

Information Visualization

W07: Shading

Graduation School of System Informatics

Department of Computational Science

Naohisa Sakamoto, Akira Kageyama

May.9, 2017

Schedule

- W01 4/11 Guidance
- W02 4/12 Setup
- W03 4/18 Introduction to Data Visualization
- W04 4/19 CG Programming
- W05 4/25 Rendering Pipeline
- W06 4/26 Coordinate Systems and Transformations
- W07 5/09 Shading
- W08 5/10 Shader Programming
- W09 5/16 Visualization Pipeline
- W10 5/17 Data Model and Transfer Function
- W11 5/23 Scalar Data Visualization 1 (Isosurface Extraction)
- W12 5/24 Implementation of Isosurface Extraction
- W13 5/30 Scalar Data Visualization 2 (Volume Rendering)
- W14 5/31 Implementation of Volume Rendering
- W15 6/06 Student Presentations

Table of Contents

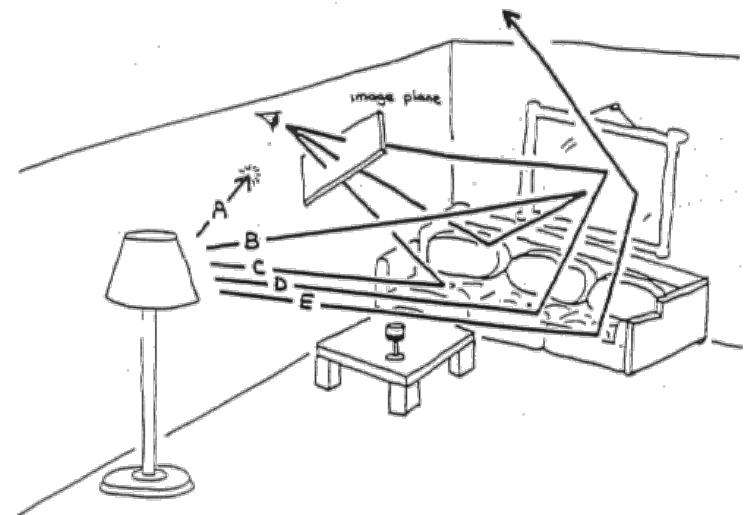
- Illumination Model
- Reflection Model
- Shading Model
- Shader Programming

Table of Contents

- Illumination Model
- Reflection Model
- Shading Model
- Shader Programming

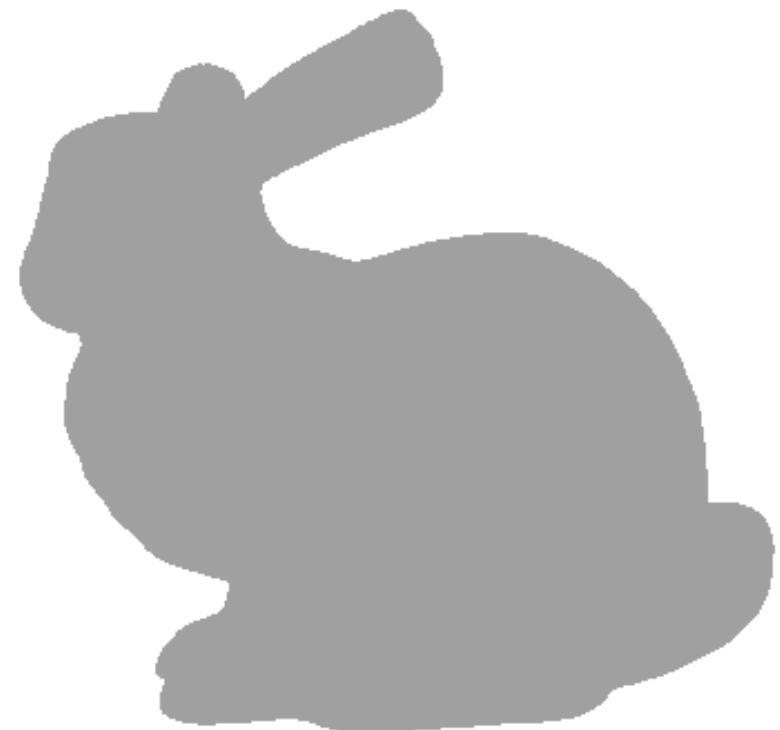
Illumination

- Illumination model (Lighting)
 - From Physics we can derive models, called "illumination models", of how light reflects from surfaces and produces what we perceive as color.
 - In general, light leaves some light source, e.g. a lamp or the sun, is reflected from many surfaces and then finally reflected to our eyes, or through an image plane of a camera.



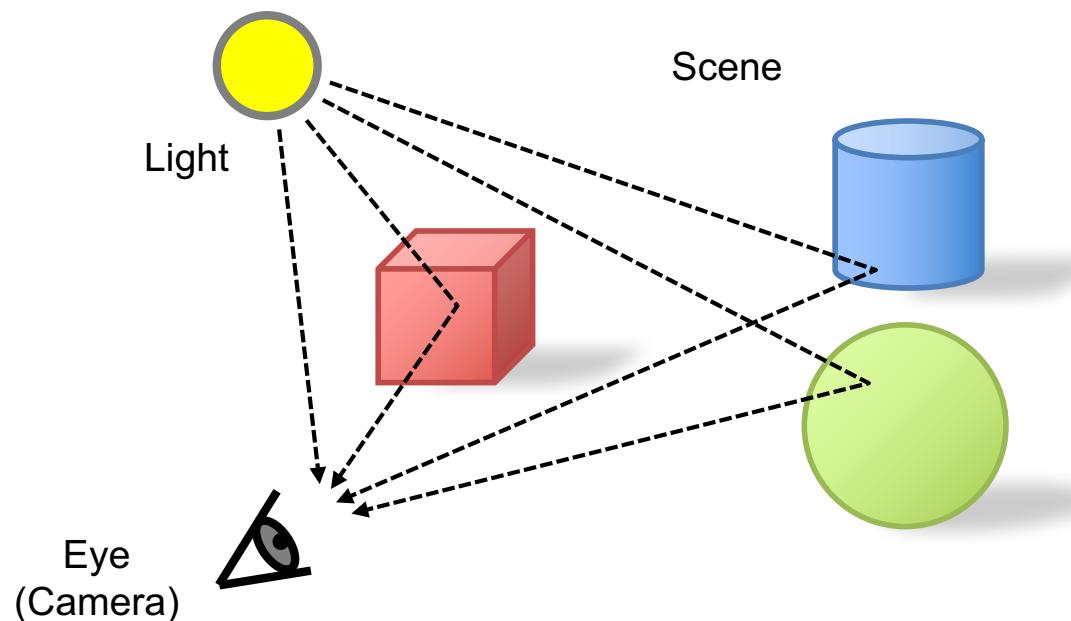
Illumination

- Lighting
 - Stanford bunny with and without lighting effects



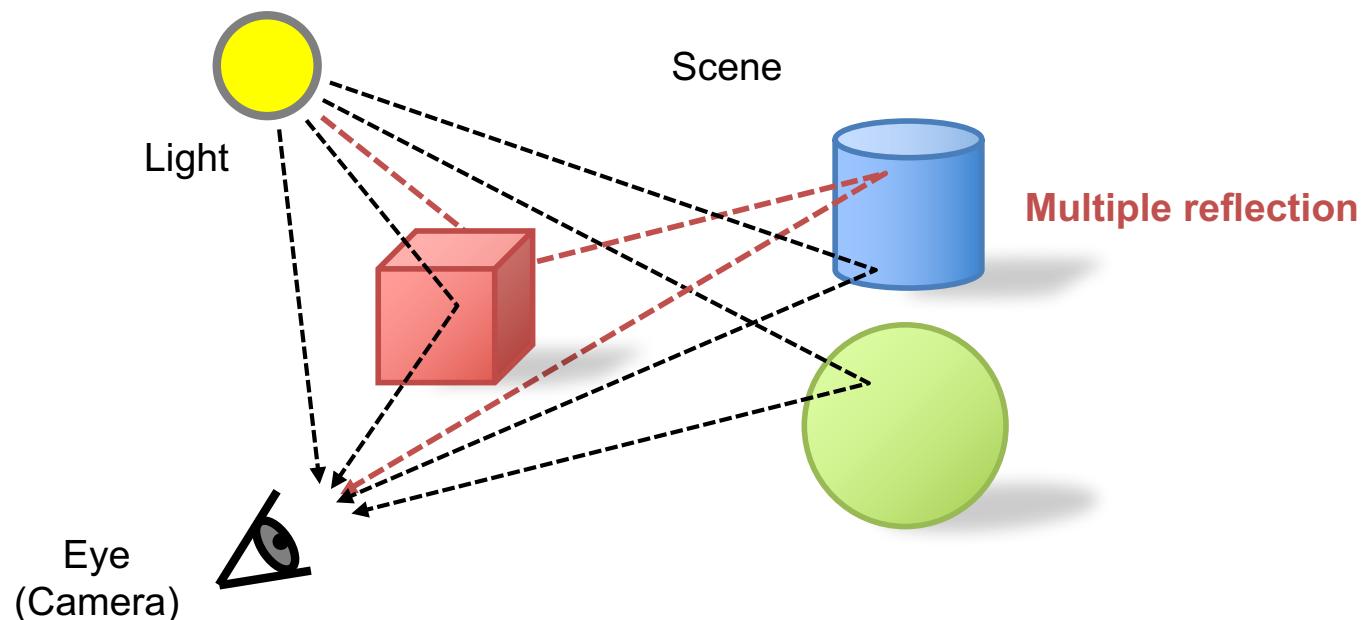
Illumination Model

- Local Illumination Model
 - The contribution from the light that goes directly from the light source and is reflected from the surface is called a "local illumination model".



Illumination Model

- Global Illumination Model
 - A "global illumination model" adds to the local model the light that is reflected from other surfaces to the current surface.



Illumination

- Global Illumination Model

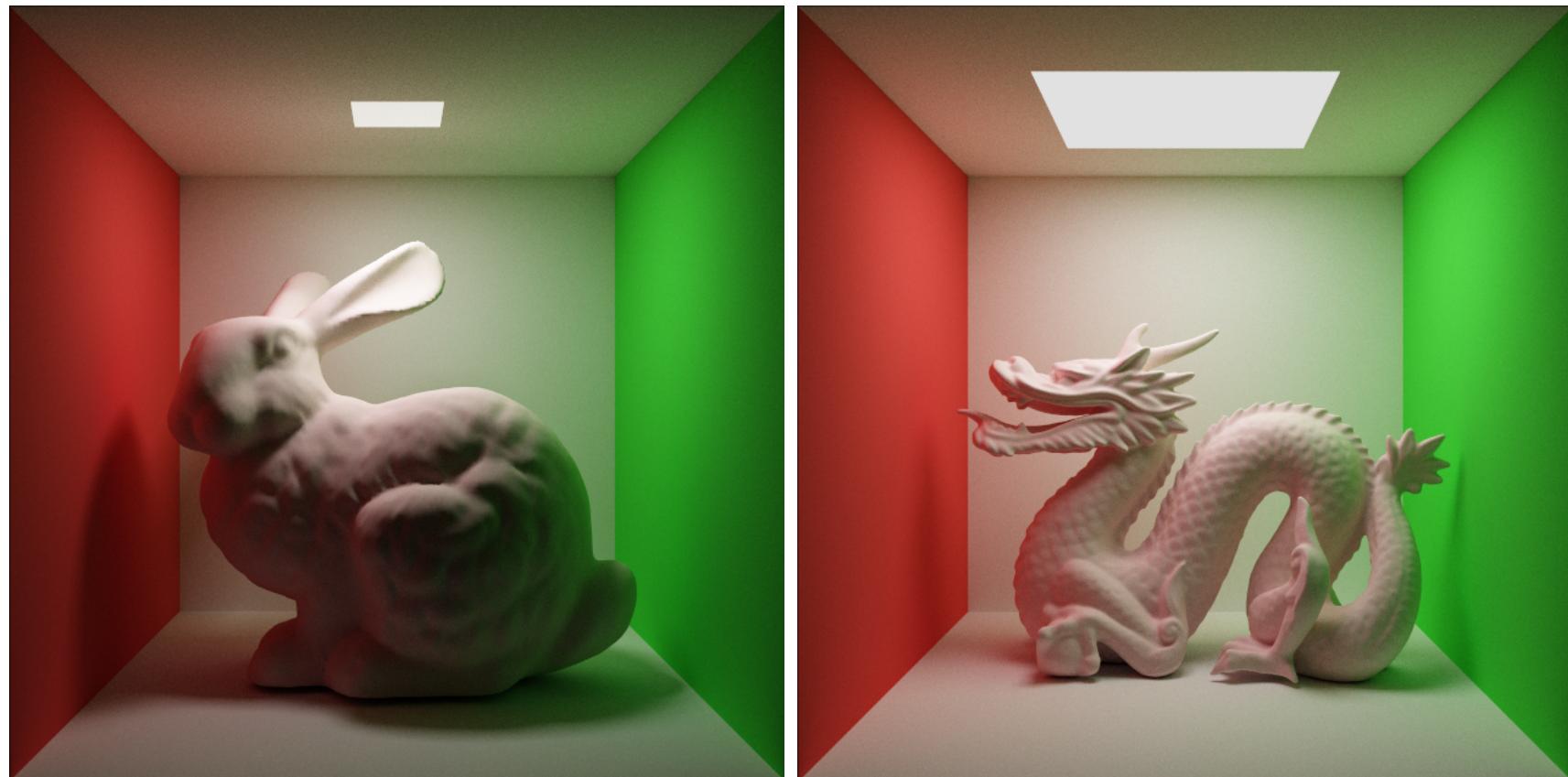
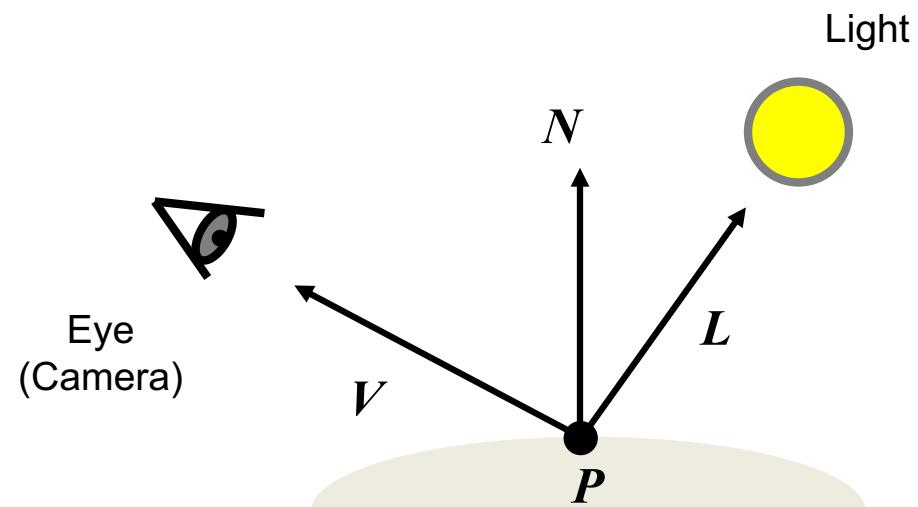


Table of Contents

- Illumination Model
- Reflection Model
- Shading Model
- Shader Programming

Reflection

- The intensity (luminosity) of the reflected light can be calculated by the following components.
 - Ambient reflection
 - Diffuse reflection
 - Specular reflection



- P Point on a surface
 N Normal vector at P
 L Light direction at P
 V Viewing direction at P

Ambient Reflection

- Reflection of light from the environment and assumed to be equal in all directions.
 - Independent of light position, object orientation, camera position or orientation

$$I_{ambient} = I_a k_a$$

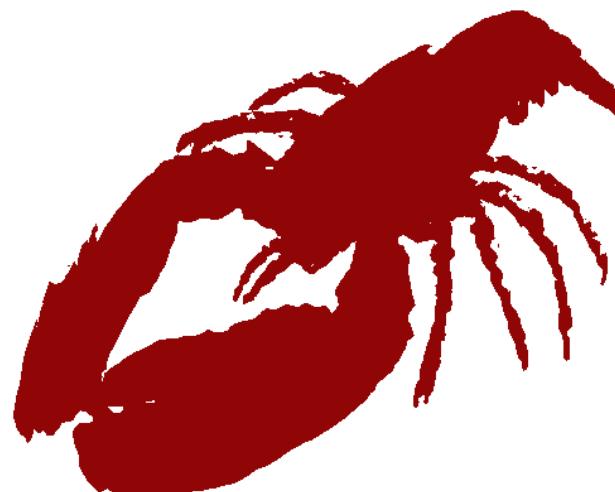
| | |
|---------------|-------------------------------------|
| $I_{ambient}$ | Intensity of the ambient reflection |
| I_a | Ambient intensity |
| k_a | Ambient reflection coefficient |

Ambient Reflection

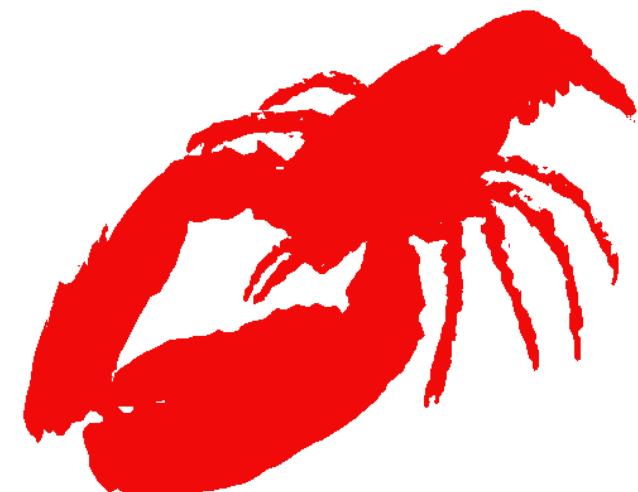
- Effect of ambient reflection coefficient k_a



$$k_a = 0.2$$



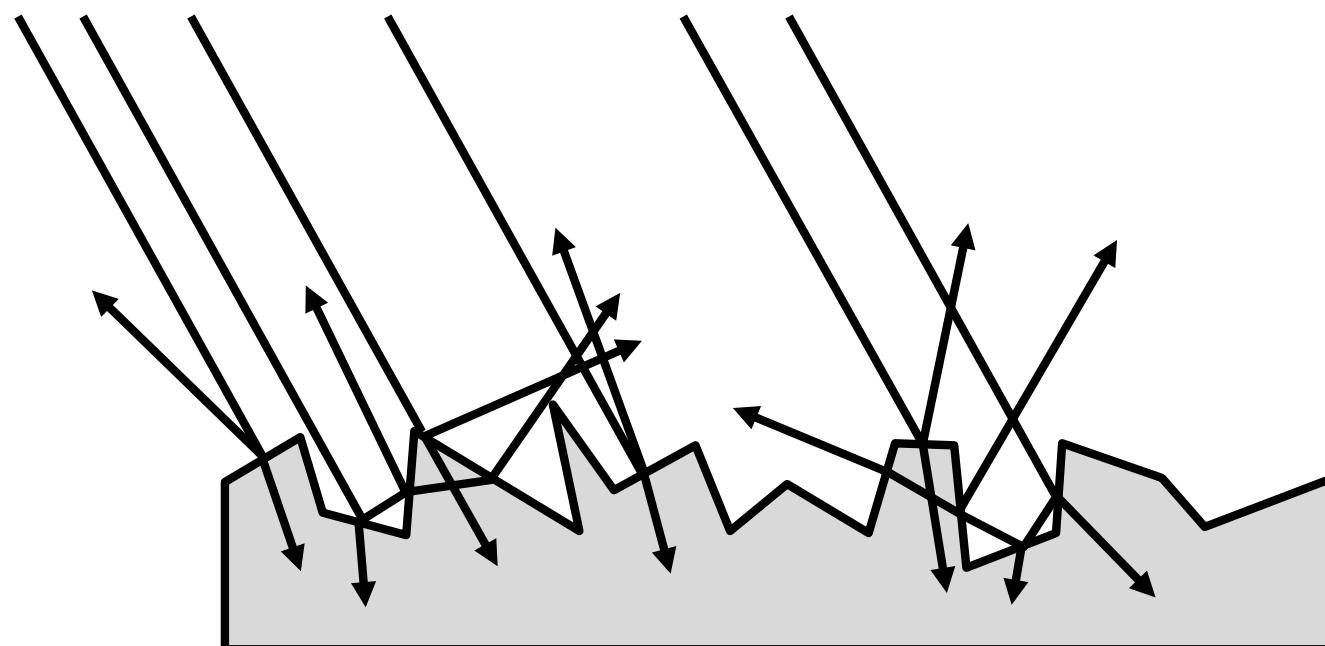
$$k_a = 0.6$$



$$k_a = 1.0$$

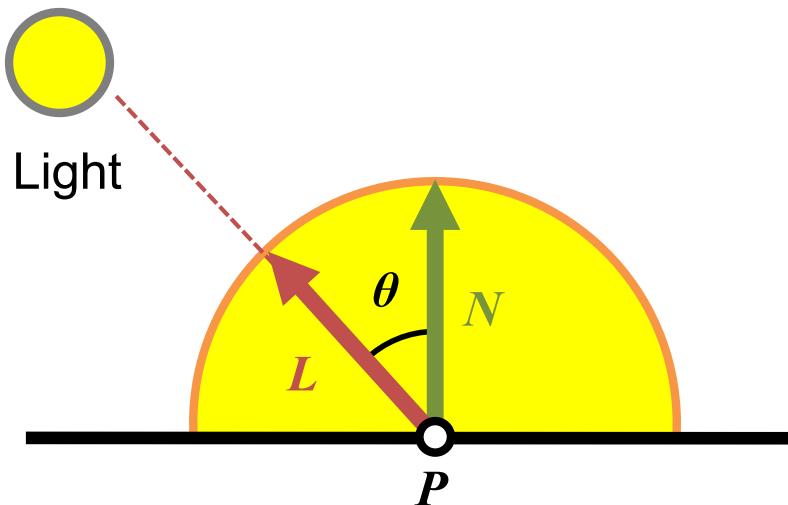
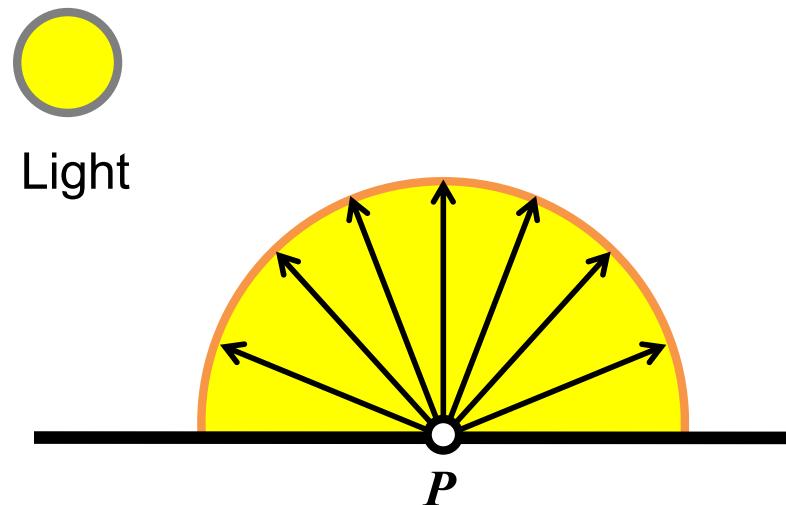
Diffuse Reflection

- Reflection of light from a surface such that incident light is reflected into many different directions.



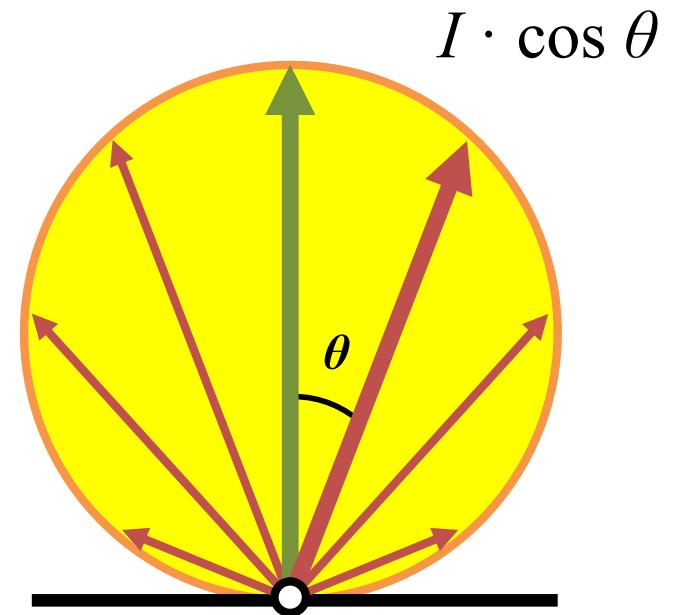
Diffuse Reflection

- Reflected intensity does not depend on the viewing direction.
- Incoming light depends on the light direction.



Diffuse Reflection

- Lambert's cosine law
 - Radiant intensity observed from an ideal diffusely reflecting surface is directly proportional to the cosine of the angle θ between the direction of the incident light and the surface normal.

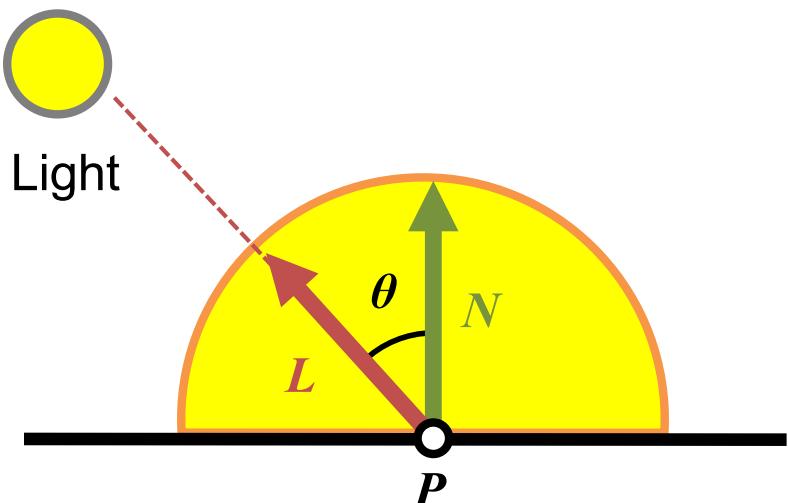


Diffuse Reflection

- Calculation of diffuse reflection

$$\begin{aligned}I_{\text{diffuse}} &= I_i k_d \cos \theta \\&= I_i k_d (\mathbf{N} \cdot \mathbf{L})\end{aligned}$$

| | |
|----------------------|-------------------------------------|
| I_{diffuse} | Intensity of the diffuse reflection |
| I_i | Intensity of the light source |
| k_d | Diffuse reflection coefficient |
| \mathbf{N} | Normal vector (unit vector) |
| \mathbf{L} | Light direction (unit vector) |

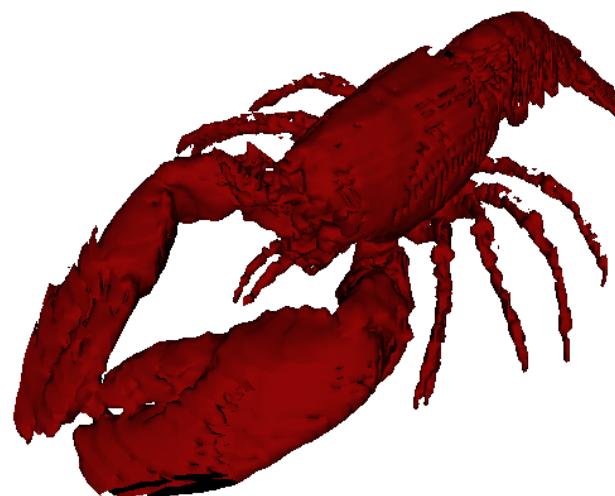


Diffuse Reflection

- Effect of diffuse reflection coefficient k_d



$$k_d = 0.2$$



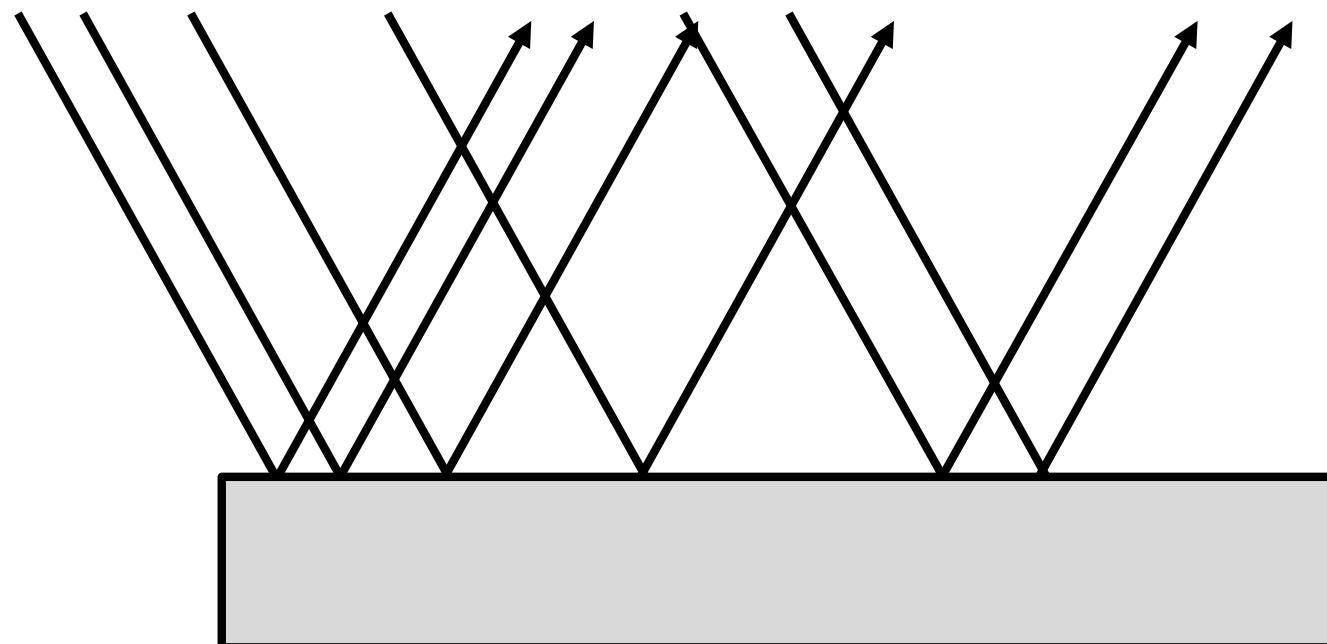
$$k_d = 0.6$$



$$k_d = 1.0$$

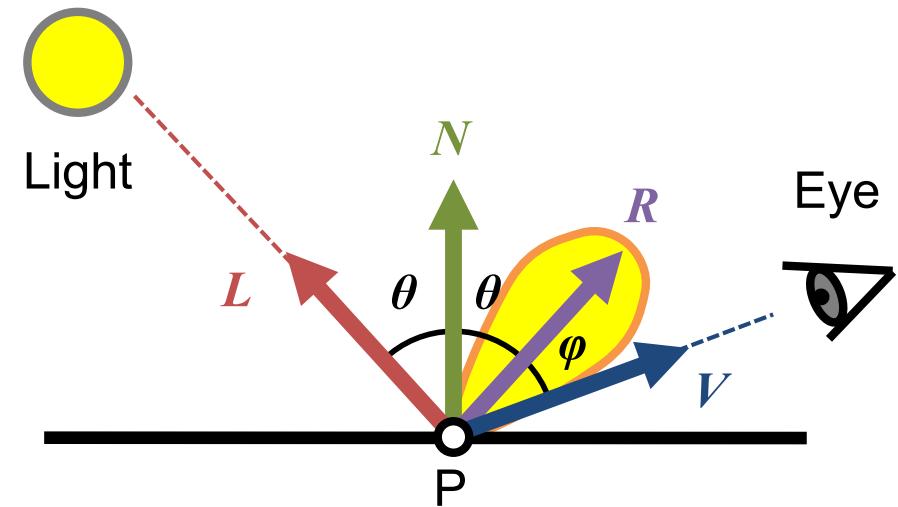
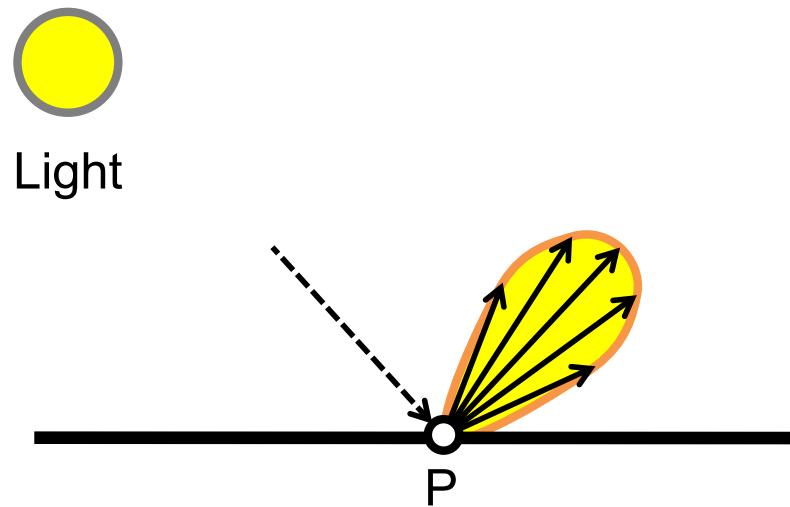
Specular Reflection

- Mirror-like reflection of light from a surface, in which light from a single incoming direction is reflected into a single outgoing direction.



Specular Reflection

- Reflected intensity does not depend on the viewing direction.
- Incoming light depends on the light direction.

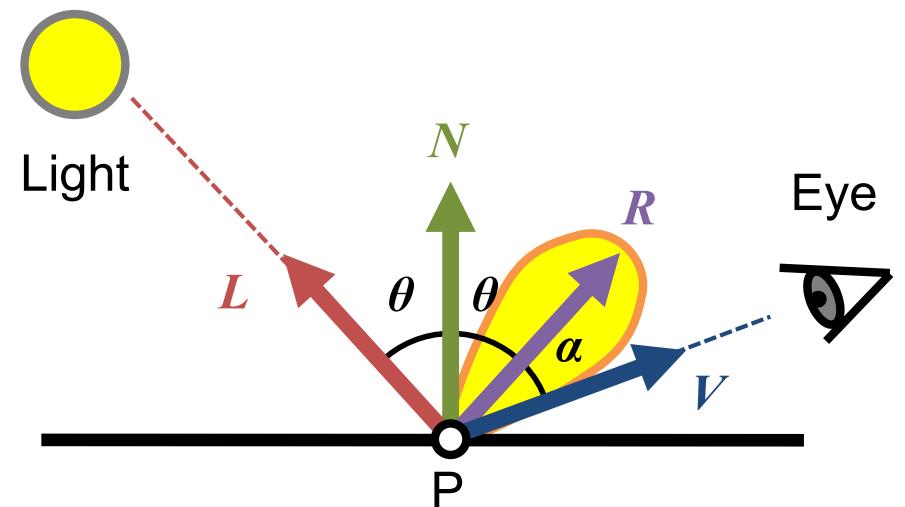


Specular Reflection

- Calculation of specular reflection

$$\begin{aligned}I_{specular} &= I_i k_s \cos^n \alpha \\&= I_i k_s (V \cdot R)^n\end{aligned}$$

| | |
|----------------|--------------------------------------|
| $I_{specular}$ | Intensity of the specular reflection |
| I_i | Intensity of the light source |
| k_s | Specular reflection coefficient |
| N | Normal vector (unit vector) |
| L | Light direction (unit vector) |
| V | Viewing direction (unit vector) |
| R | Reflection vector of L |

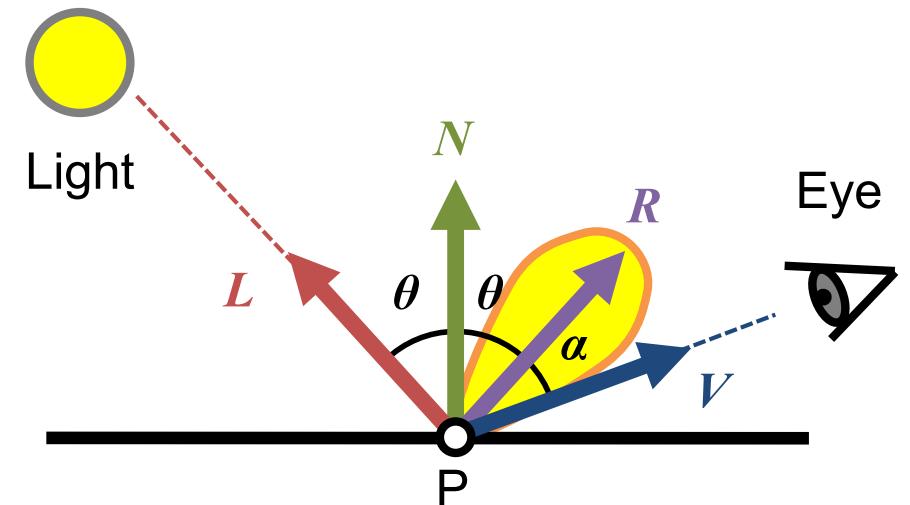


Specular Reflection

- Calculation of reflection vector R

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

| | |
|----------------|--------------------------------------|
| $I_{specular}$ | Intensity of the specular reflection |
| I_i | Intensity of the light source |
| k_s | Specular reflection coefficient |
| N | Normal vector (unit vector) |
| L | Light direction (unit vector) |
| V | Viewing direction (unit vector) |
| R | Reflection vector of L |



Specular Reflection

- Effect of specular reflection coefficient k_s



$$k_s = 0.2$$



$$k_s = 0.6$$

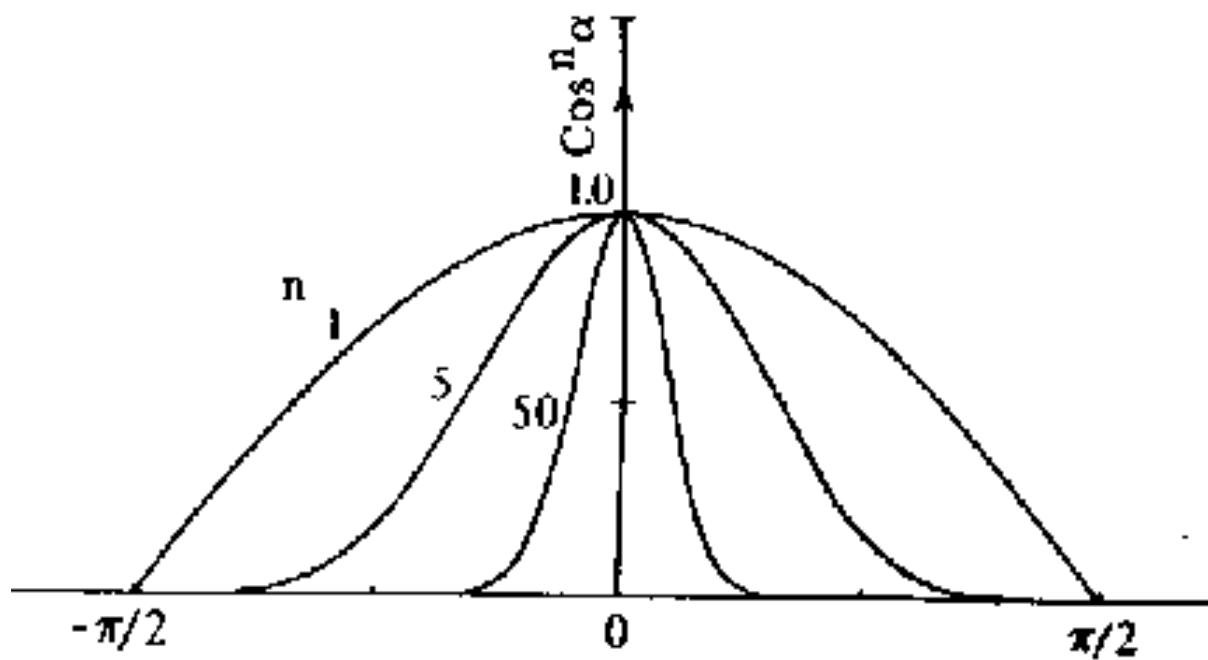


$$k_s = 1.0$$

Specular Reflection

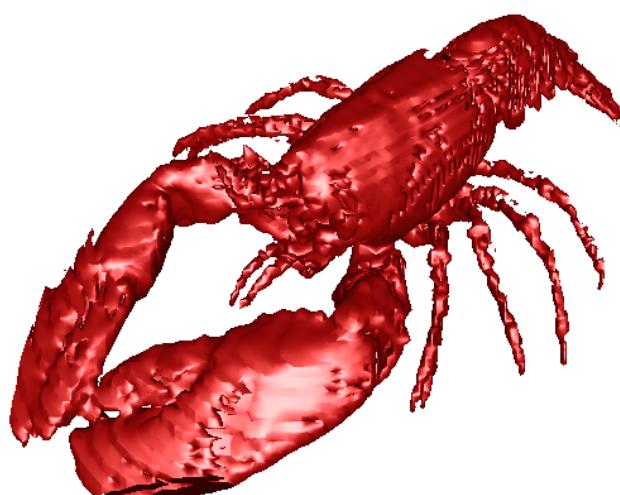
- Shininess exponent n

$$I_{specular} = I_i k_s \cos^n \alpha$$



Specular Reflection

- Effect of shininess exponent n



$n = 10$



$n = 50$



$n = 150$

Reflection Model

- Lambertian reflection model

$$I = I_{ambient} + I_{diffuse}$$

$$= I_a k_a + I_i k_d (\mathbf{N} \cdot \mathbf{L})$$

- Phong reflection model

$$I = I_{ambient} + I_{diffuse} + I_{specular}$$

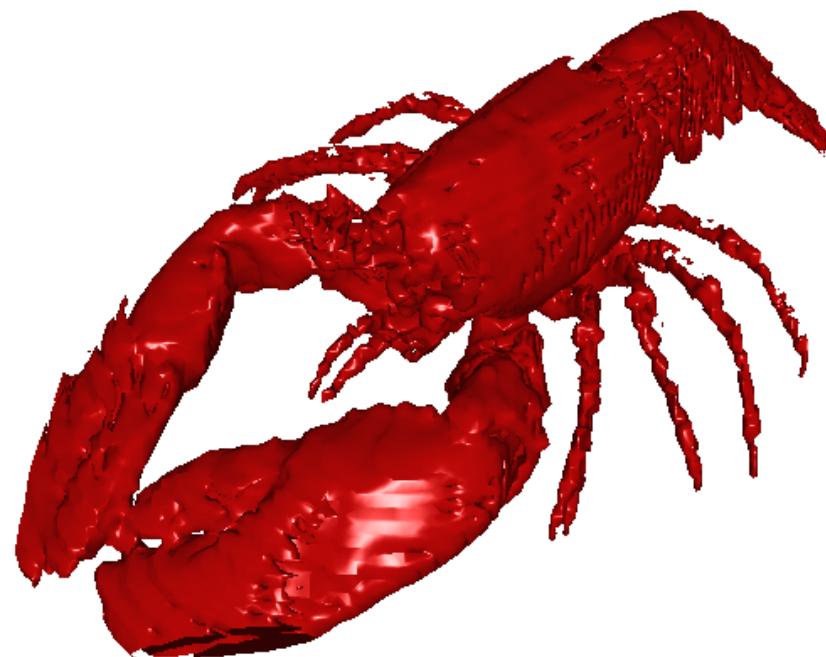
$$= I_a k_a + I_i k_d (\mathbf{N} \cdot \mathbf{L}) + I_i k_s (\mathbf{V} \cdot \mathbf{R})^n$$

Reflection Model

- Comparison with Lambertian (left) and Phong (right) reflection models



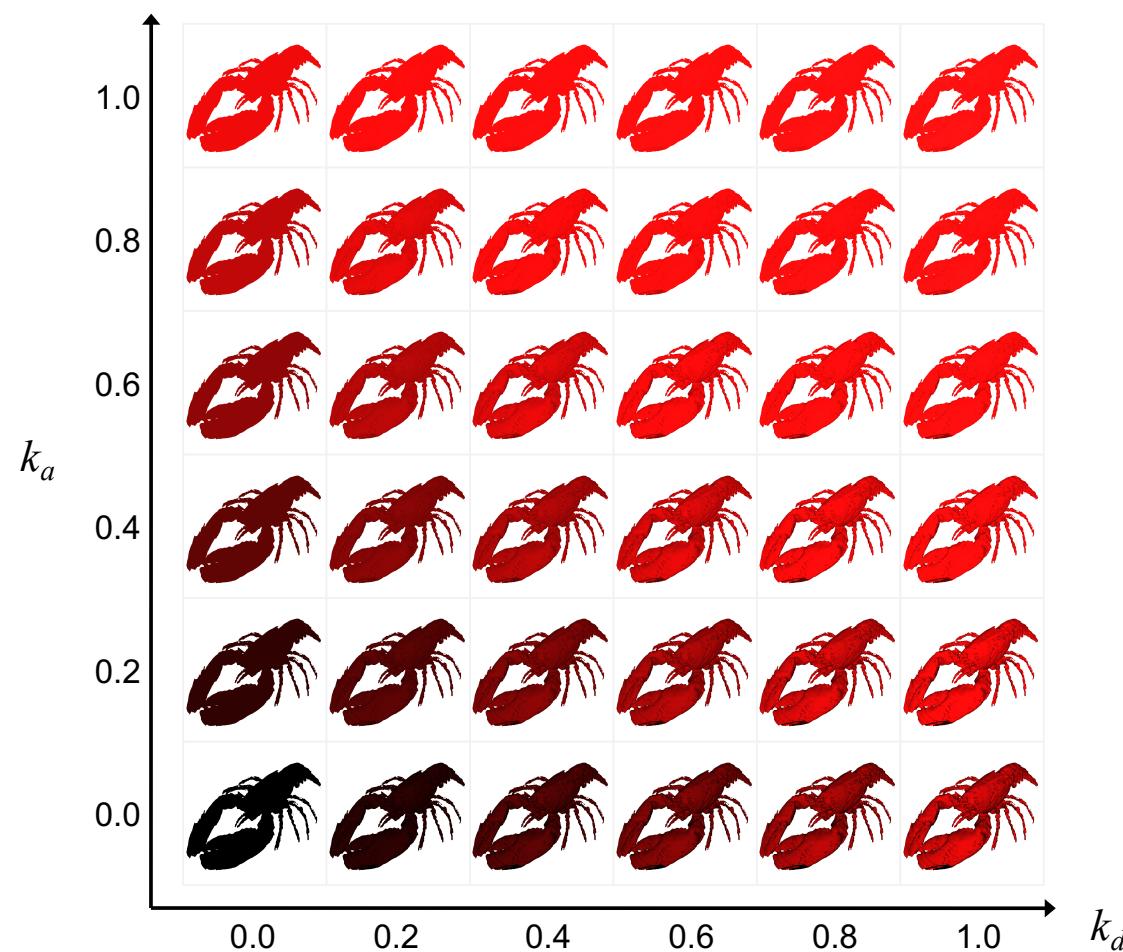
$$k_a = 0.2, k_d = 0.6$$



$$k_a = 0.2, k_d = 0.6, k_s = 0.8, n = 100$$

Lambertian Reflection Model

- Effect of k_a and k_d



Phong Reflection Model

- Effect of k_s and n

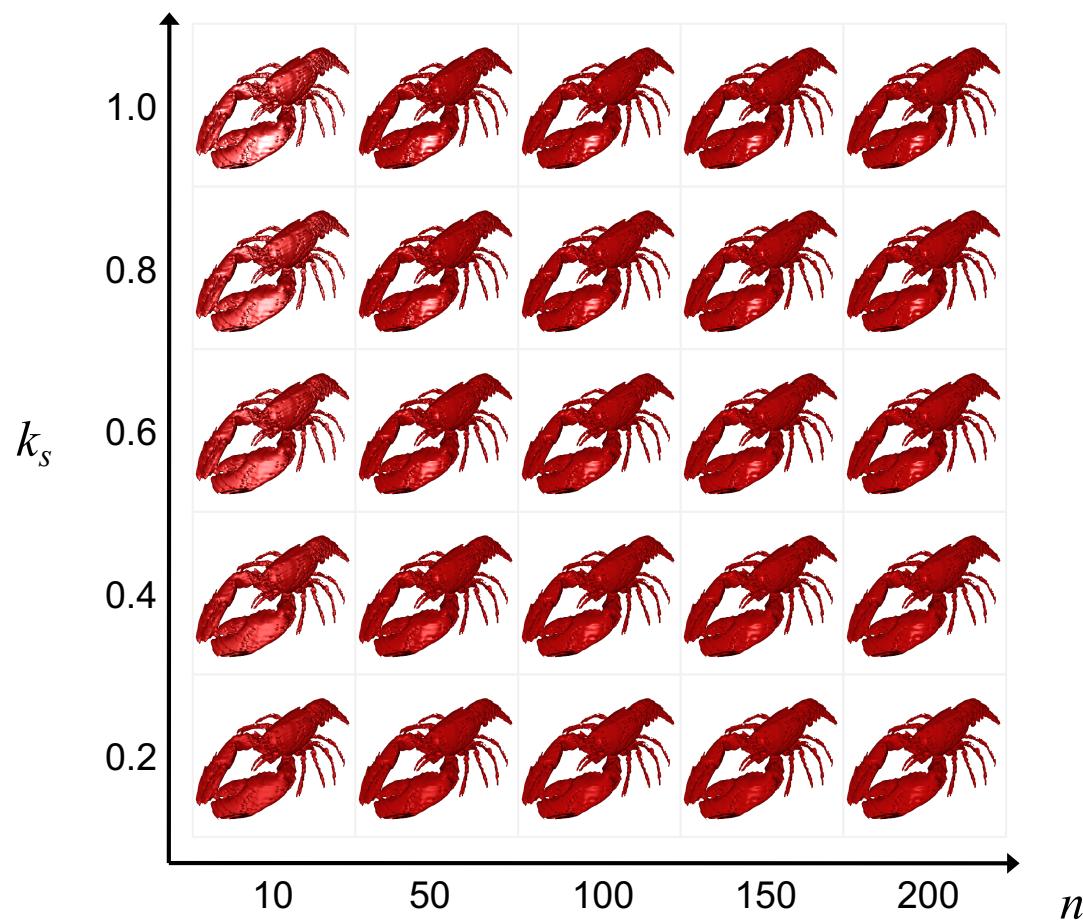


Table of Contents

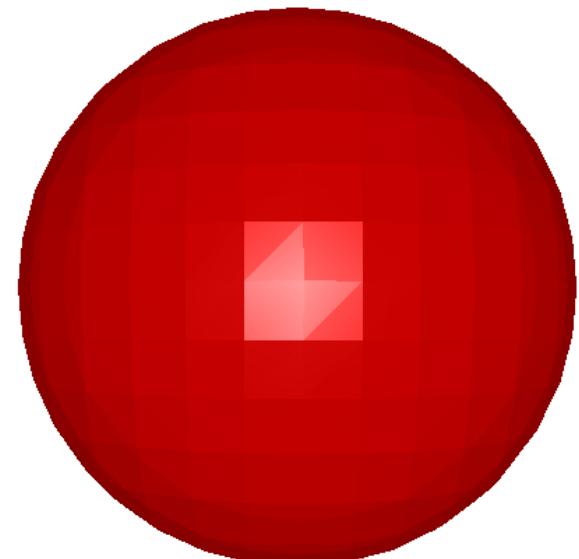
- Illumination Model
- Reflection Model
- **Shading Model**
- Shader Programming

Shading

- Process of determining the color of a pixel based on the reflection models
 - Flat shading
 - Smooth shading

Flat Shading

- Uses the same color for every pixel in a face
 - Less computationally expensive
 - Edges appear
 - Not suitable for smooth objects

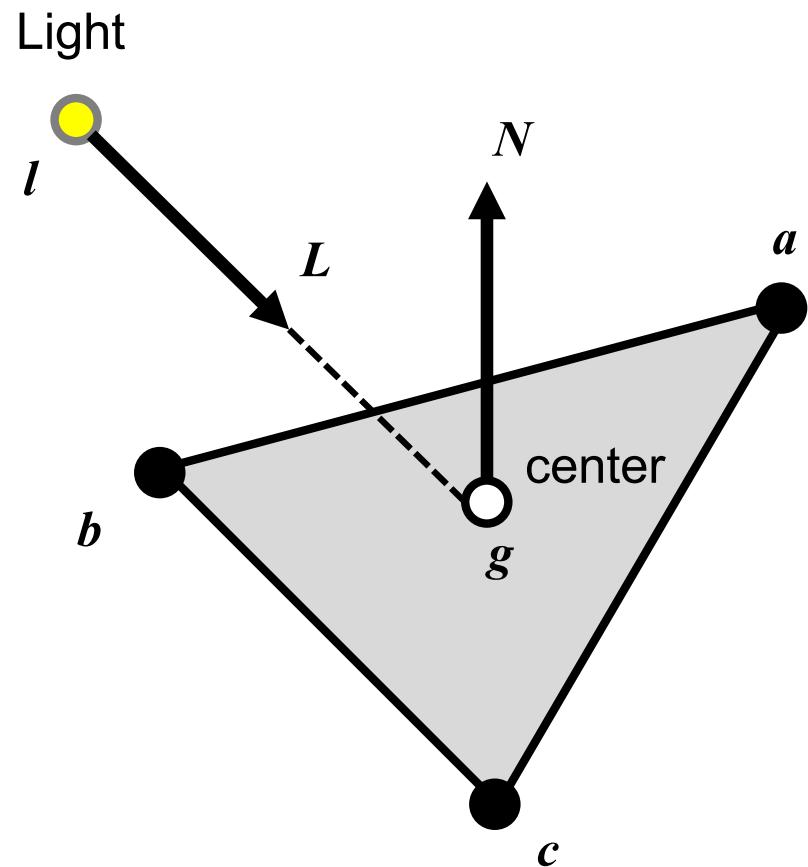


Flat Shading

- Calculation of normal vector N and light direction vector L for a face

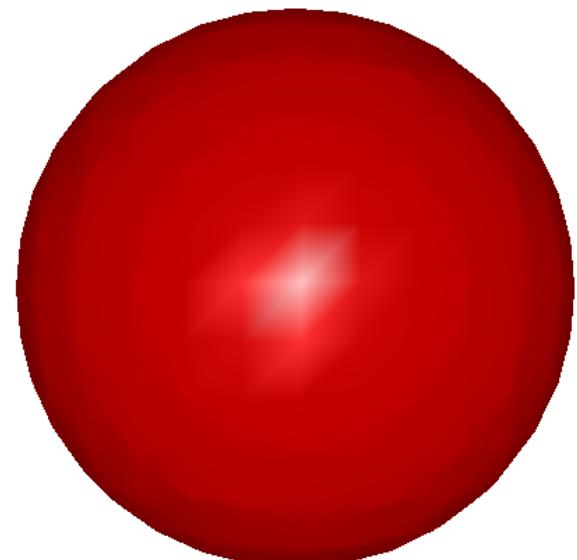
$$\vec{N} = \frac{(\vec{b} - \vec{a}) \times (\vec{c} - \vec{a})}{|(\vec{b} - \vec{a}) \times (\vec{c} - \vec{a})|}$$

$$\vec{L} = \frac{\vec{l} - \vec{g}}{|\vec{l} - \vec{g}|}$$



Smooth Shading

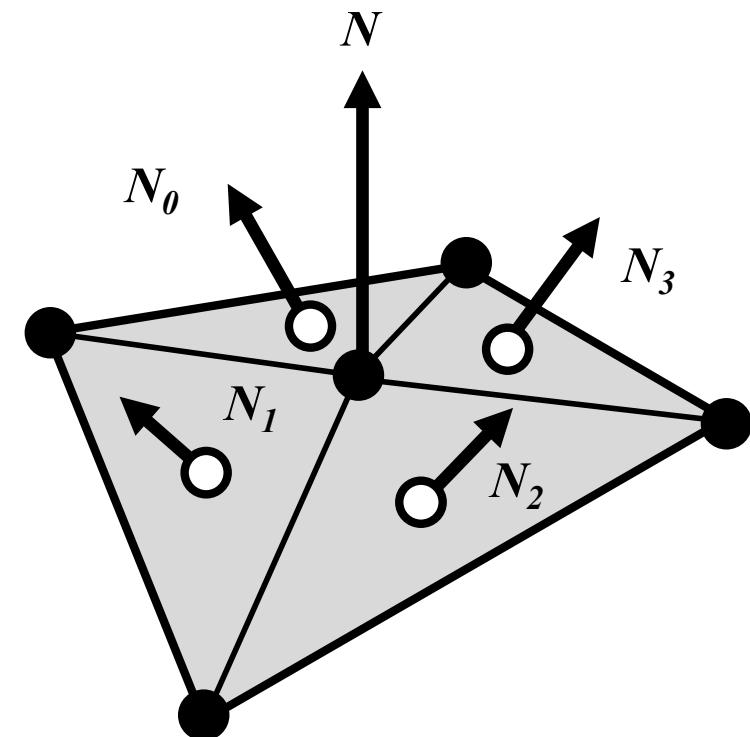
- Uses linear interpolation of colors or normal vectors between vertices
 - More computationally expensive
 - Edges disappear
 - Suitable for any objects



Smooth Shading

- Calculation of normal vector \vec{N} for a vertex

$$\vec{N} = \frac{\vec{N}_0 + \vec{N}_1 + \vec{N}_2 + \vec{N}_3}{|\vec{N}_0 + \vec{N}_1 + \vec{N}_2 + \vec{N}_3|}$$

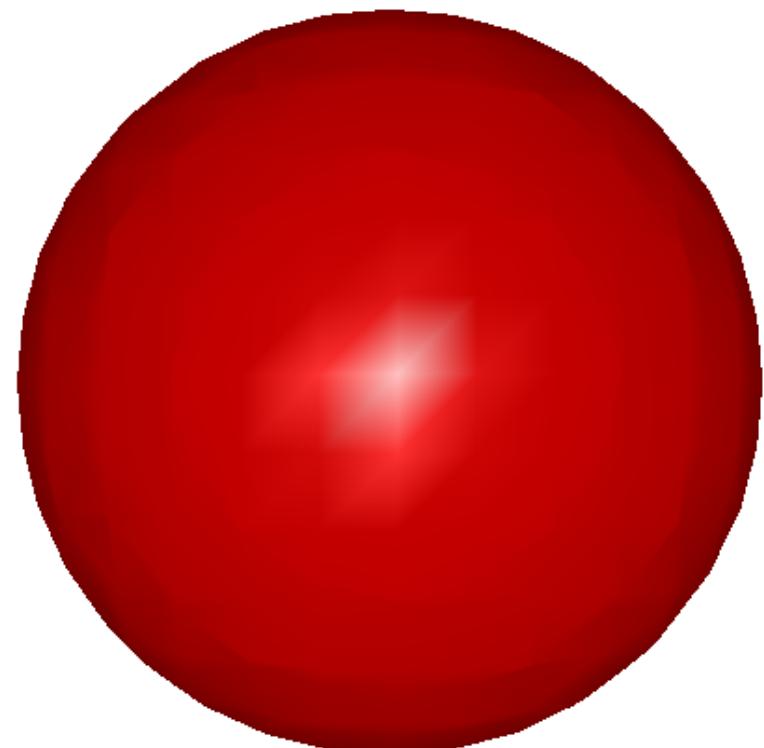
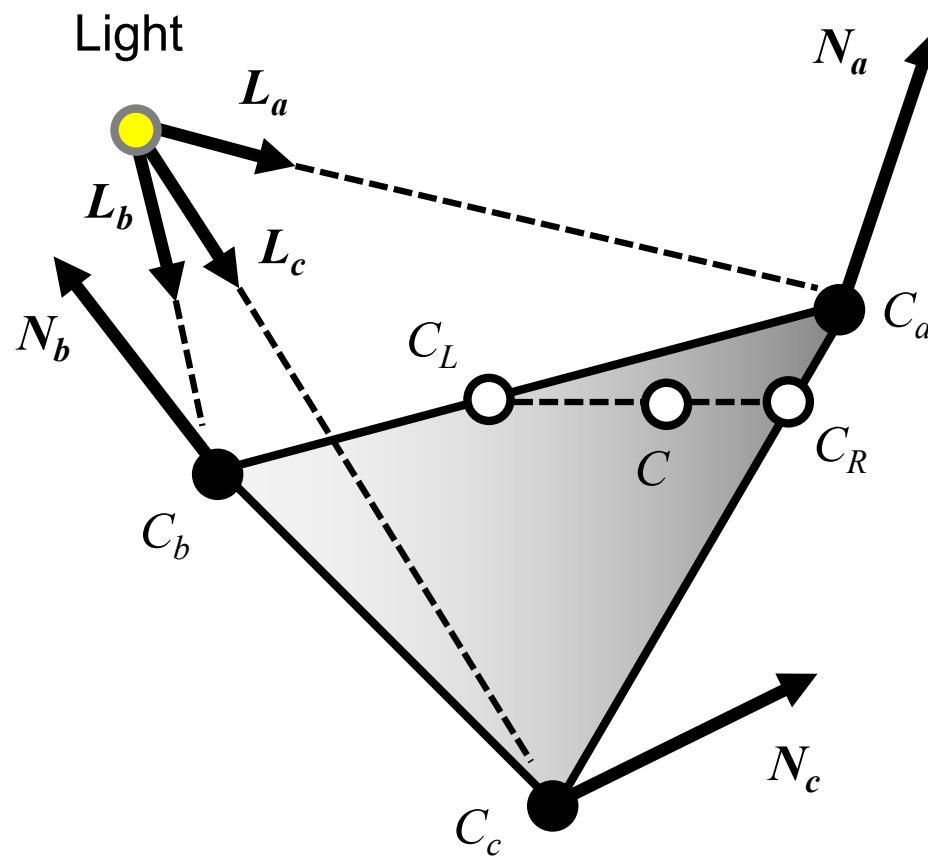


Types of Smooth Shading

- Gouraud shading
 - Color interpolation shading
- Phong shading
 - Normal interpolation shading

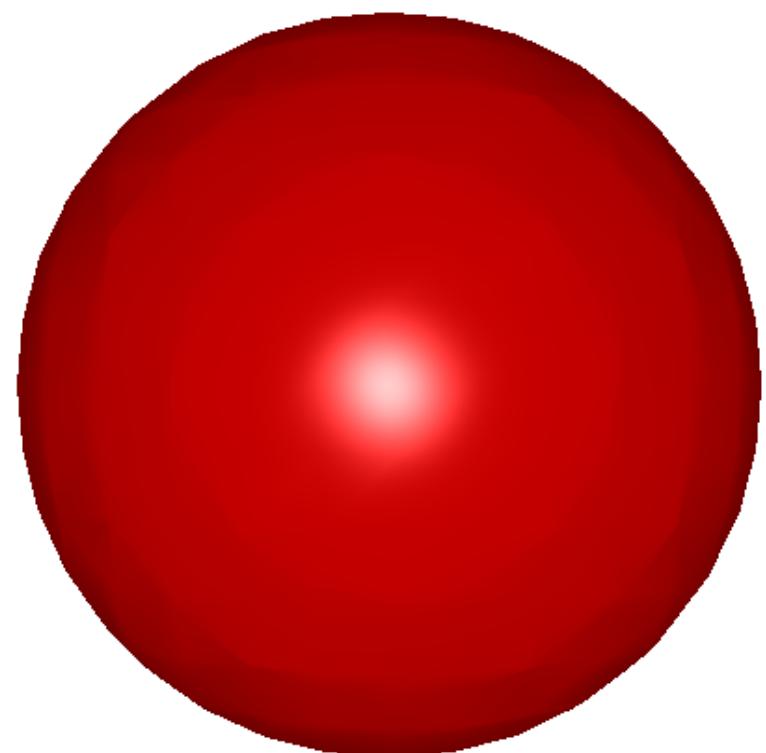
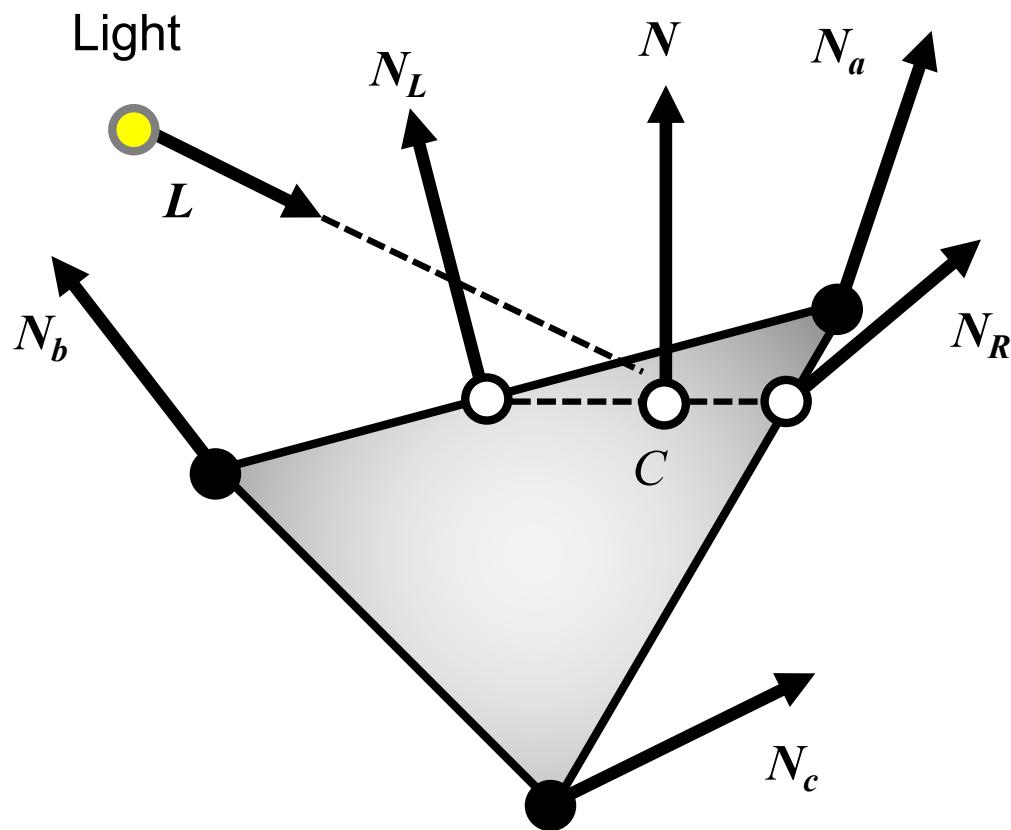
Gouraud Shading

- Colors are interpolated across the face
- Reflection model used at each vertex



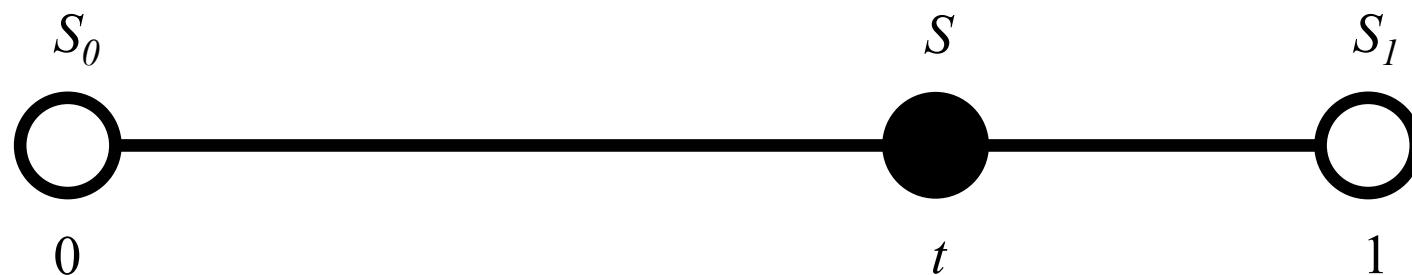
Phong Shading

- Normals are interpolated across the face
- Reflection model used at each pixel



Interpolation

- Linear interpolation



$$S = (1 - t) S_0 + t S_1$$

Interpolation

- Bilinear interpolation

$$S = (1 - t) S_a + t S_b$$

$$S_a = (1 - t_a) S_0 + t_a S_1$$

$$S_b = (1 - t_b) S_0 + t_b S_2$$

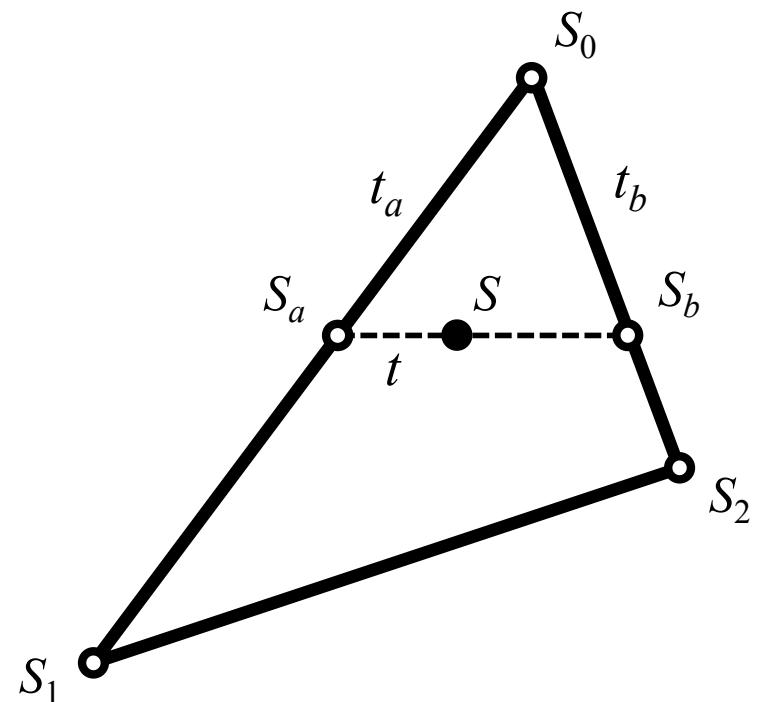


Table of Contents

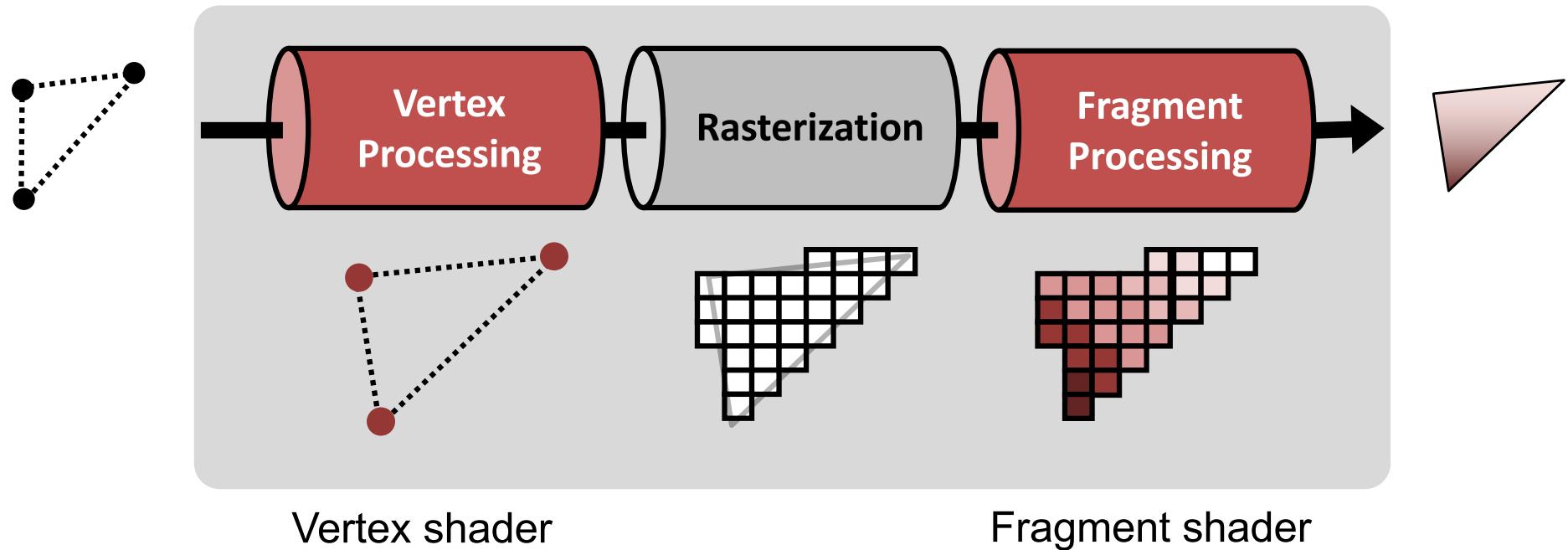
- Illumination Model
- Reflection Model
- Shading Model
- Shader Programming

Shader

- OpenGL Shading Language (GLSL)
 - High-level shading language based on the syntax of the C programming language.
 - Created by the OpenGL ARB (OpenGL Architecture Review Board) to give developers more direct control of the graphics pipeline without having to use ARB assembly language or hardware-specific languages.

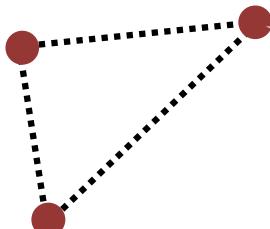
Shader Programming

- Rendering pipeline
 - Vertex shader
 - Fragment shader



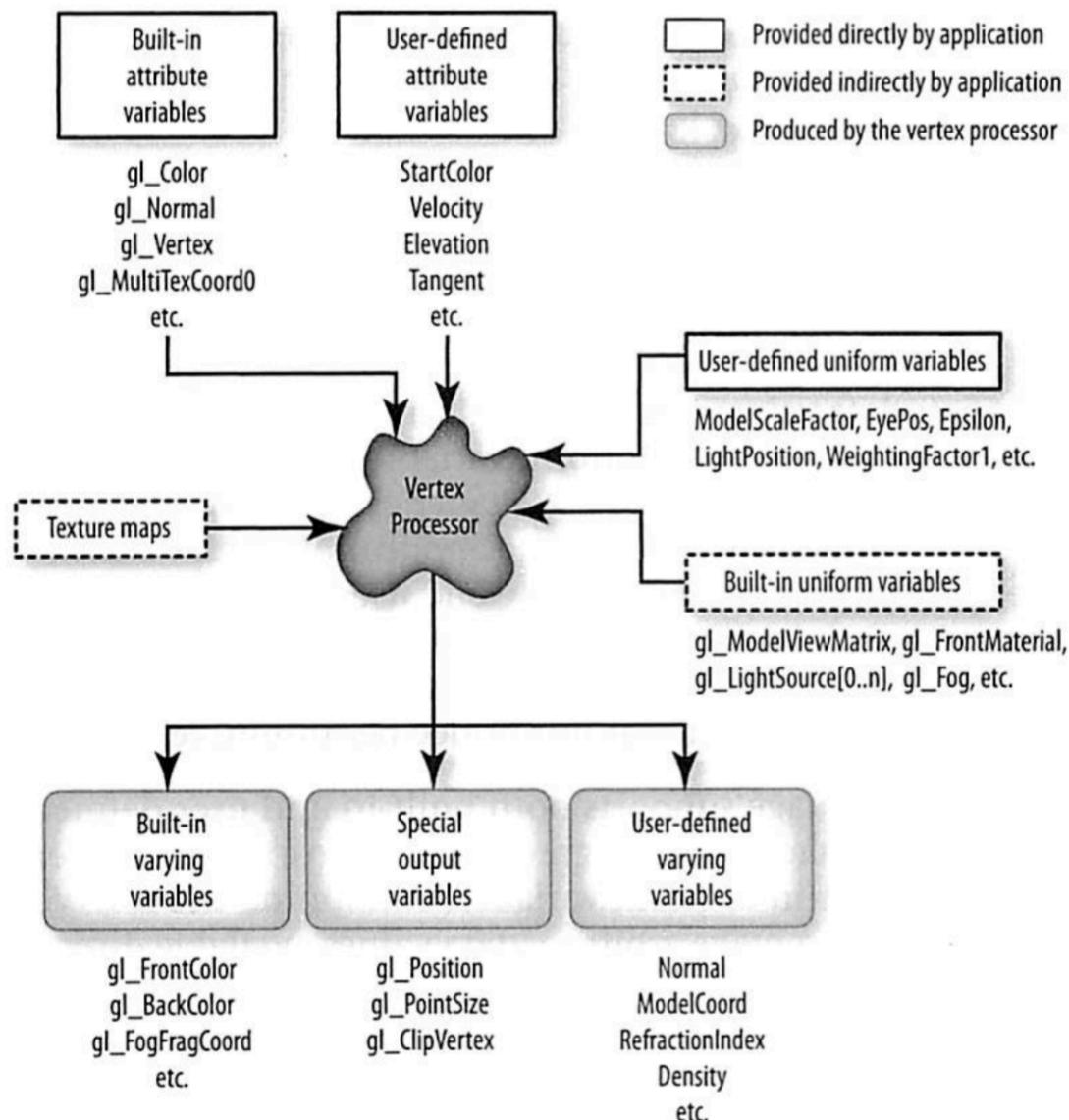
Vertex Shader

- Programmable Shader stage in the rendering pipeline that handles the processing of individual vertices.



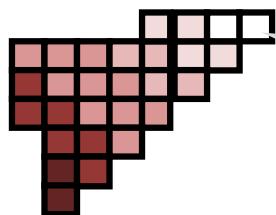
```
void main()
{
    vec4 v = vec4( position, 1.0 );
    mat4 MV = modelViewMatrix;
    mat4 P = projectionMatrix;
    gl_Position = P * MV * v;
}
```

Vertex Shader Inputs and Outputs



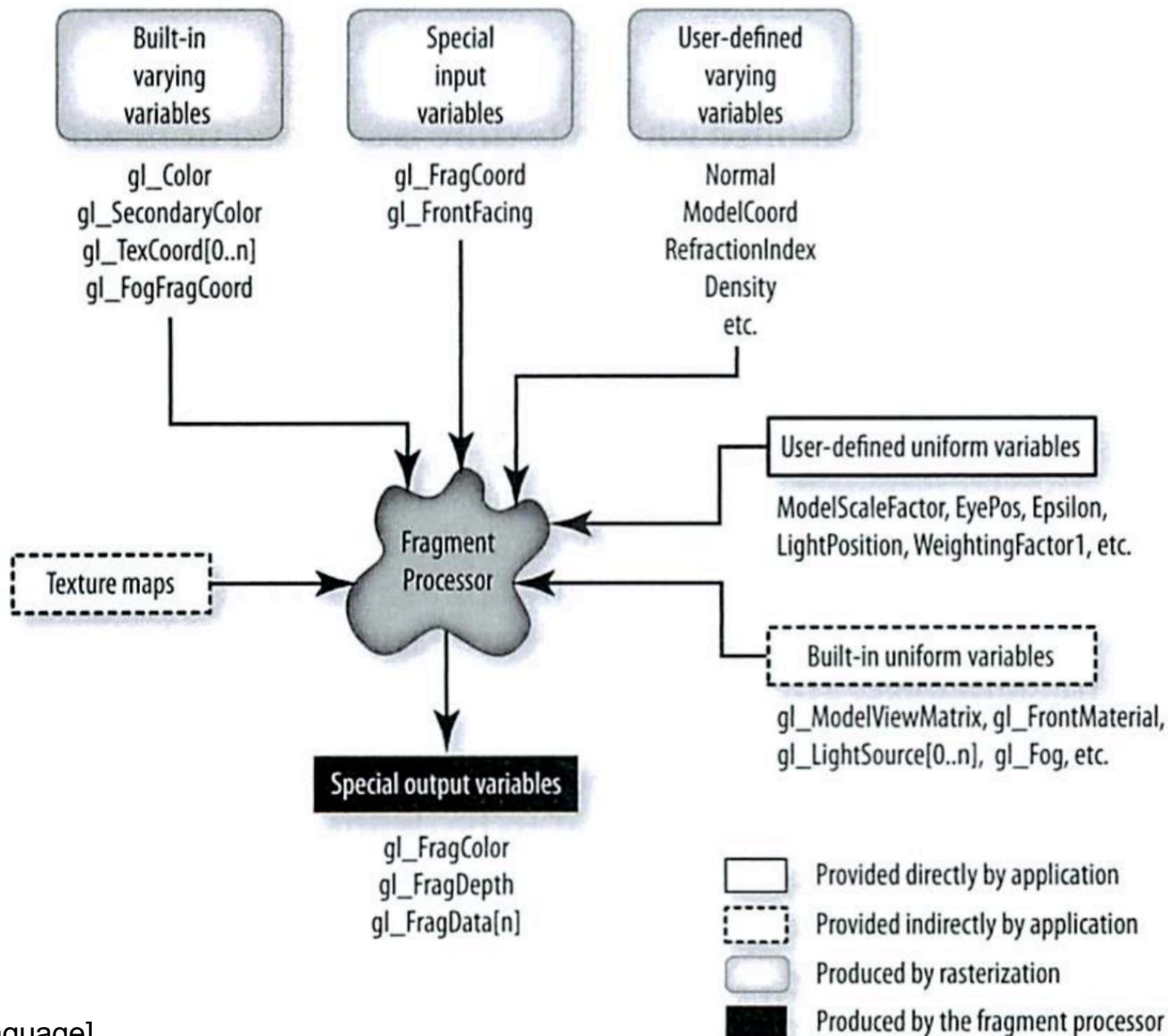
Fragment Shader

- Programmable Shader stage that will processes a Fragment generated by the Rasterization into a set of colors and a single depth value.



```
void main()
{
    vec4 C = vec4(0.8,0.1,0.1,1.0);
    gl_FragColor = C;
}
```

Fragment Shader Inputs and Outputs



Implementation

- Gouraud shading
 - Vertex shader

```
void main()
{
    point_position = modelViewMatrix * vec4( position, 1.0 );
    normal_vector = normalMatrix * normal;

    vec3 C = color;
    vec3 L = normalize( light_position - point_position.xyz );
    vec3 N = normalize( normal_vector );

    point_color = LambertianReflection( C, L, N );
    gl_Position = projectionMatrix * point_position;
}
```

Implementation

- Gouraud shading
 - Fragment shader

```
void main()
{
    gl_FragColor = vec4( point_color, 1.0 );
}
```

Implementation

- Phong shading
 - Vertex shader

```
void main()
{
    point_color = color;
    point_position = modelViewMatrix * vec4( position, 1.0 );
    normal_vector = normalMatrix * normal;

    gl_Position = projectionMatrix * point_position;
}
```

Implementation

- Phong shading
 - Fragment shader

```
void main()
{
    vec3 C = point_color;
    vec3 L = normalize( light_position - point_position.xyz );
    vec3 N = normalize( normal_vector );

    vec3 shaded_color = LambertianReflection( C, L, N );
    gl_FragColor = vec4( shaded_color, 1.0 );
}
```

Polling

- Take the poll
 - Student ID Number
 - Name