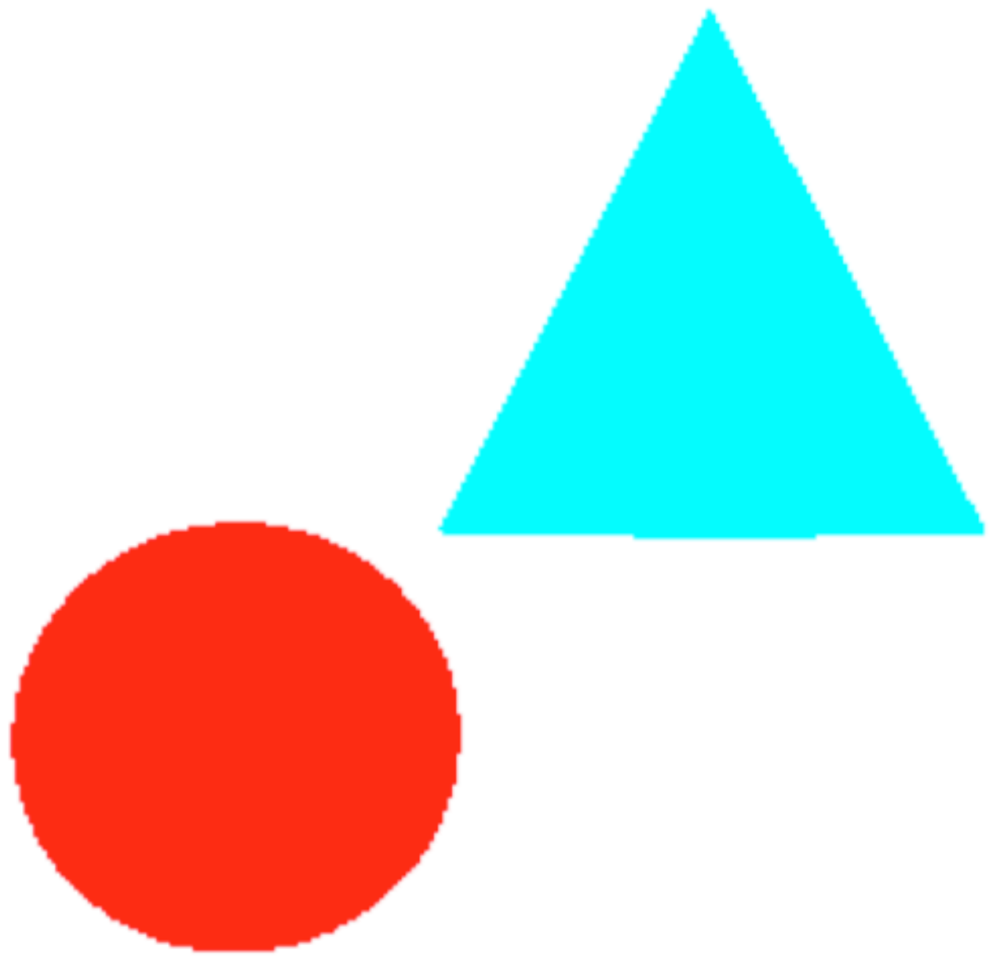
Lighting (Theory)

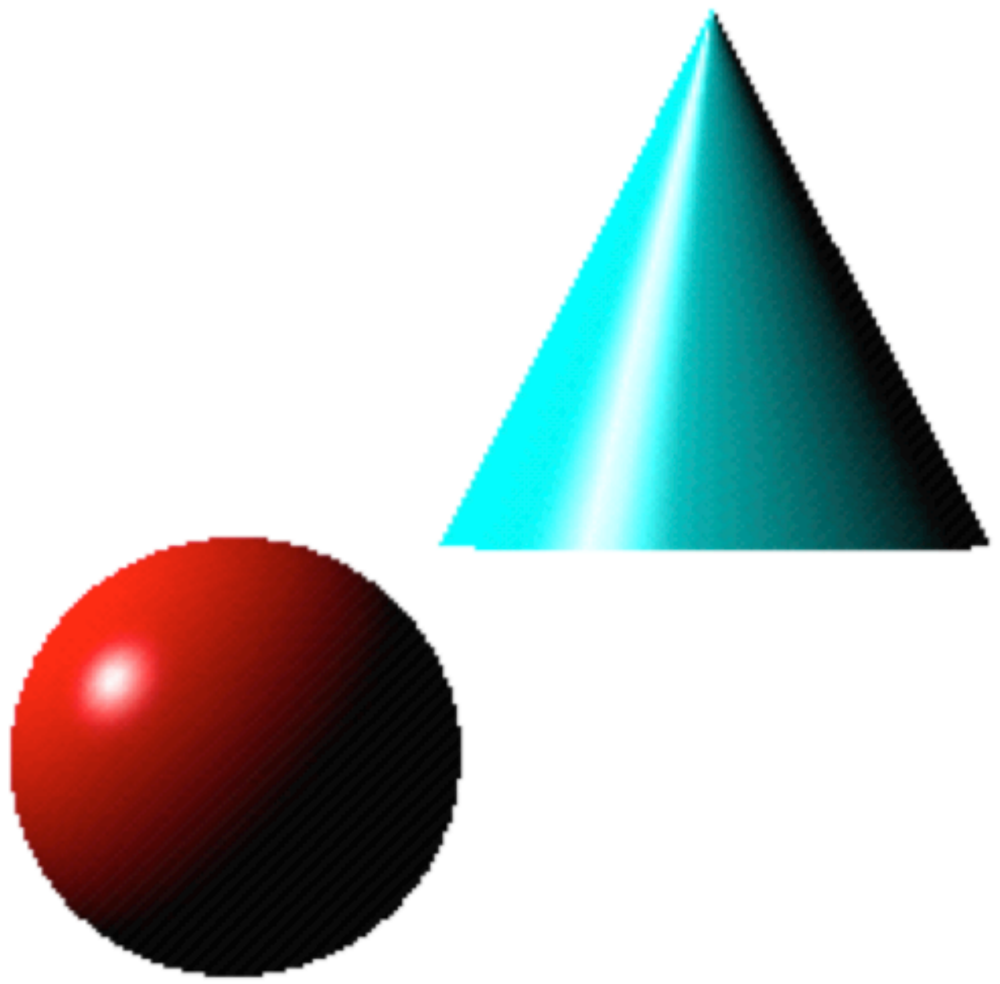
CS 385 - Class 16
17 March 2022

Lighting

Simulating Lighting in CGI

- Lighting is a key component in computer graphics
- Provides cues on:
  - shape and smoothness of objects
  - distance from lights to objects
  - object's orientation in the scene
- Most importantly, it helps makes images more realistic





Lighting Models

- Many different models exist for simulating lighting reflections
  - we'll be concentrating on the *Phong* lighting model
  - it has an *additive color model*
    - technique falls short when colors saturate to white
- Computes a color for each vertex using
  - a surface normal
  - light and material properties
  - viewer's position and viewing direction
- Color are computed at the vertices and are interpolated across polygons by the rasterizer
  - *Gouraud shading*
  - *Phong shading* does the same computation, but per pixel
    - more accurate results
    - just move the computation to the fragment shader

Phong Lighting Equation

- Illumination (lighting) at a point is the sum of three terms:
  - ambient
  - diffuse
  - specular

$$\vec{I} = \vec{I}_{ambient} + \vec{I}_{diffuse} + \vec{I}_{specular}$$

$$\vec{I}_{ambient} \rightarrow \vec{I}_{a}$$

$$\vec{I}_{diffuse} \rightarrow \vec{I}_{d}$$

$$\vec{I}_{specular} \rightarrow \vec{I}_{s}$$

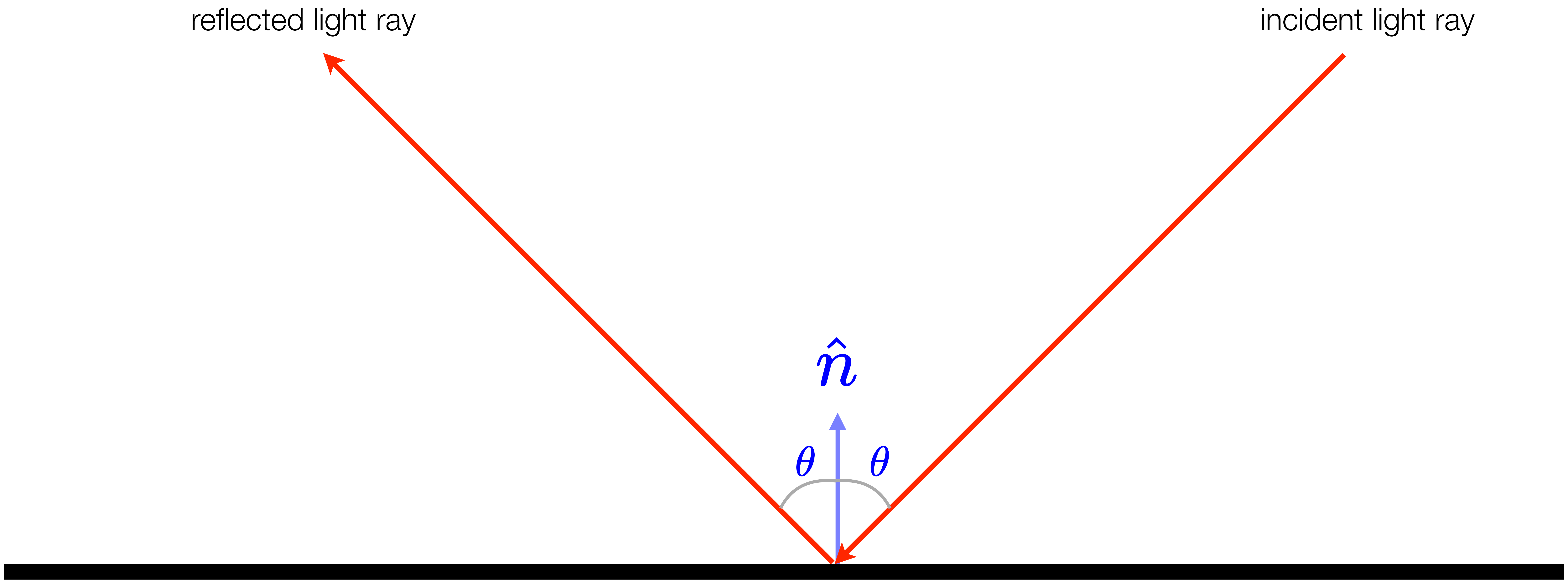
above notation used on following slides

Lighting Mathematics

Lighting Model Components

- Material properties
  - used to describe an object's reflected colors
- Surface normals
- Light properties
  - used to describe a light's color
- "Global" lighting parameters
  - fudge-factors to make things look better

Physics of Reflection



Ambient Reflections

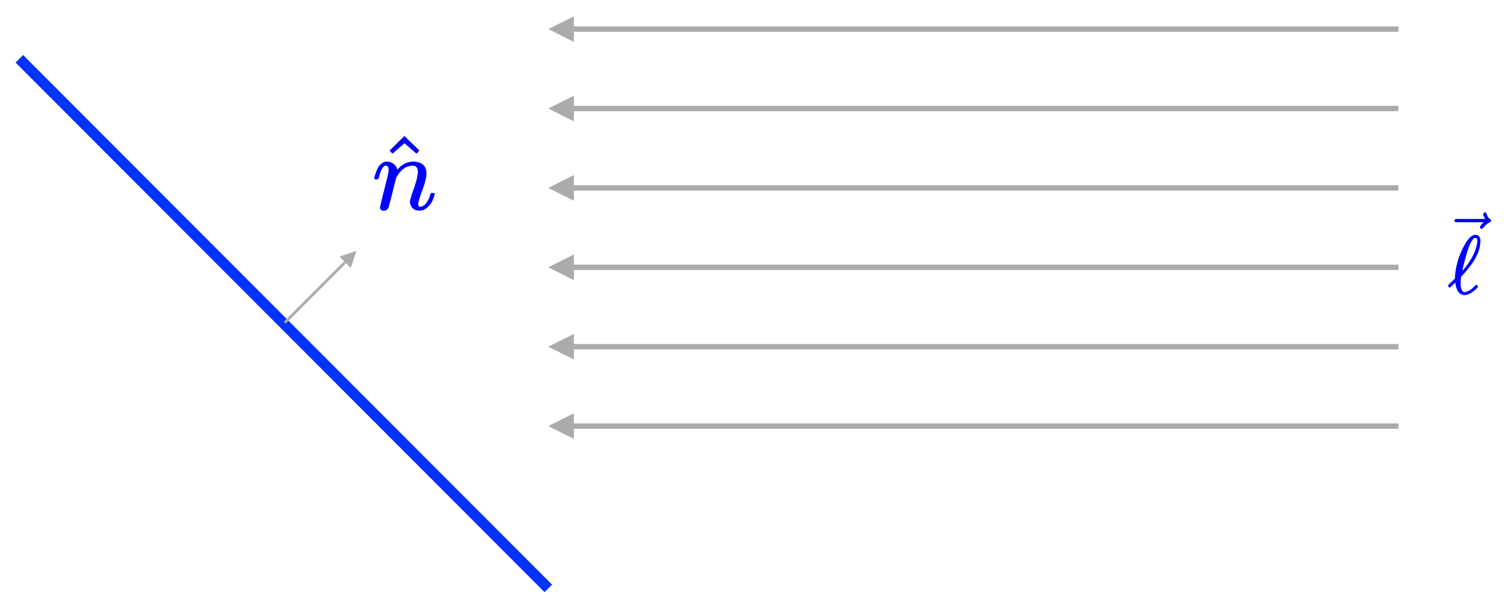
- Color of an object when not directly illuminated
  - light source not directly determinable
- Think about walking into a room with the curtains closed and the lights off

$$\vec{I}_a = \vec{g}_a + \sum_i^n \vec{l}_{a_i} \vec{m}_a$$

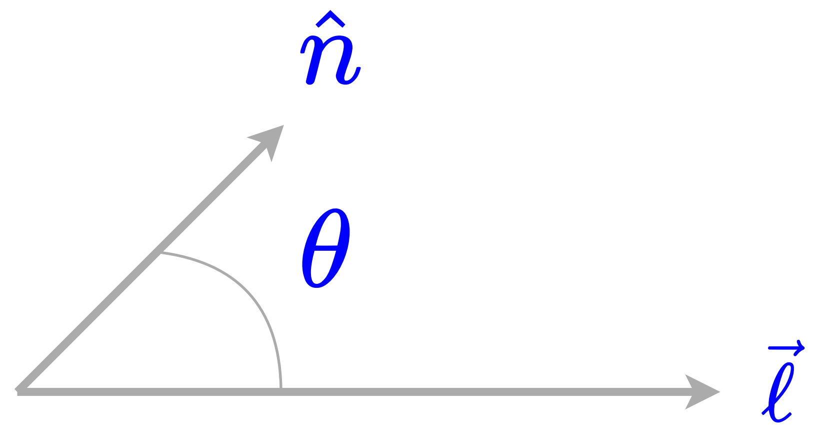
$\vec{g}_a$	global ambient color
$\vec{l}_{a_i}$	light i's ambient color
$\vec{m}_a$	ambient material color

Diffuse Reflections

- Color of an object when directly illuminated
  - often referred to as the *base color*



$$\vec{I}_d = \sum_{i}^{n} \left(\hat{l}_i \cdot \hat{n} \right) \vec{l}_{d_i} \vec{m}_d$$

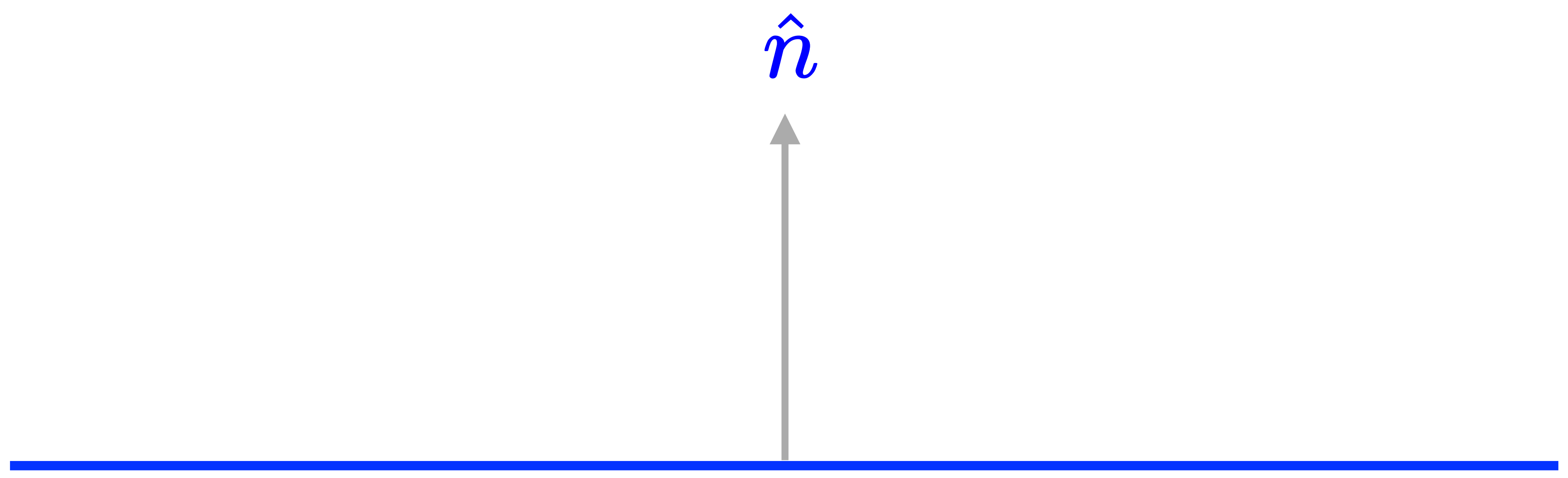


$\hat{l}_i$	normalized light direction
$\hat{n}$	surface normal
$\vec{l}_{d_i}$	ligth i's diffuse color
$\vec{m}_d$	diffuse material color

Specular Reflections

- Highlight color of an object
- Shininess exponent used to shape highlight

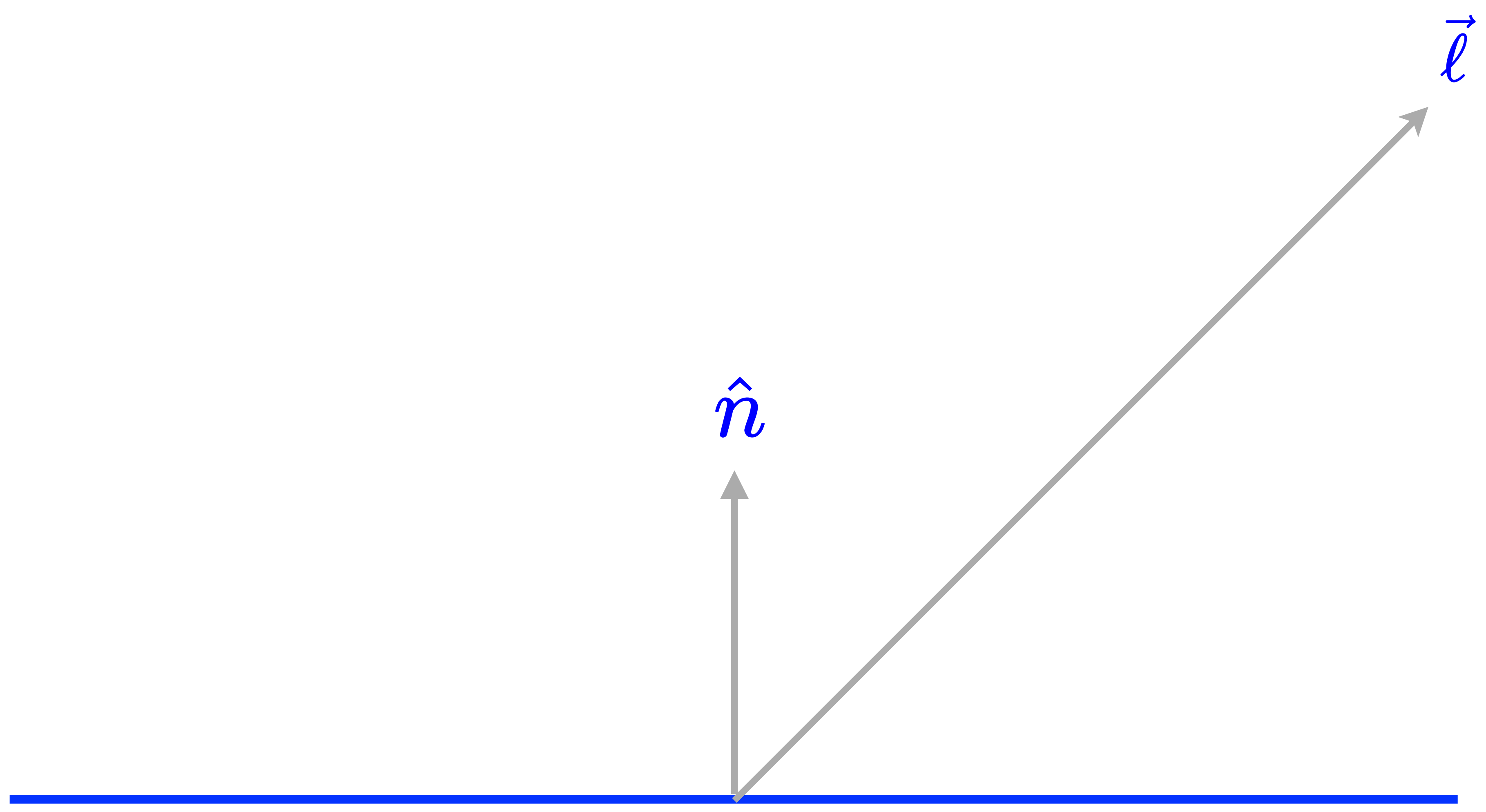
$$\vec{I}_s = \sum_i^n \left(\hat{h} \cdot \hat{n}\right)^s \vec{l}_{s_i} \vec{m}_s$$



$\hat{n}$	surface normal
$\hat{h}$	half-angle vector
$()^s$	shininess exponent
$\vec{l}_{s_i}$	light i's specular color
$\vec{m}_s$	specular material color

Specular Reflections

- Highlight color of an object
- Shininess exponent used to shape highlight

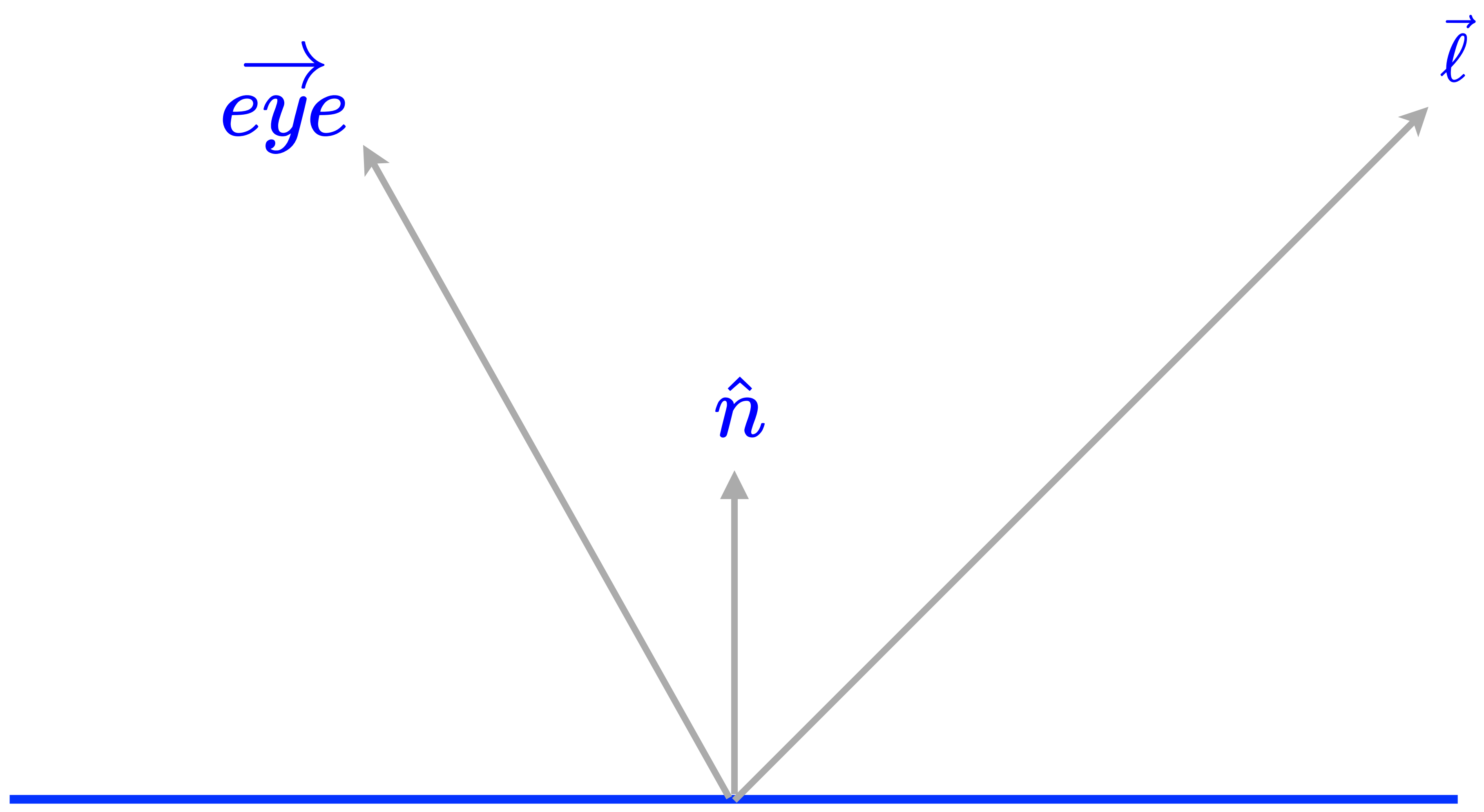


$$\vec{I}_s = \sum_i^n \left(\hat{h} \cdot \hat{n}\right)^s \vec{l}_{s_i} \vec{m}_s$$

$\hat{n}$	surface normal
$\hat{h}$	half-angle vector
$()^s$	shininess exponent
$\vec{l}_{s_i}$	light i's specular color
$\vec{m}_s$	specular material color

Specular Reflections

- Highlight color of an object
- Shininess exponent used to shape highlight



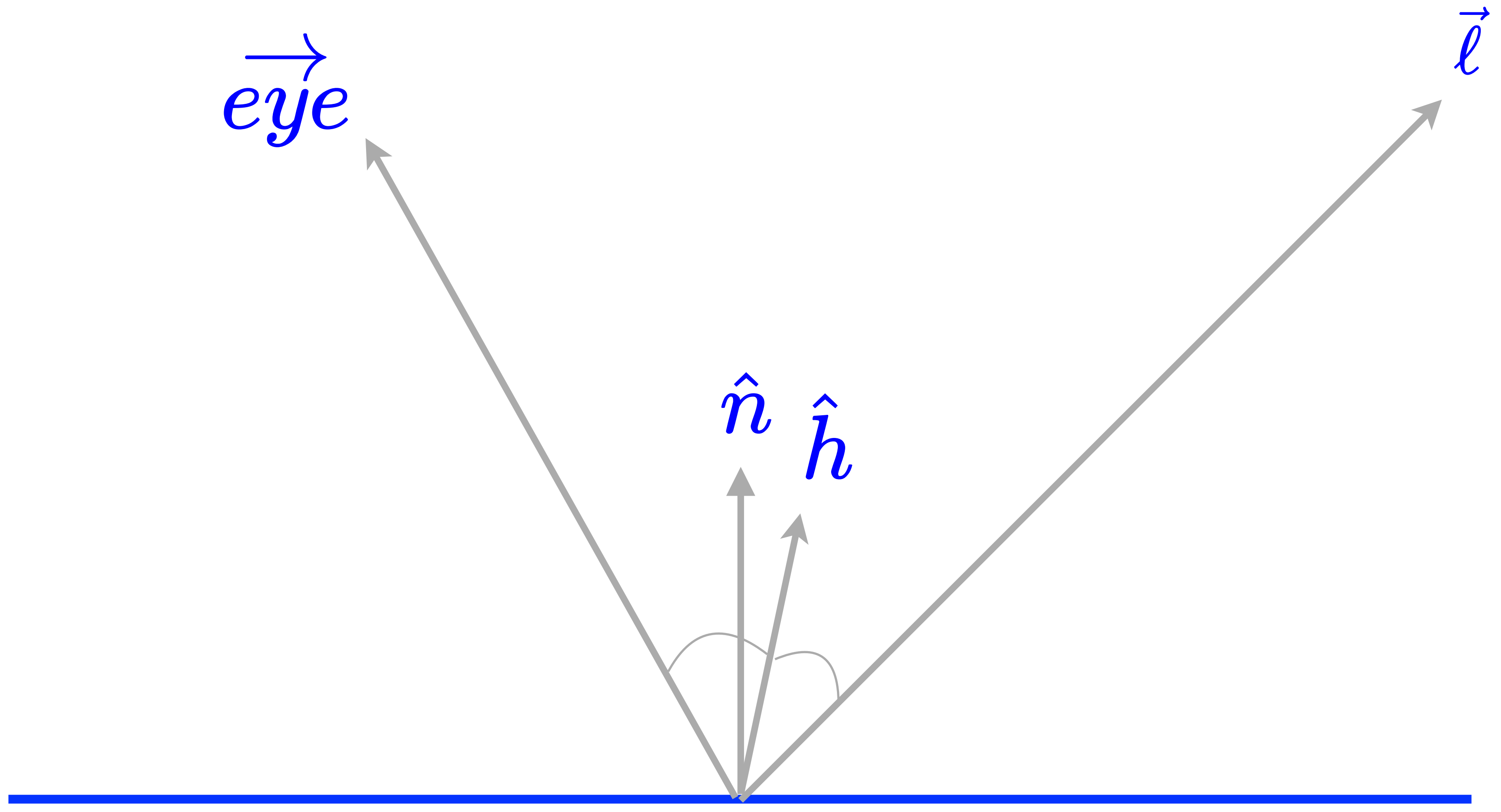
$$\vec{I}_s = \sum_i^n \left(\hat{h} \cdot \hat{n}\right)^s \vec{l}_{s_i} \vec{m}_s$$

$\hat{n}$	surface normal
$\hat{h}$	half-angle vector
$()^s$	shininess exponent
$\vec{l}_{s_i}$	light i's specular color
$\vec{m}_s$	specular material color

Specular Reflections

- Highlight color of an object
- Shininess exponent used to shape highlight

$$\vec{I}_s = \sum_i^n \left(\hat{h} \cdot \hat{n}\right)^s \vec{l}_{s_i} \vec{m}_s$$



$$\hat{h} = \left\| \frac{\vec{l} + \overrightarrow{eye}}{2} \right\|$$

$\hat{n}$	surface normal
$\hat{h}$	half-angle vector
$()^s$	shininess exponent
$\vec{l}_{s_i}$	light i's specular color
$\vec{m}_s$	specular material color

Material Properties

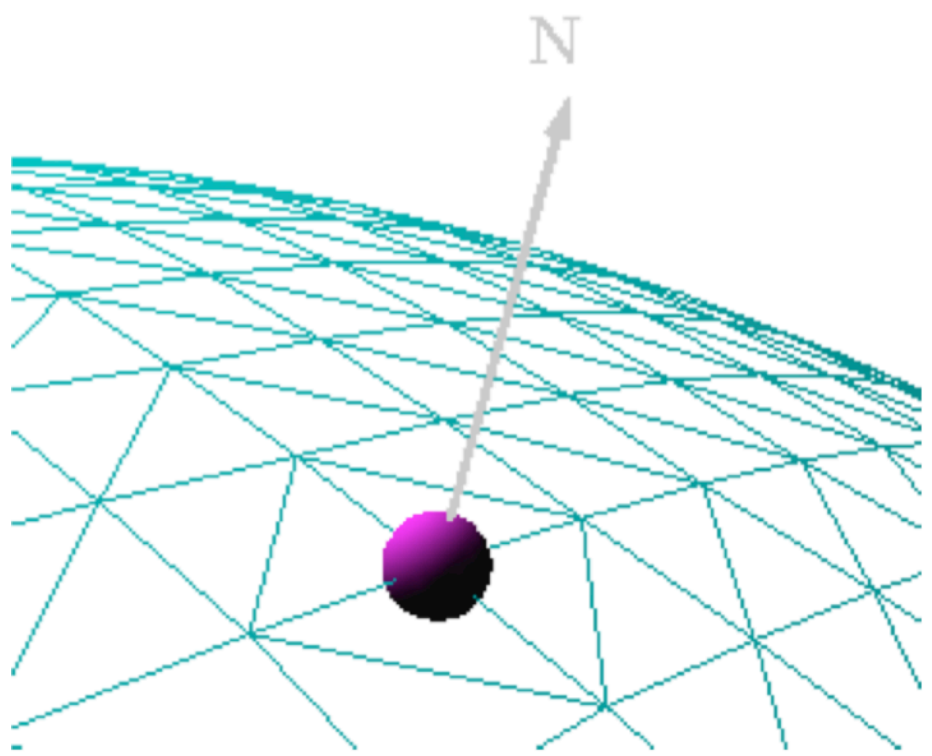
- Define the surface properties of an object
  - similar set to what we defined for a light
- You can have separate material properties for the front and back (usually, inside and outside) of an object
  - use the `gl_FrontFacing` boolean in your shader

Property	Symbol	Description
Diffuse	$\vec{m}_d$	Base color
Ambient	$\vec{m}_a$	Low-light color
Specular	$\vec{m}_s$	Highlight color
Shininess	$()^s$	Surface Smoothness
Emission	$\vec{m}_e$	"Glow" color

Surface Normals

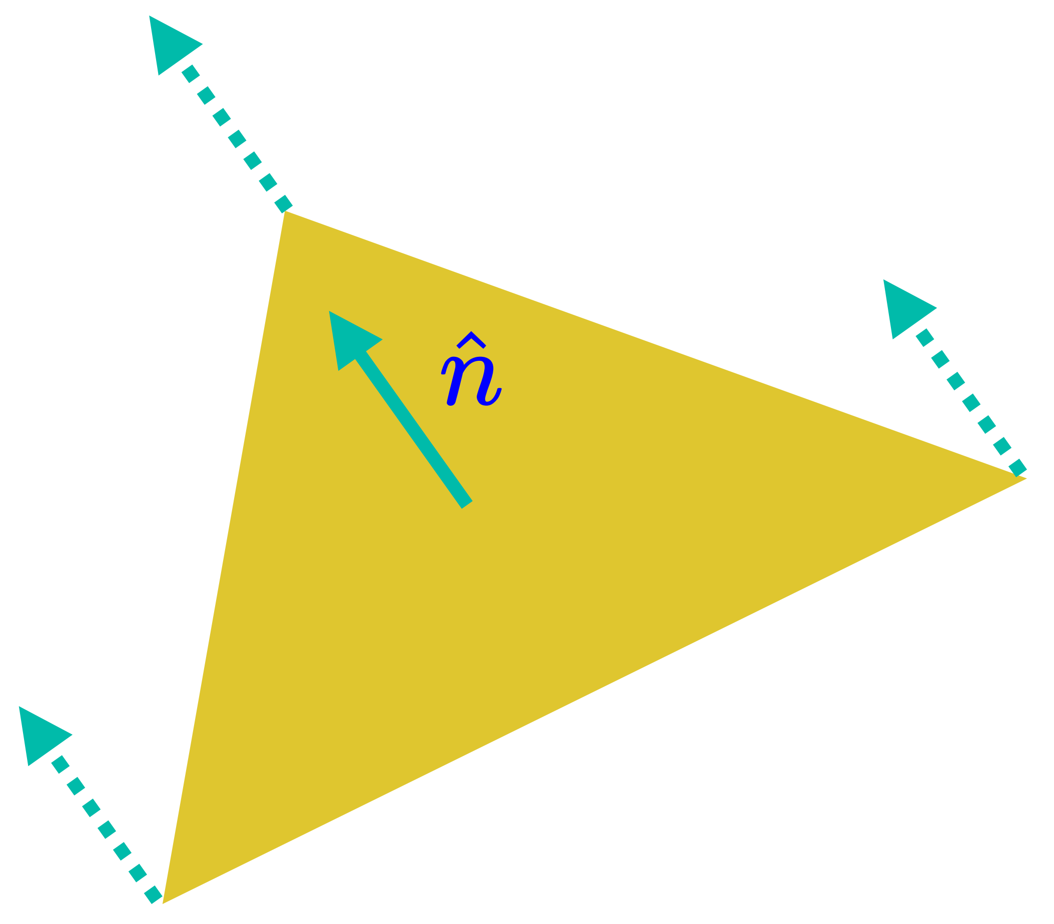
Surface Normals

- Yet another vertex attribute
- Normals define the direction that a surface reflects light
- Applications usually provide (or generate) normals as a vertex attribute
- Always use unit-length normals
  - use the `normalize` GLSL function
- Normals are used to compute the vertex's (or fragment's) illumination color



Face Normals

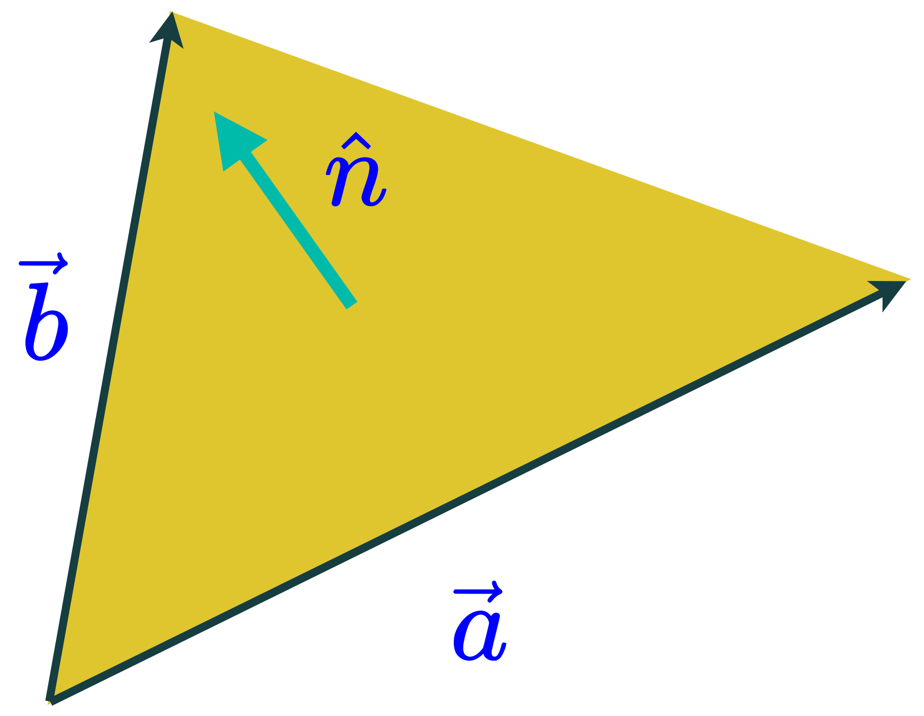
- Same normal for all vertices in a primitive
  - it's the normal to the "plane" the primitive lives in



Computing Face Normals

- We're only using planar polygons
- Compute the plane's normal using the vector cross product
- Remember, order of vectors in a cross product matters
    - another application of the *right-hand rule*

$$\vec{a} \times \vec{b} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$



$$\hat{n} = \frac{\vec{a} \times \vec{b}}{||\vec{a} \times \vec{b}||}$$

Computing Normals (Algebraically)

- If you have a mathematical formula for the surface, evaluate the gradient at a point on the surface

$$\vec{n} = \nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)_{(x,y,z)}$$

$$\hat{n} = \frac{\vec{n}}{||\vec{n}||}$$

Huh? Perhaps an example ...

- Consider the equation for a unit sphere $f(x, y, z) = x^2 + y^2 + z^2 - 1$

$$\vec{n} = \nabla f = (2x, 2y, 2z)$$

$$\hat{n} = \frac{\vec{n}}{||\vec{n}||}$$

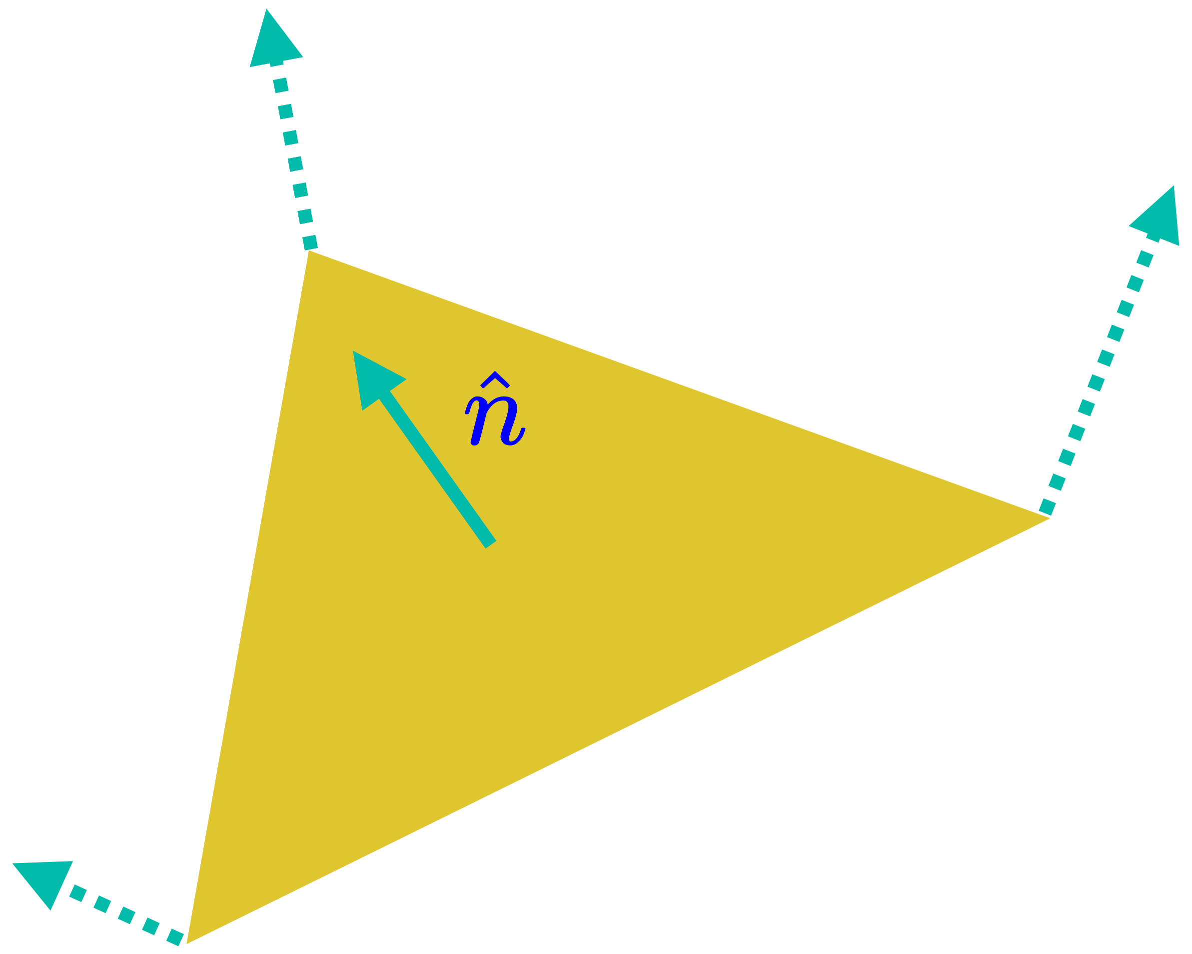
$$\boxed{\hat{n} = (x, y, z)}$$

$$\begin{aligned} ||\vec{n}|| &= \sqrt{(2x)^2 + (2y)^2 + (2z)^2} \\ &= \sqrt{4x^2 + 4y^2 + 4z^2} \\ &= 2\sqrt{x^2 + y^2 + z^2} \\ &= 2r \qquad \text{(but } r = 1 \text{ for a unit sphere)} \\ &= 2 \end{aligned}$$

- So, for a unit sphere (or any sphere in general) its normal at a point, is just the point's coordinates (normalized)

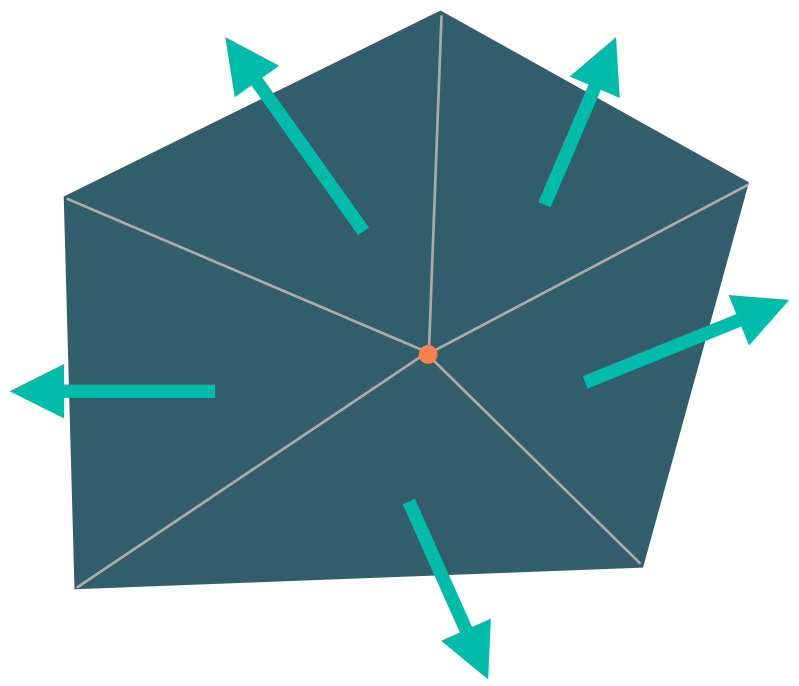
Vertex Normals

- Each vertex has its own normal
  - these might be computed analytically, or come from a modeling tool
- Primitive is Gouraud shaded based on computed colors



Computing Vertex Normals (when you don't have a formula)

- Need to know a few things:
  - face normals for all polygons
  - list of which polygons are incident to each vertex



$$\hat{n}_v = \left\| \sum_i^n \hat{n}_i \right\|$$