Realtime 3D Computer Graphics & Virtual Reality

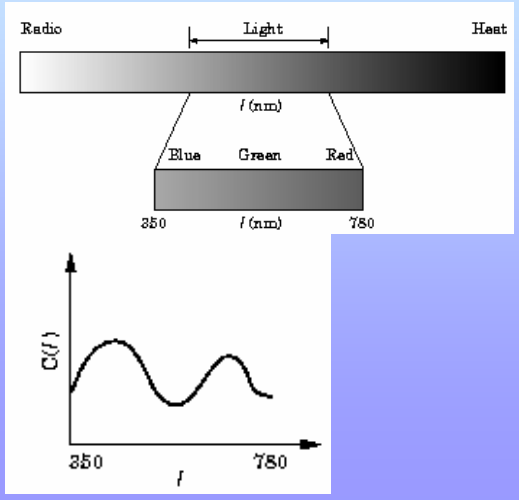
Lighting, Shading and Color

Introduction to Lighting and Shading

- Illumination models express how light affects a surface's color at a given point.
- Shading models use illumination models to determine the color across a surface.
  - Apply the illumination model at some points.
  - Interpolate the illumination at other points.
- This process is sometimes called "lighting the object".

What Creates Colors?

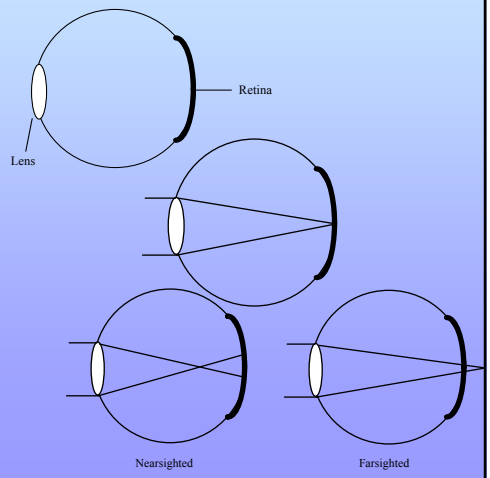
- Interaction between Light, Objects, Eyes
- What is Light?
  - Electromagnetic Radiation of a Specific Spectrum Range
- Light is a distribution C(I) of intensities I at each wavelength



Gamma

Vision: The Eye

- The eye can be viewed as a dynamic, biological camera: it has a lens, a focal length, and an equivalent of film.
- The lens must focus directly on the retina for perfect vision
- But age, malnutrition and disease can unfocus the eye, leading to near- and farsightedness

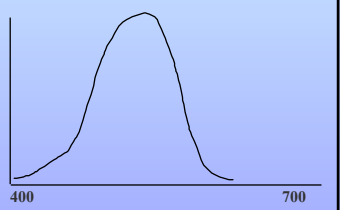


Vision: The Retina

- The retina functions as the eye's "film".
- It is covered with cells sensitive to light. These cells turn the light into electrochemical impulses that are sent to the brain.
- There are two types of cells, *rods* and *cones*

Vision: Rods

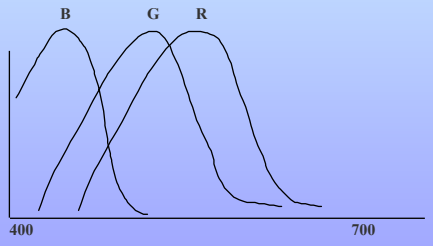
- Sensitive to most visible frequencies (brightness).
- About 120 million in eye.
- Most located outside of *fovea*, or center of retina.
- Used in low light (theaters, night) environments, result in achromatic (b&w) vision.



rod absorption function

Vision: Cones

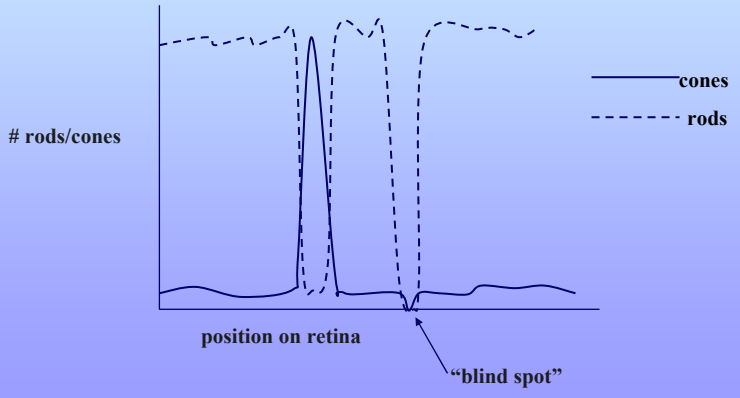
- The absorption functions of the cones are:



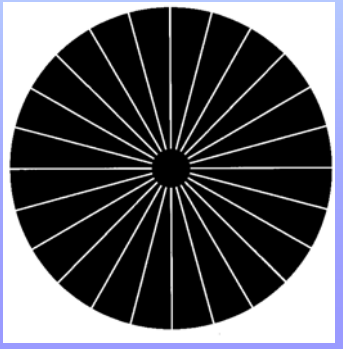
Vision: Cones

- R cones are sensitive to long wavelengths (λ), G to middle λ's, and B to short λ's.
- About 8 million in eye.
- Highly concentrated in fovea, with B cones more evenly distributed than the others.
- Used for high detail color vision, so they will concern us most.

Rod/Cone distribution



Blind spot examples

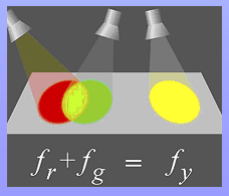


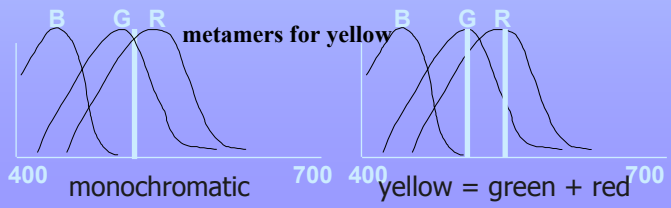
Vision: Color Constancy

- If color is just light of a certain wavelength, why does a yellow object always look yellow under different lighting (e.g. interior/exterior)?
- This is the phenomenon of *color constancy*.
- Colors are constant under different lighting because the brain responds to ratios between the R, G and B cones, and not magnitudes.

Vision: Metamers

- Because colors are represented to the brain as ratios of three signals, it is possible for different frequency combinations to appear as the same color.
- These combinations are called metamers. *This is why RGB color works!*





Vision: Sensitivity vs. Acuity

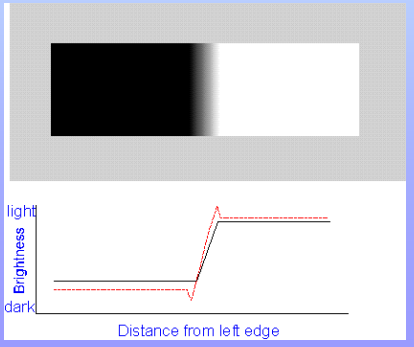
- *Sensitivity* is a measure of the dimmest light the eye can detect.
- *Acuity* is a measure of the smallest object the eye can see.
- These two capabilities are in competition:
  - In the fovea, cones are closely packed. Acuity at its highest, sensitivity at its lowest.
  - Outside the fovea, acuity decreases rapidly. Sensitivity increases correspondingly.

Vision: Center/Surround

- The light striking rods and cones in the retina is not summed uniformly. Instead, the nerves that combine the signals from the rods or cones sum with a center/surround opponency:

Vision: Center/Surround

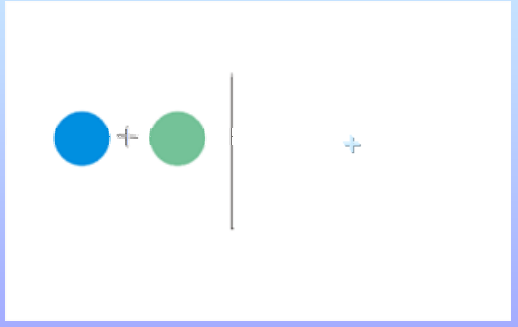
- This explains Mach banding, and another well-known optical illusion:



Mach Banding

- Flat shading of more facets does not necessarily look smoother.
- Imaginary dark and light lines appear at facet boundaries.
- These lines appear at any discontinuity or drastic change in the rate of shading.
- These lines are perceptual artifacts called Mach bands.
- Mach bands are caused by the eye's lateral inhibition.
  - When one receptor responds to a high intensity, it inhibits its neighboring receptors' responses.
  - Receptors on the bright side of a discontinuity receive less inhibition from the dark side.
  - Receptors on the dark side of a discontinuity receive more inhibition from the light side.

Color afterimage example



Stare at the plus sign on the left for about 30 seconds. As you do this you probably will see some colors around the blue and green circles. After about 30 seconds, shift your gaze to the plus sign on the right. What did you see?

You probably saw a yellow and desaturated reddish circle.

Vision: Color Opponency

- After color is turned into three signals by the R, G and B cones, it is turned into three parallel signals:
  - Achromatic: R + G
  - Blue-yellow: R + G - B
  - Red-green: B + R - G
- This is why it is not possible to see reddish green or yellowish blue (as opposed to greenish blue).
- The blue/yellow and red/green pairs are called *complementary* colors. Mixing the proper shades of them in the proper amounts produces white light.

(can you now explain the afterimage effect of the previous slide?)

Vision: Beyond the Eye

- Beyond the eye, visual signals move through different processing stages in the brain.
- There seem to be two main pathways
  - *Magnocellular*: low-resolution, motion sensitive, and primarily achromatic pathway
  - *Parvocellular*: high-resolution, static, and primarily chromatic pathway
- Color vision is processed in three dimensions: *hue, saturation,* and *luminance*

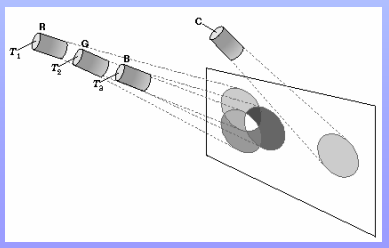
Color: Terminology

- *Hue*: the dominant wavelength of the light entering the eye
- *Saturation*: inversely related to the amount of white light in the light entering the eye (e.g. red, fully saturated; pink, not fully saturated)
- *Luminance*: the intensity of the light entering the eye (e.g. light with a dial)
  - *Lightness*: luminance from a reflecting object
  - *Brightness*: luminance from a light source
- *Chromaticity*: the hue and saturation of light (not luminance)

Representation of Light

- Do we need to represent all I_λ to represent a color C(I) ?
- No – we can approximate using a three-color additive system:

$$C = T_1 C_1 + T_2 C_2 + T_3 C_3$$



Color Models

- Used to describe color as accurately as possible
- Make use of the fact that colors can be described by combinations of three basic colors, called *primary colors*.
- An organization called the CIE (Commission International de l'Eclairage - International *Color* Commision) produced two models for defining color:
  - 1931: Measured on 10 subjects (!) on samples subtending 2 (!) degrees of the field of view
  - 1964: Measured on larger number of subjects subtending 10 degrees of field of view

CIE 1931 Model

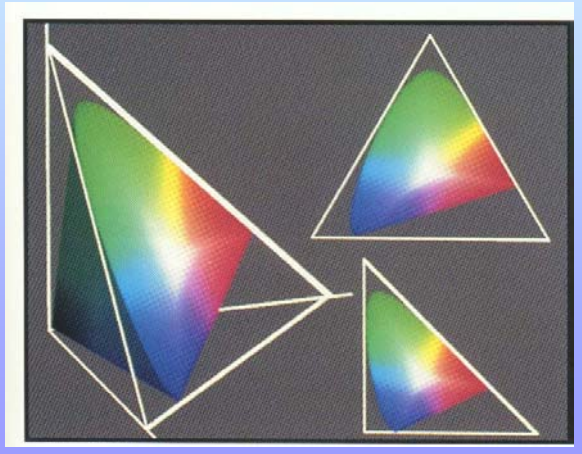
- The CIE 1931 model is the most commonly used
- It defines three primary "colors" X, Y and Z that can be used to describe all visible colors, as well as a standard white, called C.
- The range of colors that can be described by combinations of other colors is called a *color gamut*.
- Since it is impossible to find three colors with a gamut containing all visible colors, the CIE's three primary colors are imaginary. They cannot be seen, but they can be used to define other visible colors.

CIE 1931 Model

- To define a color in CIE model, provide weights for the X, Y and Z primaries, just as you would for an RGB display (e.g. color = xX + yY + zZ)
- X, Y and Z form a three dimensional color volume
- We can ignore the dimension of luminance by normalizing with total light intensity, x+y+z = 1. This gives chromaticity values:

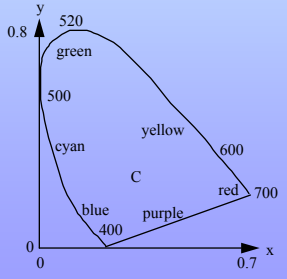
$$x' = x/(x+y+z)$$
$$y' = y/(x+y+z)$$
$$z' = 1 - x' - y'$$

CIE 1931 model

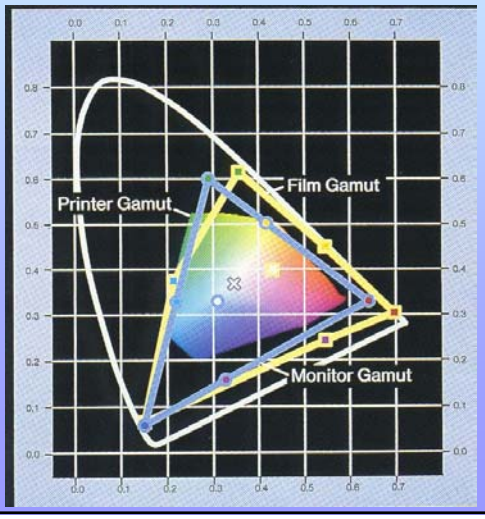


CIE 1931 Model

- Plotting x' and y' gives the CIE chromaticity diagram
- Color gamuts are found by taking the convex hull of the primary colors
- Complements are found by inscribing a line from the color through C to the edge of the diagram

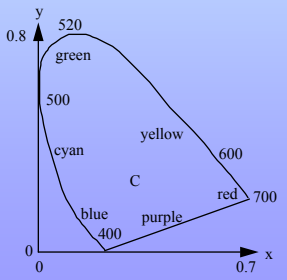


CIE 1931 model - gamuts



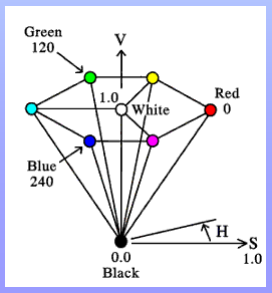
CIE 1931 Model

- The hue of a color can be found by inscribing a line from C (white) through the color to the edge of the diagram. The hue is the wavelength of the color at the intersection of the edge and the line.
- The saturation of a color can be found by taking the ratio of the distance of the color from C on the above line and the length of the whole line.



Psychological Models: HSV

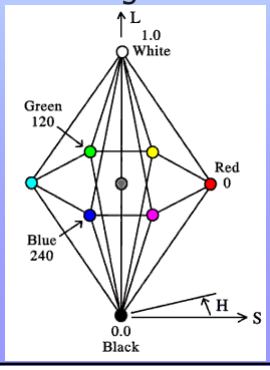
- The HSV (for hue, saturation, and value) color model equates value with luminance
- Hue varies between 0 and 360 degrees, saturation and value between 0 and 1



- The eye can see about 128 different hues, and about 130 different saturations. The number of values varies between 16 (blue) and 23 (yellow)
- What do hexagonal and triangular cross sections look like?

Psychological Models: HLS

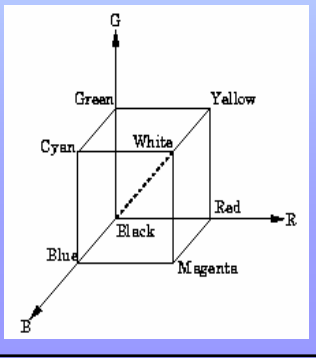
- The HLS (for hue, lightness, and saturation) color model equates lightness with luminance
- Hue varies between 0 and 360 degrees, saturation and lightness between 0 and 1



- What do circular and triangular cross sections look like?

Engineering Models: RGB

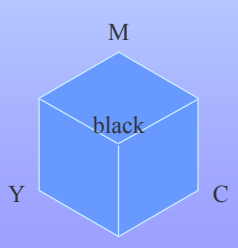
- The RGB stands for red, green and blue
- All parameters vary between 0 and 1



- Hue is defined by the one or two largest parameters
- Saturation can be controlled by varying the collective minimum value of R, G and B
- Luminance can be controlled by varying magnitudes while keeping ratios constant

Engineering Models: CMY

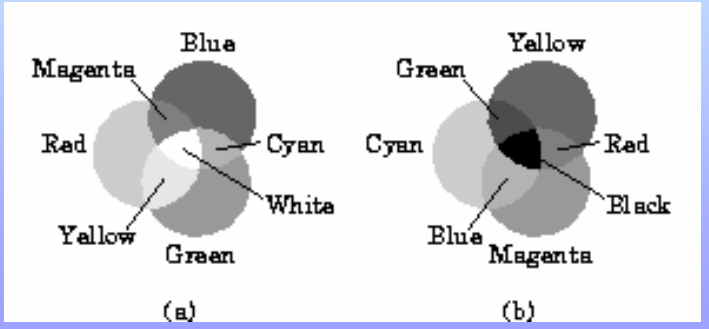
- The CMY stands for cyan (blue + green), magenta (red + blue), and yellow (red + green)
- All parameters vary between 0 and 1



- [C M Y] = [1 1 1] - [R G B]
- Used for reflective media, like printing
- Since reflective media absorb light, this model does also. It is *subtractive*. Thus red is M + Y: M subtracts green, Y subtracts blue

Subtractive Color Systems

- Print systems use subtractive color system:



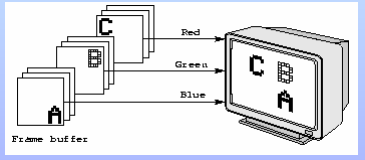
RGB CMY

Eng. Models: CMYK and YIQ

- CMYK is a variant of CMY that enhances darkness by setting K = min(C,M,Y), and printing black with intensity K.
- YIQ is the model used for broadcast television. This model can be encoded by black and white or color TV sets. Y contains the luminance values and is the only component used in a black and white TV.

Color Displays

- Frames can be displayed using RGB system



- Or with a color palette:

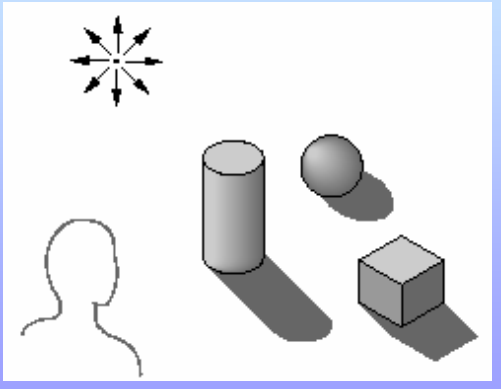
Input	Red	Green	Blue
0	0	0	0
1	2^m 2 1	0	0
.	0	2^m 2 1	0
.	.	.	.
.	.	.	.
2^k 2 1	.	.	.
	m bits	*m* bits	*m* bits

LIGHTING

- Light Sources
- Reflection

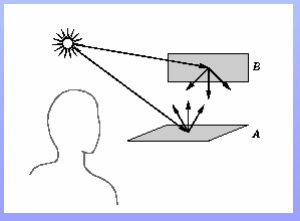
Light Sources

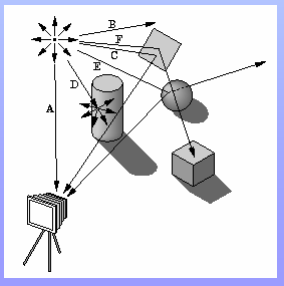
- Point Sources
- Area Sources
- Distant Source

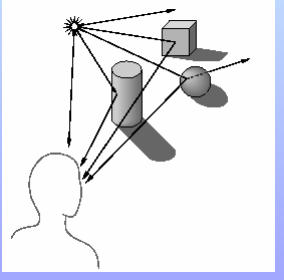


Light and Matter

- Reflection from Matter Surfaces
  - Surface Properties
- Multiple Reflections

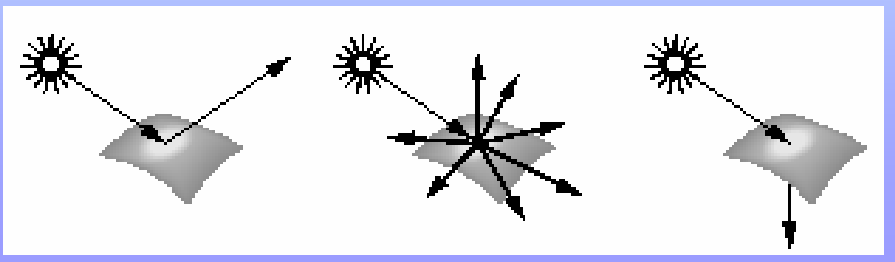






Interaction of Light and Materials

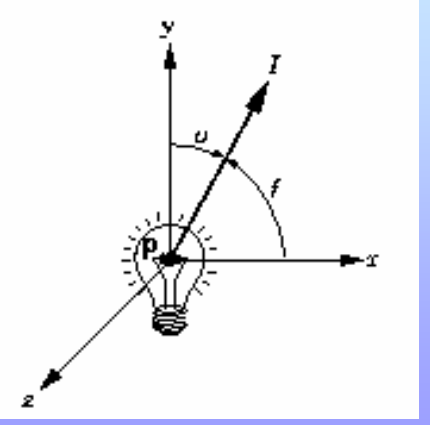
- Three Types of Interactions:
  - Specular – Reflectivity in a single direction
  - Diffuse – Reflectivity in all directions (scattering)
  - Translucent – Passing of light through a material



Light Sources

- Location (*x,y,z*)
- Direction (θ, ϕ)
- Color (λ)
- Intensity (I)

$$I(x, y, z, \theta, \phi, \lambda)$$



Intensity Functions – RGB

$$I = \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix}$$

Light Source Types

- Ambient Light
- Point Light Sources
- Spot Lights
- Distant Light Sources

Ambient Light

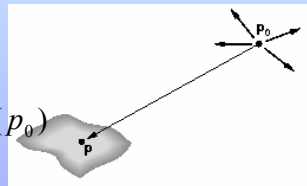
- Near uniform lighting created by highly diffused light sources
- One could model all light sources and interactions or use a concept called "ambient light" which lights all surfaces uniformly.
- Not viewer location dependent

$$I = \begin{bmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{bmatrix}$$

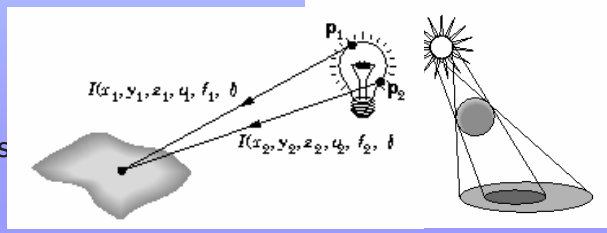
Point Light Sources

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

$$I(p_0) = \begin{bmatrix} I_r(p_0) \\ I_g(p_0) \\ I_b(p_0) \end{bmatrix} \text{ and } I(p, p_0) = \frac{1}{|p - p_0|^2} I(p_0)$$

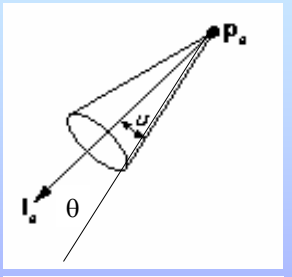


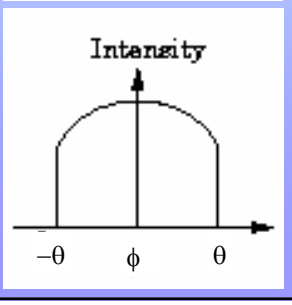
Use of point sources is more a matter of efficiency rather than realism as most sources have a dimension:



Spotlights

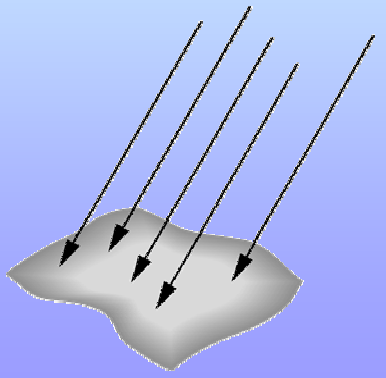
- Point source with limited direction
- Point source Ps in a direction I_s and a width of θ
- Spotlight attenuation:
  - Greater realism can be obtained by varying the intensity of light across the cone
  - Typical Function: $\cos(\phi) = \mathbf{S} \cdot \mathbf{I}$





Distant Light Sources

- 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

Illumination Models

- 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.

Illumination At A Point

- Total illumination: Sum of each light source

$$I_\lambda = I_{\lambda 1} + I_{\lambda 2} + I_{\lambda 3}$$

- Overflow is possible (>1)
  - There are various solutions for dealing with it.
    - One solution is to clamp to max allowable
    - Normalize individual terms:

$$I_\lambda = \frac{I_{\lambda 1}}{(I_{\lambda 1} + I_{\lambda 2} + I_{\lambda 3})} + \frac{I_{\lambda 2}}{(I_{\lambda 1} + I_{\lambda 2} + I_{\lambda 3})} + \frac{I_{\lambda 3}}{(I_{\lambda 1} + I_{\lambda 2} + I_{\lambda 3})}$$

Illumination at a point on an object

- For each color (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

Ambient light

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

Ambient Light Continued

- 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
- A resulting image:

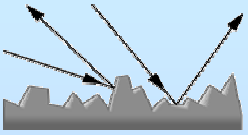
Ambient light in OpenGL

- 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 *ka* and *Odl* are combined.

Diffuse Reflection

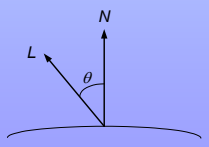


- Scattering of Reflective Light
- Consider how a dull, matte surface (e.g., chalk) scatters light.
  - When its orientation is fixed relative to a light, its illumination looks the same from all viewing angles.
  - When its orientation changes relative to a light, its illumination changes.
    - It 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

- This reflection is 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 :

  $I_{p\lambda}\ \cos(\theta)$ or $I_{p\lambda}\ (\mathbf{N} \cdot \mathbf{L})$,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 $\theta > 90$ degrees



Lambertian Reflection Continued

- 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:

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

Lambertian Reflection 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.

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:

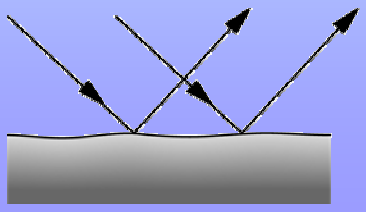
```
float direction0[] = {dx,dy,dz,0};
glLightfv(GL_LIGHT0, GL_POSITION, direction0);
```

  - 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);
```

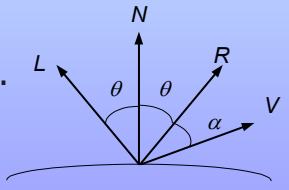
Specular Reflection

- 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.



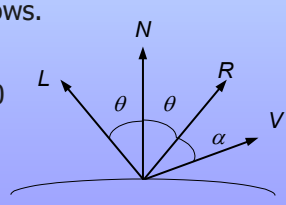
Specular Reflection

- 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**.



Specular Reflection

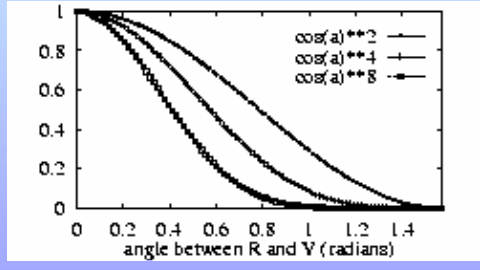
- The highlight's visible intensity depends on:
  - α
    - The highlight is most intense when $\alpha = 0$
    - The highlight becomes dimmer as α grows.
  - material properties
    - Example: Mirror reflects only with $\alpha = 0$
    - A Mirror is a Perfect Reflector



- Specular reflection is viewer location dependent

Phong Model for Non-Perfect Reflectors

- A light of intensity $I_{p\lambda}$ produces a highlight intensity proportional to
  $I_{a\lambda}$ (cos (α)) ^{n}.



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

Specular Reflection

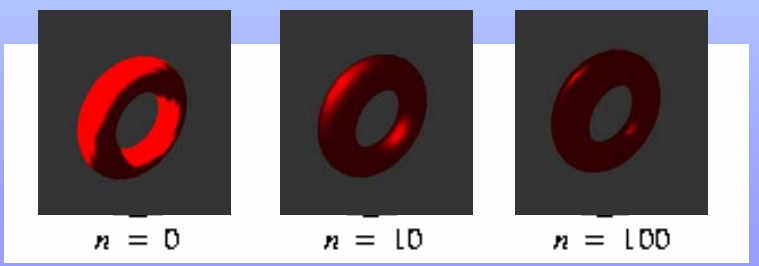
- Other material properties affect the intensity specularly reflected.
  - The overall fraction of light reflected is $W(\theta)$, often taken to be the constant k_s
    - (Ranges 0 to 1)
  - The fraction of primary λ reflected is $O_{s\lambda}$
- The specular intensity is thus:

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

Specular Reflection

- The illumination model at a point is thus so far:

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



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:

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

  - 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: Attenutation 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 : $I_\lambda' = sI_\lambda + (1 - s)\, I_{dc\lambda}$
  - Where $s = 0$ for objects in front of front plane,
    s = 1, for objects behind back plane,
    s linearly increasing between front and back planes
- FOG is OpenGL's implementation of atmospheric attenuation

Modified Illuminiation Model

- A complete model:

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

$$I_\lambda' = sI_\lambda + (1 - s)\, I_{dc\lambda}$$

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

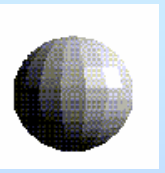
"Local Model" Shading Models

- 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

Shading Models

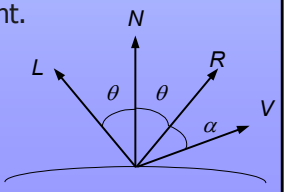
- Ideally, the renderer should apply the illumination model at every visible point on each surface
- This approach requires too much computation.
- As an alternative:
  - Apply the illumination model at a subset of points.
  - Interpolate the intensity of the other points.
  - Apply the illumination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

Flat (Constant) Shading



- Sample illumination at one point per polygon.
- Use constant interpolation:

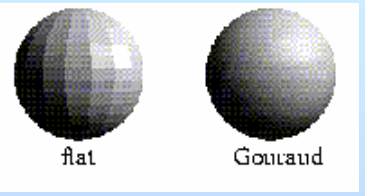
  all other points on the polygon get that point's intensity.
- This approach would be valid if:
  - The true surface really is faceted, so **N** is constant.
  - The light source is at infinity, so **L** is constant.
  - The viewer is at infinity, so **V** is constant.



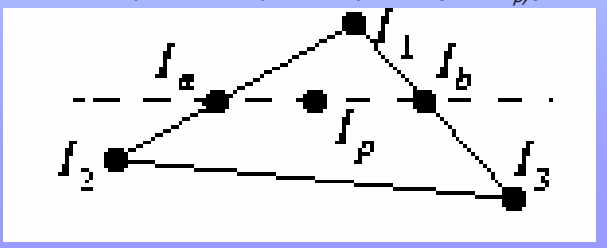
Flat shading in OpenGL

- 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

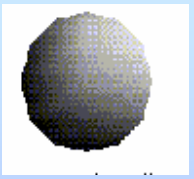
Phong Shading

- 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.
- OpenGL does not support Phong shading!

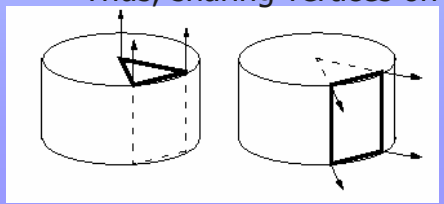
Problems Common to Gouraud and Phong Shading

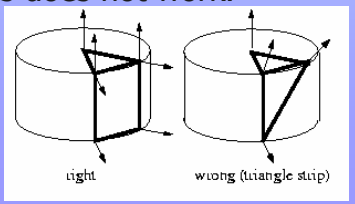


- Silhouette edges are not smoothed
- Normal vector interpolation can cause problems:
  - Interpolation may mask regular changes
  - Interpolation ignores perspective distortion
    - Due to foreshortening, a change in scanlines does not correspond to a constant change in z in OpenGL eye coordinates.
    - 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

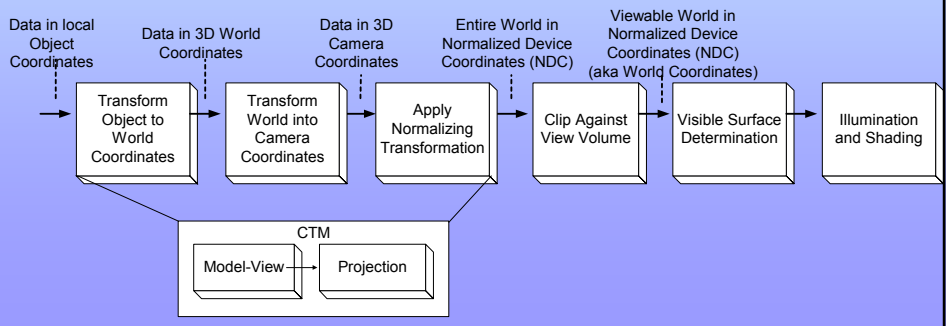
- Crease edges should not have smooth shading.
- Use multiple vertices, each with a different vertex normal.
- Example--cylinder in pieces:
  - The top is one piece (one set of triangles).
  - The sides are another piece (one set of quadrilaterals).
- Thus, sharing vertices on creases does not work.





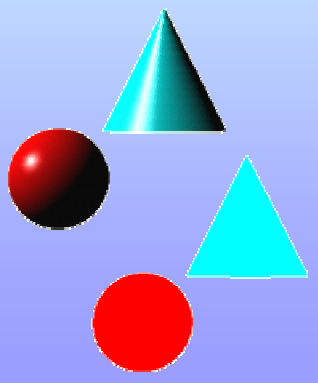
Rendering Surfaces

- 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

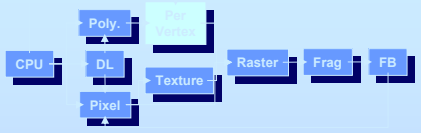
- Lighting simulates how objects reflect light
  - material composition of object
  - light's color and position
  - global lighting parameters
    - ambient light
    - two sided lighting
  - available in both color index and RGBA mode



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 );
```

`GL_DIFFUSE`	Base color
`GL_SPECULAR`	Highlight Color
`GL_AMBIENT`	Low-light Color
`GL_EMISSION`	Glow Color
`GL_SHININESS`	Surface Smoothness

  - 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

$w = 0$ Infinite Light directed along $(x \quad y \quad z)$

$w \neq 0$ Local Light positioned at $(x/w \quad y/w \quad z/w)$

Turning on the Lights

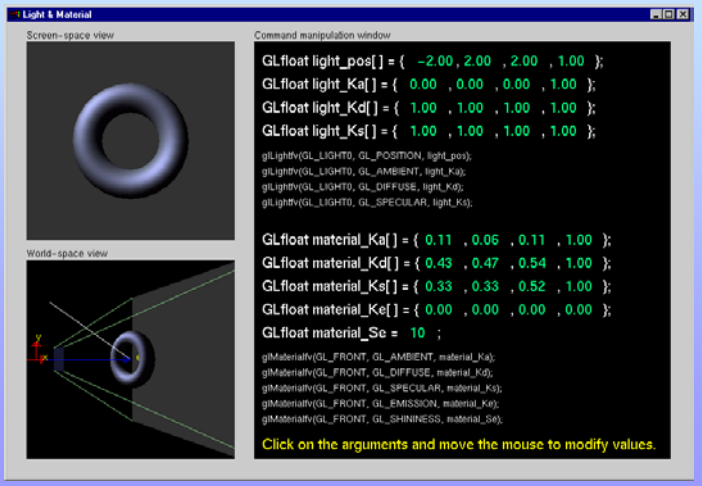
- Flip each light's switch

```
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 when 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***

$$f_i = \frac{1}{k_c + k_l d + k_q d^2}$$

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

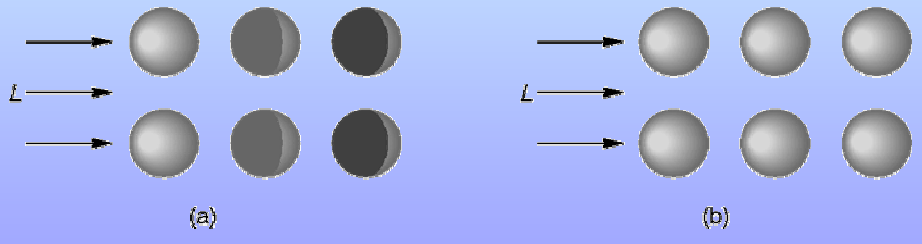
Shading and Color

Global Models

Global Models

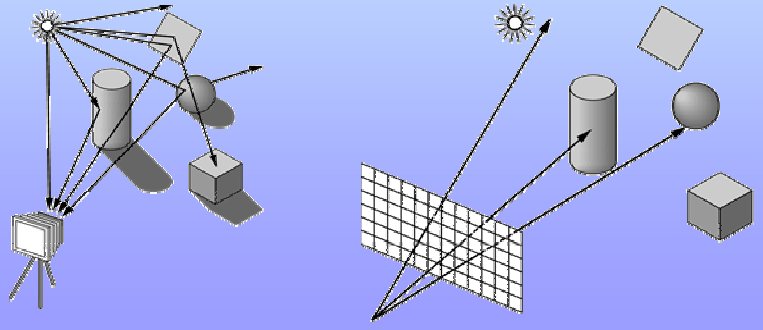
- Improvements normally mean increasing computation
- Illumination/Shading models covered so far are "local models"
  - Only deal with light sources and single surface
  - "Global models" include light reflecting of other surfaces and shadows
  - Two major approaches: raytracing and radiosity

Comparison of Local vs Global Models



Raytracing

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

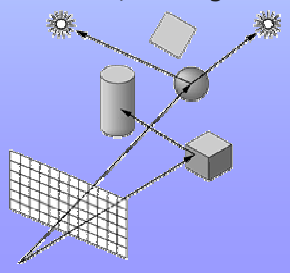


Raytracing

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

Raytracing Continued

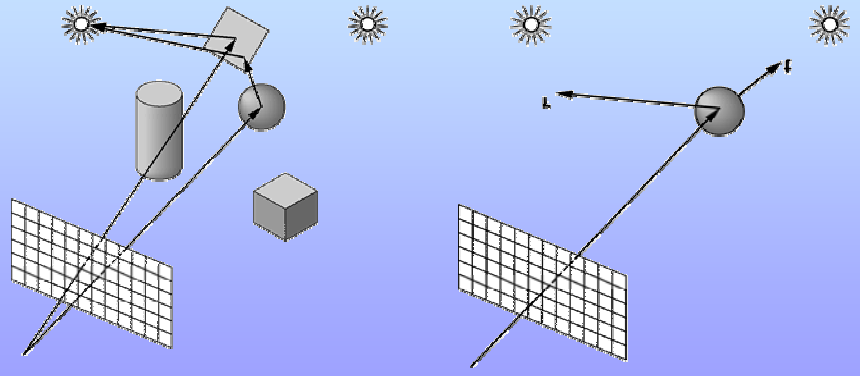
- Cast Ray Hits a surface –
  - Determine if surface is illuminated:
    - Compute shadow or feeler rays from surface to light sources
    - If feeler ray hits another surface first, then light source is blocked (in shadow)



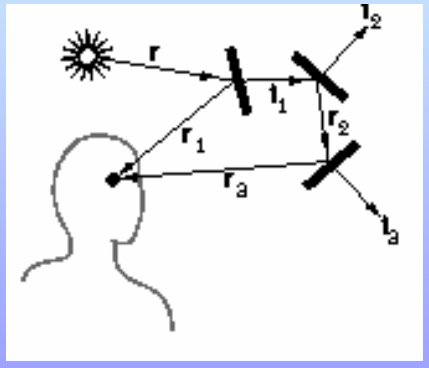
Raytracing Continued

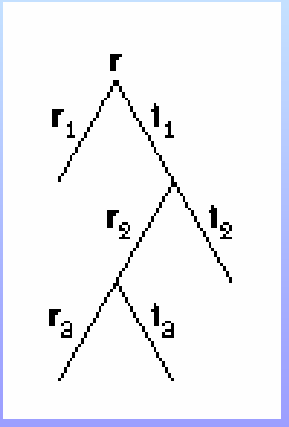
- Feeler or Shadow rays can also be used to add light reflected or transmitted from other surfaces
- At each intersection of a surface:
  - Determine the illumination of that surface point (through a recursive application of raytracing)
  - Determine how much of that illumination is transmitted via specular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

Raytracing Examples



Raytracing Environments





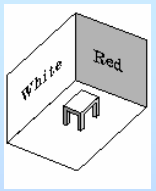
Vertex by Vertex vs Pixel by Pixel

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

Radiosity

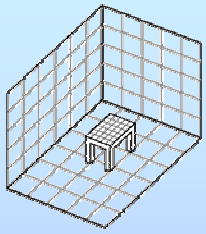
- Whereas Raytracing is very good for specular/transparent environments,
- Radiosity is very good for diffuse environments
- Radiosity uses a “global energy balance” approach

Consider



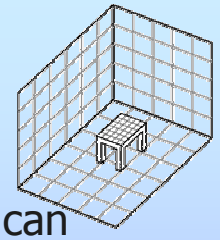
- In the real world, diffuse surfaces impact the color of each other:
  - A red wall next to a white wall:
    - The red wall will be lighten by diffuse light from the white wall
    - The white wall will have a red tint from diffuse light from the red wall
  - These are diffuse-diffuse interactions
  - 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