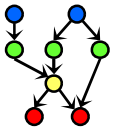


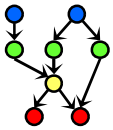
Lecture 9 of 41

Surface Detail 1 of 5: Illumination and Shading

William H. Hsu
Department of Computing and Information Sciences, KSU

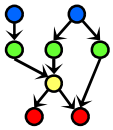
KSOL course pages: http://bit.ly/hGvXlH / http://bit.ly/eVizrE
Public mirror web site: http://www.kddresearch.org/Courses/CIS636
Instructor home page: http://www.cis.ksu.edu/~bhsu

Readings:
Today: Sections 2.5, 2.6.1-2.6.2, 4.3.2, 20.2, Eberly *2e* – see http://bit.ly/ieUq45
Next class: Section 2.7, Eberly *2e*, *Direct3D* handout
Brown CS123 slides on Illumination – http://bit.ly/fGWndj
Wayback Machine archive of Brown CS123 slides: http://bit.ly/gAhJbh



Lecture Outline

- Reading for Last Class: §2.4, 2.5 (Especially 2.5.4), 3.1.6, Eberly 2e
- Reading for Today: §2.5, 2.6.1 – 2.6.2, 4.3.2, 20.2, Eberly 2e
- Reading for Next Class: §2.7, Eberly 2e; Direct 3D Handout
- Last Time: Scan Conversion, Concluded
  - Circles and ellipses
  - Polygons: scan line interpolation (for flat/constant shading)
  - Later: Gouraud & Phong shading, z-buffering, texture mapping
- Today: Intro to Illumination and Shading
  - Views of light: microscopic *vs.* macroscopic
  - Local *vs.* global models
  - Illumination (vertex shaders) *vs.* shading (fragment/pixel shaders)
  - Light transport: Kajiya's equation, Lambert's cosine law
  - Bidirectional reflectance distribution function (BRDF) $\rho(p, \omega_i, \omega_o, \lambda)$
  - Lambertian reflectance
  - Phong illumination equation: introduction to shading

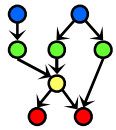


Where We Are

Lecture	Topic	Primary Source(s)
0	Course Overview	Chapter 1, Eberly 2e
1	**CG Basics: Transformation Matrices; Lab 0**	**Sections (§) 2.1, 2.2**
2	Viewing 1: Overview, Projections	§ 2.2.3 – 2.2.4, 2.8
3	Viewing 2: Viewing Transformation	§ 2.3 esp. 2.3.4; FVFH slides
4	**Lab 1a: Flash & OpenGL Basics**	**Ch. 2, 16[1], Angel *Primer***
5	Viewing 3: Graphics Pipeline	§ 2.3 esp. 2.3.7; 2.6, 2.7
6	Scan Conversion 1: Lines, Midpoint Algorithm	§ 2.5.1, 3.1; FVFH slides
7	**Viewing 4: Clipping & Culling; Lab 1b**	**§ 2.3.5, 2.4, 3.1.3**
8	Scan Conversion 2: Polygons, Clipping Intro	§ 2.4, 2.5 esp. 2.5.4, 3.1.6
9	Surface Detail 1: Illumination & Shading	§ 2.5, 2.6.1 – 2.6.2, 4.3.2, 20.2
10	**Lab 2a: Direct3D / DirectX Intro**	**§ 2.7, Direct3D handout**
11	Surface Detail 2: Textures; OpenGL Shading	§ 2.6.3, 20.3 – 20.4, *Primer*
12	Surface Detail 3: Mappings; OpenGL Textures	§ 20.5 – 20.13
13	**Surface Detail 4: Pixel/Vertex Shad.; Lab 2b**	**§ 3.1**
14	Surface Detail 5: Direct3D Shading; OGLSL	§ 3.2 – 3.4, Direct3D handout
15	Demos 1: CGA, Fun; Scene Graphs: State	§ 4.1 – 4.3, **CGA handout**
16	**Lab 3a: Shading & Transparency**	**§ 2.6, 20.1, *Primer***
17	**Animation 1: Basics, Keyframes; HW/Exam**	**§ 5.1 – 5.2**
	Exam 1 review; Hour Exam 1 (evening)	**Chapters 1 – 4, 20**
18	**Scene Graphs: Rendering; Lab 3b: Shader**	**§ 4.4 – 4.7**
19	**Demos 2: SFX; Skinning, Morphing**	**§ 5.3 – 5.5, CGA handout**
20	Demos 3: Surfaces; B-reps/Volume Graphics	§ 10.4, 12.7, **Mesh handout**

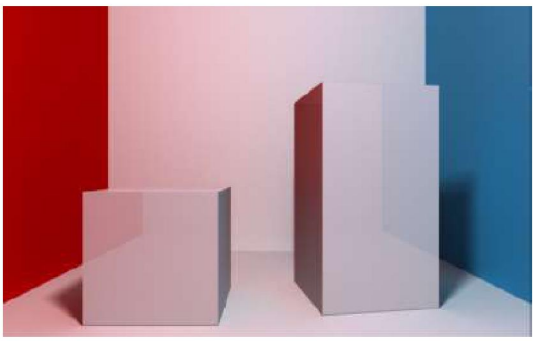
Lightly-shaded entries denote the due date of a written problem set; heavily-shaded entries, that of a machine problem (programming assignment); blue-shaded entries, that of a paper review; and the green-shaded entry, that of the term project.

Green, **blue** and **red** letters denote **exam review**, **exam**, and **exam solution review** dates.



Acknowledgements

Andy van Dam
T. J. Watson University Professor of Technology and Education &
Professor of Computer Science
Brown University
http://www.cs.brown.edu/~avd/

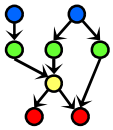


Cornell box rendered using path tracing

Illumination and Light Transport

Chapters 29, 30

These slides combine material from Andy van Dam, Spike Hughes, Travis Webb, Ben Herila, Paul Sastrasinh, and Lyn Fong

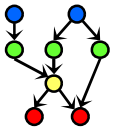


Outline

- **What is Light?**
- The Rendering Equation
- Illumination vs. Shading
- Local vs. Global Illumination
- Computing Illumination
- Modeling Reflectance
- Illumination Models
- Shading Models
- Advanced Global Illumination Techniques

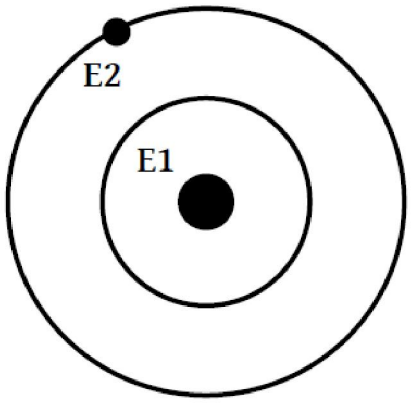
Cornell box rendered using photon mapping (notice the soft shadows)

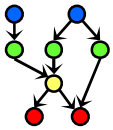
Photo credit: © Ben Herila, 2010



Microscopic View of Light [1]: Photons & Planck's Law

- Light can be thought of as small packets of energy called photons
- Every atom has a nucleus which electrons orbit at various levels
- When an electron drops from a higher level orbit to a lower, energy (photons) are emitted (*Planck's Law*)
- When an emitted photon strikes another atom and is absorbed, electrons jump from a lower orbital to a higher one
- The wavelength (color) of light emitted or absorbed corresponds to the change in orbital.
- The nature of atoms and their electrons control the type of light emission



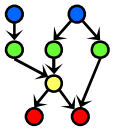


Microscopic View of Light [2]: Energy, Conductivity & Reflectivity

- Metals have electrons which are not attached to a particular nucleus, but instead have nuclei in a sea of electrons
  - These electrons can move around, making metallic objects conductive
  - These electrons also can have many possible energy states (orbit levels), which is why most conductive materials are also reflective (and transparent materials are good insulators)

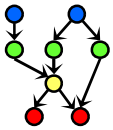
Heating something makes its electrons move more rapidly; the energy gained from heat is lost as light.

Photo credit: © Fir0002/Flagstaffotos, 2010, used under GFDL license



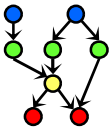
Microscopic View of Light [3]: Photon Absorption

- Other materials, like carbon, have unattached electrons as well, but they cannot move as freely
  - These electrons can interact with the atom causing it to vibrate and create heat while the electron loses energy
  - Soot absorbs photons and tends to convert it heat, which causes it to look black



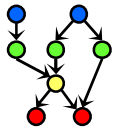
Microscopic View of Light [4]: Light Emission

- To create light, we need to excite an object's electrons (heating the object)
  - heat causes atoms to vibrate, which may result in the "kicking around" of a loose electron; the energy can be released
    - in the form a photon, light, or heat
    - as a loose electron (kinetic energy), which is what the cathode in a CRT does; those electrons cause phosphor electrons to escape their shells, fall back and release visible energy in R, G, or B range
- A material that is good at absorbing energy and turning it to heat is also good at emitting energy when heated
- As materials are heated they become better and better emitters of energy
  - All bodies at all temperatures above absolute zero will emit *some* radiation, but not much at most ordinary temperatures

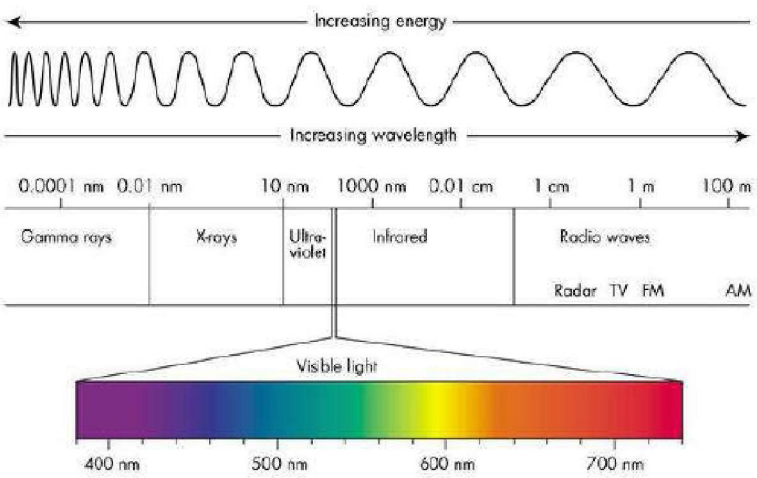


Macroscopic View of Light [1]: Radiation & Wave/Particle Duality

- Light is a type of electromagnetic radiation and can also be thought of as a continuous wave (as opposed to a discrete photon)
- The wave nature of light is best used to describe how light propagates or travels, whereas the particle description is best used to determine how energy is exchanged
- The wavelength (λ) of a wave is measured as the distance from the one peak of a wave to the next
  - Visible light has a wavelength between 400nm – 700nm, a very small portion of the entire spectrum



Macroscopic View of Light [2]: Electromagnetic Spectrum

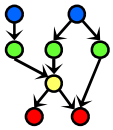


The full electromagnetic spectrum.

Note that visible light has a wavelength between 400nm – 700nm, a very small portion of the spectrum.

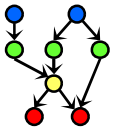
We're only interested in the visible spectrum – we define and group major wavelength ranges:

red, green, and blue



Outline

- What is Light?
- **The Rendering Equation**
- Illumination vs. Shading
- Local vs. Global Illumination
- Computing Illumination
- Modeling Reflectance
- Illumination Models
- Shading Models
- Advanced Global Illumination Techniques

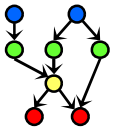


Rendering Equation [1]: (Kajiya, 1986)

- Computing *light transport* in a scene with both direct light sources (emitters) and indirect sources, i.e. inter-object reflection – basically an energy balance
- We compute integrals of the form

$$I(x,x') = g(x,x')\left[\epsilon(x,x') + \int_S \rho(x,x',x'')f(x',x'')dx''\right]$$

- $\epsilon(x,x')$ represents the amount of light emitted from the material (zero for most objects)
- $g(x,x')$ represents the geometry of the scene – we only want light that is not occluded by another object (visibility test)
- The rendering equation is the foundation of almost all rendering algorithms (radiosity, photon mapping, path tracing...)



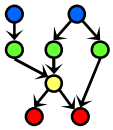
Rendering Equation [2]: Geometry, Luminance, BRDF

- Let's write the equation another way

$$I(x,x') = g(x,x')\left[\epsilon(x,x') + \int_S \rho(x,x',x'')f(x',x'')dx''\right]$$

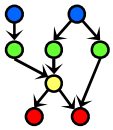
$$L(i \to j) = G(i \leftrightarrow j)\left[L_e(i \to j) + \int_S f(k \to i \to j)L(k \to i)dk\right]$$

- Light energy traveling from point i to j = light emitted from i to j, plus the integral over S (all points on all surfaces) of reflection coefficient from point k to i to j, times the light from k to i, all attenuated by a geometry factor.
  - $L(i \to j)$ is the total amount of light traveling along the ray from point i to point j
  - L is the amount of light emitted by the surface (**luminance**)
  - $F(k \to i \to j)$ is the **Bidirectional Reflectance Distribution Function (BRDF)** of the surface. Describes fraction of the light incident on the surface at i from the direction of k leaves the surface in direction of j – attenuation factor between 0 and 1
  - $G(i \leftrightarrow j)$ is a **geometry term** which involves occlusion, distance, and the angle between the surfaces



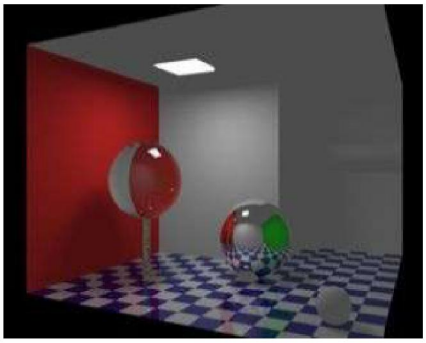
Illumination & Shading

- *Lighting*, or *illumination*, is the process of computing the intensity and color of a sample point in a scene as seen by a viewer
  - lighting is a function of the geometry of the scene (including the model, lights and camera and their spatial relationships) and material properties
- *Shading* is the process of *interpolation* of color at points in-between those with known lighting or illumination, typically vertices of triangles or quads in a mesh
  - used in many real time graphics applications (e.g., games) since calculating illumination at a point is usually expensive
- On the GPU, lighting is calculated by a *vertex shader*, while shading is done by a fragment or *pixel shader*

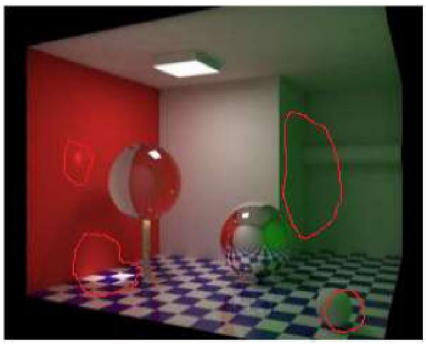


Illumination Models [1]: Local vs. Global Models

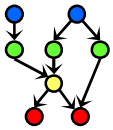
- The rendering equation takes into account global information of both direct (from emitters) and indirect illumination (inter-object reflections)
  - the most realistic illumination models try to take this global data into account
  - purely local models can also produce useful results at a far smaller cost



Radiosity (for diffuse reflection) + raytracing (for specular reflection)

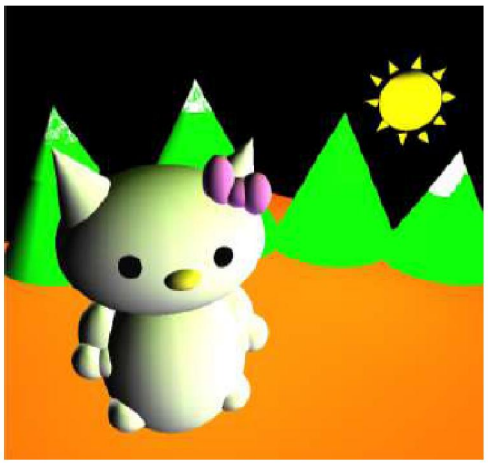


Same image, adding multi-bounce lighting, caustics, and color bleeding

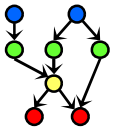


Illumination Models [2]: Local Illumination

- Take only direct lighting information into account when computing a sample
- Local illumination is an approximation to global illumination
- Usually involves an "ambient term" to set a sort of minimum bar for inter-object reflection and thus light up most visible surfaces



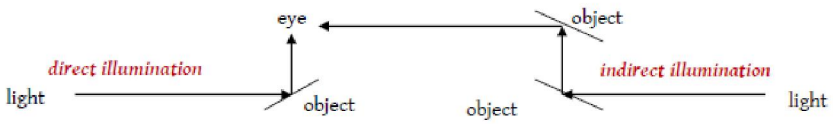
Nancy Tom's ***Sceneview*** *scene, rendered using OpenGL with only local illumination (2001).*

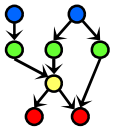


Illumination Models [3]: Global Illumination

- Simulates what happens when other objects and scene elements affect light reaching a surface element
- Lights and shadows
  - most light striking a surface element comes directly from emissive light sources in the scene (direct illumination)
  - sometimes light from a source is blocked by other objects
  - surface element is then in "shadow" from that light source
- Inter-object reflection
  - light bounces off other objects toward our surface element
  - when that light reaches our surface element, it brightens it (indirect illumination)

from Pixar's "Luxo Jr."



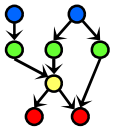


Illumination Models [4]: Tradeoffs (Pros & Cons)

- Local models concentrate on light from direct sources
  - **pro:** scene can be rendered fast
  - **con:** pay a price in lost realism; lose interesting effects of light transport because we ignore effects of all other objects in the scene when considering a particular surface element

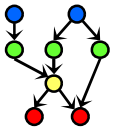
- Global models concentrate on capturing the all illumination information
  - **pro:** shadows, inter-object reflection, refraction, i.e. bending of light at translucent surfaces, volumetric effects of participating media such as air, water, and fog
  - **con:** slow

Frederic Drago and Karol Myszkowski, *Validation Proposal for Global Illumination and Rendering Techniques*
http://www.mpi-sb.mpg.de/resources/atrium/



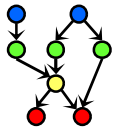
Outline

- What is Light?
- The Rendering Equation
- Illumination vs. Shading
- Local vs. Global Illumination
- **Computing Illumination**
- Modeling Reflectance
- Illumination Models
- Shading Models
- Advanced Global Illumination Techniques



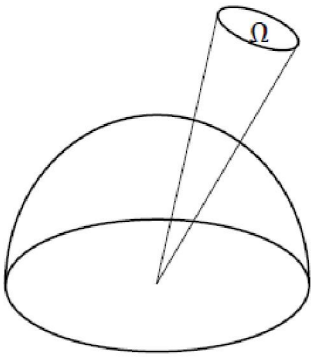
Illumination Models [5]: Computing Illumination

- Illumination can be computed by:
  - ***Light transport simulation:*** Evaluate illumination with enough samples to produce a final image without any guessing / shading
    - often used for high quality renderers and movies
    - some implementations can run in real time on the GPU, but more complex lighting models still must be run on the CPU
    - renders of highest quality can take days for a single frame, even on modern distributed CPU render farms and/or GPU render farms
  - ***Polygon rendering:*** Evaluate illumination at several samples, and shade (using a shading model) in between to produce pixels in the final image
    - often used in real-time applications such as computer games, done in GPU
    - lower quality than light transport simulation.
- Note that the BRDF is often implicit in simplest illumination models, and parameters (typically diffuse and specular reflection coefficients) are adjusted experimentally until desired outcome is achieved



Computing Illumination: Light Transport [1]

- Light may arrive at a point from a range of directions (denoted as Ω) which is often called the ***solid angle***, the 3D equivalent of a 2D angle



2D angle is related to the *arc length*, where l is the arc length and r is the radius of the circle

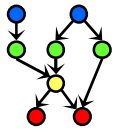
$$\theta = \frac{l}{r}$$

In 3D, solid angle (measured in ***steradians***) is related to the *surface area* it cuts out, where S is the surface area and r is the radius of the sphere [1]

$$\Omega = \frac{S}{r^2}$$

[1] *For the proof, consult the textbook or take cs224*

Image credit: © Ben Herila, 2010

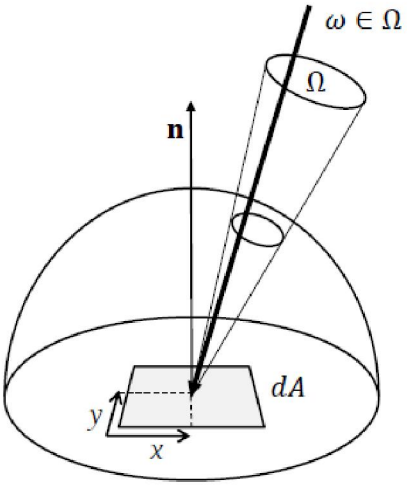


Computing Illumination: Light Transport [2]

- Amount of light arriving at a surface at a certain point in time as an integral over the solid angle Ω

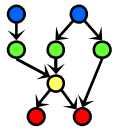
$$E = \int_A \int_{\omega \in \Omega} L(t, x, y, \omega, \lambda) |\omega \cdot \mathbf{n}| d\omega \, dA$$

- This measure of light energy is known as *irradiance*
- Let's break the integral down...



illuminating hemisphere

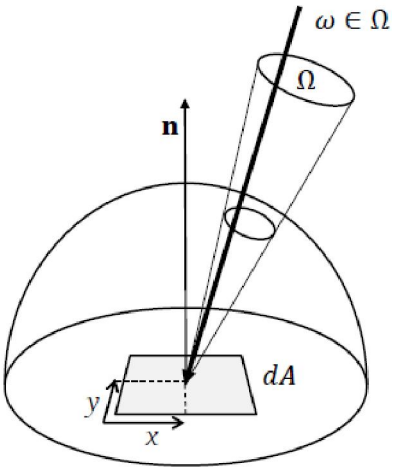
Image credit: © Ben Herila, 2010



Computing Illumination: Light Transport [3]

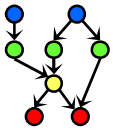
$$E = \int_A \int_{\omega \in \Omega} L(t, x, y, \omega, \lambda) |\omega \cdot \mathbf{n}| d\omega \, dA$$

- Ω is the solid angle, which includes all directions of incoming light
- ω is one direction out of all the possible directions, thus $\omega \in \Omega$
- λ is the wavelength (color) of the incoming light
- (x, y) is the point on the differential area over which we are integrating
- t is the time (constant for static scenes)



illuminating hemisphere

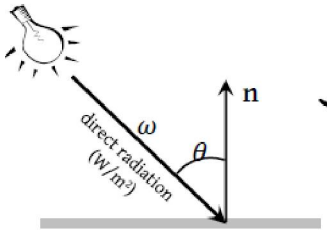
Image credit: © Ben Herila, 2010

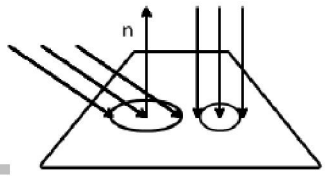


Computing Illumination: Light Transport [4]

$$E = \int_A \int_{\omega \in \Omega} L(t, x, y, \omega, \lambda) |\omega \cdot \mathbf{n}| d\omega \, dA$$

- $|\omega \cdot \mathbf{n}|$ ~ the cosine of the angle of the normal of the surface to the incoming light direction



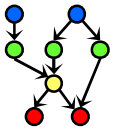


$$\{\theta = \text{acos}(\omega \cdot \mathbf{n})\}$$

Lambert's cosine law

- Light reflected from a surface is proportional to the cosine of the angle between the view direction the surface normal
- For the same amount of energy:
  - when the incoming energy is aligned with the normal vector, you get the greatest energy per unit area (flux)
  - When the angle is acute, the same amount of energy is distributed over a larger area; hence, the flux is less

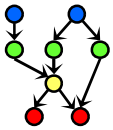
Image credit: © Ben Herila, 2010



Outline

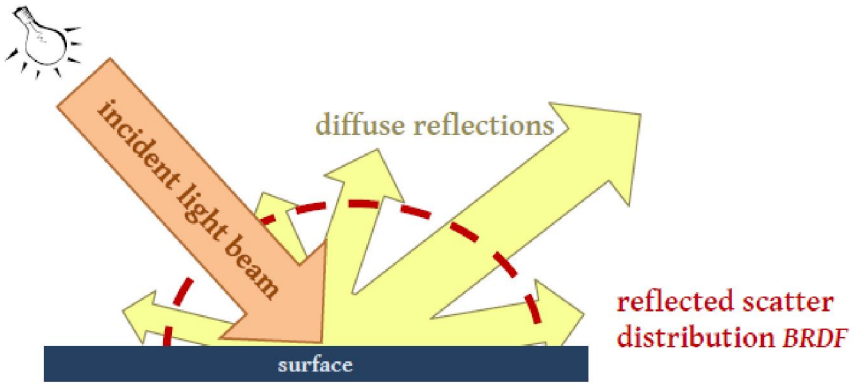
- What is Light?
- The Rendering Equation
- Illumination vs. Shading
- Local vs. Global Illumination
- Computing Illumination
- **Modeling Reflectance**
- Illumination Models
- Shading Models
- Advanced Global Illumination Techniques

bubble with reflection, by d ha rm e sh



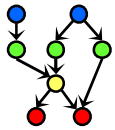
Modeling Reflectance: BRDF [1]

- Light arriving at a surface can scatter in many directions
  - The direction of scattering is determined by the material
  - Outgoing light direction is usually dependent on incoming light direction
- Measurement of reflectance is called the *bidirectional reflectance distribution function* (*BRDF*), denoted ρ (*rho*)



Adapted from slides © 2010 van Dam *et al.*, Brown University
http://bit.ly/hiSt0f Reused with permission.

Image credit: © Ben Herila, 2010

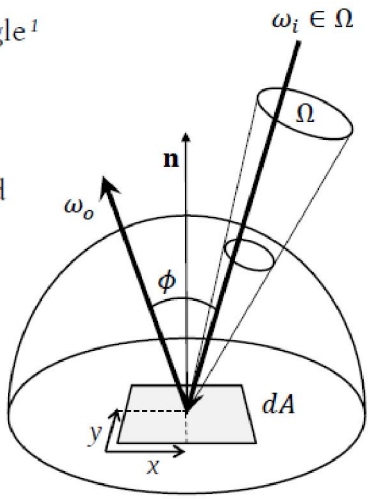


Modeling Reflectance: BRDF [2]

- The *BRDF* (ρ) can be written as the fraction of outgoing light over incoming light at a given angle[1]

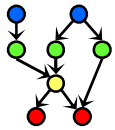
$$\rho(p, \omega_i, \omega_o, \lambda) = \frac{L(t, p, \omega_o \lambda)}{L(t, p, -\omega_i, \lambda)\cos(\phi)\, m(\Omega)}$$

- $m(\Omega)$ is the measure of the solid angle projected by the light source at the sample
- $\omega_{i,o}$ are the incoming and outgoing light directions, respectively
- $p \equiv (x, y)$ for brevity
- λ is the wavelength, as before
- t is the current time, as before



[1] *Again, for the proof, consult the textbook or take cs224*

Image credit: © Ben Herila, 2010

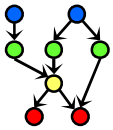


Modeling Reflectance: BRDF [3]

$$\rho(p, \omega_i, \omega_o, \lambda) = \frac{L(t, p, \omega_o \lambda)}{L(t, p, -\omega_i, \lambda)\cos(\phi)\, m(\Omega)}$$

- The BRDF of most materials satisfies ***Helmholtz reciprocity***

$$\rho(p, \omega_i, \omega_o, \lambda) = \rho(p, \omega_o, \omega_i, \lambda)$$

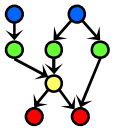


Modeling Reflectance: Deriving Rendering Equation

- Let's put together what we have so far
- Amount of light reflected at a given surface point $p = (x, y)$ is

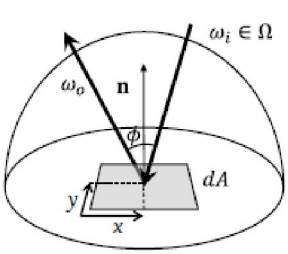
$$L(t, p, \omega_o, \lambda) = \int_{\omega \in \Omega^+} \underbrace{L(t, p, \omega_i, \lambda)}_{\text{sum of all incoming light at } \mathbf{p}} \underbrace{\rho(p, \omega_i, \omega_o, \lambda)}_{\text{multiplied by the BRDF of the surface corresponding to each light direction}} \underbrace{|\omega_i \cdot \mathbf{n}|}_{\text{attenuated by the cosine of the angle}} d\omega_i$$

- Very similar to the formula for irradiance
- Preliminary form of the rendering equation

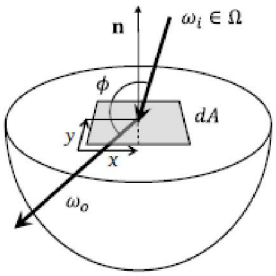


Modeling Reflectance: Materials [1]

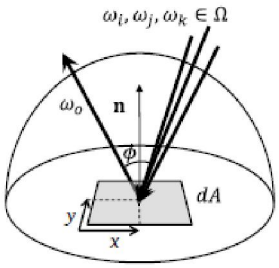
- When light hits a material, it scatters according to the properties of that material
- There are several types of light scattering
  - *Reflective* – Light scatters in the upper hemisphere
  - *Transmissive* – Light scatters in the lower hemisphere
  - *Impulse* – Light scatters in a single direction (regardless of incoming direction)



Single reflective ray
reflection angle ϕ is acute

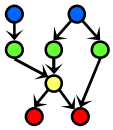


Single transmissive ray
reflection angle ϕ is obtuse



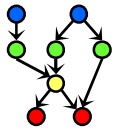
Impulse rays
incoming rays all exit in same direction

Image credit: © Ben Herila, 2010



Modeling Reflectance: Materials [2]

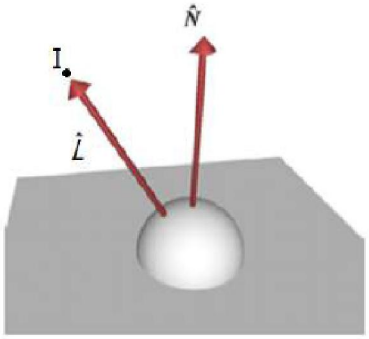
- Additional types of light scattering
  - *Mirror* – light scatters in a single direction, $\omega_r = \omega + 2(\omega \cdot \mathbf{n})\mathbf{n}$
  - *Specular* – light scatters tightly around a particular direction (shiny objects with sharp highlights)
  - *Glossy* – light scatters weakly around a particular direction
  - *Diffuse* – light scatters (possibly unevenly) in all outgoing directions e.g. paper, wood
  - *Lambertian* – light scatters equally (output radiance is equal) in all outgoing directions (i.e. viewer independent)

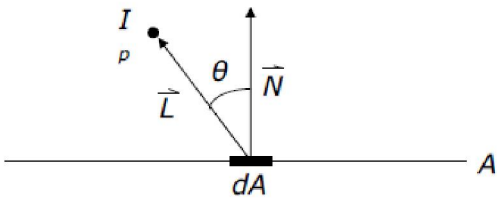


Modeling Reflectance: Lambertian (Diffuse) [1]

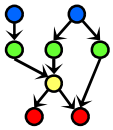
- Lambertian surfaces have uniform scattering; same apparent brightness independent of view direction and incoming light direction
- Most materials are not perfectly Lambertian; most BRDFs are viewer dependent
- Lambert's cosine law:

$$I = I_p k_d \cos\theta = I_p k_d (N \cdot L)$$



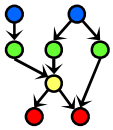


Imagine this: it's raining, and you have a sheet of paper. You can either hold the paper flat, or you can hold it at an angle. In which orientation will the paper get wetter? Now imagine it's raining *photons* – which orientation will result in more reflection?



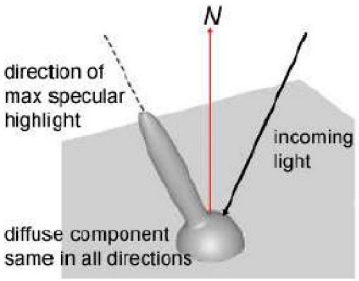
Modeling Reflectance: Lambertian (Diffuse) [2]

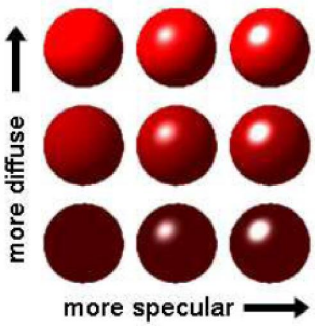
- Several lighting models attempt to approximate these phenomena
- Some are just "hacks" – no physical foundation, but looks okay
  - Phong Model
  - Blinn-Phong Model
- Some more complicated models are physically based
  - Cook-Torrance Model
  - Oren-Nayer Model
- Which model we use largely depends on our application
- Usually a performance vs. accuracy tradeoff

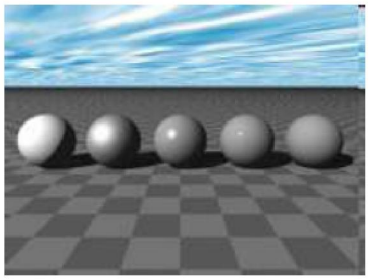


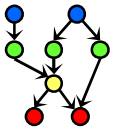
Modeling Reflectance: Specular

- Most materials are not perfectly Lambertian; most BRDFs are viewer dependent
  - this is the *specular* property of the material
  - BRDF value is greatest when $\omega_o = \omega_i$ (when the outgoing angle is opposite the incoming angle), like a perfect mirror or rippling water





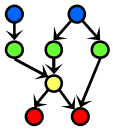




Modeling Reflectance: BSSRDF

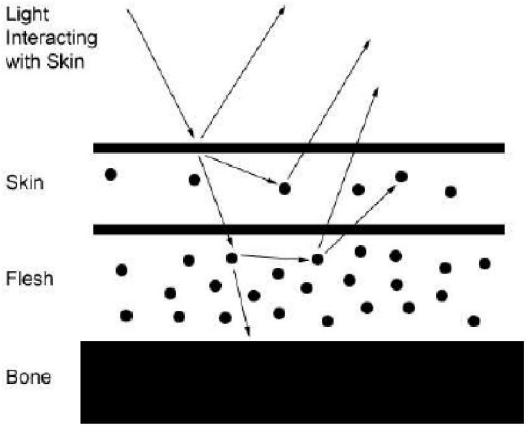
- All of these models model light when it hits a surface directly
- Light may also be scattered / absorbed while traveling through participating media (e.g., Fog)
- The BRDF can be generalized to model transmission through an object (i.e., Refraction) and sub-surface scattering by defining other terms, such as the *Bidirectional scattering surface reflectance distribution function* (BSSRDF)

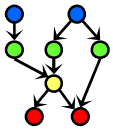
Image: No scattering (top) vs. with BSSRDF scattering (bottom).



Modeling Reflectance: Subsurface Scattering

- Some surfaces may also reflect light internally a little before letting the light escape again (subsurface scattering)
  - skin, wax, hair, milk and other fluids
- More complex models are needed to approximate these interactions





Modeling Reflectance: Advanced Techniques

- There are many more complex and more accurate BRDFs such as Blinn's and Anisotropic
- There are even more complicated distribution functions such as BSSDF or bidirectional subsurface scattering function
- Researchers collect tables of data for B*DFs of specific materials using devices like the one pictured

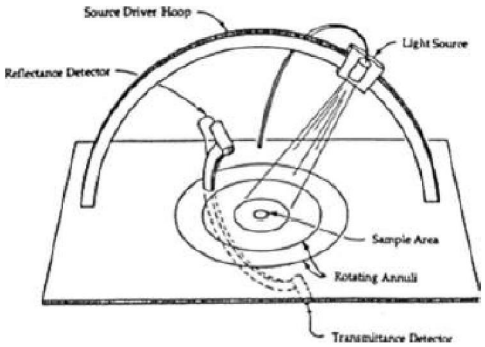
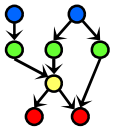
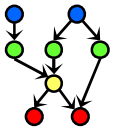


Image credit: Chuck Moidel



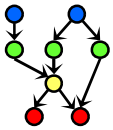
Outline

- What is Light?
- The Rendering Equation
- Illumination vs. Shading
- Local vs. Global Illumination
- Computing Illumination
- Modeling Reflectance
- **Illumination Models**
- Shading Models
- Advanced Global Illumination Techniques



Illumination Models: Lambertian [1]

- Models perfectly diffuse reflections only
  - objects which scatter light uniformly in all directions (ex. matte paint)
  - BRDF is constant at all angles for a surface point
- Materials with these properties satisfy Lambert's cosine law
  - exitant radiance observed from all directions is directly proportional to $\cos(\omega_i)$
  - the light energy observed is not dependent on the outgoing direction
- This is the type of illumination you experimented with in the WPF 3D lab (remember that?)

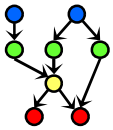


Illumination Models: Lambertian [2]

To compute the integral we saw before, simply sum over all the lights in the scene:

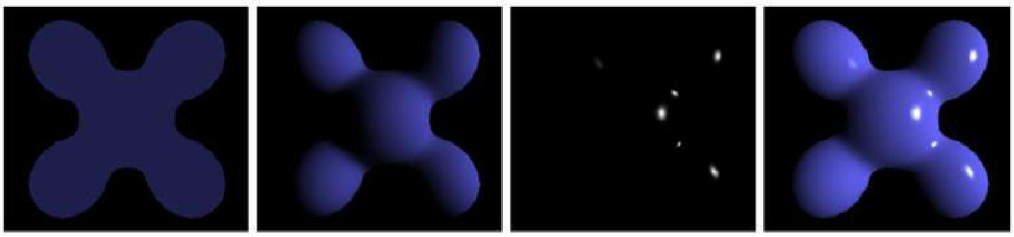
$$i_\lambda = \sum_{m \in lights} i_{d_\lambda} k_{d_\lambda} (\mathbf{n} \cdot \mathbf{L_m}) O_d$$

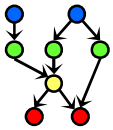
- λ is the wavelength or color (i.e. R, G, or B)
- i is the light intensity measured at the object surface
- k is the material's efficiency at reflecting light (attenuation coefficient)
- O represents the color of an object's material at a point (the BRDF for a diffuse material)
- $\mathbf{L}$ is the direction to the light from the surface point
- $\mathbf{n}$ is the surface normal at the point
- Subscript a and d represent ambient and diffuse (so k_a is the ambient coefficient)



Illumination Models: Phong [1]

- Simple model (*not physically based*)
- Splits illumination at a surface into three components
  - Ambient – Non-specific constant global lighting (hack)
  - Diffuse – Color of object under normal conditions using Lambert's model
  - Specular – Highlights on shiny objects (hack)
    - Proportional to $(\omega_o \cdot \mathbf{V})^{\alpha}$ so a larger α results in a more concentrated highlight and glossier object

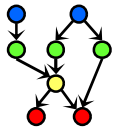




Illumination Models: Phong [2]

$$I_\lambda = i_{a_\lambda} k_a O_{d_\lambda} + \sum_{lights} i_{dir} k_d O_d \left(\mathbf{n} \bullet \vec{L}\right)$$

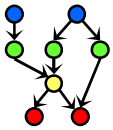
- Variables
  - λ = color component (e.g. R, G, and B)
  - i = intensity of light as measured at surface
  - i_a = the amount of ambient light used in the scene
  - k = material's efficiency at reflecting light (attenuation coefficient)
  - k_a is the ambient attenuation coefficient for this object's material (we would expect $k_a \sim k_d$)
  - O = innate color of object's material at specific point on surface



Illumination Models: Phong [3]

$$I_\lambda = i_{a_\lambda} k_a O_{d_\lambda} + \sum_{lights} i_{dir} k_d O_d \left(\mathbf{n} \bullet \vec{L}\right)$$

- Ambient component
  - effect on surface constant regardless of orientation, no geometric information
  - total hack (crudest possible approximation to inter-object reflection), but makes all objects a little visible
- Diffuse component
  - uses Lambert's diffuse-reflection cosine law
  - i_{dir} (light's intensity) and ℓ (light's direction) vary for each light source
  - k_d is the diffuse attenuation coefficient
  - O_d = innate color of object's diffuse material property at specific point on surface

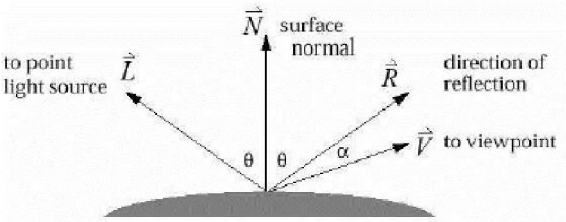


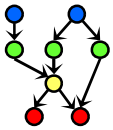
Illumination Models: Phong [4]

- The full Phong model is a combination of the Lambertian and specular terms (summing over all the lights)

$$I_\lambda = i_{a_\lambda} k_{a_\lambda} O_{d_\lambda} + \sum_{m \in lights} f_{att} i_{d_\lambda} \left[k_{d_\lambda} (\mathbf{n} \cdot \boldsymbol{L_m}) O_{d_\lambda} + k_{s_\lambda} (\mathbf{R_m} \cdot \mathbf{V})^{\alpha} O_{s_\lambda} \right]$$

- Subscript s represents specular (so k_s) would be the specular coefficient
- $\boldsymbol{R_m}$ is the reflected direction of the light ray about the surface normal
- f_{att} is the lighting attenuation function
  - function of distance from the light

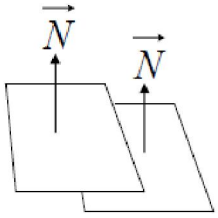




Illumination Models: Phong [5]

- By the inverse square law, density of light energy decreases by the inverse square of the distance from the surface, due to spherical radiation pattern, so

$$I = (f_{att})(I_p k_d)(n \cdot \vec{L}), \text{ where } f_{att} = \frac{1}{d_{light}^2}$$



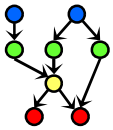
- This will attenuate the light intensity according to distance, so farther objects appear darker, i.e. surfaces with equal $(k_d)(n \cdot \vec{L})$ vary in appearance if they are at different distances from the light – important if two surfaces overlap

- But this doesn't look good – objects illuminated by point lights will look as if they were very harshly lit (things get dark too fast)

- Instead, use the following heuristic

$$f_{att} = \min\left\{\frac{1}{c_1 + c_2 d_{light} + c_3 d_{light}^2}, 1\right\}$$

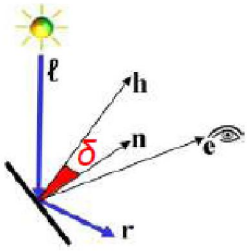
where c_1, c_2, c_3 are experimentally-defined constants (this is a hack)



Illumination Models: Blinn-Phong (See Eberly 2e)

- Variation on Phong model which modifies the specular term to be more computationally efficient
- The new specular term uses the half angle between the viewer and the light (constant throughout the scene) instead of the angle between the surface and the light (varying for each triangle)

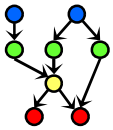
$$\sum_{m \in lights} i_d k_s O_s (\mathbf{n} \cdot \mathbf{h})^{\alpha}$$



- **h** is the half angle

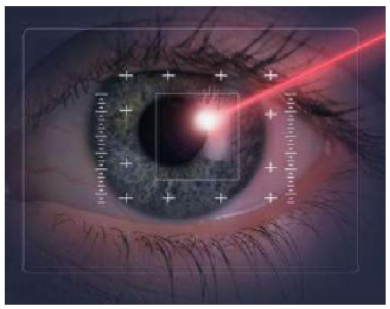
$$h = \frac{l + e}{|l + e|}$$

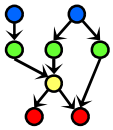
- l is the direction from the point to the light
- Blinn-Phong or Phong?
  - Doesn't matter, they look slightly different but they're both hacks



Illumination Models: Computing Results

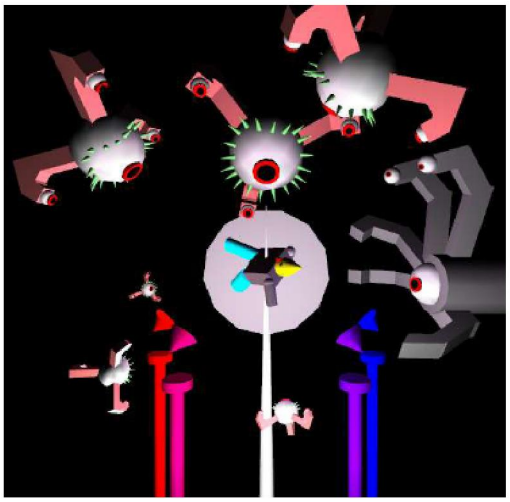
- Look closely at the specular term: $k_s O_{s\lambda} (\bar{R} \bullet \bar{V})^n$
- We can compute this term recursively
  - shoot a ray from eye through a pixel on the screen
  - calculate intersections of ray with all primitives in the scene, pick the closest one
  - shoot a specular reflection ray from the intersection point to closest object to calculate its specular contribution
  - apply our simple illumination model at intersection point
  - can even be done in hardware thanks to modern GPU's
- This is **Ray Tracing** – simple global model
- You will be implementing soon

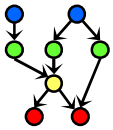




Outline

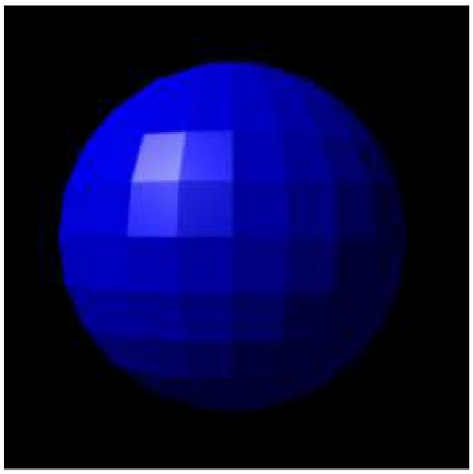
- What is Light?
- The Rendering Equation
- Illumination vs. Shading
- Local vs. Global Illumination
- Computing Illumination
- Modeling Reflectance
- Illumination Models
- **Shading Models**
- Advanced Global Illumination Techniques



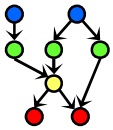


Shading Models: Flat/Constant (No Interpolation)

- We define a normal at each polygon (not at each vertex)
- **Lighting:** Evaluate the lighting equation at the center of each polygon using the associated normal
- **Shading:** Each sample point on the polygon is given the calculated lighting value



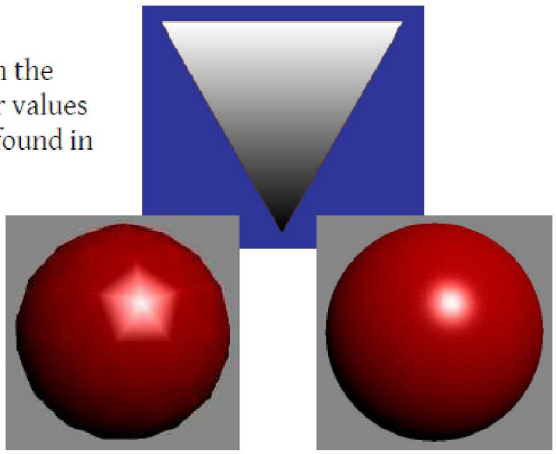
`GL_CONSTANT`

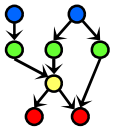


Shading Models: Gouraud (Color Interpolation)

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

`GL_SMOOTH`

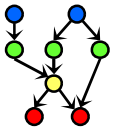




Shading Models: Phong (Vertex Normal Interpolation)

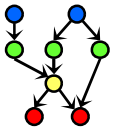
- Each vertex has an associated normal vector
- **Lighting:** Evaluate the lighting equation at each vertex using the associated normal vector
- **Shading:** For every sample point on the polygon we interpolate the normals at vertices of the polygon and compute the color using the lighting equation with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

OpenGL implementation?
Stay tuned…



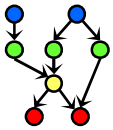
Preview: Advanced Global Illumination (GI) Techniques

- In practice, calculating global illumination for a scene is a very complex and computationally intensive task
  - Many algorithms (both real time and non real time) have been developed to calculate or approximate global illumination
- Real time
  - Ambient occlusion, image based lighting
- Not real time
  - Metropolis light transport, photon mapping, radiosity, point based color bleeding



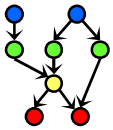
Preview: More on Shading (Surface Detail 2, 4, 5)

- Shading in OpenGL
  - Flat/constant: `GL_CONSTANT`
  - Gouraud: `GL_SMOOTH`
- Shading Languages
  - Renderman Shading Language (RSL) – http://bit.ly/g229q4
  - OpenGL Shading Language (OGLSL or GLSL) – http://bit.ly/fX8V0Y
  - Microsoft High-Level Shading Language (HLSL) – http://bit.ly/eVnjp5
  - nVidia Cg – http://bit.ly/ewoRic
- Vertex *vs.* Pixel Shaders
- How to Write Shaders



Summary

- Last Time: Scan Conversion, Concluded
  - Circles and ellipses
  - Polygons: scan line interpolation (for flat/constant shading)
- Today: Intro to Illumination and Shading
  - Local illumination (Phong, *etc.*) *vs.* global (ray tracing, radiosity)
  - Illumination (vertex shaders) *vs.* shading (fragment/pixel shaders)
  - Light transport
    - Kajiya's equation
    - Lambertian reflectance: cosine law
  - Bidirectional reflectance distribution function (BRDF) $\rho(p, \omega_i, \omega_o, \lambda)$
  - Phong illumination equation
  - Shading
    - Gouraud: color interpolation (per-vertex Phong illumination)
    - Phong: normal interpolation (per-pixel Phong illumination)
- Next: Direct3D Tutorial



Terminology

- Illumination – Computing Light that Reaches a Surface Patch
  - Local – based on computations over surface patch, light direction
  - Global – based on interobject reflectance and transmission
- Inputs
  - Light properties – point light source *vs.* emissive light source
  - Reflectance properties – how light is reflected back into scene
    - Diffuse: omnidirectional
    - Specular: directed, based on highlights
- Computations
  - Lambertian reflectance – based on cosine law
  - Bidirectional reflectance distribution function (BRDF) ρ
    - Measurement of reflectance
    - $\rho(p, \omega_{in}, \omega_{out}, \lambda)$ where $p \in (x, y)$, λ is wavelength
  - Phong illumination equation – ambient, diffuse, specular
- Shading – Application of Illumination to Find/Interpolate Color