VGP351 – Week 4

- Agenda:
  - Physical theory of light
  - Lighting models for graphics
  - Shading models for graphics
  - Types of lights

Lighting

- Lighting, in graphics, is the art of *approximately* simulating the manner in which light interacts with materials

Lighting

- Lighting, in graphics, is the art of *approximately* simulating the manner in which light interacts with materials
- Remember:

  "Light makes right."

  – Andrew Glassner

  "If it looks good, it is good."

  – Michael Abrash

Lighting

- Two fundamental theories of how light works
  - Wave theory of light – Christiaan Huygens proposed in 1690 that light is emitted in all directions as a series of waves

Double-Slit Experiment

- Thomas Young's 1801 double-slit experiment supports the wave theory
  - Light emitted through two thin slits causes alternating light and dark bands projected on a surface

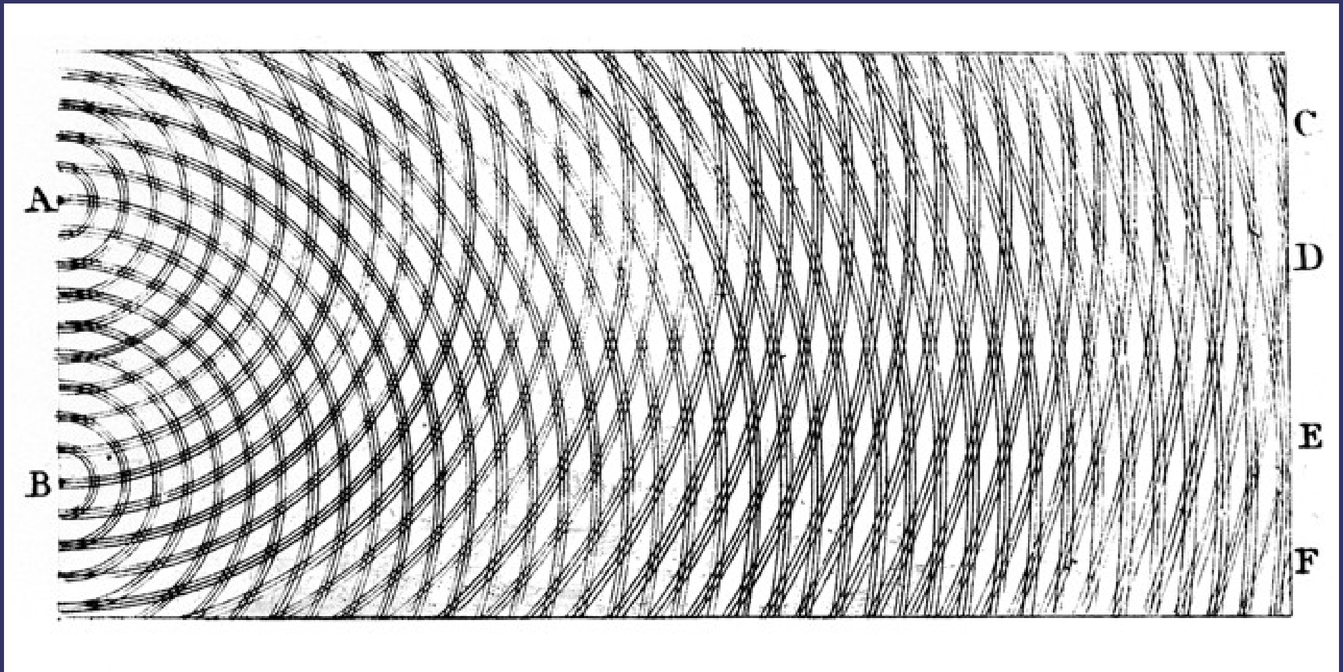


Image from http://en.wikipedia.org/wiki/File:Young_Diffraction.png

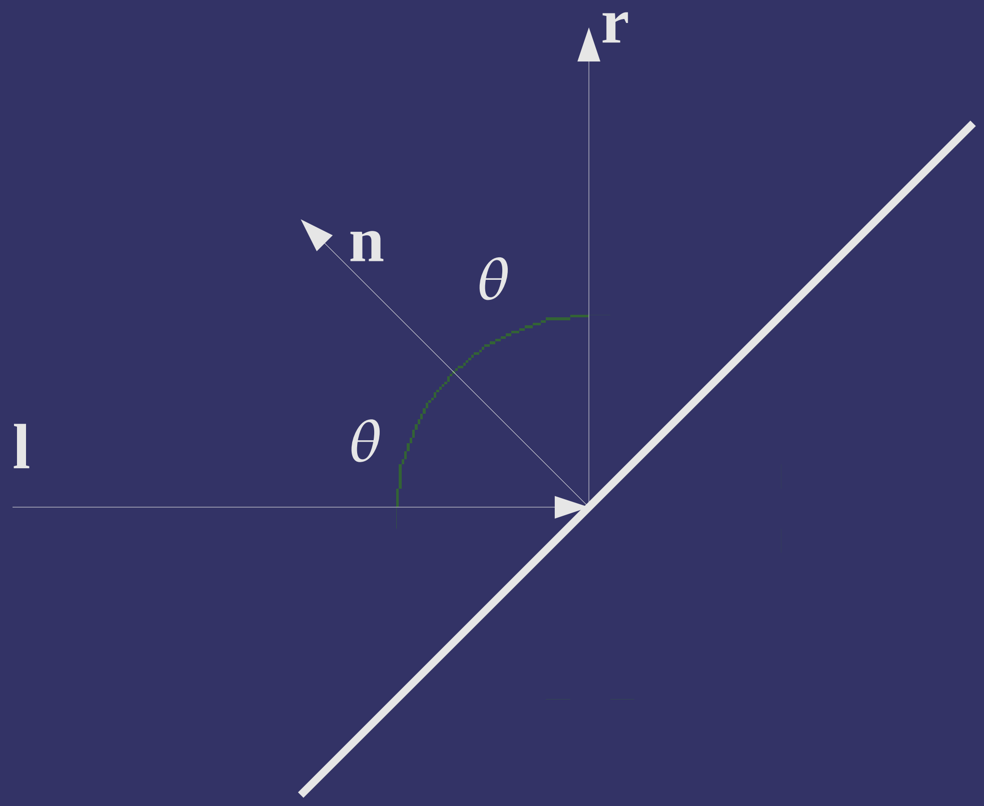
Lighting

- Two fundamental theories of how light works
  - Wave theory of light – Christiaan Huygens proposed in 1690 that light is emitted in all directions as a series of waves
  - Particle theory of light – Ibn al-Haytham proposed in 1021 that light beams are made of minuscule energy particles that travel in a straight line at a fixed speed

Ibn al-Haytham

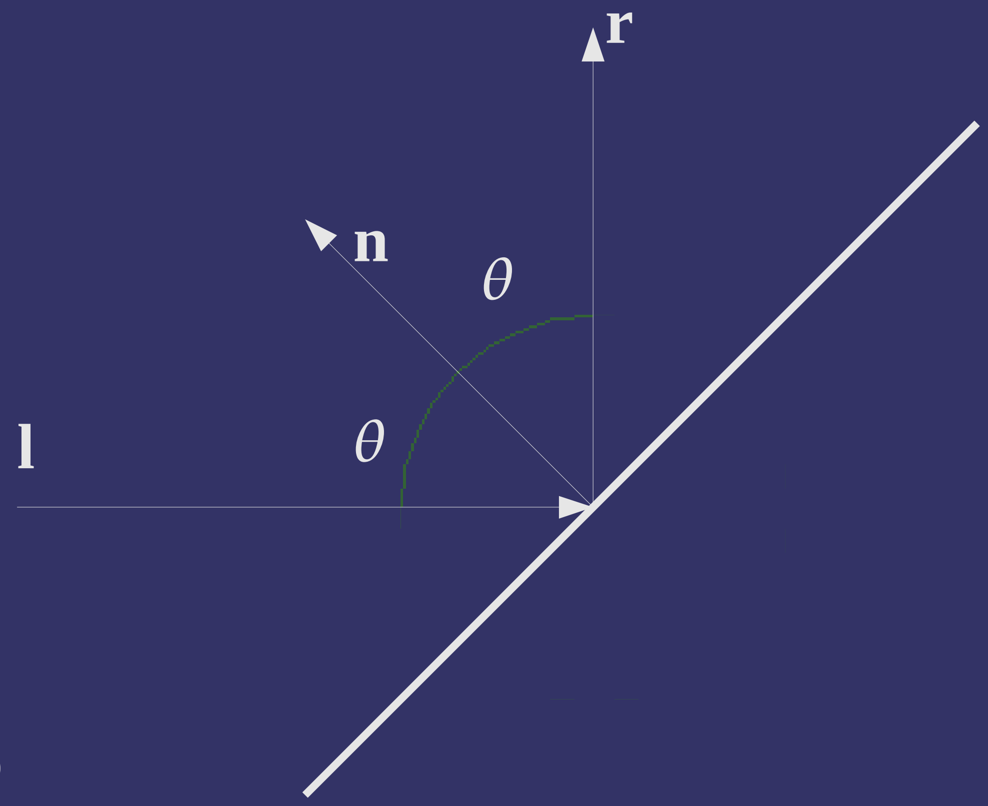
Particle Theory – Reflection

- Particle theory of light correctly predicts reflection



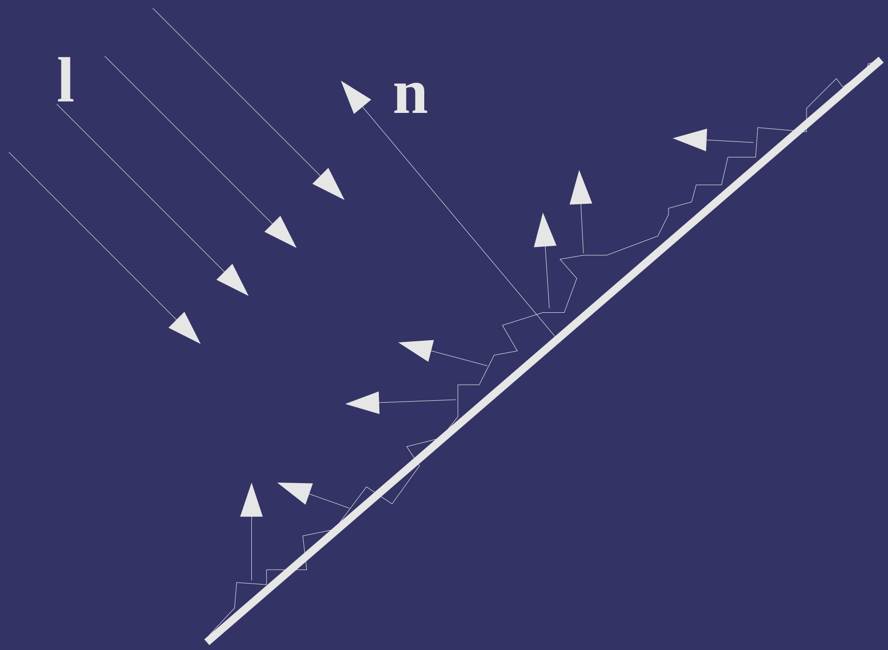
Particle Theory – Reflection

- Particle theory of light correctly predicts reflection
  - This perfect, mirror-like reflection is called *specular reflection*



Particle Theory – Reflection

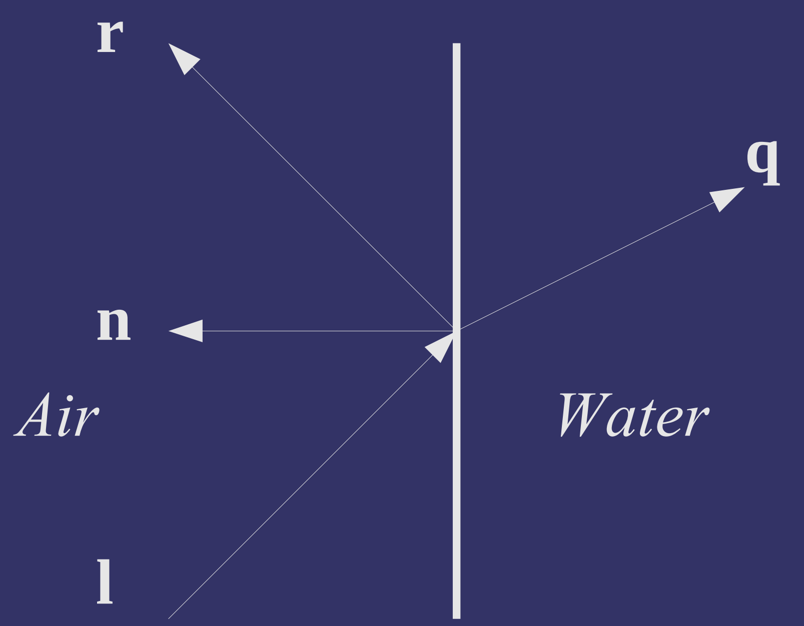
- What about “rough” surfaces?
  - Light rays scatter in all directions
  - This is called *diffuse reflection*



27-April-2010

Wave Theory – Refraction

- When light leaves one material and enters another, it changes direction
  - At the *interface* the speed changes, and the light bends



27-April-2010

Wave Theory – Refraction

Image from http://en.wikipedia.org/wiki/File:Refraction-with-soda-straw.jpg

Lighting

- Two fundamental theories of how light works
  - Wave theory of light – Christiaan Huygens proposed in 1690 that light is emitted in all directions as a series of waves
  - Particle theory of light – Ibn al-Haytham proposed in 1021 that light beams are made of minuscule energy particles that travel in a straight line at a fixed speed

Ibn al-Haytham

- So... which is it?
  - It exhibits *both* characteristics depending on the situation
  - See also http://dir.salon.com/story/comics/tomo/2004/07/06/tomo/

Computer Lighting Models

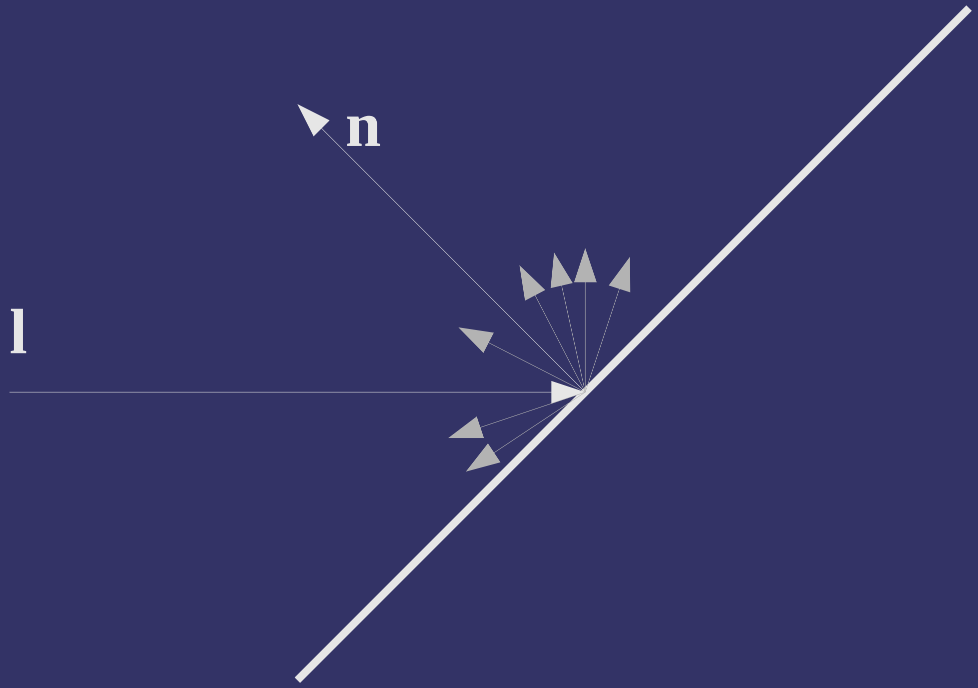
- Every model is a simplification of the physical phenomena
  - We'll look at three *simple* models today:
    - Lambertian reflectance
    - Phong reflection model
    - Blinn-Phong reflection model
  - We'll look at a number of more complex models next term

Lambertian Reflectance

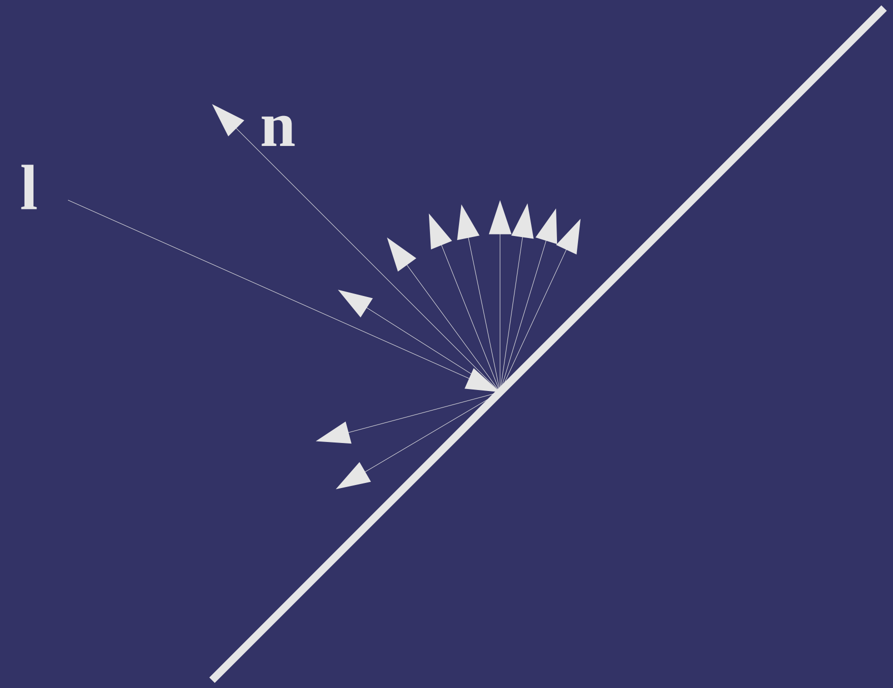
- Reflection from ideal diffuse reflectors obeys Lambert's Cosine Law:

  The radiant intensity reflected is proportional to the cosine between surface normal and the incoming light

Lambertian Reflectance



Lambertian Reflectance



27-April-2010

Lambertian Reflectance

- Reflection from ideal diffuse reflectors obeys Lambert's Cosine Law:

$$\mathbf{i}_d = \frac{\mathbf{l} \cdot \mathbf{n}}{|\mathbf{l}||\mathbf{n}|} * \mathbf{c}_d * \mathbf{l}_d$$

Lambertian Reflectance

- Reflection from ideal diffuse reflectors obeys Lambert's Cosine Law:

$$\mathbf{i}_{\mathrm{d}} = \frac{\mathbf{l} \cdot \mathbf{n}}{|\mathbf{l}||\mathbf{n}|} * \mathbf{c}_{\mathrm{d}} * \mathbf{l}_{\mathrm{d}}$$

Vector <u>from</u> the surface to the light

Intensity of the light

Diffuse color of the surface

Lambertian Reflectance

- Reflection from ideal diffuse reflectors obeys Lambert's Cosine Law:

$$\mathbf{i}_d = \frac{\max(\mathbf{n} \cdot \mathbf{l}, 0)}{|\mathbf{n}||\mathbf{l}|} * \mathbf{c}_d * \mathbf{l}_d$$

Why is this necessary?

Lambertian Reflectance

- Reflection from ideal diffuse reflectors obeys Lambert's Cosine Law:

$$\mathbf{i}_\mathrm{d} = \frac{\max(\mathbf{n} \cdot \mathbf{l}, 0)}{|\mathbf{n}||\mathbf{l}|} * \mathbf{c}_\mathrm{d} * \mathbf{l}_\mathrm{d}$$

Because $\mathbf{n} \cdot \mathbf{l}$ can be negative. Negative light is nonsense!

Lambertian Reflectance

- Reflection from ideal diffuse reflectors obeys Lambert's Cosine Law:

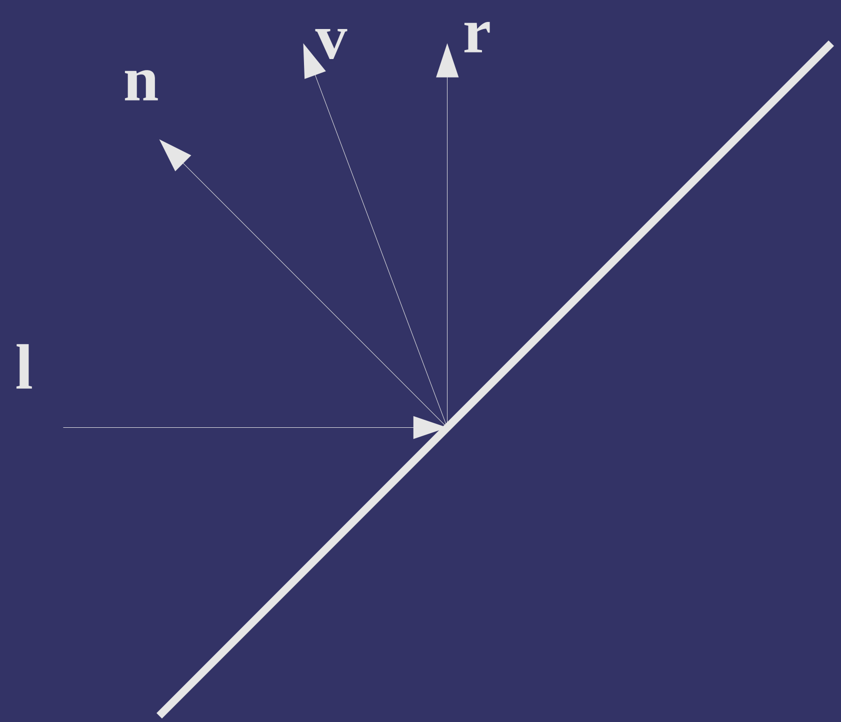
$$\mathbf{i}_d = \frac{\max(\mathbf{n} \cdot \mathbf{l}, 0)}{|\mathbf{n}||\mathbf{l}|} * \mathbf{c}_d * \mathbf{l}_d$$

- Note: viewer is *not* involved in this calculation
  - Hence, diffuse lighting is *view independent*

Phong Reflectance

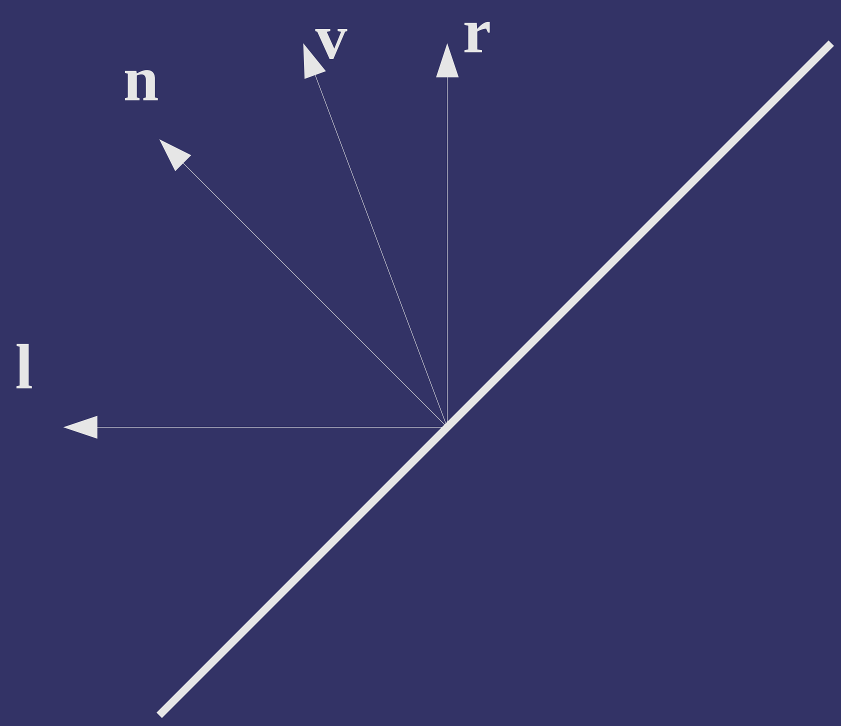
- Adds a mirror-like reflection factor to the diffuse factor

$$\mathbf{i}_s = \left(\frac{\mathbf{r} \cdot \mathbf{v}}{|\mathbf{r}||\mathbf{v}|} \right)^s * \mathbf{c}_s * \mathbf{l}_s$$



Phong Reflectance

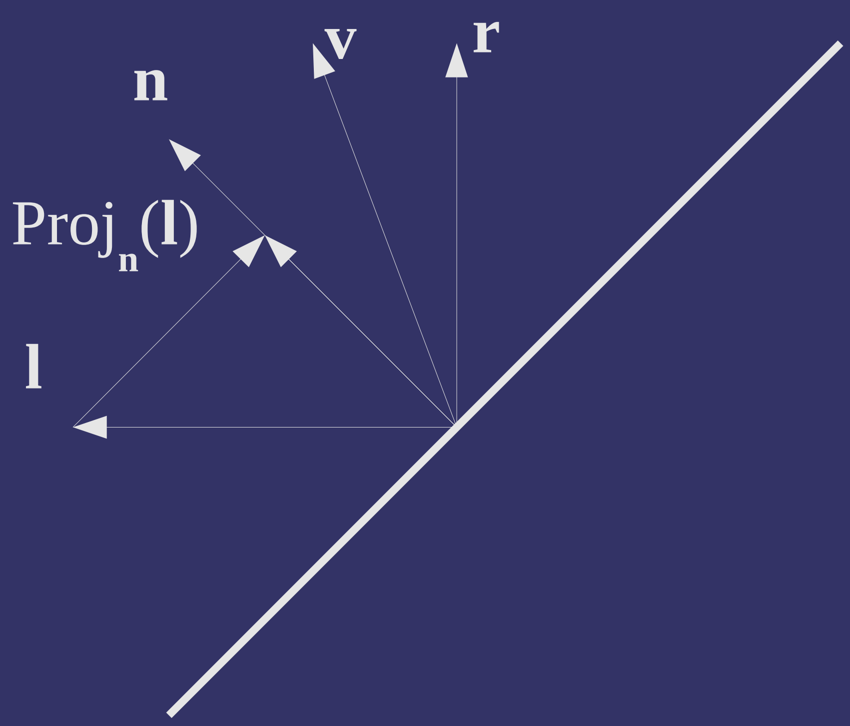
- Adds a mirror-like reflection factor to the diffuse factor
  - **n**, **v**, and **l** are known in advance, but **r** is not...but it can be calculated in a few steps



27-April-2010

Phong Reflectance

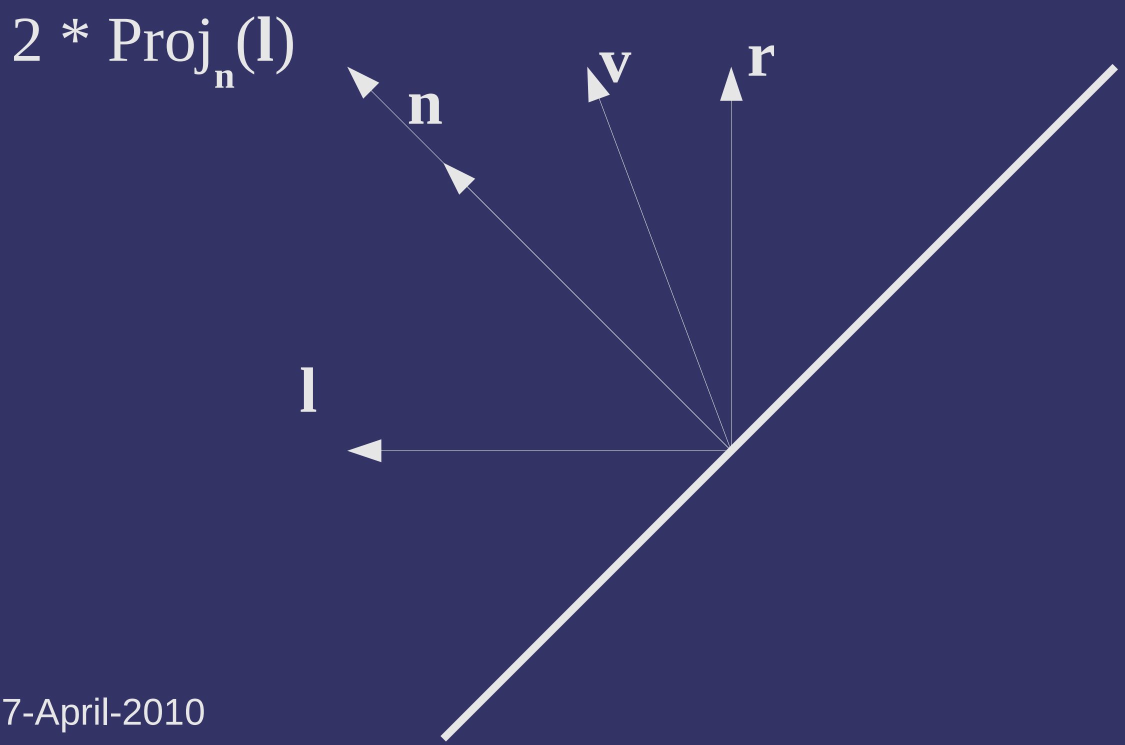
- Adds a mirror-like reflection factor to the diffuse factor
  - **n**, **v**, and **l** are known in advance, but **r** is not...but it can be calculated in a few steps



27-April-2010

Phong Reflectance

- Adds a mirror-like reflection factor to the diffuse factor
  - **n**, **v**, and **l** are known in advance, but **r** is not...but it can be calculated in a few steps

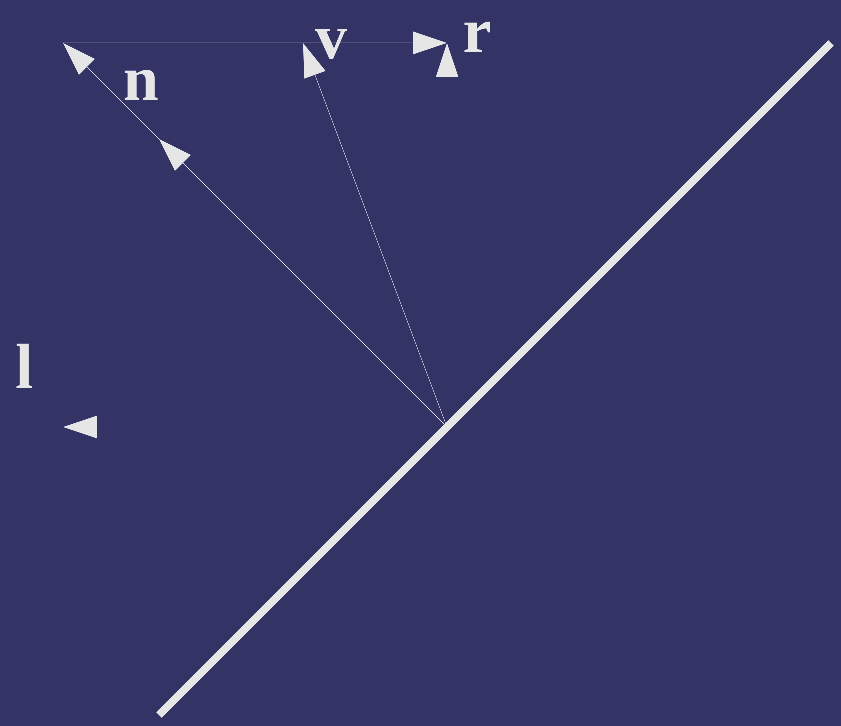


27-April-2010

Phong Reflectance

- Adds a mirror-like reflection factor to the diffuse factor
  - **n**, **v**, and **l** are known in advance, but **r** is not...but it can be calculated in a few steps

$$2 * \text{Proj}_{\mathbf{n}}(\mathbf{l}) - \mathbf{l}$$



Phong Reflectance

- Adds a mirror-like reflection factor to the diffuse factor
  - $\mathbf{n}$, $\mathbf{v}$, and $\mathbf{l}$ are known in advance, but $\mathbf{r}$ is not...but it can be calculated in a few steps

$$\mathbf{r} = \frac{2(\mathbf{n} \cdot \mathbf{l})}{|\mathbf{n}||\mathbf{l}|}\mathbf{n} - \mathbf{l}$$

$$\mathbf{i}_s = \left(\frac{\mathbf{r} \cdot \mathbf{v}}{|\mathbf{r}||\mathbf{v}|}\right)^s * \mathbf{c}_s * \mathbf{l}_s$$

Phong Reflectance

- Adds a mirror-like reflection factor to the diffuse factor
  - $\mathbf{n}$, $\mathbf{v}$, and $\mathbf{l}$ are known in advance, but $\mathbf{r}$ is not...but it can be calculated in a few steps

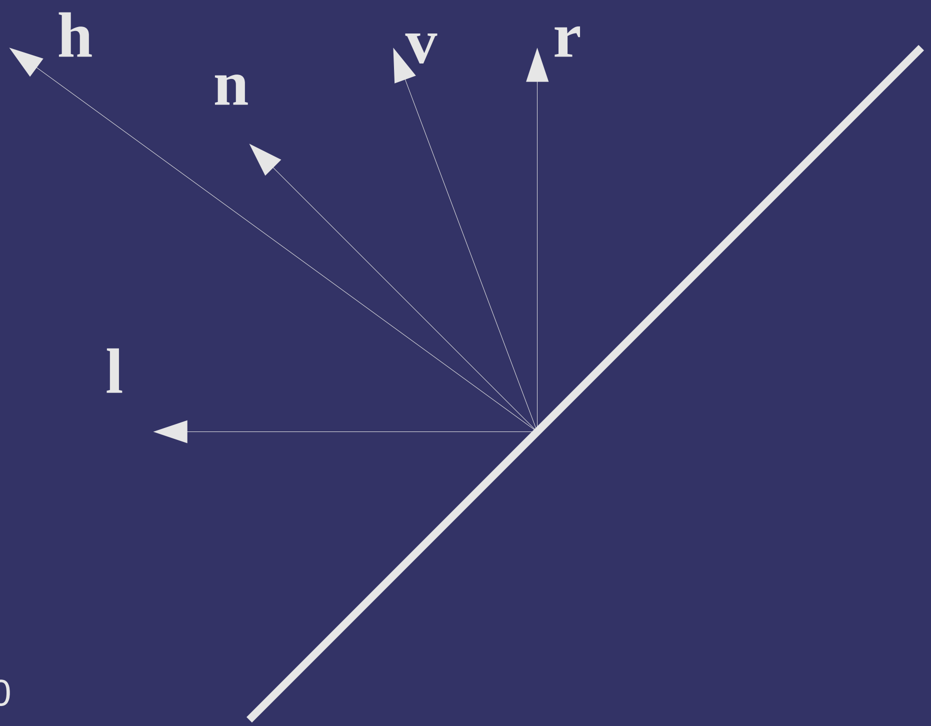
$$\mathbf{r} = \frac{2(\mathbf{n} \cdot \mathbf{l})}{|\mathbf{n}||\mathbf{l}|}\mathbf{n} - \mathbf{l}$$

$$\mathbf{i}_s = \left(\frac{\mathbf{r} \cdot \mathbf{v}}{|\mathbf{r}||\mathbf{v}|}\right)^s * \mathbf{c}_s * \mathbf{l}_s$$

  - This is a lot of math... very expensive to calculate.

Blinn-Phong Reflectance

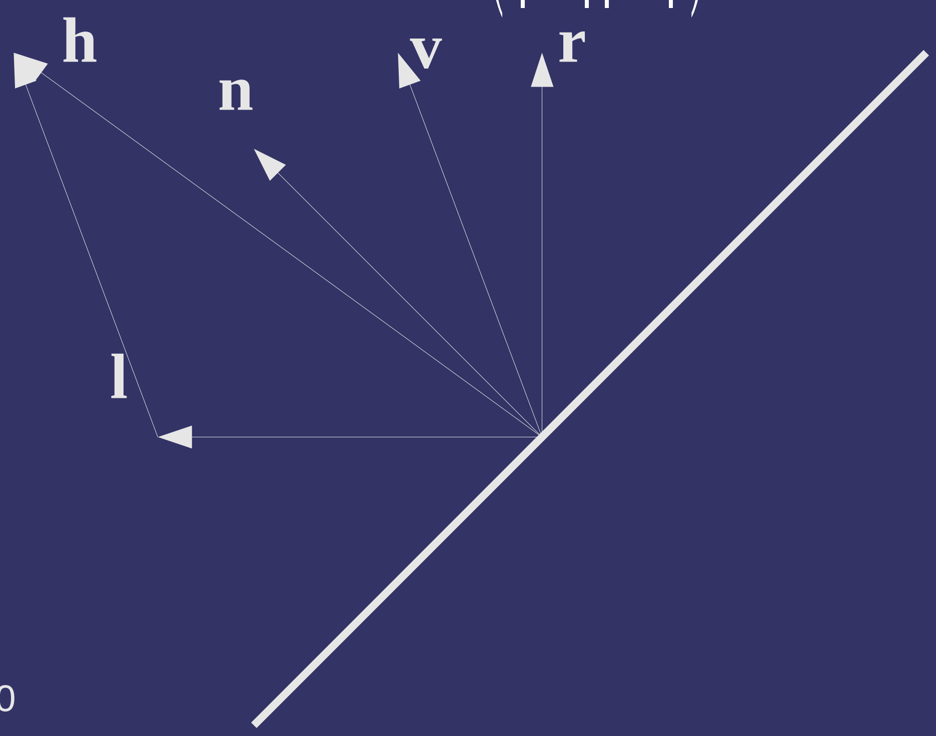
- James Blinn improved Phong's model in 1977
  - Observed that as $\mathbf{v} \cdot \mathbf{r}$ increases, so does $\mathbf{n} \cdot \mathbf{h}$, where $\mathbf{h}$ is a vector half way between $\mathbf{v}$ and $\mathbf{l}$



Blinn-Phong Reflectance

- James Blinn improved Phong's model in 1977
  - Observed that as **v·r** increases, so does **n·h**, where **h** is a vector half way between **v** and **l**

$$\mathbf{h} = \mathbf{l} + \mathbf{v}, \quad \mathbf{i}_s = \left(\frac{\mathbf{n} \cdot \mathbf{h}}{|\mathbf{n}||\mathbf{h}|} \right)^s * \mathbf{c}_s * \mathbf{l}_s$$



27-April-2010

Shininess

- What is the magic s in the exponent of both equations?

$$\mathbf{r} = \frac{2(\mathbf{n} \cdot \mathbf{l})}{|\mathbf{n}||\mathbf{l}|}\mathbf{n} - \mathbf{l}$$

$$\mathbf{i}_s = \left(\frac{\mathbf{r} \cdot \mathbf{v}}{|\mathbf{r}||\mathbf{v}|}\right)^s * \mathbf{c}_s * \mathbf{l}_s$$

$$\mathbf{h} = \mathbf{l} + \mathbf{v}$$

$$\mathbf{i}_s = \left(\frac{\mathbf{n} \cdot \mathbf{h}}{|\mathbf{n}||\mathbf{h}|}\right)^s * \mathbf{c}_s * \mathbf{l}_s$$

Shininess

- What is the magic s in the exponent of both equations?
  - Controls the "size" of the specular highlight
  - As s increases, the highlight gets smaller
    - The dot-product is always less than 1.0, so raising it to some power makes it smaller faster.

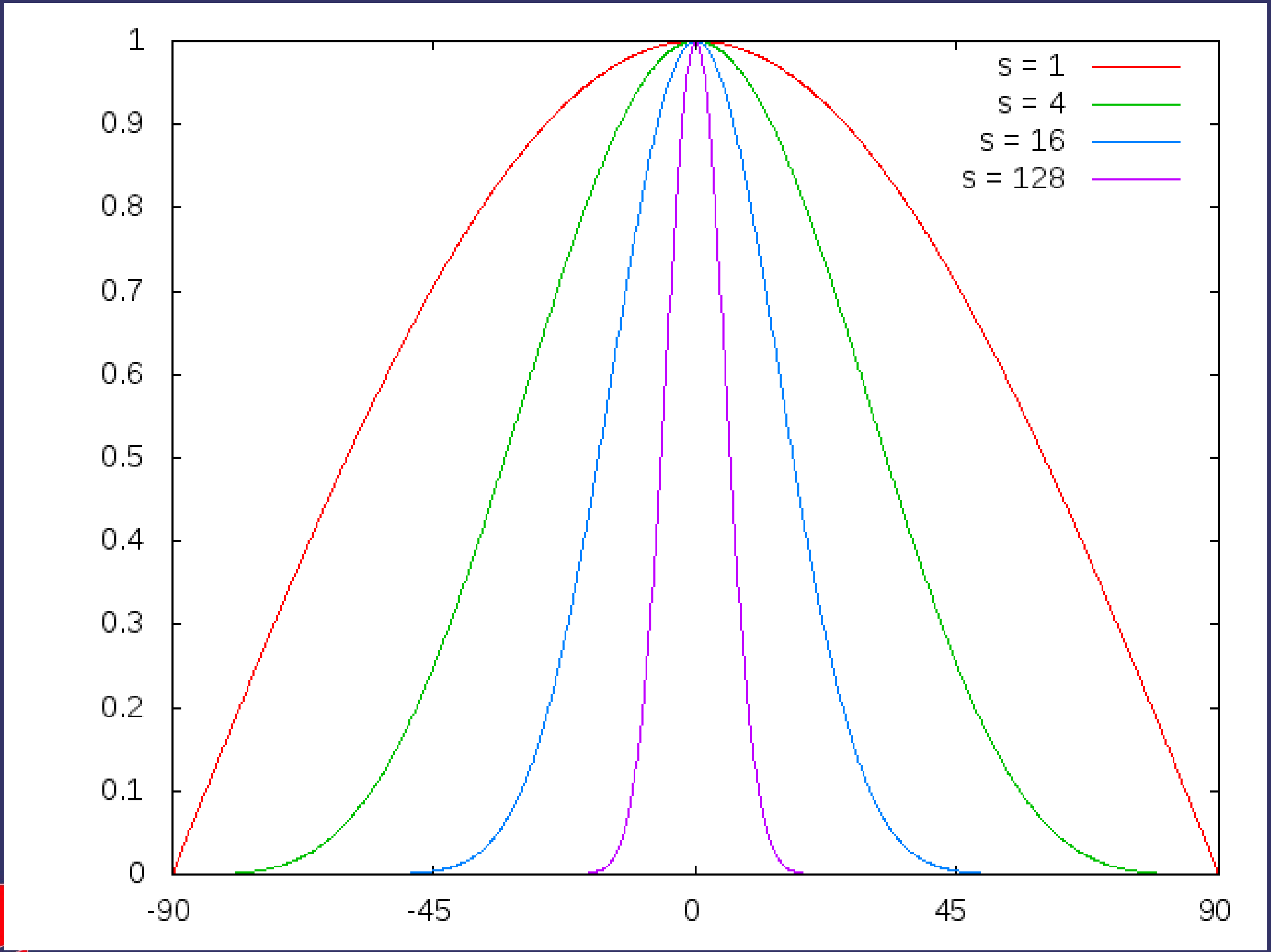
$$\mathbf{r} = \frac{2(\mathbf{n} \cdot \mathbf{l})}{|\mathbf{n}||\mathbf{l}|}\mathbf{n} - \mathbf{l}$$

$$\mathbf{i}_s = \left(\frac{\mathbf{r} \cdot \mathbf{v}}{|\mathbf{r}||\mathbf{v}|}\right)^s * \mathbf{c}_s * \mathbf{l}_s$$

$$\mathbf{h} = \mathbf{l} + \mathbf{v}$$

$$\mathbf{i}_s = \left(\frac{\mathbf{n} \cdot \mathbf{h}}{|\mathbf{n}||\mathbf{h}|}\right)^s * \mathbf{c}_s * \mathbf{l}_s$$

Shininess



Blinn-Phong vs. Phong

- The Blinn-Phong equation is an approximation of the Phong equation
  - Yes... an approximation of an approximation

$$(\mathbf{r} \cdot \mathbf{v})^{s} \approx (\mathbf{n} \cdot \mathbf{h})^{4s}$$

Ambient

- The lighting model so far is a purely *direct lighting* model
  - Most real world light bounces off of other objects, and is call *indirect lighting*
  - We can account for the background, indirect light by adding a simple ambient component

$$\mathbf{i}_a = \mathbf{c}_a * \mathbf{l}_a$$

- *This is the biggest hack of all!*

Shading Models

- We know how to calculate lighting values, but the question remains: how often do we calculate it?

Flat Shading

- Simplest answer: calculate lighting once per polygon
  - Fast!
  - Depending on the circumstances, the quality may be good enough...but usually not

Gouraud Shading

- Calculate lighting once per vertex, interpolate colors across polygon
  - A little slower: more math, have to do interpolation

Gouraud Shading

- Calculate lighting once per vertex, interpolate colors across polygon
  - A little slower: more math, have to do interpolation

For all intents and purposes, this is free.

Gouraud Shading

- Calculate lighting once per vertex, interpolate colors across polygon
  - A little slower: more math, have to do interpolation
  - Looks better
  - Works well for diffuse, but works poorly for specular

Gouraud Shading

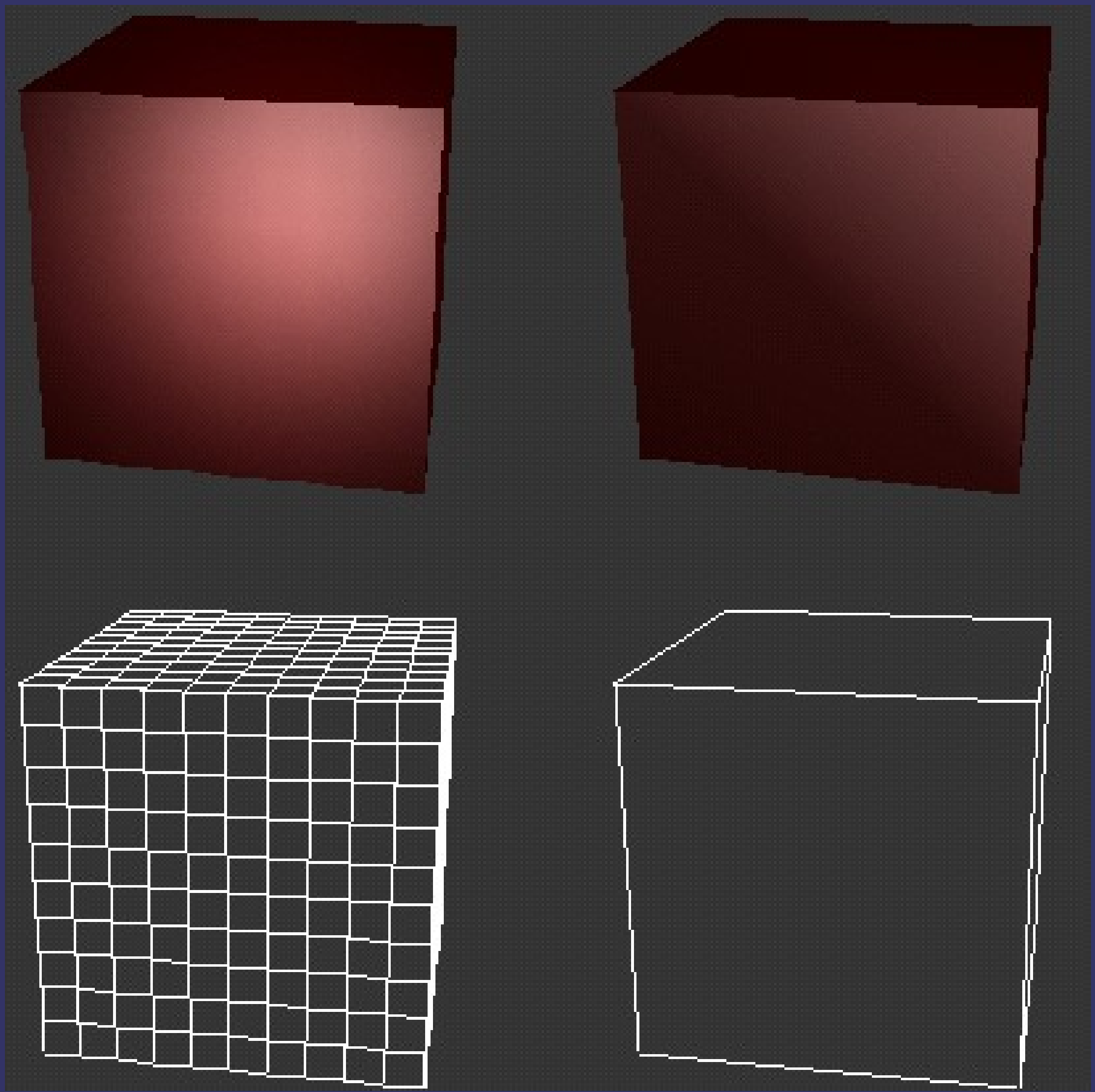
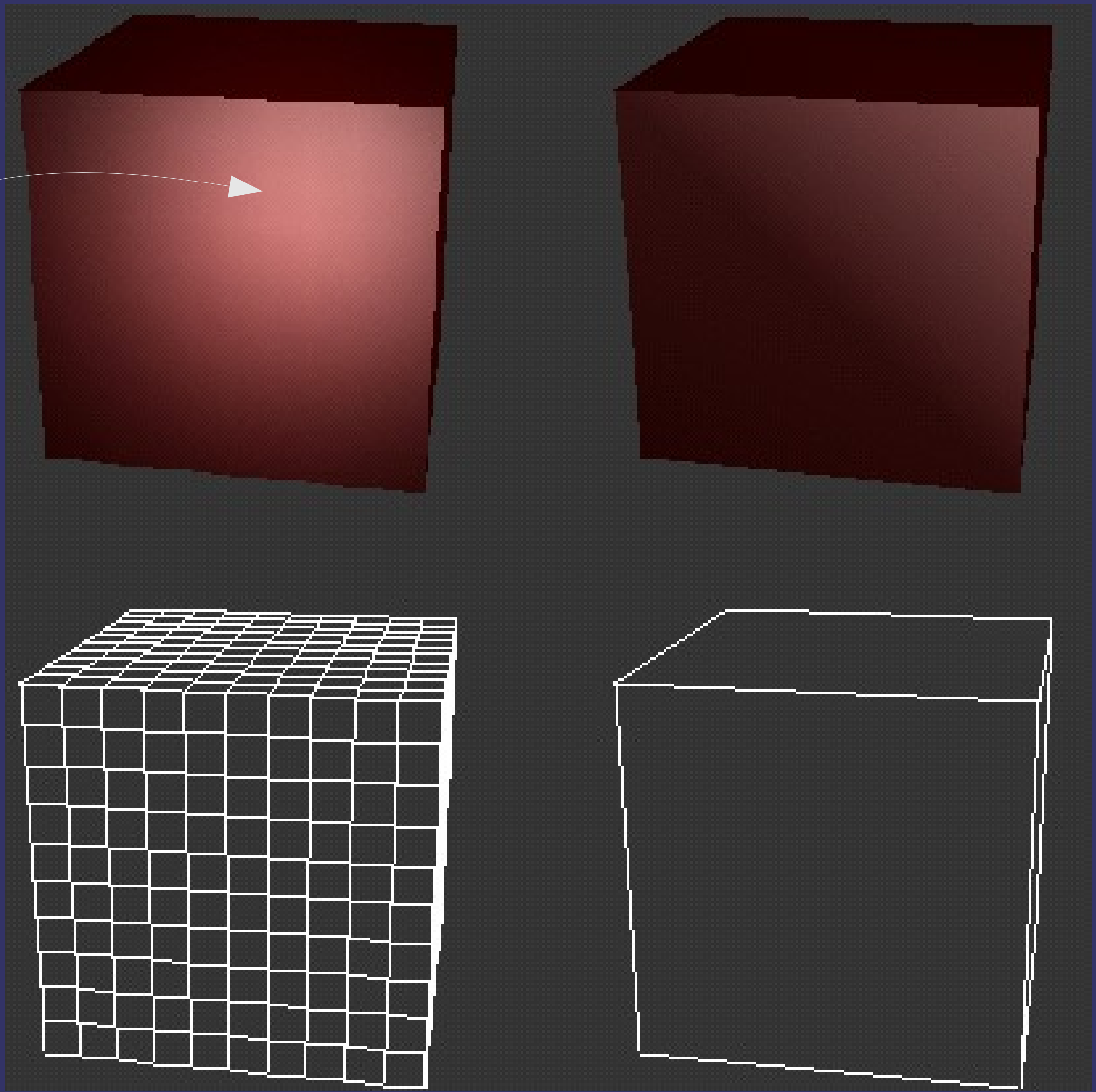


Image from M. Kilgard, “Avoiding 16 Common OpenGL Pitfalls”, 1998.

Gouraud Shading



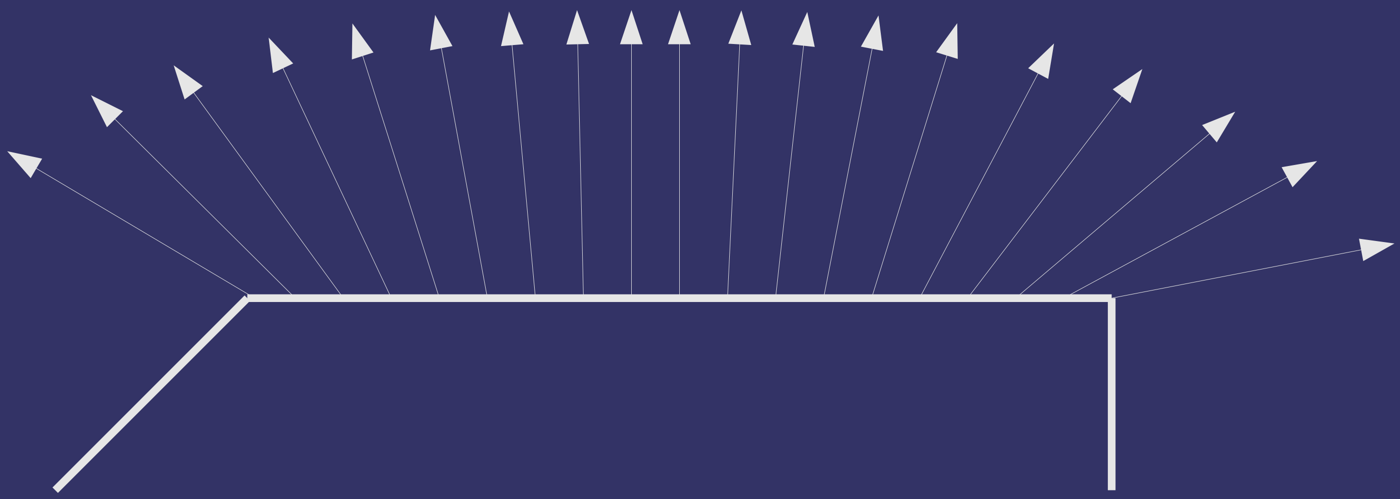
Note the lines at the polygon boundaries. This is called *mach banding*.

Image from M. Kilgard, “Avoiding 16 Common OpenGL Pitfalls”, 1998.

Phong Shading

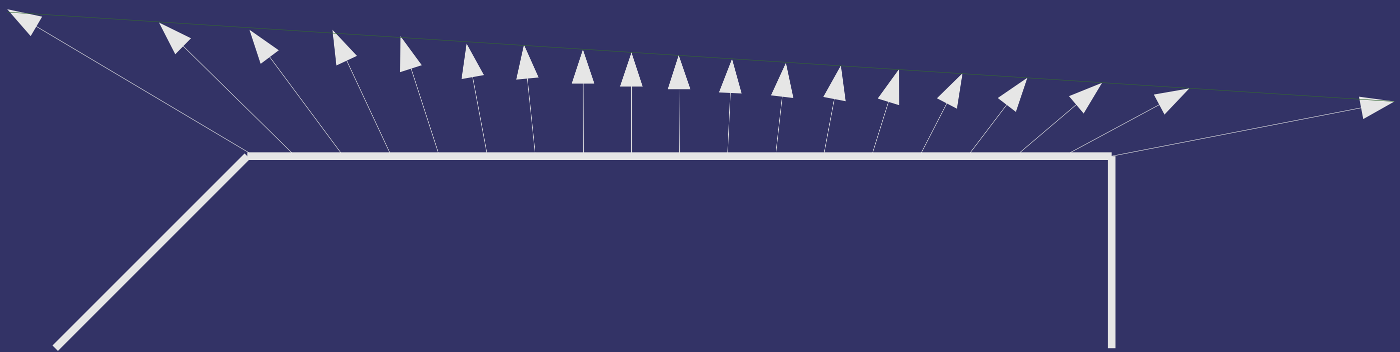
- Next logical step: interpolate lighting parameters, calculate lighting per pixel
  - Looks much better...doesn't miss the specular highlight!
  - Much more expensive to calculate
    - Has really only been practical for real-time rendering for the last couple years
    - Not only requires the lighting to be recalculated per pixel, but interpolated vectors may need to be re-normalized per pixel

Phong Shading



27-April-2010

Phong Shading



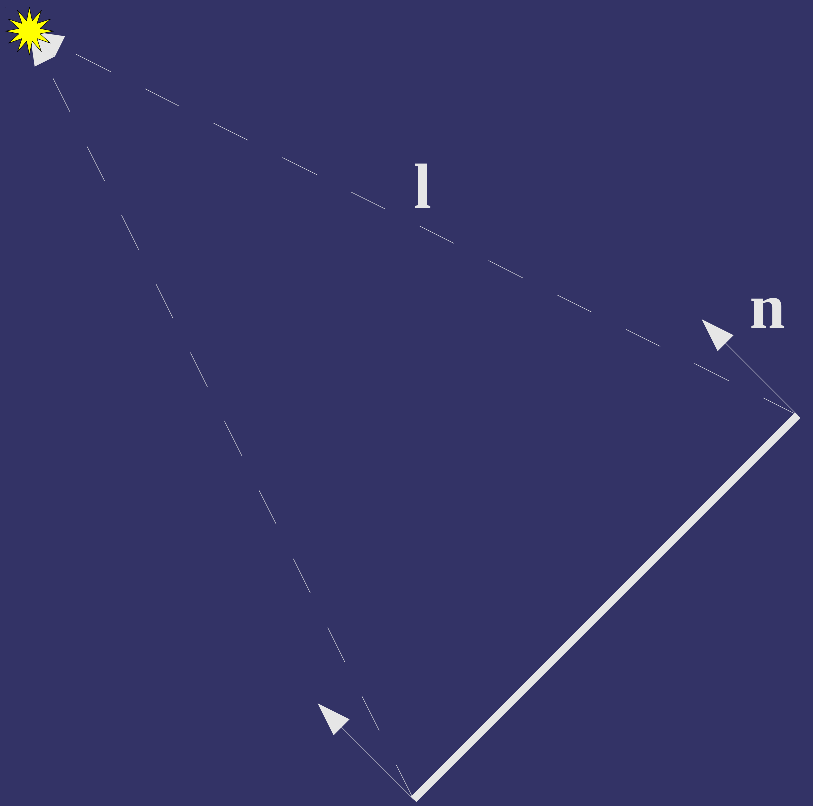
27-April-2010

Types of Lights

- Several common types of lights used in graphics:
  - Point light
  - Directional light
    - Also called infinite light
  - Area lights
  - Spot lights

Point Lights

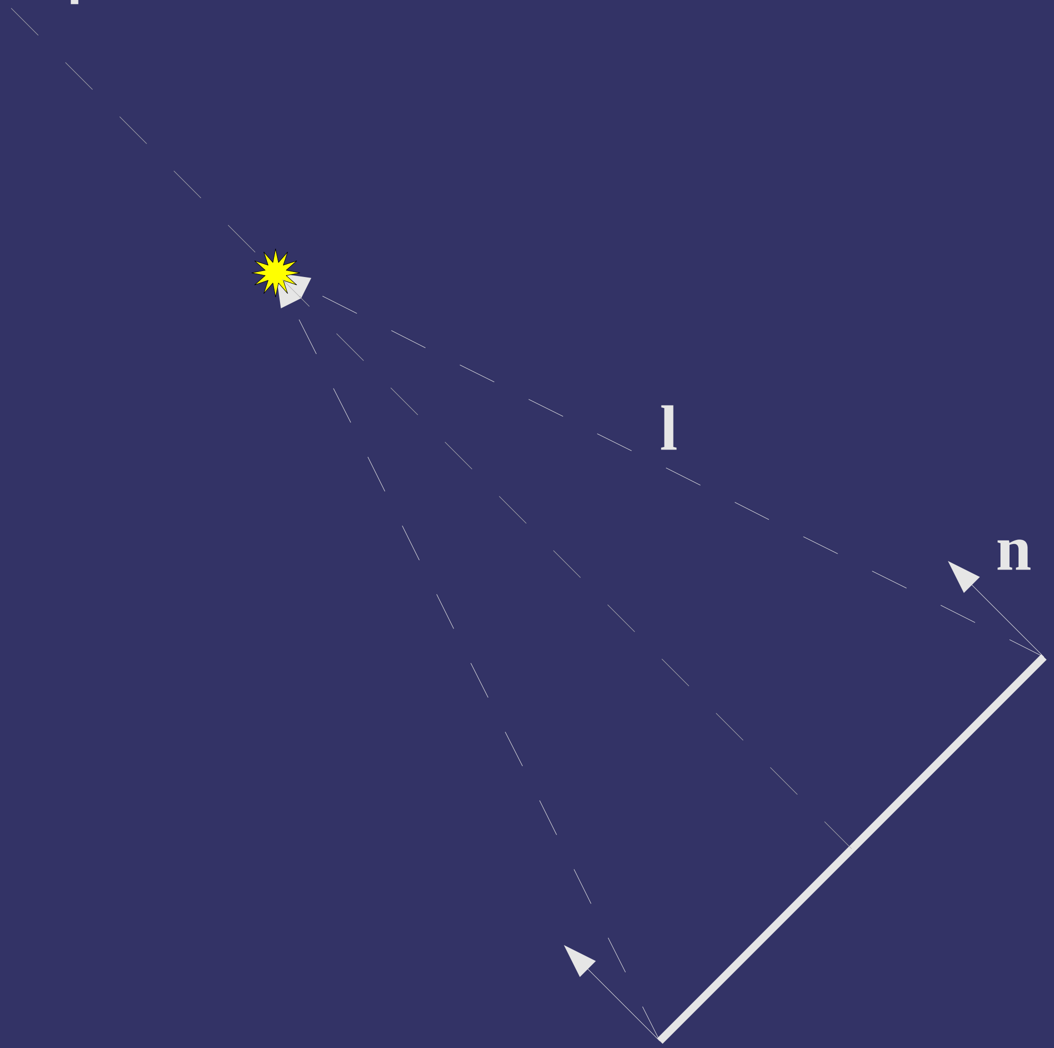
- Calculate the **l** vector by subtracting the vertex position from the light position and normalize the result



27-April-2010

Point Lights

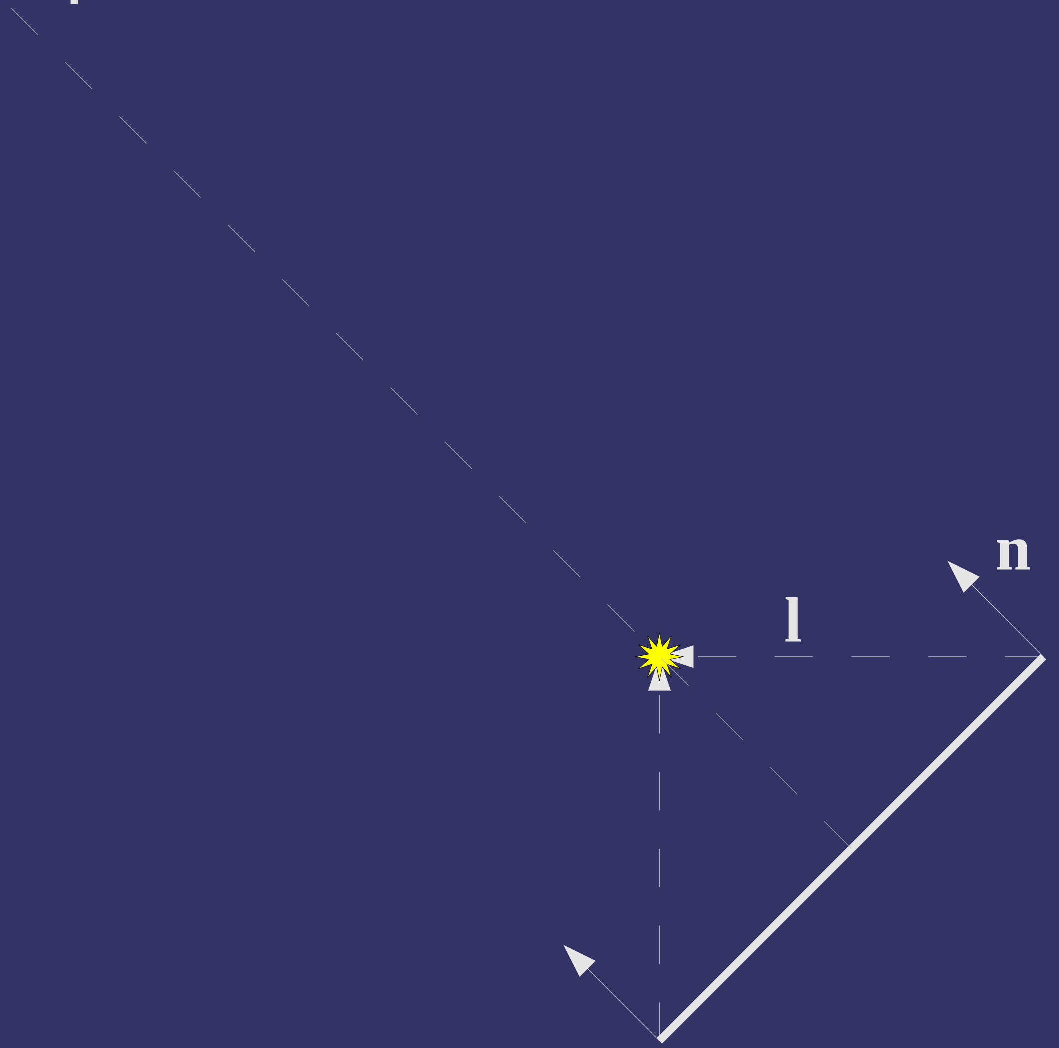
- Calculate the **l** vector by subtracting the vertex position from the light position and normalize the result



27-April-2010

Point Lights

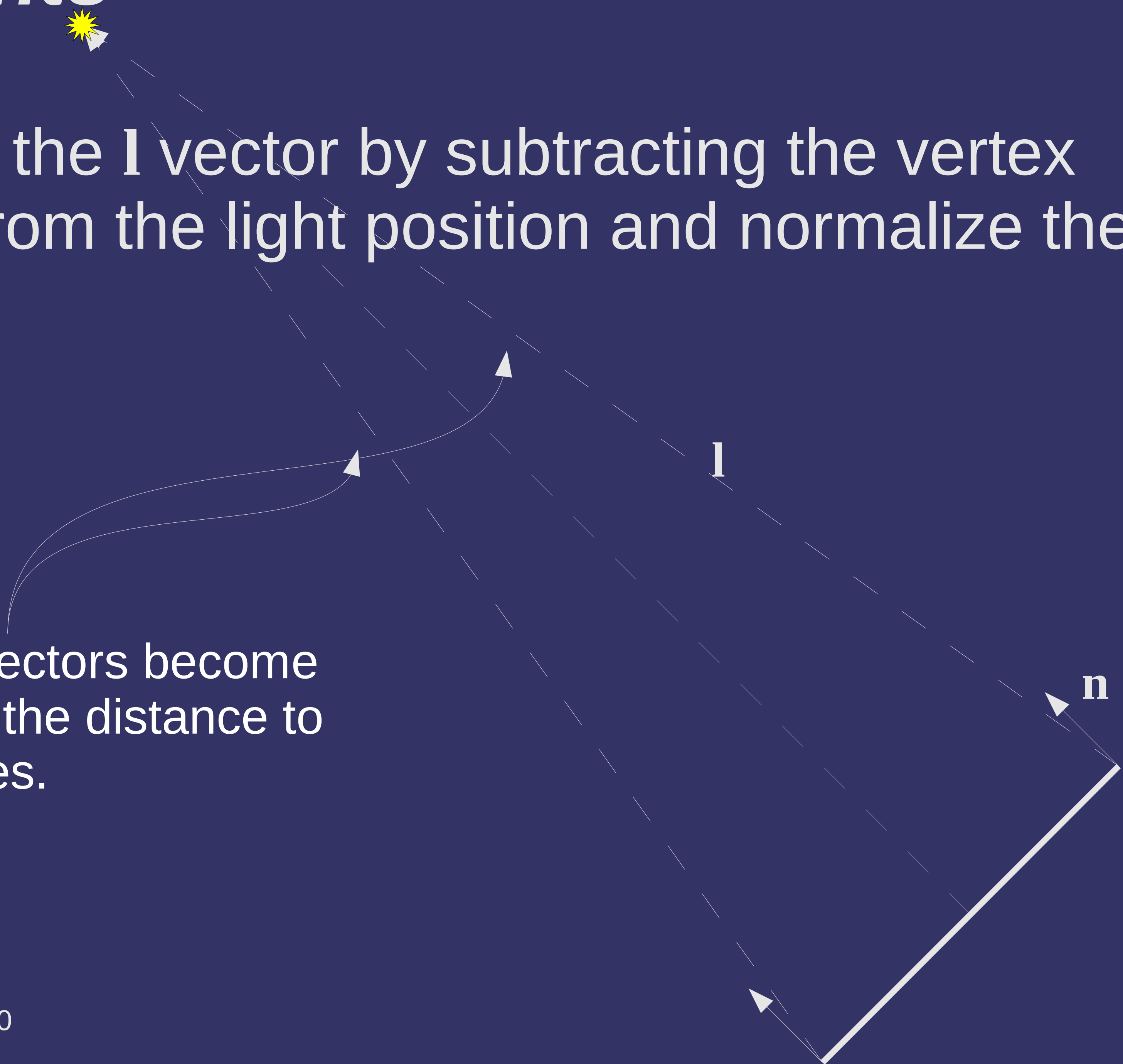
- Calculate the **l** vector by subtracting the vertex position from the light position and normalize the result



27-April-2010

Point Lights

- Calculate the **l** vector by subtracting the vertex position from the light position and normalize the result



Note how the **l** vectors become more parallel as the distance to the light increases.

Directional Lights

- As the light becomes infinitely far away, all of the calculated **l** vectors become parallel
  - When this happens, we can simplify the math and treat the light has *just* a direction
  - Since the direction doesn't change, we don't have to interpolate it
    - Still have to transform it into the space where lighting will be calculated

Area Lights

- Both these models treat lights as infinitesimal points
  - All real lights have some surface area
  - Lights with larger surface areas are considered “softer”
    - This results in shadows with smoother boundaries
    - This is why we have frosted light bulbs and lamp shades instead of bare, clear glass bulbs
  - Techniques exist for handling these sorts of lights, but they are expensive and (currently) impractical for most real-time use
    - We'll discuss this more next term

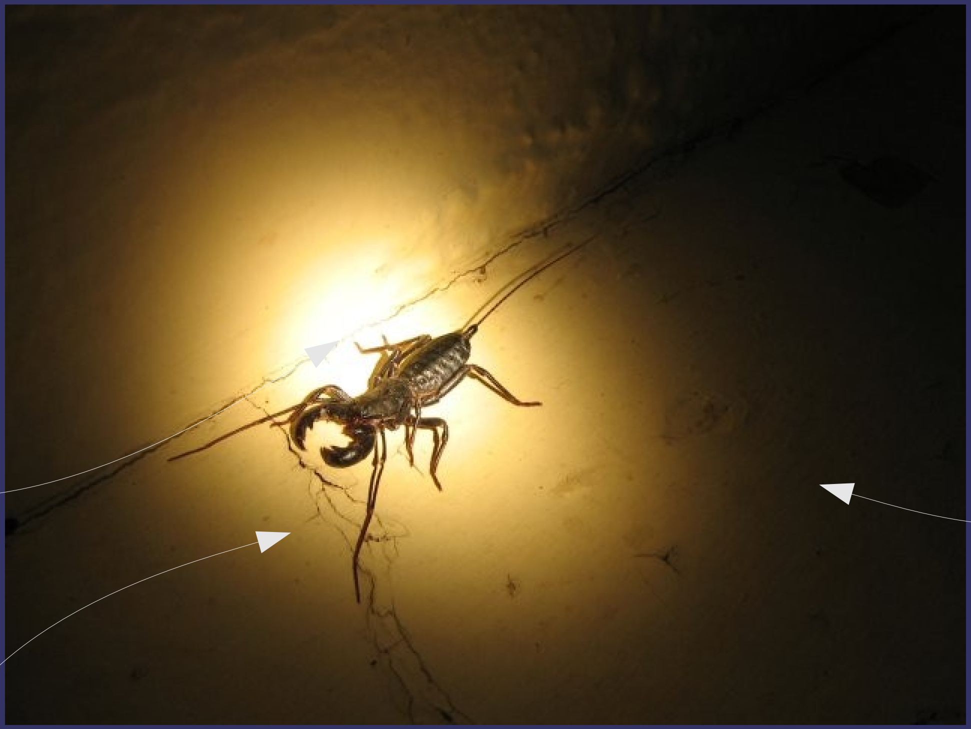
Spot Light

- Most lights don't emit light in all directions
  - Some range over which the light intensity is 100%
  - Some range over which the light intensity gradually decreases
    - This range may be zero
  - Remaining range where no light is emitted

Spot Light

Full intensity light

No light



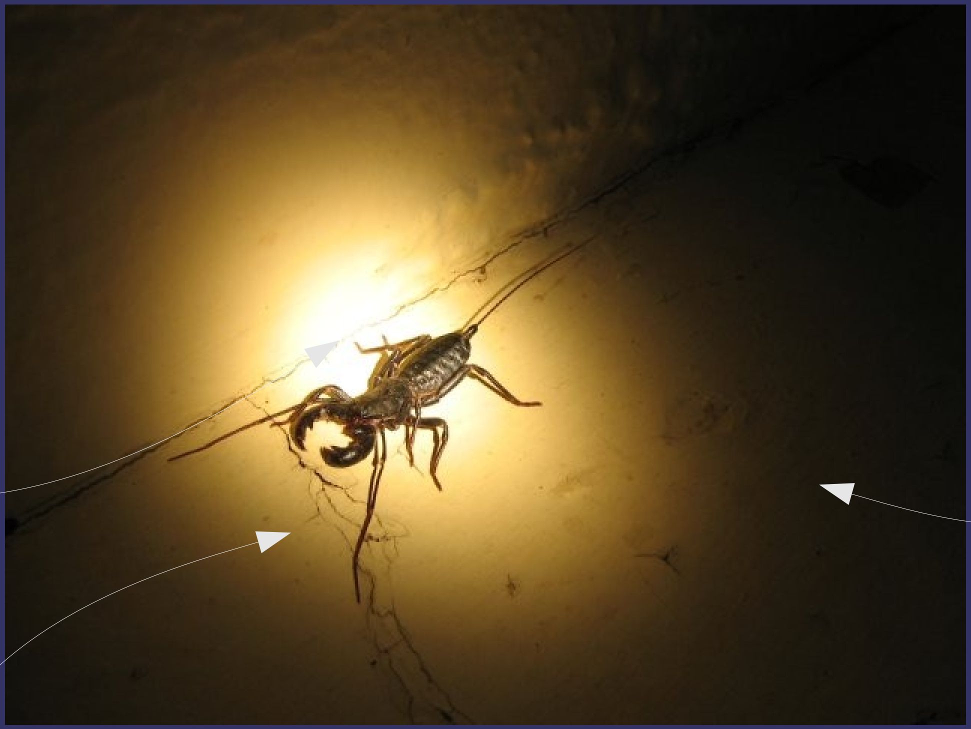
Partial intensity light

Image, by *satanoid*, from http://www.everystockphoto.com/photo.php?imageId=673587

27-April-2010

Spot Light

Full intensity light



~~No light~~
Ambient light

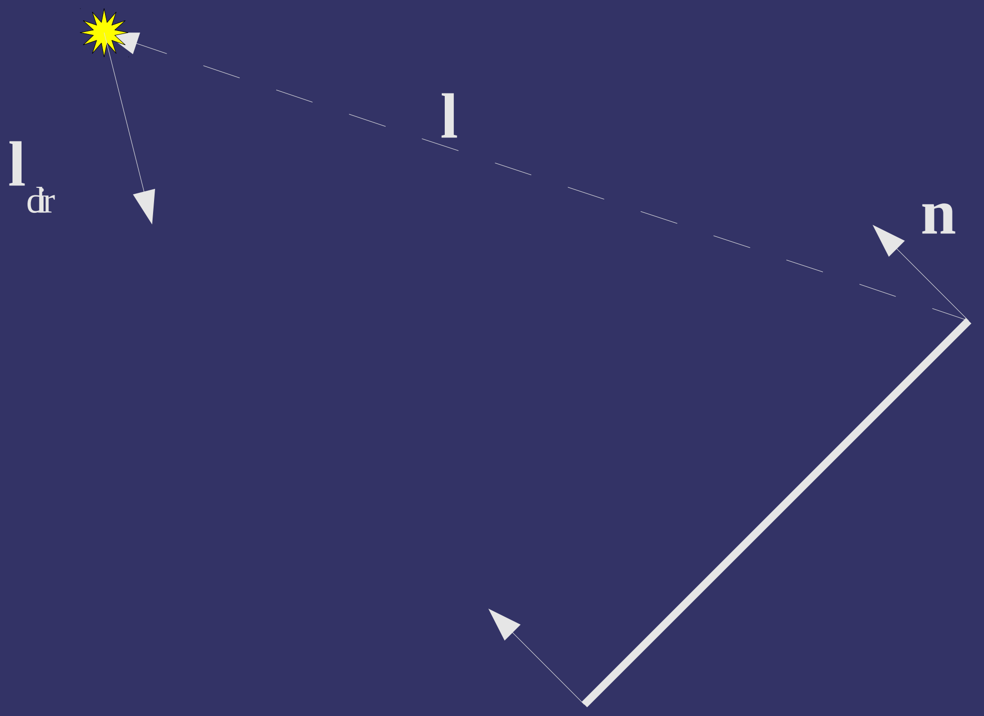
Partial intensity light

Image, by *satanoid*, from http://www.everystockphoto.com/photo.php?imageId=673587

27-April-2010

Spot Light

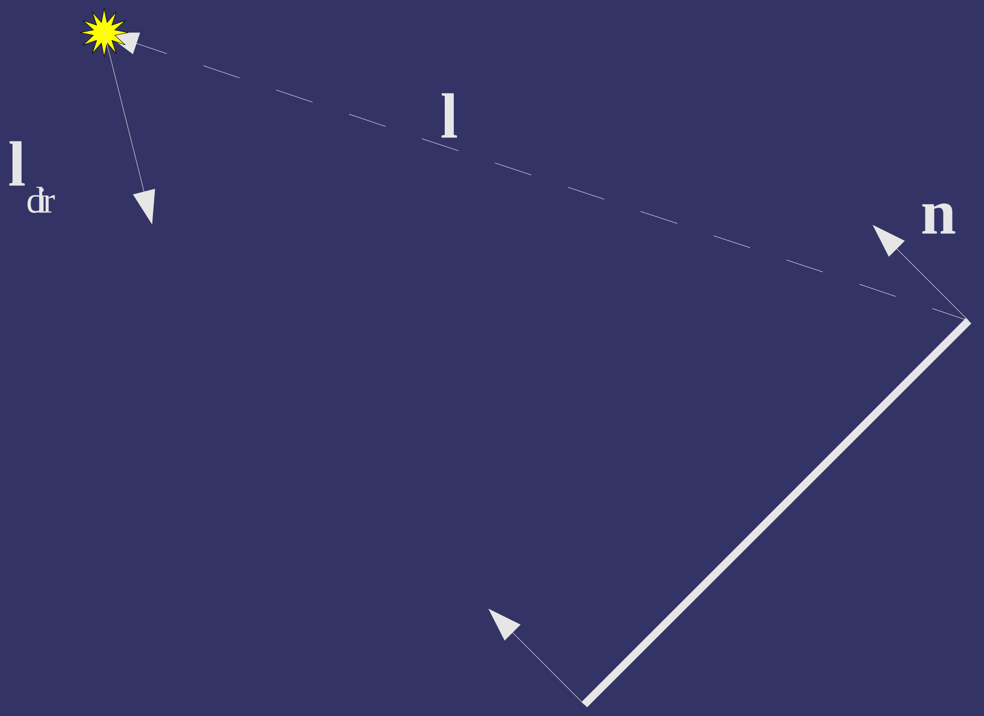
- Need additional light parameters:
  - $\mathbf{l}_{dir}$ – direction the light is pointing
  - l_{att} – Absolute cut-off angle
  - l_{exp} – Exponent for cut-off equation



27-April-2010

Spot Light

$$\mathbf{i} = \begin{cases} \left(\mathbf{l}_{\text{dir}} \cdot -\mathbf{l}\right)^{l_{\text{exp}}} * \mathbf{i}_{\text{L}} & \text{if } \left(\mathbf{l}_{\text{dir}} \cdot -\mathbf{l}\right) > \cos\left(l_{\text{cut}}\right) \\ 0 & \text{otherwise} \end{cases}$$



27-April-2010

Distance Attenuation

- The farther a light is from an object, the less light gets to that object
  - Three separate factors control the attenuation
  - k_c – constant attenuation factor
  - k_l – Linear attenuation factor
  - k_q – Quadratic attenuation factor

$$d = |\mathbf{l}|$$

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

Next week...

- Bounding volumes
  - Bounding spheres
  - Axis-aligned bounding boxes (AABBs)
  - Oriented bounding boxes (OBBs)
  - Heirarchies of BVs
- More occlusion
  - Hierarchical frustum culling

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of Intel or the Art Institute of Portland.

OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.

Khronos and OpenGL ES are trademarks of the Khronos Group.

Other company, product, and service names may be trademarks or service marks of others.