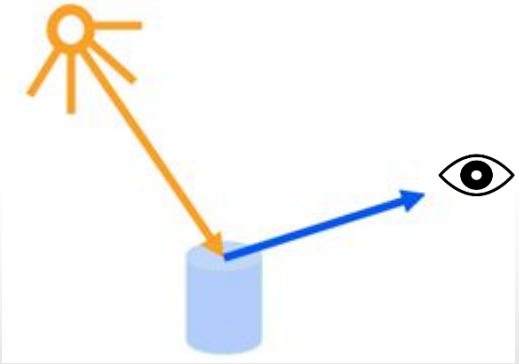
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

CG Basics VI
Illumination

Illumination Models

- In order to produce realistic images, we must simulate the appearance of surfaces under various lighting conditions

- Illumination model
  - Computes the intensity of the light that is reflected at a given point of the surface

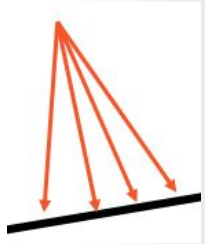


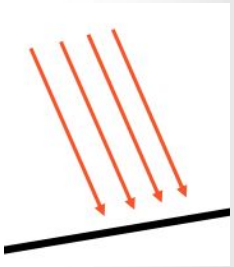
Illumination Models

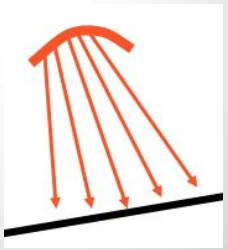
- Lighting effects are determined by the interaction of light sources with object surfaces
- In order to simulate them, we must define
  - Light source properties
    - Position
    - Intensity for each frequency (color)
    - Directional distribution
  - Surface properties (material)
    - Reflectance for each frequency (color)
    - The absorbed and transmitted light by a surface are not considered

Light Sources

- **Three types of light sources are considered**
  - **Point source**
    - All light rays originate at a point and radially diverge
  - **Parallel source**
    - Light rays are parallel
    - May be modelled as a point source at infinite distance (the sun)
  - **Distributed source**
    - All rays originate at a finite area in space
    - It models a nearby source (fluorescent light)



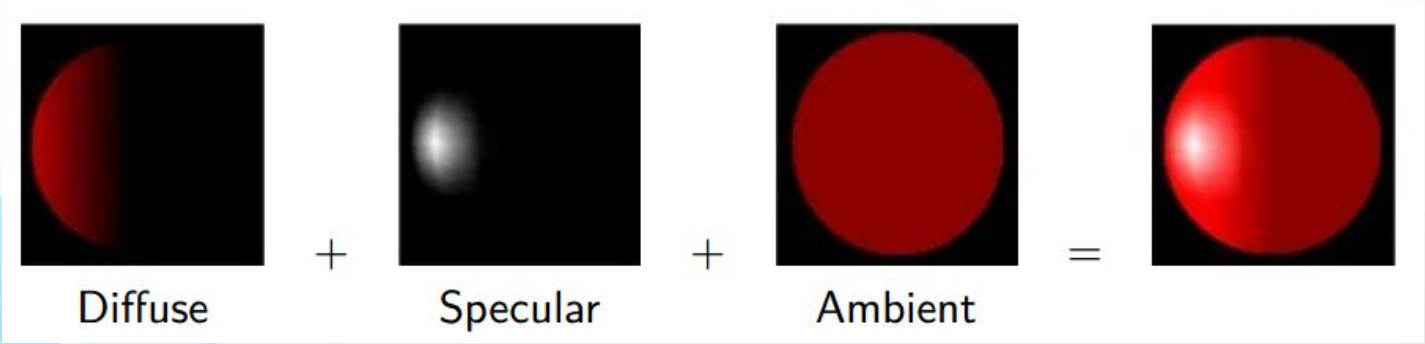




Phong Model

- Light's emitted intensity can be split into 3 components
  - Ambient intensity: background illumination
  - Diffuse intensity : non-shiny illumination
  - Specular intensity: shiny highlights

- Each one consists of 3 color components (color of the light)



Phong Model

- Surface (material) properties are captured by reflectivity coefficient vectors $K = (k_r, k_g, k_b)$
  - $K_a \rightarrow$ Ambient coefficient
    - Fraction of ambient light that is reflected by the surface
  - $K_d \rightarrow$ Diffuse coefficient
    - Fraction of diffuse light that is reflected by the surface
  - $K_s \rightarrow$ Specular coefficient
    - Fraction of specular light that is reflected by the surface
- And the shininess s of the material

Phong Model

- **Ambient illumination**
  - Environmental light
  - Light reflected or scattered from other objects in the scene
  - Accurate simulation is computationally expensive!

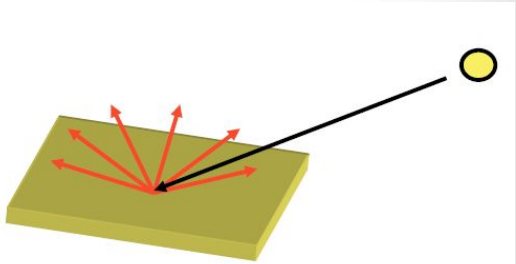
- **Simple approximation**

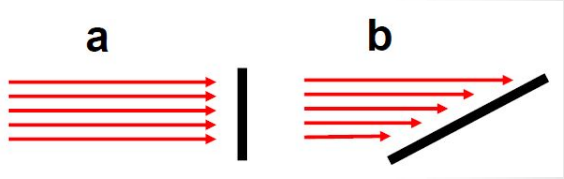
$$I_{amb} = K_a \, I_a$$

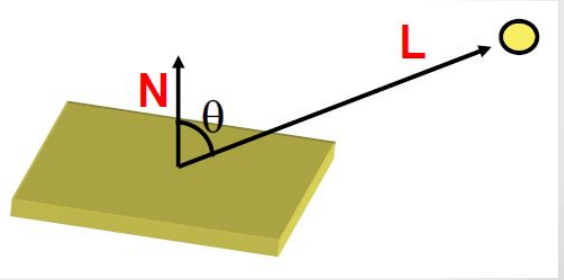
$K_a \in [0,1] \rightarrow$ Surface ambient coeff.
$I_a \rightarrow$ Ambient light intensity

Phong Model

- **Diffuse reflection**
  - When light hits an object with a rough surface, it is reflected in all directions
  - The brightness at each point is proportional to $cos\theta$
    - A surface perpendicular to the light direction *a*, is more illuminated than a surface *b* at n oblique angle
  - The surface appears equally bright from all viewing directions







Phong Model

- **Diffuse reflection**

$$I_{diff} = K_d \, I_p \, cos\theta = K_d \, I_p (N \cdot L)$$

$K_d \in [0,1] \rightarrow$ Surface diffuse coefficient
$I_p \rightarrow$ Point light intensity
$N \rightarrow$ Surface normal
$L \rightarrow$ Light direction

If N and L have unitary length $\rightarrow cos\theta = (N \cdot L)$!!!

Phong Model

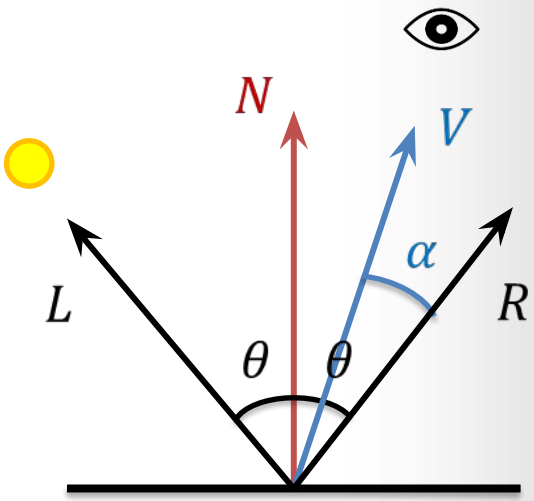
- **Diffuse reflection**
  - Commonly there are two type of light in a scene
    - A background ambient light
    - A point ligh source
  - The equation that combines both models is

$$I = I_{amb} + I_{diff} = K_a I_a + K_d I_p (N \cdot L)$$

  - Note that this equation is just for one color channel, it should be replicated for each one $I = (I_r, I_g, I_b)$

Phong Model

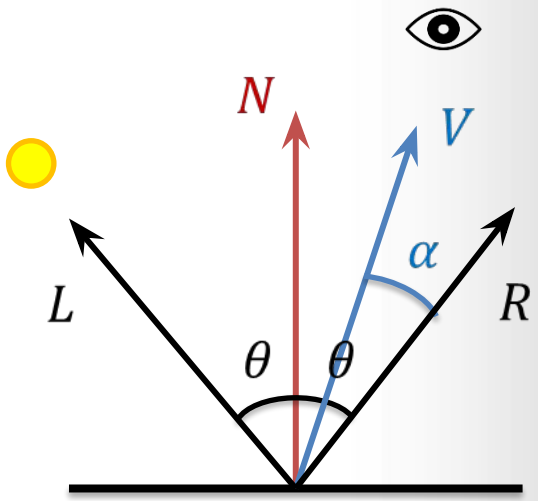
- **Specular reflection**
  - Models shiny and glossy surfaces with highlights
  - Light comes from a particular direction
  - Reflectance intensity changes with reflected angle
  - An ideal specular surface (mirror) reflects light exclusively in one direction R
  - Glossy objects are not ideal mirrors and reflect in the immediate vicinity of R



Phong Model

- **Specular reflection**
  - **Reflected specular intensity falls off as some power (shininess of the surface s) of $cos\ \alpha$**

$$I_{spec} = K_s\, I_p\, cos^s\alpha = K_s\, I_p (R \cdot V)^s$$



$K_s \rightarrow$ Surface specular coefficient
$I_p \rightarrow$ Point light intensity
$R \rightarrow$ Reflected direction
$V \rightarrow$ Observer direction
$s \rightarrow$ Surface shininess

Phong Model

- The illumination equation with the three illumination components is

$$I = I_{amb} + I_{diff} + I_{spec} = K_a I_a + I_p(K_d(N \cdot L) + K_s (R \cdot V)^s)$$

- If k light sources are present in the scene

$$I = I_{amb} + \sum_k (I_{diff}^k + I_{spec}^k)$$

Shading

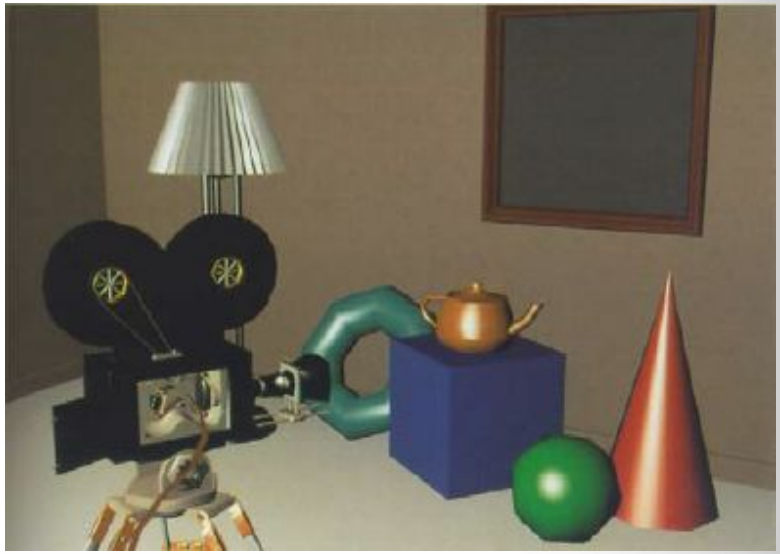
- We know how to color single points on a surface, but how do we color the whole object
- The computation of the color of an object is performed during rasterization and is called shading
- Three ways of performing shading
  - Flat shading: one lighting calculation per polygon
  - Gouraud shading: one lighting calculation per vertex
  - Phong shading: lighting calculation per pixel

Shading

- **Flat shading**
  - Color is computed once for each surface polygon
  - All pixels in a polygon have the same color
  - Fast and simple method
  - Works for objects made of flat faces
  - Disadvantage
    - Artificial changes in brightness are introduced on either side of the boundaries

Shading

- **Gouraud shading**
  - Color is computed once for each vertex of a polygon
  - The colors of the polygon's pixels are interpolated from the vertices' colors
  - The normal of a vertex is computed by interpolating the normals of all adjacent faces
  - Disadvantages
    - Gouraud shading interpolates linearly and so can make specular highlights much bigger
    - Highlights that occur in the middle of a polygon can be missed

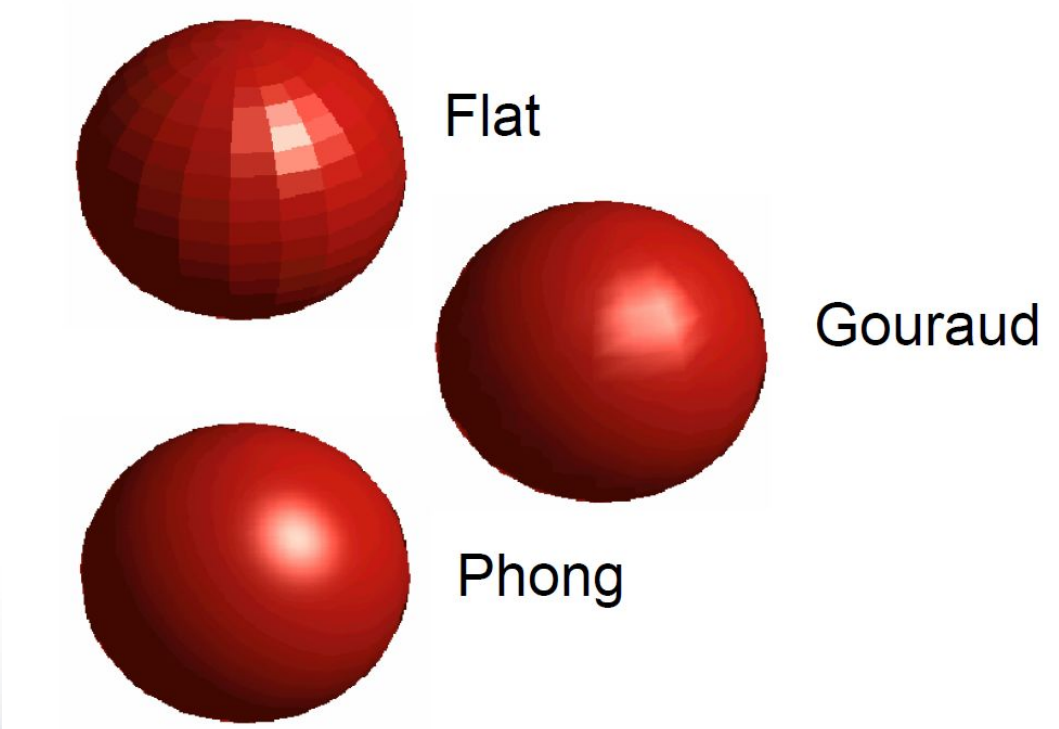


Shading

- **Phong shading**
  - A more accurate shading method
  - Color is computed at each pixel
  - Normal vectors are interpolated over the pixels of the polygon
  - Able to produce highlights that occur in the middle of the polygon

Shading

- **Examples**



Shading

- **Examples**
  - **Different surface shininess**

$s = 1$ $s = 25$ $s = 50$ $s = 100$

Shading in WebGL

- **In order to implement the shading models in WebGL we must follow different steps**
  - **Define light properties (for instance, in *drawScene()*)**

```
// Light direction (directional light example)
var lightingDirection = [0.0, 1.0, 0.5];
var normalized_L = vec3.create();
vec3.normalize(lightingDirection, normalized_L);
gl.uniform3fv(shaderProgram.L_Uniform, normalized_L);

// Light intensities
gl.uniform3f(shaderProgram.Ia_Uniform, 0.3, 0.3, 0.3);
gl.uniform3f(shaderProgram.Ip_Uniform, 0.5, 0.5, 0.5);
```

  - **Define surface properties (for instance, in *drawScene()*)**

```
// Surface properties Ka and Kd
gl.uniform3f(shaderProgram.Ka_Uniform, 1.0, 0.3, 0.0);
gl.uniform3f(shaderProgram.Kd_Uniform, 1.0, 0.3, 0.0);
```



Shading in WebGL

- **In order to implement the shading models in WebGL we must follow different steps**
  - **Normals must be transformed to eye coordinates**
    - **We cannot use the modelview matrix because translations and scaling should not be applied to the normal**
    - **We use the 3x3 top-left matrix of the modelview that codifies the rotations to be applied to the normals**

```
function sendMatricesToShader() {
    gl.uniformMatrix4fv(shaderProgram.pMatrixUniform, false, pMatrix);
    gl.uniformMatrix4fv(shaderProgram.mvMatrixUniform, false, mvMatrix);

    var normalMatrix = mat3.create();
    mat4.toInverseMat3(mvMatrix, normalMatrix);
    mat3.transpose(normalMatrix);
    gl.uniformMatrix3fv(shaderProgram.nMatrixUniform, false, normalMatrix);
}
```

Shading in WebGL

- **Shaders to implement Gouraud Shading with a directional light (surfaces without specular comp.)**

```
attribute vec3 aVertexPosition;
attribute vec3 aVertexNormal;

uniform mat4 uMVMatrix;
uniform mat4 uPMatrix;
uniform mat3 uNMatrix;

uniform vec3 L;
uniform vec3 Ia;
uniform vec3 Ip;

uniform vec3 Ka;
uniform vec3 Kd;

varying vec3 shadedColor;

void main(void) {
    gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);

    vec3 N = uNMatrix * normalize(aVertexNormal);
    float dotNL = max(dot(N, L), 0.0);
    shadedColor = Ka * Ia + Kd * Ip * dotNL;
}
```

Vertex shader

```
precision mediump float;

varying vec3 shadedColor;

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

Fragment shader

Shading in WebGL

- **Shaders to implement Phong Shading (there is a specular component and a point light source)**

```
attribute vec3 aVertexPosition;
attribute vec3 aVertexNormal;

uniform mat4 uMVMatrix;
uniform mat4 uPMatrix;
uniform mat3 uNMatrix;

uniform vec3 uLightSourcePosition;

varying vec3 vertNormalEye;
varying vec4 vertPositionEye;
varying vec4 lightSourcePositionEye;

void main(void) {
    vertPositionEye = uMVMatrix * vec4(aVertexPosition, 1.0);
    gl_Position = uPMatrix * vertPositionEye;
    vertNormalEye = uNMatrix * normalize(aVertexNormal);
    lightSourcePositionEye = uMVMatrix * vec4(uLightSourcePosition, 1.0);
}
```

Vertex shader

Shading in WebGL

- **Shaders to implement Phong Shading (there is a specular component and a point light source)**

```
precision mediump float;

uniform vec3 Ia, Is, Id;
uniform vec3 Ka, Ks, Kd;
uniform float s;

varying vec3 vertNormalEye;
varying vec4 vertPositionEye;
varying vec4 lightSourcePositionEye;

void main(void) {
    vec3 L = normalize(lightSourcePositionEye.xyz - vertPositionEye.xyz);
    vec3 N = normalize(vertNormalEye);
    float dotNL = max(dot(N, L), 0.0);

    vec3 V = normalize(-vertPositionEye.xyz);
    vec3 R = reflect(-L, N);
    float dotRVs = pow(max(dot(R, V), 0.0), s);

    vec3 shadedColor = Ka * Ia + Kd * Id * dotNL + Ks * Is * dotRVs;
    gl_FragColor = vec4(shadedColor, 1.0);
}
```

Fragment shader

Shading in WebGL

- **Shaders to implement Phong Shading (there is a specular component and a point light source)**
  - **Changes in the code must be done in order to define and send to the shaders**
    - **Light source position**
    - **Specular intensity of the light**
    - **Specular coefficient and shininess of the material**
  - **Note that different intensities and surface coefficients (colors in both cases) can be used**
    - **We can have colored lights**
    - **Materials with different colors when reflecting light**

Questions?

www.citm.upc.edu