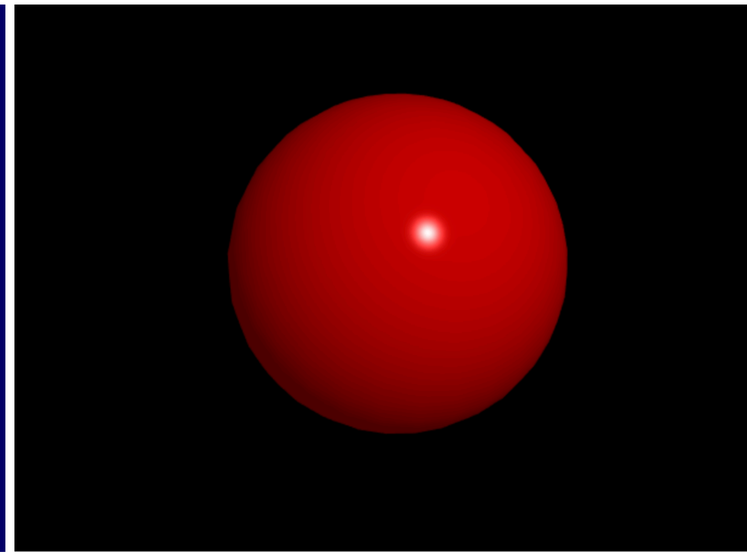
COSC342: Computer Graphics

2017





Lecture 15

ILLUMINATION

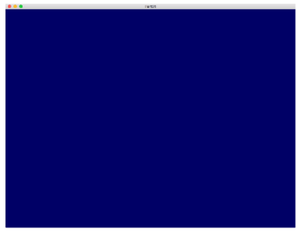
Stefanie Zollmann

LAST LECTURE

OpenGL Essentials

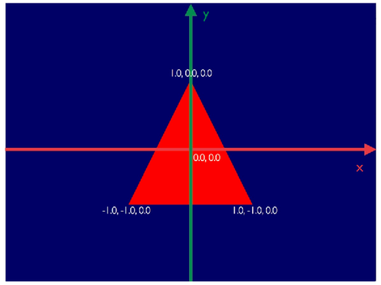
OPENGL CONTEXT

Context creation with GLFW (library for creation and management of windows with OpenGL contexts)



EXAMPLE: VERTEX BUFFER OBJECT

- Let's create our first triangle using a vertex buffer object



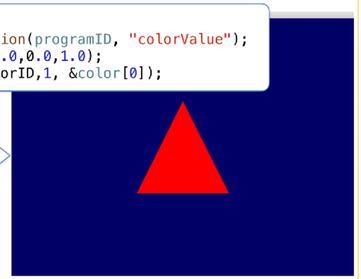

```
// Representation of the 3 vertices of
our triangle
//An array of 3 vectors each consisting
of x,y,z
    static const GLfloat data[] = {
        -1.0f, -1.0f, 0.0f,
        1.0f, -1.0f, 0.0f,
        0.0f,  1.0f, 0.0f,
    };
```


FRAGMENT SHADER

Simple Fragment Shader outputting specified colour:

```
#version 330 core

// Ouput data
out vec3 color;

uniform vec4 colorValue;
void main()
{
 // Output color = red
 color = colorValue.rgb;

}
```

```
// add color parameter to shader
GLint colorID = glGetUniformLocation(programID, "colorValue");
glm::vec4 color = glm::vec4(1.0,0.0,0,0.0,1.0);
glProgramUniform4fv(programID,colorID,1, &color[0]);
```



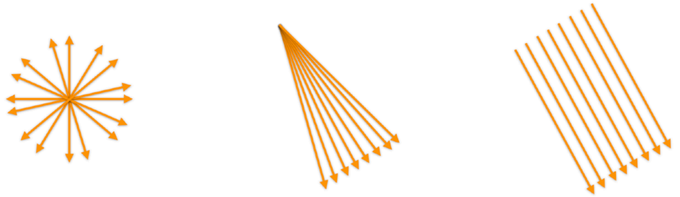
OpenGL Context

OpenGL Objects

Shaders

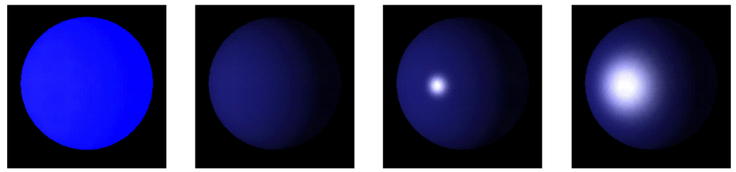
TODAY

LIGHT SOURCES



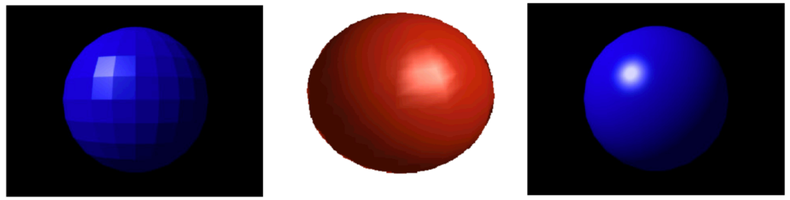
Point Light | Spot Light | Directional Light

REFLECTION MODEL



Ambient | Diffuse | Specular | Combined

SHADING MODELS



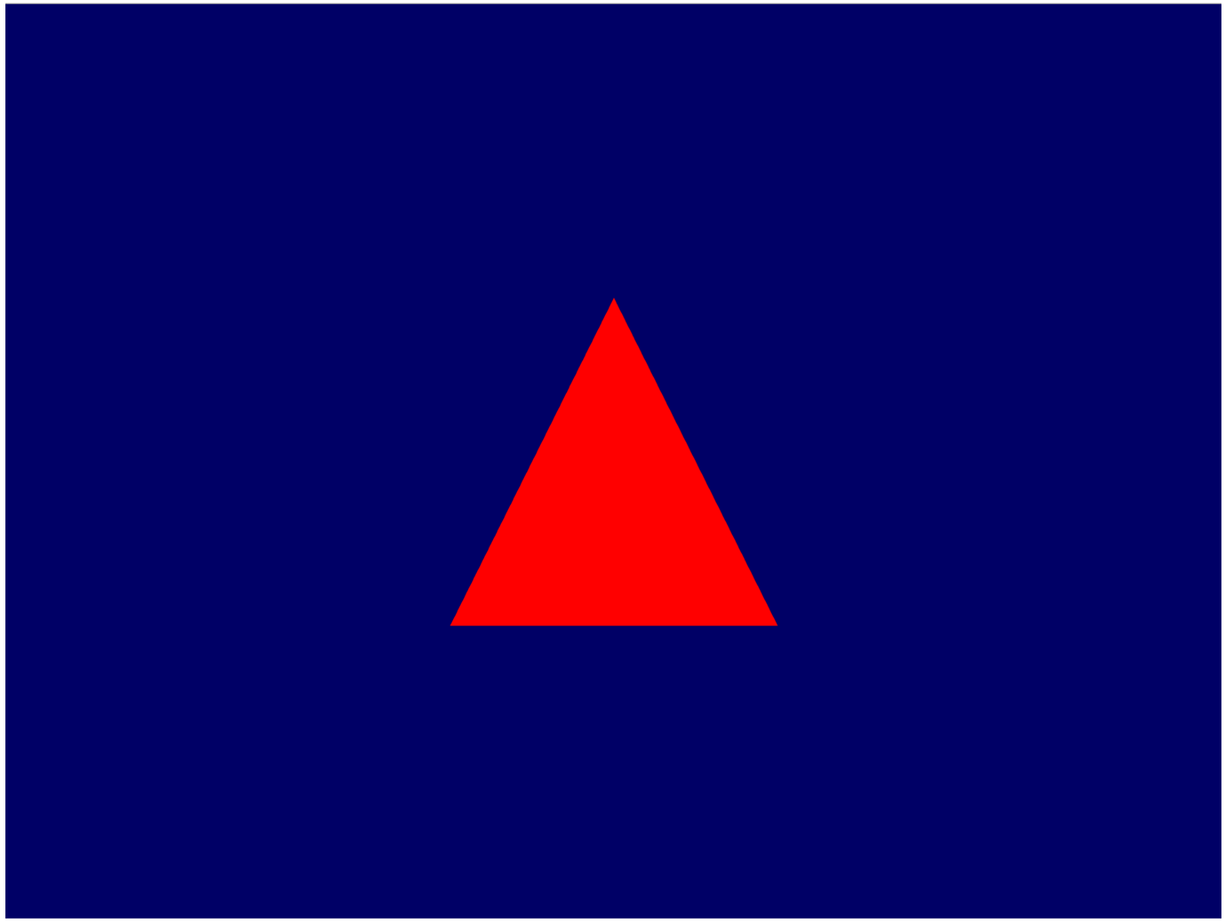
Flat Shading | Gouraud Shading | Phong Shading

Light Sources

Reflection Model

Shading

WHAT IS MISSING?



WHY ILLUMINATION?

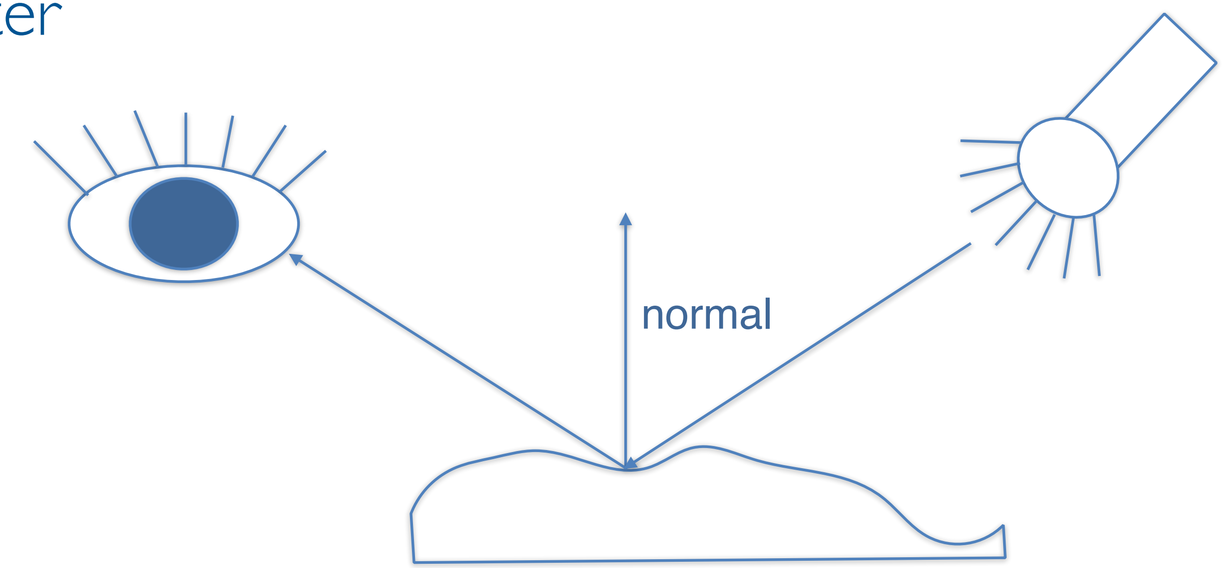
- Illumination is important for perception and understanding of 3D scenes
- Has visual cues for humans
- Provides information about
  - Positioning of light sources
  - Characteristics of light sources
  - Materials
  - Viewpoint

ILLUMINATION MODEL

- Can be complex
- Equation for computing illumination
- Includes:
  - Light attributes (intensity, colour, position, direction, shape)
  - Surface attributes (colour, reflectivity, transparency)
  - Interaction between lights and objects
- General rendering equation
  - Introduced 1986 by Kajiya
  - Global illumination model

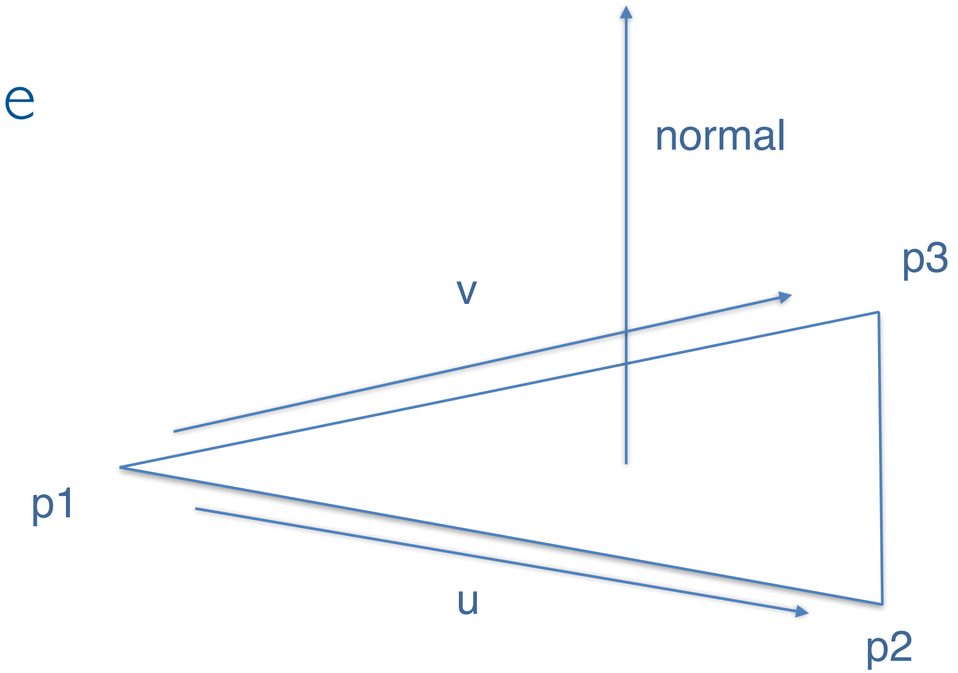
LOCAL ILLUMINATION MODEL

- OpenGL cannot render full global illumination
- We need an simplified approximation
- Local illumination model
  - Does not consider light reaching after bouncing off other objects
- Function of:
  - Viewer position
  - Light source
  - Surface material properties
  - Geometry

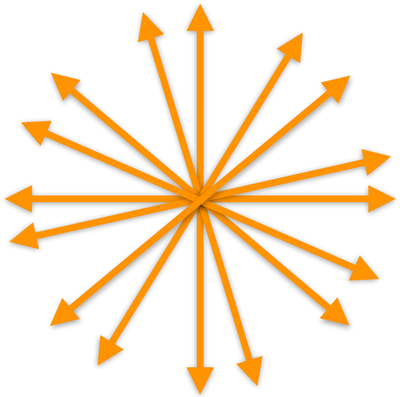


NORMAL

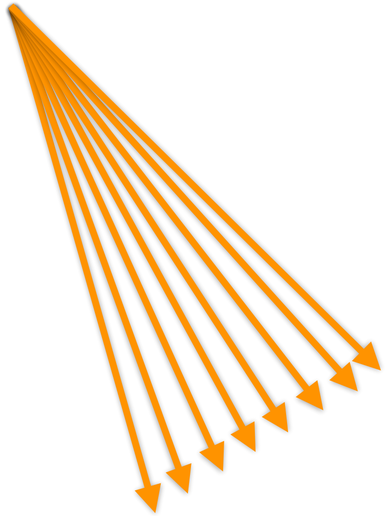
- Perpendicular to tangent plane of surface
- For triangles:
  - Cross product of two edges of that triangle
  - $n = u \times v$
  - $u = p2 - p1$
  - $v = p3 - p1$
  - $n_x = u_y v_z - u_z v_y$
  - $n_y = u_z v_x - u_x v_z$
  - $n_z = u_x v_y - u_y v_x$



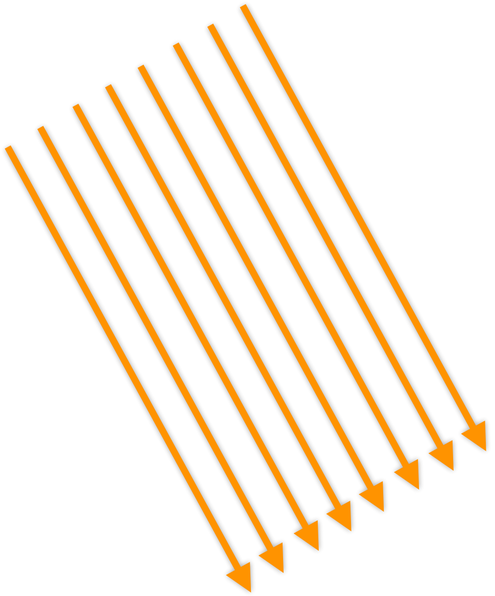
LIGHT SOURCES



Point Light



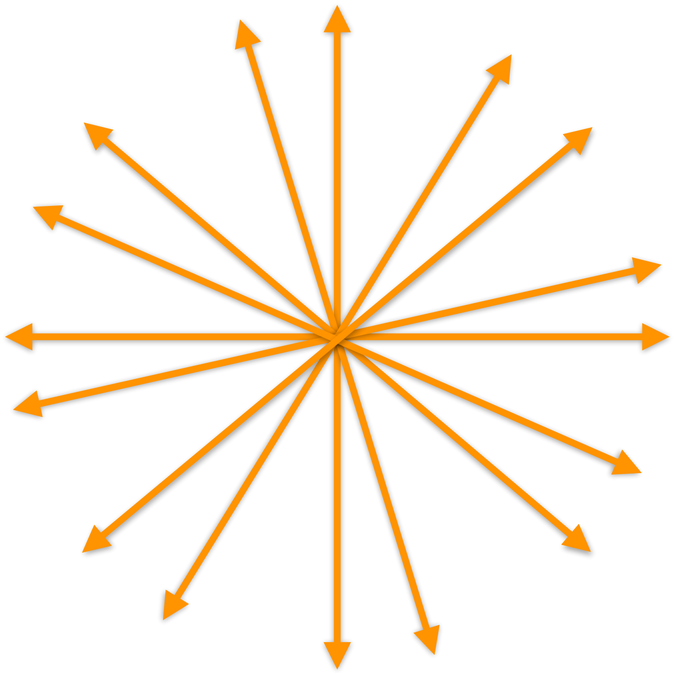
Spot Light



Directional Light

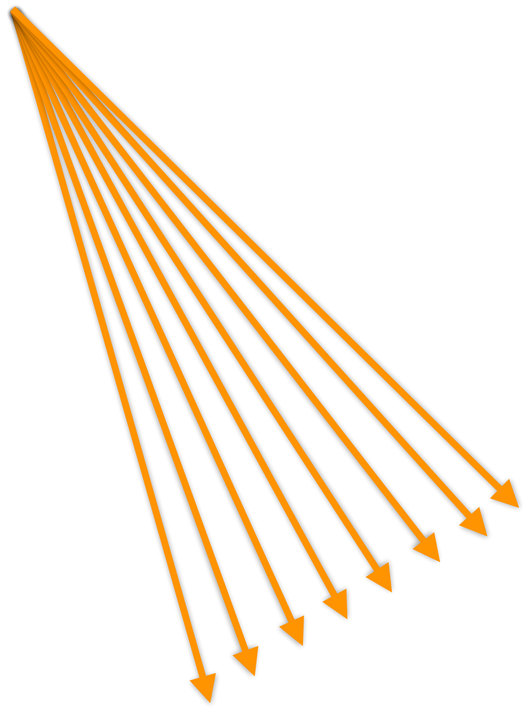
POINT LIGHT

- Starts at one point and spreads out in all directions
- Defined by position
- Intensity decreases with the square of distance
- Direction is different at each vertex (light direction = light position – vertex position)
- Example: light bulb



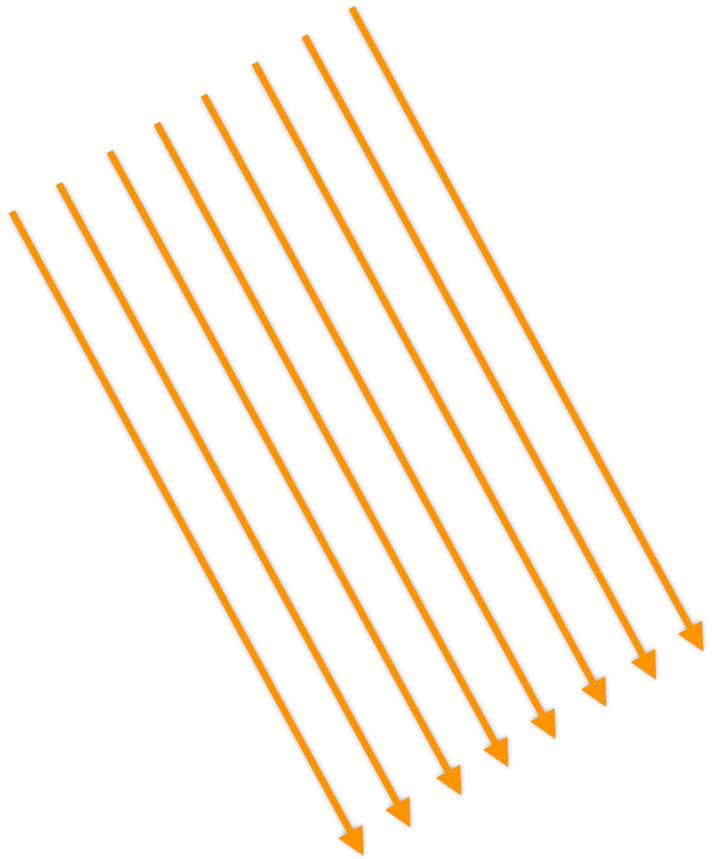
SPOT LIGHT

- Light starts at one point and spreads out as cone with defined angle
- Described by position, direction and width of beam
- Useful for dramatic light effects (e.g. theatre spot light)

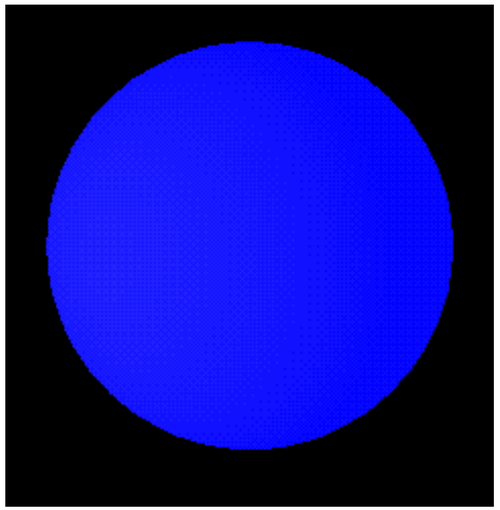


DIRECTIONAL LIGHT

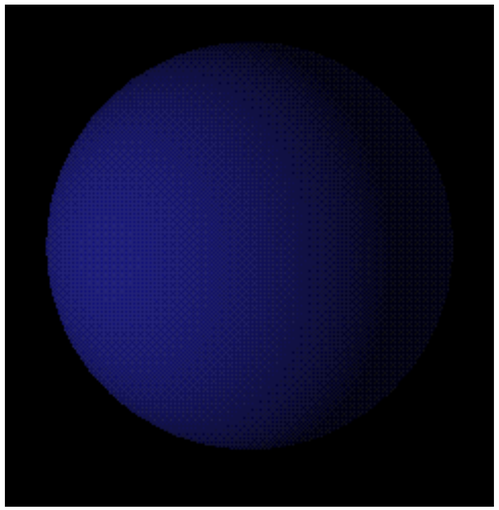
- Described by direction only
- No position
- Direction is same for all points
- Used for light sources that are infinitely far away
- Intensity does not change depending on distance
- Used for modelling sun light



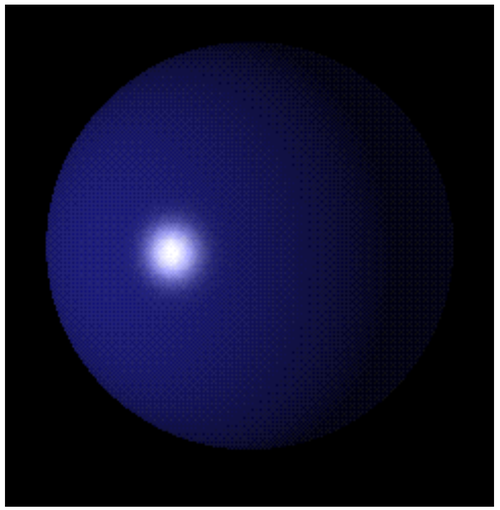
REFLECTION MODEL



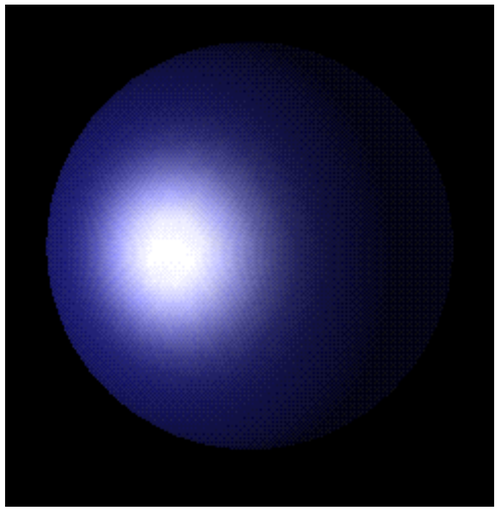
Ambient



Diffuse



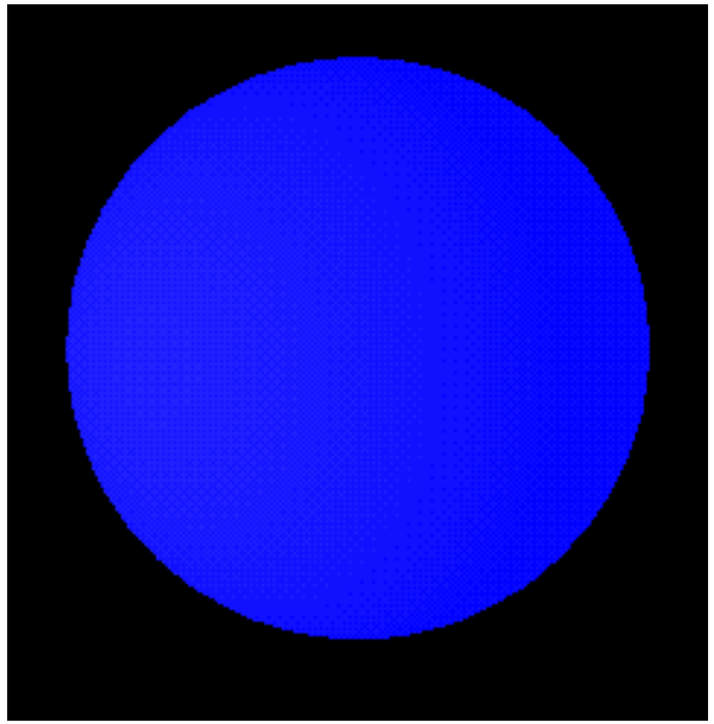
Specular



Combined

AMBIENT COMPONENT

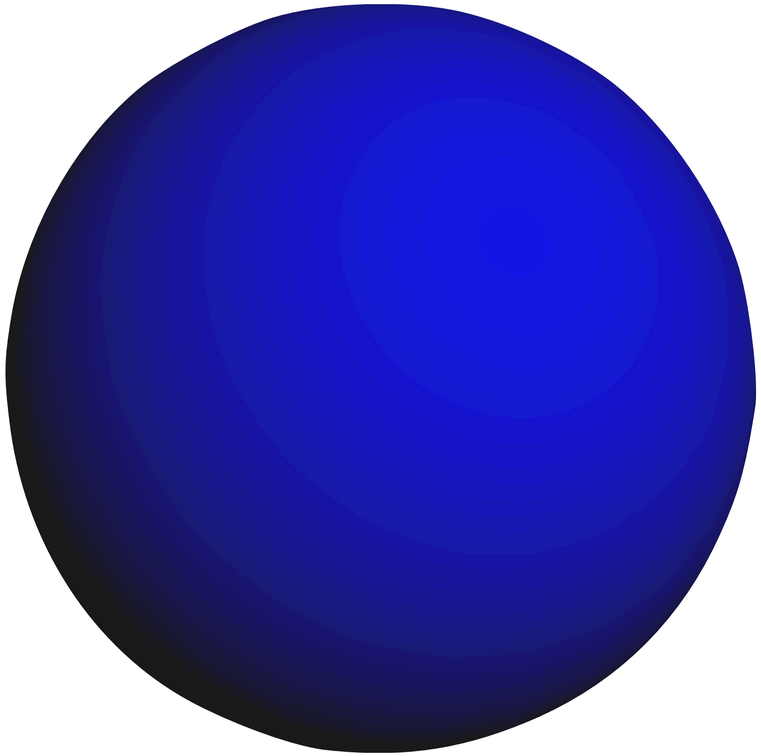
- Indirect illumination from light that has been reflected multiple times
- Does not come from a specific direction
- "Base" lighting
- Consists of:
  - Ambient light component I_a
  - Ambient material factor k_a



$I_{ambient} = I_a k_a$

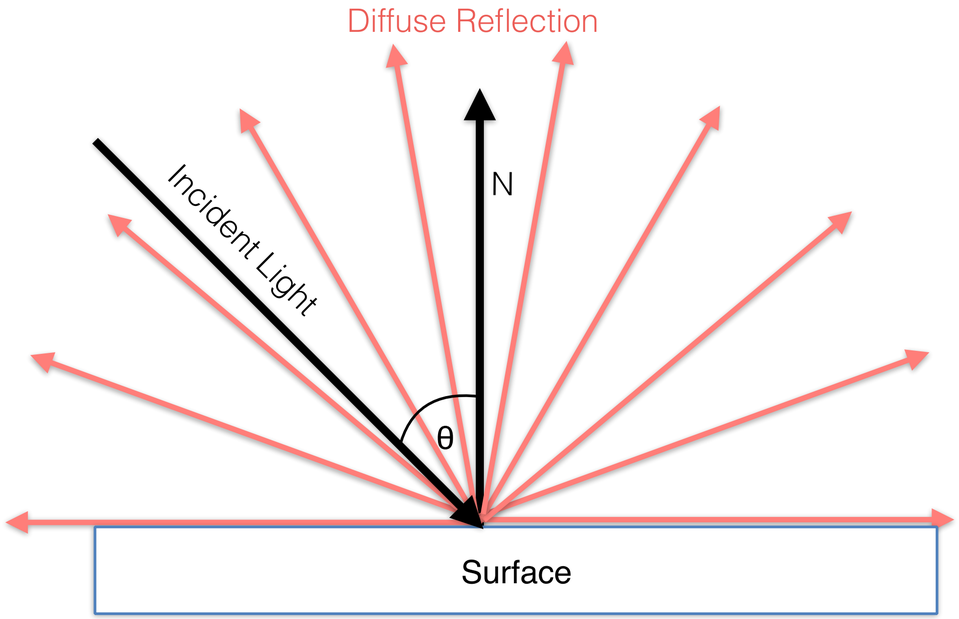
DIFFUSE COMPONENT

- Also called Lambertian reflection
- Ideal diffuse surface reflects light equally in all directions
- Incident ray is reflected in many directions
  - Independent of view angle (reflects equally in all directions)
  - But dependent on direction of incoming light (angle between normal N and incident light L : angle of incidence θ)



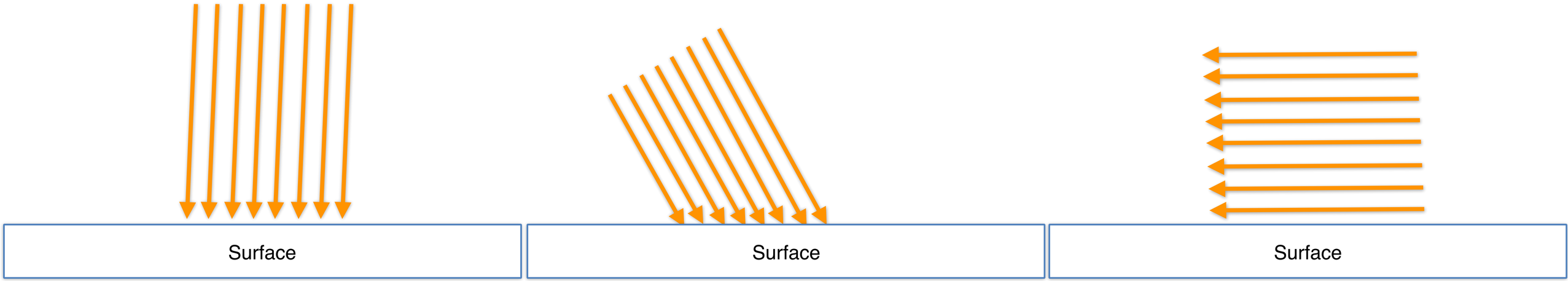
DIFFUSE COMPONENT

- Also called Lambertian reflection
- Ideal diffuse surface reflects light equally in all directions
- Incident ray is reflected in many directions
  - Independent of view angle (reflects equally in all directions)
  - But dependent on direction of incoming light (angle between normal N and incident light L : angle of incidence θ)



DIFFUSE COMPONENT

- Incoming light rays with perpendicular angle to the surface reflect more light
- The larger the angle θ between normal and incoming light rays, the less light is reflected



θ = 0°: Maximum Brightness

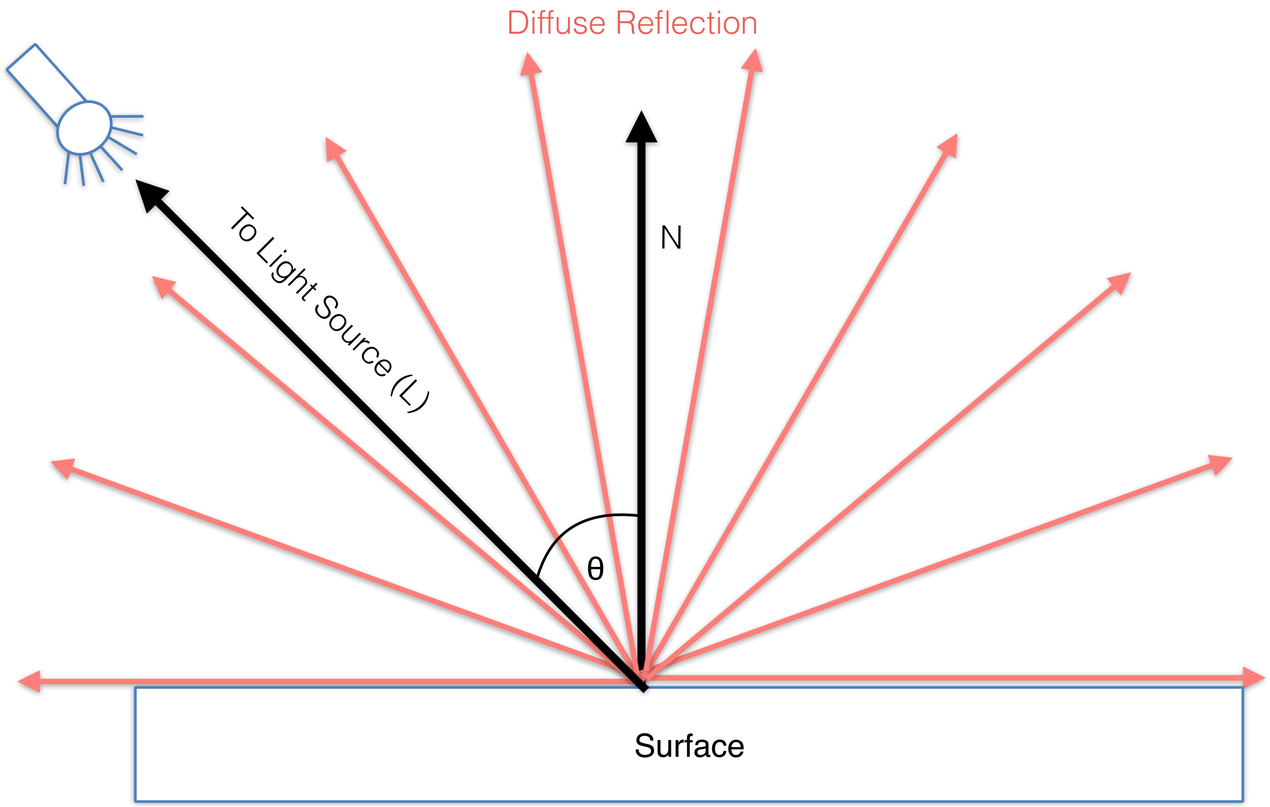
θ = 45°:

θ = 90°: Dark

DIFFUSE COMPONENT

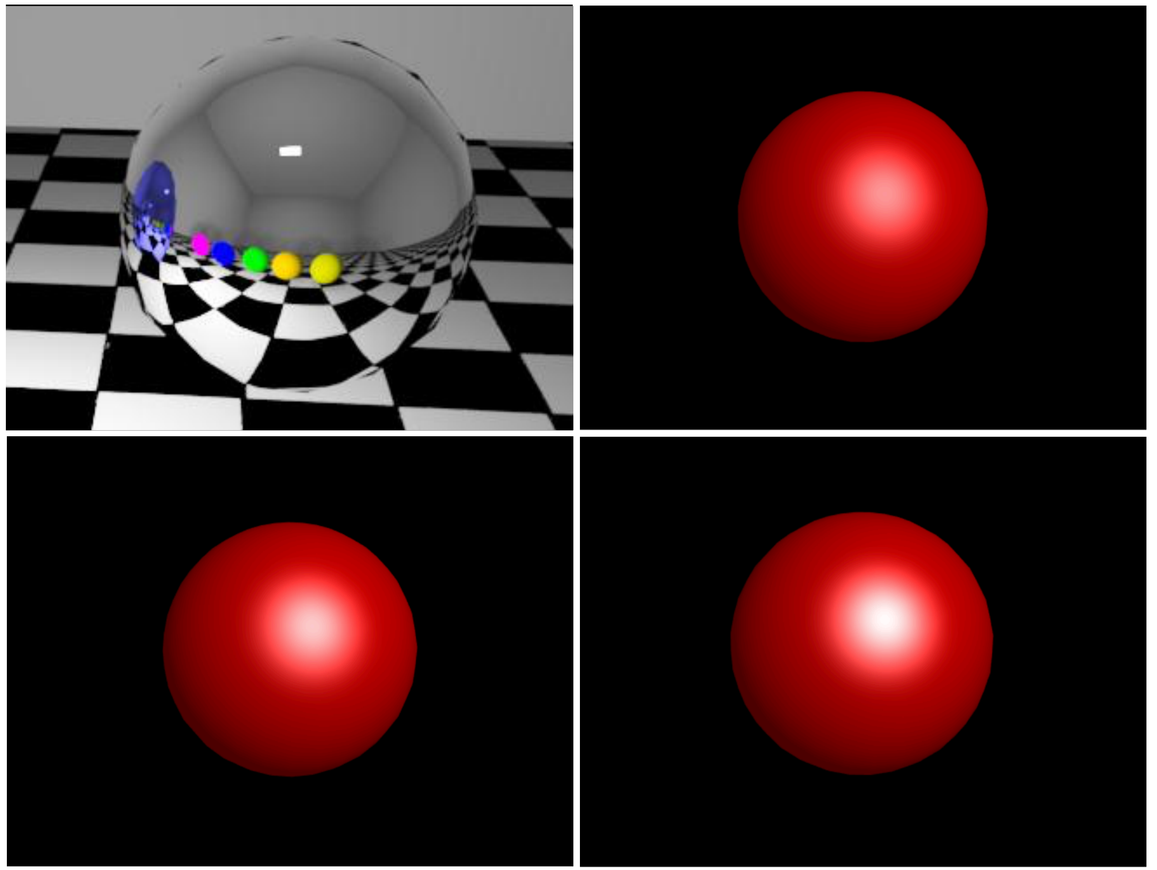
$I_{diffuse} = I_d \, k_d \cos\theta = I_d \, k_d \, (N \text{ dot } L)$

- Diffuse light component I_d
- Diffuse material factor k_d
- Light direction l
- Surface normal n



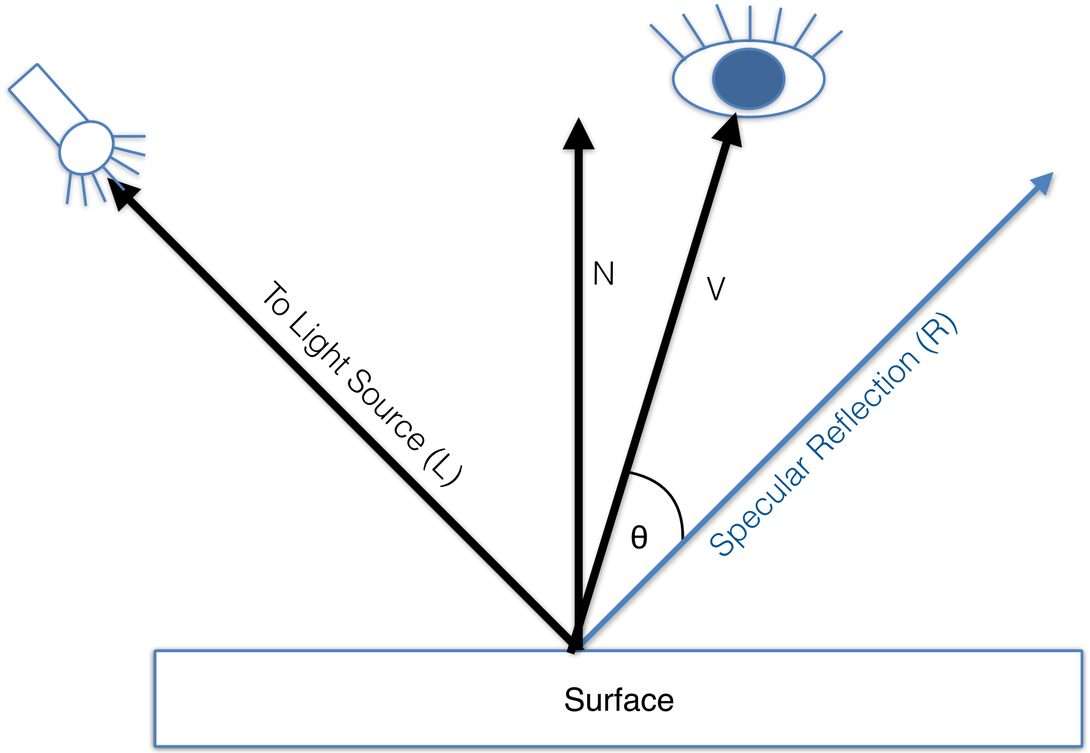
SPECULAR COMPONENT

- Simulates highlights from shiny objects
- Called specular highlight
- For ideal reflectors: angle of incidence equals angle of reflection (only visible of R equals V)
- For non-perfect reflectors: highlight is visible of a range of angles



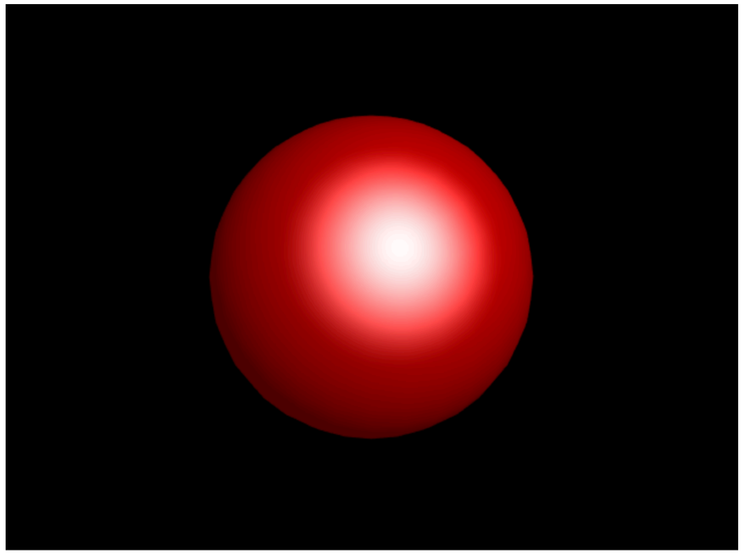
SPECULAR COMPONENT

- Consists of:
  - Direction to light source
  - Reflected ray R = 2(N dot L) N-L (in GLSL using reflect method)
  - Specular material factor k_s
  - Specular exponent n (the larger, the smaller the highlight)
  - k_s and n_s have no physical meaning (a lot of tweaking required to achieve desired result)

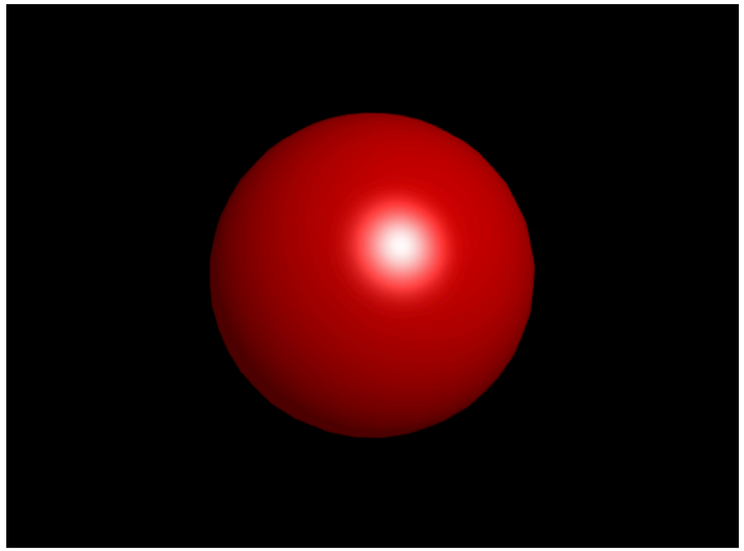


$$I_{specular} = I_s \, k_s \, (R \text{ dot } V)^{ns}$$

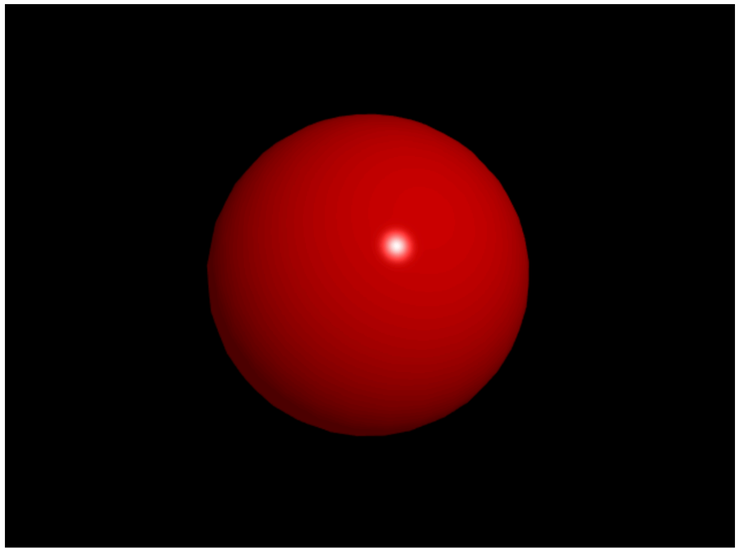
SPECULAR COMPONENT



$n_s = 2.0$



$n_s = 10.0$



$n_s = 100.0$

REFLECTION MODELS

- Combine components
- Different ways to do the computation
  - **Phong**
  - **Blinn-Phong**
  - Cook-Torrance
  - Oren-Nayar

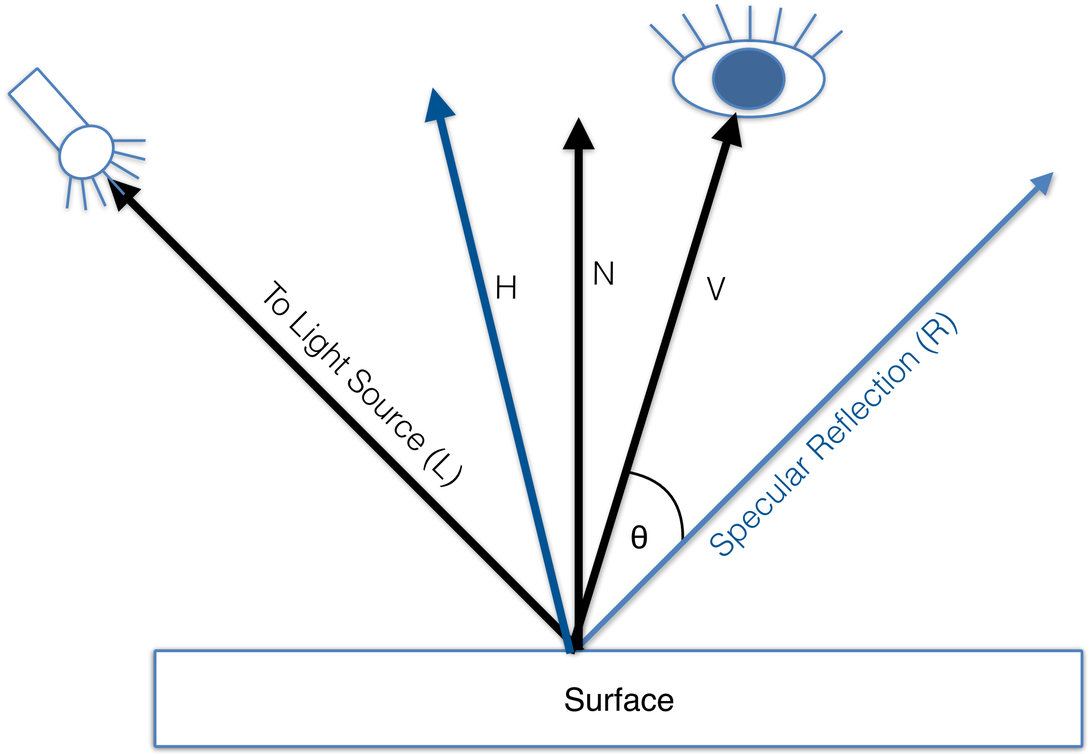
PHONG REFLECTION MODEL

Illumination = $I_{ambient} = I_a k_a$ $I_{diffuse} = I_d k_d (N \text{ dot } L)$ $I_{specular} = I_s k_s (R \text{ dot } V)^{ns}$

$$\text{Illumination} = I_a k_a + I_d k_d (N \text{ dot } L) + I_s k_s (R \text{ dot } V)^{ns}$$

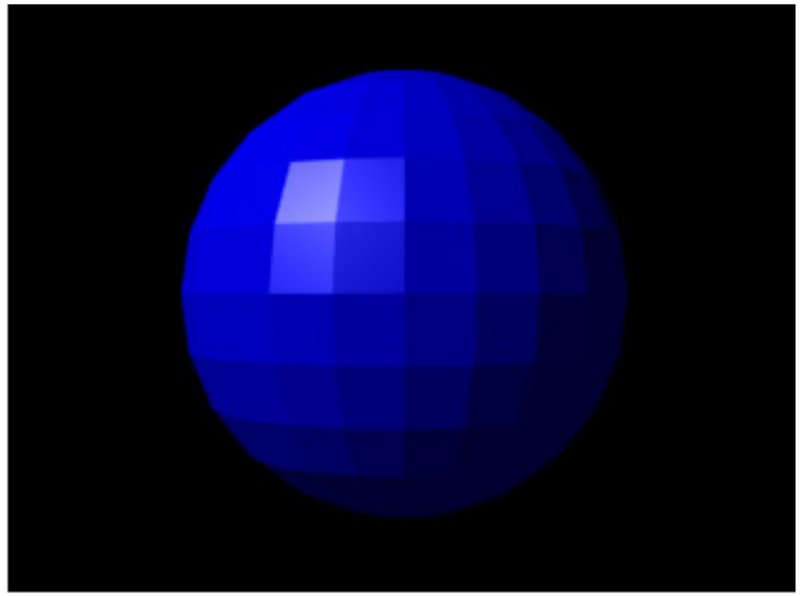
BLINN-PHONG REFLECTION MODEL

- Modification of phong reflection model by Jim Blinn
- Phong requires to recalculate the dot product (R dot V)
- Blinn-Phong uses halfway vector (H) between the viewer and light-source vectors
- H = normalize(L + V) = (L+V)/(|L+V|)

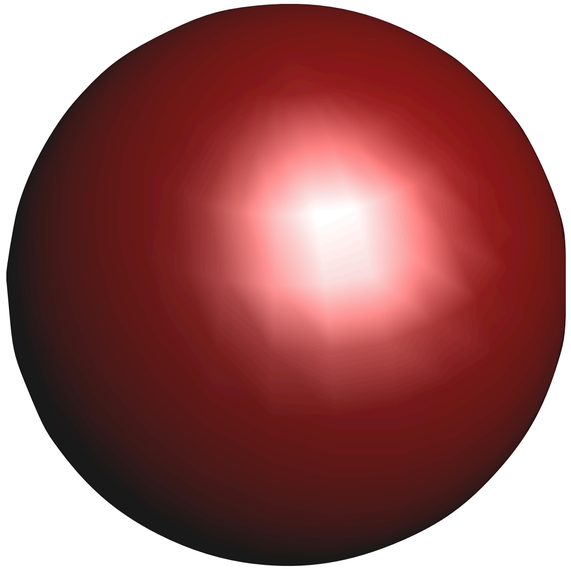


$I_{specular} = I_s k_s (H \text{ dot } N)^{ns}$

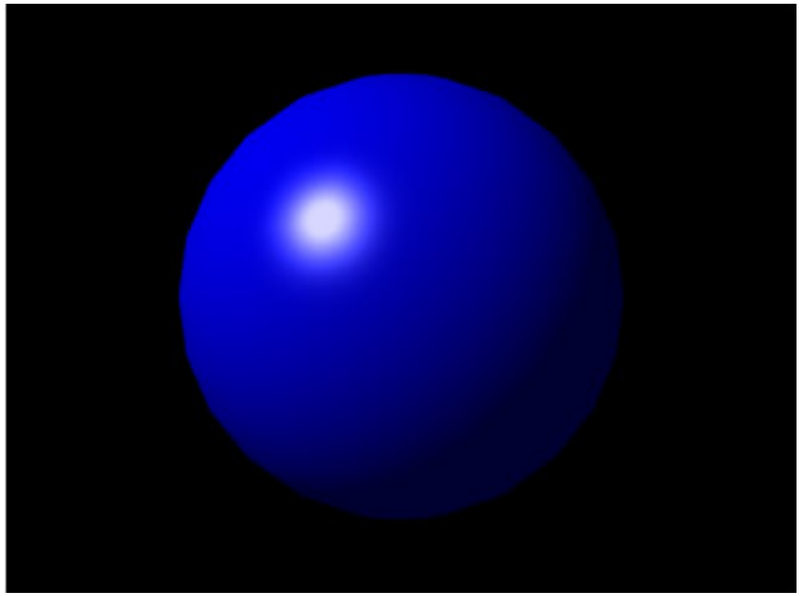
SHADING MODELS



Flat Shading



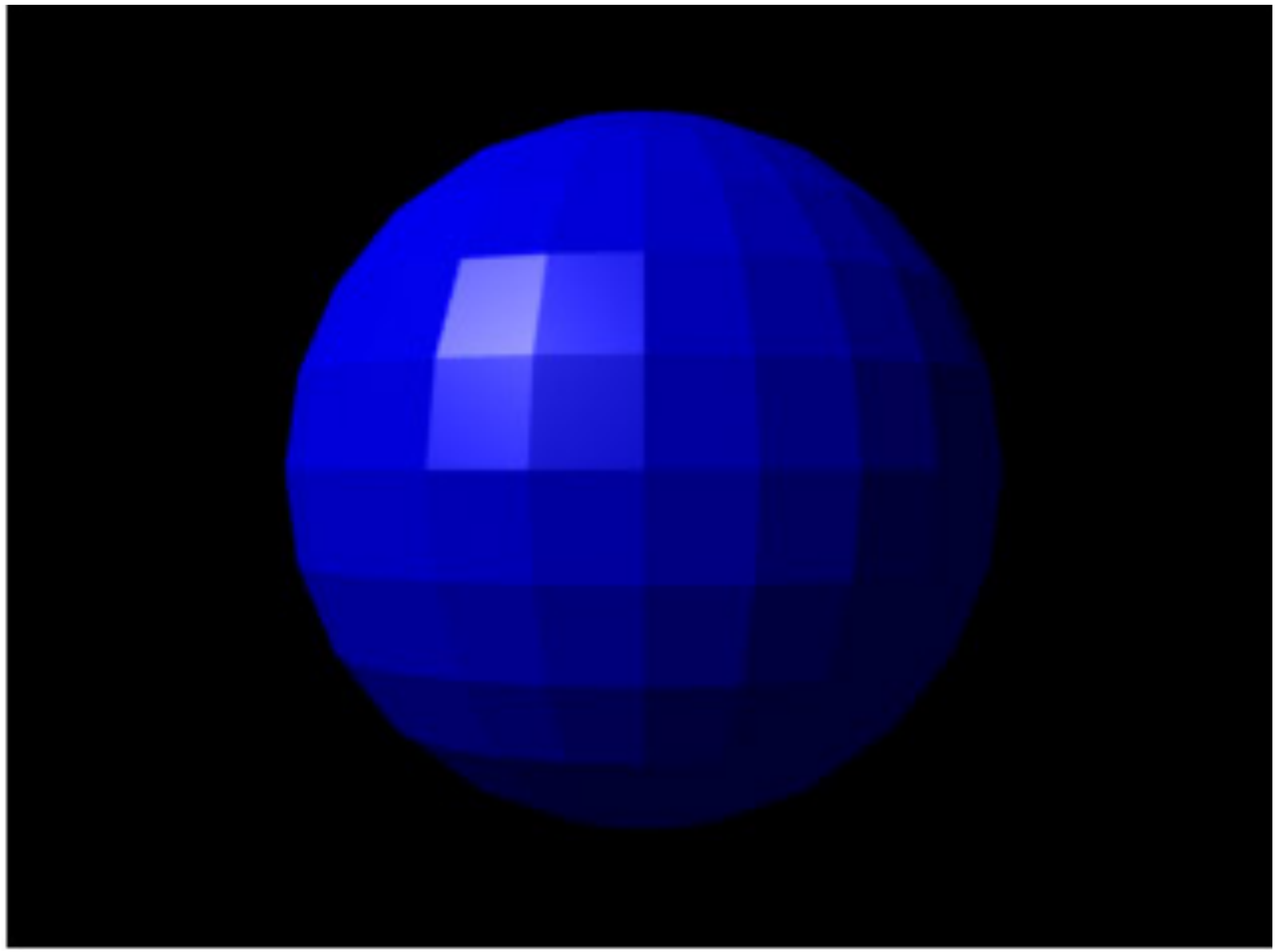
Gouraud Shading



Phong Shading

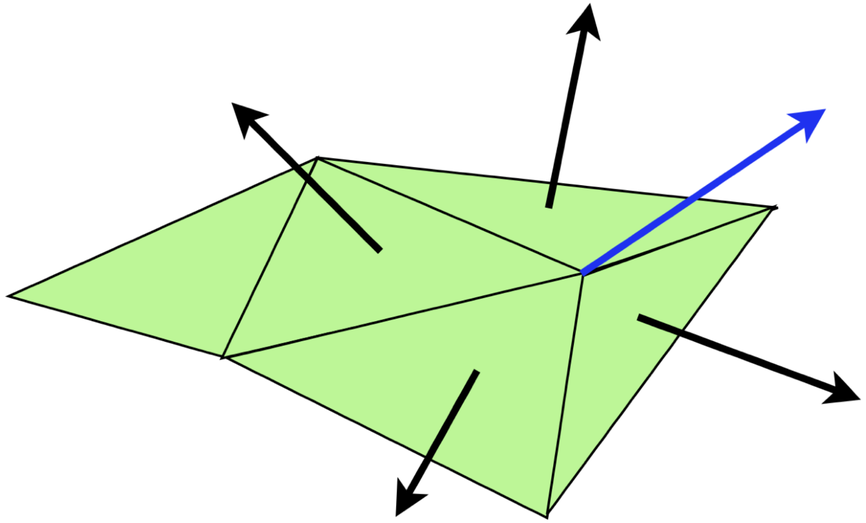
FLAT SHADING

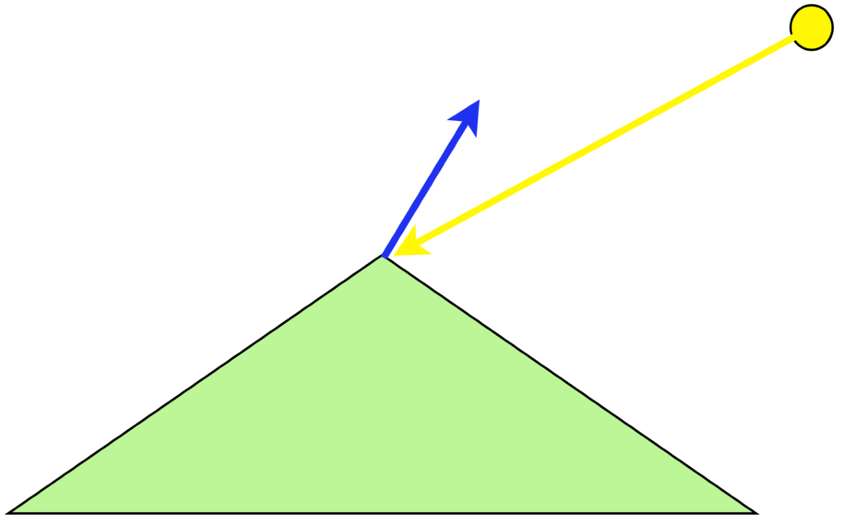
- Per polygon
- Was used for high speed rendering
- All vertices of one polygons have the same colour
- Difference between polygons
- No smooth transitions



GOURAUD SHADING

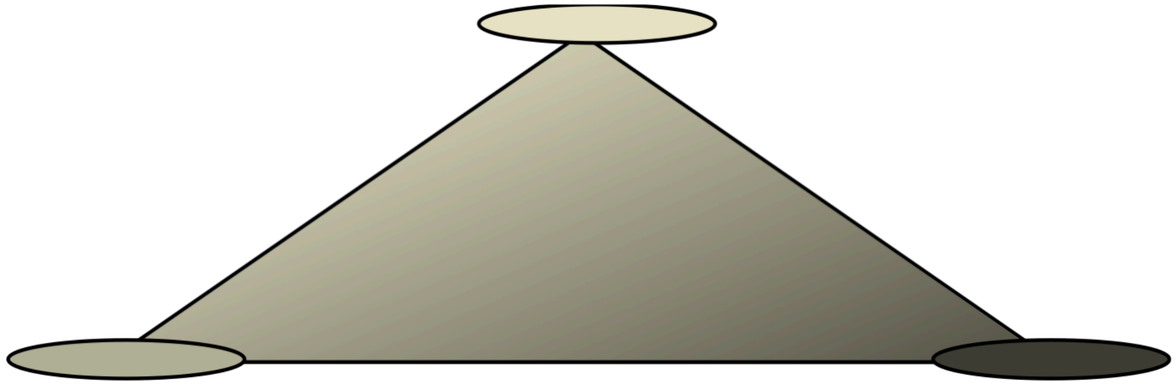
- Per vertex
- Interpolative shading
- Calculate polygon vertex colour
- Interpolate colours for interior points





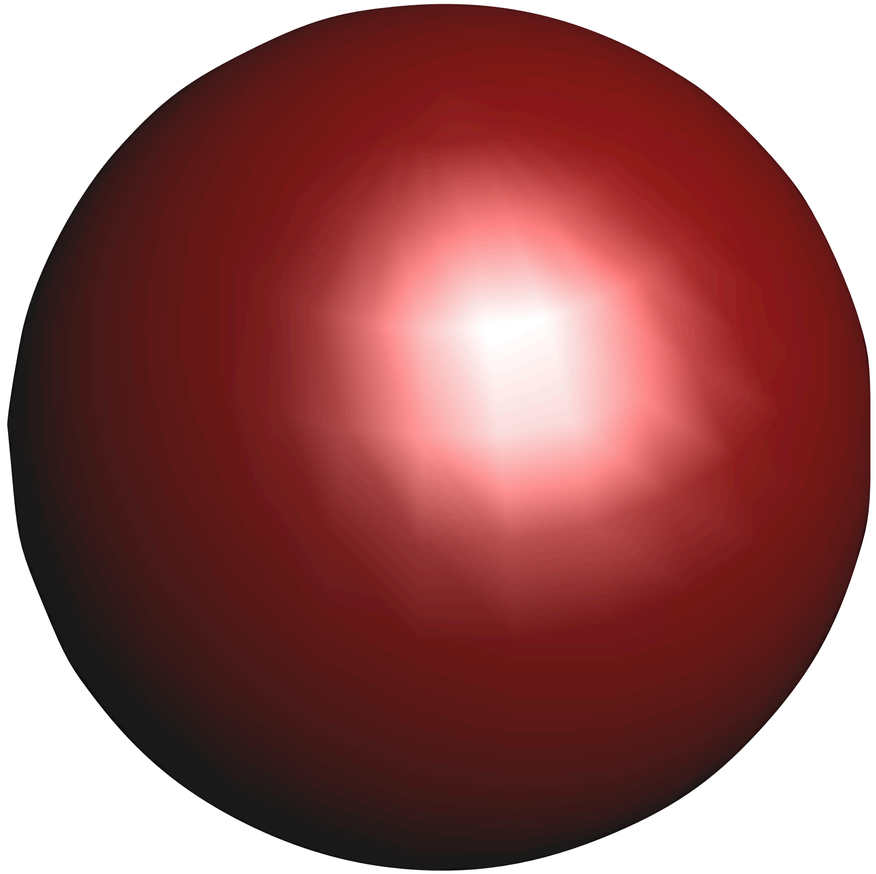
GOURAUD SHADING

- Per vertex
- Interpolative shading
- Calculate polygon vertex colour
- Interpolate colours for interior points



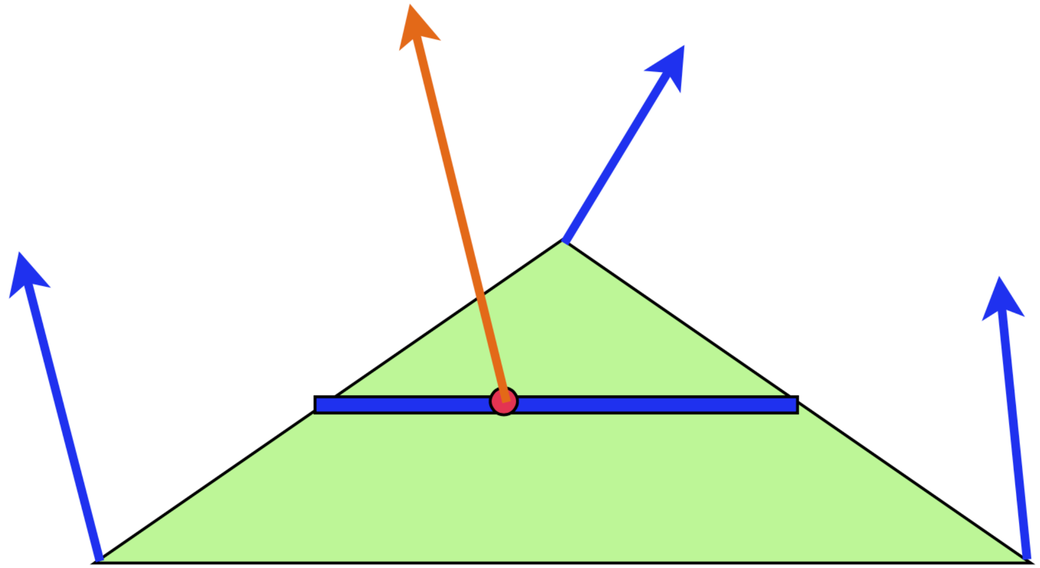
GOURAUD SHADING

- Per vertex
- Interpolative shading
- Calculate polygon vertex colour
- Interpolate colours for interior points



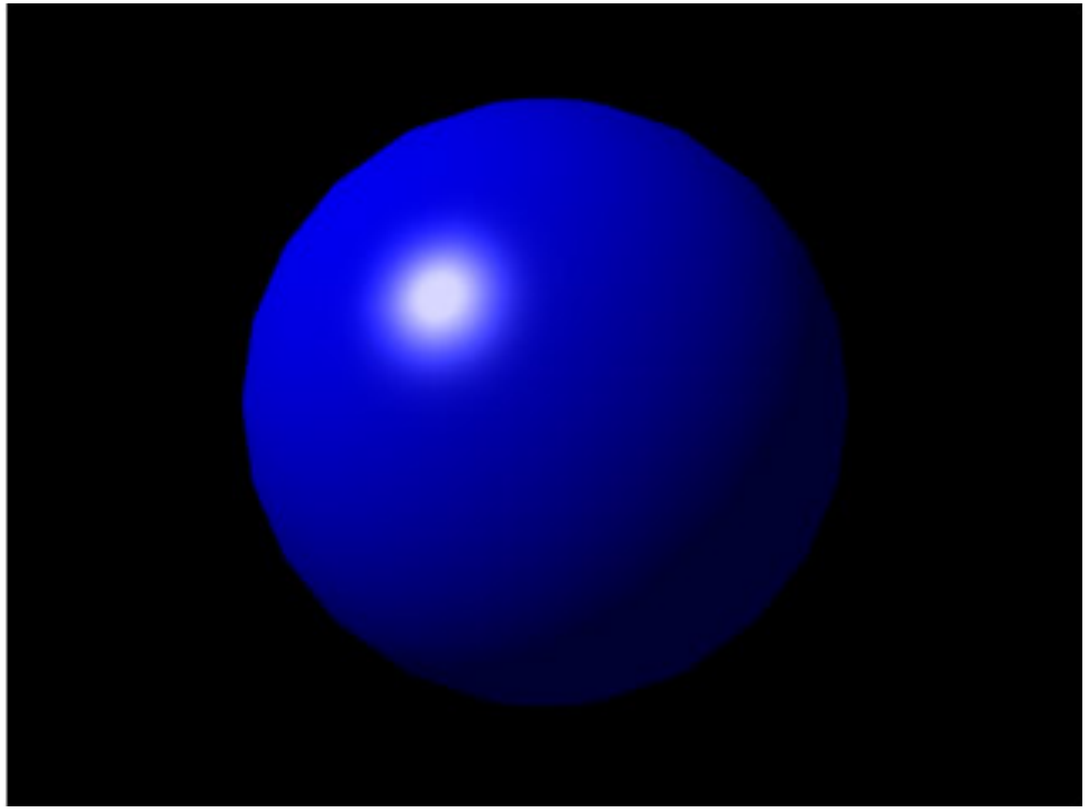
PHONG SHADING

- This is NOT Phong illumination
- Per fragment
- Interpolates the surface normals instead of the intensity values
- Then do calculation of intensities using the interpolated normal
- Gives better results, especially for highlights



PHONG SHADING

- This is NOT Phong illumination
- Per fragment
- Interpolates the surface normals instead of the intensity values
- Then do calculation of intensities using the interpolated normal
- Gives better results, especially for highlights



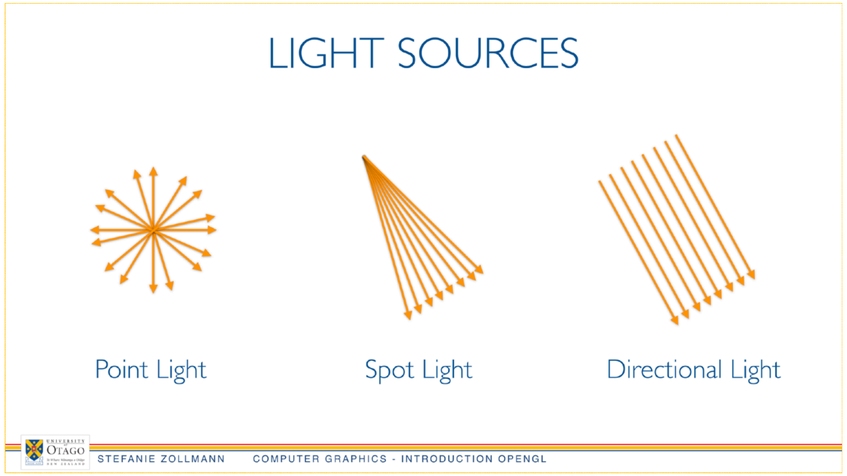
MATERIAL DEFINITIONS

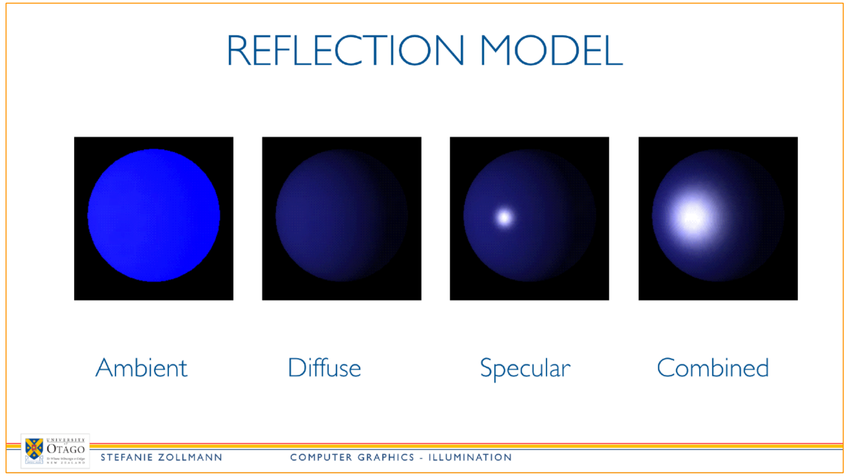
- Material Template Library (MTL) can be used to define material settings
- Defines ambient (Ka), diffuse (Kd), specular (Ks) colours and the specular exponent (Ns)
- Also allows to define opacity (d) - 1.0 means fully opaque
- Set texture maps (map_Kd)

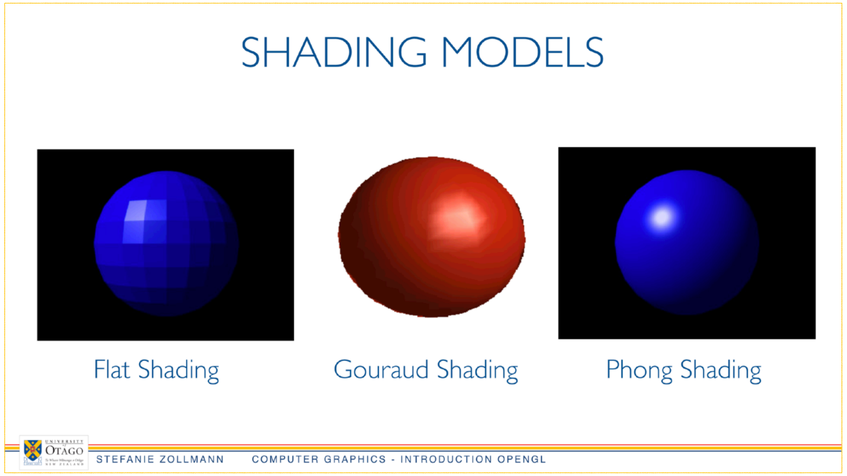
```
newmtl EarthMaterial

Ka 0.640000 0.640000 0.640000
Kd 0.640000 0.640000 0.640000
Ks 0.050000 0.050000 0.050000
Ns 30.0000
d 0.5
illum 2
map_Kd ColorMap.bmp
```

Example

SUMMARY







Light Sources

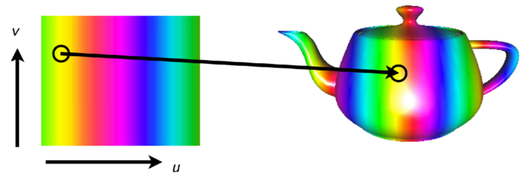
Reflection Model

Shading

WHAT'S NEXT

TEXTURE MAPPING

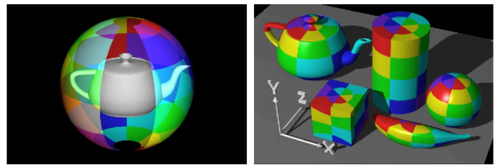
- Process of finding u,v coordinates for each vertex



Texture Mapping

PARAMETRISATION

- Spherical mapping
  - (x,y,z) value of a point is converted into spherical coordinates
  - Wraps texture around the object

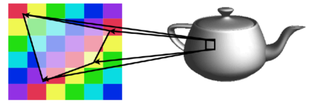


Rosalee Wolfe: Teaching Texture Mapping Visually

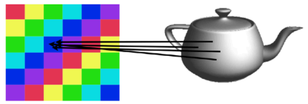
Parametrisation

CHALLENGES

- Undersampling: one pixel maps to an area covering many texture pixels (texels).



- Oversampling: many pixels map to an area contained by only one texel.



Challenges

Thank You!

For more material visit

http://www.cs.otago.ac.nz/cosc342/