More Lighting (Theory)

CS 385 - Class 17
29 March 2022

Review

Phong Lighting Equation

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

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

Ambient Reflections

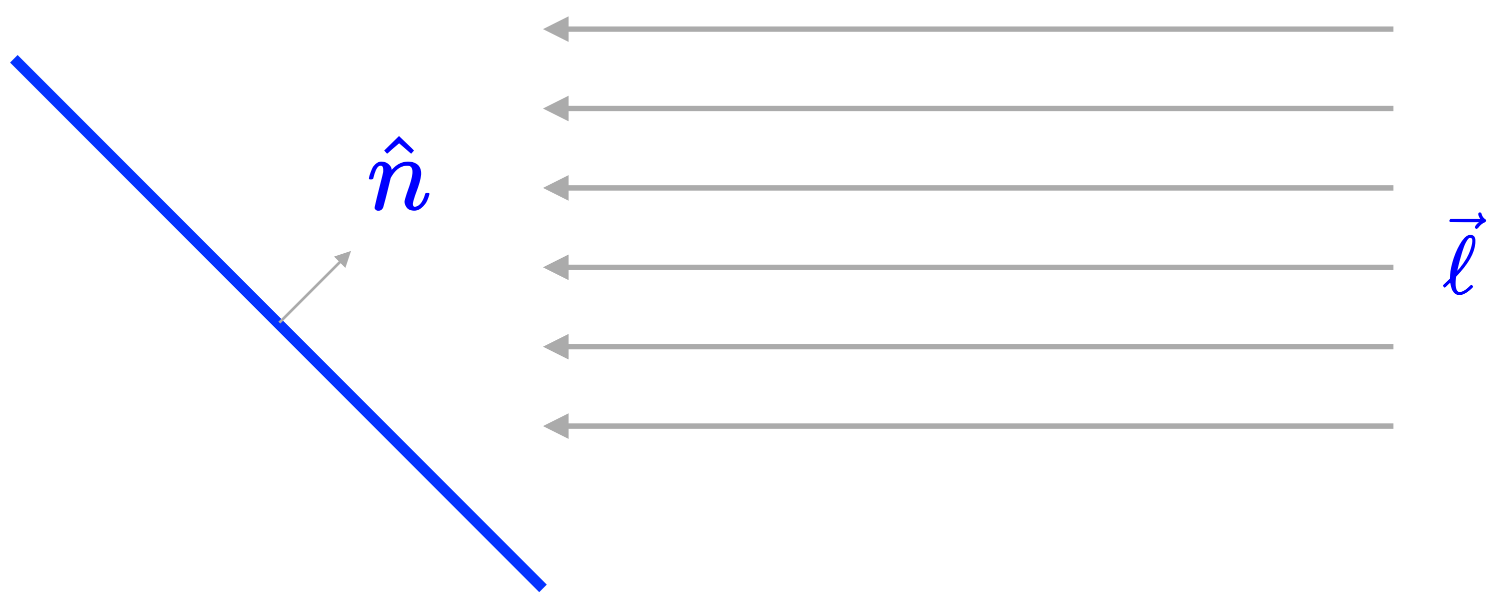
- Color of an object when not directly illuminated
  - light source not directly determinable
- Think about walking into a room with the curtains closed and the lights off

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

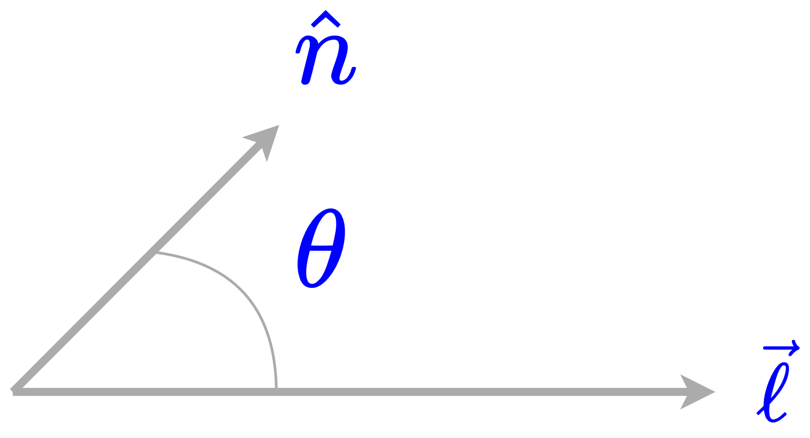
$\vec{g}_a$	global ambient color
$\vec{l}_{a_i}$	light i's ambient color
$\vec{m}_a$	ambient material color

Diffuse Reflections

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



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

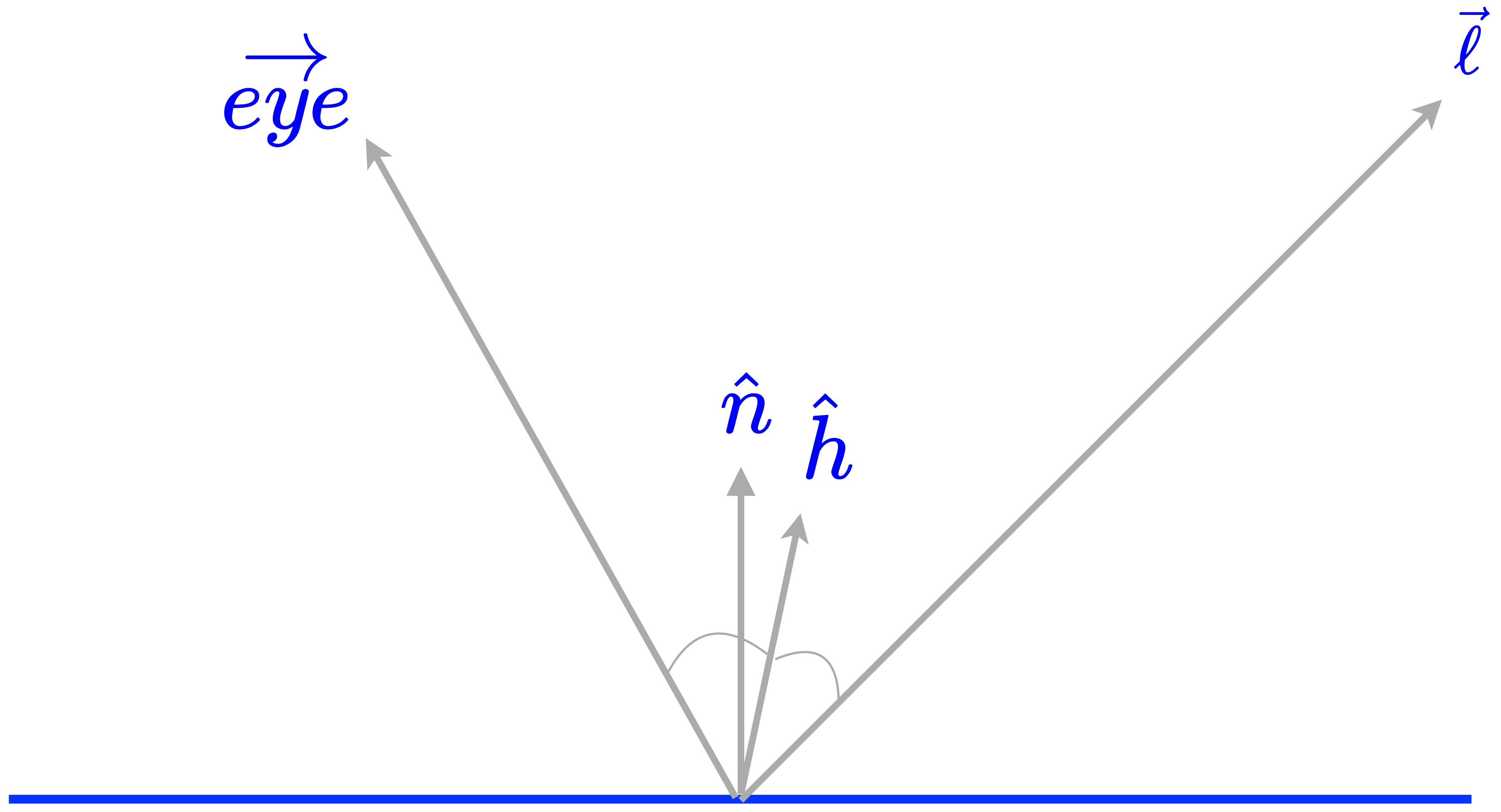


$\hat{l}_i$	normalized light direction
$\hat{n}$	surface normal
$\vec{l}_{d_i}$	ligth i's diffuse color
$\vec{m}_d$	diffuse material color

Specular Reflections

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

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

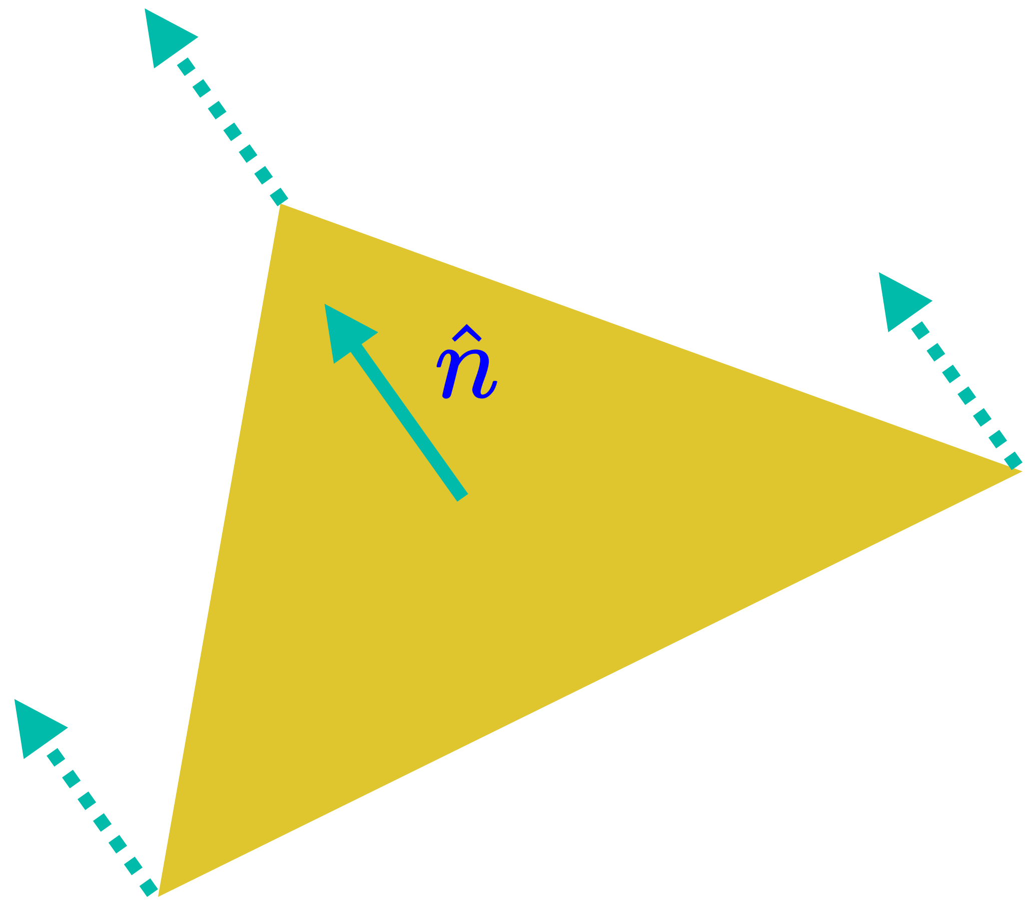


$$\hat{h} = \left\| \frac{\vec{l} + \overrightarrow{eye}}{2} \right\|$$

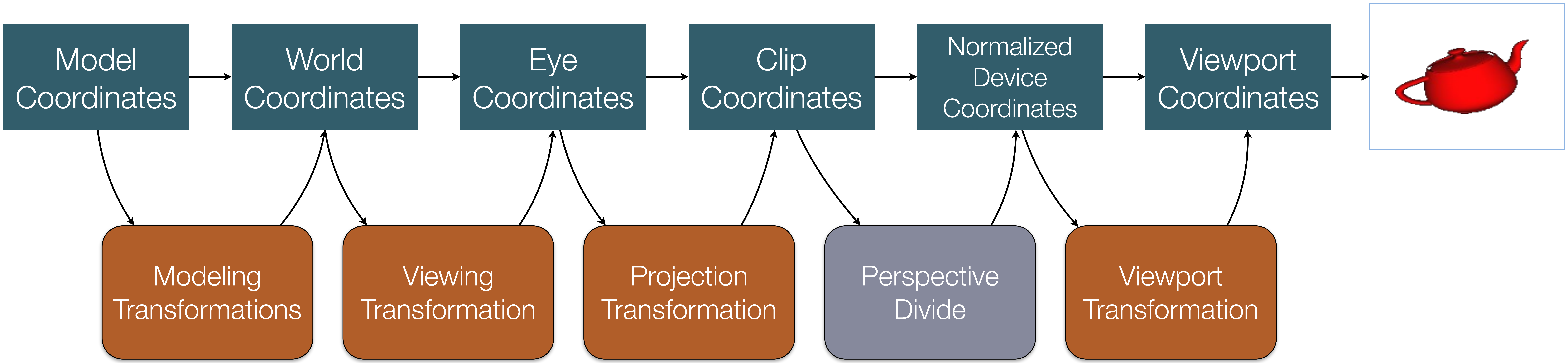
$\hat{n}$	surface normal
$\hat{h}$	half-angle vector
$()^s$	shininess exponent
$\vec{l}_{s_i}$	light i's specular color
$\vec{m}_s$	specular material color

Lighting Normals

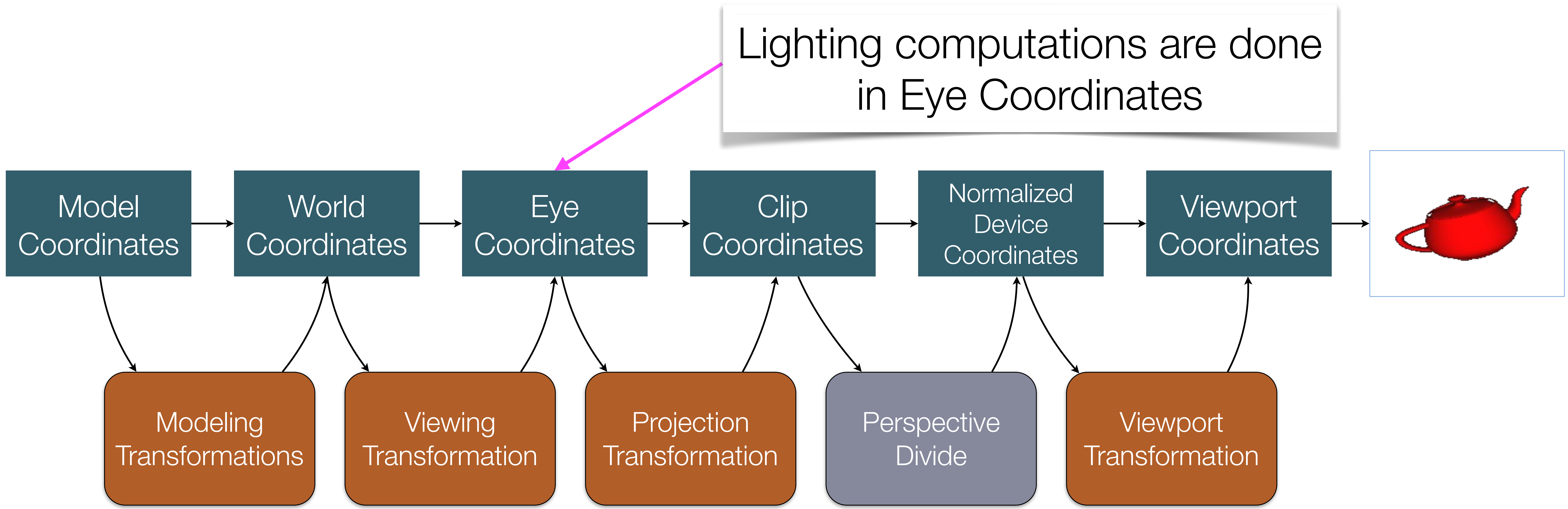
- Recall that we use a vector, called a *normal* to aid in lighting computations
  - as compared to a *point*, vectors only have three components: (x, y, z)
  - they only specify a direction



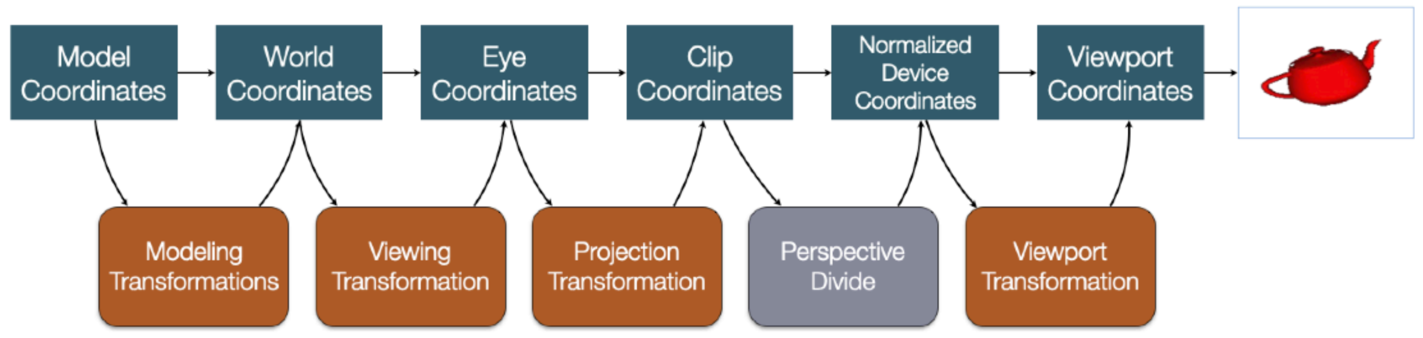
Coordinate Systems in Computer Graphics



Coordinate Systems in Computer Graphics



Transformation Redux



- Recall that we transform vertex *positions* using matrices
  - each transform moves the position from one coordinate system to the next

```
gl_Position = P * MV * aPosition;
```

Transform	Operation	Change the *Shape* of Space	Normal Affected?
Translate	*Move* origin from one position to another	No	No
Rotate	Update the *orientation* of space	No	Yes
Scale	*Stretch/Shrink* space	Yes	Yes
Projection	Transform 3D world to 2D plane	Yes, but ...	N/A

The Normal Matrix

Transforming Normals

- We need to transform normals when we transform positions
- But it's not the same operation!
- Transform into *eye* coordinates
  - don't include the projection transformation
- However, normals only have three components
  - `MV` is a 4×4 matrix
  - the important parts are in the *upper left* 3×3 *matrix*

Introducing the Normal Matrix, N

- It's the *inverse-transpose* of the *upper-left* 3×3 of the model-view matrix
- Right!

$$M = \begin{pmatrix} \boxed{\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix}} & \begin{matrix} x \\ y \\ z \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{pmatrix}$$

$$N = (M^{-1})^T$$

JavaScript

```
function render() {
  // set up matrix stack
  var MV = ms.current();
  var M = mat3(vec3(MV[0]), vec3(MV[1]), vec3(MV[2]));
  var N = inverse(transpose(M));
  // update normal matrix uniform variable
}
```

Matrix Machinations: Inverses and Transposes

Matrix Inverses

- For the value a, its *inverse*, a^{-1}, is the value that satisfies the *identity relationship:* $a\ a^{-1} = I$

Type	Inverse	Identity Value	Name
Real Number (including rationals and integers)	$\frac{1}{a}$	1	one
Matrix	M^{-1}	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	Identity Matrix

Matrix Transpose

- The matrix specified by exchanging the rows and columns of a matrix

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & z & 1 \end{pmatrix}$$

Example: Translation Matrix Inverse

- Consider a translation generated by `translate(x, y, z)`
- What's its inverse?

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Example: Translation Matrix Inverse

- Consider a translation generated by `translate(x, y, z)`
- What's its inverse?
  - that is, what matrix multiplying the translation would generate an identity matrix?

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - Good News! In graphics, these are pretty simple to figure out

Example: Translation Matrix Inverse

- Consider: if you move, say, three feet to the left, how do you get back to your original location?

Example: Translation Matrix Inverse

- A translation of $(-x, -y, -z)$ undoes a translation of (x, y, z)

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M \qquad M^{-1} \qquad = \qquad I$$

- Transpose? Who cares!
  - well, we do, but not for this exercise

Example: Scale Matrix Inverse

- A scale of $\left(\frac{1}{x}, \frac{1}{y}, \frac{1}{z}\right)$ undoes a scale of (x, y, z)

$$\begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1/x & 0 & 0 & 0 \\ 0 & 1/y & 0 & 0 \\ 0 & 0 & 1/z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M \qquad M^{-1} \qquad = \qquad I$$

- Notice that $S^T = S$
  - S is a *diagonal matrix*; its transpose is itself
  - we'll use this fact later

Example: Rotation Matrix Inverse

- A rotation of angle $-\theta$ undoes a rotation of θ around the same axis

$$R(\theta, \overrightarrow{a})^{-1} = R(-\theta, \overrightarrow{a})$$

- But, rotation matrices have another interesting inverse property: their *transpose* is also their inverse

$$R(\theta, \overrightarrow{a})^{T} = R(\theta, \overrightarrow{a})^{-1} \qquad \Longrightarrow \qquad R(\theta, \overrightarrow{a}) = \left(R(\theta, \overrightarrow{a})^{-1}\right)^{T}$$

Why, oh why?

- Why is this important?
- Lighting is computed in *eye coordinates*, and we need to transform both the positions and normals into that coordinate system
- The big issue with normals is if they're scaled
  - we need to account for that in their transform
  - but scale operations may be combined with a lot of other modeling transformations

Deriving the Normal Matrix

- Transforms without translations can be represented by a 3×3 matrix
- We need to "undo" all scale operations in our `MV` matrix
- Conveniently, any 3×3 transform can be a represented as two rotations and a scale:
- To "undo" something is to use its *inverse*

$$M = R_1 \; S \; R_2$$

$$\begin{aligned} M^{-1} &= \left(R_1 \; S \; R_2\right)^{-1} \\ &= R_1^{-1} \; S^{-1} \; R_2^{-1} \end{aligned}$$

$$\begin{aligned} (M^{-1})^T &= \left(R_1^{-1} \; S^{-1} \; R_2^{-1}\right)^T \\ &= (R_1^{-1})^T \; (S^{-1})^T \; (R_2^{-1})^T \qquad \text{but } (R^{-1})^T = R \\ &= R_1 \; S^{-1} \; R_2 \end{aligned}$$

$$N = \left(M^{-1}\right)^T$$

Midterm Review

Midterm

- Distributed on this coming Thursday (31 March @ 5 PM) on Canvas
  - due the following Thursday (7 April)
- Questions all drawn from class lecture materials and programming assignments
- Topics
  - Coordinate systems
  - Object geometric representation
  - Shaders and rendering
  - Vectors: their uses and related math