universidade federal de pelotas
CDTec
centro de desenvolvimento tecnológico

computação
Universidade Federal de Pelotas

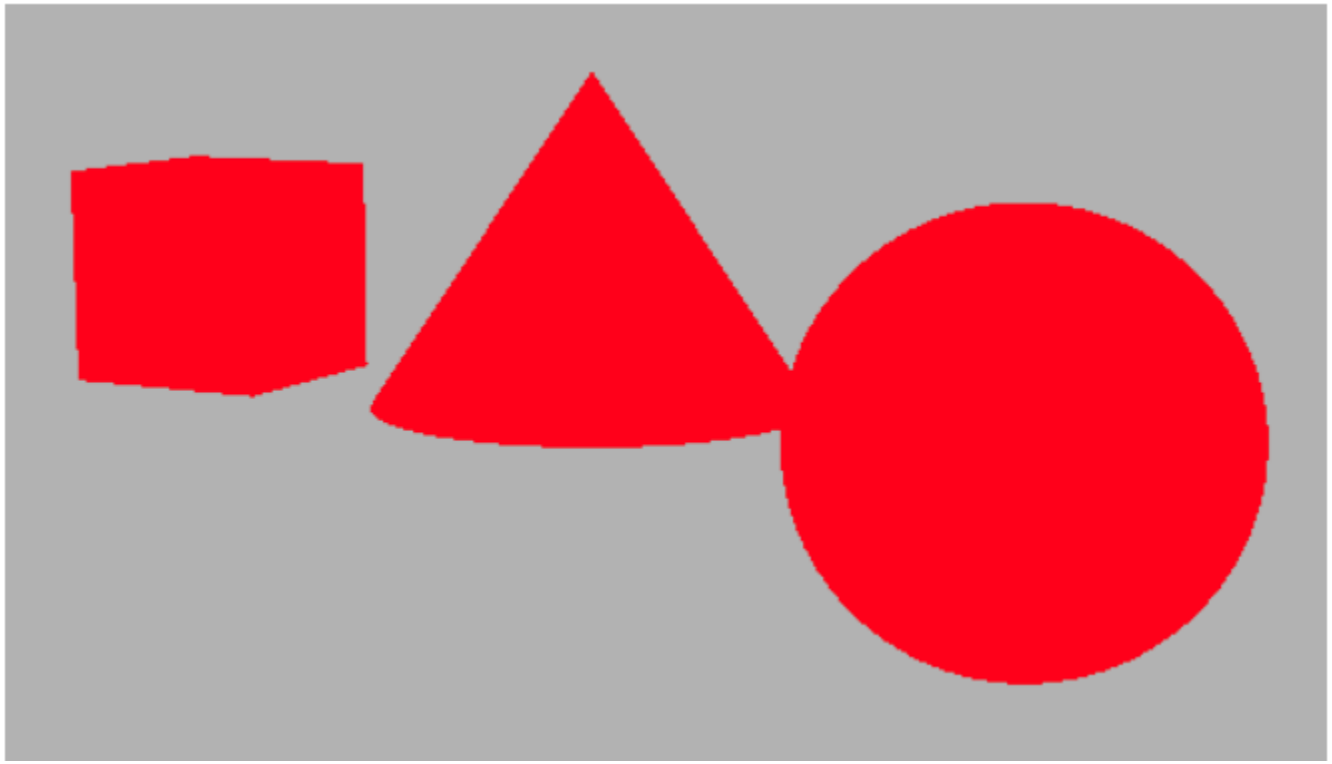
Computer Graphics
Today: Illumination

Prof. PhD Rafael P. Torchelsen
rafael.torchelsen@inf.ufpel.edu.br

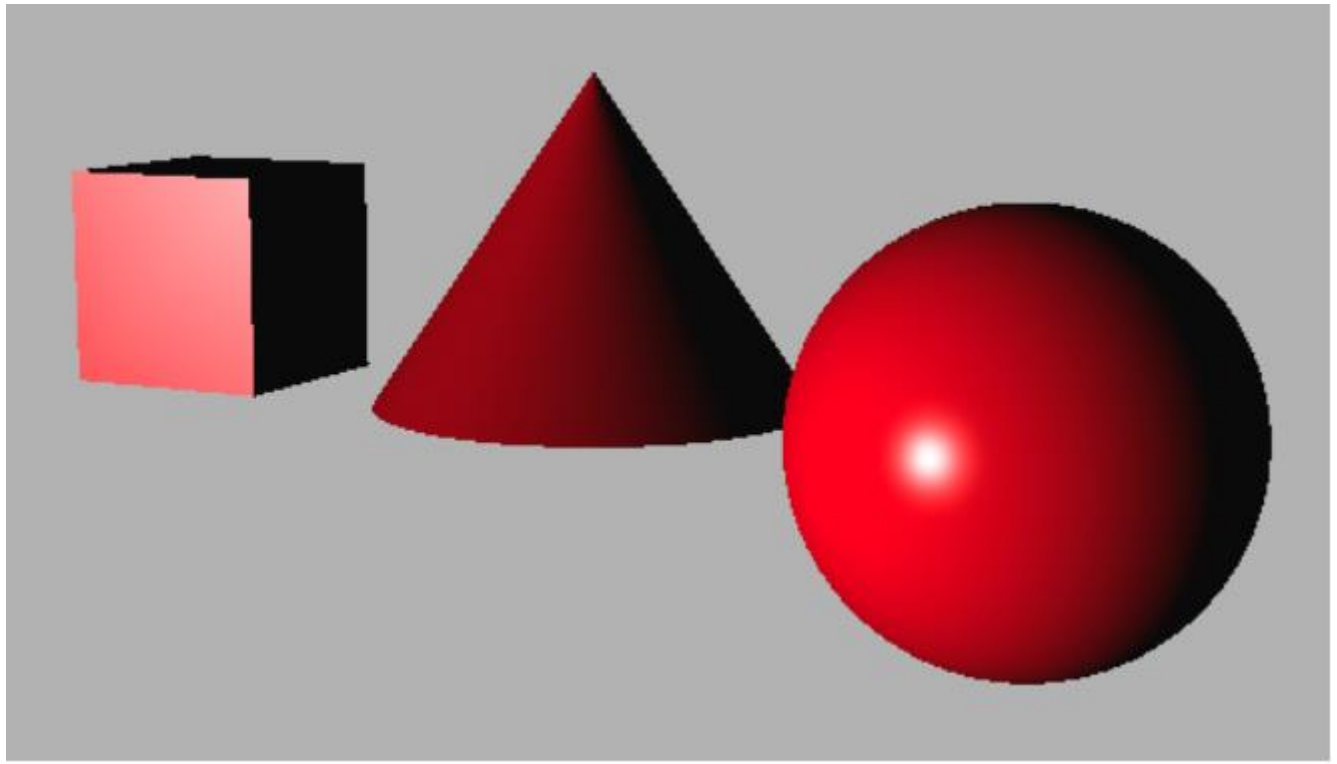
No light

No light + Shape colors

computação
Universidade Federal de Pelotas



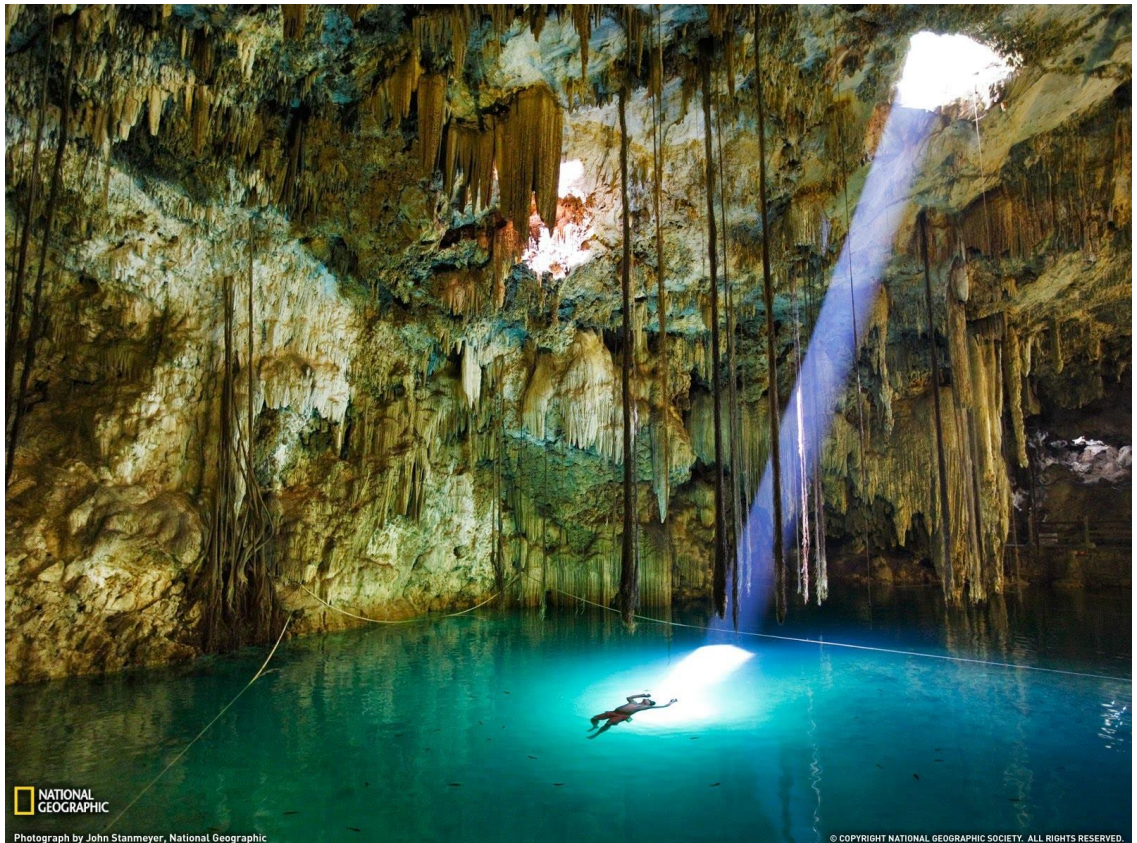
Light



What is light?

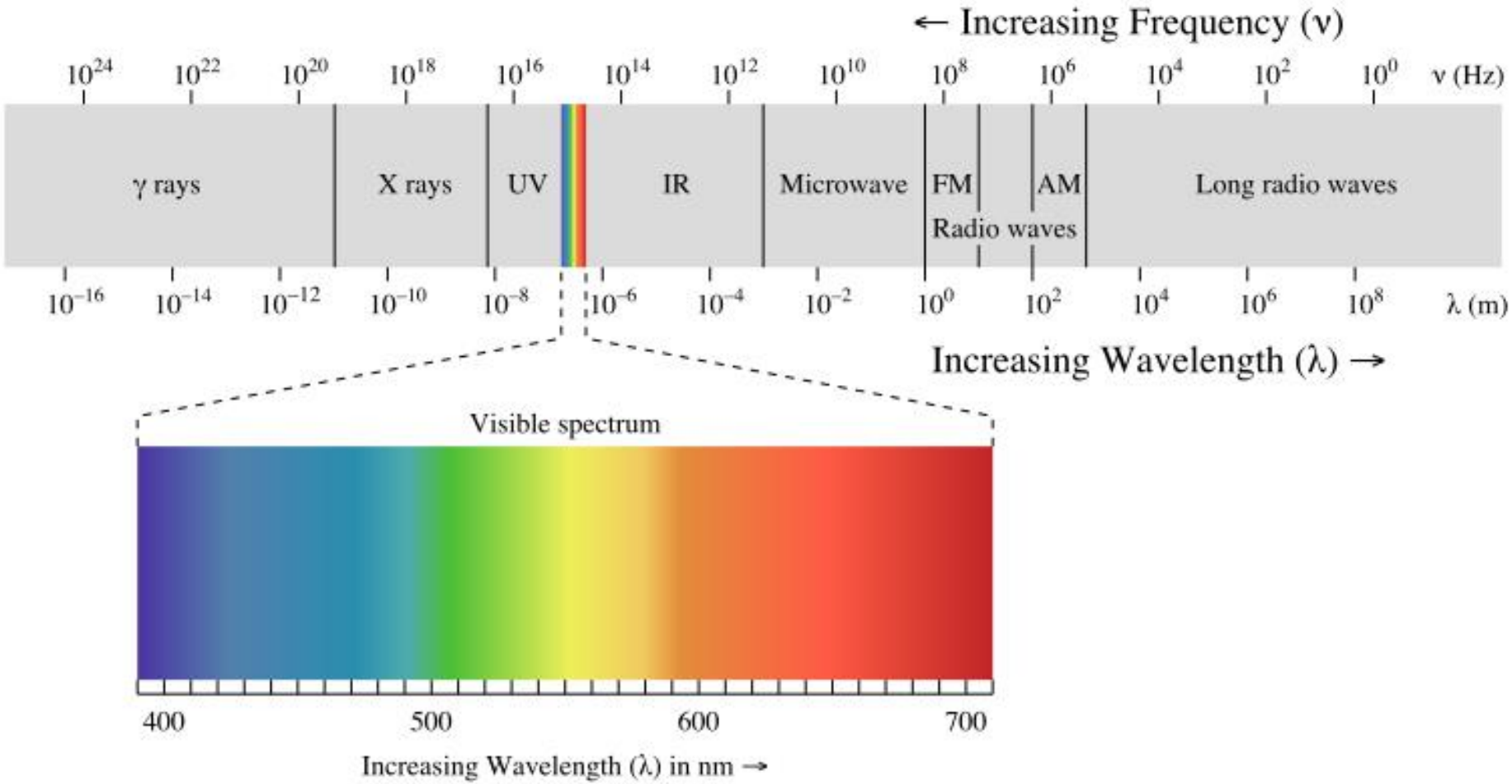
computação
Universidade Federal de Pelotas

Visible light (commonly referred to **simply as light**) is electromagnetic radiation that is **visible to the human eye**, and is responsible for the **sense of sight**



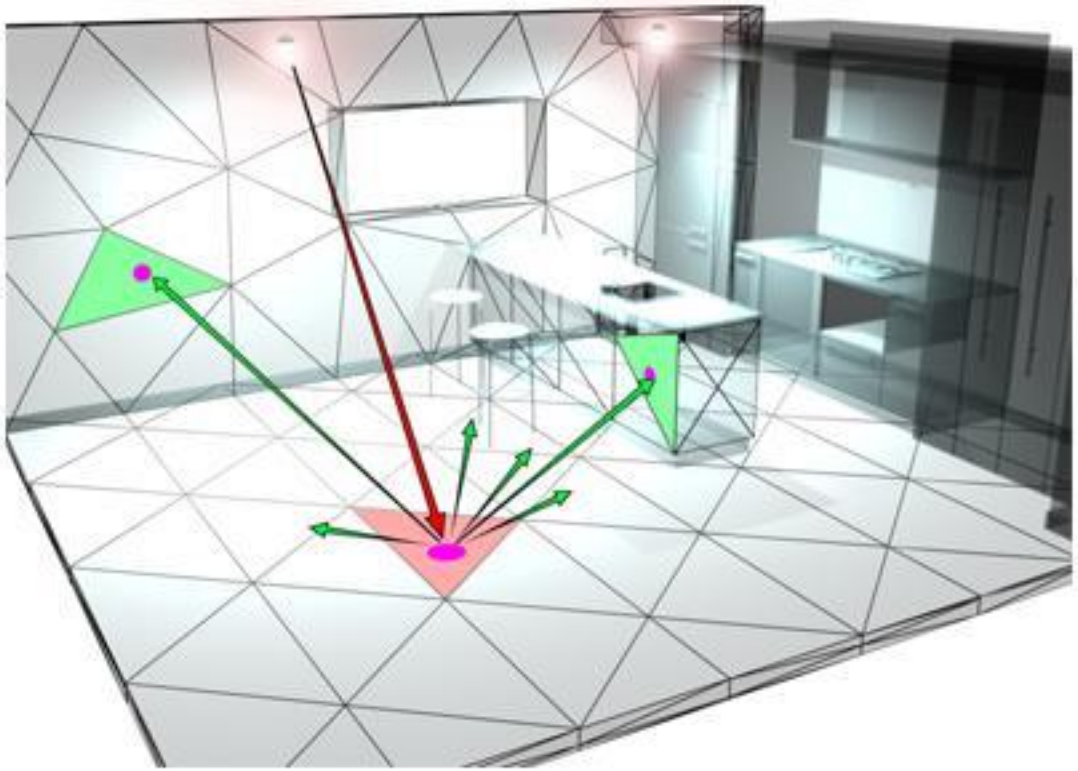
What is light?

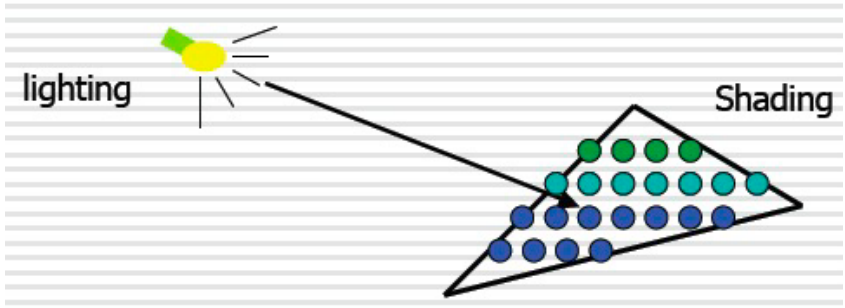
computação
Universidade Federal de Pelotas



Simulation

computação
Universidade Federal de Pelotas





Illumination models

computação
Universidade Federal de Pelotas

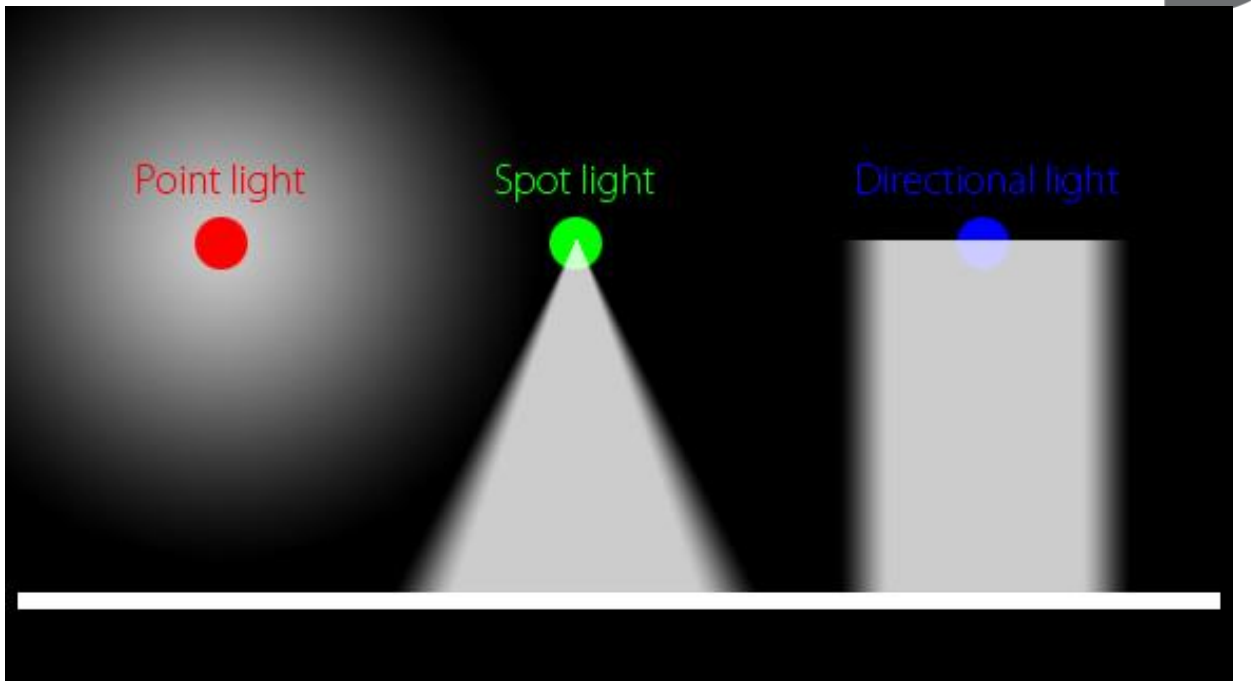
- An “illumination model” describes **inputs**, assumptions, and **outputs** used to calculate illumination (color / brightness) of surface elements
  - Usually includes:
    - **Light attributes** (intensity, color, position, direction, shape)
    - **Object surface** properties (color, reflectivity, transparency, etc.)
    - **Interaction** between **lights** and **objects**

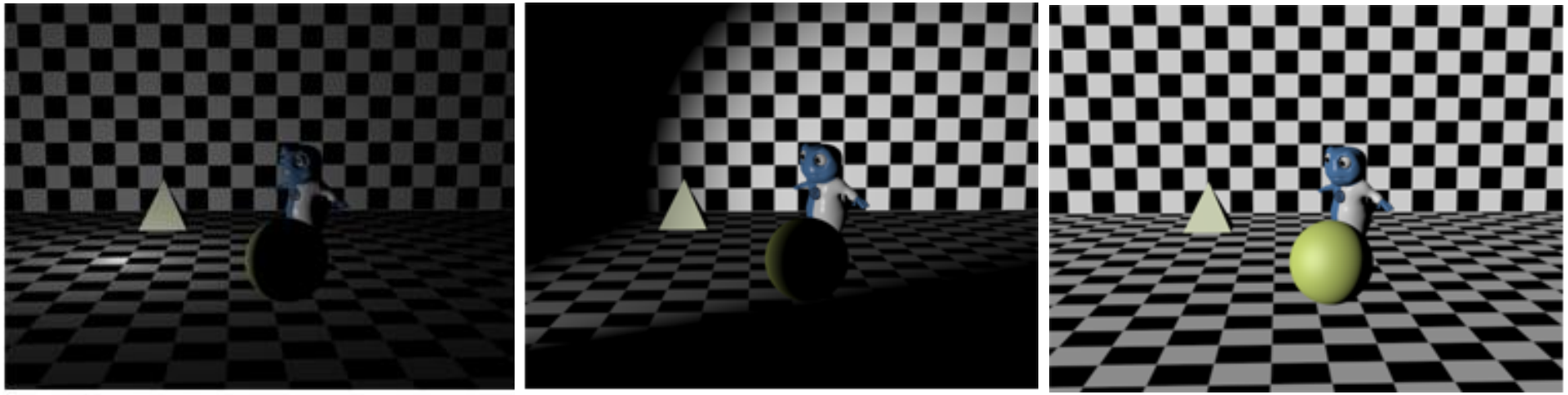
Illumination models

computação
Universidade Federal de Pelotas

- **Physically-based** Illumination Models
  - Some models of illumination are based on **real physics**
  - Require accurate input data and make few assumptions
    - **Rarely** have **enough information** to specify all inputs
  - Takes a **long time** to **compute**
- **Non-Physically-Based** Illumination Models
  - Just need a model that **looks good enough** for a specific application
  - Emphasis on **speed** and efficiency (use of resources like memory, CPU, etc.)

Illumination models



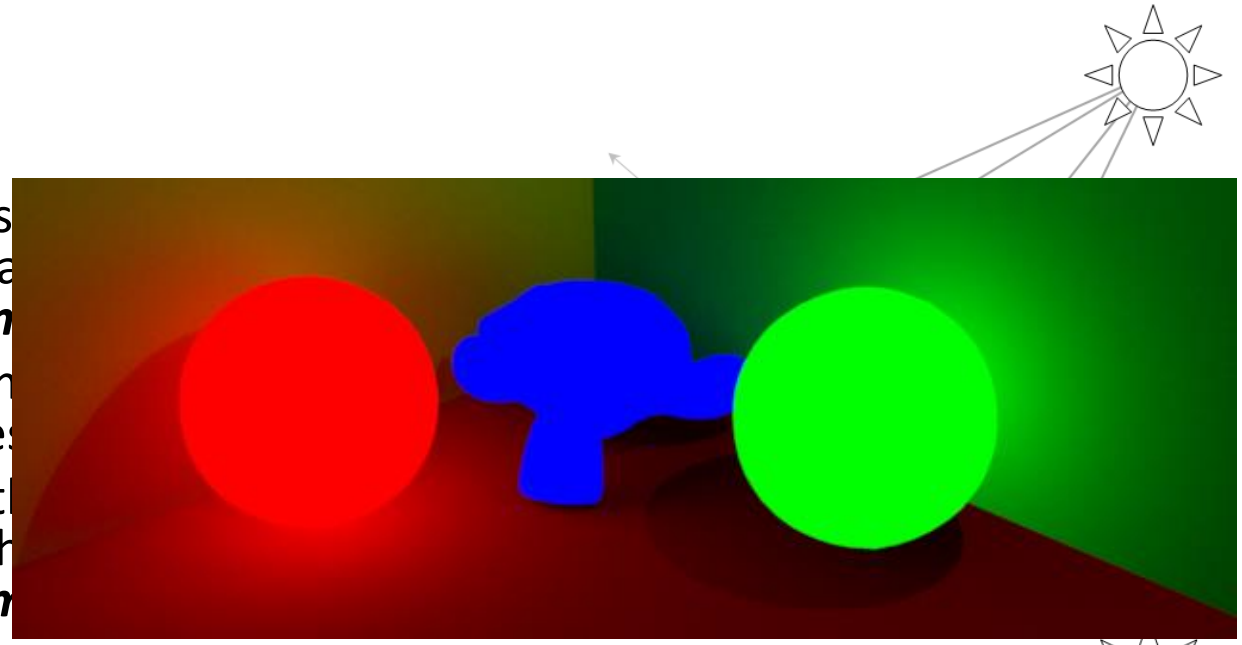


Illumination models

computação
Universidade Federal de Pelotas

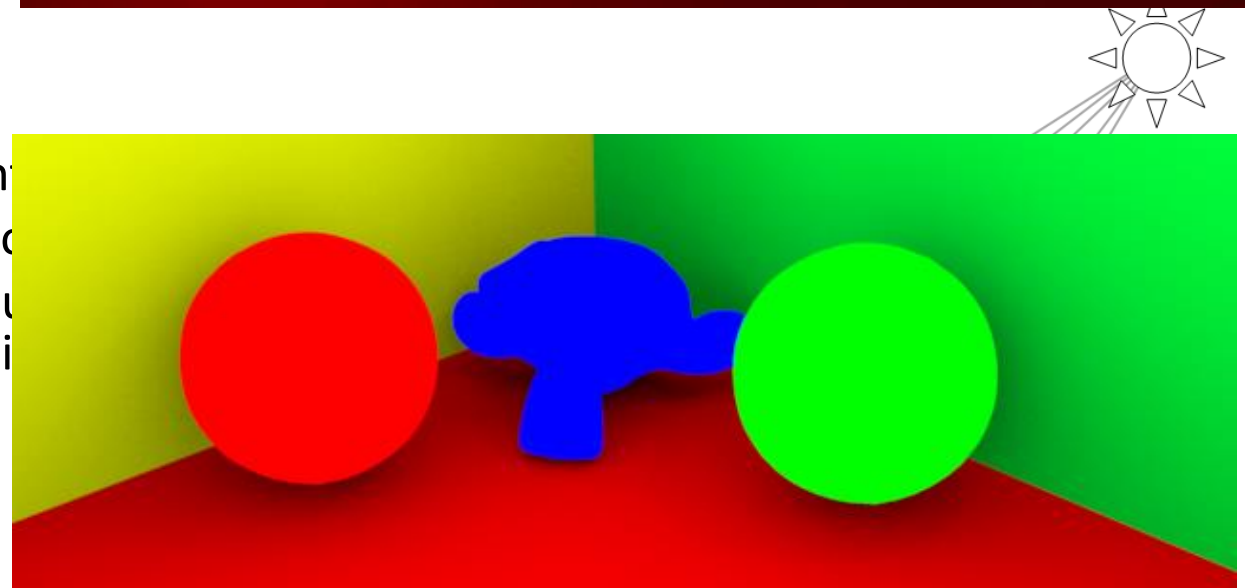
- **Global Illumination**
  - Most light striking a s
    comes directly from a
    source (***direct illumin***
  - However, objects in t
    receive light from ligh
    objects (***indirect illum***

Advanced topic



- **Local Illumination**
  - Takes only direct ligh
  - Is an approximation o

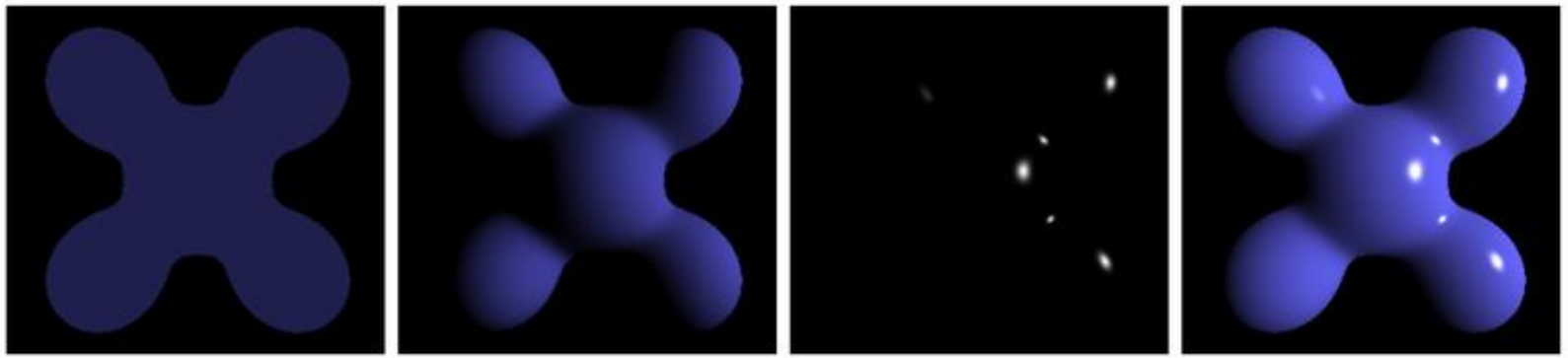
Today



Local Illumination (Phong)

computação
Universidade Federal de Pelotas

- Reduce the light model to 3 simple components:
  - **Ambient**
  - **Diffuse**
  - **Specular**
- Illumination (color intensity) of a point is equal to:
  - **Intensity** = ambient + diffuse + specular
- **Materials reflect** each component **differently**
  - Specified as material reflection coefficients:

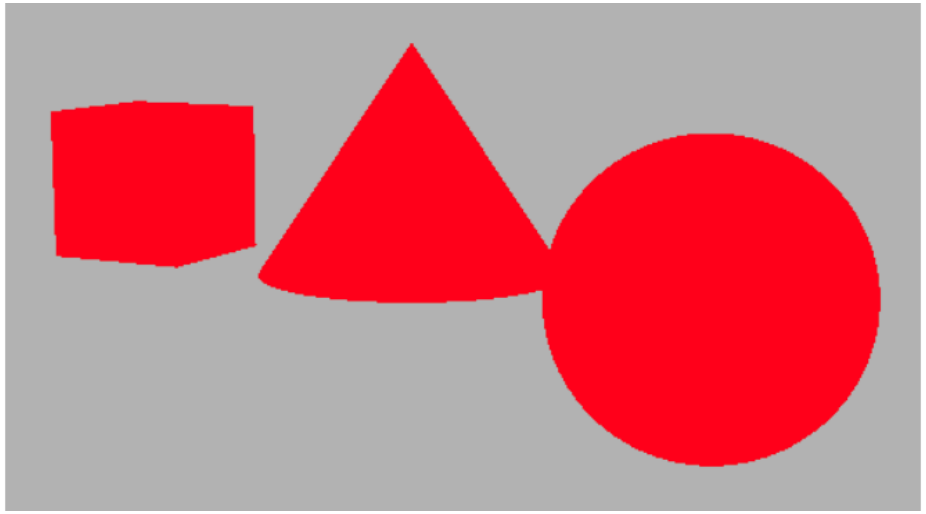


Ambient Light Contribution

computação
Universidade Federal de Pelotas

- Ambient light = background light
  - It is used to **simulate** indirect lighting

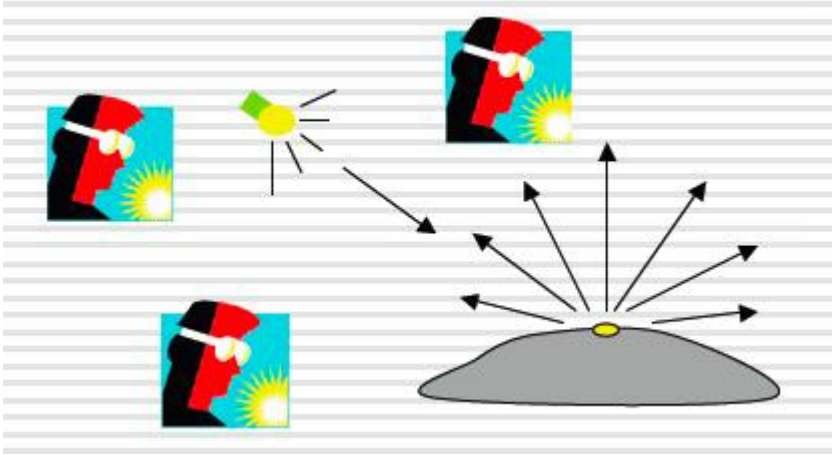
- Ambient component
  - **Independent** of
    - Object's position
    - Viewer's position
    - Light source's position
  - **Dependent** of
    - A constant factor (in each of the R, G, B channels)



Diffuse Light Contribution

computação
Universidade Federal de Pelotas

- **Diffuse light** = illumination that a surface receives from a light source that **reflects equally in all directions**

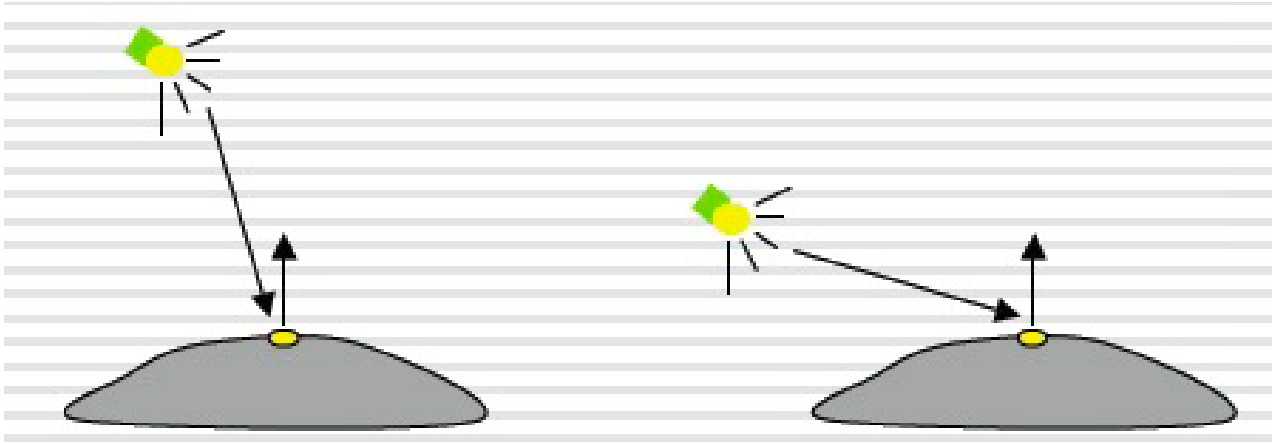


  - **Independent** of:
    - Viewer's position
  - **Dependent** of:
    - Light's position
    - Surface property (normal, reflectance property)

Diffuse Light Calculation

computação
Universidade Federal de Pelotas

- Need to know **how much light** a point on the object receives **from** the light **source**
- Solution based on **Lambert's Law**



Point receives more light

Point receives less light

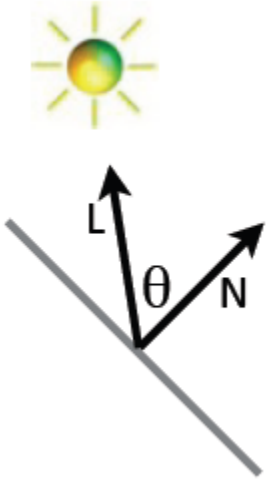
Diffuse Light Calculation

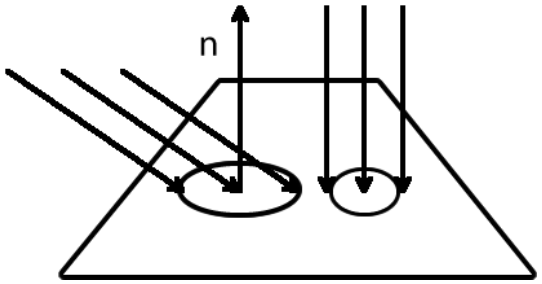
computação
Universidade Federal de Pelotas

- **Lambert's Law**: The radiant energy D that a small surface patch (or a point) receives from a light source is:
  - Diffuse = $\boldsymbol{K_d\, I\, cos\,(\theta)}$
    - K_d = diffuse reflection constant
    - I = light intensity
    - θ = angle between the light vector and the surface normal

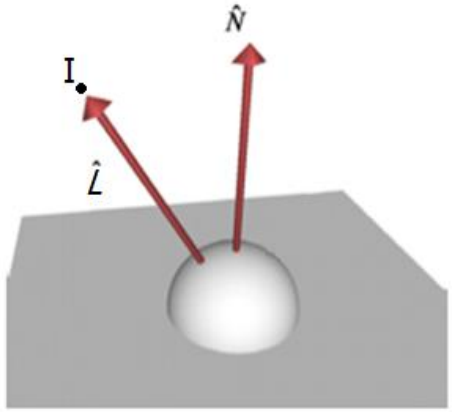
Material

- How do we compute cos (θ)?
  - $N \cdot L$ (dot product between N and L)





As the angle between the light and the normal increases, the light's energy spreads across a larger area



The hemisphere represents equal magnitude of the reflected intensity for any outgoing vector.

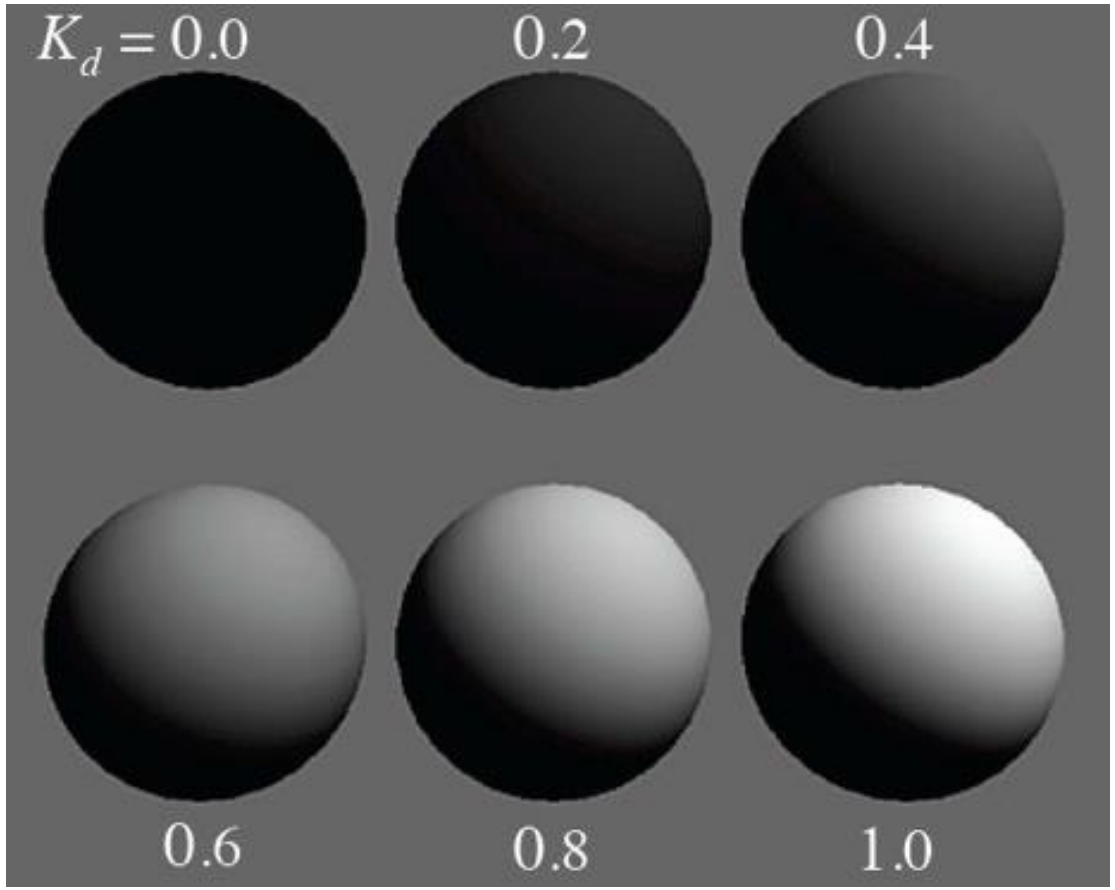
Diffuse Light Examples

computação
Universidade Federal de Pelotas

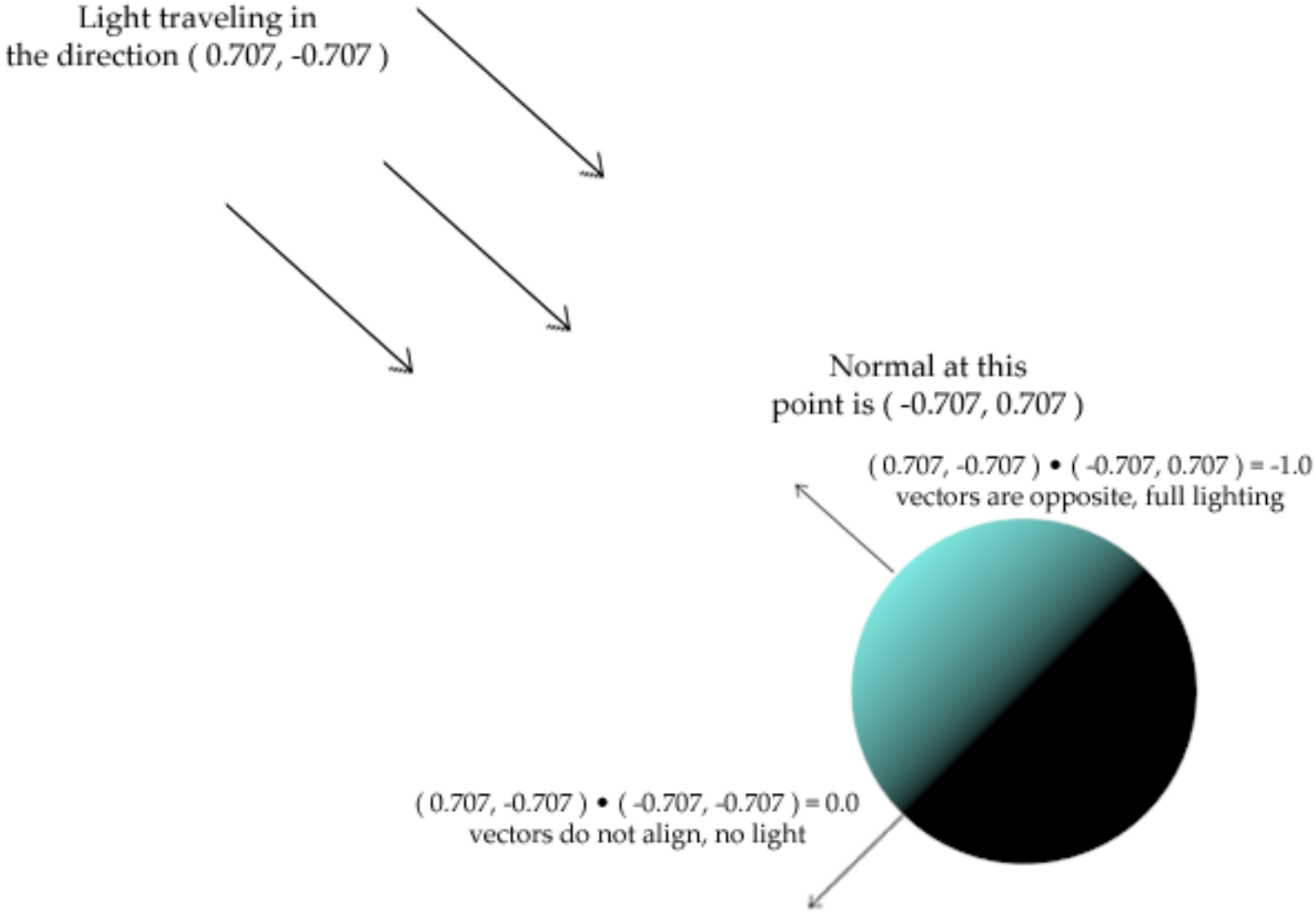
Directional

Diffuse = $K_d\ I \cos(\theta)$

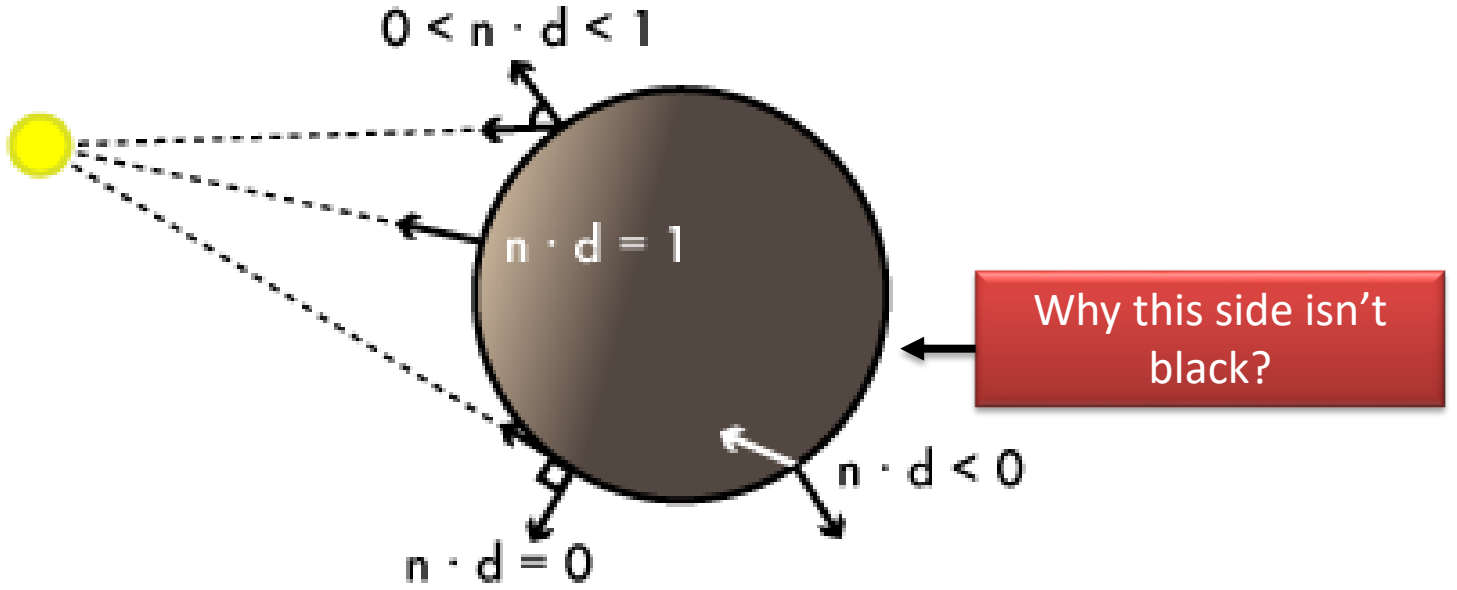
- For I = 1.0



Diffuse Light Examples



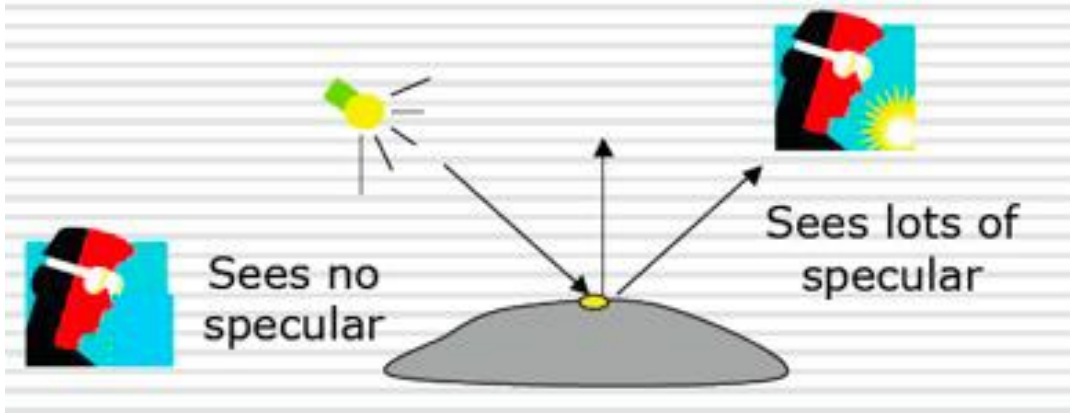
Diffuse + Ambient

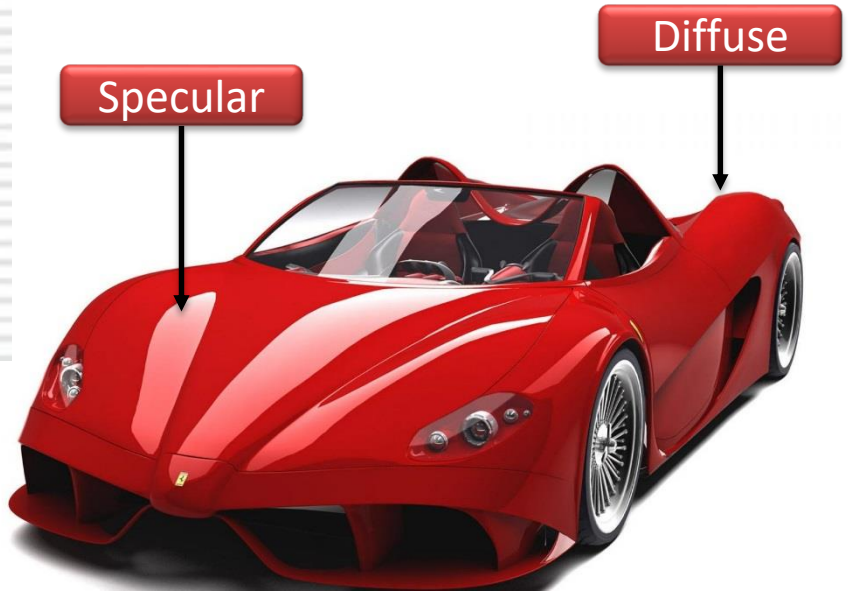


diffuse factor = (normal · light direction) * light's diffuse color
ambient and diffuse factor = diffuse factor + light's ambient color
shaded color = surface's diffuse color * ambient and diffuse factor

Specular Light Contribution

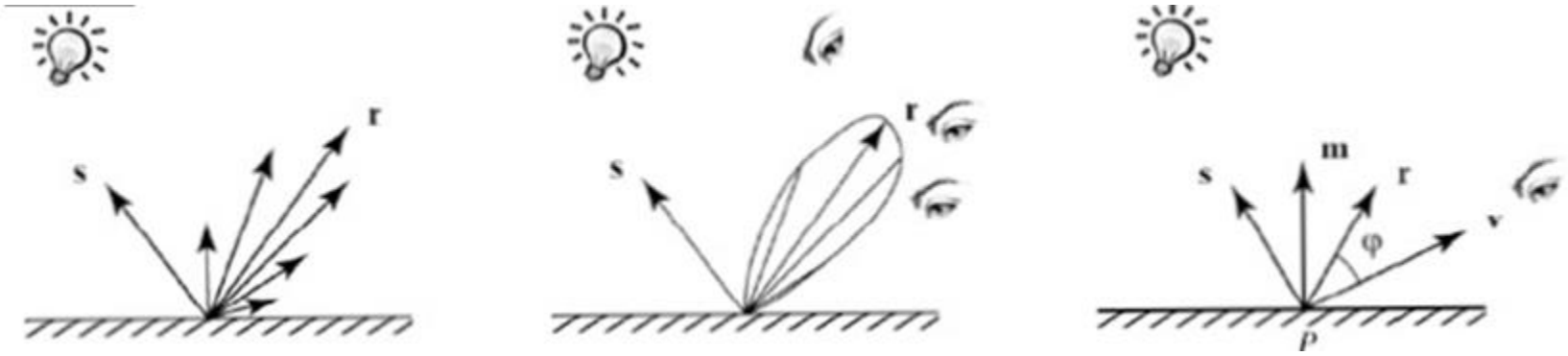
- Specular light = light reflection from **shiny surfaces**
- Color **depends on material** and how it scatters light
  - Shiny surfaces (metal, mirror, etc.) reflect more light
- Specular light **depends** on both **light** source position and **view** position





Specular Light Calculation

computação
Universidade Federal de Pelotas

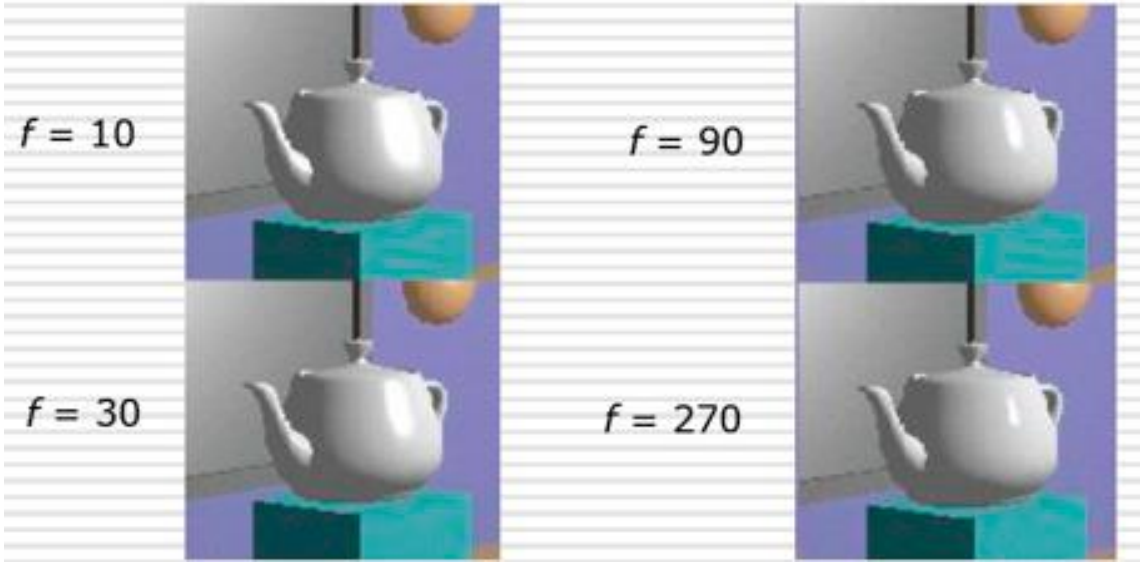


- ϕ is the **angle** between the view **direction** and the **reflective** vector
- When ϕ is small, the viewer sees more specular light

Specular Light Calculation

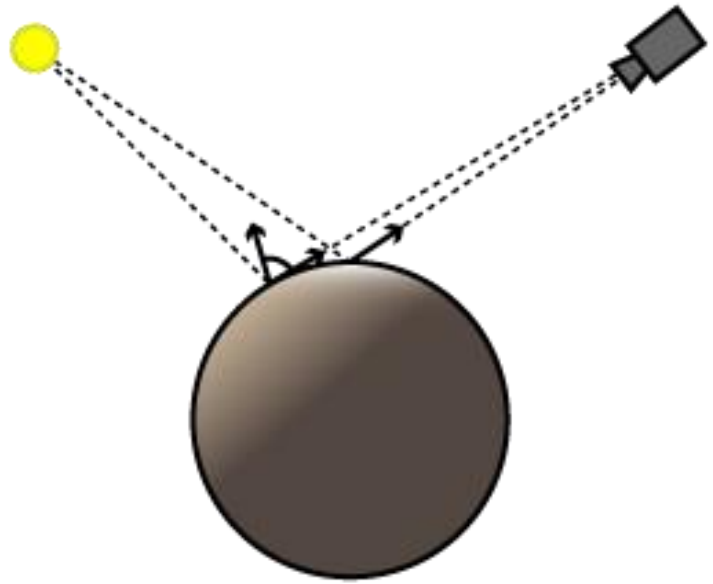
computação
Universidade Federal de Pelotas

- The Phong lighting model (not the Phong shading model)
  - Specular = $K_s \; I \; \cos^f(\phi)$
    - K_s = specular reflection constant
    - I = light intensity
    - ϕ = angle between reflective ray and view vector
    - f = surface property for specular highlight



Specular Light Calculation

computação
Universidade Federal de Pelotas



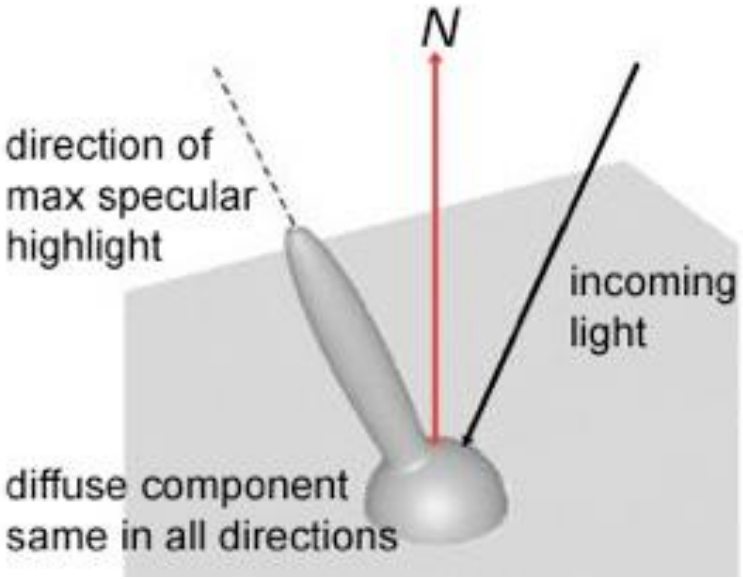
specular factor = (reflected light direction · viewer direction)$^{\text{shininess}}$ * light's specular color

shaded color = specular factor * surface's specular color + diffuse shaded color

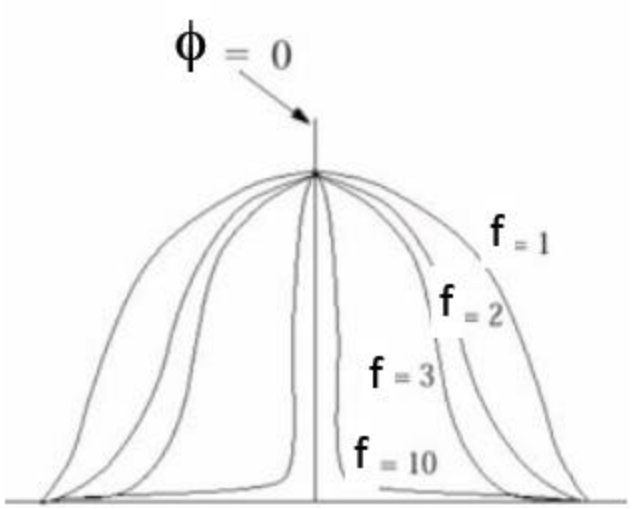
Specular Light Calculation

computação
Universidade Federal de Pelotas

Specular = $K_s \ I \ \cos^f(\phi)$



Diffuse hemisphere with specular Gaussian surface

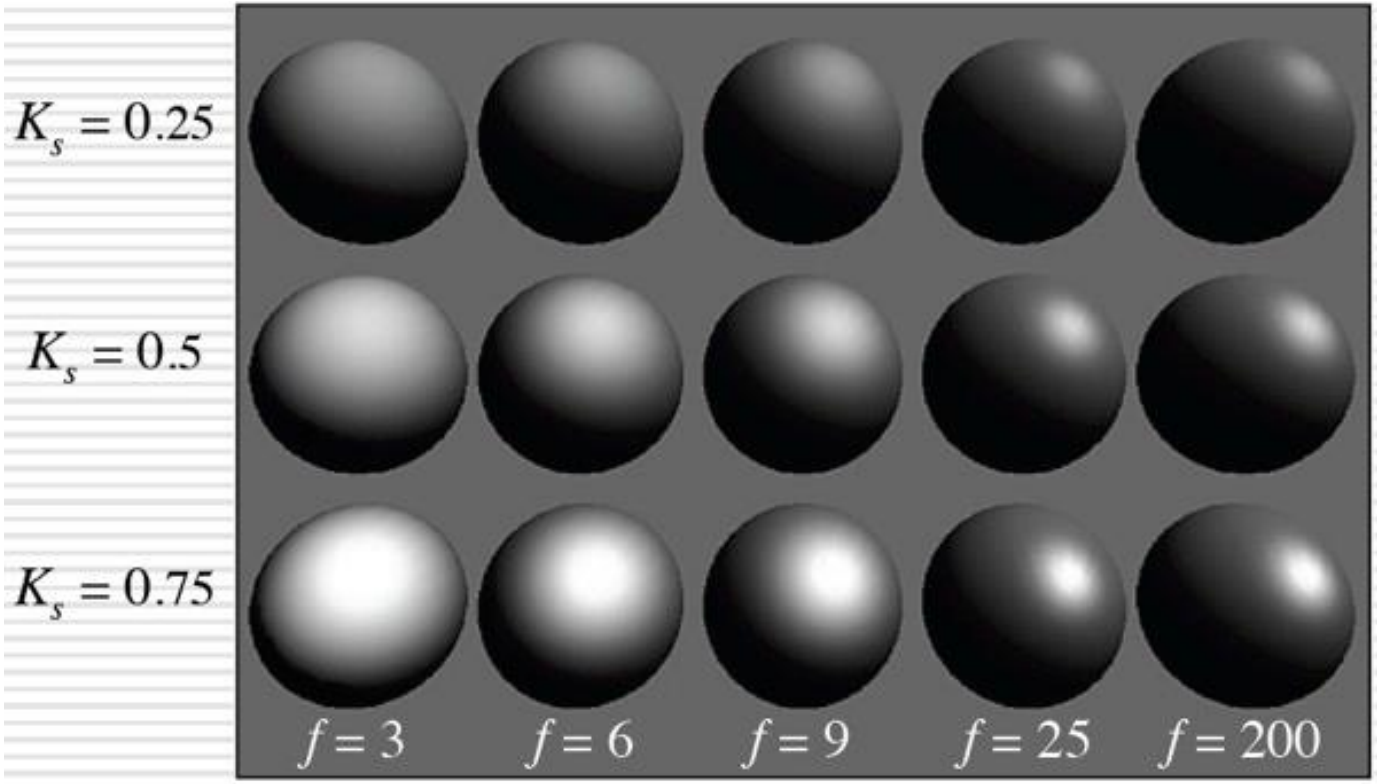


Shape of the Gaussian

Specular Light Examples

Specular = $K_s \ I \ \cos^f(\phi)$

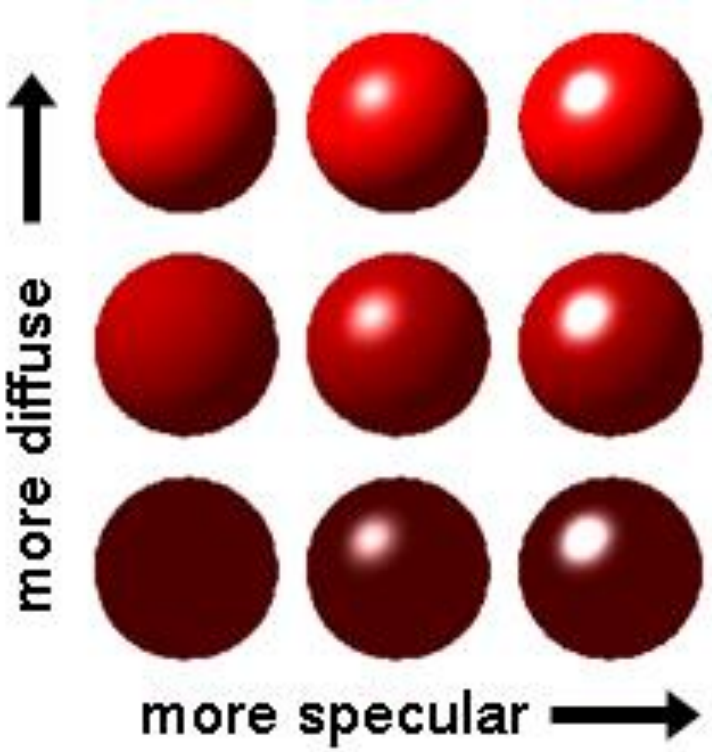
- For I = 1.0



Variation in Diffuse and Specular

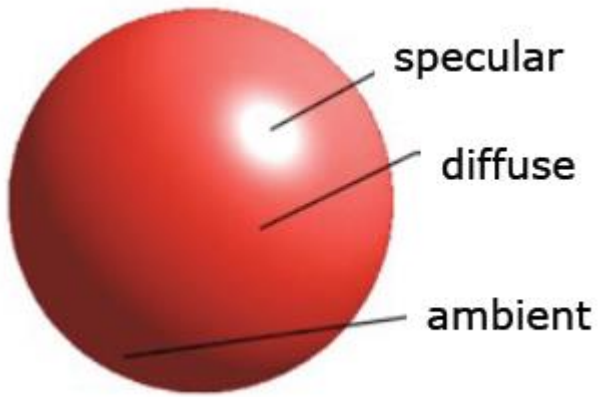
computação
Universidade Federal de Pelotas

- Putting Diffuse and Specular together in our local illumination model



Lighting Equation

computação
Universidade Federal de Pelotas



- Direct Illumination = ambient + diffuse + specular
- color = $K_a\ I + K_d\ I\ (N \cdot L) + K_s\ I\ (R\ \cdot V)^f$

- If there are m lights, we sum over each light
  - color = $K_a\ I + \sum_m (K_d\ I_m\ (N \cdot L_m) + K_s\ I_m\ (R_m\ \cdot V)^f)$
  - color = $K_a\ I + \sum_m\ I_m (K_d (N \cdot L_m) + K_s\ (R_m\ \cdot V)^f)$

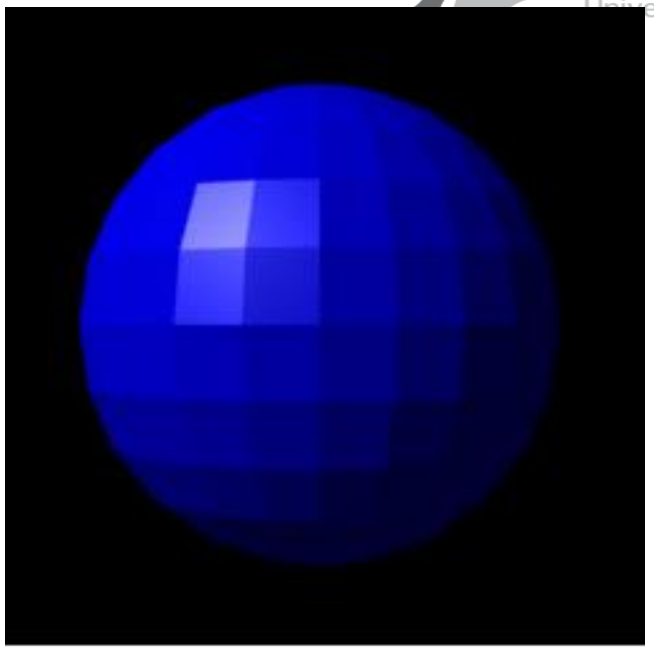
Adding in Object Color

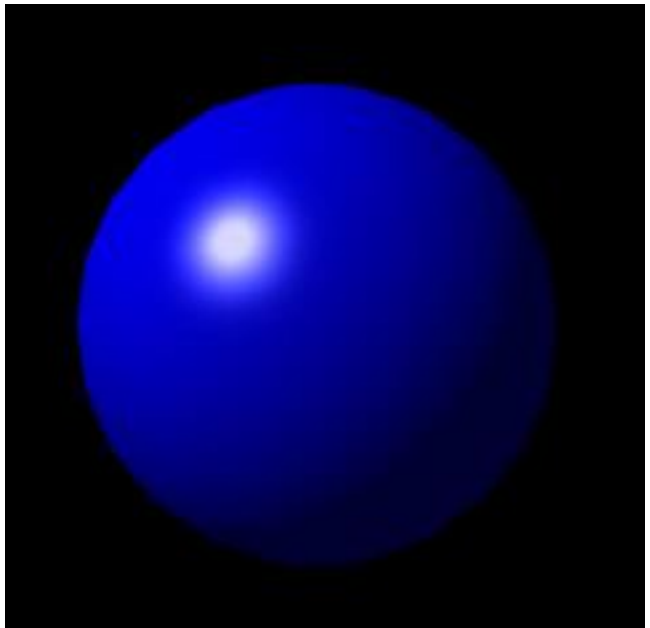
computação
Universidade Federal de Pelotas

- Since the color we perceive is based on the color of the **light** and the color of the **object**, we need to model both terms:
  - color = $K_a\ I\ O_a + K_d\ I\ O_d(N \cdot L) + K_s\ I\ O_s\ (R\ \cdot V)^f$
- For multiple lights:
  - color = $K_a\ I\ O_a + \sum_m I_m\ (K_d\ O_d(N \cdot L) + K_s\ \ O_s\ (R\ \cdot V)^f)$
- Adding in light attenuation (how quickly light falls off as distance increases):
  - color = $K_a\ I\ O_a + \sum_m f_{att_m}\ I_m\ (K_d\ O_d(N \cdot L) + K_s\ \ O_s\ (R\ \cdot V)^f)$

Shading vs. Lighting

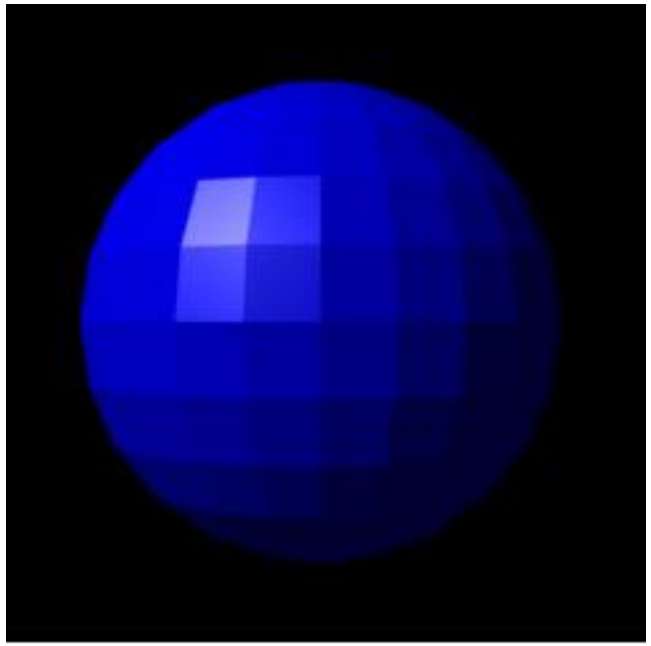
- **How is Shading different from Lighting?**
- Lighting evaluates the **lighting equation** at each point on the surface of an object
- Shading is kind of a "**hack**" that computes only lighting equations on the vertices, and interpolates the pixels in between





Shading Models – Flat/Constant

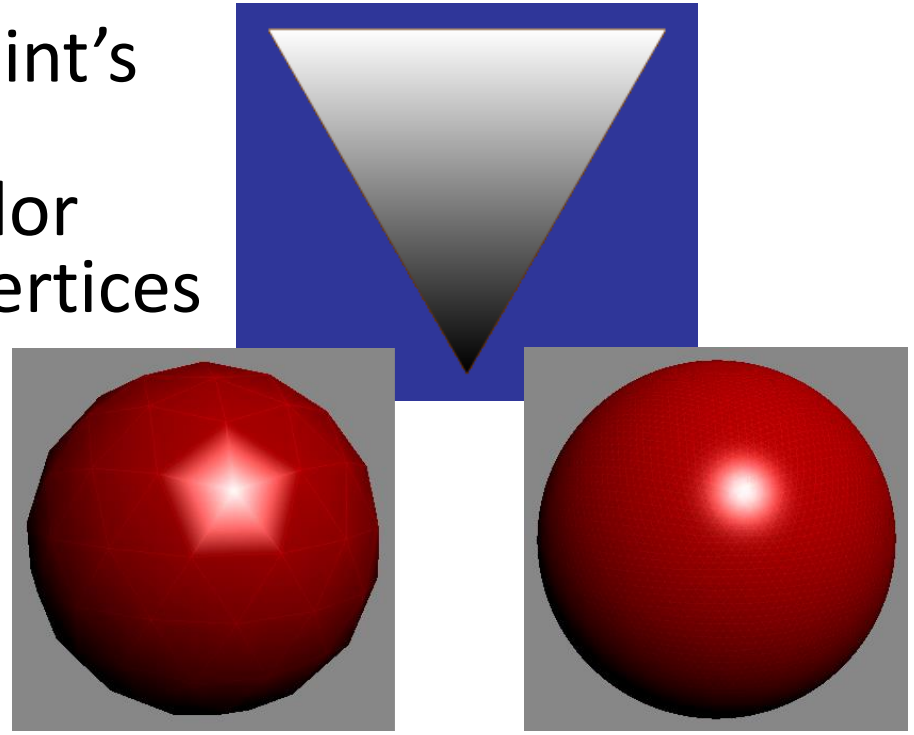
- We define a normal at each polygon
  - not at each vertex
- **Lighting:** Evaluate the lighting equation at the center of each polygon using the associated normal
- **Shading:** Each sample point on the polygon is given the interpolated lighting value at the vertices (i.e. a total hack)



Shading Models – Gouraud

computação
Universidade Federal de Pelotas

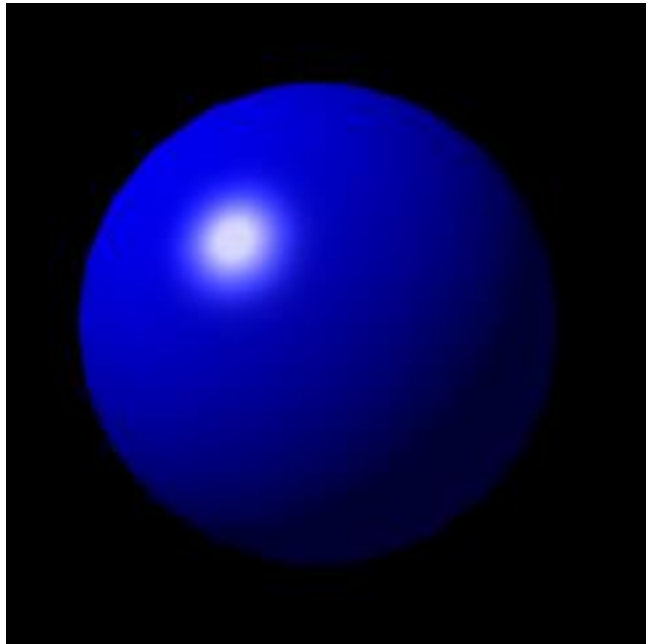
- We define a normal vector at each vertex
- **Lighting**: Evaluate the lighting equation at each vertex using the associated normal vector
- **Shading**: Each sample point's color on the polygon is interpolated from the color values at the polygon's vertices which were found in the lighting step



Shading Models – **Phong**

computação
Universidade Federal de Pelotas

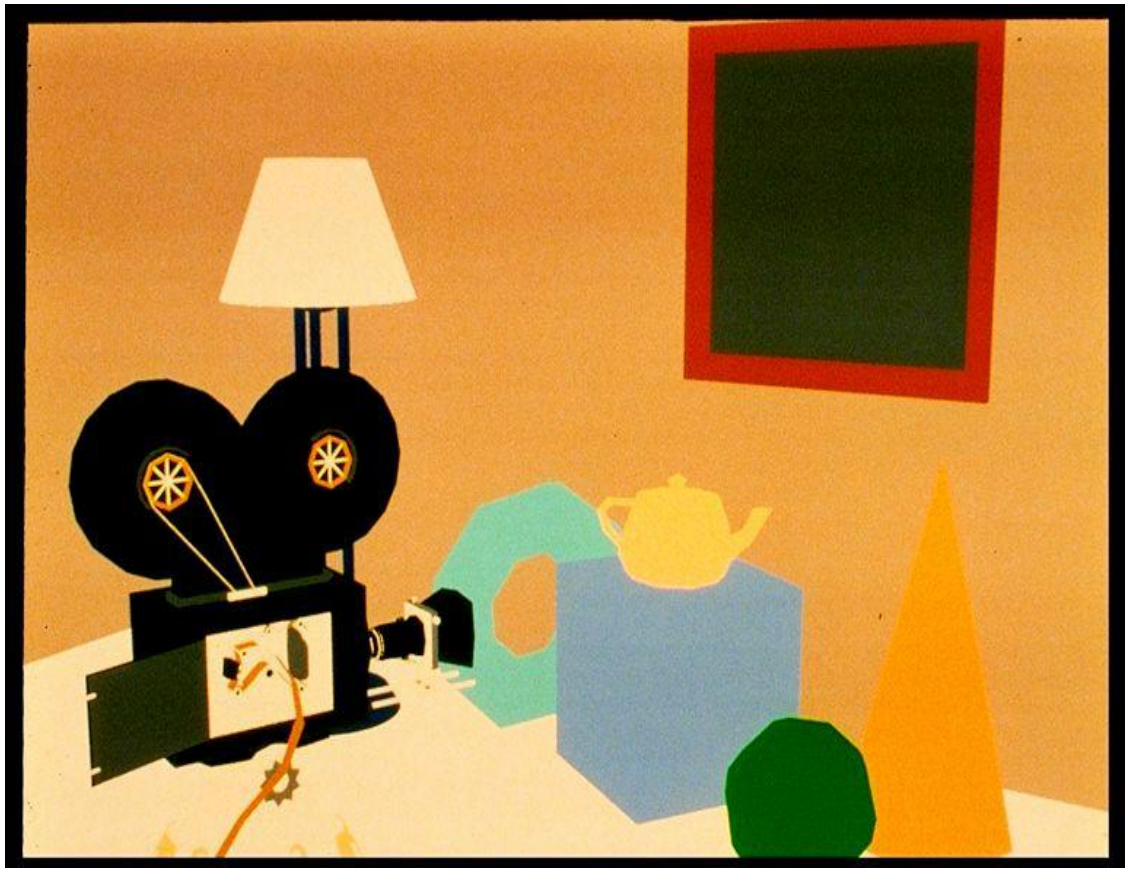
- Each vertex has an associated normal vector
- **Lighting:** Evaluate the lighting equation at each vertex using the associated normal vector
- **Shading:** For every sample point on the polygon we **interpolate the normals** at vertices of the polygon and compute the color using the lighting equation with the interpolated normal at each interior pixel



Shading Models Compared

computação
Universidade Federal de Pelotas

Constant shading: no interpolation, pick a single representative intensity and propagate it over entire object. Loses almost all depth cues.

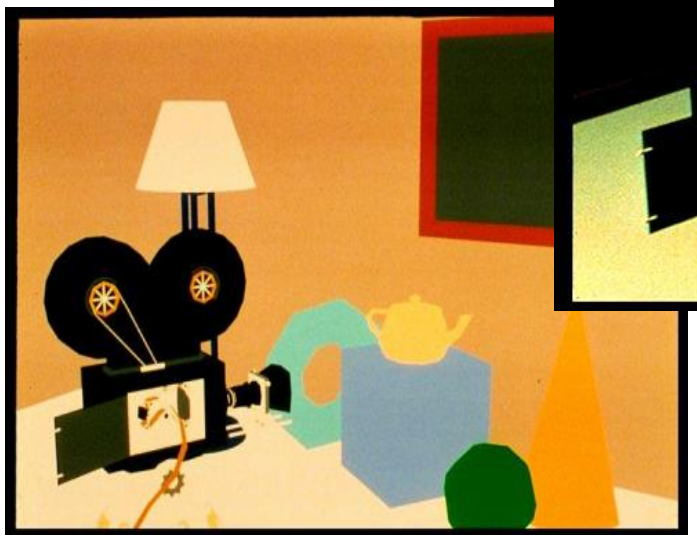


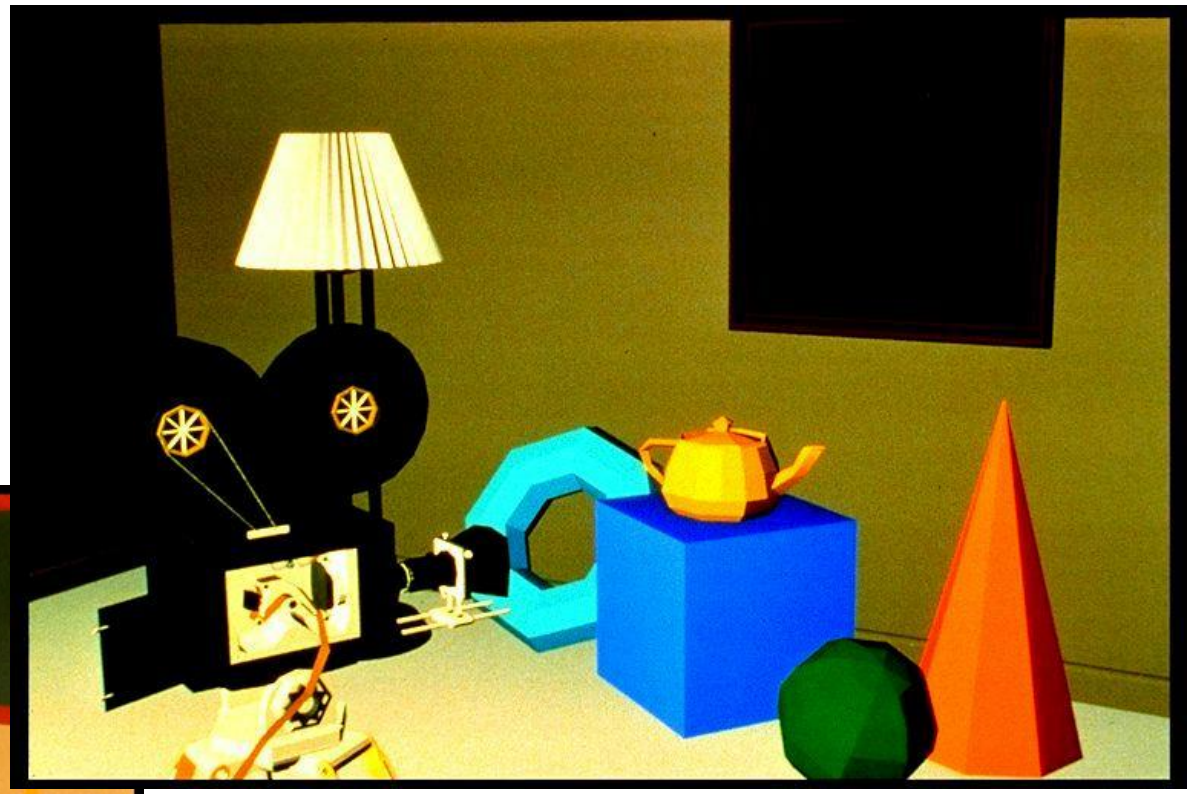
Pixar “Shutterbug” images from:
www.siggraph.org/education/materials/HyperGraph/scanline/shade_models/shading.htm

Shading Models Compared

computação
Universidade Federal de Pelotas

Flat or Faceted Shading: constant intensity over each face



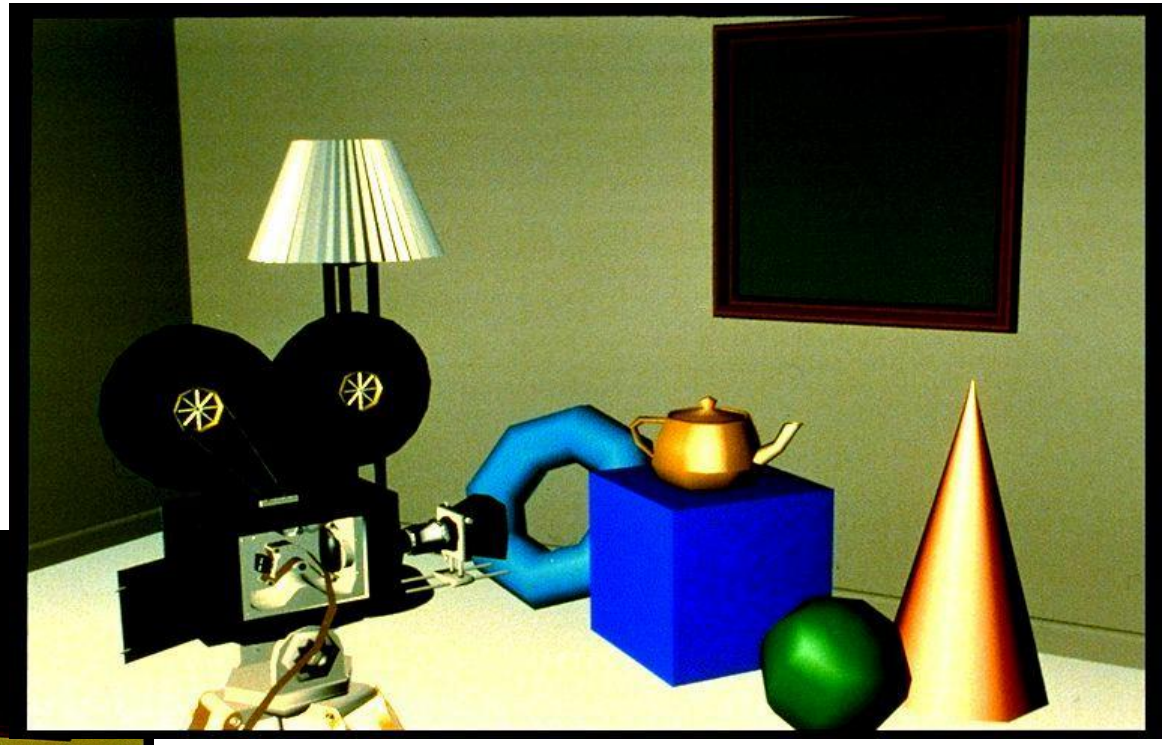


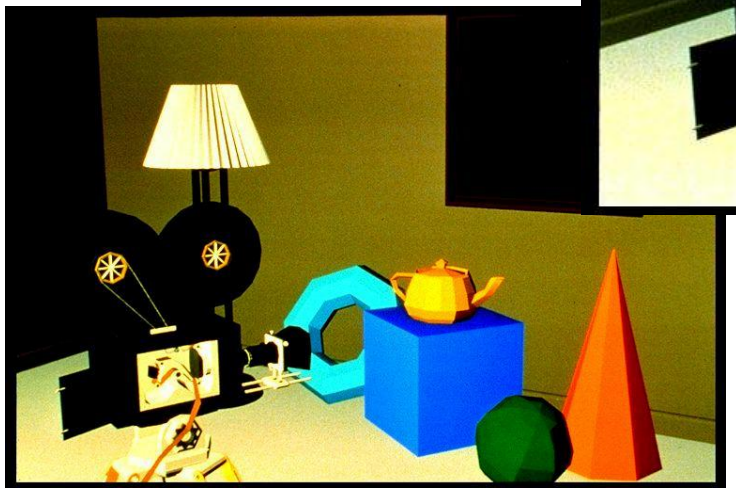
Constant Shading

Shading Models Compared

computação
Universidade Federal de Pelotas

Gouraud Shading:
Linear Interpolation of intensity across triangles to eliminate edge discontinuity





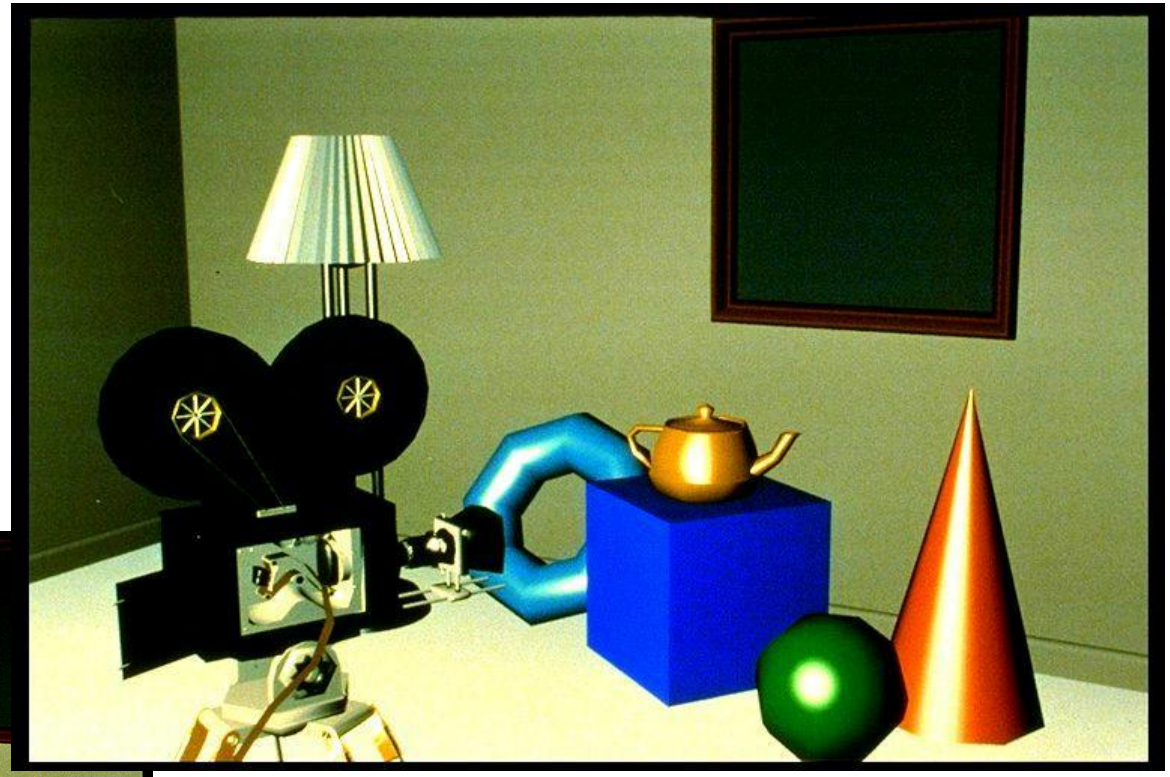
Flat Shading

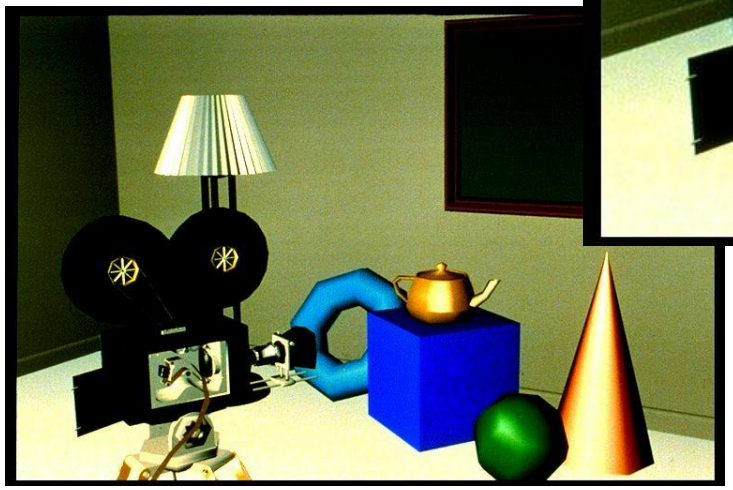
Shading Models Compared

computação
Universidade Federal de Pelotas

Phong Shading:
Interpolation of vertex surface normals

Note: specular highlights but no shadows. Still a pure local illumination model





Gouraud Shading

Homework

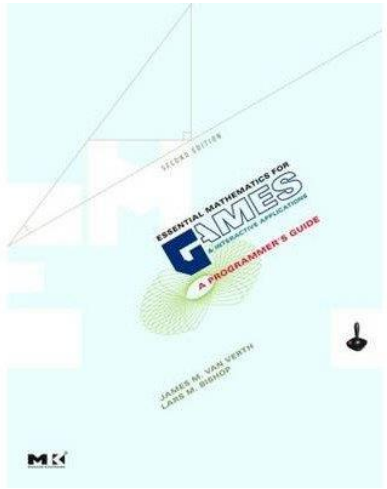
computação
Universidade Federal de Pelotas

- https://webgl2fundamentals.org/webgl/lessons/webgl-3d-lighting-directional.html
- https://webgl2fundamentals.org/webgl/lessons/webgl-3d-lighting-point.html
- https://webgl2fundamentals.org/webgl/lessons/webgl-3d-lighting-spot.html

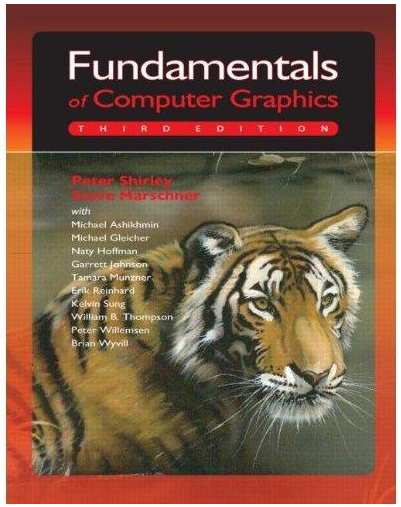
  - Modify the tutorial to include a switch to change between the shadings:
    - Flat/constant
    - Gouraud
    - Phong
  - Multiple lights (at least 5)
  - Different materials
    - Render at least two models with different material

Books

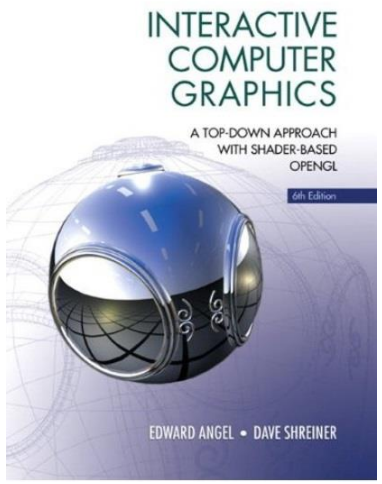
computação
Universidade Federal de Pelotas



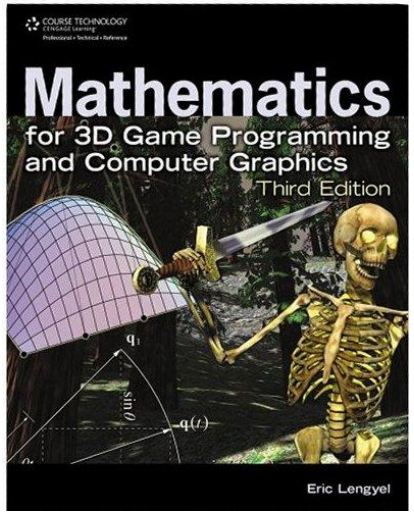
Chapter 8



Chapter 10



Chapter 5



Chapter 7

Read at least one!

Videos

computação
Universidade Federal de Pelotas

- https://youtu.be/lH61rdpJ5v8
- https://youtu.be/jWaeoKI4cwY
- https://youtu.be/tAqmloARskQ
- https://youtu.be/wUK11cAmxdQ