CAP4730: Computational Structures in Computer Graphics

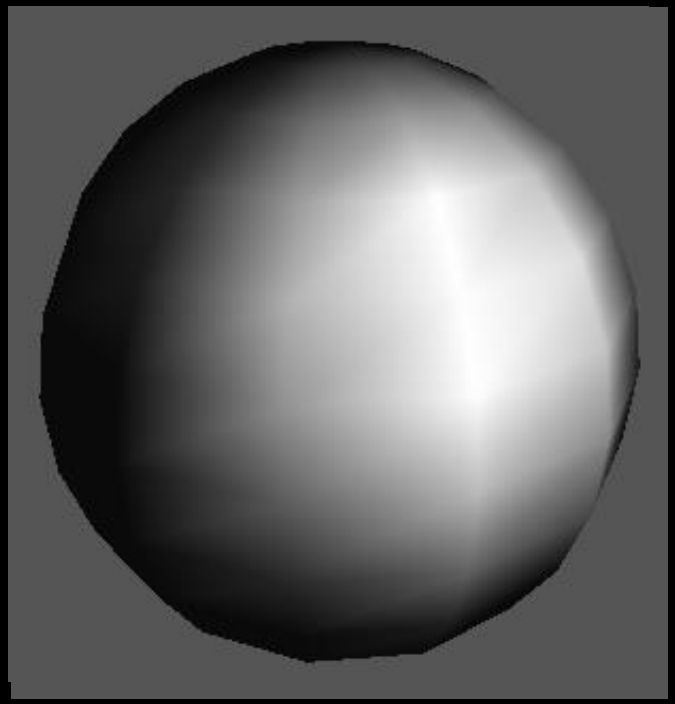
Lighting and Shading

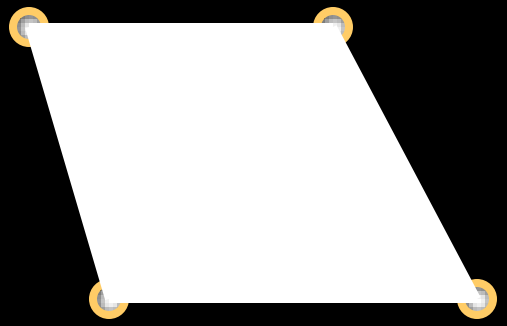
Outline

- Lighting
- Lighting models
  - Ambient
  - Diffuse
  - Specular
- Surface Rendering Methods
- Ray-Tracing

What we know

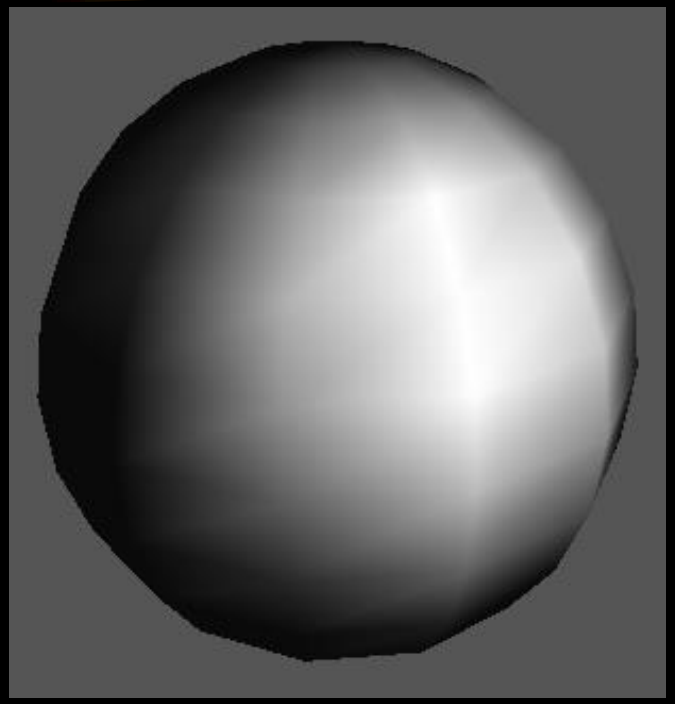
- We already know how to render the world from a viewpoint.





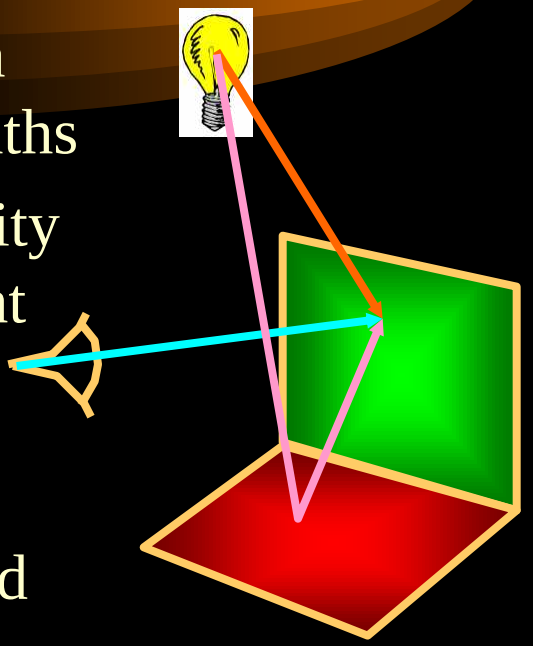
"Lighting"

- Two components:
  - **Lighting Model** or **Shading Model -** how we calculate the intensity at a point on the surface
  - **Surface Rendering Method -** How we calculate the intensity at each pixel



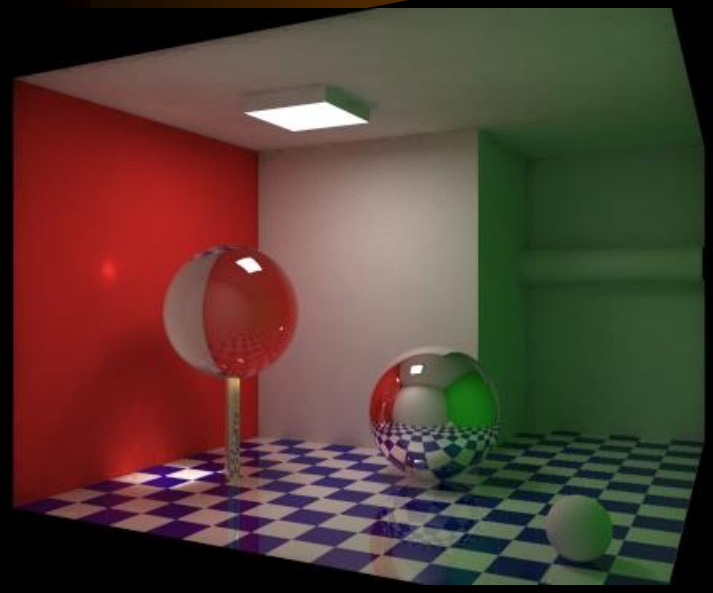
Jargon

- Illumination - the transport of light from a source to a point via direct and indirect paths
- Lighting - computing the luminous intensity for a specified 3D point, given a viewpoint
- Shading - assigning colors to pixels
- Illumination Models:
  - Empirical - approximations to observed light properties
  - Physically based - applying physics properties of light and its interactions with matter



The lighting problem…

- What are we trying to solve?
- **Global illumination** – the transport of light within a scene.
- What factors play a part in how an object is “lit”?
- Let’s examine different items here…

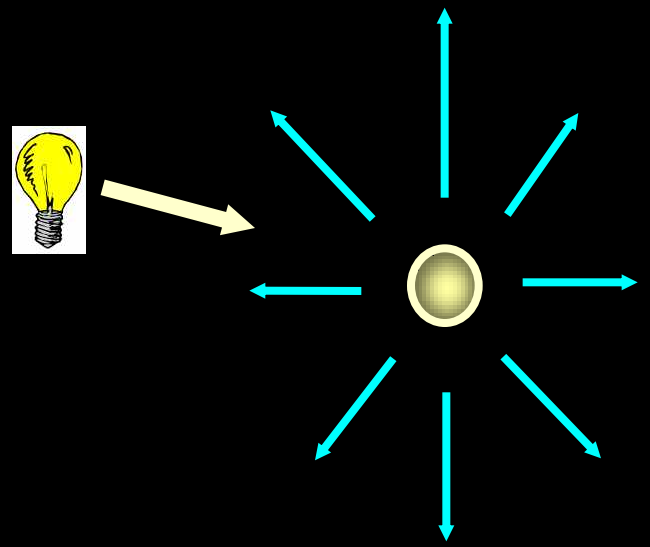


Two components

- Light Source Properties
  - Color (Wavelength(s) of light)
  - Shape
  - Direction
- Object Properties
  - Material
  - Geometry
  - Absorption

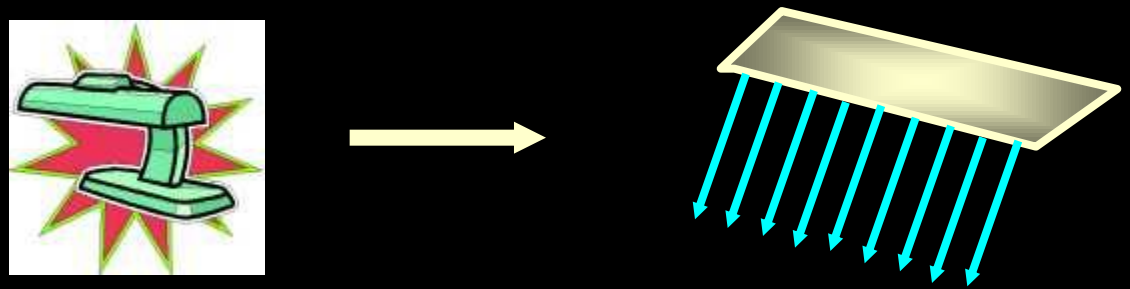
Light Source Properties

- Color
  - We usually *assume* the light has one wavelength
- Shape
  - point light source - approximate the light source as a 3D point in space. Light rays emanate in all directions.
    - good for small light sources (compared to the scene)
    - far away light sources



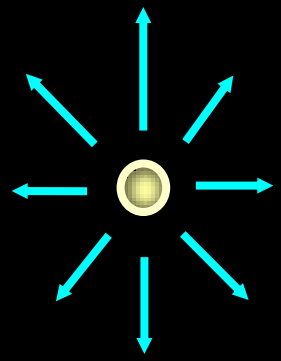
Distributed Lights

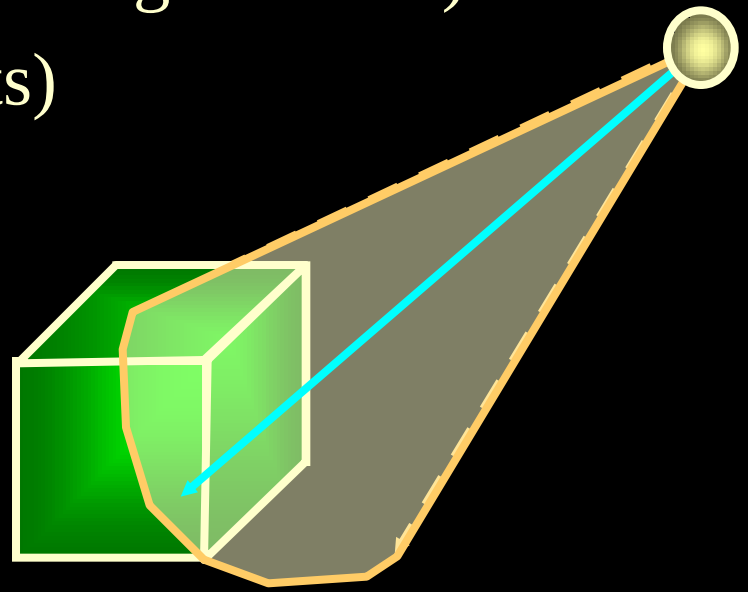
- Light Source Shape continued
  - distributed light source - approximating the light source as a 3D object. Light rays *usually* emanate in specific directions
    - good for larger light sources
    - area light sources



Light Source Direction

- In computer graphics, we usually treat lights as *rays* emanating from a source. The *direction* of these rays can either be:
  - Omni-directional (point light source)
  - Directional (spotlights)





Light Position

- We can specify the position of a light one of two ways, with an *x*, *y*, and *z* coordinate.
  - What are some examples?
  - These lights are called ***positional lights***
- Q: Are there types of lights that we can simplify?

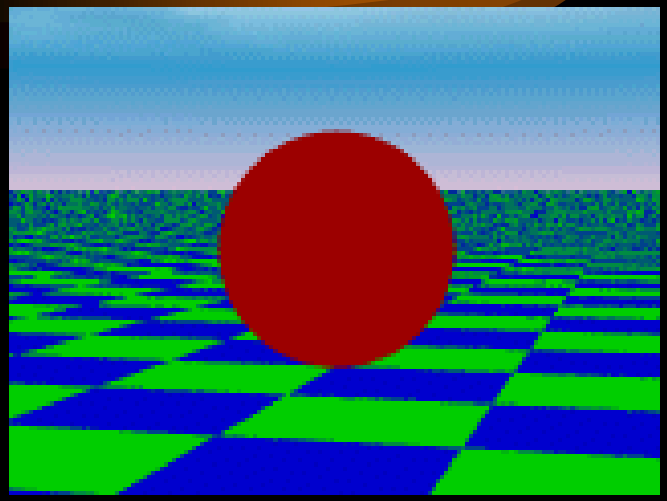
A: Yep! Think about the sun. If a light is significantly far away, we can represent the light with *only* a direction vector. These are called ***directional lights.*** How does this help?

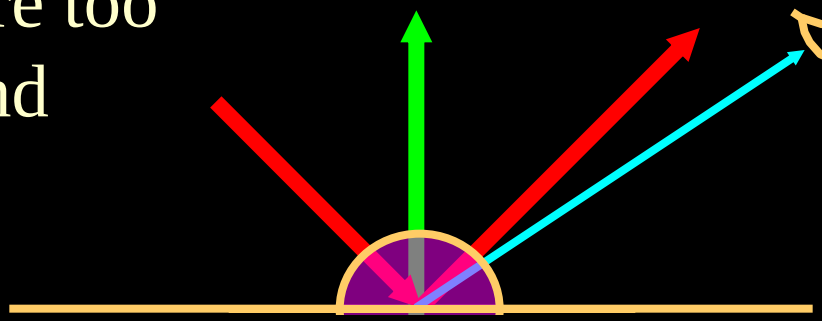
Contributions from lights

- We will breakdown what a light does to an object into three different components. This APPROXIMATES what a light does. To actually compute the rays is too expensive to do in real-time.
  - Light at a pixel from a light = Ambient + Diffuse + Specular contributions.
  - $I_{light} = I_{ambient} + I_{diffuse} + I_{specular}$

Ambient Term - Background Light

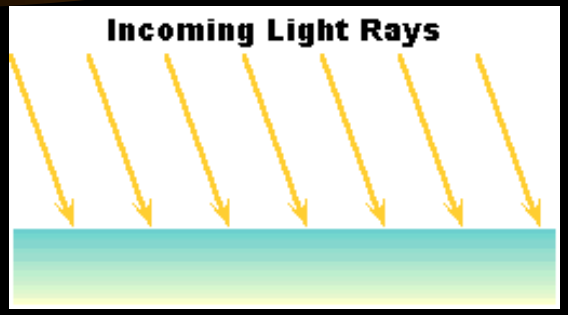
- The ambient term is a HACK!
- It represents the approximate contribution of the light to the general scene, regardless of location of light and object
- Indirect reflections that are too complex to completely and accurately compute
- $I_{ambient}$ = color

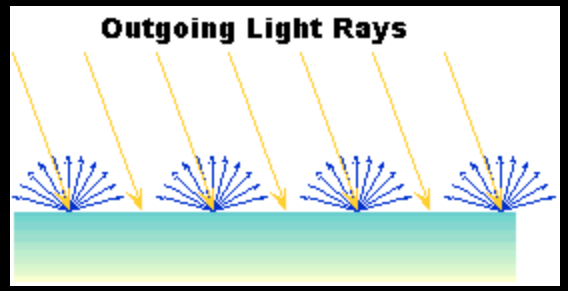




Diffuse Term

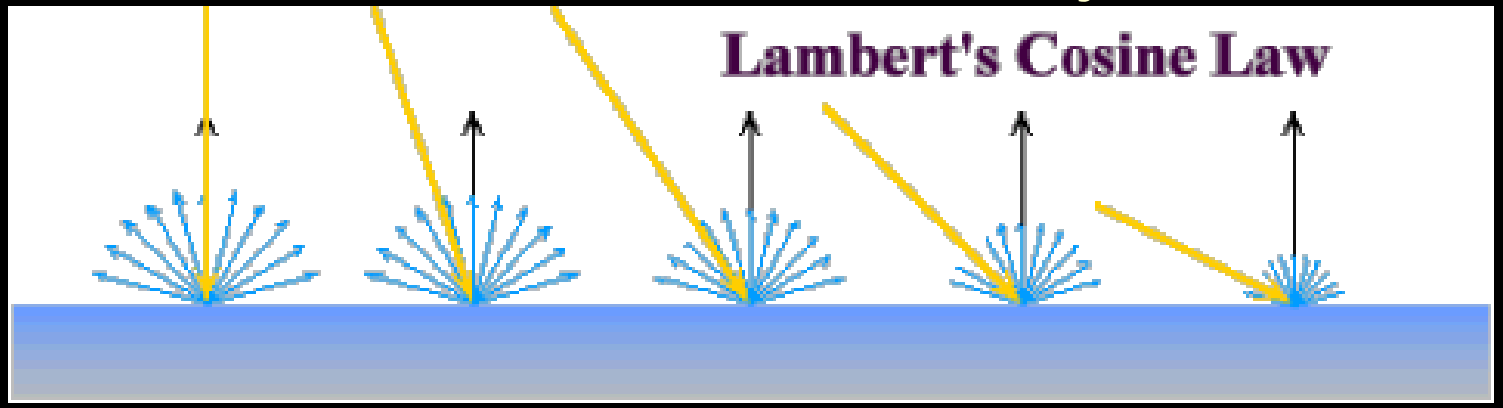
- Contribution that a light has on the surface, ***regardless of viewing direction***.
- Diffuse surfaces, on a microscopic level, are very rough. This means that a ray of light coming in has an equal chance of being reflected in *any* direction.
- What are some ideal diffuse surfaces?





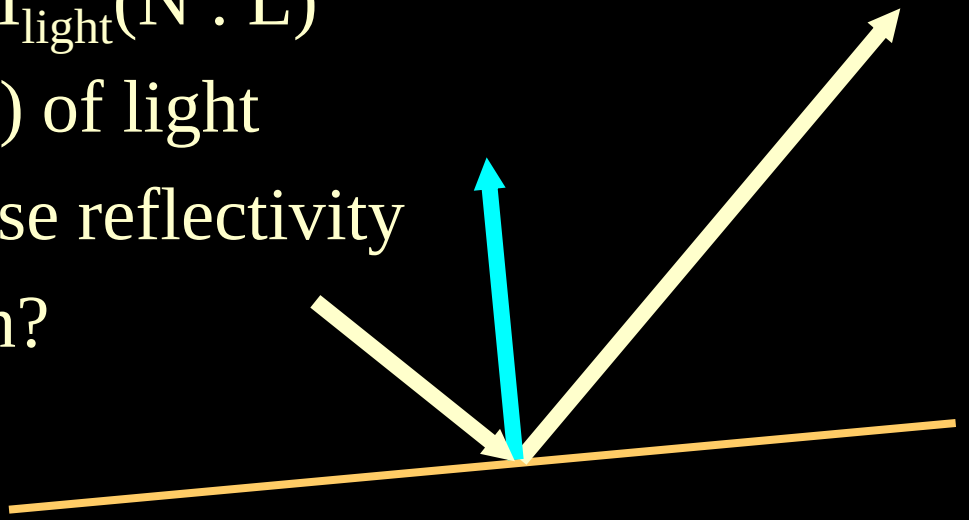
Lambert's Cosine Law

- Diffuse surfaces follow Lambert's Cosine Law
- Lambert's Cosine Law - reflected energy from a small surface area in a particular direction is proportional to the cosine of the angle between that direction and the surface normal.
- Think about surface area and # of rays



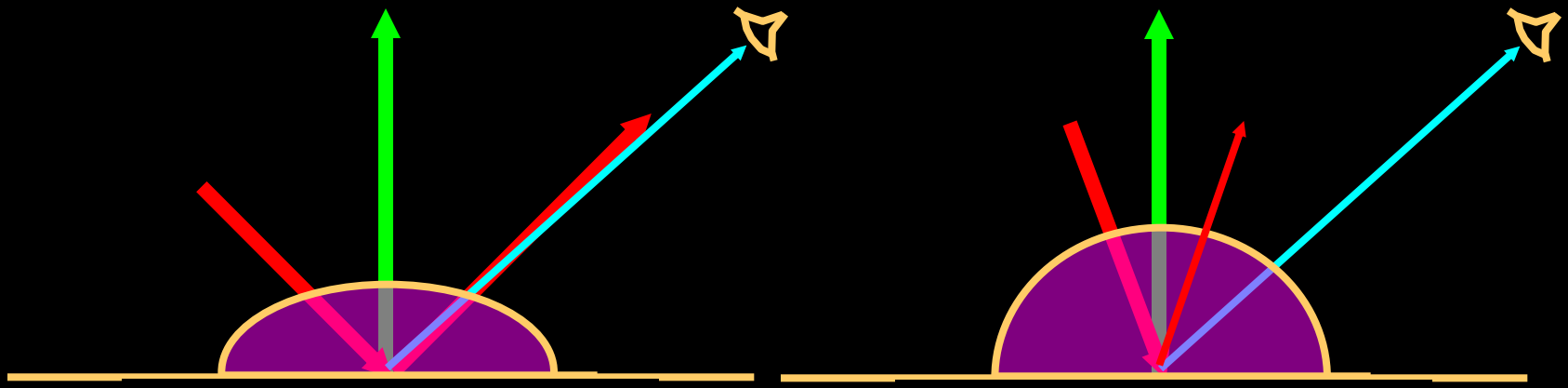
Diffuse Term

- To determine how much of a diffuse contribution a light supplies to the surface, we need the surface normal and the direction on the incoming ray
- What is the angle between these two vectors?
- $I_{diffuse} = k_d I_{light} \cos\theta = k_d I_{light}(N \cdot L)$
- I_{light} = diffuse (intensity) of light
- k_d [0..1] = surface diffuse reflectivity
- What CS are L and N in?
- How expensive is it?



Example

- What are the possible values for theta (and thus the dot product?)



http://graphics.lcs.mit.edu/classes/6.837/F98/Lecture18/Slide11.html

Specular Reflection

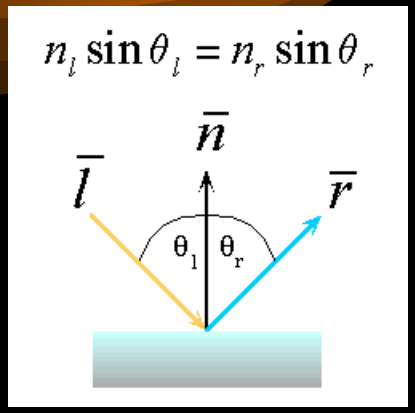
- Specular contribution can be thought of as the “shiny highlight” of a plastic object.
- On a microscopic level, the surface is very smooth. Almost all light is reflected.
- What is an ideal purely specular reflector?
- What does this term depend on?

  Viewing Direction

  Normal of the Surface

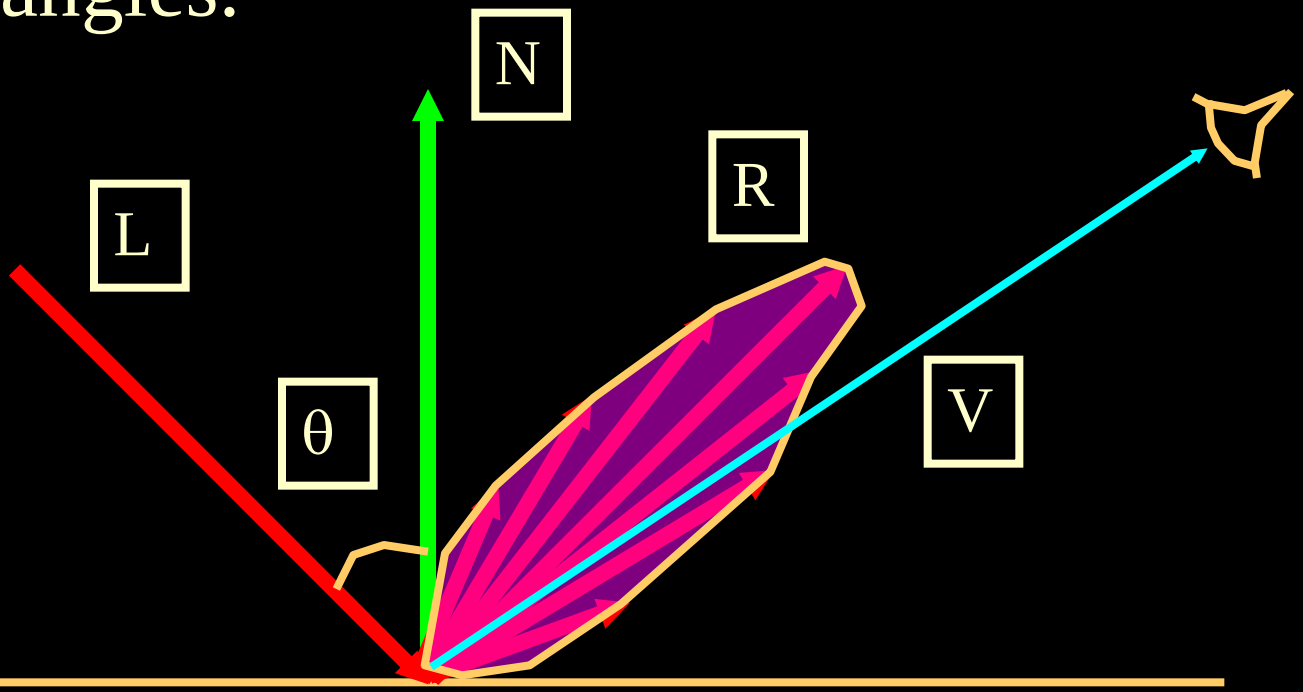
Snell's Law

- Specular reflection applies Snell's Law.
  - The incoming ray, the surface normal, and the reflected ray all lie in a common plane.
  - The angle that the reflected ray forms with the surface normal is determined by the angle that the incoming ray forms with the surface normal, and the relative speeds of light of the mediums in which the incident and reflected rays propagate according to:
  - We assume $\theta_l = \theta_r$

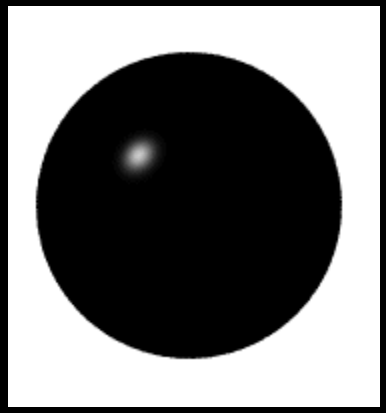


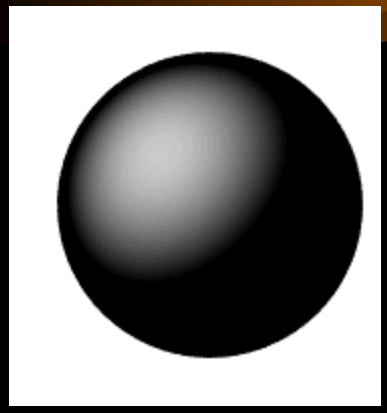
Snell's Law is for IDEAL surfaces

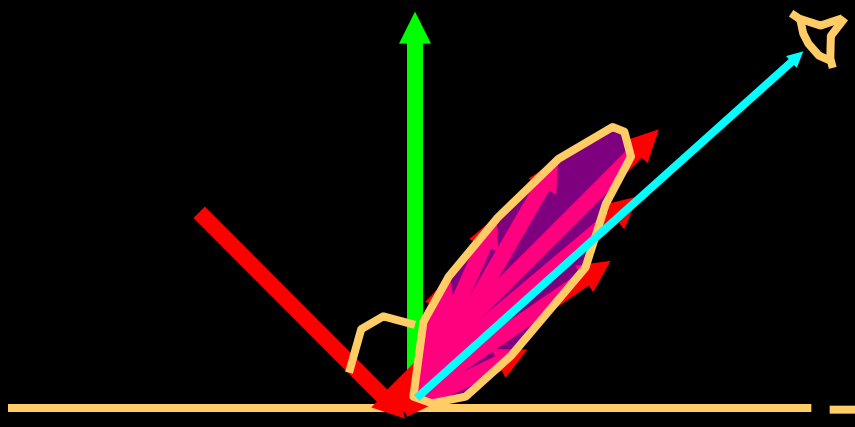
- Think about the amount of light reflected at different angles.

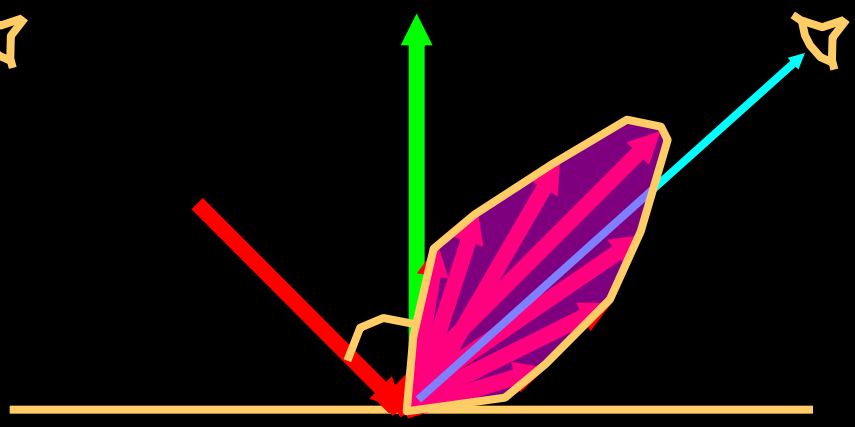


Different for shiny vs. dull objects



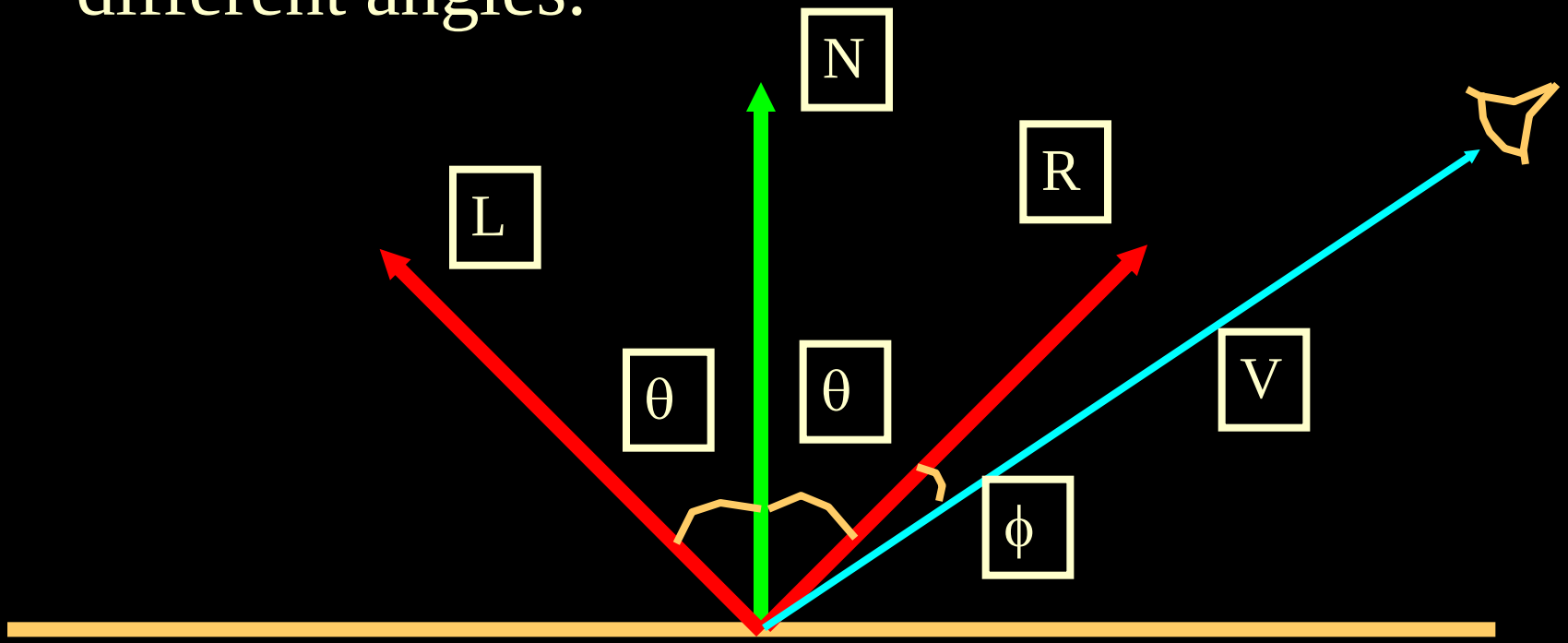






Snell's Law is for IDEAL surfaces

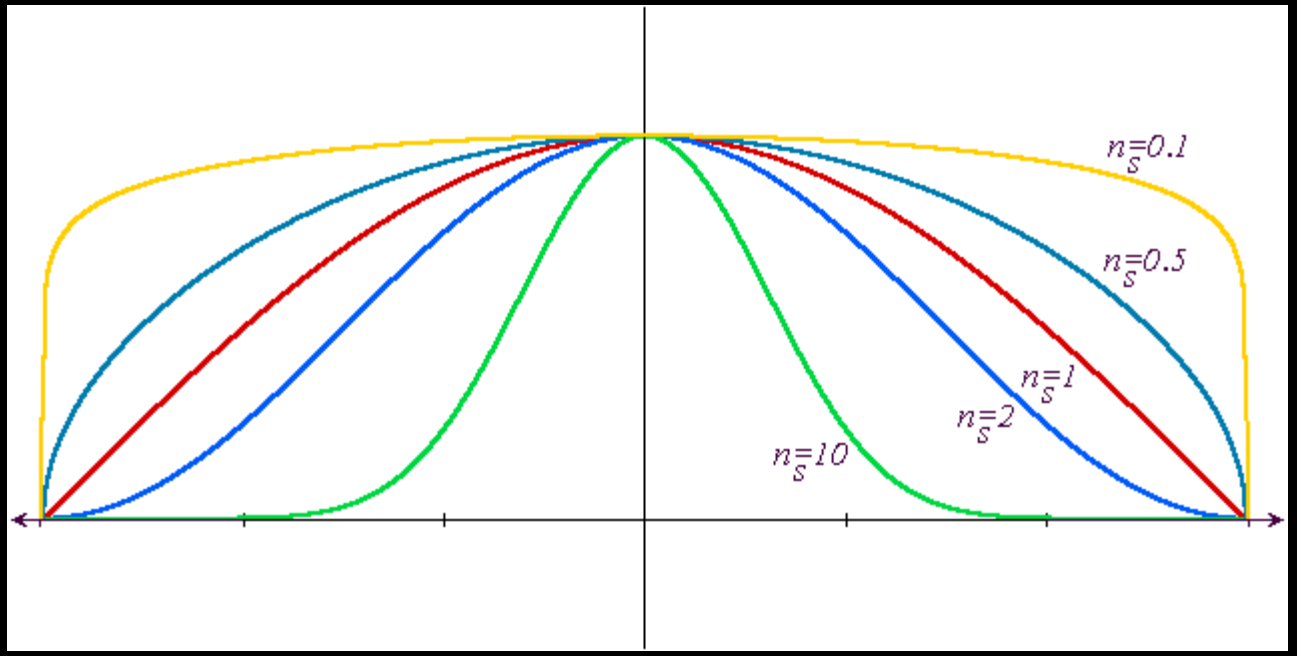
- Think about the amount of light reflected at different angles.



Phong Model
Phong Reflection Model

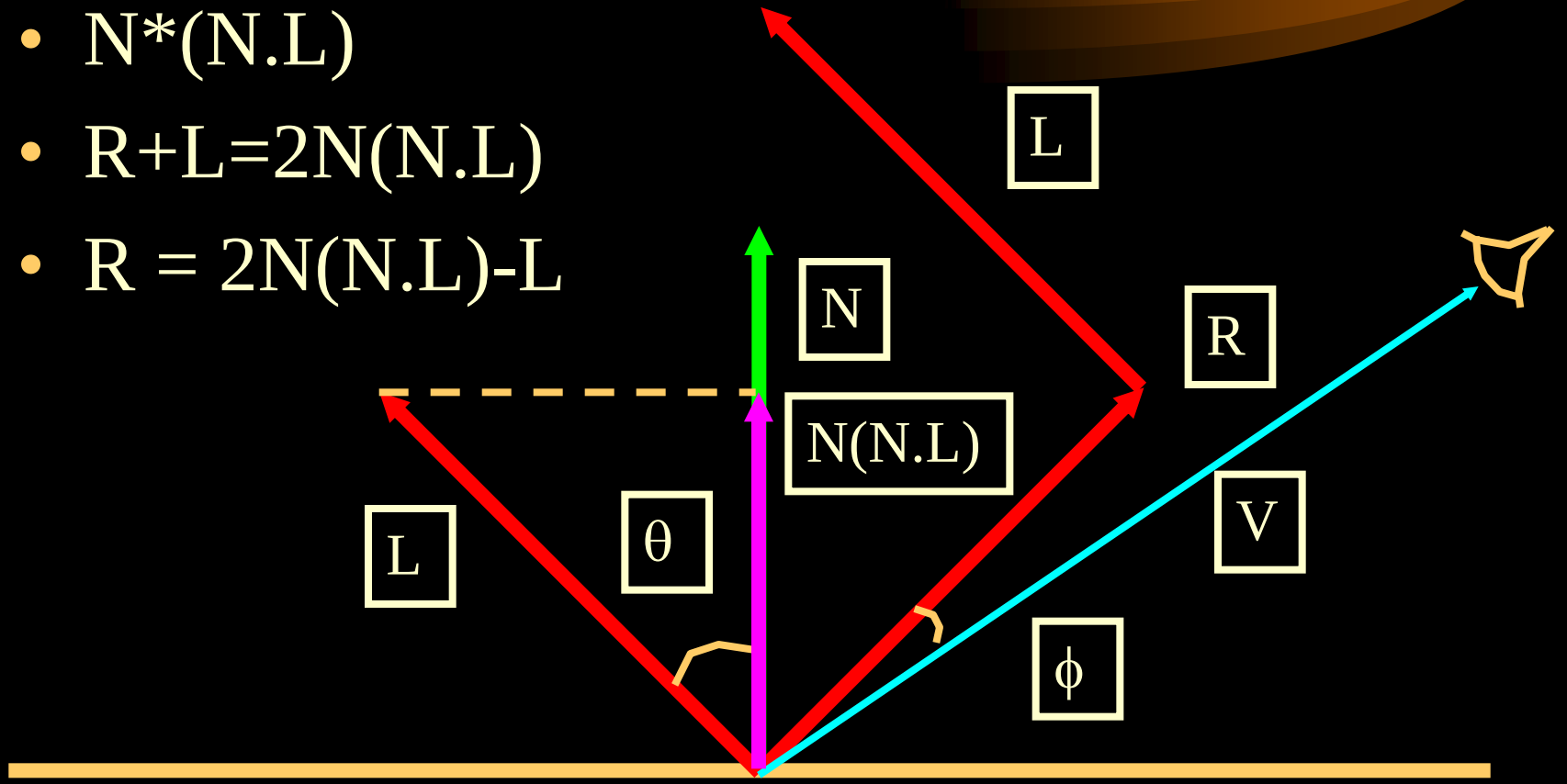
- An approximation is sets the intensity of specular reflection proportional to $(\cos \phi)^{\text{shininess}}$
- What are the possible values of $\cos \phi$?
- What does the value of *shininess* mean?
- How do we represent shinny or dull surfaces using the Phong model?
- What is the real thing we probably SHOULD do?
- $I_{specular} = k_s I_{light} (\cos \phi)^{shininess} = k_s I_{light} (V.R)^{shininess}$

Effect of the shininess value



How do we compute R?

- N*(N.L)
- R+L=2N(N.L)
- R = 2N(N.L)-L

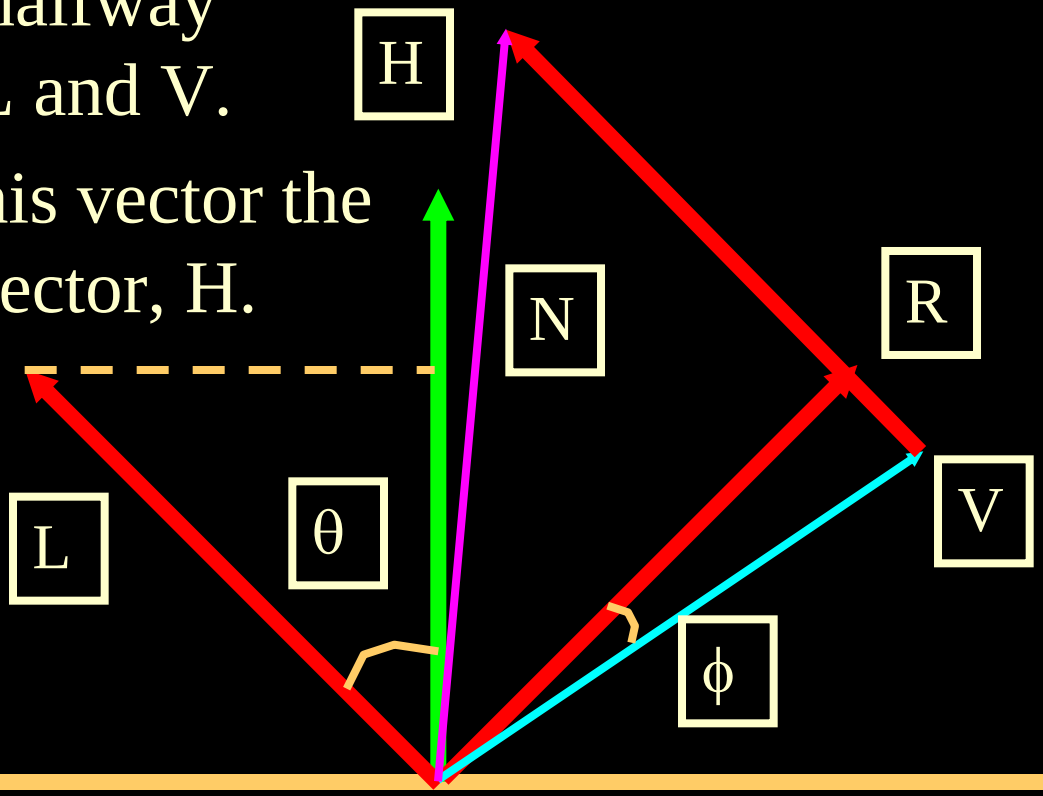


$$H = \frac{L+V}{|L+V|}$$

$$I_{specular} = k_s I_{light\_specularity} (N \cdot H)^{shininess}$$

Simplify this

- Instead of R, we compute halfway between L and V.
- We call this vector the halfway vector, H.



Let's compare the two

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

$$I_{specular} = k_s I_{light\_specularity} (V \cdot R)^{shininess}$$

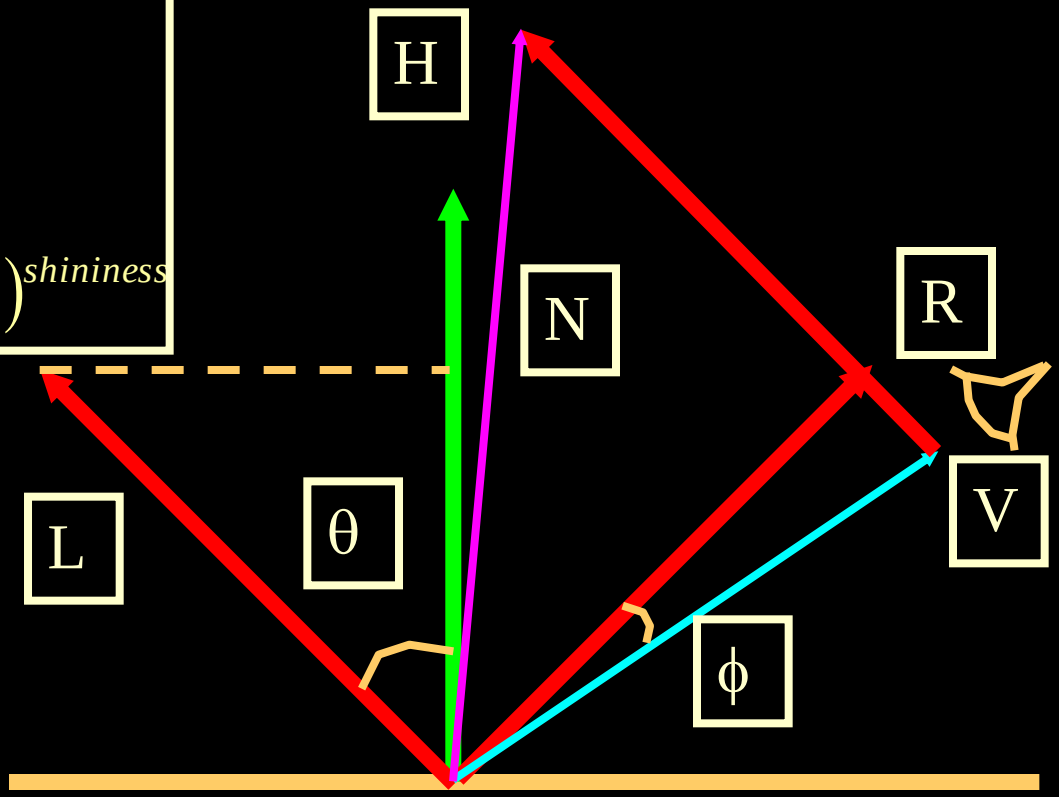
$$H = \frac{L+V}{|L+V|}$$

$$I_{specular} = k_s I_{light\_specularity} (N \cdot H)^{shininess}$$

Q: Which vectors stay constant when viewpoint is far away?

A: V and L vectors -> H

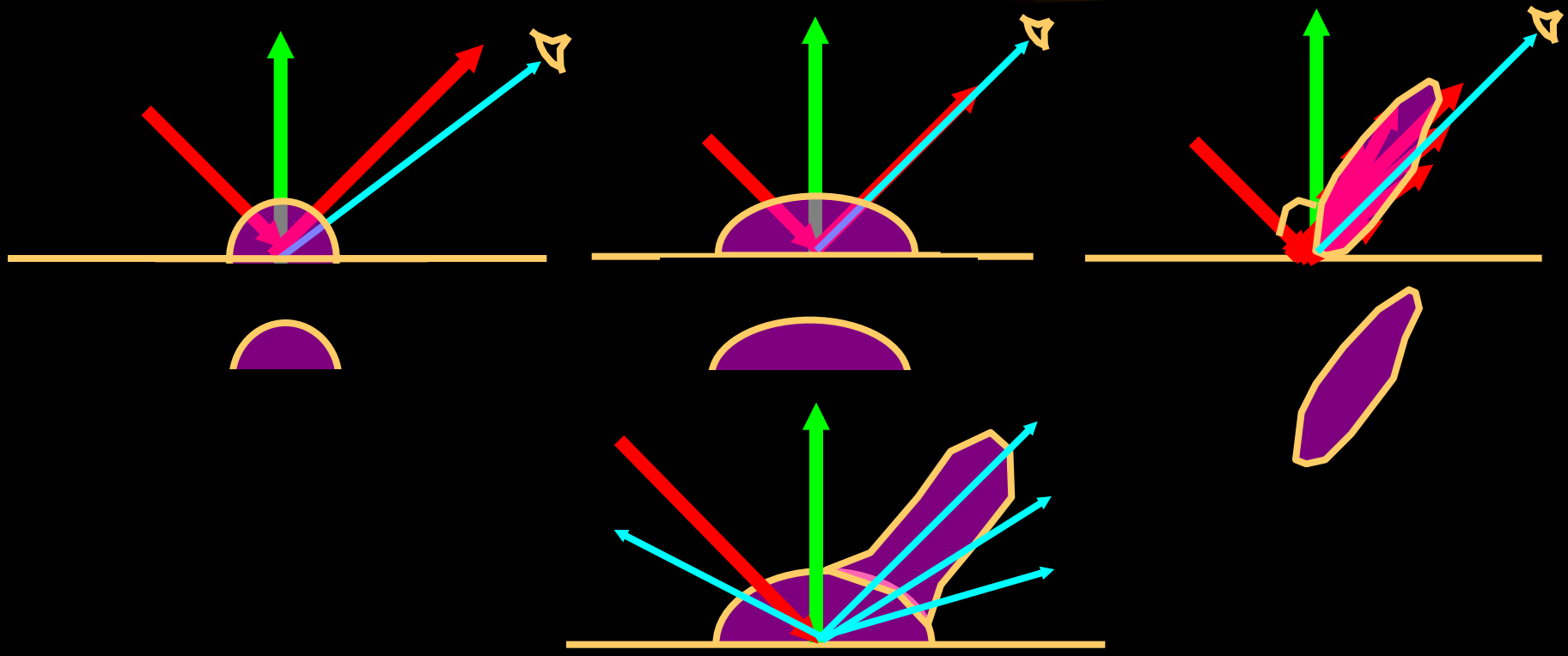
Q: What does this buy us?



Combining the terms

- Ambient - the combination of light reflections from various surfaces to produce a uniform illumination. Background light.
- Diffuse - uniform light scattering of light rays on a surface. Proportional to the “amount of light” that hits the surface. Depends on the surface normal and light vector.
- Sepecular - light that gets reflected. Depends on the light ray, the viewing angle, and the surface normal.

Ambient + Diffuse + Specular



Lighting Equation

$$I_{final} = I_{ambient} k_{ambient} + I_{diffuse} k_{diffuse} (N \cdot L) + I_{specular} k_{specular} (N \cdot H)^{shininess}$$

$$I_{final} = \sum_{l=0}^{lights-1} I_{l_{ambient}} k_{ambient} + I_{l_{diffuse}} k_{diffuse} (N \cdot L) + I_{l_{specular}} k_{specular} (N \cdot H)^{shininess}$$

$I_{lambient}$ = light source *l*'s ambient component

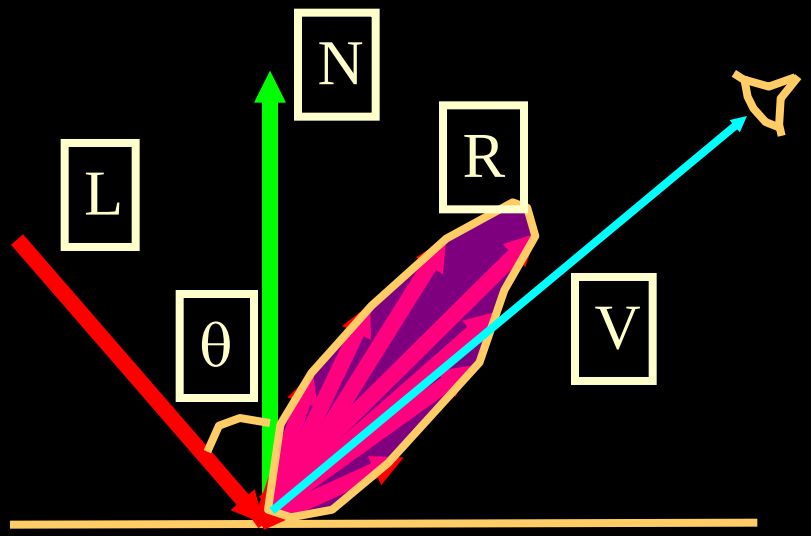
$I_{ldiffuse}$ = light source *l*'s diffuse component

$I_{lspecular}$ = light source *l*'s specular component

$k_{ambient}$ = surface material ambient reflectivity

$k_{diffuse}$ = surface material diffuse reflectivity

$k_{specular}$ = surface material specular reflectivity



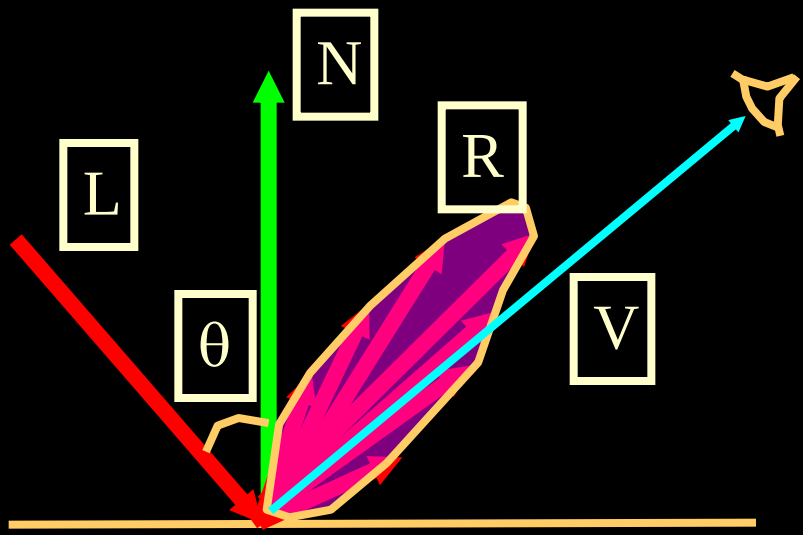
shininess = specular reflection parameter (1 -> dull, 100+ -> very shiny)

Clamping & Spotlights

$$I_{final} = I_{ambient} k_{ambient} + I_{diffuse} k_{diffuse} (N \cdot L) + I_{specular} k_{specular} (N \cdot H)^{shininess}$$

$$I_{final} = \sum_{l=0}^{lights-1} I_{l_{ambient}} k_{ambient} + I_{l_{diffuse}} k_{diffuse} (N \cdot L) + I_{l_{specular}} k_{specular} (N \cdot H)^{shininess}$$

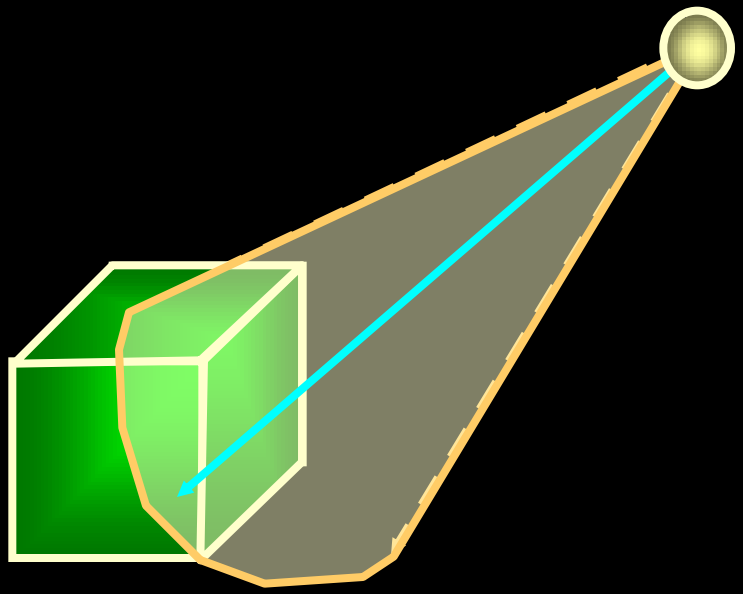
- What does the value I_{final} mean?
- How do we make sure it doesn't get too high?
- Spotlights? How do them?

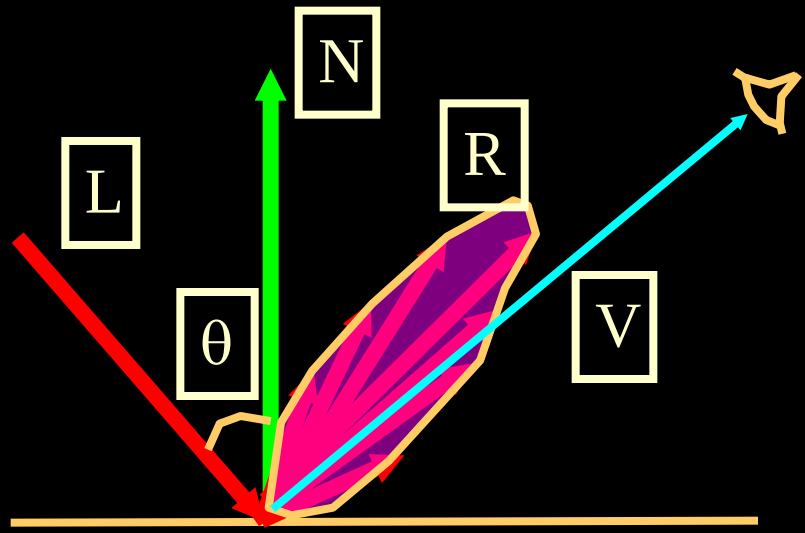


How would we light a green cube?

$$I_{final} = I_{ambient} k_{ambient} + I_{diffuse} k_{diffuse} (N \cdot L) + I_{specular} k_{specular} (N \cdot H)^{shininess}$$

$$I_{final} = \sum_{l=0}^{lights-1} I_{l_{ambient}} k_{ambient} + I_{l_{diffuse}} k_{diffuse} (N \cdot L) + I_{l_{specular}} k_{specular} (N \cdot H)^{shininess}$$





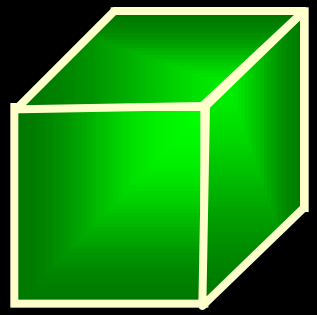
Attenuation

- One factor we have yet to take into account is that a light source contributes a higher incident intensity to closer surfaces.
- The energy from a point light source falls off proportional to $1/d^2$.
- What happens if we *don't* do this?

What would attenuation do for:

- Actually, using *only* $1/d^2$, makes it difficult to correctly light things. Think if d=1 and d=2. Why?
- Remember, we are approximating things. Lighting model is too simple AND most lights are not point sources.
- We use:

$$f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$$



Subtleties

- What's wrong with:

$$f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$$

What's a good fix?

$$f(d) = \min\left(1, \frac{1}{a_0 + a_1 d + a_2 d^2}\right)$$

Full Illumination Model

$$I_{final} = I_{l_{ambient}} k_{ambient} + \sum_{l=0}^{lights-1} f(d_l)\left[I_{l_{diffuse}} k_{diffuse}(N \cdot L) + I_{l_{specular}} k_{specular}(N \cdot H)^{shininess}\right]$$

$$f(d) = \min\left(1, \frac{1}{a_0 + a_1 d + a_2 d^2}\right)$$

Run demo

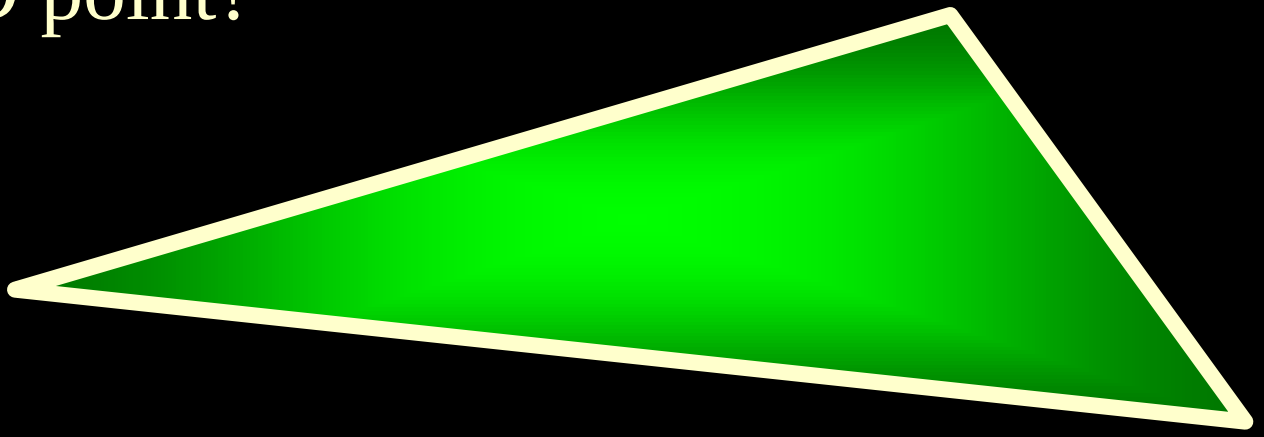
Putting Lights in OpenGL

- 1. glEnable(GL_LIGHTING);
- 2. Set up Light properties
  - glLightf(…)
- 3. Set up Material properties
  - glMaterial(…)

Shading

$$I_{final} = I_{l_{ambient}} k_{ambient} + \sum_{l=0}^{lights-1} f(d_l)\left[I_{l_{diffuse}} k_{diffuse}(N \cdot L) + I_{l_{specular}} k_{specular}(N \cdot H)^{shininess}\right]$$

- When do we do the lighting equation?
- What is the cost to compute the lighting for a 3D point?



Shading

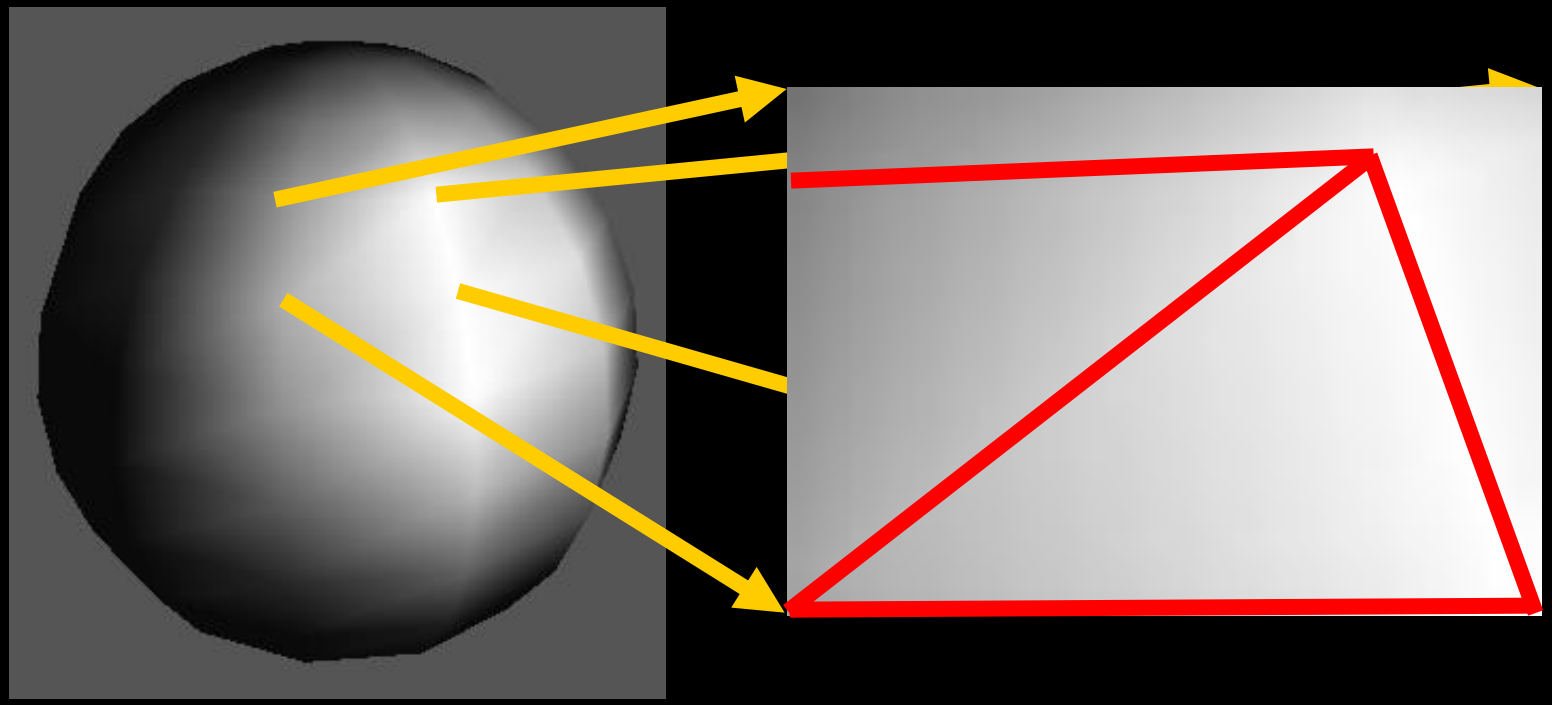
- Shading is how we “color” a triangle.
- Constant Shading
- Gouraud Shading
- Phong Shading

Constant Shading

- Constant Intensity or Flat Shading
- One color for the entire triangle
- Fast
- Good for some objects
- What happens if triangles are small?
- Sudden intensity changes at borders

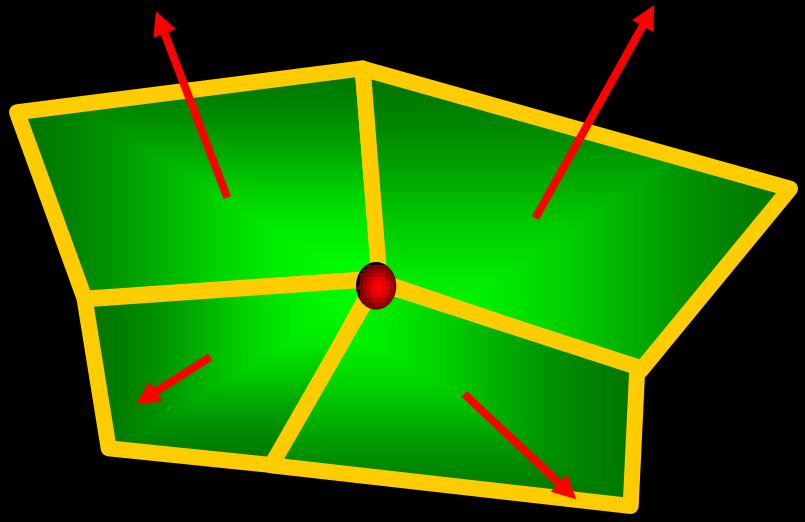
Gouraud Shading

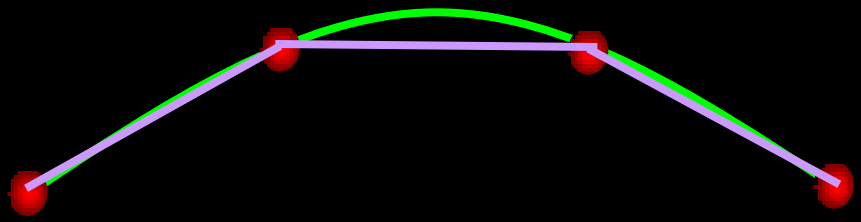
- Intensity Interpolation Shading
- Calculate lighting at the vertices. Then interpolate the colors as you scan convert



Gouraud Shading

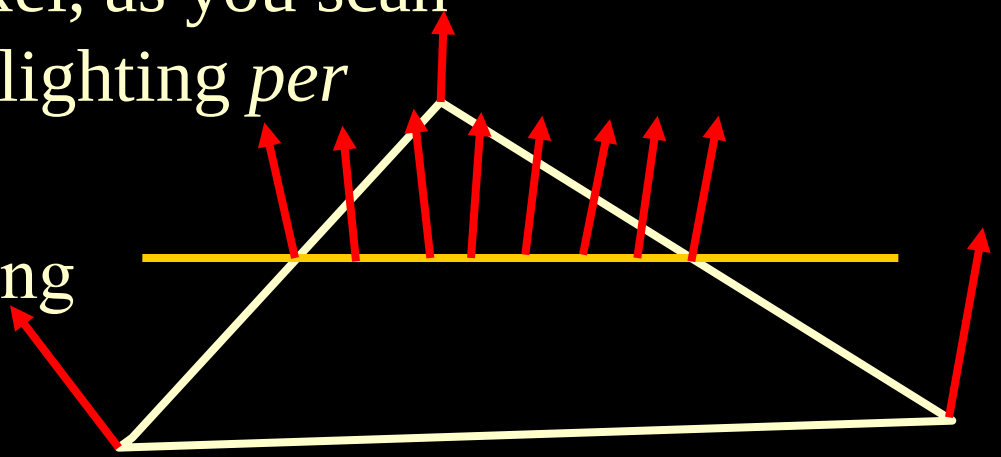
- Relatively fast, only do three calculations
- No sudden intensity changes
- What can it not do?
- What are some approaches to fix this?
- Question, what is the normal at a vertex?





Phong Shading

- Interpolate the normal, since that is the information that represents the "curvature"
- Linearly interpolate the vertex normals. For ***each*** pixel, as you scan convert, *calculate* the lighting *per pixel*.
- True "per pixel" lighting
- Not done by most hardware/libraries/etc



Shading Techniques

- Constant Shading
  - Calculate one lighting calculation (pick a vertex) per triangle
  - Color the *entire* triangle the same color
- Gouraud Shading
  - Calculate three lighting calculations (the vertices) per triangle
  - Linearly interpolate the colors as you scan convert
- Phong Shading
  - While you scan convert, linearly interpolate the normals.
  - With the interpolated normal at each pixel, calculate the lighting at each pixel