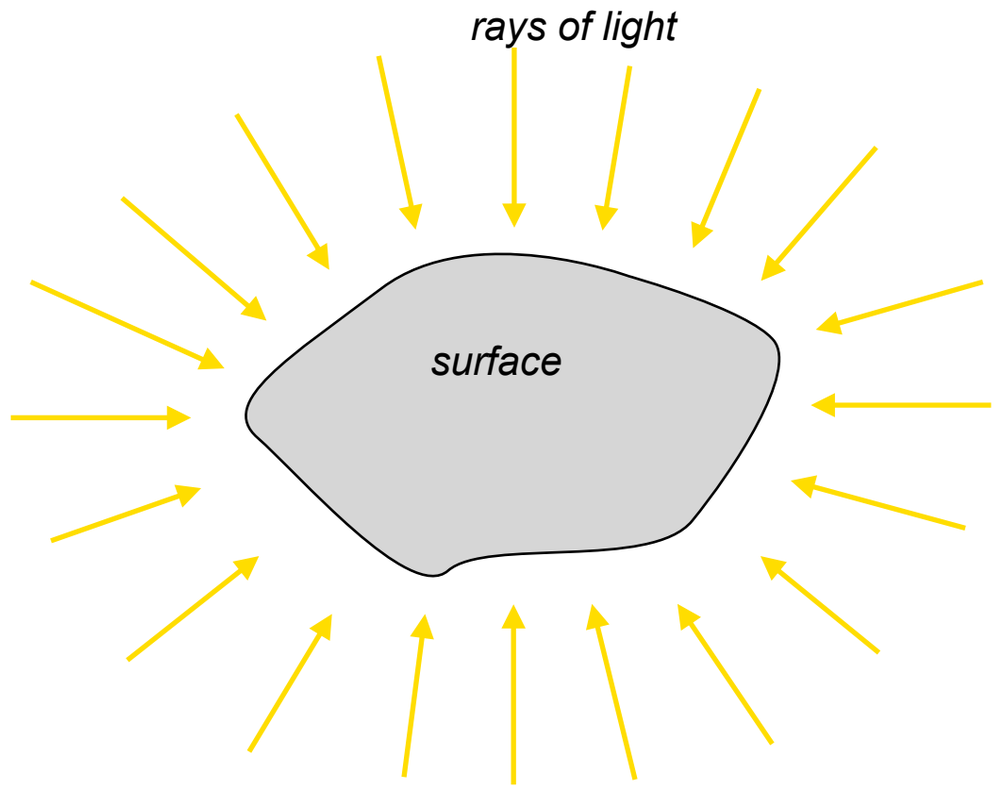


A local lighting model simulates the illumination of a surface by only considering light from explicitly specified light sources.

Methods:

- Phong lighting / reflection model

Global Lighting



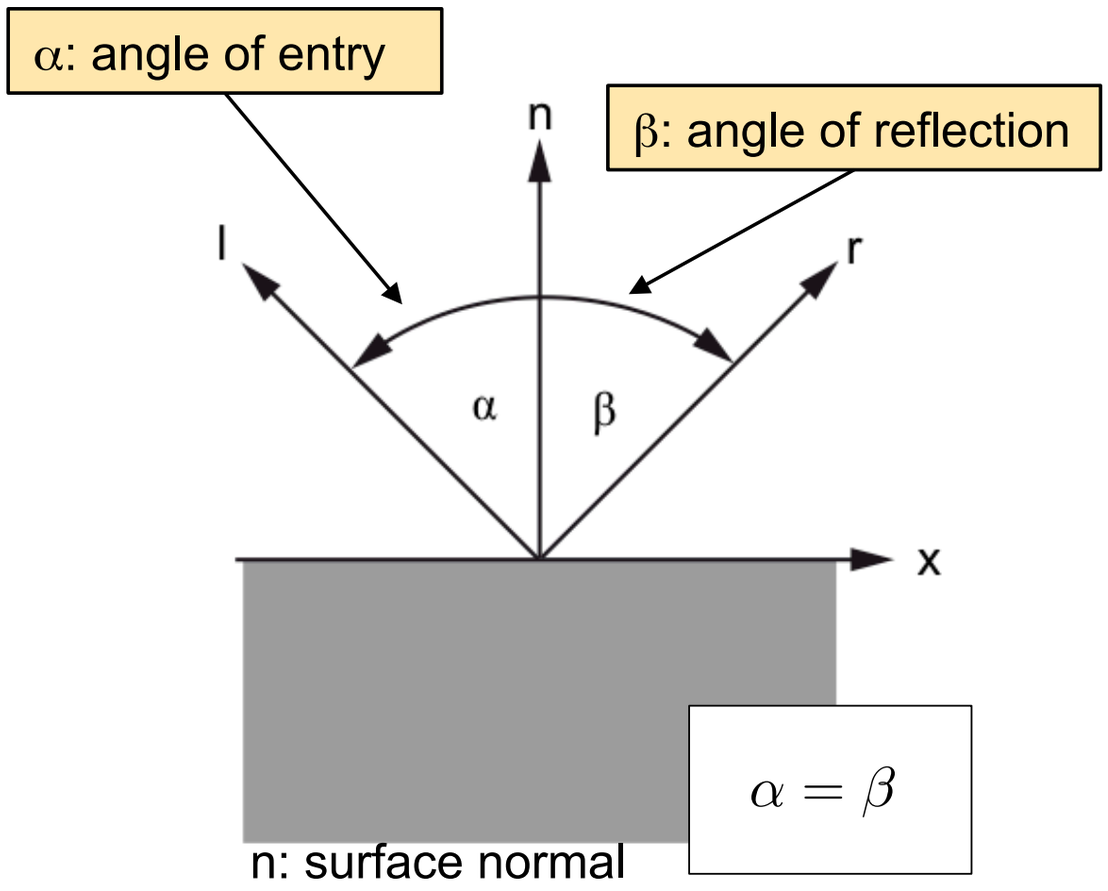
A global lighting model simulates the illumination of a surface by considering all incoming light rays regardless of their origin.

Methods:

- Raytraycing
- Radiosity

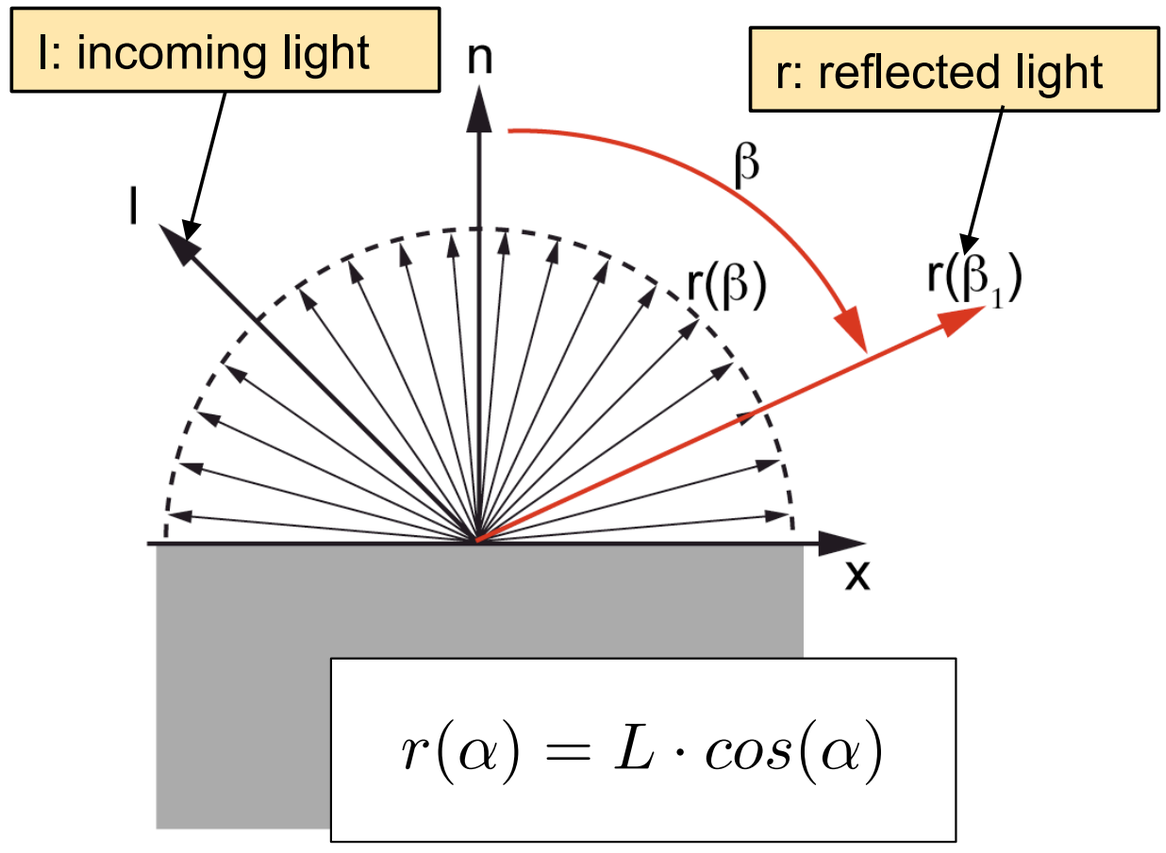
Physics of light

Ideal reflection law



$$\alpha = \beta$$

Lambert's cosine law



$$r(\alpha) = L \cdot cos(\alpha)$$

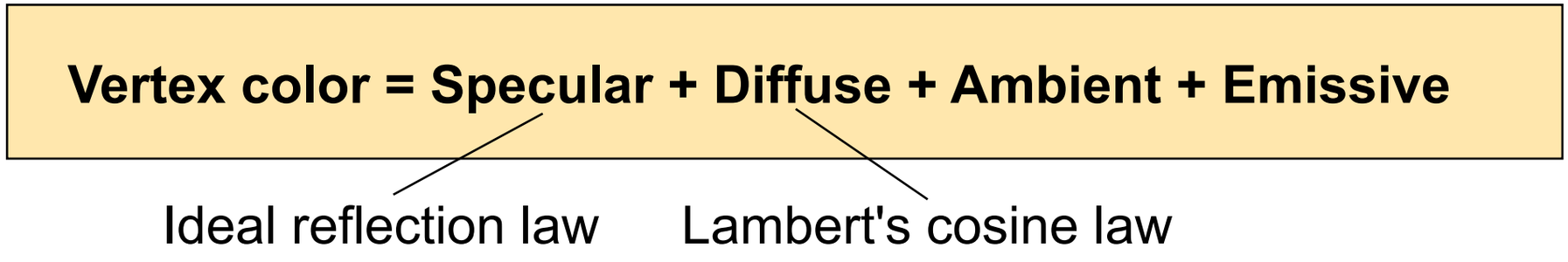
The color of an object is the perception of reflected light.

The character of light reflected from the surface of an object depends on:

- light source: composition, direction, geometry
- surface: orientation, material properties

There is always a material model and a light model!

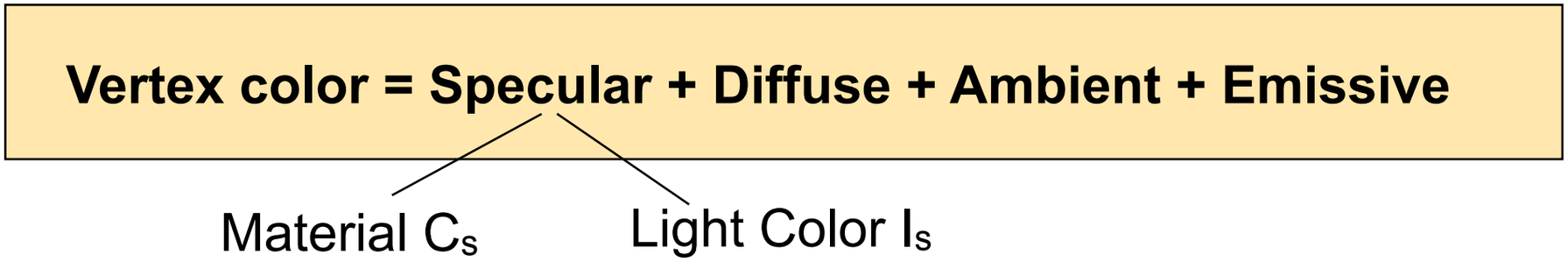
The OpenGL Standard Lighting Model



The OpenGL Standard Lighting Model (or sometimes the Phong Reflection Model) incorporates four reflection components:

- **Specular Reflection**: Light that comes from a particular direction is reflected into one particular direction. Light reflected on a mirror, shiny metals are examples.
- **Diffuse Reflection**: Light that comes from one direction is scattered equally in all directions so it appears equally bright no matter where the eye is located. Light reflected on a sheet of paper is an example.
- **Ambient Reflection**: Reflection of undirected light. It simulates global illumination models; light that is reflected i.e. from walls.
- **Emissive Reflection / Light**: Simulation of light originating from an object. E.g., Lamp, sun, etc.

Material and Light



A reflection model includes three entities:

- a material: the material is defined as RGB color value C as input.
- a light color: the light color is defines as RGB color value I as input.
- several vectors which depend on the particular reflection model.

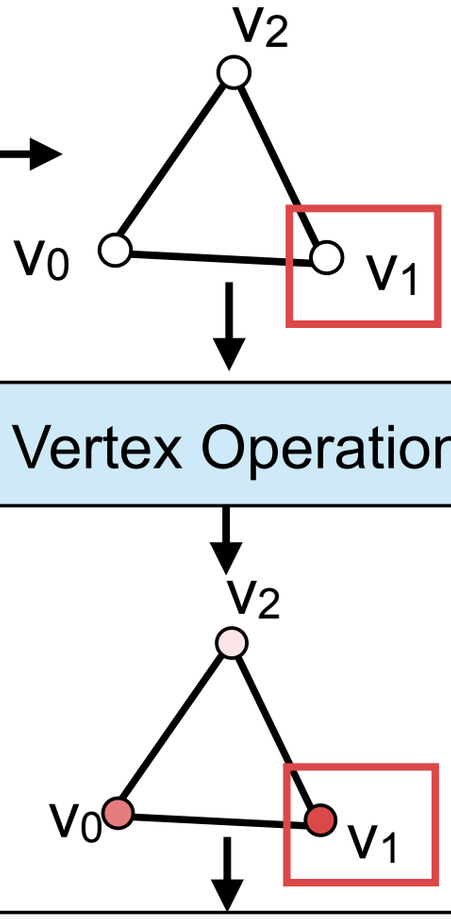
The output is

- a color value

Rendering: from model to pixel

ARLAB

Display List of a 3D model



- Calculate vertex colors
- Primitive Assembly
- Shading: fills the primitive (e.g., polygon, quad, etc.) with color

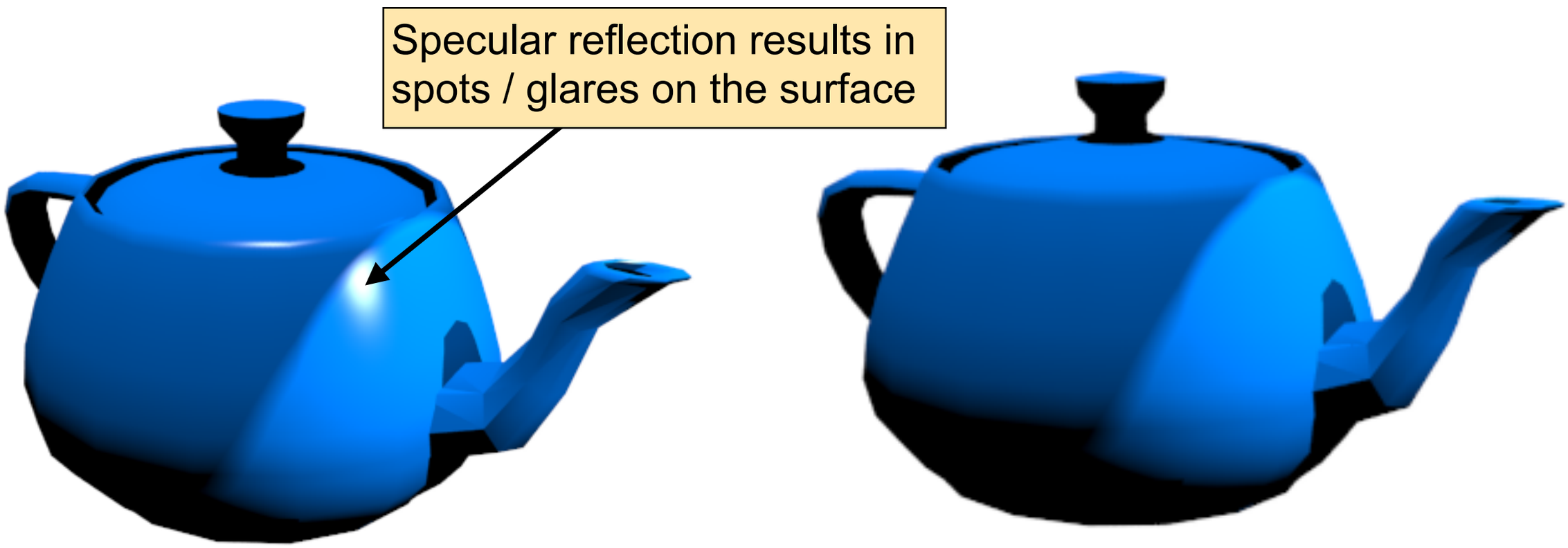
Each primitive (point, line, polygon, quad etc.) is processed individually.

Image data in Frame Buffer

Specular Reflection

ARLAB

A spotlight source and a directional light illuminate this object

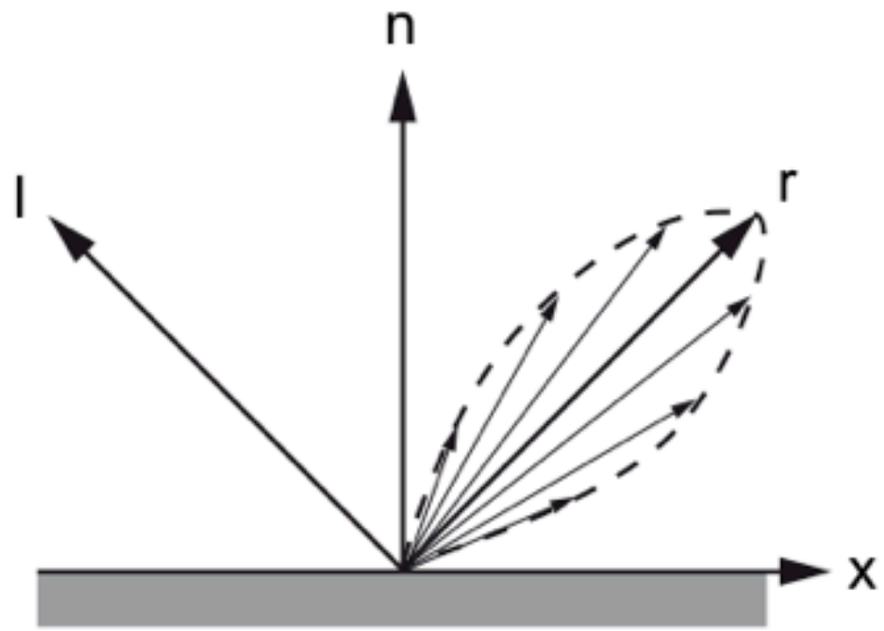


with specular reflection

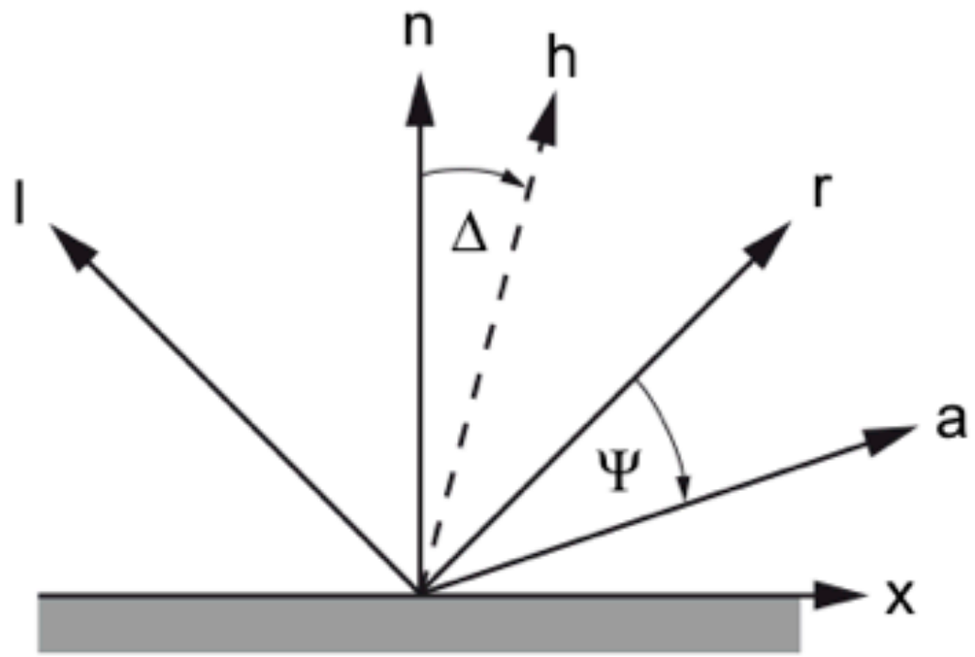
without specular reflection

Specular Reflection

ARLAB



(a)



(b)

The specular reflection component

$$specular = (\max(\boldsymbol{h}\cdot\boldsymbol{n},0)^{S})\cdot I_S\cdot C_S$$

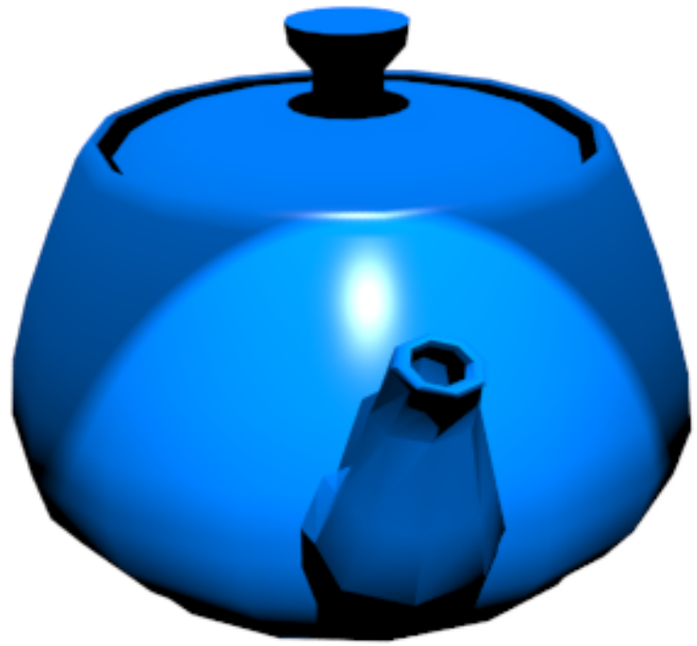
$$= \max(\boldsymbol{h}\cdot\boldsymbol{n},0)^{S}\cdot\begin{pmatrix} R_{L,S}\cdot R_{C,S} \\ G_{L,S}\cdot G_{C,S} \\ B_{L,S}\cdot B_{C,S} \\ A_{L,S}\cdot A_{C,S} \end{pmatrix}$$

half-way vector:

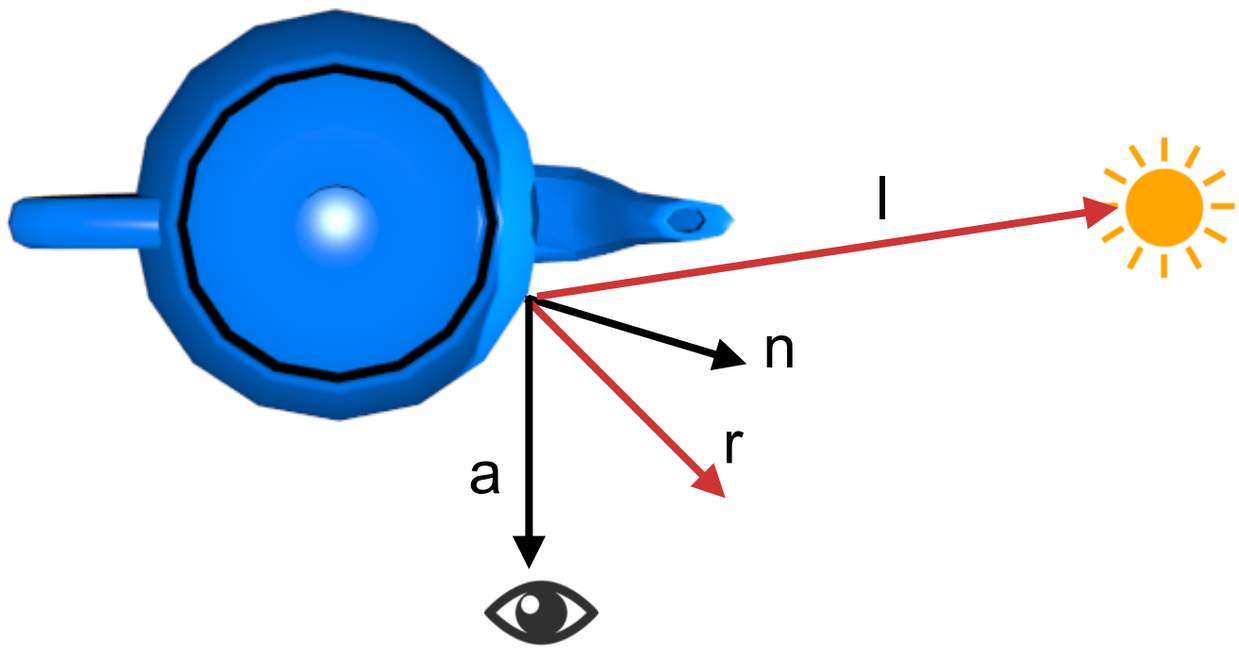
$$\boldsymbol{h} = \frac{(\boldsymbol{l} + \boldsymbol{a})}{|\boldsymbol{l}+\boldsymbol{a}|}$$

l : Specular light direction vector,
I_S: Specular light color
n: Normal,
a: Eye-point
r: reflection vector
h: Halfway Vector
S: Specular reflection coefficient
C_s:Specular material color

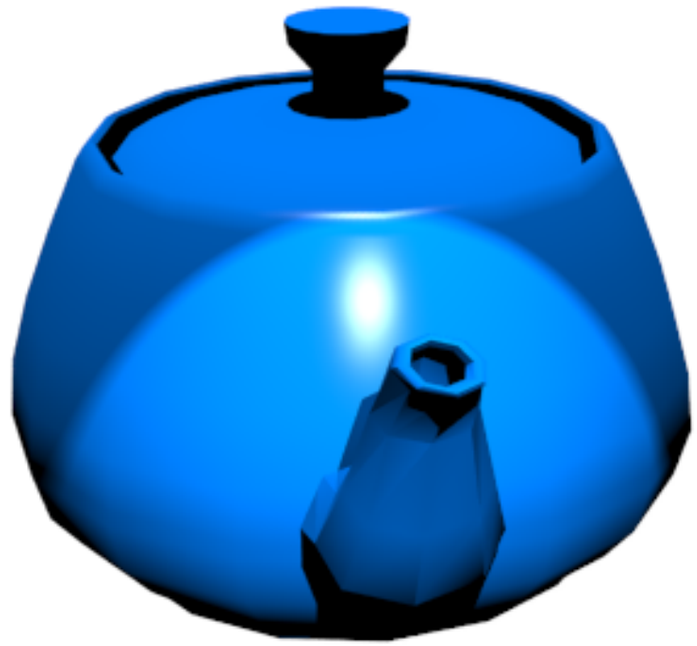
Example



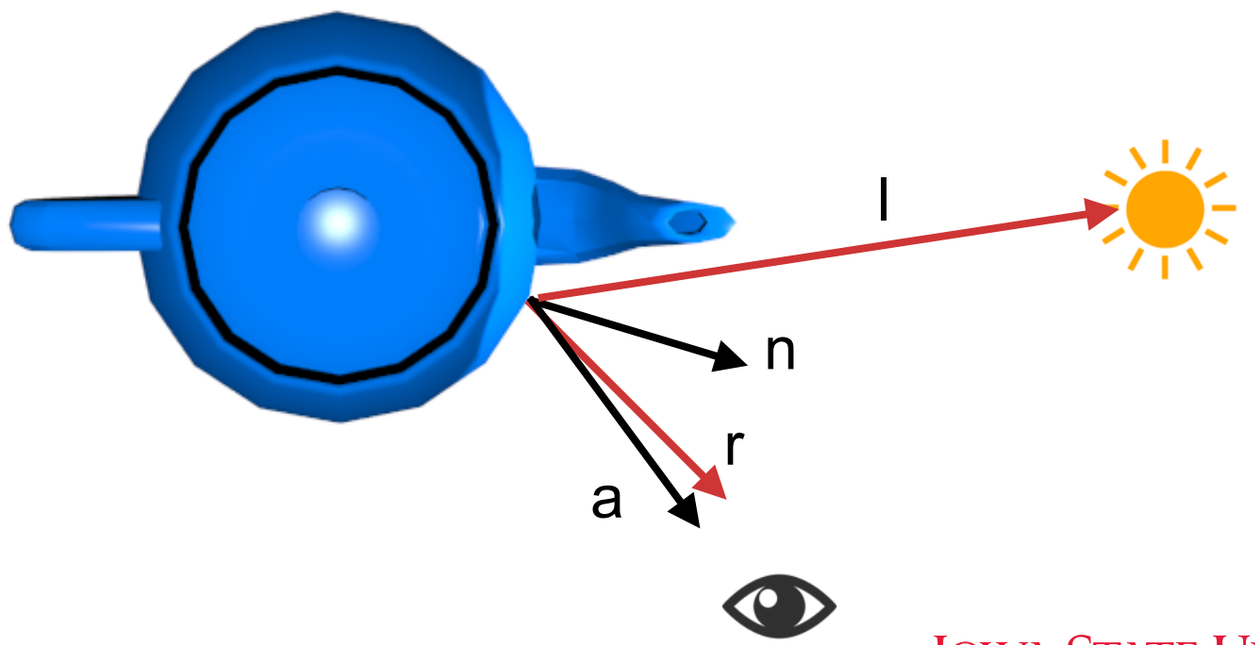
A large angle between eye point and reflection vector results in dim spot.



Example



If the angle is small, the hotspot is bright.

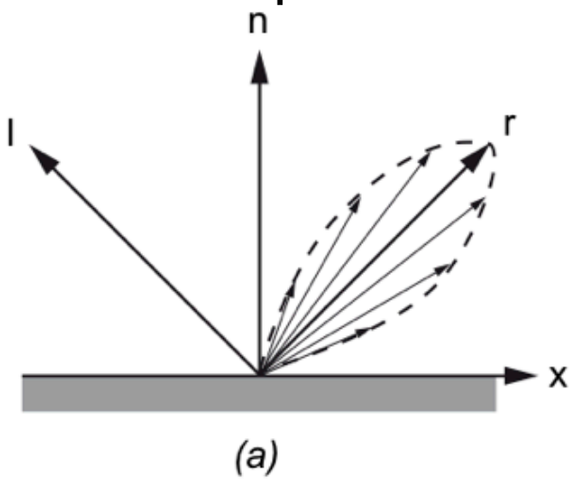


But the angle is not the parameter that controls the size of this spot.

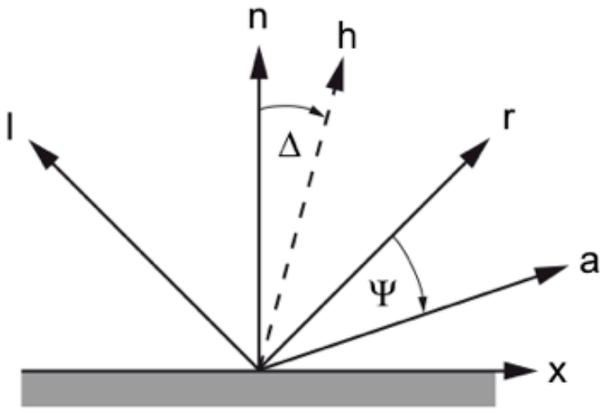
Specular Reflection (2/3)

ARLAB

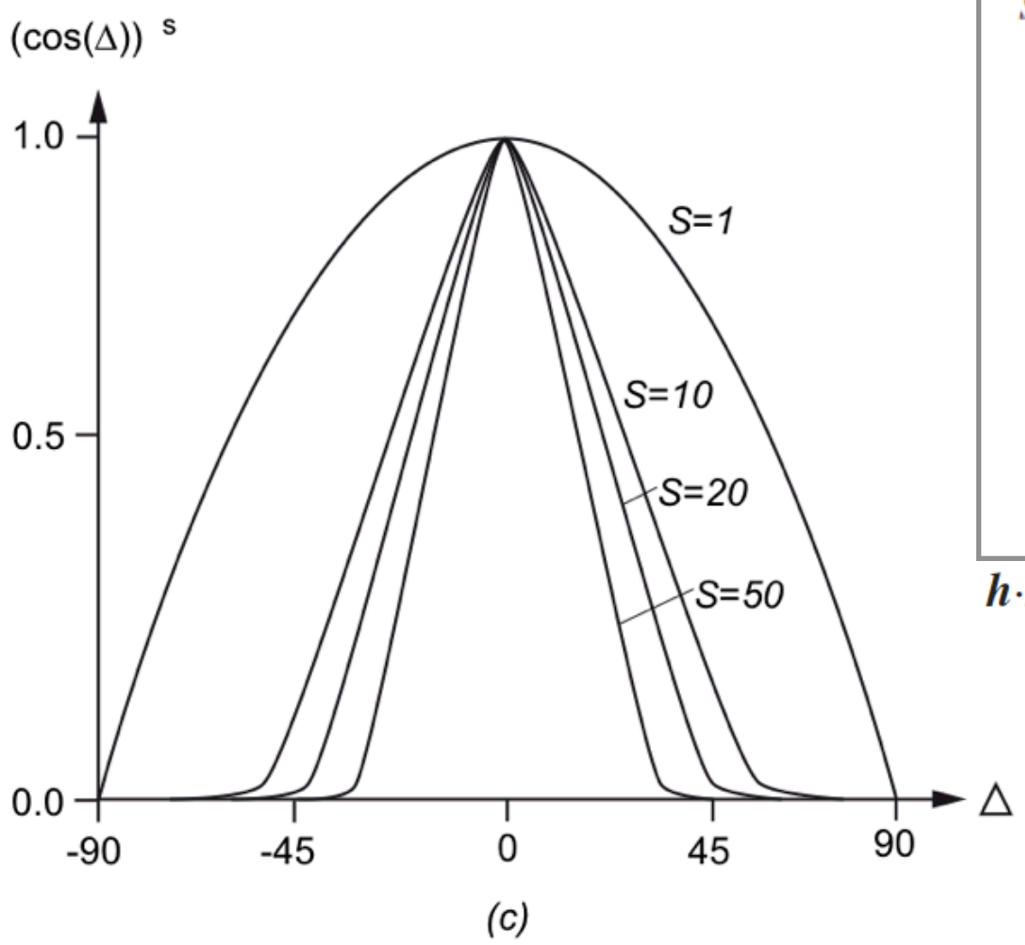
Shininess parameter s to alter the size of the reflection.



(a)



(b)



(c)

$$specular = (\max(\boldsymbol{h}\cdot\boldsymbol{n},0)^{S})\cdot I_S\cdot C_S$$

$$= \max(\boldsymbol{h}\cdot\boldsymbol{n},0)^{S}\cdot\begin{pmatrix} R_{L,S}\cdot R_{C,S} \\ G_{L,S}\cdot G_{C,S} \\ B_{L,S}\cdot B_{C,S} \\ A_{L,S}\cdot A_{C,S} \end{pmatrix}$$

$$\boldsymbol{h} = \frac{(\boldsymbol{l} + \boldsymbol{a})}{|\boldsymbol{l}+\boldsymbol{a}|}$$

$\boldsymbol{h}\cdot\boldsymbol{n}$:= *dot product*

(d1) S=10

(d2) S=20

(d3) S=50

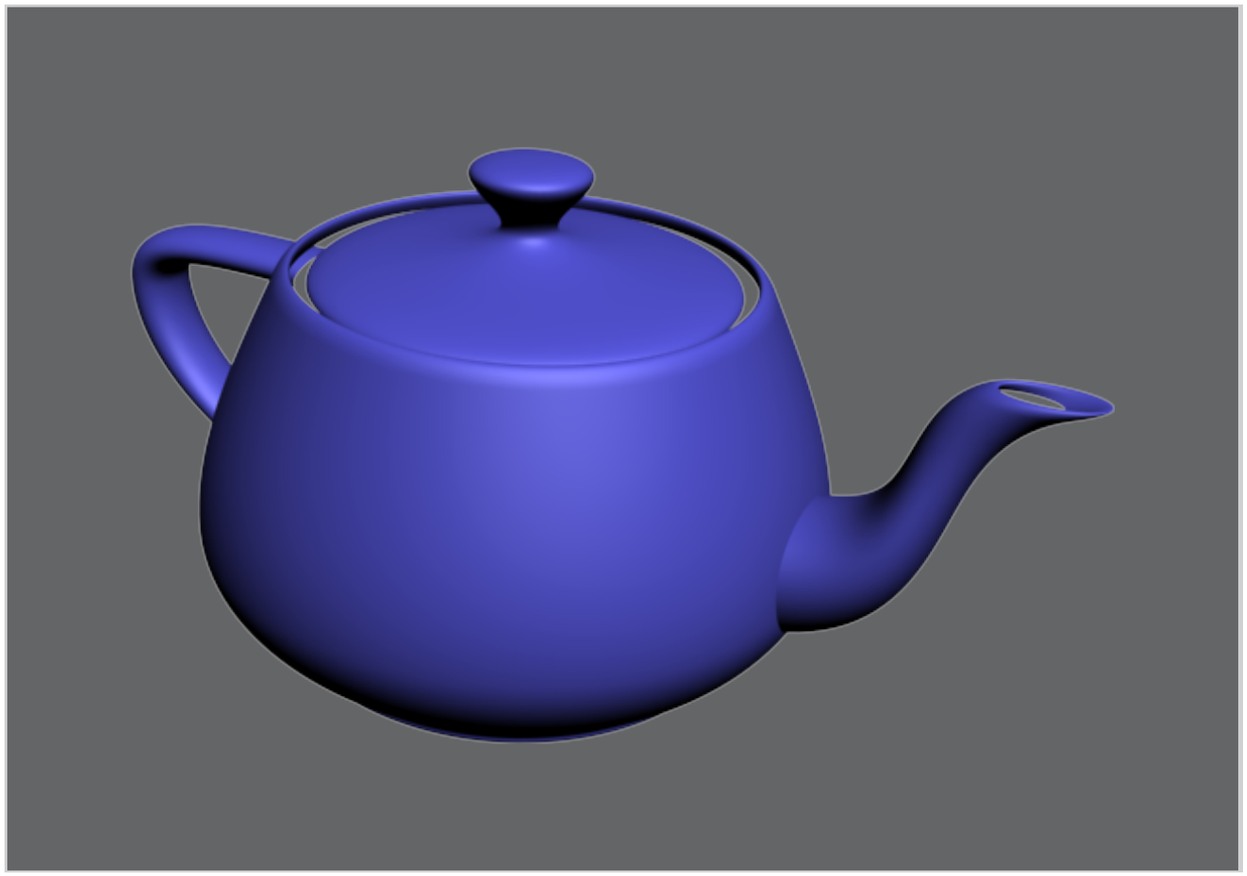
l : Specular light direction vector,
I_S: Specular light color
n: Normal,
a: Eye-point
r: reflection vector
h: Halfway Vector
S: Specular reflection coefficient
C_s:Specular material color

Specular Reflection (3/3)

- Creates highlights and glares on surfaces.
- Needs curved surfaces; highlights on even surfaces are difficult to create
- Does not affect the entire 3D model
- Depends on the position of the viewer

Diffuse Reflection (1/2)

ARLAB



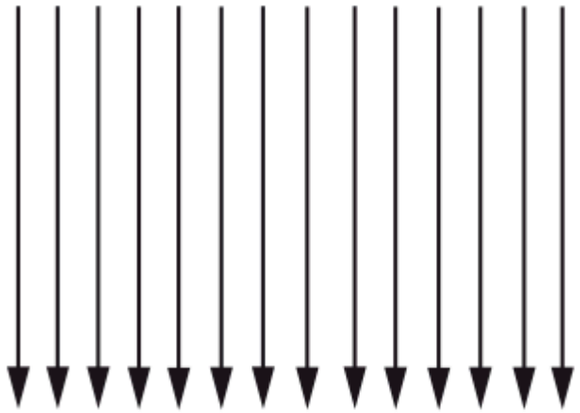
Diffuse reflection cause a consistent illumination of the model.

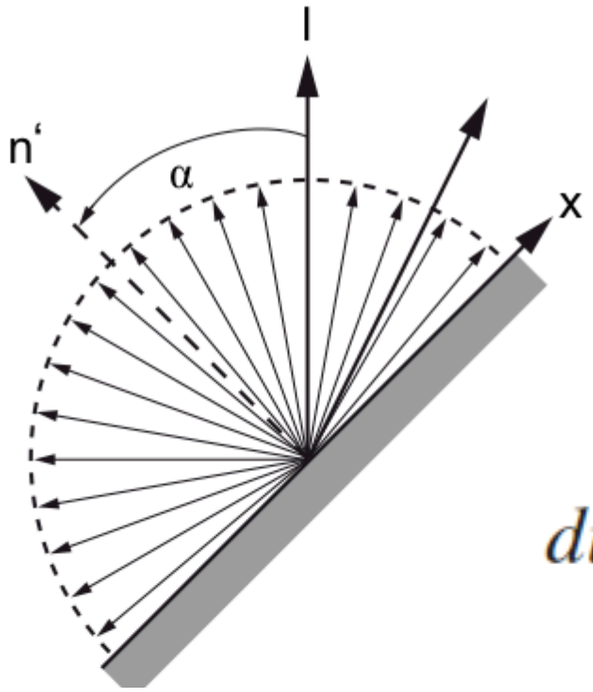
- Is affected by the color of the incident diffuse light and the angle of the incident light relative to the normal direction
- Plays the most important role for the perception of shapes
- Viewer position doesn’t affect diffuse reflection

Diffuse Reflection (2/2)

ARLAB

parallel incoming light

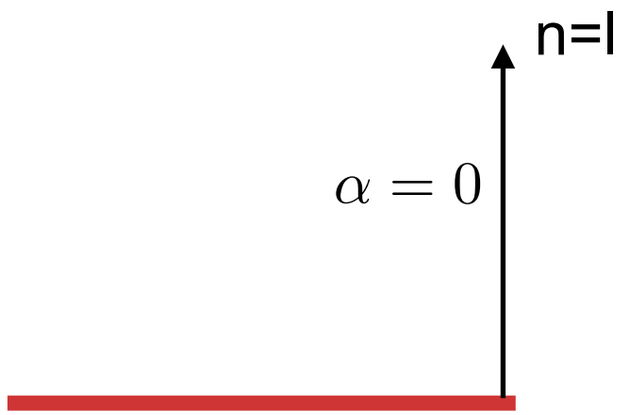


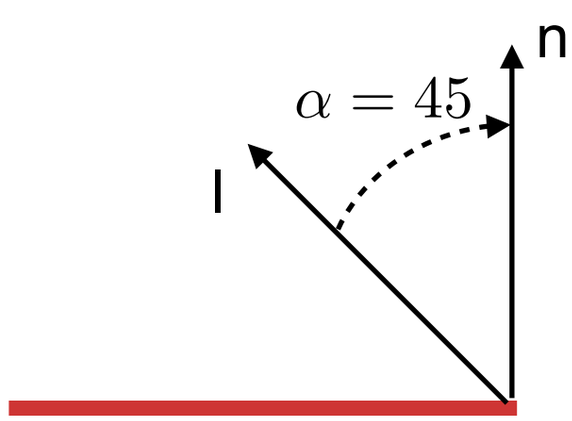


The incoming light I is scattered into all directions.

$$diffus = \max(\boldsymbol{l}\cdot\boldsymbol{n},0)\cdot I_D\cdot C_D$$

$$= \max(\boldsymbol{l}\cdot\boldsymbol{n},0)\cdot\begin{pmatrix} R_{L,D}\cdot R_{C,D} \\ G_{L,D}\cdot G_{C,D} \\ B_{L,D}\cdot B_{C,D} \\ A_{L,D}\cdot A_{C,D} \end{pmatrix}$$

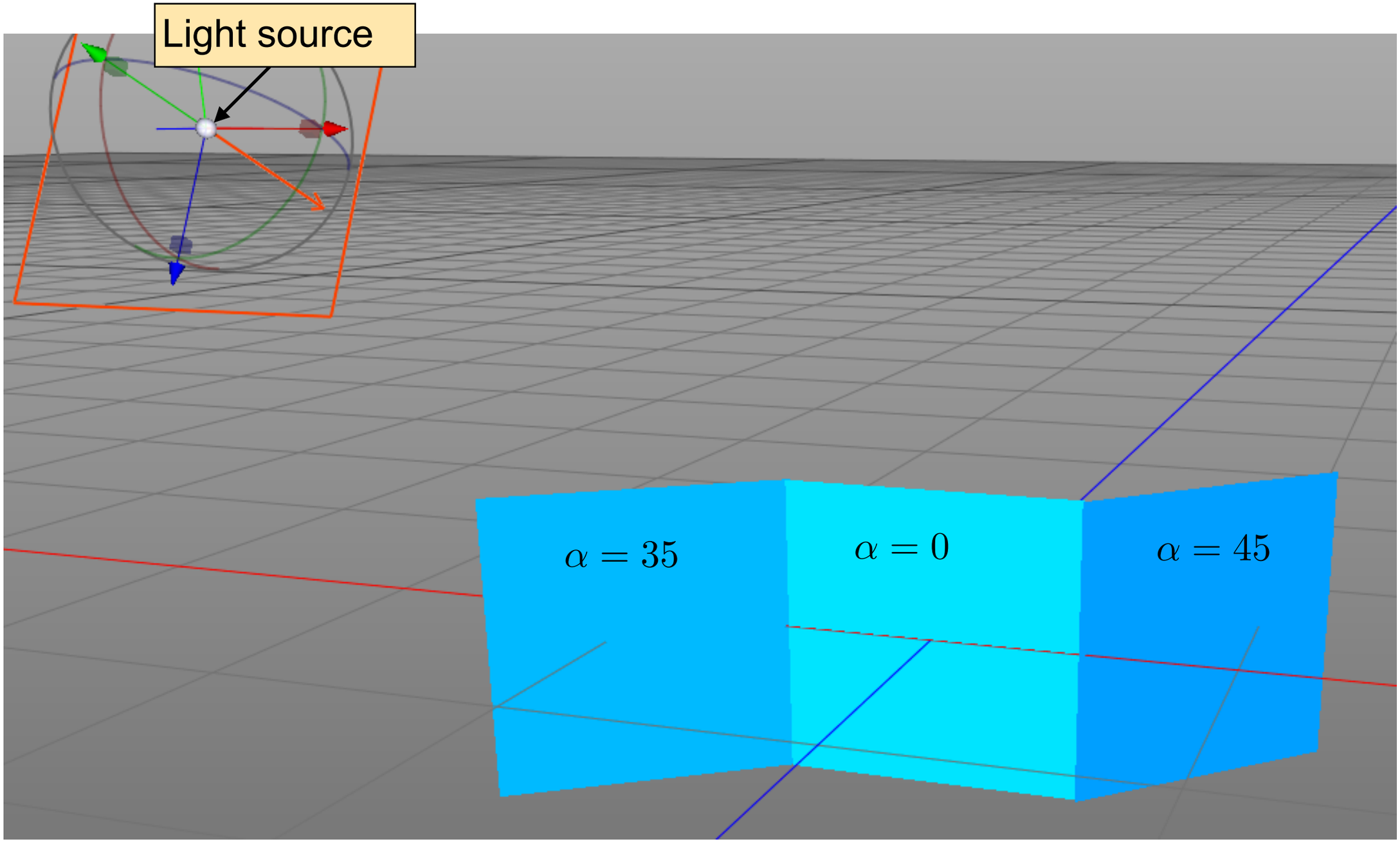




$\boldsymbol{l}\cdot\boldsymbol{n}$:= *dot product*

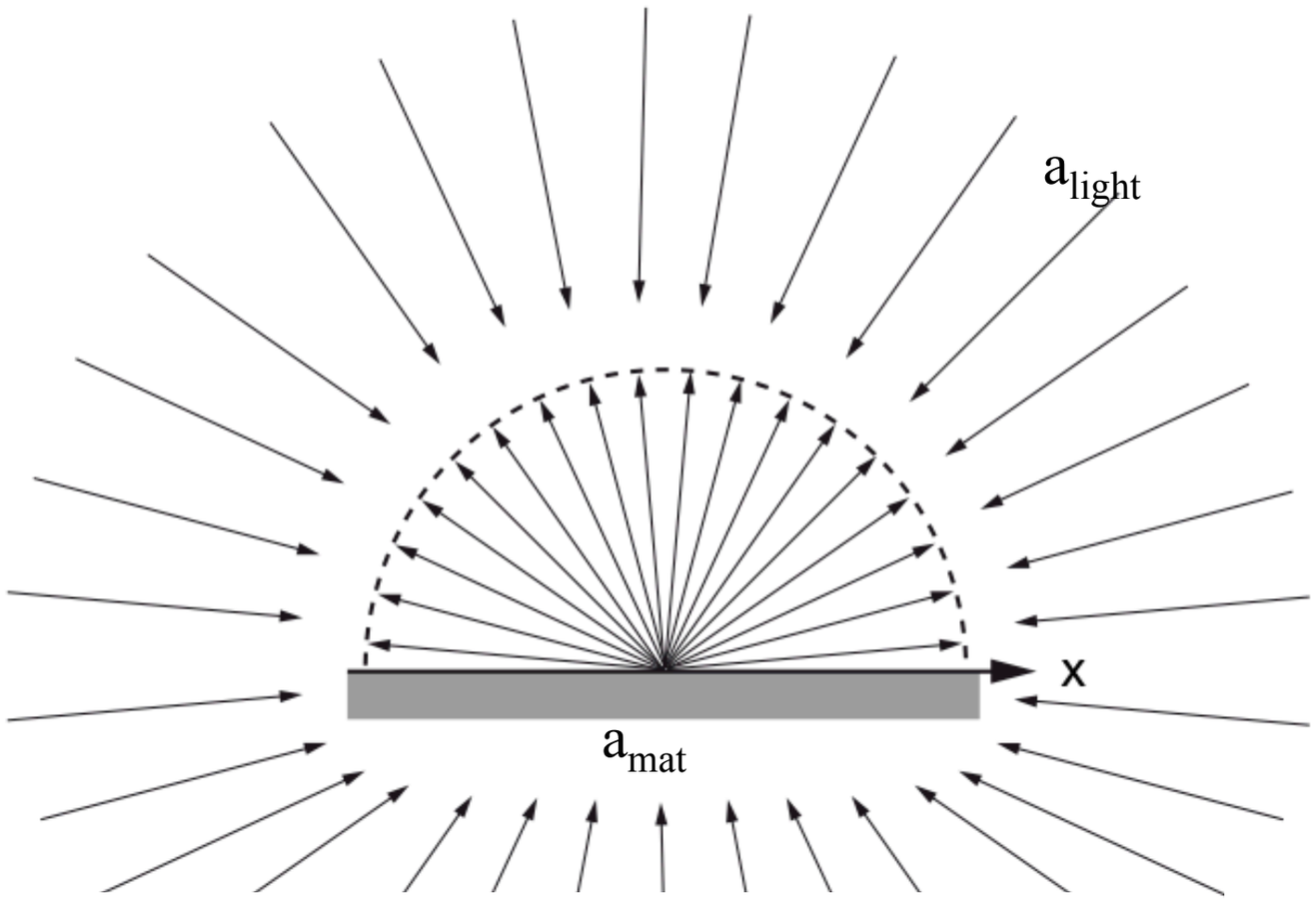
l: Diffuse light direction vector,
I_D: Diffuse light color
n: Normal,
C_D: Diffuse material color

Example



Ambient Reflection (1/2)

ARLAB



Ambient reflection emulates the effect of omnidirectional light shining on the surface.

A 3D model illuminated only by ambient light and reflect ambient light only, appears plain-colored on the screen.

$$ambient = a_{light} \cdot a_{mat} = \begin{pmatrix} R_{a_{light}} \\ G_{a_{light}} \\ B_{a_{light}} \\ A_{a_{light}} \end{pmatrix} \cdot \begin{pmatrix} R_{a_{mat}} \\ G_{a_{mat}} \\ B_{a_{mat}} \\ A_{a_{mat}} \end{pmatrix}$$

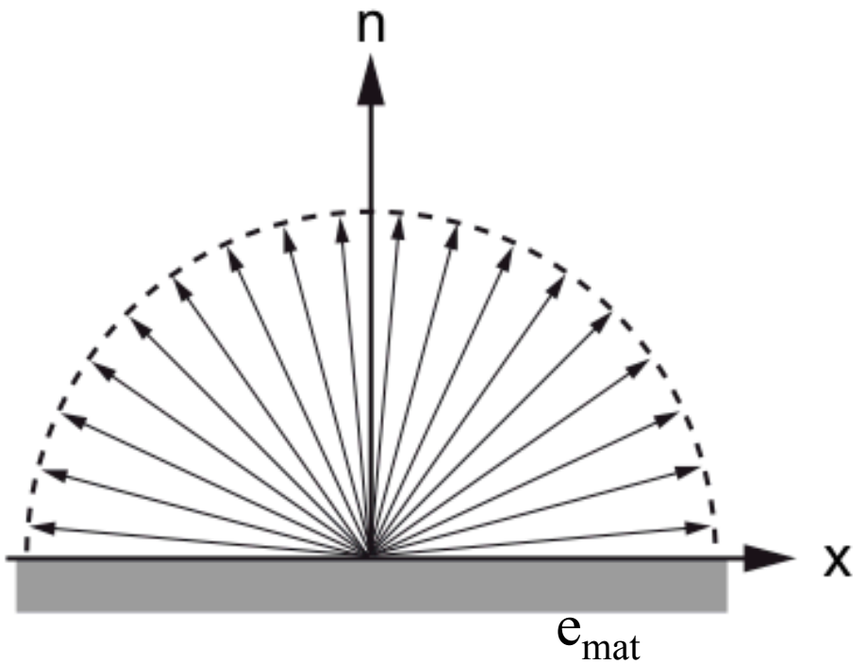
a_{light}: Ambient light color
a_{mat}: Ambient material color

Ambient Reflection (2/2)

ARLAB

- Affects the overall color of the object
- Is most noticeable where an object receives no direct light (e.g., at the visible edges)
- Total ambient reflectance is affected by global ambient light (reflection from all non-emissive objects) and the light from all light sources
- Does not depend on the viewer's position

Emissive Reflection / Light



Emissive light emits light equally in all direction.
e_{mat}: Emissive light color

An emissive illuminated model appears in the lighting color on screen; e.g., white.

$$e_{mat} = \begin{pmatrix} R_{e_{mat}} \\ G_{e_{mat}} \\ B_{e_{mat}} \\ A^*_{e_{mat}} \end{pmatrix}$$

*The alpha value A is not considered in this case

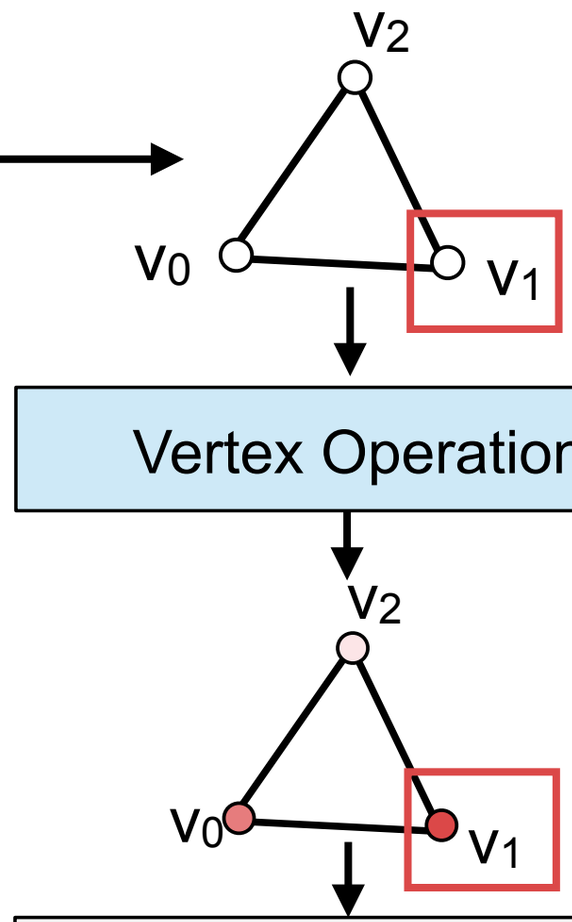
ARLAB

Shading

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Rendering: from model to pixel

Display List of a 3D model



- Calculate vertex colors
- Primitive Assembly

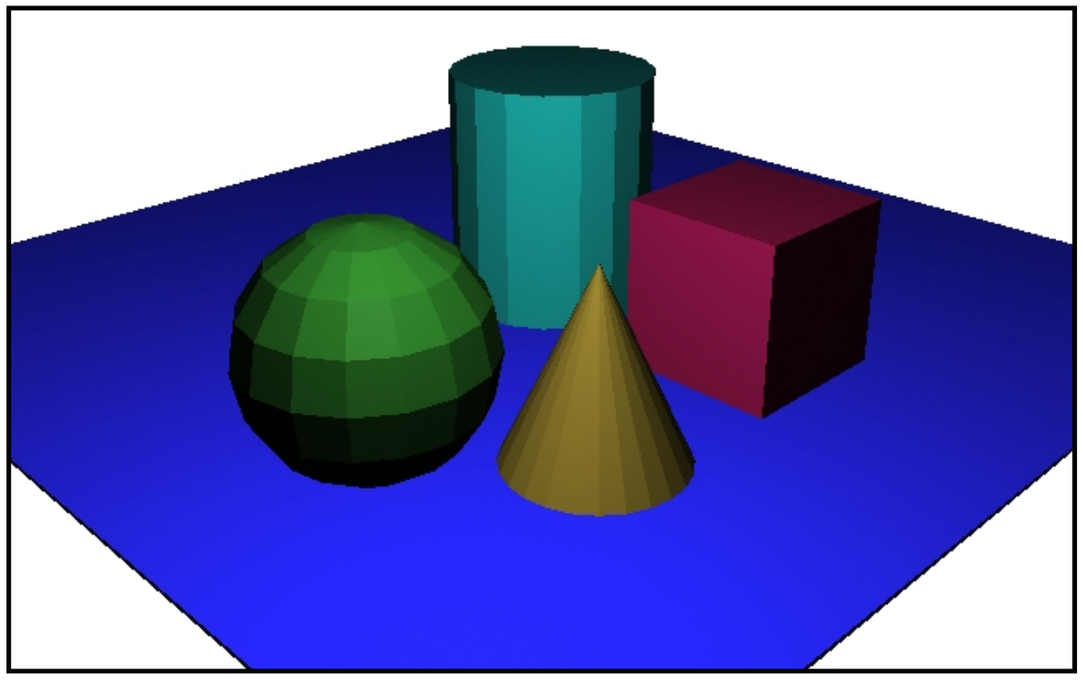
Each primitive (point, line, polygon, quad etc.) is processed individually.

- Shading: fills the primitive (e.g., polygon, quad, etc.) with color

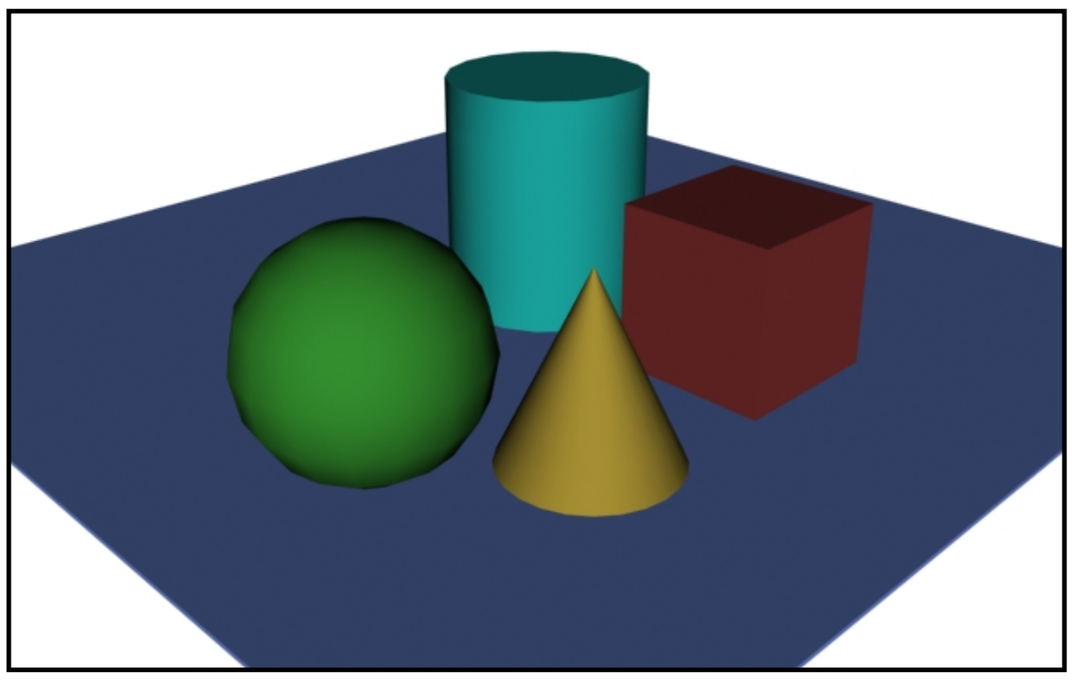
Image data in Frame Buffer

Shading

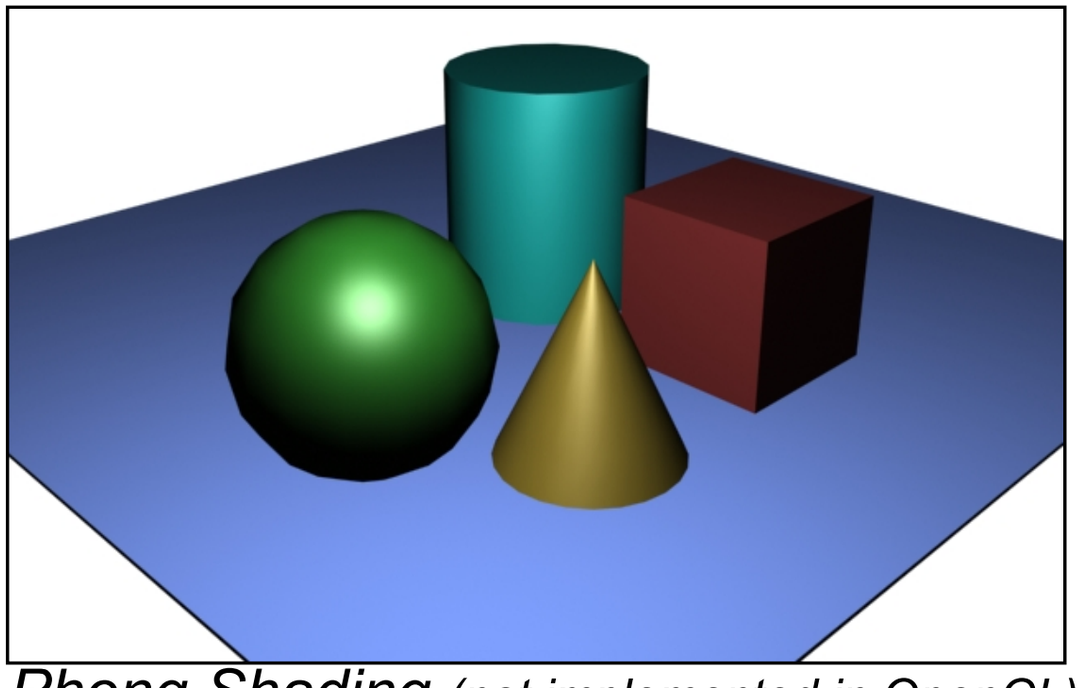
ARLAB



Flat Shading



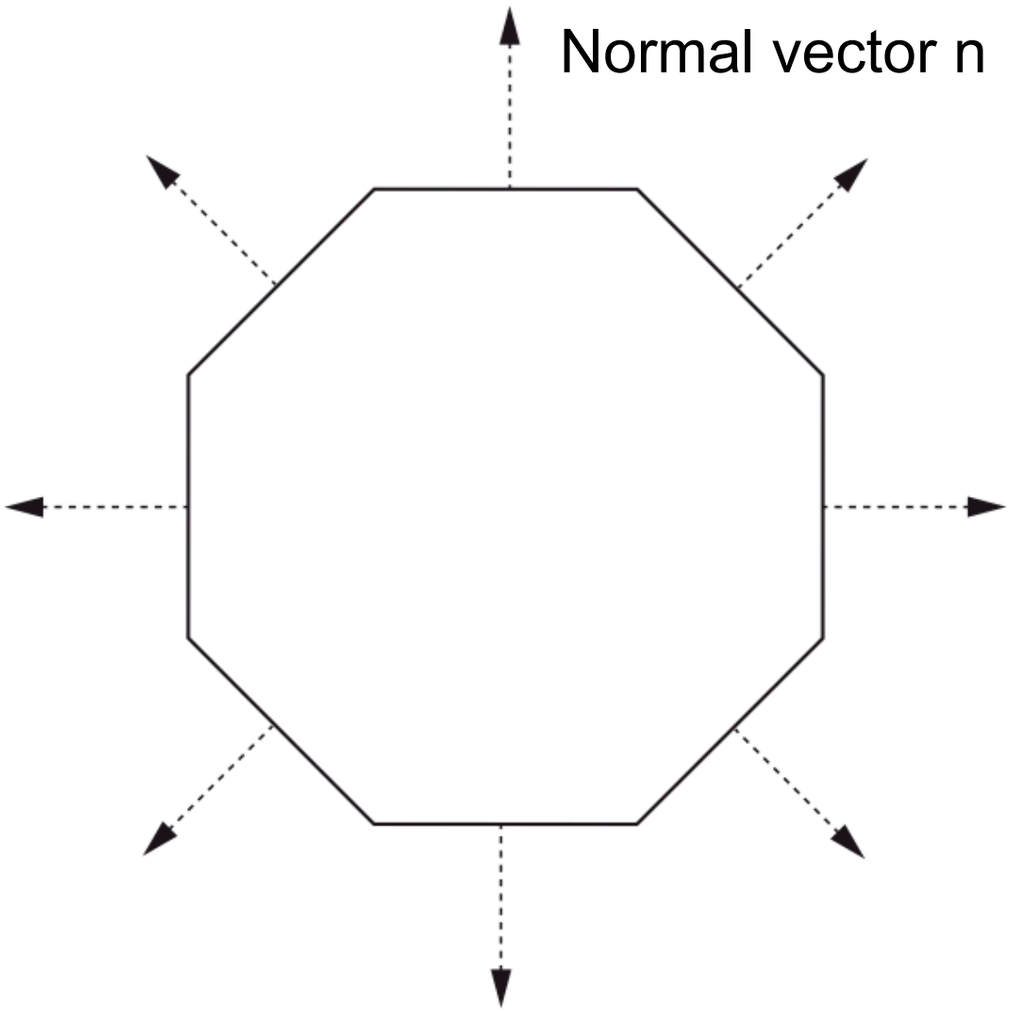
Gouraud Shading

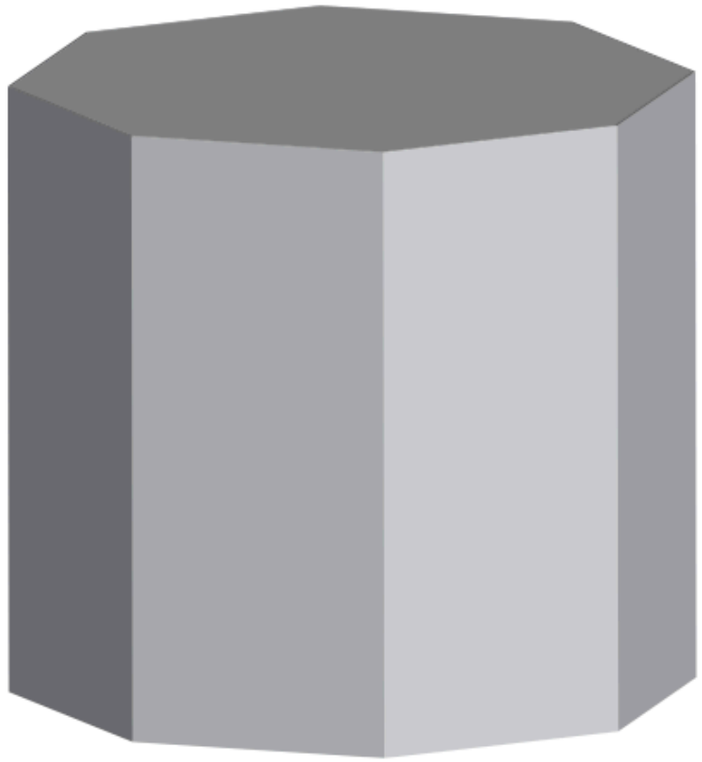


Phong Shading (not implemented in OpenGL)

Shading is the process that fills the primitives (e.g., polygons, triangles, quads etc.) with color. The task is to determine a color value for each pixel on screen.

Flat Shading

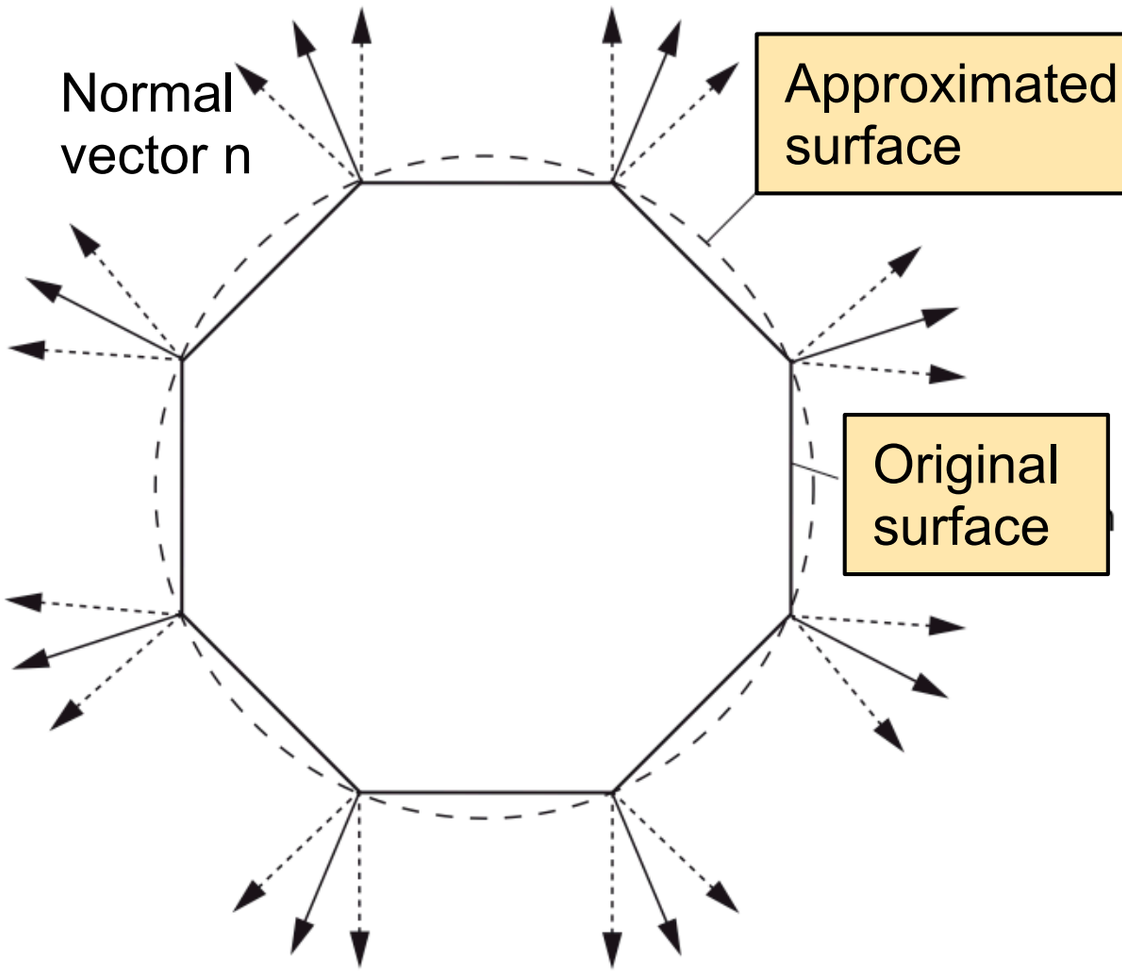


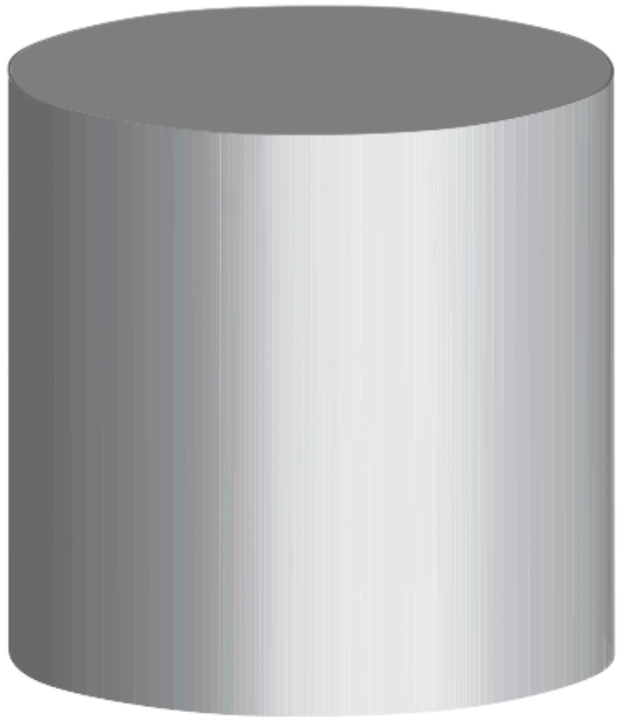


The Flat Shading method calculates one color for each surface with respect to the surface normal.

- Accentuates the individual polygons
- One color is assigned to each polygon
- Very fast
- Color is computed for one vertex and assigned to entire polygon
- Specular highlights are rendered poorly

Gouraud Shading (1/2)

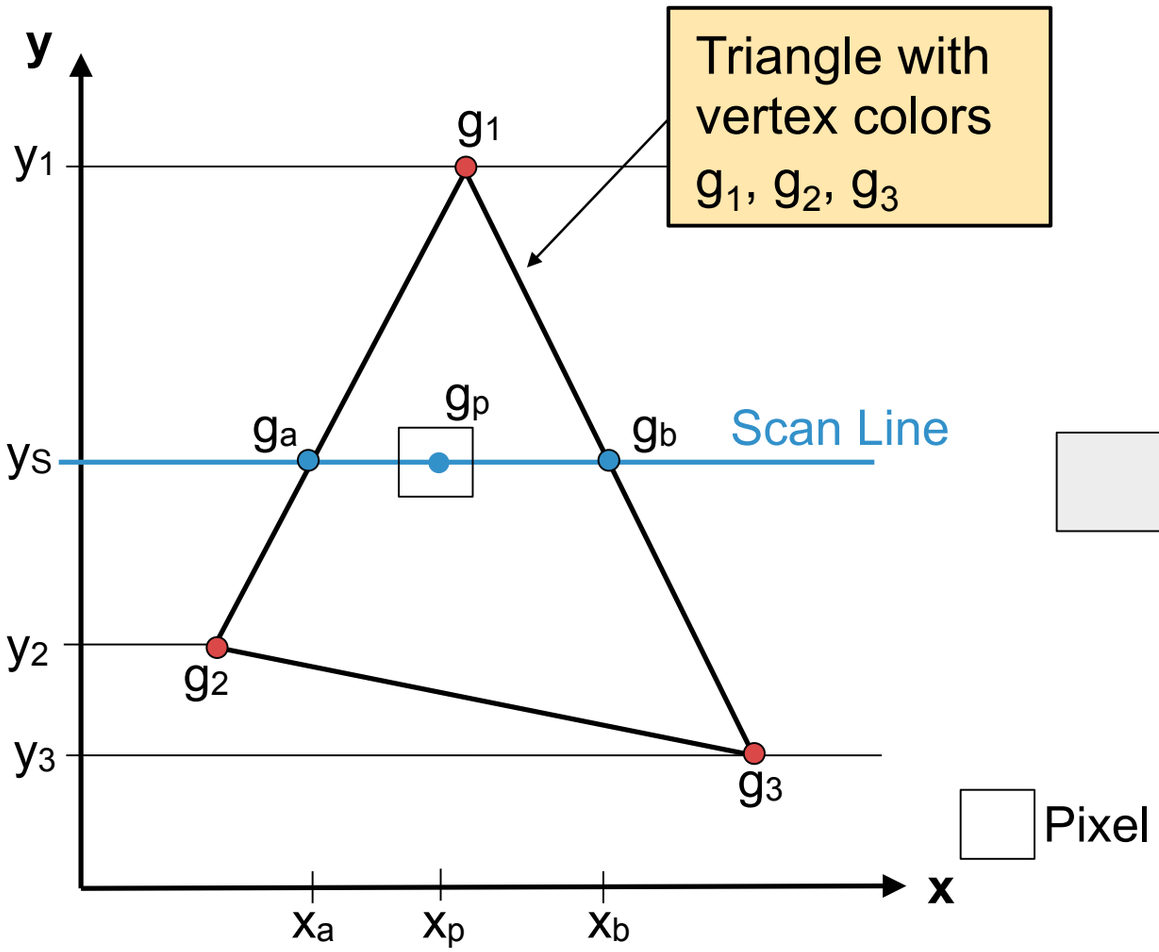


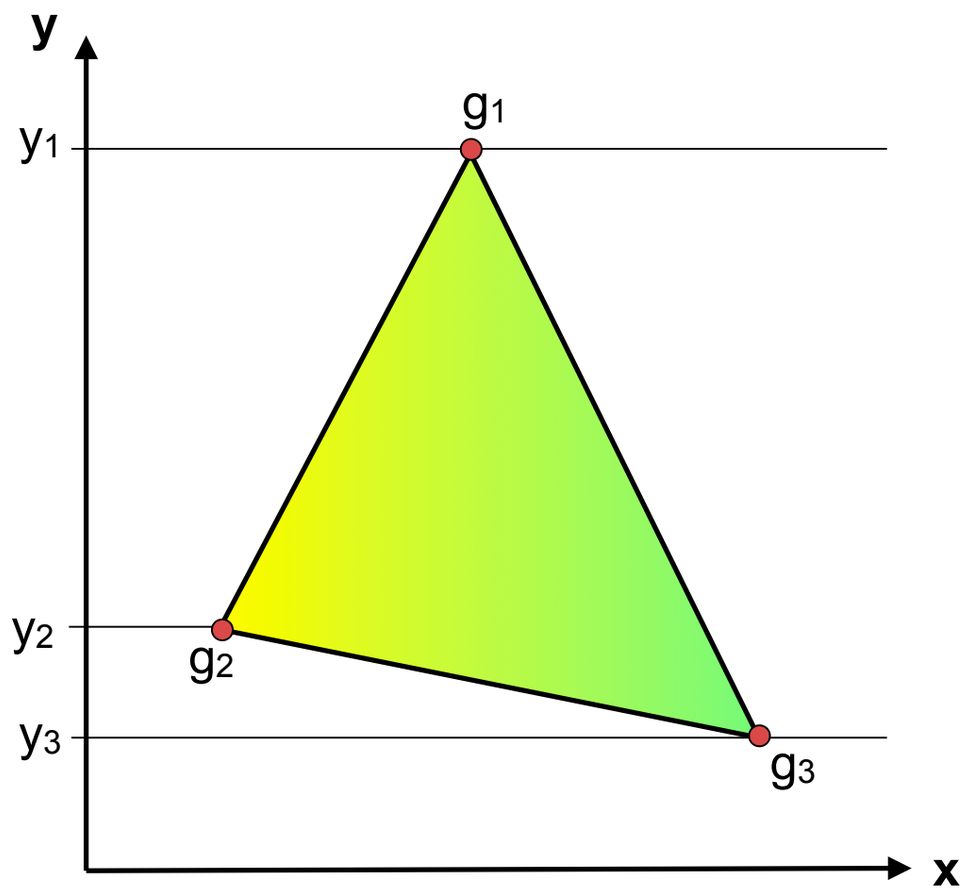


Calculate the normal vectors at each contact vertex between two surfaces in order to approximate a smooth surface.

- Smooths the edges between polygons
- Color is linearly interpolated from vertex points
- Specular highlights appear interpolated

Gouraud Shading (2/2)





The interpolation of a pixel color takes two steps: first, the color at the triangle edges are calculated. Secondly, the pixel color is calculated.

$$g_a = g_1 - (g_1 - g_2) \cdot \frac{y_1 - y_s}{y_1 - y_2}$$

$$g_b = g_1 - (g_1 - g_3) \cdot \frac{y_1 - y_s}{y_1 - y_3}$$

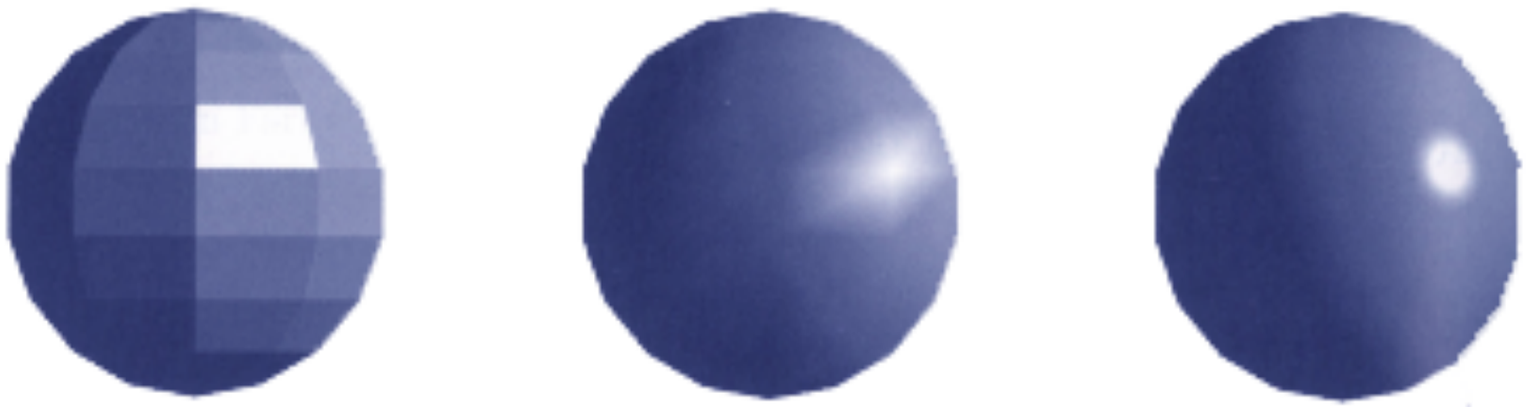
$$g_p = g_b - (g_b - g_a) \cdot \frac{x_b - x_p}{x_b - x_a}$$

$g_{a,\,b}$: Color at the polygon edge
g_p: Color of the pixel

Comparison

ARLAB

Flat Shading **Gouraud Shading** **Phong Shading**



81 Polygons

2500 Polygons

25000 Polygons

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Thank you!

Questions

Rafael Radkowski, Ph.D.
Iowa State University
Virtual Reality Applications Center
1620 Howe Hall
Ames, Iowa 5001, USA

+1 515.294.5580
+1 515.294.5530(fax)

rafael@iastate.edu

VRAC|HCI

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY