Universität Bielefeld

# Virtual Reality for Human Computer Interaction

## Appearance: Phong Lighting Model

# Ambient light

- Ambient reflection depends on **ambient light intensity** $I_{a\lambda}$ and object's **ambient material properties**:
  - Objects diffuse color $O_d$ ($O_{dr}$, $O_{dg}$, $O_{db}$)
  - Overall fraction reflected is the *ambient-reflection coefficient* $k_a$, (0 to 1)
  - The overall fraction of primary reflected is: $k_a\ O_{d\lambda}$
  - Specification allows independent control of the overall intensity of reflection and of its color.

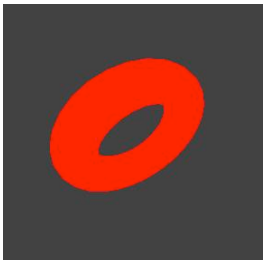
➡ **<u>The illumination model at an object point thus far:</u>**

$$I_\lambda = I_{a\lambda}\ k_a\ O_{d\lambda}$$

➢ Ambient light is not viewer location dependent

➢ Resulting images:

+



# Ambient light in OpenGL

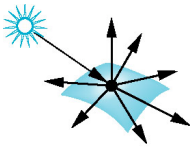
- Enable a global ambient light:

```
float globalAmbient[] = {r,g,b,1};
glLightModelfv (GL_LIGHT_MODEL_AMBIENT, globalAmbient);
```

- OpenGL allows an ambient term in individual lights (e.g., **`GL_LIGHT0`**)

- Specify ambient material property:
  - `float ambient[] = {r,g,b,1};`
  - `glMaterialfv (GL_FRONT_AND_BACK, GL_AMBIENT, ambient);`
- Note that $k_a$ and $O_{d\lambda}$ are combined.

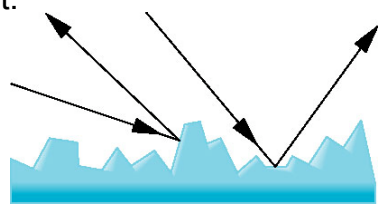


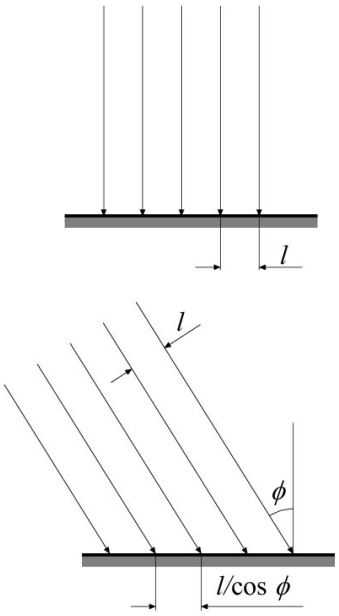


# Diffuse Reflection

(b)

- Scattering of Reflective Light.
- Consider how a dull, matte surface (e.g., chalk) scatters light:





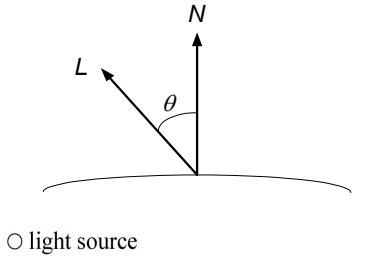

  - When its orientation is fixed relative to a light, its illumination looks the same from all viewing angles.
  - When its orientation changes relative to a light, its illumination changes.
    - It is brightest when the light shines directly on it.
    - It is dimmer when it makes an angle to the light.
  - This reflection is diffuse (**Lambertian**) reflection.

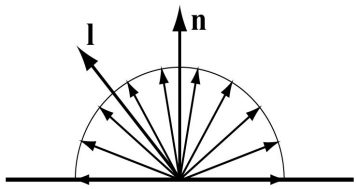
# Lambertian Reflection

- Diffuse (Lambertian) reflection: what we see is according to Lambert's law the vertical component of the incoming light
- This vertical component at *p* is :

  $I_{p\lambda}$ cos(θ) or

  $I_{p\lambda}$ (**N** · **L**),where:
  - The unit surface normal at a point, p, is **N**.
  - L is a unit vector pointing to the light source
  - θ is the angle between **N** and **L**.
- The reflected light is 0 for θ > 90 degrees






# Lambertian Reflection

- The diffusely reflected light depends on the surface's material properties :
  - Objects diffuse color $O_d$ ($O_{dr}$ , $O_{dg}$ , $O_{db}$)
  - The overall fraction reflected is the *diffuse-reflection coefficient* $k_d$ , range(0 to 1)
  - The overall fraction of primary reflected is: $k_d\, O_{d\lambda}$
- Given point light source, the diffuse intensity at it is:

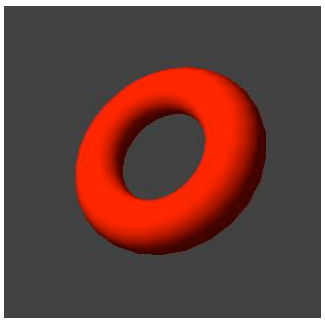
$$I_{p\lambda}\, k_d\, O_{d\lambda}\, (\mathbf{N} \cdot \mathbf{L})$$

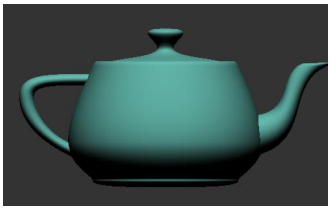
# Lighting model continued

➡ **<u>The illumination model at a point is thus so far:</u>**

$$I_\lambda = I_{a\lambda}\, k_a\, O_{d\lambda} + I_{p\lambda}\, k_d\, O_{d\lambda}\, (\mathbf{N} \cdot \mathbf{L})$$

- Diffuse lighting is not viewer location dependent
- The dot product is calculated at every point
- The L vector is calculated at every point except:
  - The light's position is infinitely far away.
  - All rays are parallel by the time they reach the scene.
- The resulting images look like:







# Diffuse Reflection in OpenGL

- Specify the light's color:

```
float diffuse0[] = {r,g,b,1};
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
```

- Specify the light's direction:

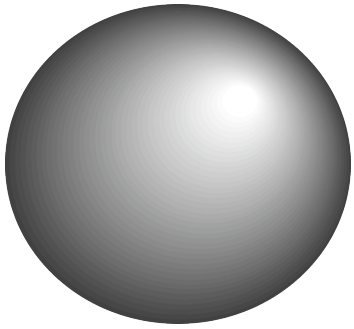
```
float direction0[] = {dx,dy,dz,0};
glLightfv(GL_LIGHT0, GL_POSITION, direction0);
```

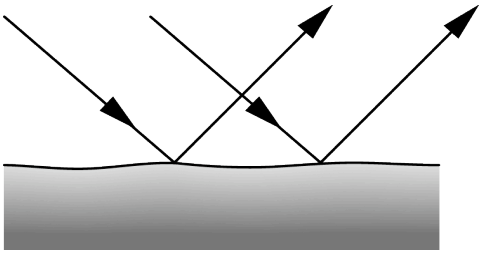
  - The parameter being set is **`GL_POSITION`**
  - The 0 in the last element of **`direction0`** indicates that this light is a directional light.
- Specify diffuse material property:

```
float diffuse[] = {r,g,b,1};
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, diffuse);
```

# Specular Reflection

- Consider a glossy, shiny surface (e.g., plastic, metal).
  - The surface reflects a bright highlight.
  - The highlight changes with viewing angle.
- This reflection is specular reflection.
- Reflection is highest in a certain direction.
- Reflection intensity depends on angle between reflection distribution and viewer position.

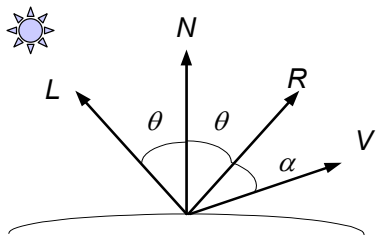


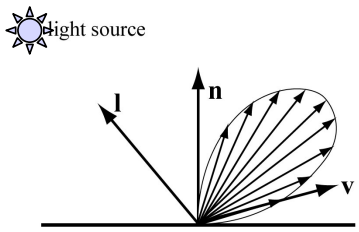




# Specular Reflection

- More precisely:
  - **N** is the unit normal at point p.
  - **L** is the unit vector pointing to the light source.
  - $\theta$ is the angle between **N** and **L**.
  - **R** is the vector of mirror reflection.
    - **R** also makes angle with N.
    - **R** is on the "other side" of **L**.
  - **V** is a unit vector pointing to the camera.
  - $\alpha$ is the angle between **R** and **V**.
- The highlight's visible intensity depends on:
  - $\alpha$
    - The highlight is most intense when $\alpha = 0$
    - The highlight becomes dimmer as $\alpha$ grows.
  - material properties
    - Example: Mirror reflects only with $\alpha = 0$
    - A Mirror is a Perfect Reflector



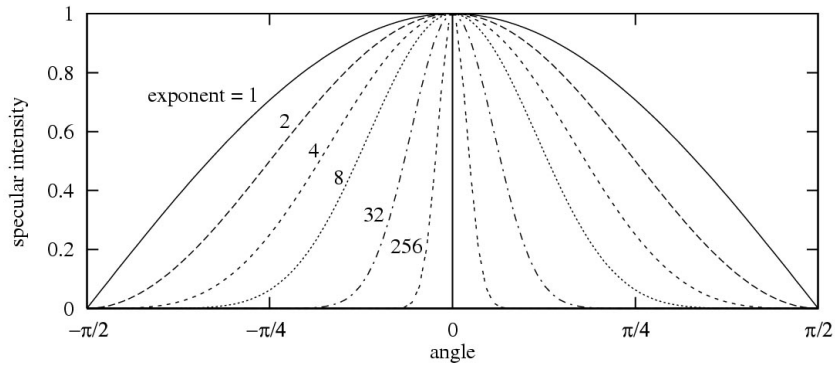



# Phong Model for Non-Perfect Reflectors

- A light of intensity $I_{p\lambda}$ produces a highlight intensity proportional to

$$I_{a\lambda} \, ( \cos (\alpha) )^{n}.$$

- The exponent, *n* is a material property
  - *(specular-reflection coefficient)*
  - Varies from 1 to several hundred
    (from broad gentle falloff to sharp focused falloff):

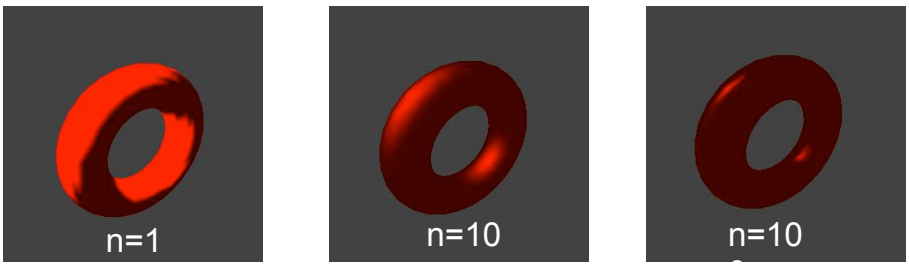





# Specular Reflection

- Other material properties affect the intensity specularly reflected.
  - The overall fraction of light reflected is $W(\theta)$, often taken to be the constant $k_s$ (ranges 0 to 1)
  - The fraction of primary $\lambda$ reflected is $O_{s\lambda}$
- The specular intensity is thus:

$$I_{p\lambda} \, k_s \, O_{s\lambda} \, ( \cos (\alpha) )^{n} \quad \text{or} \quad I_{p\lambda} \, k_s \, O_{s\lambda} \, ( R \cdot V )^{n}$$

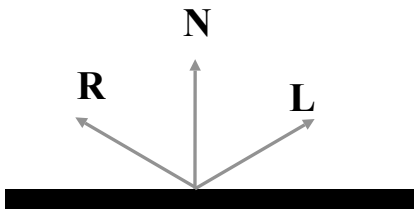



➡**<u>The illumination model at a point is thus so far:</u>**

$$I_{\lambda} = I_{a\lambda} \, k_a \, O_{d\lambda} + I_{p\lambda} \, k_d \, O_{d\lambda} \, (\mathbf{N} \cdot \mathbf{L}) + I_{p\lambda} \, k_s \, O_{s\lambda} \, ( \mathbf{R} \cdot \mathbf{V} )^{n}$$

# Approximating Phong Lighting

- Calculating Phong Lighting equation requires calculation of perfect light reflection vector.

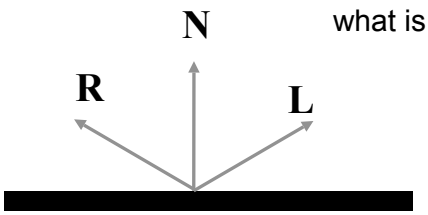



- How is the reflection vector calculated?

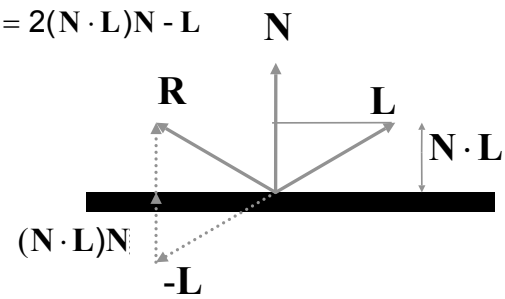


# Approximating Phong Lighting

- Calculating Phong Lighting equation requires calculation of perfect light reflection vector.



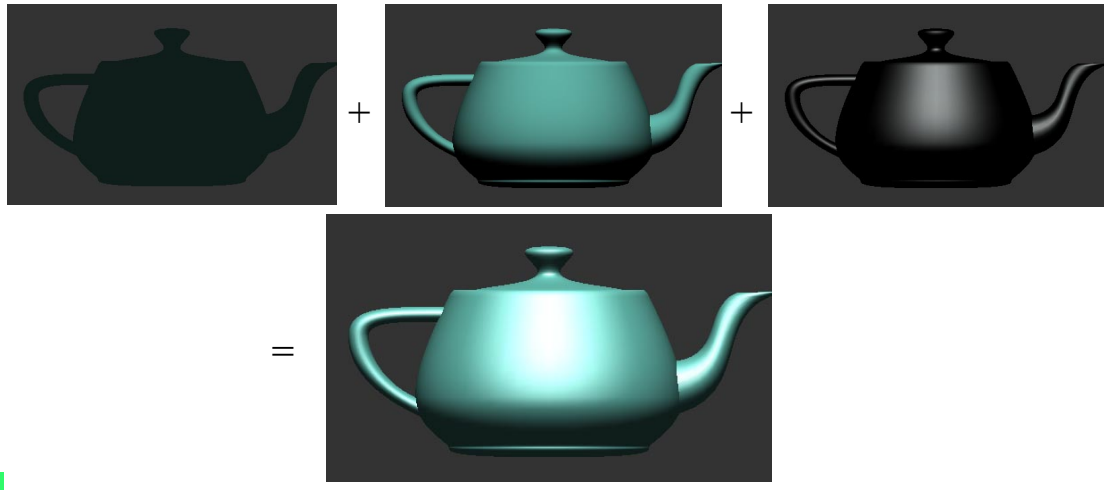

what is R? $\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L}$




- Term is maximum when R and V are equal.
- A popular variation by (Blinn, 1977): $I_{spec} = (\mathbf{N} \cdot \mathbf{H})^n$
- Where H is the normalized halfway vector between L and V: $\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|}$
- This approximation is expressed as: $(\mathbf{R} \cdot \mathbf{V})^n = (\mathbf{N} \cdot \mathbf{H})^{4n}$

# Applying the lighting model

$$I_\lambda = I_{a\lambda}\, k_a\, O_{d\lambda} + I_{p\lambda}\, k_d\, O_{d\lambda}\, (\mathbf{N} \cdot \mathbf{L}) + I_{p\lambda}\, k_s\, O_{s\lambda}\, (\mathbf{R} \cdot \mathbf{V})^n$$





# Specular Reflection in OpenGL

- Specify the light that can be specularly reflected:

```
float specular0[] = {r,g,b,1};
glLightfv(GL_LIGHT0, GL_SPECULAR, specular0);
```

- Specify specular material properties:

```
glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, n);
float specular[] = {r,g,b,1};
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, specular);
```

  - Note that $k_s$ and $O_{s\lambda}$ are combined.

# Light-Source Attenuation

- To deal with light source distance, we introduce: $f_{att}$
- One option – light energy falls off at inverse square:

$$f_{att} = \frac{1}{d_l^{\,2}}$$

  - What can we say about the practical issues of the term?
  - progression, missing parametrization, physical inadequacy in rest of calculation and device chain

  ➡ in In reality – this does not work well
- Alternative:

$$f_{att} = \min\left(\frac{1}{c_1 + c_2 d_L + c_3 d_L^{\,2}}, 1\right)$$

  - Where $c_1$, $c_2$, and $c_3$ are user defined constants for a light source
- OpenGL: Attenuation can be set for each light source



# Atmospheric Attenuation

- Handles distance from observer to object
- more distant objects rendered with lower intensity than closer ones
- Define front and back depth-cue planes, and a (low intensity) color $I_{dc\lambda}$ at the back depth-cue plane
- Set : $I_\lambda' = f_{fog} I_\lambda + (1 - f_{fog})\, I_{dc\lambda}$
  - Where $f_{fog} = 0$ for objects in front of front plane,
    $f_{fog} = 1$, for objects behind back plane,
    $f_{fog}$, the **fog factor**, increasing between front and back planes
  - if $f_{fog}$ increases, fog effect decreases…
- FOG is OpenGL's implementation of atmospheric attenuation

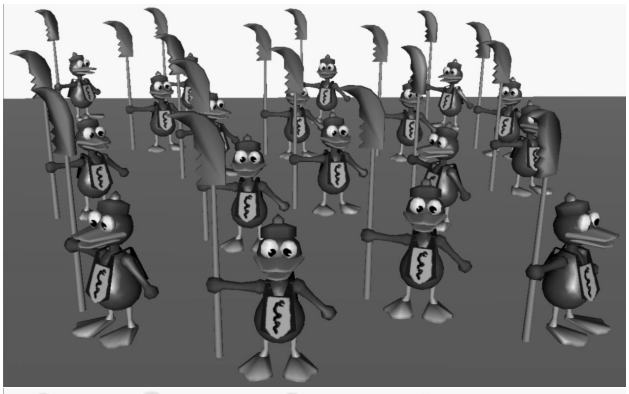
➡ **<u>The illumination model at a point is finally:</u>**

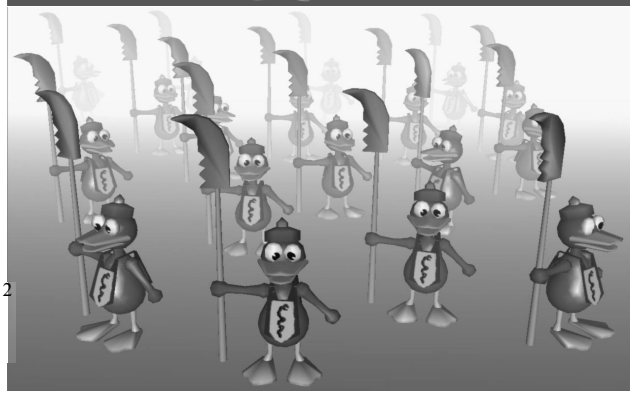
$$I_\lambda = I_{a\lambda}\, k_a\, O_{d\lambda} + f_{att}\,[I_{p\lambda}\, k_d\, O_{d\lambda}\,(\mathbf{N} \cdot \mathbf{L}) + I_{p\lambda}\; k_s\, O_{s\lambda}\,(\mathbf{R} \cdot \mathbf{V})^n]$$

$$I_\lambda' = f_{fog} I_\lambda + (1 - f_{fog})\, I_{dc\lambda}$$

# Fog example

- Often just a matter of
  - Choosing fog color
  - Choosing fog model
  - Turning it on
- How to compute $f_{fog}$?
- 3 ways
  - linear: $f_{fog} = \frac{z_{end} - z_p}{z_{end} - z_{start}}$
  - exponential: $f_{fog} = e^{-d_f z_p}$
  - exponential-squared: $f_{fog} = e^{-(d_f z_p)^2}$

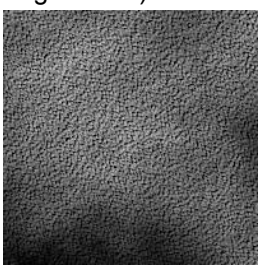


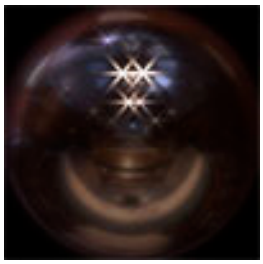




# Appearance

- Appearance can be greatly enhanced by using textures.
- Textures are 2d or 3D arrays of values which are fed as an additional parameter source into the render pipe's calculations.
- Textures are used to describe appearance details.
- Simple textures are images which define a pixels color or a color modification of the color produced by the lighting calculation.
- Normal maps or bump maps describe fine grained surface structures.
- Shadow maps describe shadows cast onto a geometry.
- Environment maps reflect light from the environment (e.g., for simulating mirrors)
- …

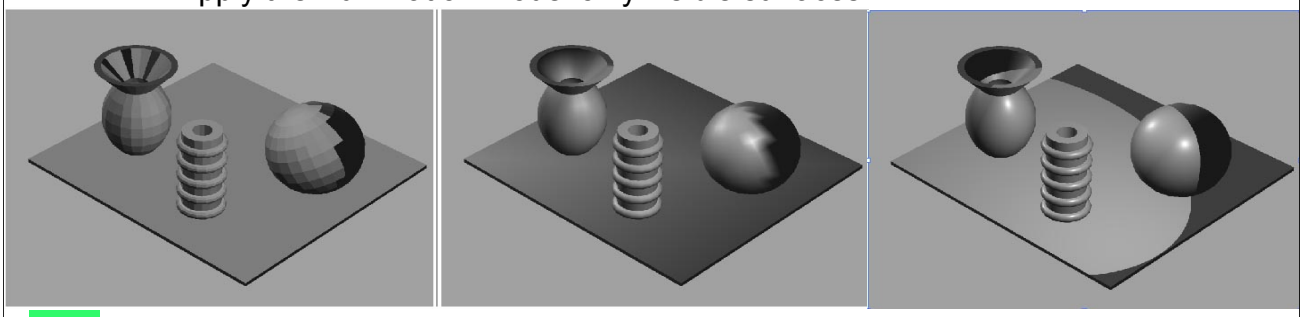




# What now?

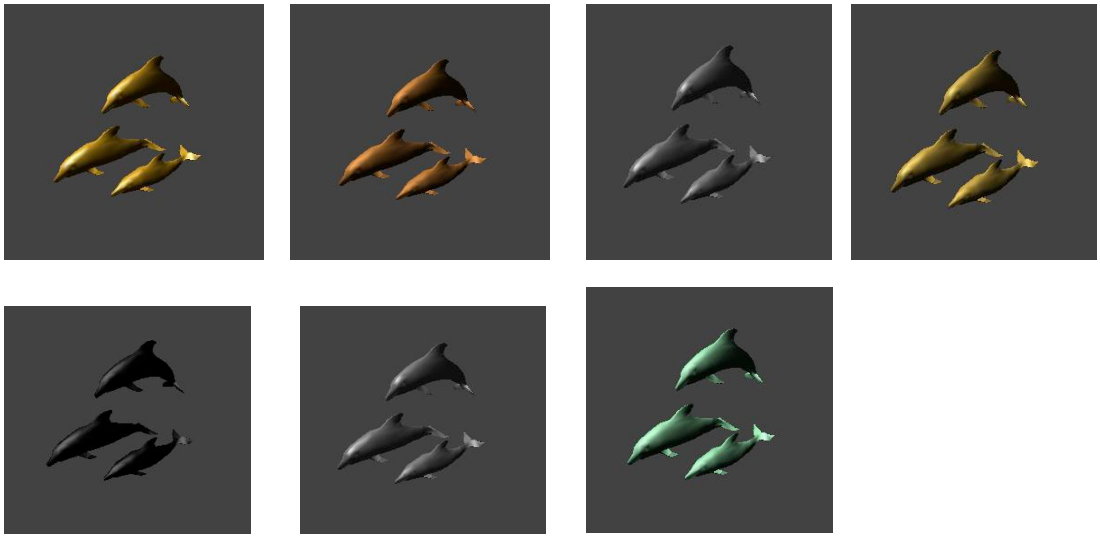
**Shading**

- Ideally, the renderer should apply the illumination model at every visible point on each surface
- This approach requires too much computation.
- As an alternative:
  - Apply the illumination model at a subset of points.
  - Interpolate the intensity of the other points.
  - Apply the illumination model only visible surfaces





# examples



# Further Expansions of the Illumination Models

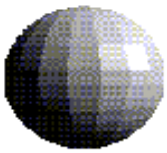
- Presented illumination model only involves:
  - Light sources
  - Materials at object point
  - Known as a “Local Model”
- Real lighting involves:
  - Light reflection from one object to another
    - (Global Models)
    - I.e. additional lighting sources for an object point
  - Transparency
- Raytracing, Radiosity Approaches



# “Local Model” Shading Models

- Determines the shade of a object point or pixel by applying the illumination model
- The models are only loosely physical, and emphasize:
  - Empirical success
  - Efficiency

# Flat (Constant) Shading



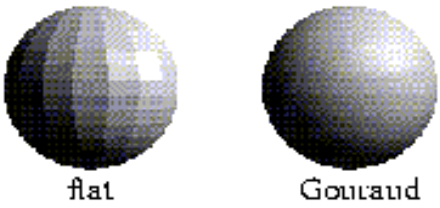
- Sample illumination at one point per polygon.
- Use constant interpolation:
  all other points on the polygon get that point's intensity.
- This approach would be valid if:
  - The true surface really is faceted, so **N** is constant.
  - The light source is at infinity, so **L** is constant.
  - The viewer is at infinity, so **V** is constant.



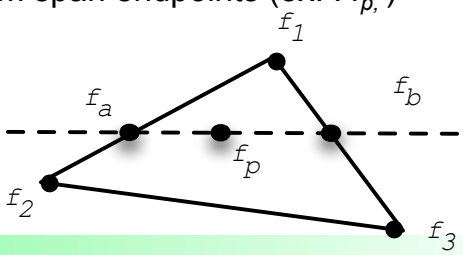
# Flat shading in OpenGL

- Just enable it:
  - `glShadeModel(GL_FLAT);`

# Gouraud Shading




- Apply the illumination model at each polygonal vertex.
  - (Example: $f_1$, $f_2$, $f_3$)
- Interpolate intensities as part of scan conversion
- Bi-linear interpolation:
  - Interpolate span endpoints from edge vertices (ex. . $f_a$, , $f_b$, )
  - Interpolate points within a span from span endpoints (ex. . $f_p$, )






# Gouraud Shading (Continued)

- Reduces Mach bands (but not entirely).
- Misses interior highlights
- Smears highlights along edges
- Some repetitive 3D patterns can be missed completely

# Gouraud Shading in OpenGL

- Just Enable It:
  - `glShadeModel (GL_SMOOTH);`
- Set Normals for all Vertices