Universität Bielefeld

Virtual Reality for Human Computer Interaction

Appearance: Lighting

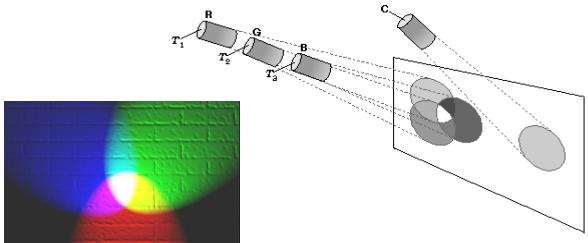
Representation of Light and Color

Representation of Light and Color

- Do we need to represent all I_λ to represent a color C(I) ?

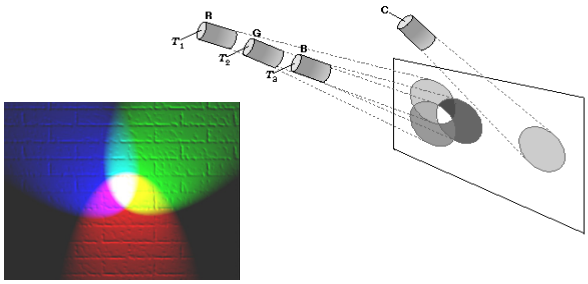
Representation of Light and Color

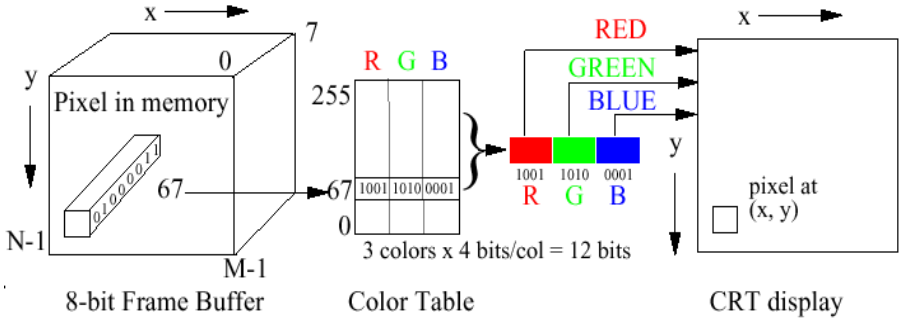
- Do we need to represent all I_λ to represent a color C(I) ?
- No – we can approximate using a three-color additive system (taking into account the described problems)



Representation of Light and Color

- Do we need to represent all I_λ to represent a color C(I) ?
- No – we can approximate using a three-color additive system (taking into account the described problems)
- Frames can be displayed using RGB system:

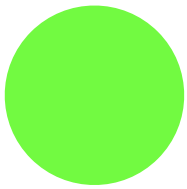




Motivation

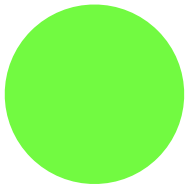
Motivation

- Suppose we build a model of a green sphere using many polygons and just color it.
  We get something like:



Motivation

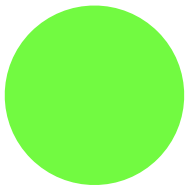
- Suppose we build a model of a green sphere using many polygons and just color it.
  We get something like:



- ➢ The image of the sphere looks flat!

Motivation

- Suppose we build a model of a green sphere using many polygons and just color it.
  We get something like:

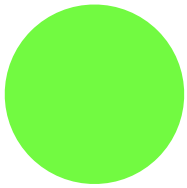


➢ The image of the sphere looks flat!

- But light-material interactions should cause each point to have a different color or shade to generate depth perception.

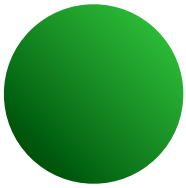
Motivation

- Suppose we build a model of a green sphere using many polygons and just color it.
  We get something like:



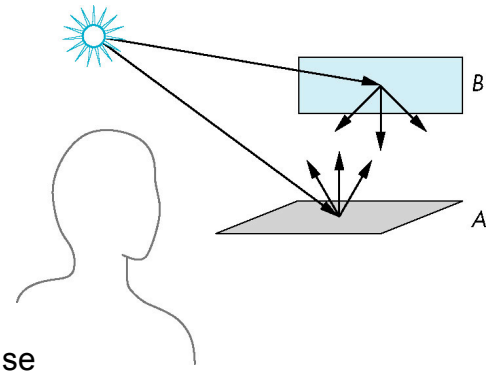
➢ The image of the sphere looks flat!

- But light-material interactions should cause each point to have a different color or shade to generate depth perception.
- Need to consider
  - Light sources
  - Material properties
  - Location of viewer
  - Surface orientation



Principle Lighting Model

1. **Lighting** (or illumination): Description or model of light-object-eye interaction.
2. **Shading:** (Algorithmical) lighting application across a primitive.



- Physically, surfaces may reflect or emit light or both.
- Color that we see is determined by multiple interactions between light and surfaces.
- Recursive process:
  Light from A is reflected on B is reflected on A is reflected on B…
- Equations could be derived which use principles like conservation of energy to describe this process.
- This results in integral equation which can not be solved analytically…
- …but **global model** lighting approaches like **ray-tracing** and **radiosity** use numerical approximations which are becoming real-time capable (depending on parameterization and HW-support).

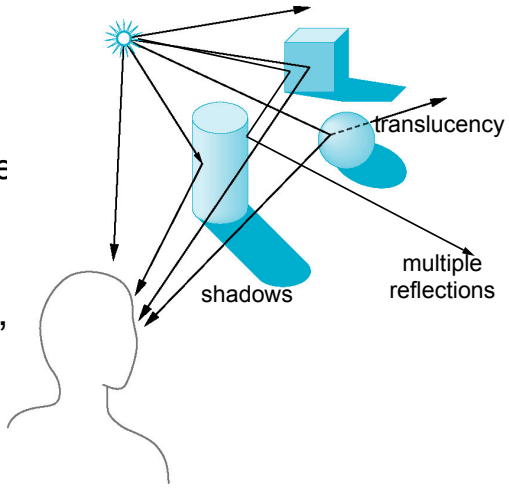
Principle Lighting Model

Principle Lighting Model

- Correct shading requires a global calculation involving all objects and light sources.
  - Incompatible with pipeline model which shades each polygon independently (local rendering).
  - Numerical solutions are expensive but can in principle be sped up using dedicated hardware.

Principle Lighting Model

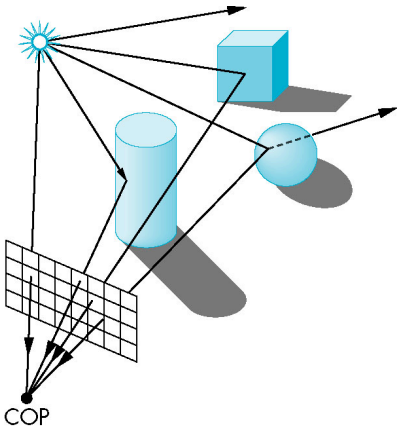
- Correct shading requires a global calculation involving all objects and light sources.
  - Incompatible with pipeline model which shades each polygon independently (local rendering).
  - Numerical solutions are expensive but can in principle be sped up using dedicated hardware.
- For real time computer graphics, approaches are utilized which imitate physically correct light-matter-eye interaction, hence which “look right”.
  - Exist many techniques for approximating global effects



Local Lighting Model

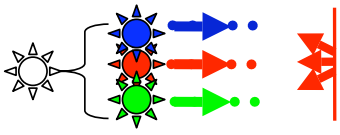
Local model:

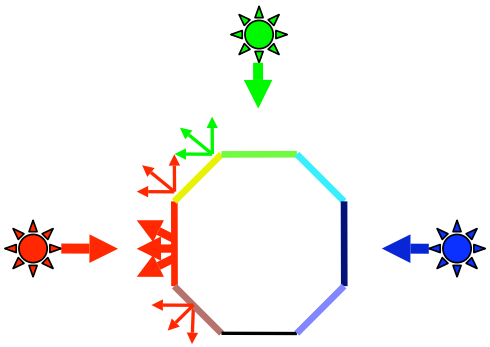
- Following rays of light from light emitting surfaces (**light-sources**) instead of looking at a global energy balance.
- Derive a model which describes how these rays interact with reflecting surfaces.
- Will focus on single interaction in contrast to multiple interaction (like used in ray-tracing).
- This approach requires **light sources** and **reflection model**.
- Viewer sees only light which reaches eye.
  - No reflection inbetween:
    Perception of light source's color.
  - With surface reflection:
    Perception based on light source's color and surface material.
- Viewer's eye is exchanged for COP (Center of Projection) and projection plane.



Local Lighting Model

- Light that strikes an object is
  - partially absorbed and
  - partially scattered (reflected).
- The amount reflected determines
  - the color and
  - brightness of the object.
  - ➢ A surface appears red under white light because the red component of the light is reflected and the rest is absorbed
- The reflected light is scattered in a manner that depends on
  - the smoothness and
  - orientation of the surface.

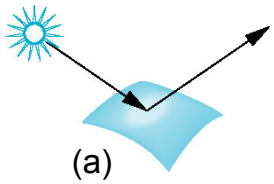




Reflecting Surfaces

Reflecting Surfaces

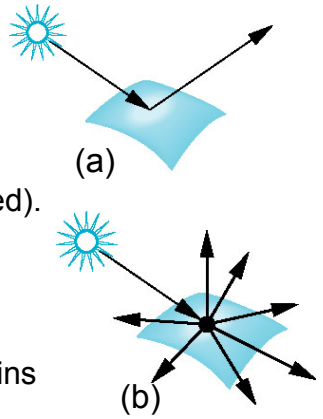
- (a) Specular surfaces:
  - Appear shiny because most of reflected light is scattered in a narrow range of abgles close to angle of reflection.
  - Ideal reflectors: Mirrors (parts can be still absorbed).
  - Angle of incidence is equal angle of reflection.



(a)

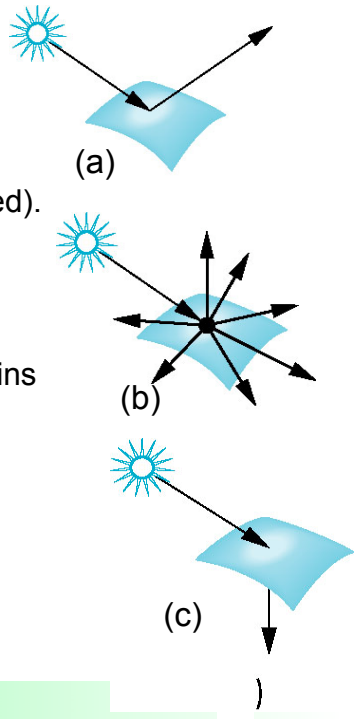
Reflecting Surfaces

- (a) Specular surfaces:
  - Appear shiny because most of reflected light is scattered in a narrow range of abgles close to angle of reflection.
  - Ideal reflectors: Mirrors (parts can be still absorbed).
  - Angle of incidence is equal angle of reflection.
- (b) Diffuse surfaces:
  - Reflected light is scattered in all directions.
  - E.g., walls painted with matte or flat paint or terrains seen from hight.
  - Perfect diffuse surfaces scatters equally in all directions.



Reflecting Surfaces

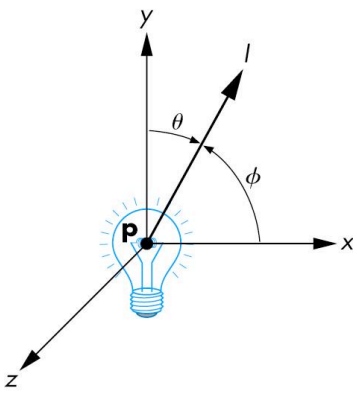
- (a) Specular surfaces:
  - Appear shiny because most of reflected light is scattered in a narrow range of abgles close to angle of reflection.
  - Ideal reflectors: Mirrors (parts can be still absorbed).
  - Angle of incidence is equal angle of reflection.
- (b) Diffuse surfaces:
  - Reflected light is scattered in all directions.
  - E.g., walls painted with matte or flat paint or terrains seen from hight.
  - Perfect diffuse surfaces scatters equally in all directions.
- (c) Transluscent surfaces:
  - Allow some light to penetrate the surface and to emerge from another location -> Refraction.
  - Some incident light may be reflected as well.



Light Sources

Light Sources

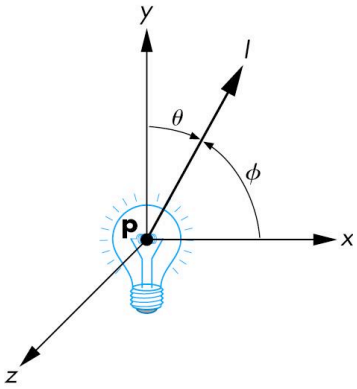
- In general, light sources should integrate light coming from all points on the source.
- Light can leave a surface by
  - self-emission and/or
  - reflection.
- Each point p=(x,y,z) on the surface is characterized by
  - the direction of emission (θ, ϕ) and
  - the intensity of energy at each wavelength λ and hence
  - the illumination function

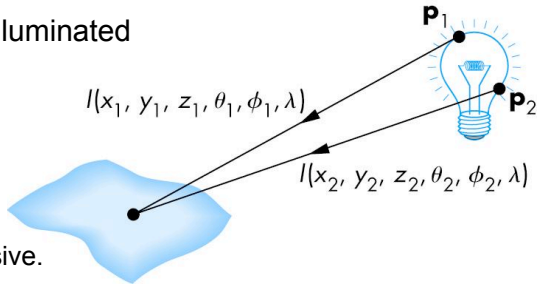


Light Sources

- In general, light sources should integrate light coming from all points on the source.
- Light can leave a surface by
  - self-emission and/or
  - reflection.
- Each point p=(x,y,z) on the surface is characterized by
  - the direction of emission (θ, ϕ) and
  - the intensity of energy at each wavelength λ and hence
  - the illumination function
- To calculate the source's contribution to an illuminated surface one has to
  - integrate over the source's surface,
  - account for the emission angles and
  - account for the distance between source and surface.

  ➢ Integration (analytical or numerical) is expensive.

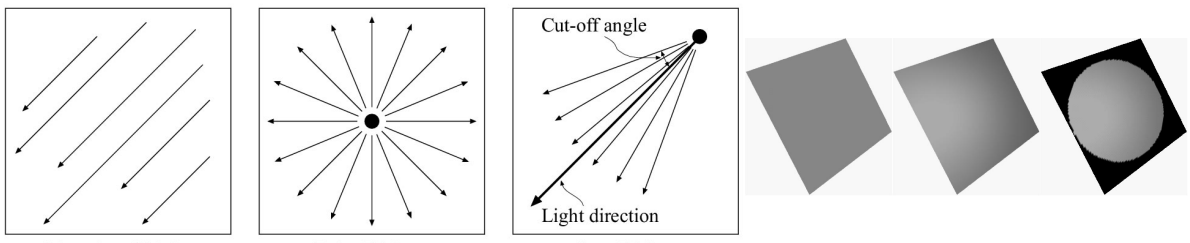




Light Sources

- An approximation to light-material interaction
  - uses 4 different light sources to
  - calculate an **intensity function I**
  - using the three color model of the human visual system.

1. **Ambient light**: Same amount of light everywhere, can model contribution of many sources and reflecting surfaces.
2. **Point source**: Model with position and color.
3. **Distant (directional) light**: Point source in infinite distance (parallel rays).
4. **Spotlight**: Point source with restricted light.



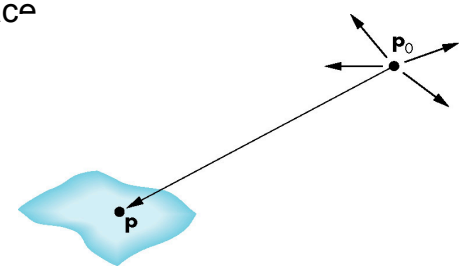
Directional Light | Point Light | Spot Light

Ambient Light

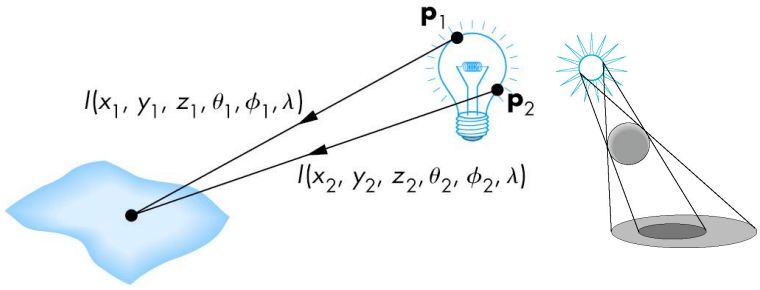
- Near uniform lighting created by highly diffused light sources.
- One could model all light sources and interactions or use a concept called “ambient light” which
  - lights all surfaces uniformly.
  - is not viewer location dependent.
- $\mathbf{m}_{amb}$ vector is a material attribute
- $\mathbf{s}_{amb}$ vector is a light source attribute

Point Light Sources

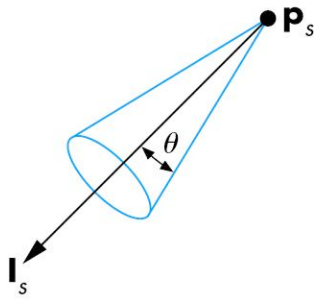
- Ideal point emits light in all directions.
- Intensity of illumination is inverse square of distance between source and surface



Use of point sources is more a matter of efficiency rather than realism as most sources have a dimension:

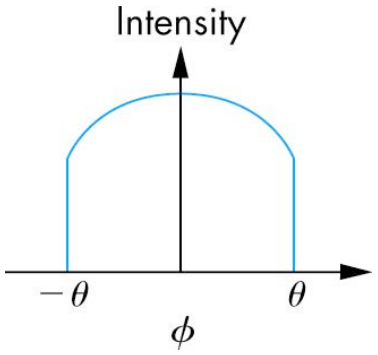


Spotlights and Distant Lights



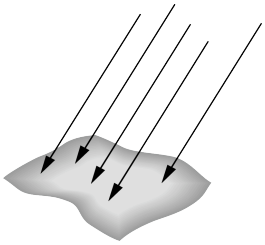
Spotlights:

- Point source with limited direction.
- Point source Ps in a direction I_s and a width of θ.
- Spotlight attenuation:
  - Greater realism can be obtained by varying the intensity of light across the cone
  - Typical Function: $\cos(\phi) = \mathbf{S} \cdot \mathbf{I}$



Distant lights:

- Light sources that are distant to the surface
- Light is parallel:



Lighting in OpenGL

- Light sources can be turned on/off:

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
```

- Support: multiple lights
  - (but performance suffers)
- For each light:
  - Ambient, Diffuse, Specular per RGB
  - Position, Direction
  - Spotlight Exponent and Cutoff Angle
  - Light to Surface Distance Attenuation

Lighting At A Point

- Lighting at a point on an object's surface:

```
For each color in(Red, Green, Blue):
   For each light source:
       For each light type (ambient, diffuse, specular):
              Determine the amount of light reaching the point
              (Typically Ignore Shadowing)
              Determine the amount of light reflected
              (Based on properties of the surface)
```

- I_λ => sum of all light reflection from each light source

Lighting At A Point

- Illumination, I, at a point is modeled as the sum of several terms:
  - More terms give more plausible results.
  - Fewer terms give more efficient computations.
- Each additive term of I is expressed in primary colors, I_r, I_g and I_b, i.e. I_λ where λ is r, g, or b (typically defined as a range from 0 to 1)
- Each of these colors (I_λ) is computed independently.
- Components (I_λ), can be used to express how much light a source emits and a surface reflects.
- Total illumination: Sum of each light source

$$I_\lambda = I_{\lambda 1} + I_{\lambda 2} + I_\lambda$$

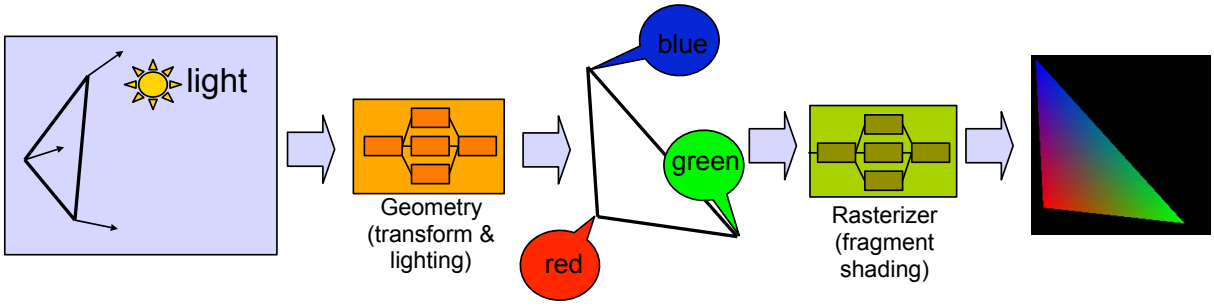
- Various solutions for dealing with possible overflow (>1), e.g.,
  - clamp to max allowable
  - normalize individual terms:

Applying a lighting model

Applying a lighting model

- Calculating lighting using objects defined by their surfaces:
  - In which coordinate system should the lighting be applied?
  - For which points on objects' surfaces should lighting be applied?
    - Sampling into surface may be to coarse
    - or may be to detailed and may produce unnecessary computational overhead
    - and sampling artifacts.

Applying a lighting model



- Calculating lighting using objects defined by their surfaces:
  - In which coordinate system should the lighting be applied?
  - For which points on objects' surfaces should lighting be applied?
    - Sampling into surface may be to coarse
    - or may be to detailed and may produce unnecessary computational overhead
    - and sampling artifacts.
- Idea:
  - Lighting calculation **per vertex** and surface approximation **in screen space**.
  - Supported by **pipeline architecture**.

Universität Bielefeld

Virtual Reality for Human Computer Interaction

Appearance: Phong Lighting Model

Ambient light

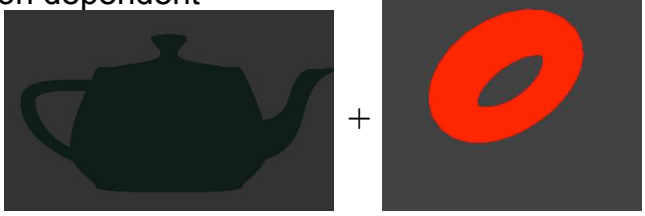
- Ambient reflection depends on **ambient light intensity** $I_{a\lambda}$ and object's **ambient material properties**:
  - Objects diffuse color O_d (O_{dr}, O_{dg}, O_{db})
  - Overall fraction reflected is the *ambient-reflection coefficient* k_a, (0 to 1)
  - The overall fraction of primary reflected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

➡ **The illumination model at an object point thus far:**

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

➢ Ambient light is not viewer location dependent

➢ Resulting images:



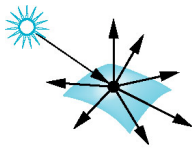
Ambient light in OpenGL

- 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`)

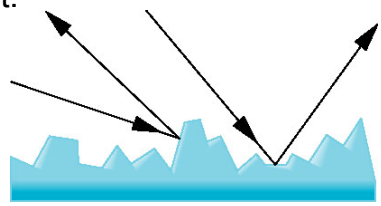
- Specify ambient material property:
  - `float ambient[] = {r,g,b,1};`
  - `glMaterialfv (GL_FRONT_AND_BACK, GL_AMBIENT, ambient);`
- Note that k_a and $O_{d\lambda}$ are combined.

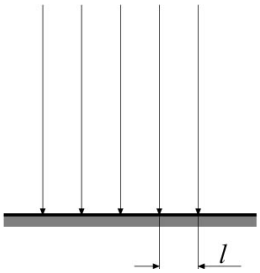


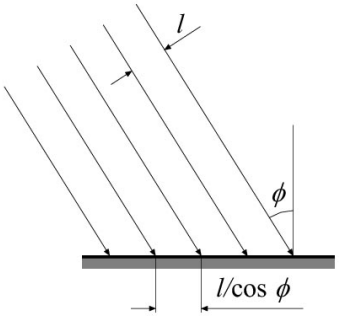
Diffuse Reflection

(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.

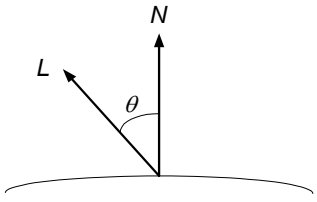
Lambertian Reflection

- 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 :

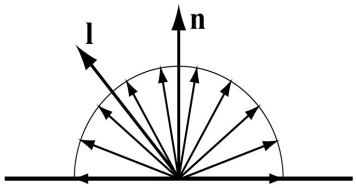
  $I_{p\lambda}$ cos(θ) or

  $I_{p\lambda}$ (**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



○ light source



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:

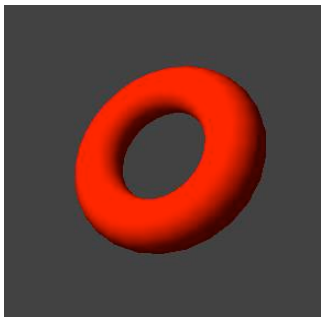
$$I_{p\lambda}\, k_d\, O_{d\lambda}\, (\mathbf{N} \cdot \mathbf{L})$$

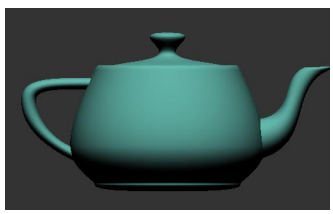
Lighting model continued

➡ **The illumination model at a point is thus so far:**

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

- 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);
```

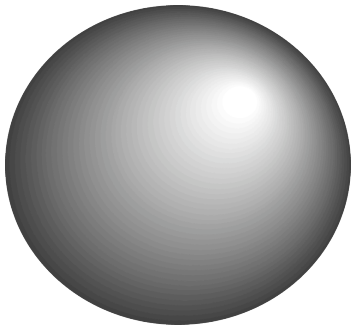
  - The parameter being set is **`GL_POSITION`**
  - The 0 in the last element of **`direction0`** indicates that this light is a directional light.

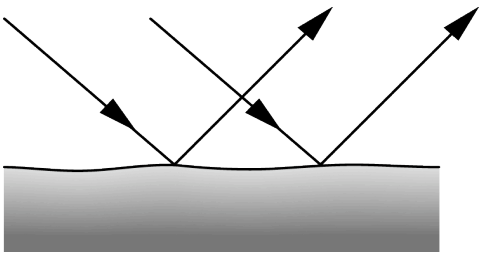
- Specify diffuse material property:

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


Specular Reflection

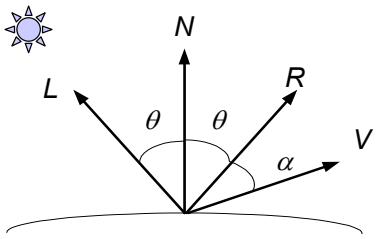
- Consider a glossy, shiny surface (e.g., plastic, metal).
  - The surface reflects a bright highlight.
  - The highlight changes with viewing angle.
- This reflection is specular reflection.
- Reflection is highest in a certain direction.
- Reflection intensity depends on angle between reflection distribution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

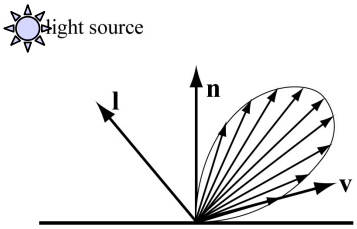




Specular Reflection

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



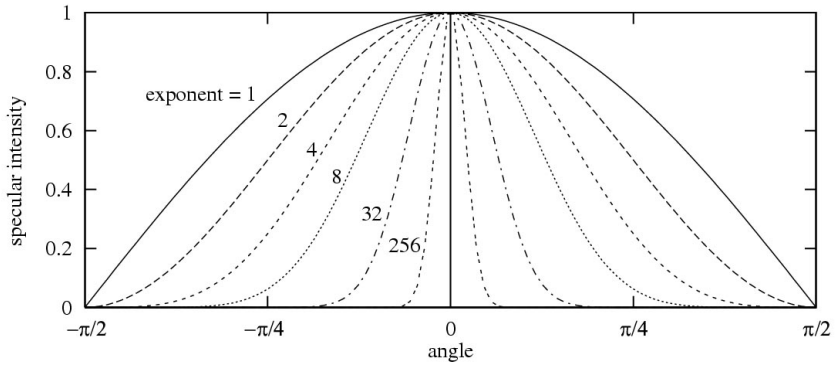


Phong Model for Non-Perfect Reflectors

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

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

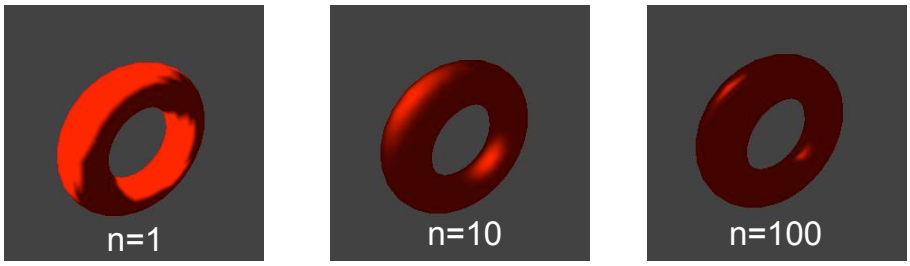
- The exponent, *n* is a material property
  - *(specular-reflection coefficient)*
  - Varies from 1 to several hundred
    (from broad gentle falloff to sharp focused falloff):



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 λ reflected is $O_{s\lambda}$
- The specular intensity is thus:

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

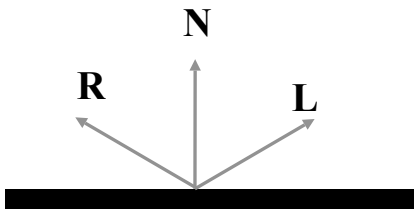


➡**The illumination model at a point is thus so far:**

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

Approximating Phong Lighting

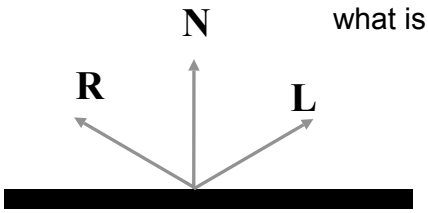
- Calculating Phong Lighting equation requires calculation of perfect light reflection vector.



- How is the reflection vector calculated?

Approximating Phong Lighting

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

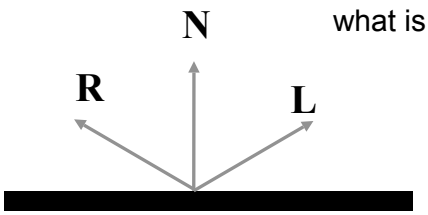


what is R?

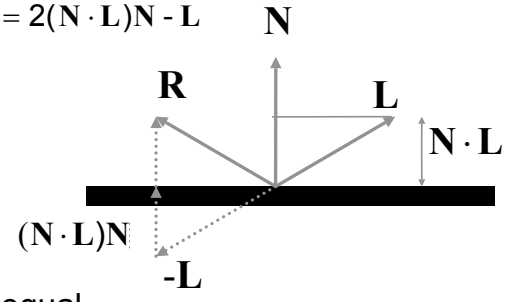
- Term is maximum when R and V are equal.
- A popular variation by (Blinn, 1977):
- Where H is the normalized halfway vector between L and V:
- This approximation is expressed as:

Approximating Phong Lighting

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



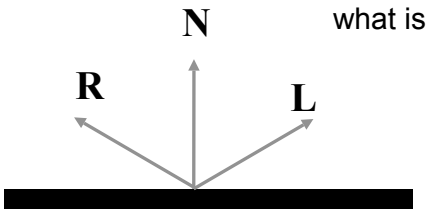
what is R? $\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L}$



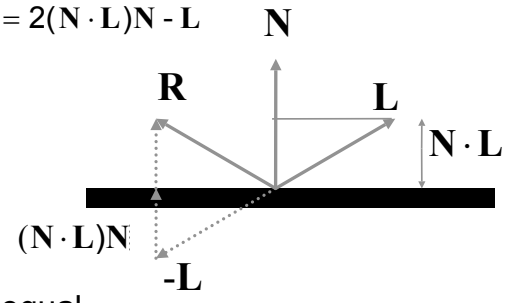
- Term is maximum when R and V are equal.
- A popular variation by (Blinn, 1977):
- Where H is the normalized halfway vector between L and V:
- This approximation is expressed as:

Approximating Phong Lighting

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



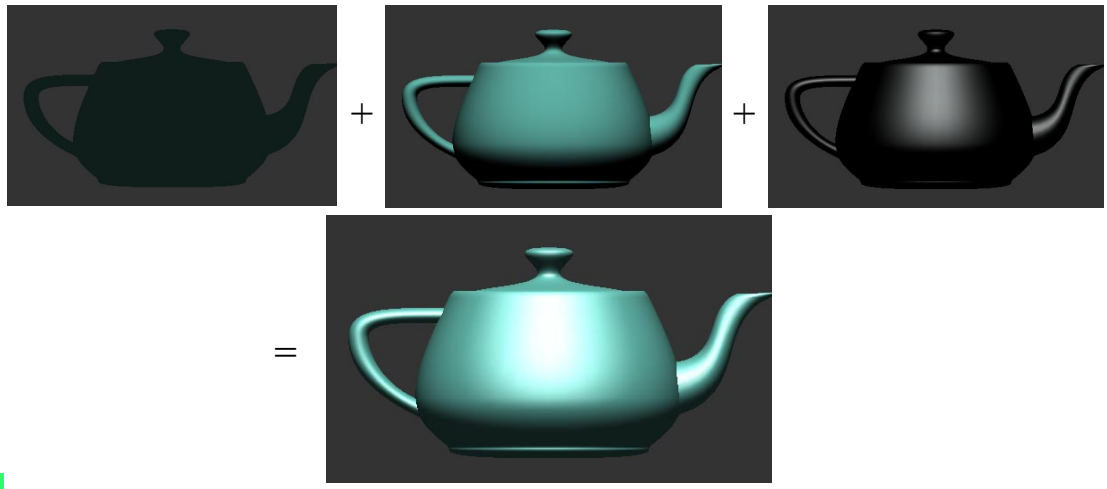
what is R? $\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L}$



- 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:

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

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?

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

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

➡ **The illumination model at a point is finally:**

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

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

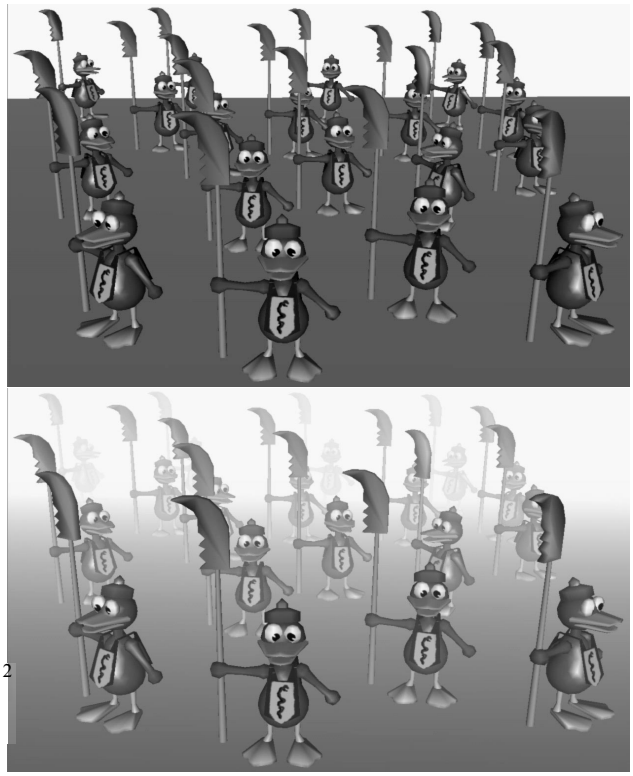
Fog example

$$f_{fog} = e^{-d_f z_p}$$

$$f_{fog} = e^{-(d_f z_p)^2}$$

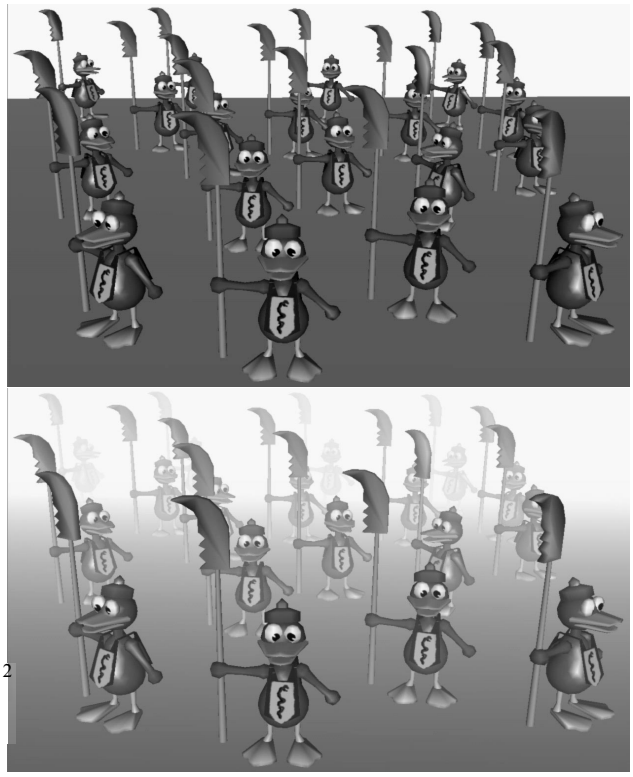
Fog example

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



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

- Appearance can be greatly enhanced by using textures.

Appearance

- Appearance can be greatly enhanced by using textures.

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.

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.

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.

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.

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.

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)

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?

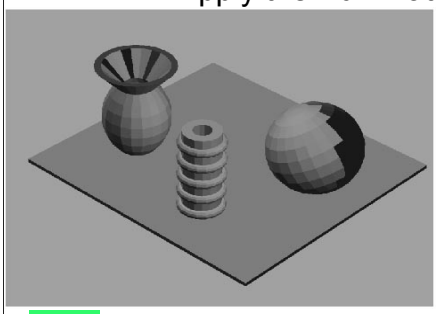
Shading

- 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

What now?

Shading

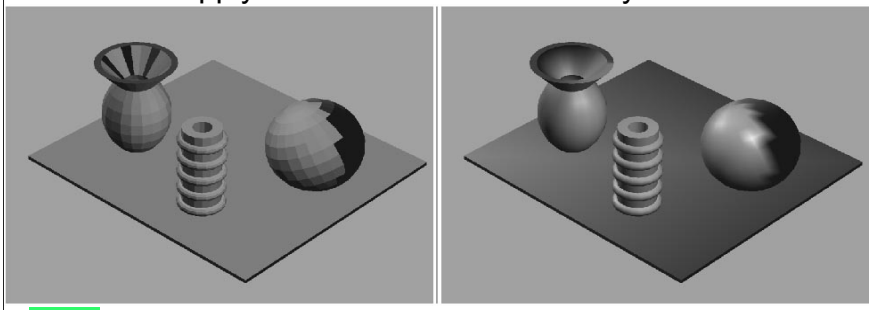
- Ideally, the renderer should apply the illumination model at every visible point on each surface
- This approach requires too much computation.
- As an alternative:
  - Apply the illumination model at a subset of points.
  - Interpolate the intensity of the other points.
  - Apply the illumination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



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

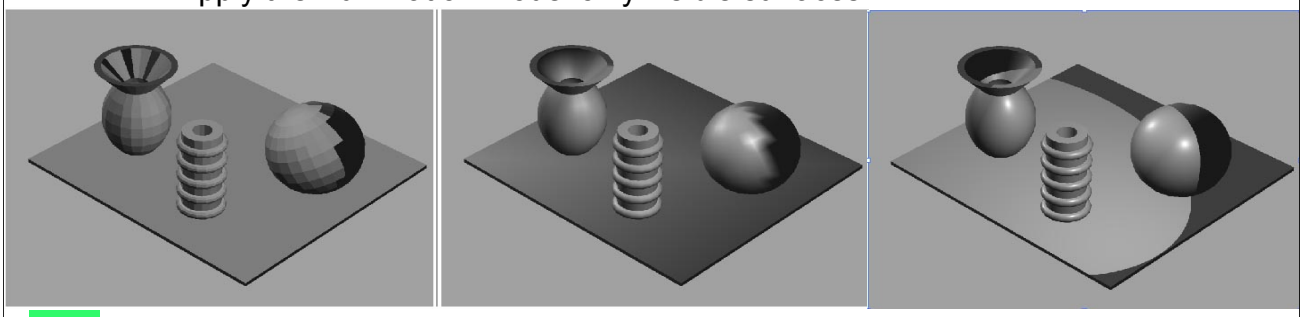


Realtime 3D Computer Graphics / Virtual Reality – WS 2006/2007 – Marc Erich Latoschik

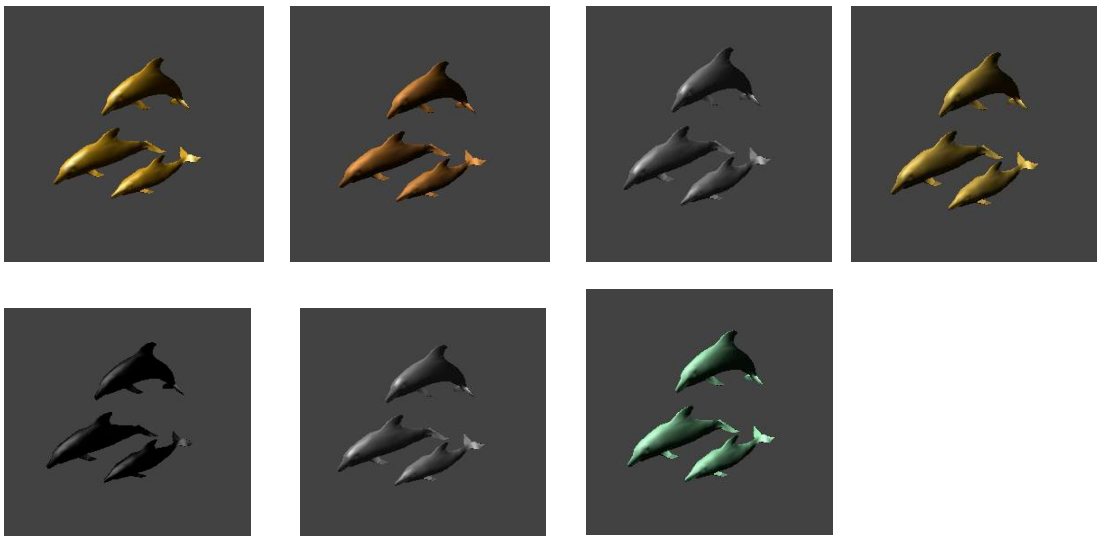
What now?

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



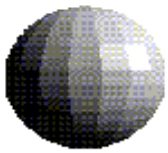
Further Expansions of the Illumination Models

- 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

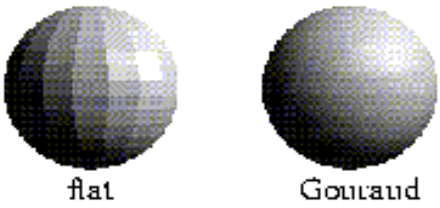


- Sample illumination at one point per polygon.
- Use constant interpolation:
  all other points on the polygon get that point's intensity.
- This approach would be valid if:
  - The true surface really is faceted, so **N** is constant.
  - The light source is at infinity, so **L** is constant.
  - The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

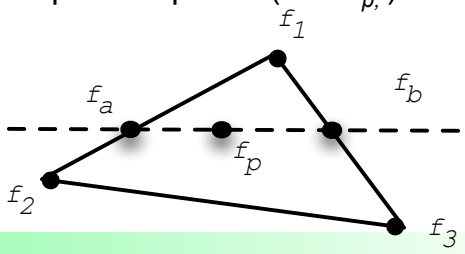
Flat shading in OpenGL

- 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)

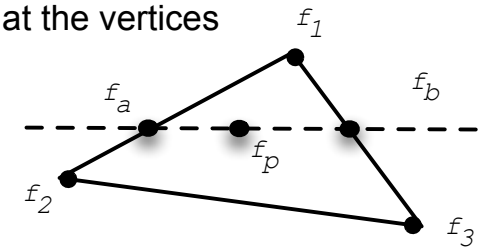
- Reduces Mach bands (but not entirely).
- Misses interior highlights
- Smears highlights along edges
- Some repetitive 3D patterns can be missed completely

Gouraud Shading in OpenGL

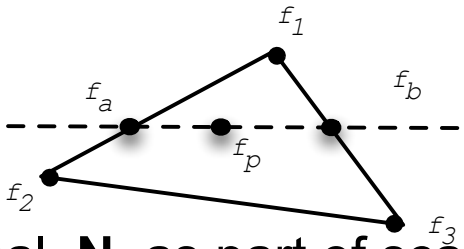
- Just Enable It:
  - `glShadeModel (GL_SMOOTH);`
- Set Normals for all Vertices

Phong Shading

- Improves the Gouraud Shading Model by:
  - Apply the illumination model at each pixel
    - This requires a normal (N) at each pixel
    - N at each pixel is interpolated from N at vertixes
  - Gouraud applies the illumination model at each vertex, then interpolates pixel value
    - This requires a normal only at the vertices



Phong Shading

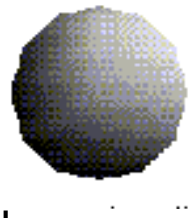


- Interpolate a pixel's normal, **N**, as part of scan conversion.
  - To get **N** at span endpoints, interpolate from edge vertices' normals.
  - To get **N** within a span, interpolate from span end-points.

Phong Advantages

- This approach avoid some errors in specular illumination that appear with Gouraud shading.
  - Gouraud misses specular highlights within polygons.
  - Gouraud spreads specular highlights along edges.
- Standard OpenGL does not support Phong shading!
- But Phong shading can be implemented using programmable shaders.

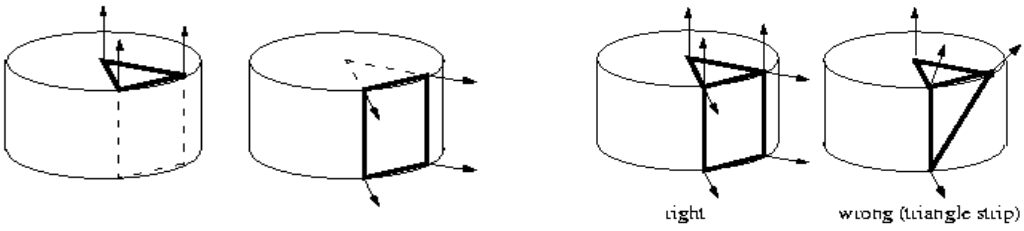
Problems Common to Gouraud and Phong Shading



- Silhouette edges are not smoothed
- Normal vector interpolation can cause problems:
  - Interpolation may mask regular changes
  - Interpolation ignores perspective distortion
    - Due to foreshortening, a change in scanlines does not correspond to a constant change in z in OpenGL eye coordinates.
    - So the scanline halfway between two vertices does not correspond to z halfway between the vertices' zs.
    - But pixels on that scanline get an interpolated quantity (intensity of N) that does correspond to the halfway z.

Problems with Gouraud & Phong Shading: Vertex normals at creases

- Crease edges should not have smooth shading.
- Use multiple vertices, each with a different vertex normal.
- Example--cylinder in pieces:
  - The top is one piece (one set of triangles).
  - The sides are another piece (one set of quadrilaterals).
- Thus, sharing vertices on creases does not work.

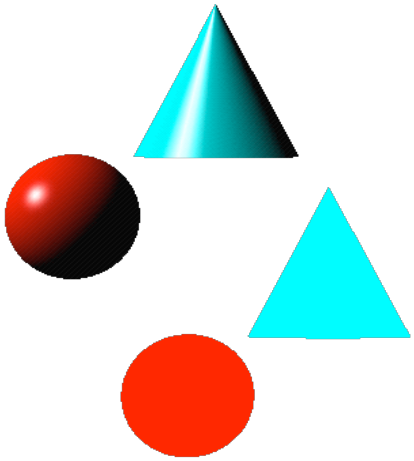


Rendering Surfaces

- Visible surface determination
  - Compute set of surfaces visible from the viewpoint
- Illumination and shading (local, direct illumination-models): Render depth, lighting effects, material properties to improve 3D perception.

Lighting Principles

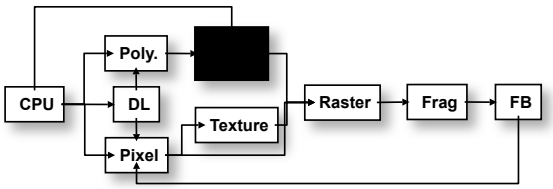
- Lighting simulates how objects reflect light
  - material composition of object
  - light's color and position
  - global lighting parameters
    - ambient light
    - two sided lighting
  - available in both color index and RGBA mode



How OpenGL Simulates Lights

- Phong lighting model
  - Computed at vertices
- Lighting contributors
  - Surface material properties
  - Light properties
  - Lighting model properties

Surface Normals



- Normals define how a surface reflects light

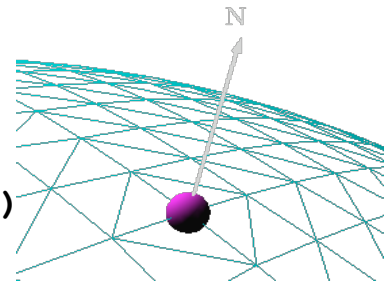
```
glNormal3f( x, y, z )
```

  - Current normal is used to compute vertex's color
  - Use *unit* normals for proper lighting
    - scaling affects a normal's length

```
glEnable( GL_NORMALIZE )
```

or

```
glEnable( GL_RESCALE_NORMAL )
```



Material Properties

- Define the surface properties of a primitive

```
glMaterialfv( face, property, value );
```

  - separate materials for front and back

Light Properties

```
glLightfv( light, property, value );
```

- ***light*** specifies which light
  - multiple lights, starting with **GL_LIGHT0**

```
glGetIntegerv( GL_MAX_LIGHTS, &n );
```

- ***properties***
  - colors
  - position and type
  - attenuation

Light Sources (cont.)

- Light color properties
  - **GL_AMBIENT**
  - **GL_DIFFUSE**
  - **GL_SPECULAR**

Types of Lights

- OpenGL supports two types of Lights
  - Local (Point) light sources
  - Infinite (Directional) light sources
- Type of light controlled by w coordinate

Turning on the Lights

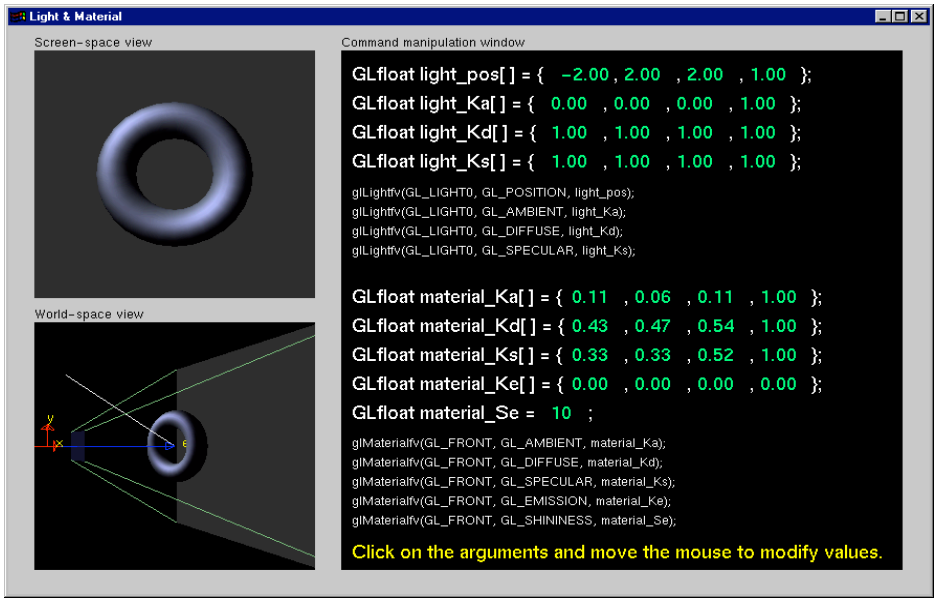
- Flip each light's switch

```
glEnable( GL_LIGHTn );
```

- Turn on the power

```
glEnable( GL_LIGHTING );
```

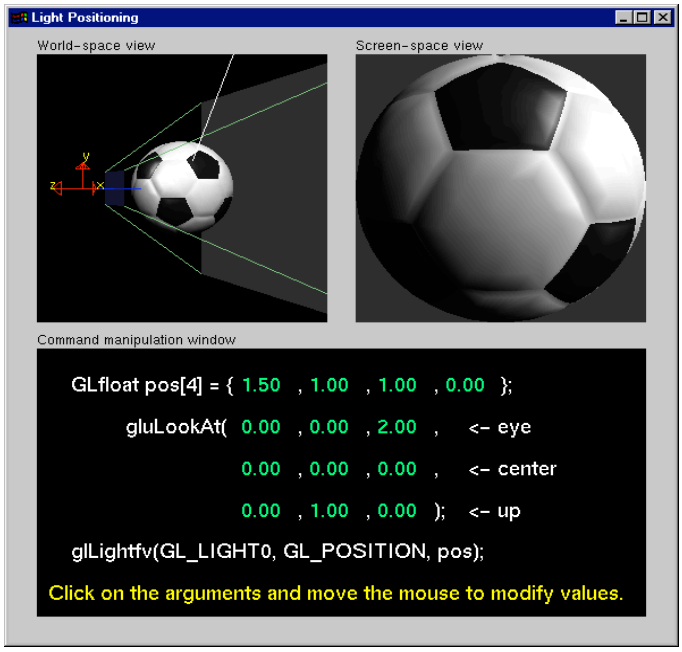

Light Material Tutorial



Controlling a Light's Position

- Modelview matrix affects a light's position
  - Different effects based on <u>when</u> position is specified
    - eye coordinates
    - world coordinates
    - model coordinates
  - Push and pop matrices to uniquely control a light's position

Light Position Tutorial



Advanced Lighting Features

- Spotlights
  - localize lighting affects
    - ***GL_SPOT_DIRECTION***
    - ***GL_SPOT_CUTOFF***
    - ***GL_SPOT_EXPONENT***

Advanced Lighting Features

- Light attenuation
  - decrease light intensity with distance
    - ***GL_CONSTANT_ATTENUATION***
    - ***GL_LINEAR_ATTENUATION***
    - ***GL_QUADRATIC_ATTENUATION***

Light Model Properties

```
glLightModelfv( property, value );
```

- Enabling two sided lighting
  `GL_LIGHT_MODEL_TWO_SIDE`
- Global ambient color
  `GL_LIGHT_MODEL_AMBIENT`
- Local viewer mode
  `GL_LIGHT_MODEL_LOCAL_VIEWER`
- Separate specular color
  `GL_LIGHT_MODEL_COLOR_CONTROL`

Tips for Better Lighting

- Recall lighting computed only at vertices
  - model tessellation heavily affects lighting results
    - better results but more geometry to process
- Use a single infinite light for fastest lighting
  - minimal computation per vertex

Other methods of improving realism

- Texture and bump (wrinkle) mapping
- Transparencies & color blendings
  - (Alpha Channel Blending)
- Light shadowing and shadow polygons
- Special reflections, refractions, transparencies
  - Often used in the movie industry to “fake” highly reflective surfaces such as glasses
- Modeling curved surfaces, physics-based models, fractals

Universität Bielefeld

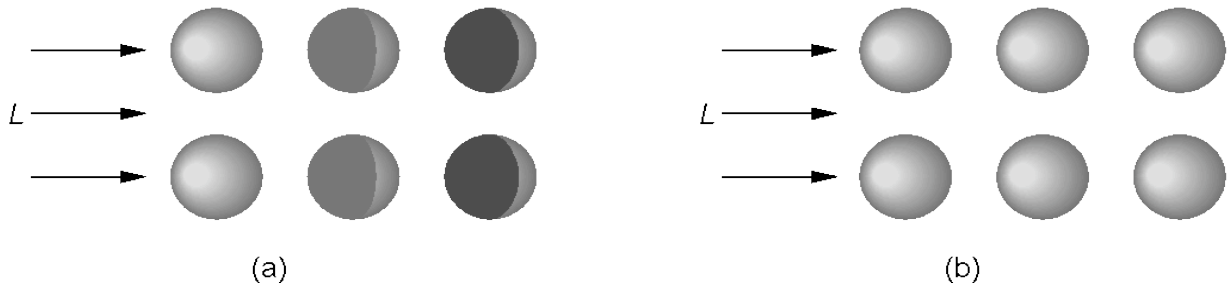
Shading and Color

Global Models

Global Models

- Improvements normally mean increasing computation
- Illumination/Shading models covered so far are “local models”
  - Only deal with light sources and single surface
  - “Global models” include light reflecting of other surfaces and shadows
  - Two major approaches: raytracing and radiosity

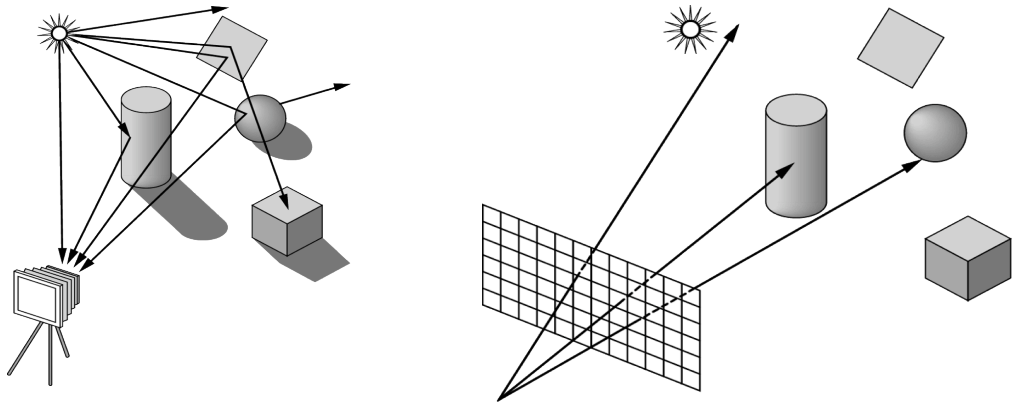
Comparison of Local vs Global Models



(a) (b)

Raytracing

- Follows light rays throughout scene
  - Restrict to following light rays that reach the eye

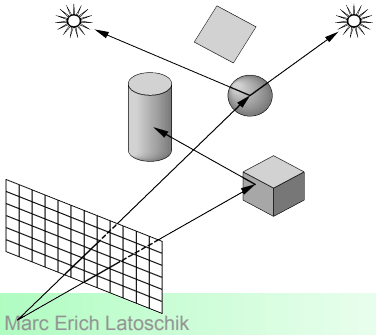


Raytracing

- Cast a ray from a pixel until:
  - Goes to infinity – assign pixel background color
  - Hits a light – assign pixel light color

Raytracing Continued

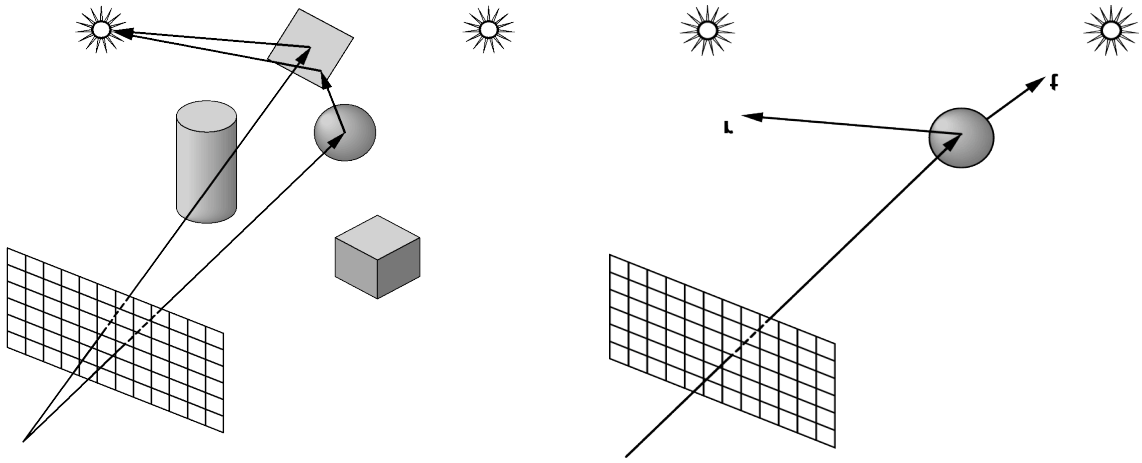
- Cast Ray Hits a surface –
  - Determine if surface is illuminated:
    - Compute shadow or feeler rays from surface to light sources
    - If feeler ray hits another surface first, then light source is blocked (in shadow)



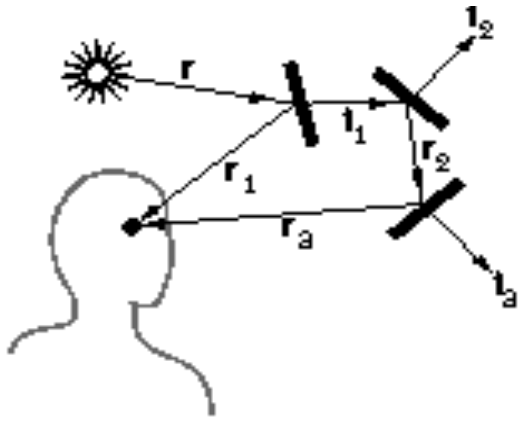
Raytracing Continued

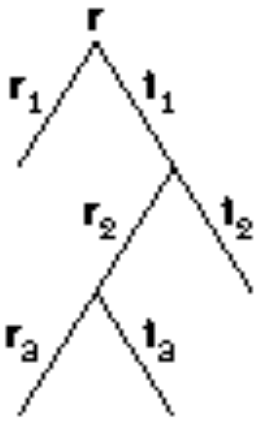
- Feeler or Shadow rays can also be used to add light reflected or transmitted from other surfaces
- At each intersection of a surface:
  - Determine the illumination of that surface point (through a recursive application of raytracing)
  - Determine how much of that illumination is transmitted via specular (reflective) or transparent effects along the original feeler ray.
    - Diffuse effects are ignored

Raytracing Examples



Raytracing Environments





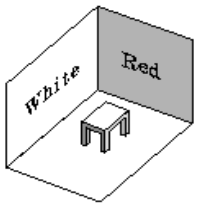
Vertex by Vertex vs Pixel by Pixel

- Different from our previous approach
  - Previous approach is vertex by vertex
  - This is pixel by pixel
  - Raytracing also includes hidden surface removal

Radiosity

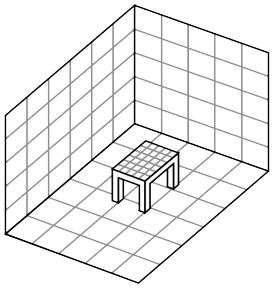
- Whereas Raytracing is very good for specular/ transparent environments,
- Radiosity is very good for diffuse environments
- Radiosity uses a “global energy balance” approach

Consider



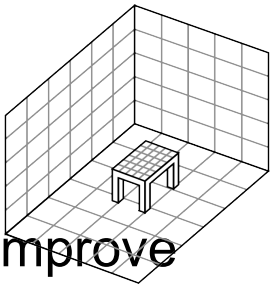
- In the real world, diffuse surfaces impact the color of each other:
  - A red wall next to a white wall:
    - The red wall will be lighten by diffuse light from the white wall
    - The white wall will have a red tint from diffuse light from the red wall
  - These are diffuse-diffuse interactions
  - Not taken into account in either local models or raytracing

Radiosity



- Radiosity is a numerical method for approximating diffuse-diffuse interaction
- Basic Approach:
  - Break scene up into small flat polygons (patches) each which are perfectly diffuse and constant shade
  - Consider the patches pairwise to determine their light interaction (form factors)
  - For each patch:
    - Determine the color by calculating the light energy from all form factors that include this patch
  - Once the patch colors are determined, render using a flat shading model

Radiosity



- Each round of patch calculation can improve the realism of the image
  - For example, imaging a three walls of three different color – R, G, B
  - The 1st round calculate patches:
    - RG+RB, GR+GB, BR+BG
  - The 2nd round improves this by calculating the interactions of these mixed color patches
    - (RG+RB)(GR+GB)+(RG+RB)(BR+BG), …

Radiosity Calculation Costs

- Each round is an $O(n^2)$ for n patches
- Most of the time, only one round is calculated
- One major advantage of using Radiosity*:
  - Since there is no specular or transparent lighting effects, lighting is **not** viewer dependent!
  - This means that one can walk through a radiosity-rendered scene!
  - Most often used for architectural renderings and walkthroughs

*Although normally used with a flat shading model, it is possible to use gouraud or phong shading

Summary

- Illumination Models
- Local Models
- Global Models
- Key: All techniques “fake” reality

Appearance

- **N** is the unit normal at point p.
- **L** is the unit vector pointing to the light source.
- θ 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.

Appearance

- **Phong lighting model:**

- **N** is the unit normal at point p.
- **L** is the unit vector pointing to the light source.
- θ 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.

Appearance

- **Phong lighting model:**

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

- **N** is the unit normal at point p.
- **L** is the unit vector pointing to the light source.
- θ 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.

Appearance

- **N** is the unit normal at point p.
- **L** is the unit vector pointing to the light source.
- θ 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.

- **Phong lighting model:**

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

- $I_{a\lambda}$ is channel intensity of current light source for channel λ.

Appearance

- **N** is the unit normal at point p.
- **L** is the unit vector pointing to the light source.
- θ 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.

- **Phong lighting model:**

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

- $I_{a\lambda}$ is channel intensity of current light source for channel λ.
- k_a is a reflection type dependent factor (ambient, diffuse, specular).

Appearance

- **N** is the unit normal at point p.
- **L** is the unit vector pointing to the light source.
- θ 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.

- **Phong lighting model:**

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

- $I_{a\lambda}$ is channel intensity of current light source for channel λ.
- k_a is a reflection type dependent factor (ambient, diffuse, specular).
- $O_{d\lambda}$ is a material dependent reflection factor for that channel.

Appearance

- **N** is the unit normal at point p.
- **L** is the unit vector pointing to the light source.
- θ 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.

- **Phong lighting model:**

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

- $I_{a\lambda}$ is channel intensity of current light source for channel λ.
- k_a is a reflection type dependent factor (ambient, diffuse, specular).
- $O_{d\lambda}$ is a material dependent reflection factor for that channel.
- f_{att} is Light source attenuation.

Appearance

- **N** is the unit normal at point p.
- **L** is the unit vector pointing to the light source.
- θ 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.

- **Phong lighting model:**

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

- $I_{a\lambda}$ is channel intensity of current light source for channel λ.
- k_a is a reflection type dependent factor (ambient, diffuse, specular).
- $O_{d\lambda}$ is a material dependent reflection factor for that channel.
- f_{att} is Light source attenuation.
- Atmospheric attenuation effect (OpenGL):

Appearance

- **N** is the unit normal at point p.
- **L** is the unit vector pointing to the light source.
- θ 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.

- **Phong lighting model:**

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

- $I_{a\lambda}$ is channel intensity of current light source for channel λ.
- k_a is a reflection type dependent factor (ambient, diffuse, specular).
- $O_{d\lambda}$ is a material dependent reflection factor for that channel.
- f_{att} is Light source attenuation.
- Atmospheric attenuation effect (OpenGL):

$I_\lambda' = sI_\lambda + (1 - s)\, I_{dc\lambda}$ where $0<s<1$ for objects between near/far planes.

Realtime 3D Computer Graphics / Virtual Reality – WS 2006/2007 – Marc Erich Latoschik