Scene Graphs

Connelly Barnes

CS 4810: Graphics

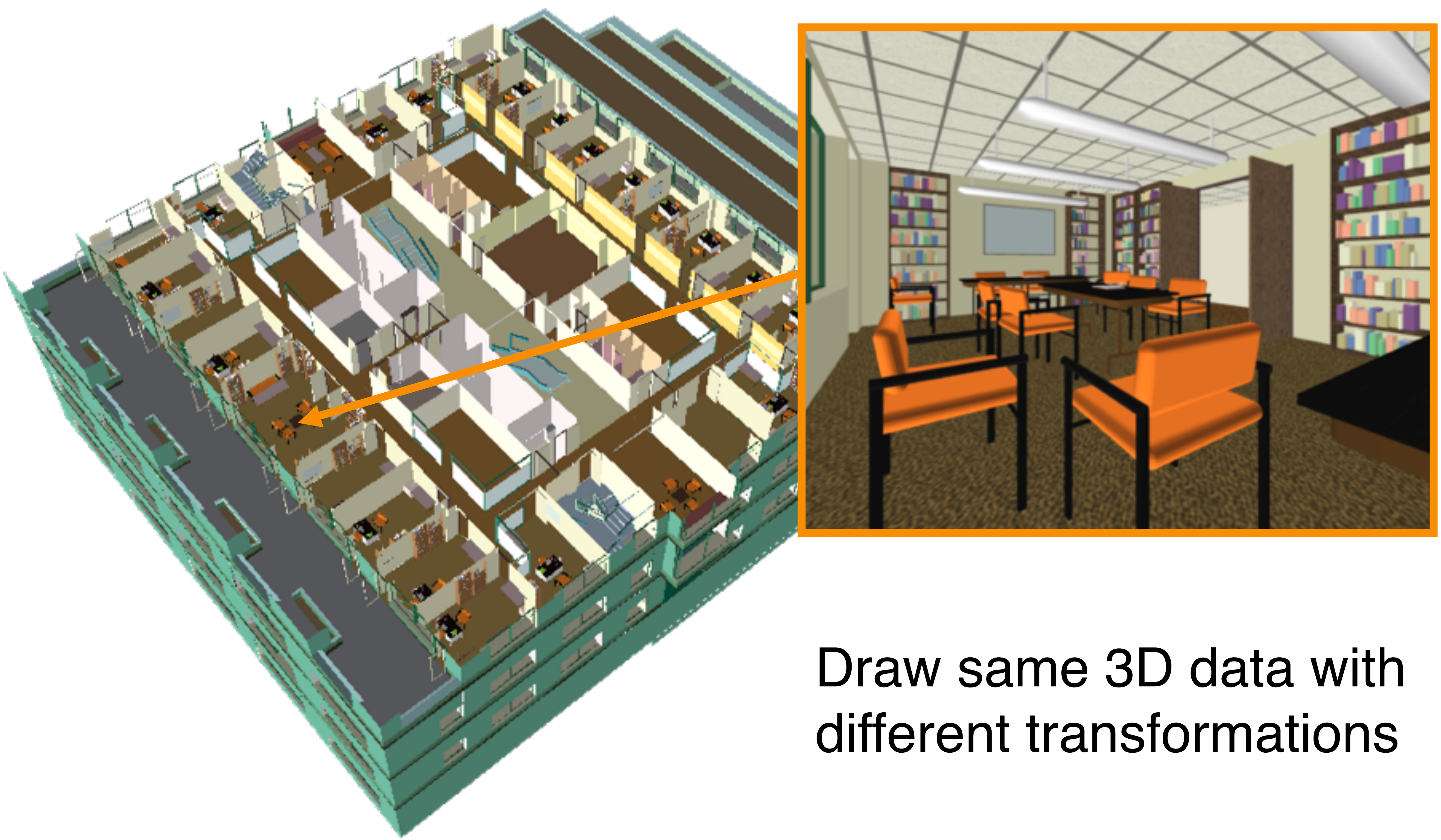
Acknowledgment: slides by Jason Lawrence, Misha Kazhdan, Allison Klein, Tom Funkhouser, Adam Finkelstein and David Dobkin

Overview

- 2D Transformations
  - Basic 2D transformations
  - Matrix representation
  - Matrix composition
- 3D Transformations
  - Basic 3D transformations
  - Same as 2D
- Transformation Hierarchies
  - Scene graphs
  - Ray casting

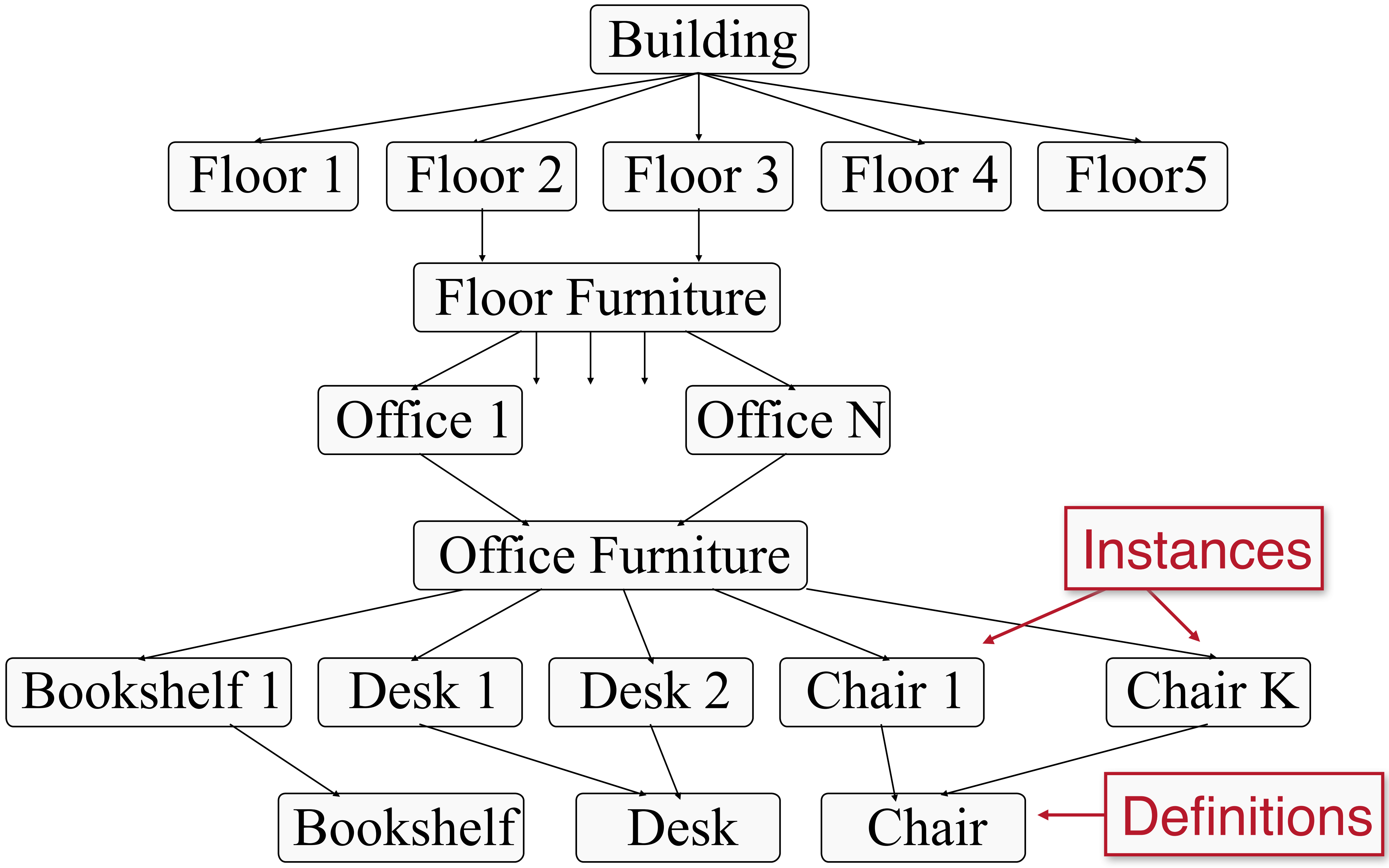
Transformation Example 1

- An object may appear in a scene multiple times



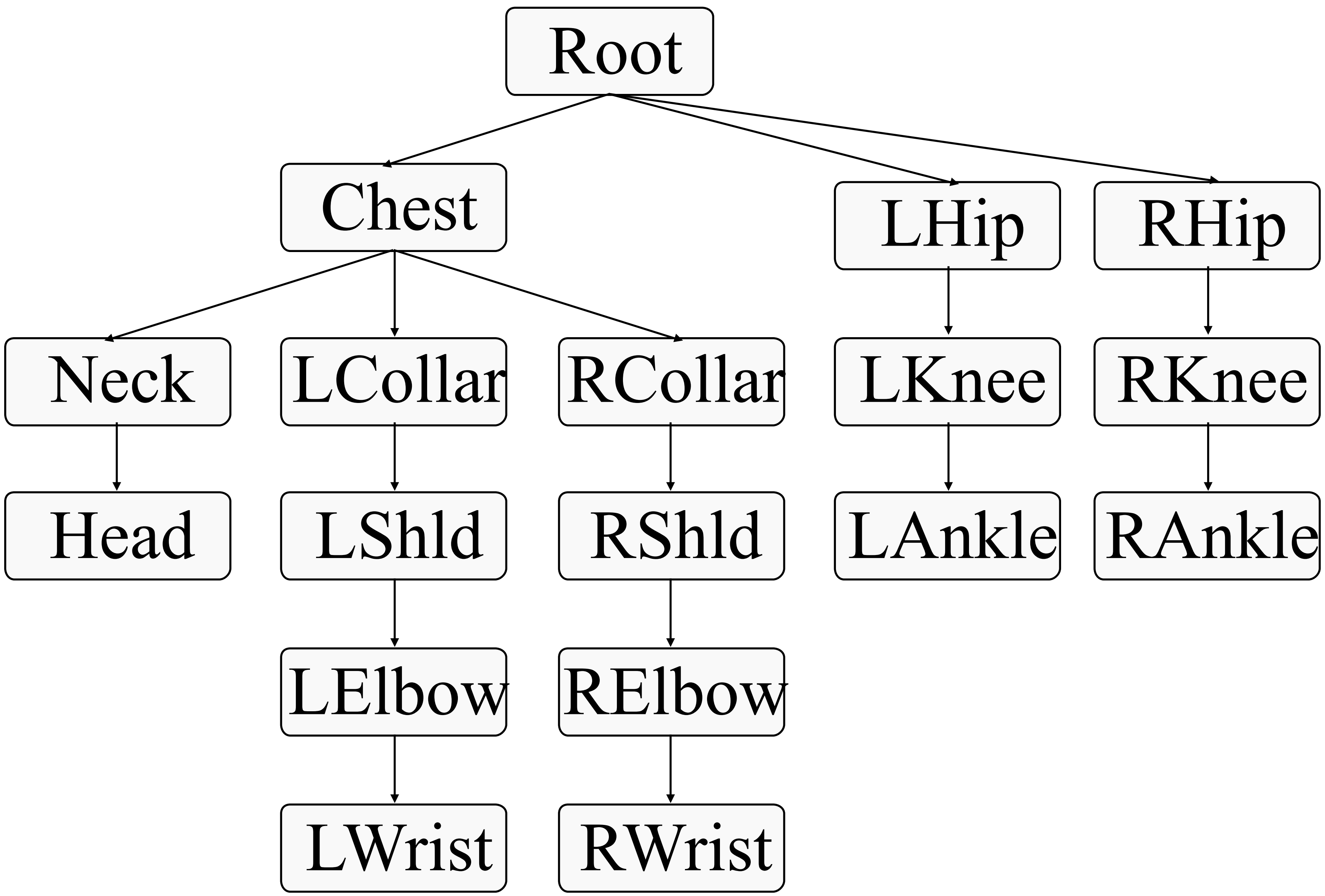
Draw same 3D data with different transformations

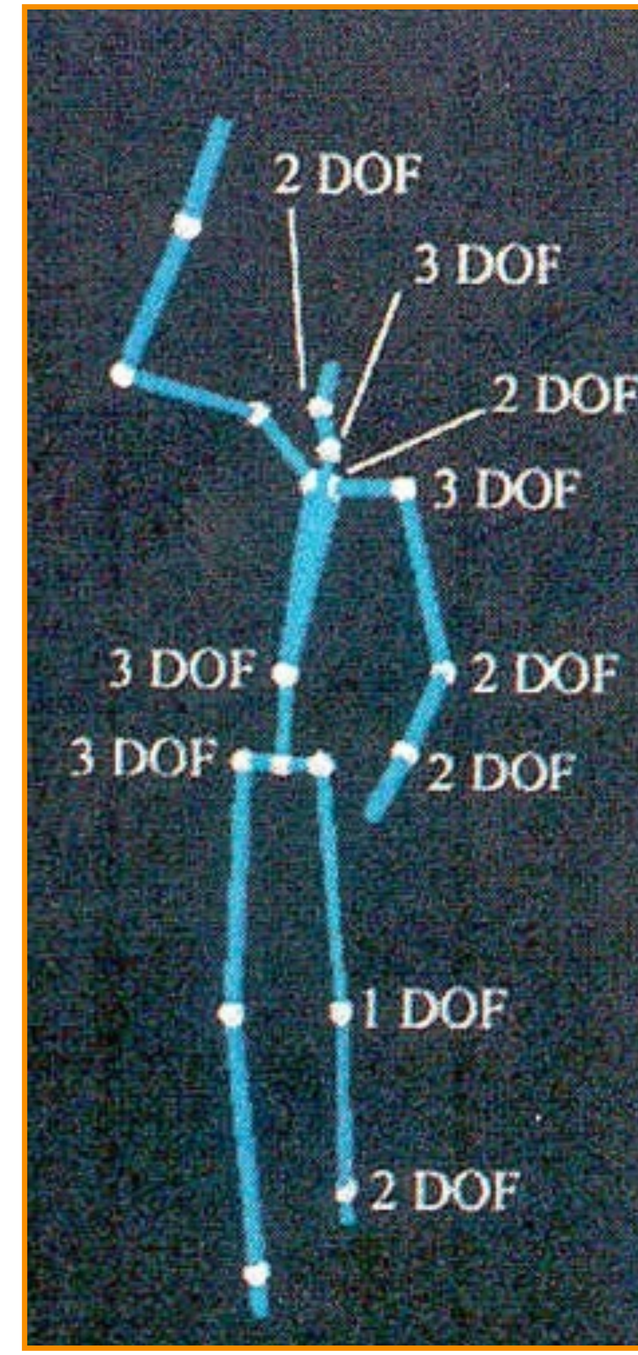
Transformation Example 1



Transformation Example 2

- Well-suited for humanoid characters





Rose et al. `96

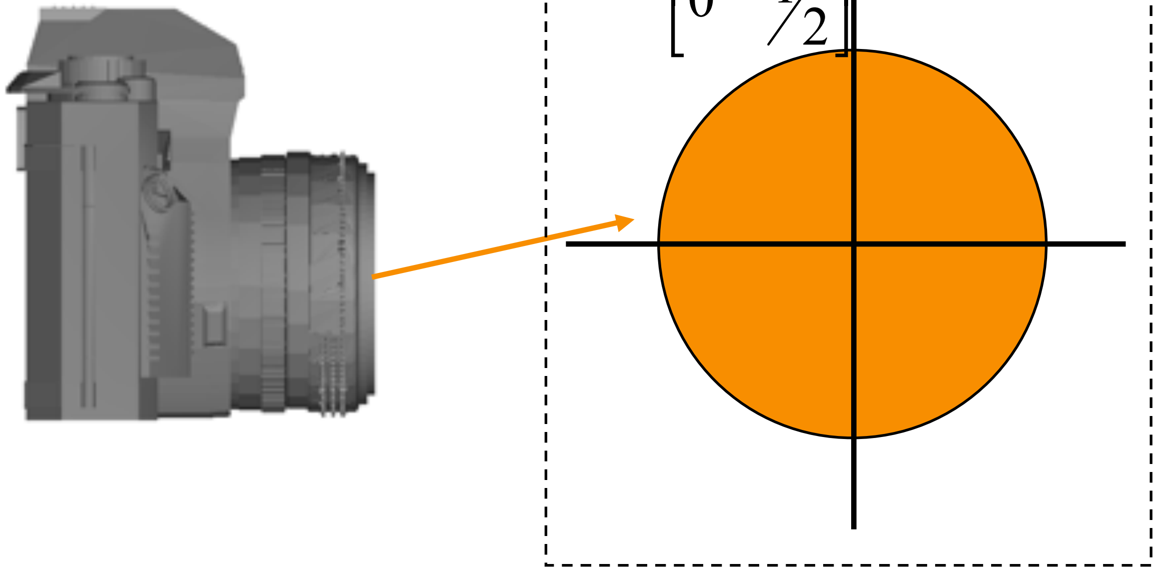
Scene Graphs

- Allow us to have multiple instances of a single model – providing a reduction in model storage size
- Allow us to model objects in local coordinates and then place them into a global frame – particularly important for animation

Scene Graphs

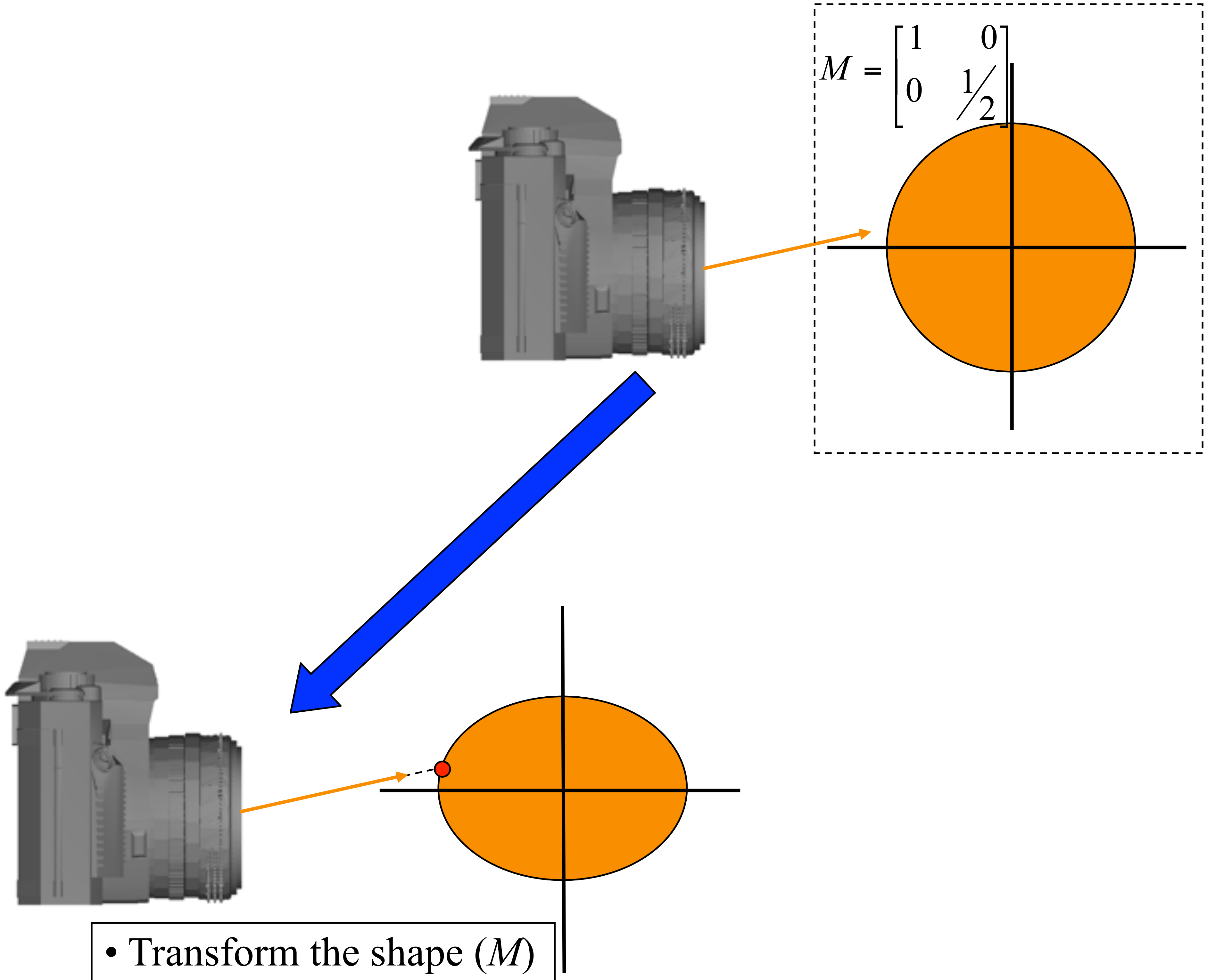
- Allow us to have multiple instances of a single model – providing a reduction in model storage size
- Allow us to model objects in local coordinates and then place them into a global frame – particularly important for animation
- Accelerate ray-tracing by providing a hierarchical structure that can be used for bounding volume testing

Ray Casting with Hierarchies



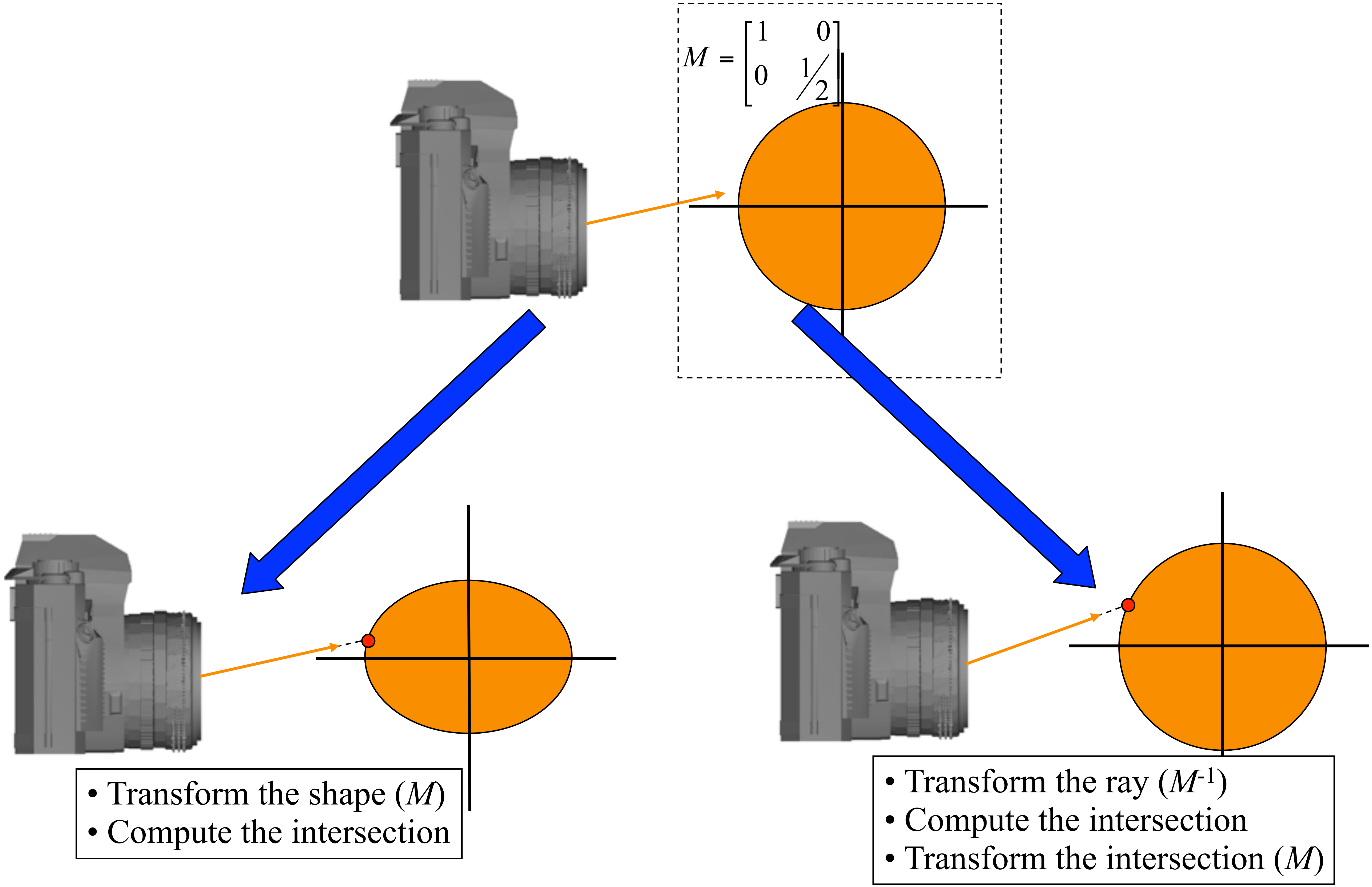
$$M = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

Ray Casting with Hierarchies



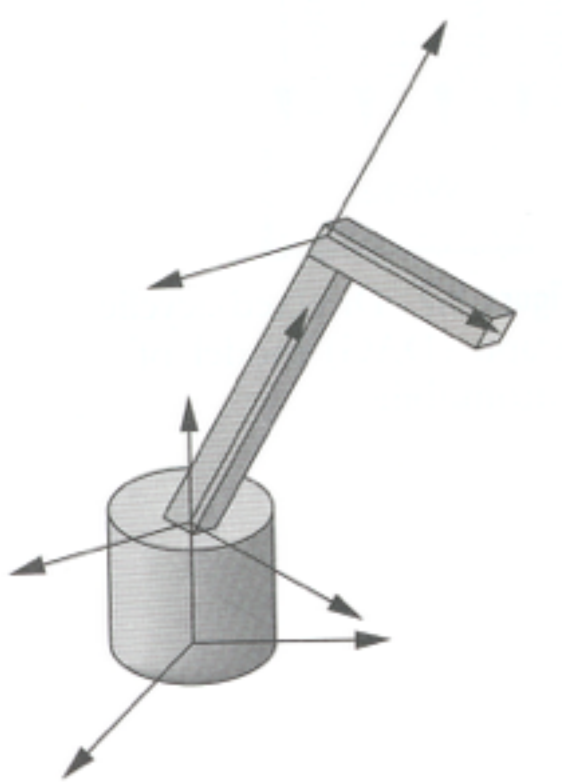
- Transform the shape (M)
- Compute the intersection

Ray Casting with Hierarchies

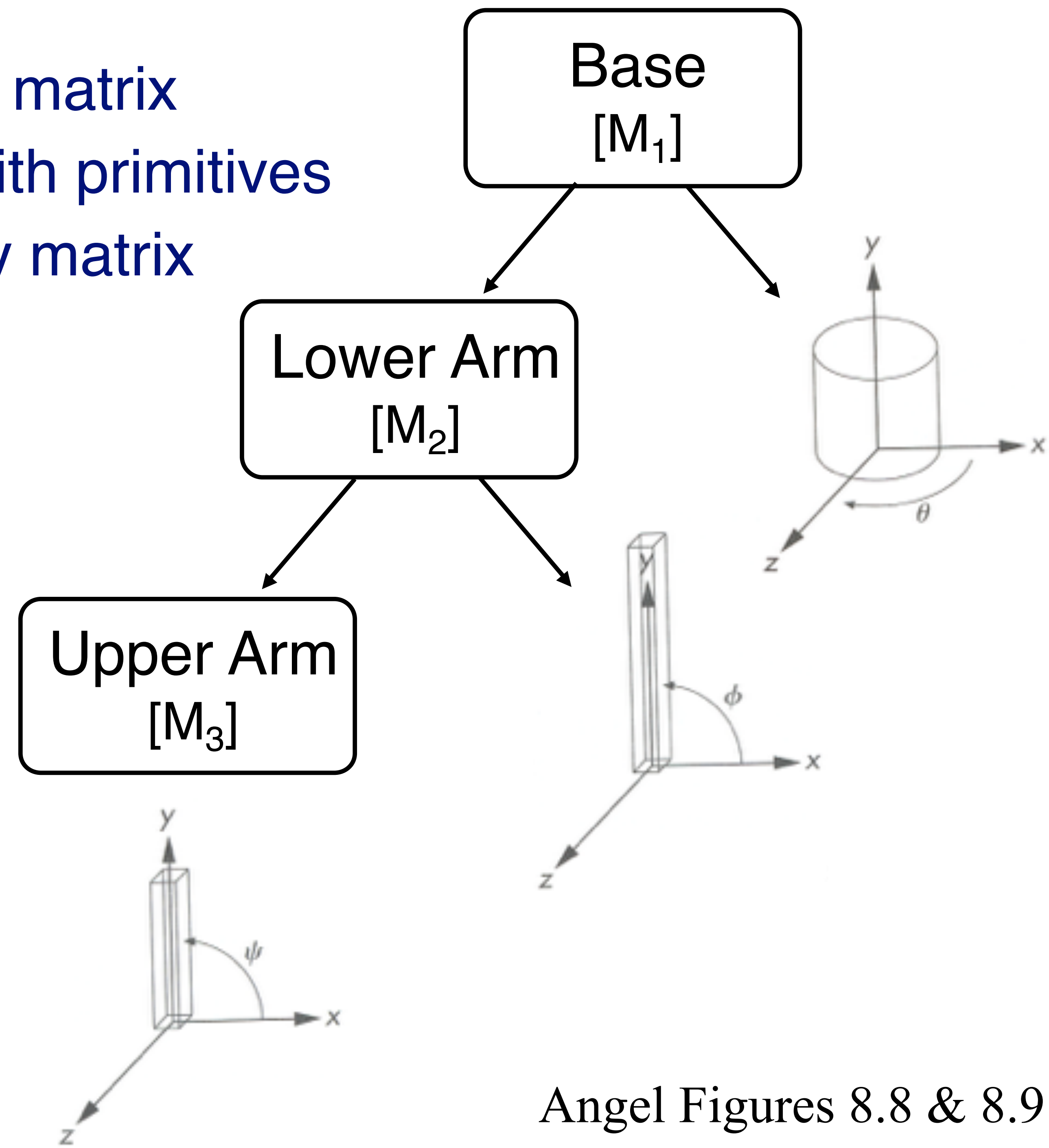


Ray Casting With Hierarchies

- Transform rays, not primitives
  - For each node ...
    - » Transform ray by inverse of matrix
    - » Intersect transformed ray with primitives
    - » Transform hit information by matrix



Robot Arm



Angel Figures 8.8 & 8.9

Applying a Transformation

- Position
- Direction
- Normal

$$\underset{M}{\overset{\text{Affine}}{\begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}}$$

Applying a Transformation

- Position
  - Apply the full affine transformation:

    $p' = M(p) = (M_T \times M_L)(p)$

- Direction

- Normal

$$\underset{M}{\overset{\text{Affine}}{\begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}}$$

Applying a Transformation

- Position
- Direction
  - Apply the linear component of the transformation:

    $p' = M_L(p)$

- Normal

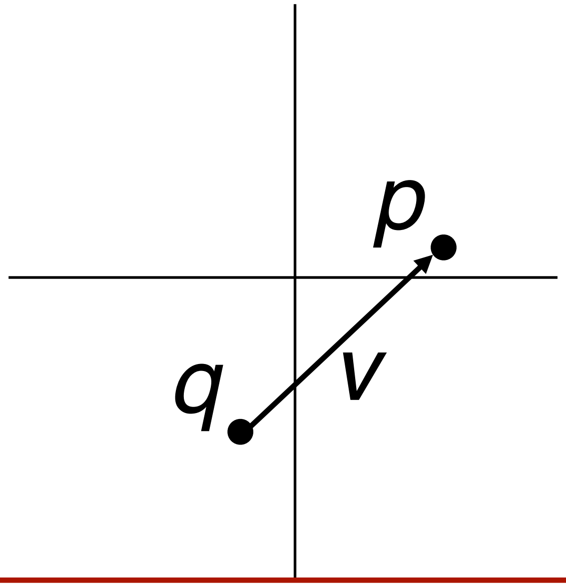
$$\underset{M}{\overset{\text{Affine}}{\begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}}$$

Applying a Transformation

- Position
- Direction
  - Apply the linear component of the transformation:

    $p' = M_L(p)$

A direction vector *v* is defined as the difference between two positional vectors *p* and *q*: ***v=p-q***.



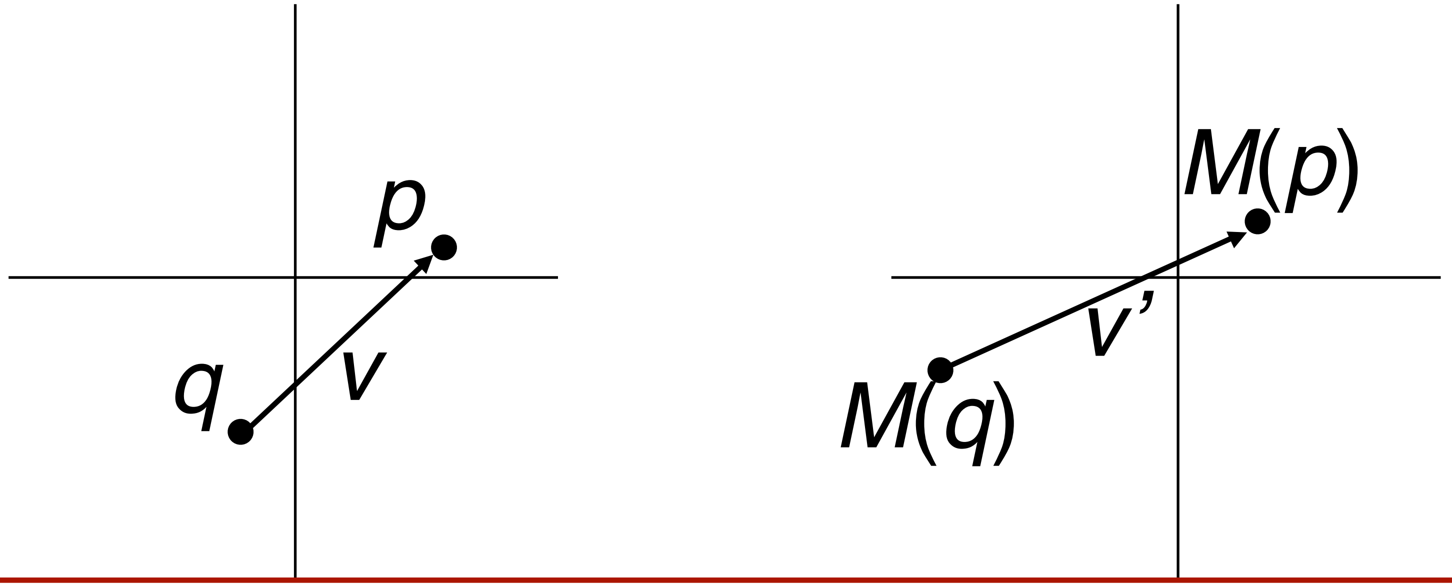
Applying a Transformation

- Position
- Direction
  - Apply the linear component of the transformation:

    $p' = M_L(p)$

A direction vector *v* is defined as the difference between two positional vectors *p* and *q*: ***v*=*p*-*q***.

Applying the transformation M, we compute the transformed direction as the difference between the transformed positions: ***v'*=M(*p*)-M(*q*)**.



Applying a Transformation

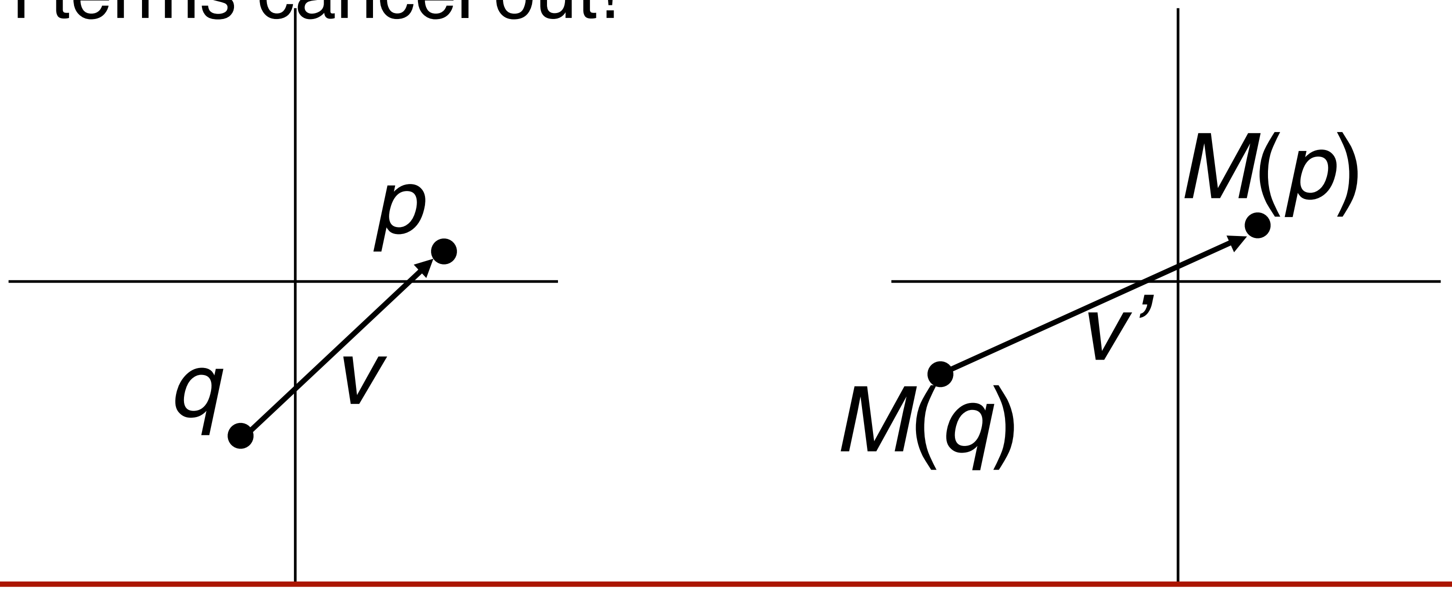
- Position
- Direction
  - Apply the linear component of the transformation:

$$p' = M_L(p)$$

A direction vector v is defined as the difference between two positional vectors p and q: $\boldsymbol{v=p-q}$.

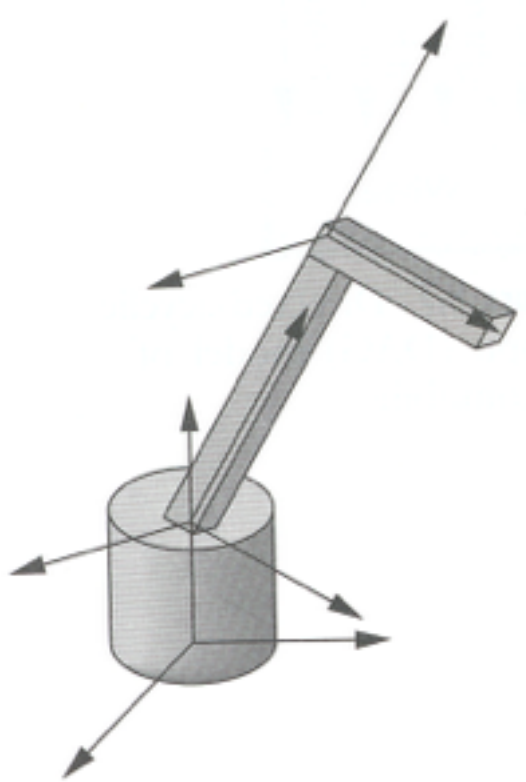
Applying the transformation M, we compute the transformed direction as the difference between the transformed positions: $\boldsymbol{v' = M(p) - M(q)}$.

The translation terms cancel out!

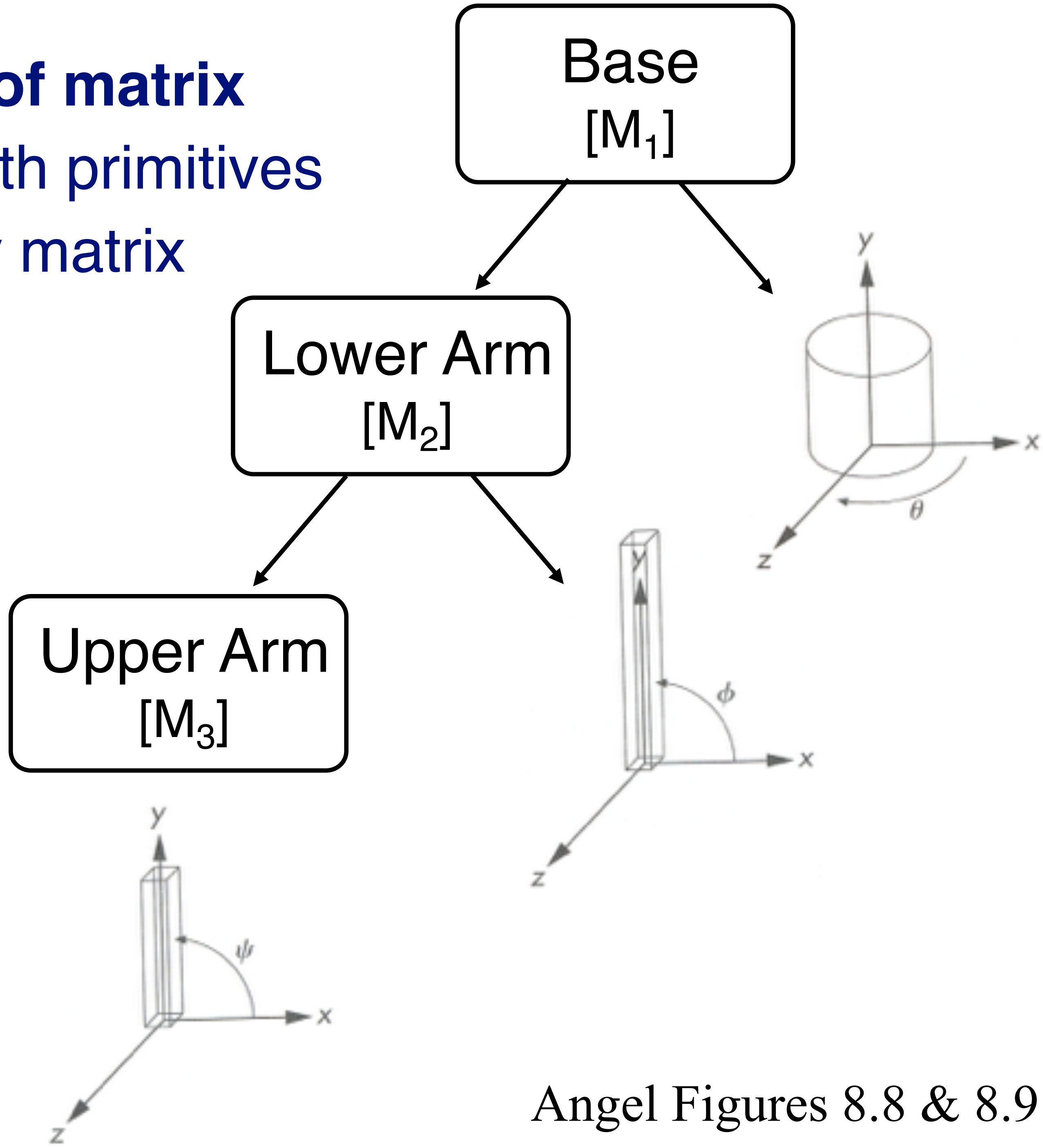


Ray Casting With Hierarchies

- Transform rays, not primitives
  - For each node ...
    - » **Transform ray by inverse of matrix**
    - » Intersect transformed ray with primitives
    - » Transform hit information by matrix



Robot Arm



Angel Figures 8.8 & 8.9

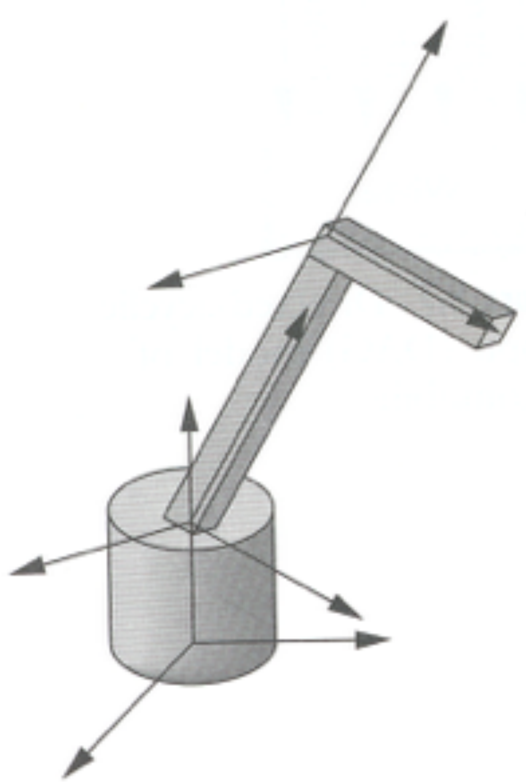
Transforming a Ray

- If M is the transformation mapping a scene-graph node into the "world" (or "global") coordinate system, then we transform a ray r by:
  - $r'.\text{start} = M^{-1}(r.\text{start})$
  - $r'.\text{direction} = M_L^{-1}(r.\text{direction})$

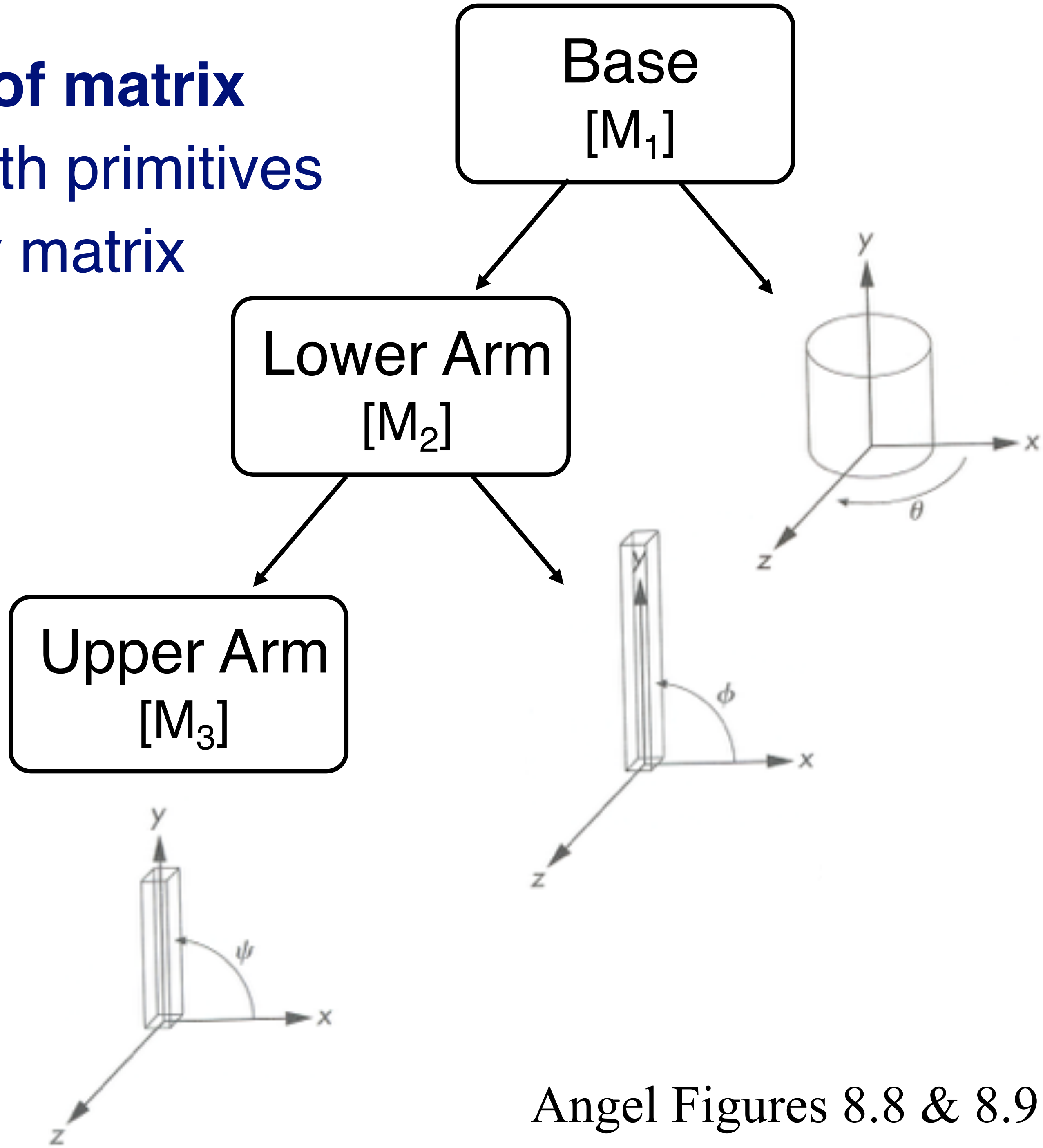
$$\underset{M}{\overset{\text{Affine}}{\begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}}$$

Ray Casting With Hierarchies

- Transform rays, not primitives
  - For each node ...
    - **Transform ray by inverse of matrix**
    - Intersect transformed ray with primitives
    - Transform hit information by matrix



Robot Arm



Angel Figures 8.8 & 8.9

Applying a Transformation

- Position
- Direction
- Normal

$p' = ?$

$$\underset{M}{\overset{\text{Affine}}{\begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}}$$

Normal Transformation

2D Example:

$$\underset{M}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}}$$

Normal Transformation

2D Example:

$$\underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}}_{M} = \underbrace{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}}_{M_T} \times \underbrace{\overset{\text{Scale}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}}_{M_L}$$

If *v* is a direction in 2D, and *n* is a vector perpendicular to *v*, we want the transformed *n* to be perpendicular to the transformed *v*:

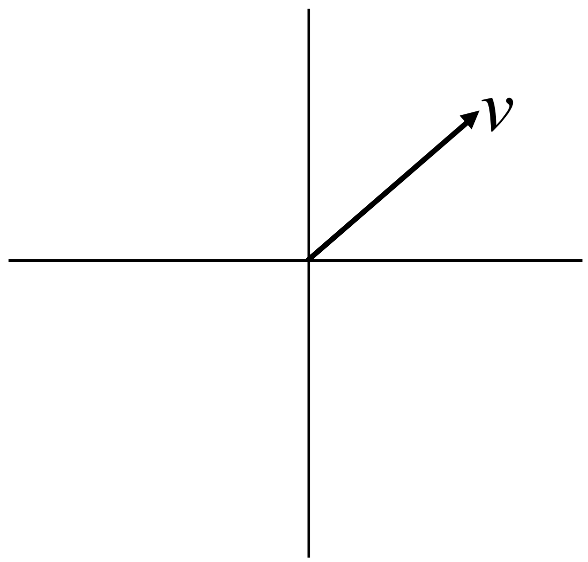
$$\hat{v} \cdot \hat{n} = 0 \quad \Longrightarrow \quad M_L(\hat{v}) \cdot \hat{n}' = 0$$

Normal Transformation

2D Example:

$$\underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}}_{M} = \underbrace{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}}_{M_T} \times \underbrace{\overset{\text{Scale}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}}_{M_L}$$

Say $\hat{v} = (2, 2)$ …

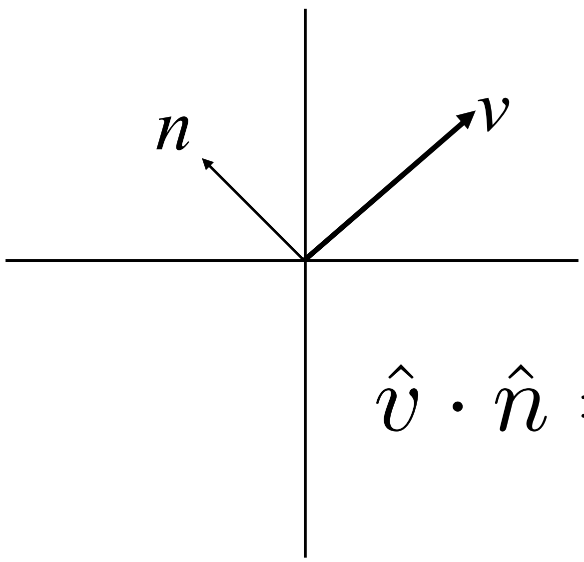


Normal Transformation

2D Example:

$$\underset{M}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}}$$

Say $\hat{v} = (2, 2)$ … then $\hat{n} = (-\sqrt{.5}, \sqrt{.5})$



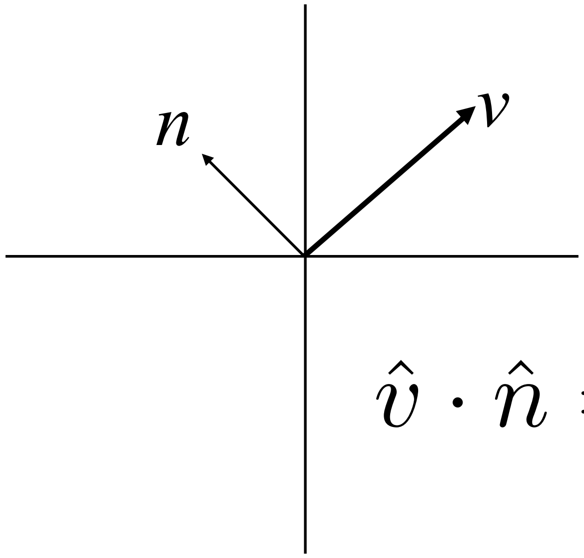
$$\hat{v} \cdot \hat{n} = 0$$

Normal Transformation

2D Example:

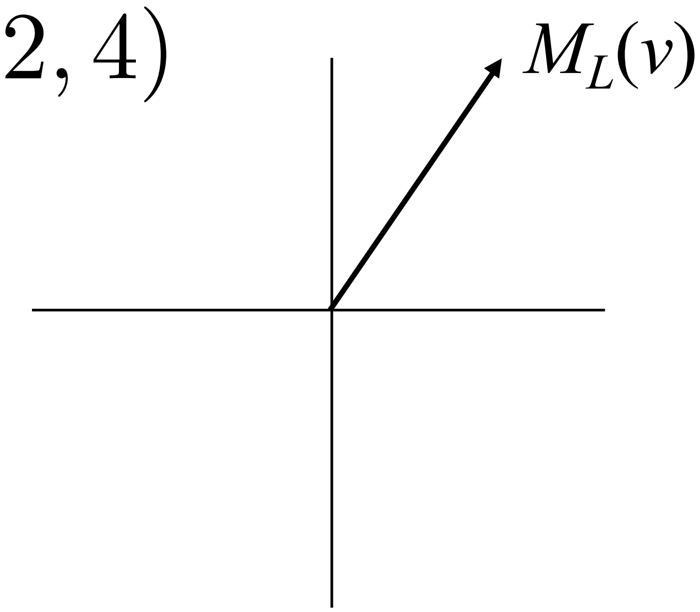
$$\underset{M}{\begin{bmatrix}1 & 0 & 1\\0 & 2 & 1\\0 & 0 & 1\end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix}1 & 0 & 1\\0 & 1 & 1\\0 & 0 & 1\end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix}1 & 0 & 0\\0 & 2 & 0\\0 & 0 & 1\end{bmatrix}}}$$

Say $\hat{v} = (2, 2)$ … then $\hat{n} = (-\sqrt{.5}, \sqrt{.5})$



$$\hat{v} \cdot \hat{n} = 0$$

$$M_L(\hat{v}) = (2, 4)$$



Normal Transformation

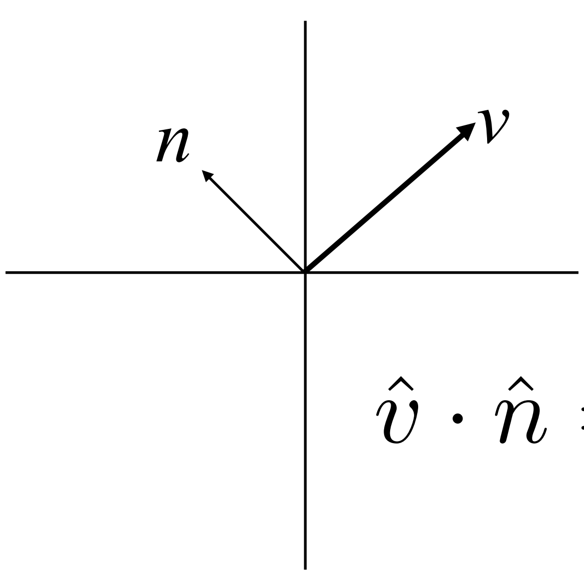
2D Example:

$$\underset{M}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}}$$

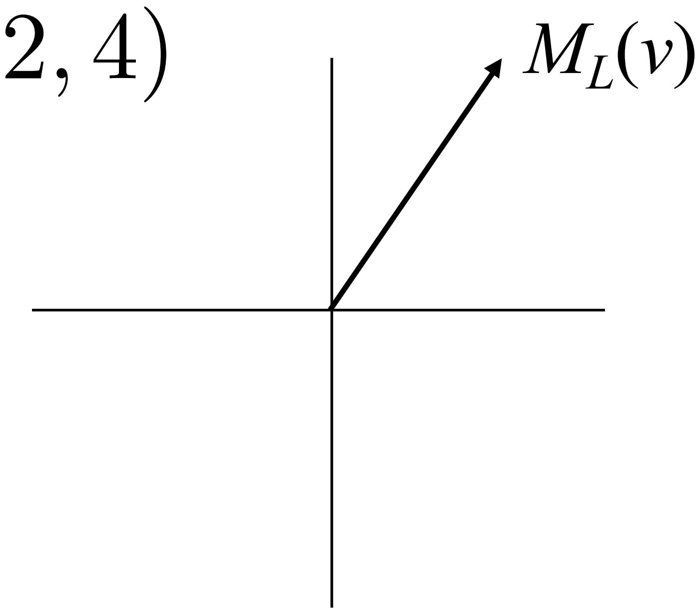
Say $\hat{v} = (2, 2)$ … then $\hat{n} = (-\sqrt{.5}, \sqrt{.5})$

$$M_L(\hat{n}) = (-\sqrt{.5}, \sqrt{2})$$

$$M_L(\hat{v}) = (2, 4)$$



$$\hat{v} \cdot \hat{n} = 0$$



Normal Transformation

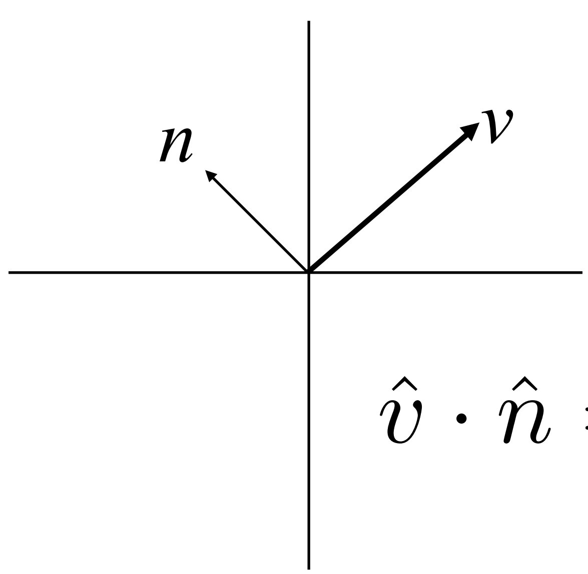
2D Example:

$$\underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}_{M_T \text{ (Translate)}} \times \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{M_L \text{ (Scale)}}$$

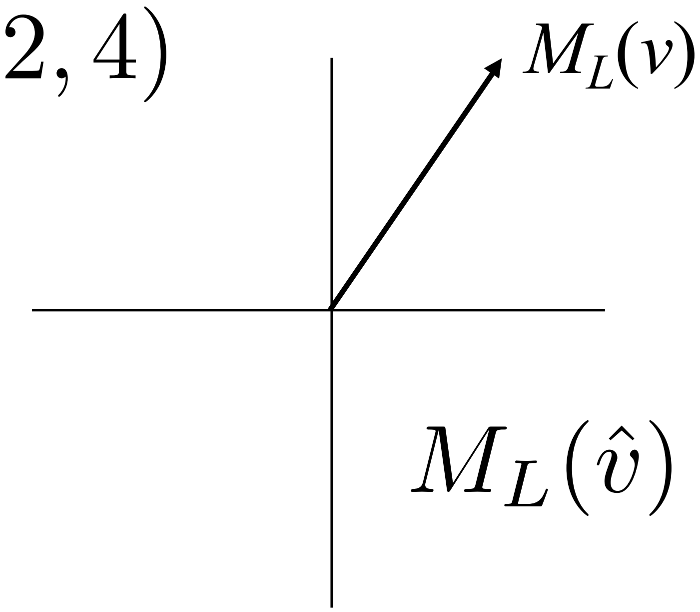
Say $\hat{v} = (2, 2)$ … then $\hat{n} = (-\sqrt{.5}, \sqrt{.5})$

$$M_L(\hat{n}) = (-\sqrt{.5}, \sqrt{2})$$

$$M_L(\hat{v}) = (2, 4)$$



$$\hat{v} \cdot \hat{n} = 0$$



$$M_L(\hat{v}) \cdot M_L(\hat{n}) \neq 0$$

Normal Transformation

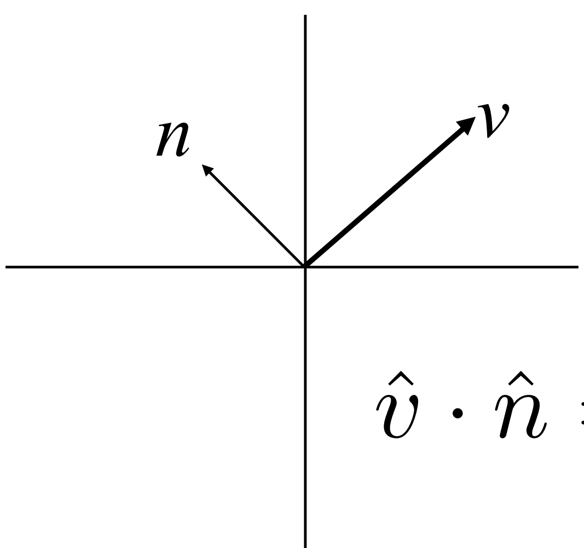
2D Example:

$$\underset{M}{\begin{bmatrix}1&0&1\\0&2&1\\0&0&1\end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix}1&0&1\\0&1&1\\0&0&1\end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix}1&0&0\\0&2&0\\0&0&1\end{bmatrix}}}$$

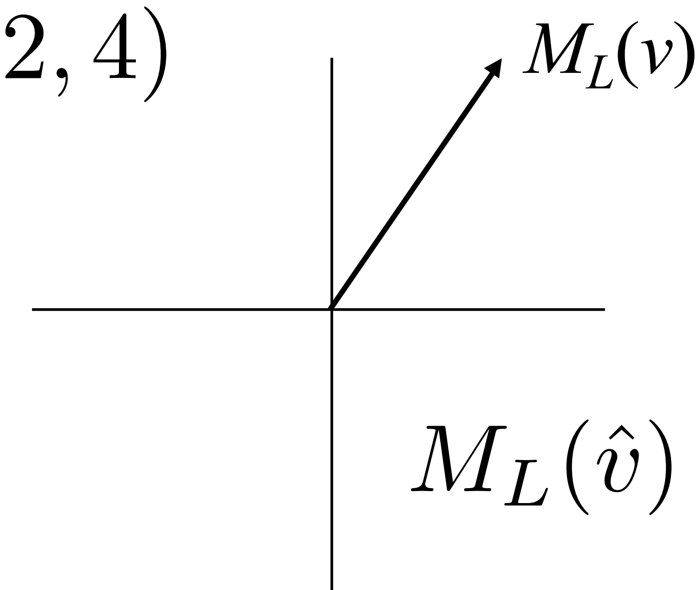
Simply applying the directional part of the transformation to *n* does not result in a vector that is perpendicular to the transformed *v*.

Say $\hat{v} =$ … $-\sqrt{.5}, \sqrt{2})$

$$M_L(\hat{v}) = (2, 4)$$



$$\hat{v} \cdot \hat{n} = 0$$



$$M_L(\hat{v}) \cdot M_L(\hat{n}) \neq 0$$

Recall

Transposes:

- The transpose of a matrix *M* is the matrix *M^t^* whose (i,j)-th coeff. is the (j,i)-th coeff. of M:

$$M = \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{bmatrix} \qquad M^t = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & 2m_{33} \end{bmatrix}$$

Recall

Transposes:

- The transpose of a matrix *M* is the matrix M^t whose (i,j)-th coeff. is the (j,i)-th coeff. of M:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \qquad M^t = \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{bmatrix}$$

- If *M* and *N* are two matrices, then the transpose of the product is the inverted product of the transposes:

$$(MN)^t = N^t M^t$$

Recall

Dot-Products :

- The dot product of two vectors $v=(v_x,v_y,v_z)$ and $w=(w_x,w_y,w_z)$ is obtained by summing the product of the coefficients:

$$\hat{v} \cdot \hat{w} = v_x w_x + v_y w_y + v_z w_z$$

Recall

Dot-Products :

- The dot product of two vectors $v=(v_x,v_y,v_z)$ and $w=(w_x,w_y,w_z)$ is obtained by summing the product of the coefficients:

$$\hat{v} \cdot \hat{w} = v_x w_x + v_y w_y + v_z w_z$$

- Can also express as a matrix product:

$$\hat{v} \cdot \hat{w} = \hat{v}^t \hat{w} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}$$

Recall

Transposes and Dot-Products:

- If M is a matrix, the dot product of v with M applied to w is the dot product of the transpose of M applied to v with w:

Recall

Transposes and Dot-Products:

- If *M* is a matrix, the dot product of *v* with *M* applied to *w* is the dot product of the transpose of *M* applied to *v* with *w*:

$$\langle v, Mw \rangle = v^t M w$$

Recall

Transposes and Dot-Products:

- If M is a matrix, the dot product of v with M applied to w is the dot product of the transpose of M applied to v with w:

$$\begin{aligned}\langle v, Mw\rangle &= v^t M w \\ &= (v^t M) w\end{aligned}$$

Recall

Transposes and Dot-Products:

- If M is a matrix, the dot product of v with M applied to w is the dot product of the transpose of M applied to v with w:

$$\begin{aligned}\langle v, Mw\rangle &= v^t M w\\ &= (v^t M) w\\ &= (M^t v)^t w\end{aligned}$$

Recall

Transposes and Dot-Products:

- If *M* is a matrix, the dot product of *v* with *M* applied to *w* is the dot product of the transpose of *M* applied to *v* with *w*:

$$
\begin{aligned}
\langle v, Mw \rangle &= v^t M w \\
&= (v^t M) w \\
&= (M^t v)^t w \\
\langle v, Mw \rangle &= \langle M^t v, w \rangle
\end{aligned}
$$

Applying a Transformation

- If we apply the transformation M to 3D space, how does it act on normals?

Applying a Transformation

- If we apply the transformation *M* to 3D space, how does it act on normals?
- A normal *n* is defined by being perpendicular to some vector(s) *v*. The transformed normal *n'* should be perpendicular to *M*(*v*):

$$\langle n, v \rangle = \langle n', Mv \rangle$$

Applying a Transformation

- If we apply the transformation *M* to 3D space, how does it act on normals?
- A normal *n* is defined by being perpendicular to some vector(s) *v*. The transformed normal *n'* should be perpendicular to *M*(*v*):

$$\begin{aligned}\langle n, v\rangle &= \langle n', Mv\rangle \\ &= \langle M^t n', v\rangle\end{aligned}$$

Applying a Transformation

- If we apply the transformation *M* to 3D space, how does it act on normals?

- A normal *n* is defined by being perpendicular to some vector(s) *v*. The transformed normal *n'* should be perpendicular to *M*(*v*):

$$\begin{aligned}\langle n, v\rangle &= \langle n', Mv\rangle \\ &= \langle M^t n', v\rangle\end{aligned}$$

$$\Updownarrow$$

$$n = M^t n'$$

Applying a Transformation

- If we apply the transformation *M* to 3D space, how does it act on normals?
- A normal *n* is defined by being perpendicular to some vector(s) *v*. The transformed normal *n'* should be perpendicular to *M*(*v*):

$$\begin{aligned} \langle n, v \rangle &= \langle n', Mv \rangle \\ &= \langle M^t n', v \rangle \end{aligned}$$

$$\Updownarrow$$

$$n = M^t n'$$
$$n' = (M^t)^{-1} n$$

Applying a Transformation

- Position

  $p' = M(p)$

- Direction

  $p' = M_L(p)$

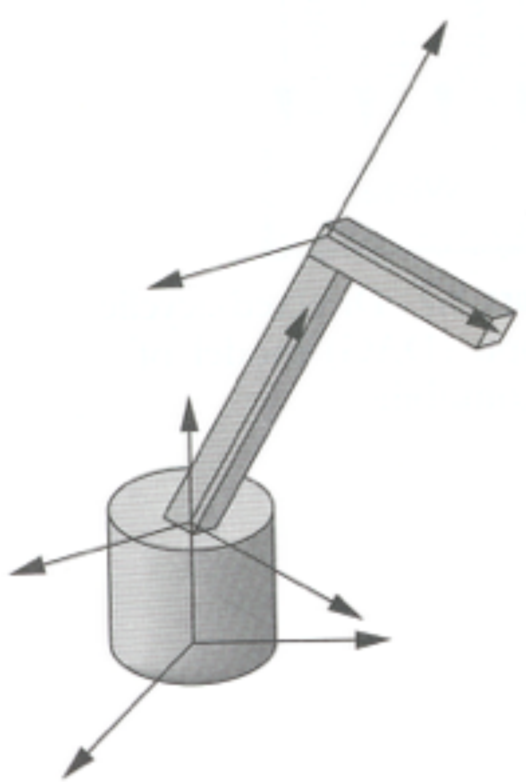
- Normal

  $p' = ((M_L)^t)^{-1}(p)$

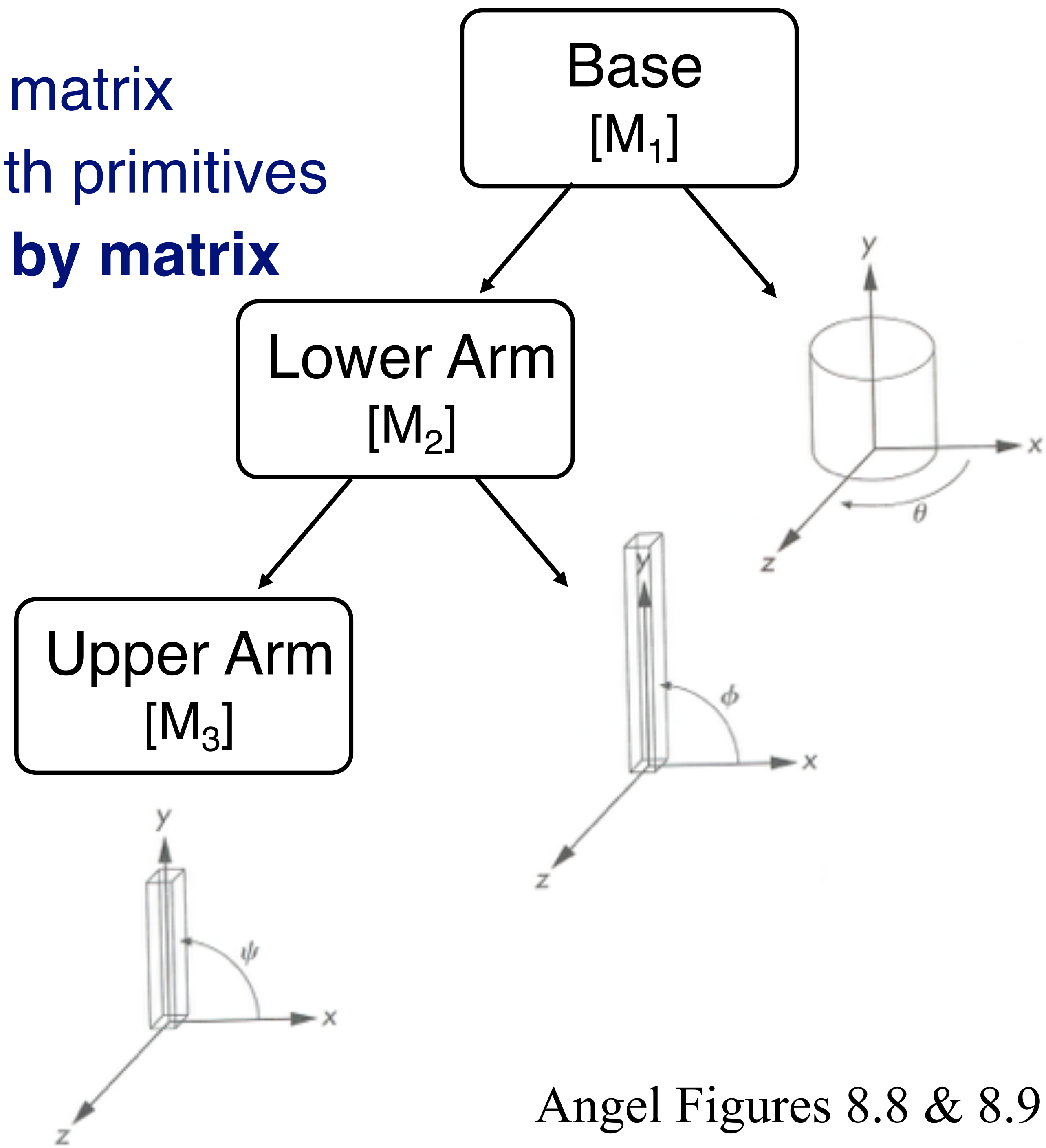
$$
\underset{M}{\overset{\text{Affine}}{\begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}}
$$

Ray Casting With Hierarchies

- Transform rays, not primitives
  - For each node ...
    - » Transform ray by inverse of matrix
    - » Intersect transformed ray with primitives
    - » **Transform hit information by matrix**



Robot Arm



Angel Figures 8.8 & 8.9

Transforming a Ray

- If M is the transformation mapping a scene-graph node into the global coordinate system, then we transform the hit information *hit* by:
  - *hit'*.position = M(*hit*.position)
  - *hit'*.normal = $((M_L)^t)^{-1}$(*hit*.normal)

$$\underset{M}{\overset{\text{Affine}}{\begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}}$$