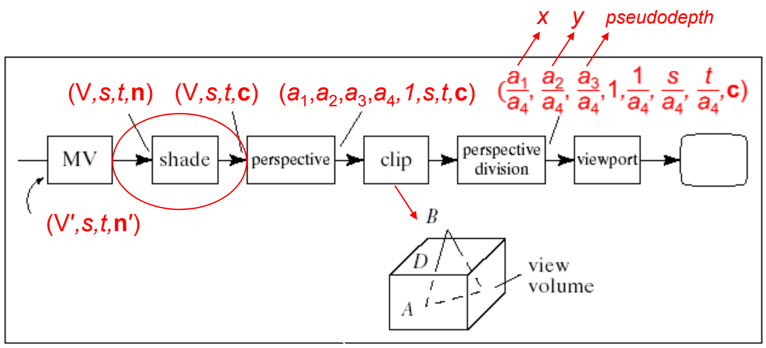
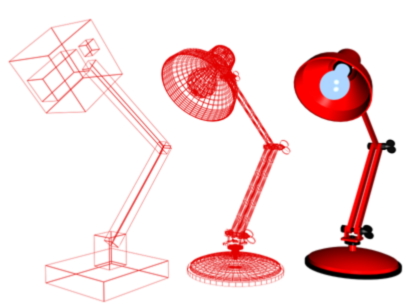
Reminder: The Pipeline



Rendering Styles

- ***Blocked, wireframe, & shaded renderings***

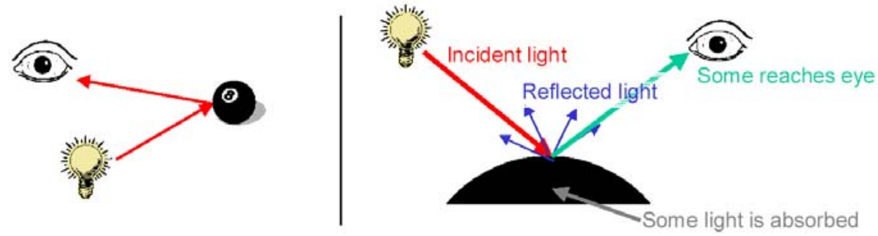


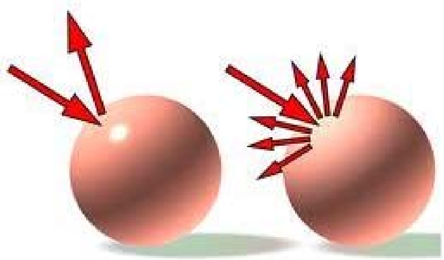
Object Appearance

Light transport in a scene

- Light is emitted from light sources
- Light interacts with surfaces
  - *On impact with an object, some light is reflected and some is absorbed*
  - *Distribution of reflected light determines “finish” (matte, glossy, …)*
- Composition of light arriving at camera determines the appearance of the scene

Interaction of Light With a Surface at a Single Point





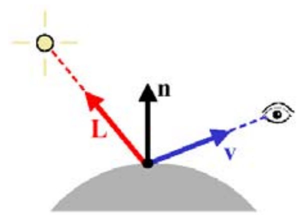
Modeling Light Sources

- Generally, light sources are complex
  - *The sun, light bulbs, fluorescent lights, monitors, …*
- Simple, point light sources
  - *The light source is a single infinitesimal point*
  - *Emits light equally in all directions (isotropic illumination)*
  - *Outgoing light is a set of rays originating at light source*

A Basic Local Illumination Model

We are interested only in the light that finally arrives at the view point

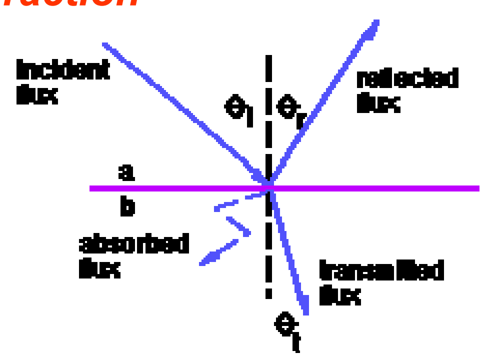
- This is a function of the light and viewing positions, and local surface reflectance



- We characterize light using RGB triples and operate on each channel separately (light superposition)

Local Illumination Physics

Law of reflection and Snell's law of refraction

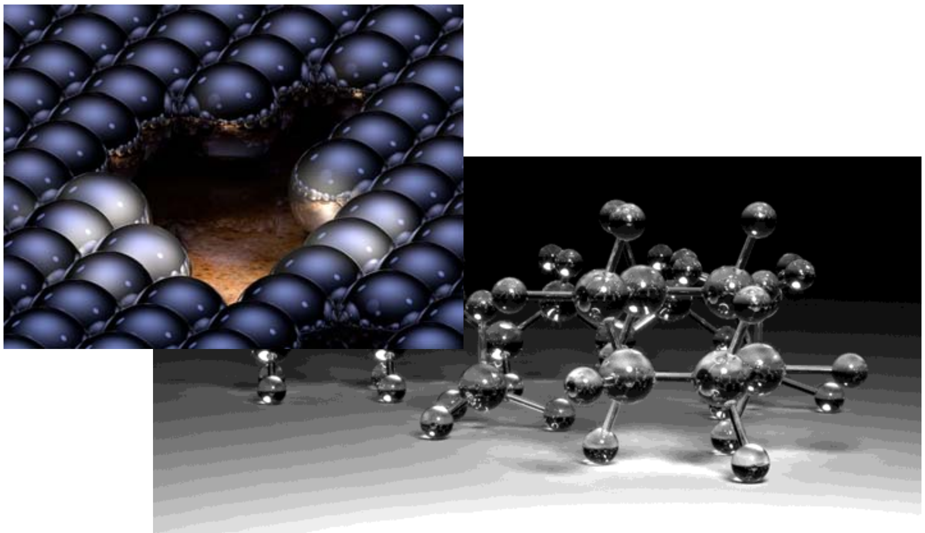


$$\theta_i = \theta_r$$

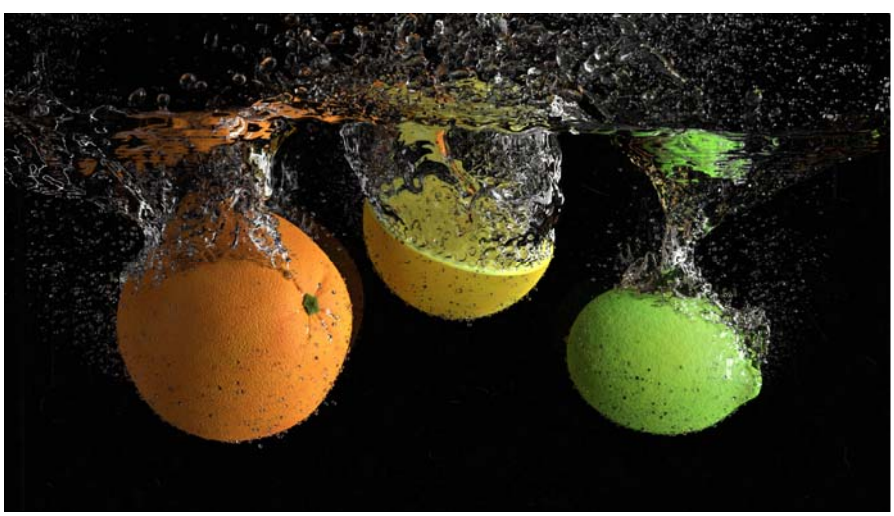
$$\frac{n_b}{n_a} = \frac{\sin(\theta_i)}{\sin(\theta_t)}$$

n: Refractive index

Reflection and Refraction (Ray Tracing Rendering)

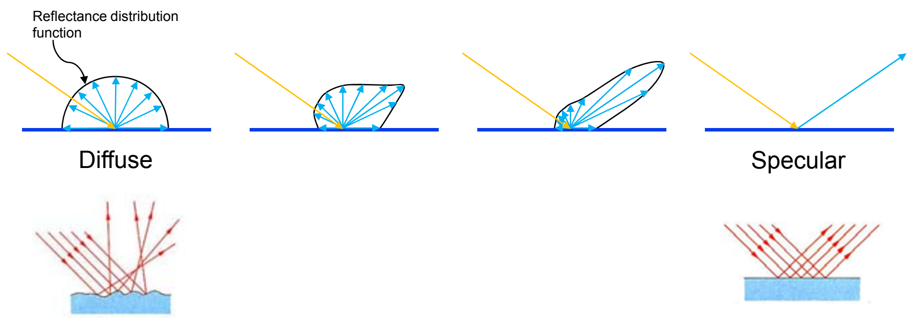


Refraction



What Are We Trying to Model ?

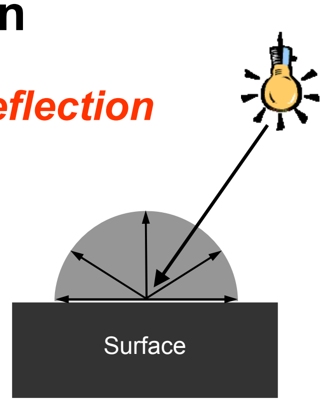
From diffuse to specular reflectance



Diffuse Reflection

This is the simplest kind of reflection

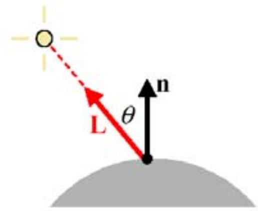
- Also called "Lambertian reflection" (Lambert's Law)
- Models dull, matte surfaces – materials like chalk
- Ideal diffuse reflection
  - *Scatters incoming light equally in all directions*
  - *Identical appearance from all viewing directions*
  - *Reflected intensity depends only on the direction of the light source*



Lambert's Law for Diffuse Reflection

Purely diffuse object

$$I = I_L k_d \cos\theta$$
$$= I_L k_d (\mathbf{n} \cdot \mathbf{L})$$



I : resulting intensity

I_L : light source intensity

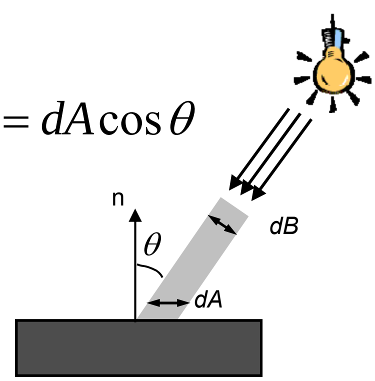
k_d : (diffuse) surface reflectance coefficient

$k_d \in [0,1]$

θ : angle between normal & light direction

Proof of Lambert's Cosine Law

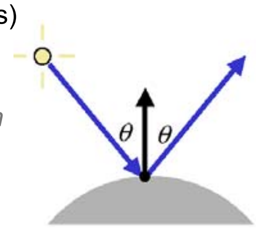
$$dB = dA\cos\theta$$



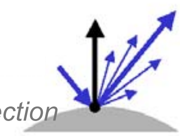
Specular Reflection

Shiny surfaces

- Their appearance changes as the viewpoint moves
- They have glossy “specular highlights” (specularities)
- A mirror is a perfect specular reflector
  - *Incoming light is reflected about normal direction*
  - *Nothing reflected in other directions*



- Most surfaces are imperfect specular reflectors
  - *Reflect light rays in cone about perfect reflection direction*

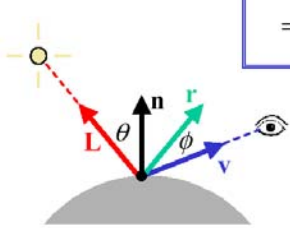


The Phong Model

A common specular reflection term is added

- It is purely empirical – there is no physical basis for it

$$\begin{aligned} I &= I_L k_d \cos\theta + I_L k_s \cos^n \phi \\ &= I_L k_d (\mathbf{n}\cdot\mathbf{L}) + I_L k_s (\mathbf{r}\cdot\mathbf{v})^n \end{aligned}$$



I : resulting intensity

I_L : light source intensity

k_s : (specular) surface reflectance coefficient

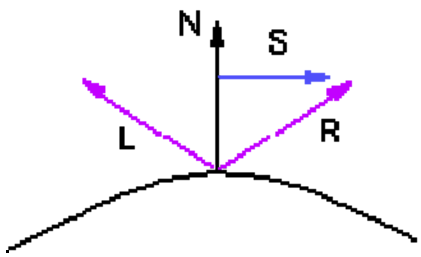
$k_s \in [0,1]$

ϕ : angle between viewing & reflection direction

n : "shininess" factor

Computing R

All vectors are unit length!

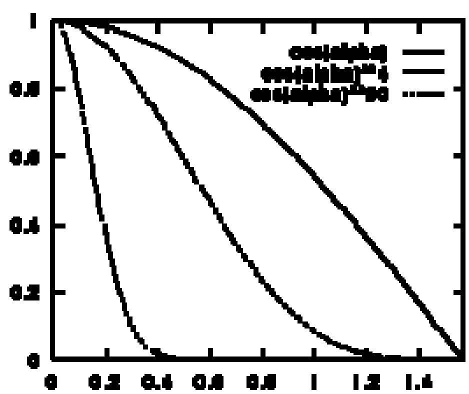


$$R = (N \bullet L)\ N + S$$

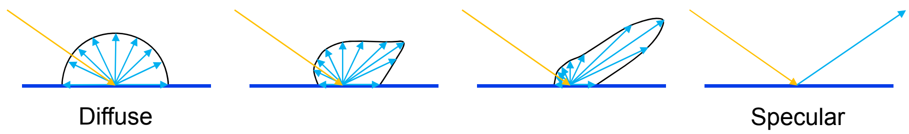
$$S = (N \bullet L)\ N - L$$

$$R = 2N\ (N \bullet L) - L$$

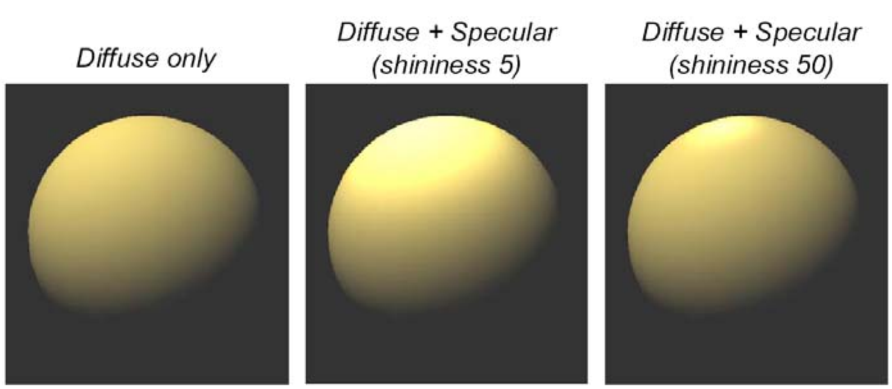
The Effect of the Exponent *n*



Comparison



Phong Model Examples



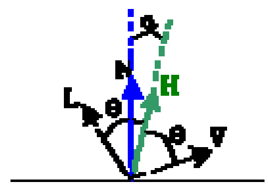
The Blinn-Torrance Specular Model

Agrees better with experimental results

$I_s = I_L\, k_s\, (H \cdot V)^n$

Halfway vector $H = \dfrac{L+V}{\|L+V\|}$

- Advantages
  - *Theoretical basis*
  - *N·H cannot be negative if N·L > 0 and N·V > 0*
  - *If the light is directional and we have orthographic projection then N·H is constant*



The Ambient Glow

So far, areas not directly illuminated by any light appear black

- this tends to look rather unnatural
- in the real world, there's lots of ambient light

To compensate, we invent new light source

- assume there is a constant ambient "glow"
- this ambient glow is *purely fictitious*

Just add in another term to our illumination equation

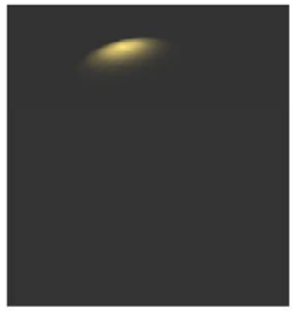
$$I = I_L k_d \cos\theta + I_L k_s \cos^n \phi + I_a k_a$$

I_a : ambient light intensity

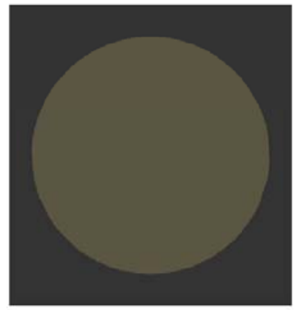
k_a : (ambient) surface reflectance coefficient

Our Three Basic Components of Illumination

Diffuse

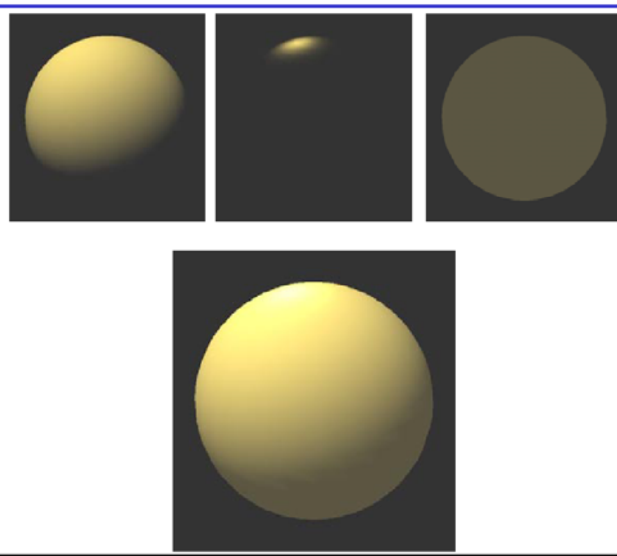


Specular



Ambient

Combined for the Final Result



Lights and Materials

Light properties

$I_{d(iffuse)}$, $\underbrace{I_{s(pecular)}}_{\text{Add Specular Light}}$, $I_{a(mbient)}$

Material properties:

$k_{d(iffuse)}$, $k_{s(pecular)}$, $k_{a(mbient)}$

$$I_r = I_{d\_r}k_{d\_r}(N\cdot L) + I_{s\_r}k_{s\_r}(R\cdot V)^n + I_{a\_r}k_{a\_r}$$

$$I_g = I_{d\_g}k_{d\_g}(N\cdot L) + I_{s\_g}k_{s\_g}(R\cdot V)^n + I_{a\_g}k_{a\_g}$$

$$I_b = I_{d\_b}k_{d\_b}(N\cdot L) + I_{s\_b}k_{s\_b}(R\cdot V)^n + I_{a\_b}k_{a\_b}$$

Questions

If you shine red light (1,0,0) on a diffuse white object (1,1,1) what color does the object appear to have?

What if you shine red light (1,0,0) on a diffuse green object (0,1,0) ?

If the object is shiny, what is the color of the highlight?

Special cases

$I_r = I_{d\_r} k_{d\_r} (N \cdot L) + I_{s\_r} k_{s\_r} (R \cdot V)^n + I_{a\_r} k_{a\_r}$

$I_g = I_{d\_g} k_{d\_g} (N \cdot L) + I_{s\_g} k_{s\_g} (R \cdot V)^n + I_{a\_g} k_{a\_g}$

$I_b = I_{d\_b} k_{d\_b} (N \cdot L) + I_{s\_b} k_{s\_b} (R \cdot V)^n + I_{a\_b} k_{a\_b}$

- What should be done if I >1?
  Clamp the value of I to 1
- What should be done if N·L < 0?
  Clamp the value of I to 0 or flip the normal
- How can we handle multiple light sources?
  Sum the intensity of the individual contributions

Shading Polygons: Flat Shading

Illumination equations are evaluated at surface locations

- so where do we apply them?

We could just do it once per polygon

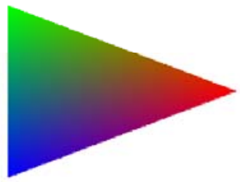
- fill every pixel covered by polygon with the resulting color

OpenGL — glShadeModel(GL_FLAT)

Shading Polygons: Gouraud Shading

Alternatively, we could evaluate at every vertex

- compute color for each covered pixel
- linearly interpolate colors over polygon



Misses details that don't fall on vertex

- specular highlights, for instance

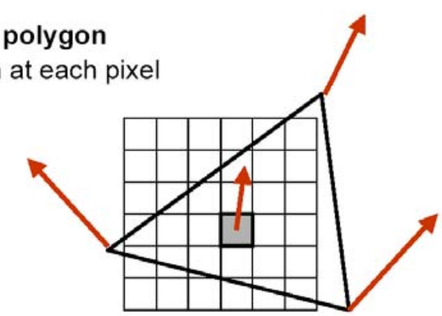
OpenGL — glShadeModel(GL_SMOOTH)

Shading Polygons: Phong Shading

Don't just interpolate colors over polygons

Interpolate surface normal over polygon

- evaluate illumination equation at each pixel

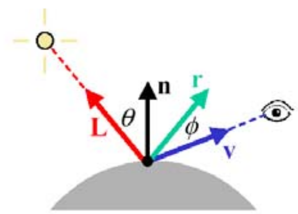


Summarizing the Shading Model

We describe local appearance with illumination equations

- consists of a sum of set of components — light is additive
- treat each wavelength independently
- currently: diffuse, specular, and ambient terms

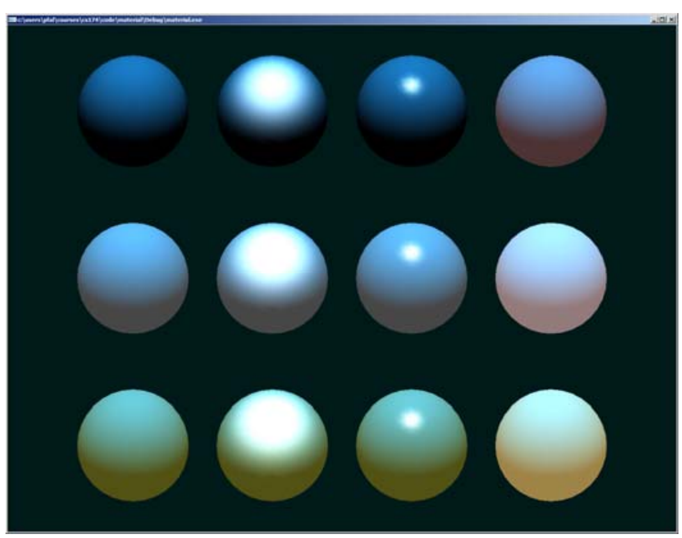
$$I = I_L k_d \cos\theta + I_L k_s \cos^n \phi + I_a k_a$$



Must shade every pixel covered by polygon

- flat shading: constant color
- Gouraud shading: interpolate corner colors
- Phong shading: interpolate corner normals

Examples



Guerrilla CG Tutorial 03: Smooth Shading

Guerrilla CG Tutorial 04: Smooth Shading Examples

Problems with Shading Algorithms

Orientation dependence

Silhouettes

Perspective distortion

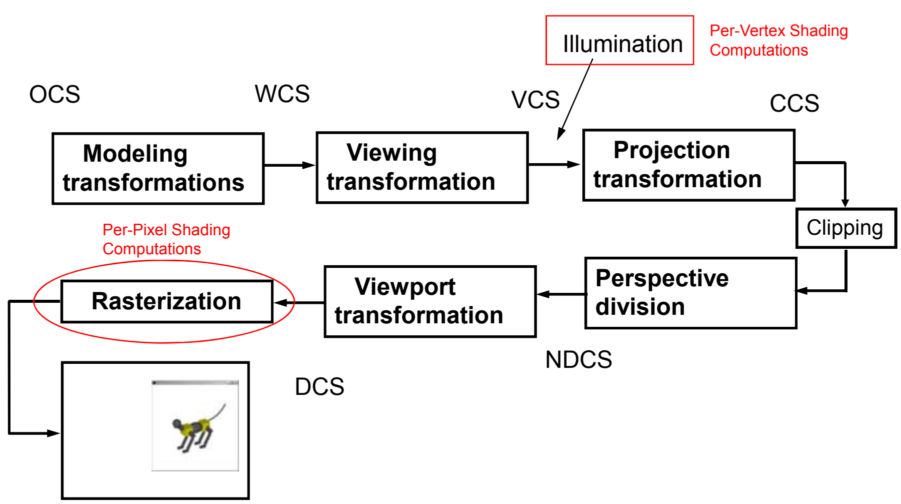
- It happens in screen space, so need to use hyperbolic interpolation

T-vertices

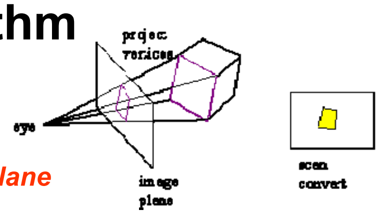
- If you do not have smooth normals, color changes if polygon order changes

Generation of vertex normals

Illumination in the Graphics Pipeline



Z-Buffer Algorithm



for each polygon in model

project vertices of polygon onto image plane

for each pixel inside projected polygon

calculate pixel z-value

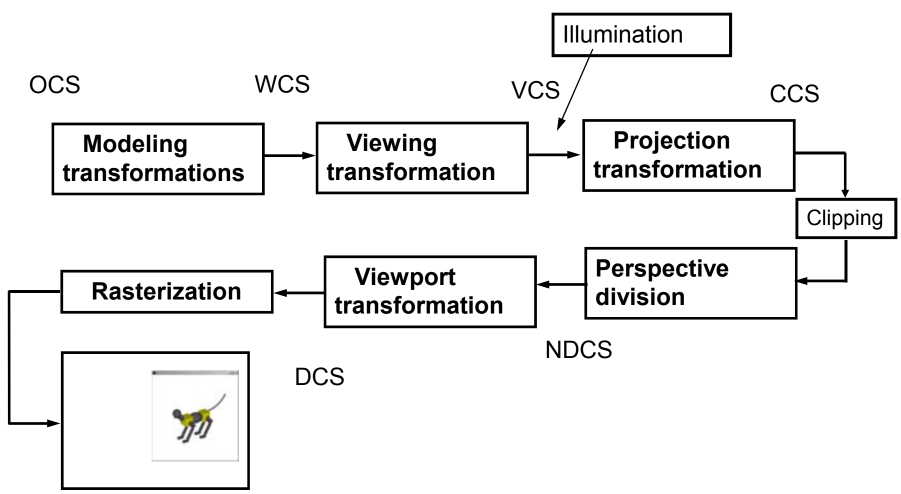
if z-value is smaller than pixel's z-value currently in z-buffer
calculate pixel color
draw pixel
update pixel z-value in z-buffer

end

end

end

Completion of the Z-Buffer Graphics Pipeline



What Have We Ignored?

- Some local phenomena
  - *Shadows – every point is illuminated by every light source*
  - *Attenuation – intensity falls off with square of distance to light source*
  - *Transparent objects – light can be transmitted through surfaces*
- Global illumination
  - *Reflections of objects in other objects*
  - *Indirect diffuse light – ambient term is just a hack*
- Realistic surface detail
  - *An orange sphere doesn't have the texture of an orange fruit*
- Realistic light sources

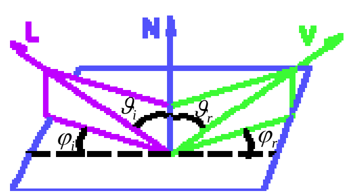
Advanced Concepts

Physics-based illumination models

Bidirectional reflectance distribution function: BRDF

$\rho(\vartheta_i, \varphi_i, \vartheta_r, \varphi_r, \lambda)$

λ : light wavelength



Global Illumination

Computing light interface between all surfaces

Radiosity

Ray tracing

Courtesy of Henrik Wann Jensen

Radiosity

Physics-based

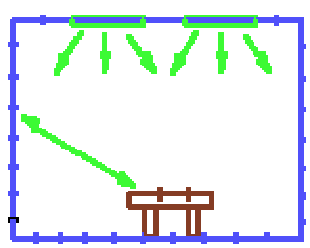
- heat transfer
- illumination engineering

Suited for diffuse reflection

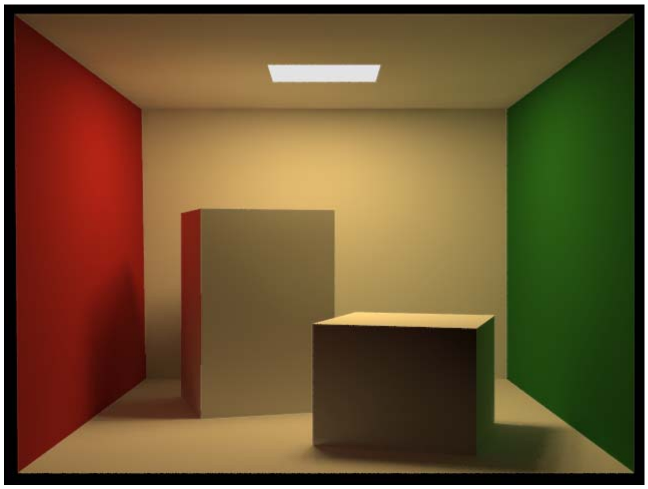
- Infinite inter-reflections

Area light sources

- Soft shadows



Example



Radiosity Algorithm

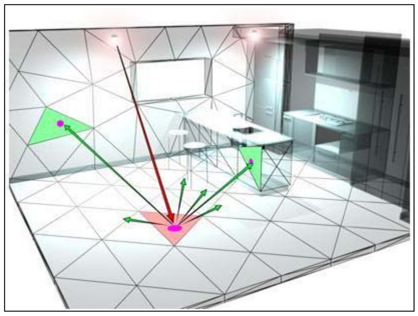
Break scene into small patches (polygons)

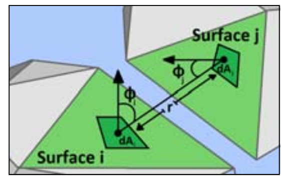
Assume uniform reflection and emission per patch

Energy balance for all patches:

Light leaving surface = Emitted light + Reflected light

Scene Polygonalization and Form Factors





Notation

- **Flux**: energy per unit time (*W*)
- **Radiosity** *B*: exiting flux density (*W*/m^2) for surfaces
- *E*: exiting flux density for light sources
- **Reflectivity** *R*: fraction of incoming light that is reflected (unitless)
- **Form factor** $F_{i,j}$: fraction of energy leaving polygon A_i and arriving at polygon A_j
  - *determined by the geometry of polygons i and j*

Energy Balance

$$\overbrace{B_i A_i}^{\text{Light leaving surface}} = \overbrace{E_i A_i}^{\text{Emitted light}} + \overbrace{R_i \sum_j B_j F_{j,i} A_j}^{\text{Reflected light}}$$

Therefore

$$B_i = E_i + R_i \sum_j B_j F_{j,i} \frac{A_j}{A_i}$$

Now $F_{j,i} A_j = F_{i,j} A_i$ (form-factor reciprocity)

Therefore

$$B_i = E_i + R_i \sum_j B_j F_{i,j}$$

or

$$E_i = B_i - R_i \sum_j B_j F_{i,j}$$

Linear System

Assume constant radiosity polygons (n of them)

Compute form factors F_{ij} for $1 \le i,j \le n$

Assemble a system of n linear equations:

$$\begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_{n-1} \\ E_n \end{bmatrix} = \begin{bmatrix} 1 - R_1 F_{1,1} & -R_1 F_{1,2} & \cdots & -R_1 F_{1,n} \\ -R_2 F_{2,1} & 1 - R_2 F_{2,2} & \cdots & -R_2 F_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ -R_{n-1} F_{n-1,1} & \cdots & 1 - R_{n-1} F_{n-1,n-1} & -R_{n-1} F_{n-1,n} \\ -R_n F_{n,1} & \cdots & -R_n F_{n,n-1} & 1 - R_n F_{n,n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_{n-1} \\ B_n \end{bmatrix}$$

n x n matrix

Solve the system for the exiting fluxes B_i

Comparison Between Direct Illumination and Radiosity

Shadow Details

Radiosity Factory

Museum

Radiosity Summary

Object space algorithm

Suited for diffuse (inter-)reflections

Area light sources

Nice, soft shadows