Scene Graphs and Barycentric Coordinates

Michael Kazhdan

(601.457/657)

Last Time

2D Transformations

- Basic 2D transformations
- Matrix representation
- Matrix composition

3D Transformations

- Basic 3D transformations
- Same as 2D

Homogeneous Coordinates

Represent transformations by 4×4 matrices

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- The top-left 3×3 block represents the linear part of the transformation
- The last column represents the translation
- Transformations (translations/rotations/scales) can be composed using simple matrix multiplication

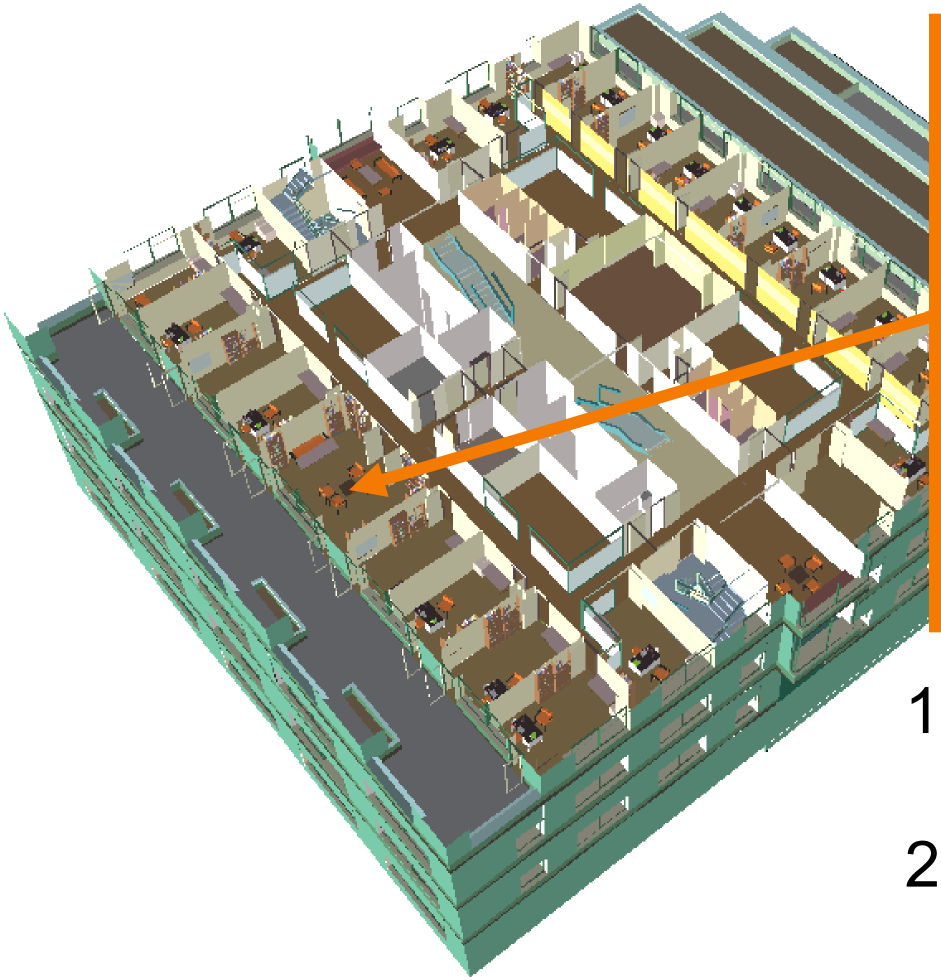
Overview

Transformation Hierarchies
- Scene graphs
- Ray casting

Barycentric Coordinates

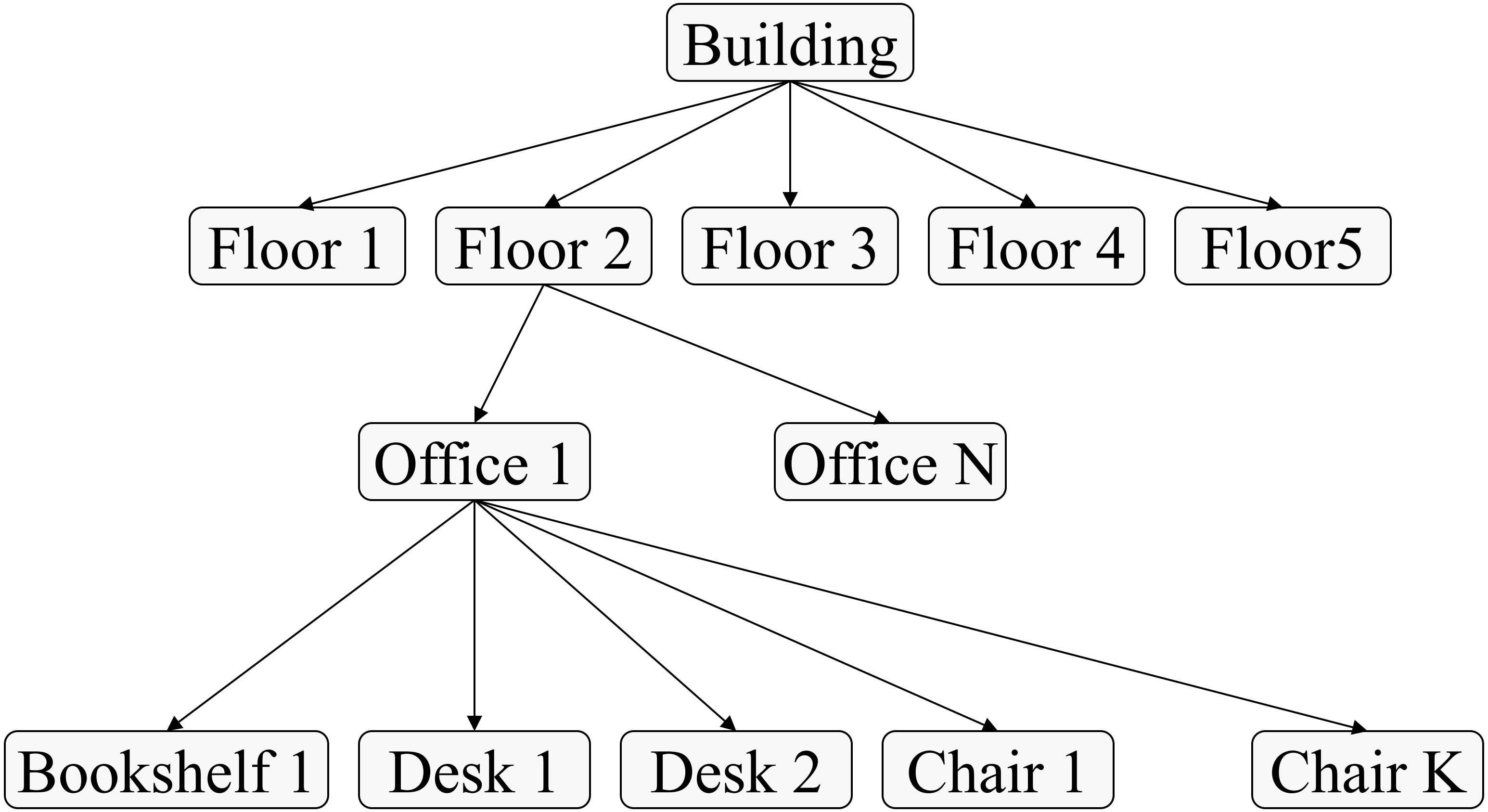
Scene Graphs: Static Environments

An object may appear in a scene multiple times

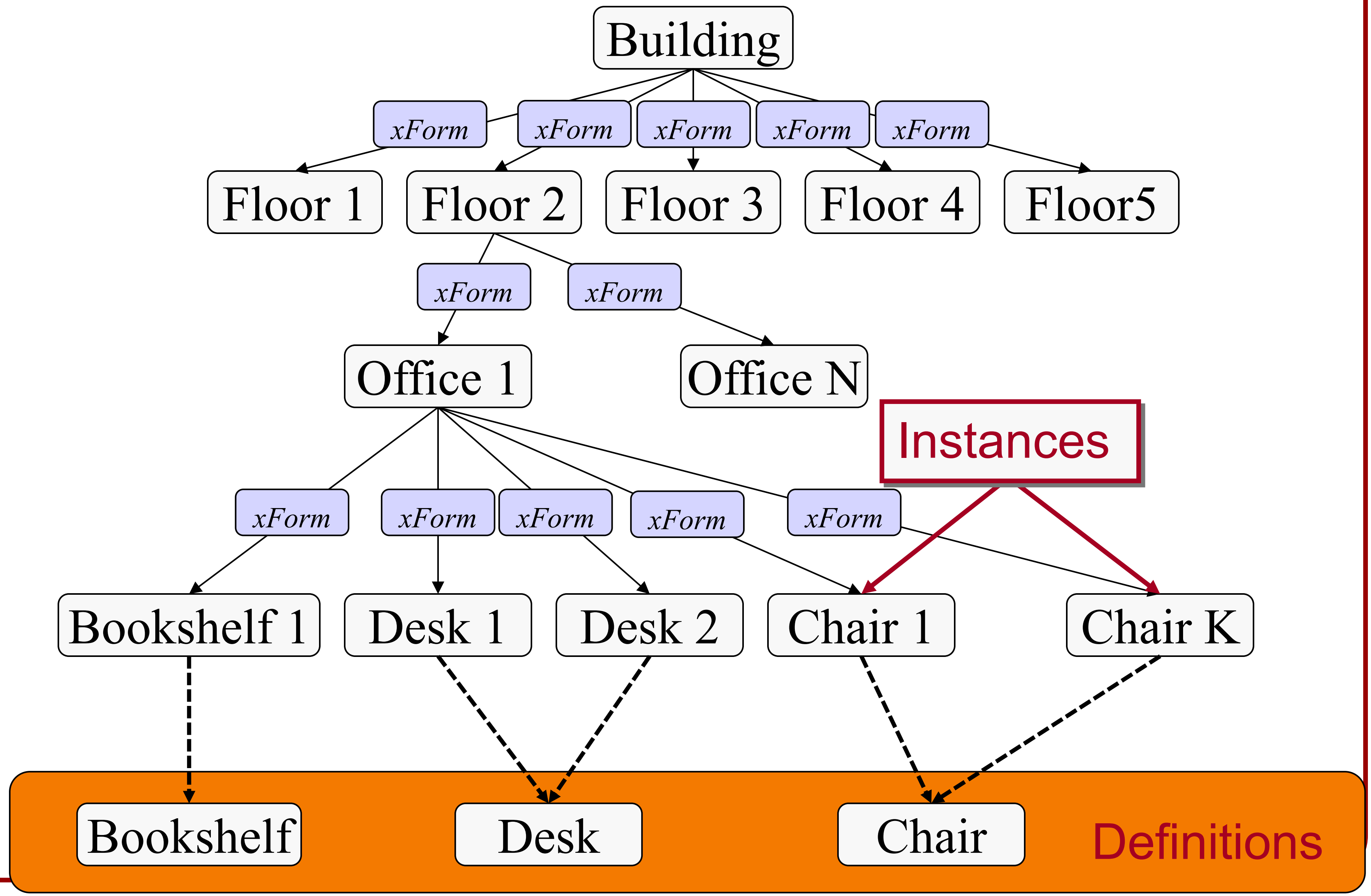


1. Model objects in their own coordinate systems
2. Draw same 3D data with different transformations

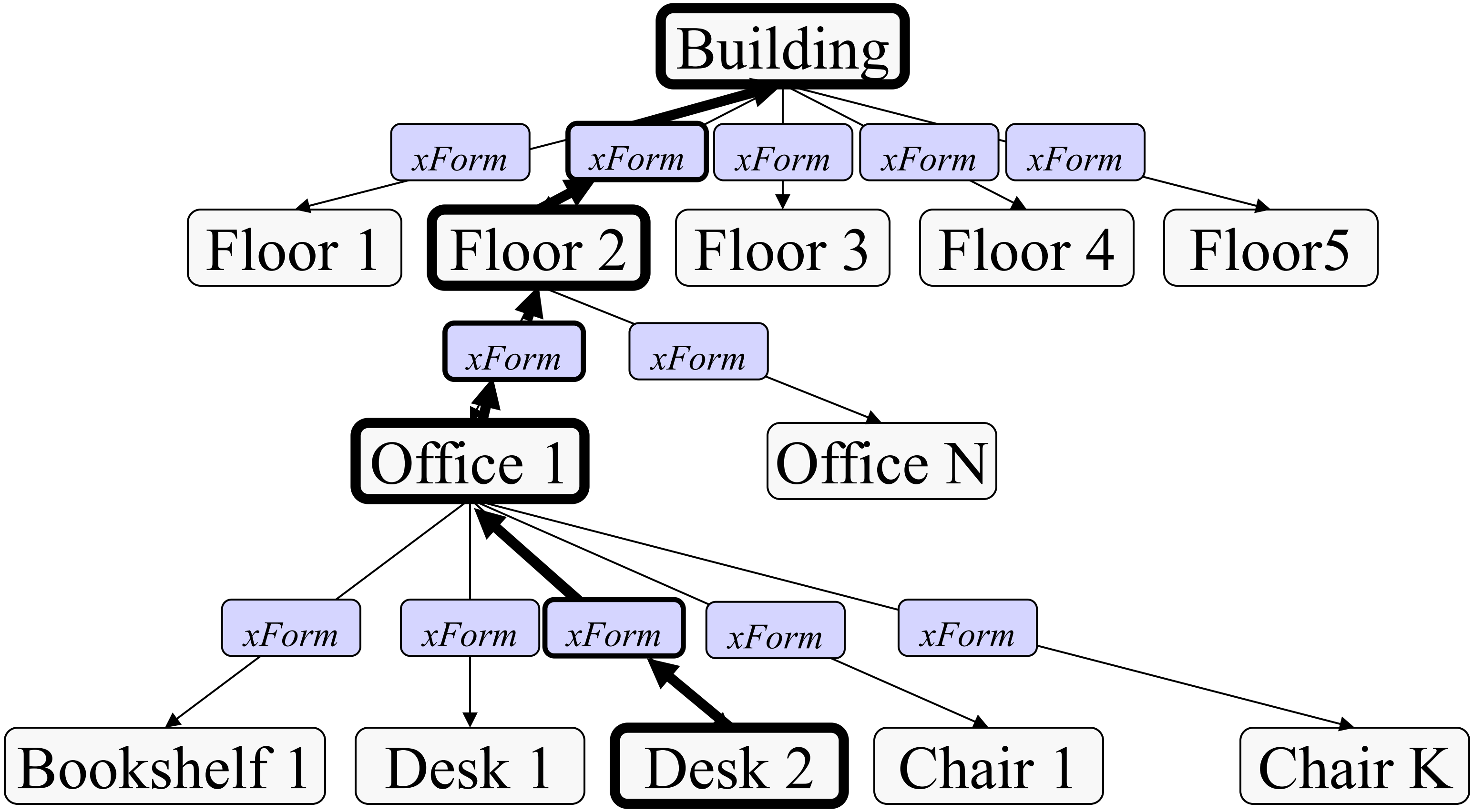
Scene Graphs: Static Environments



Scene Graphs: Static Environments



Scene Graphs: Static Environments

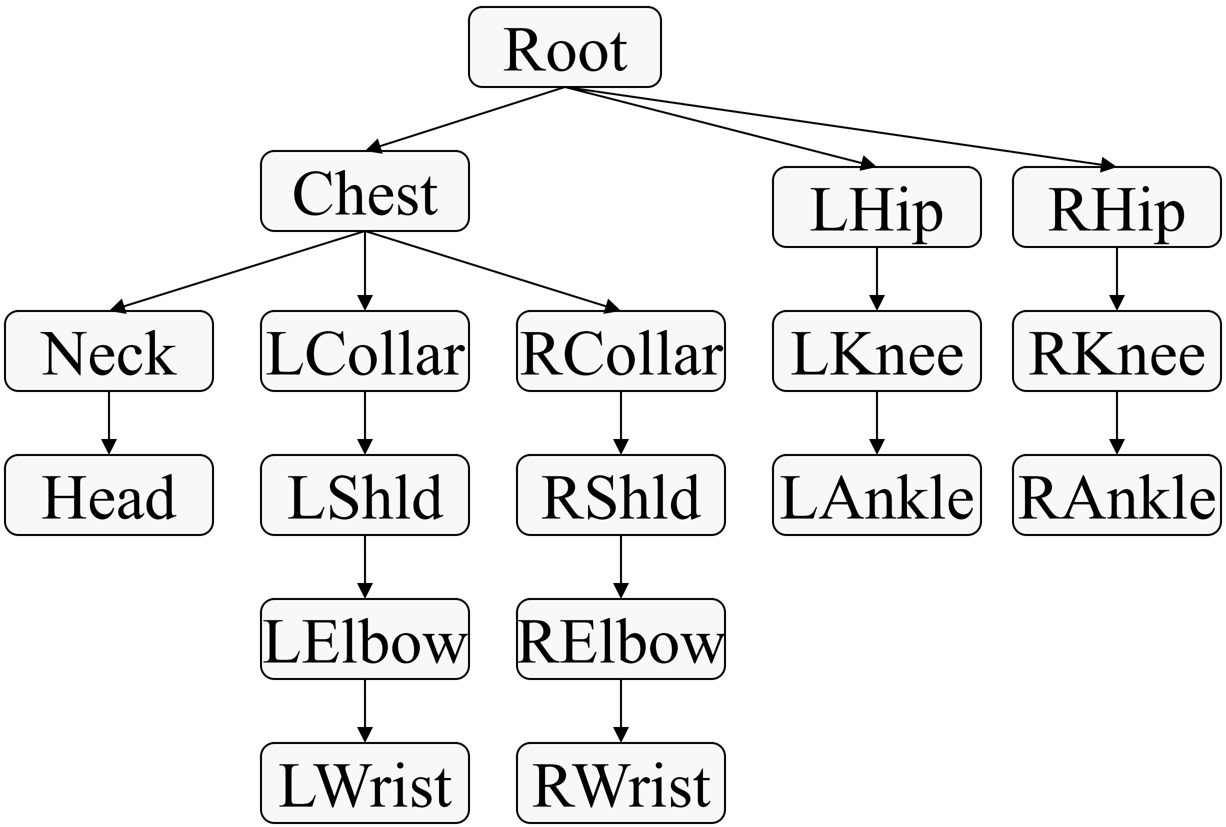


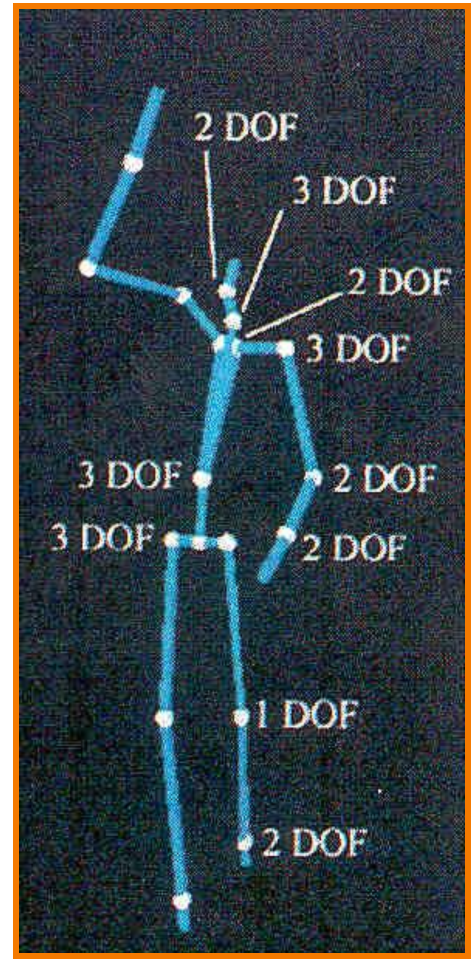
The transformation applied to a node of the scene graph is the composition of transformations along the path to the root.

⇒ Global transformations only requires transforming the root

Scene Graphs: Dynamic Characters

Well-suited for articulated characters





Rose et al. `96

Scene Graphs

- **Local Modeling**

  Allows modeling in local coordinates and then placing into a global frame – particularly important for animation

- **Instancing**

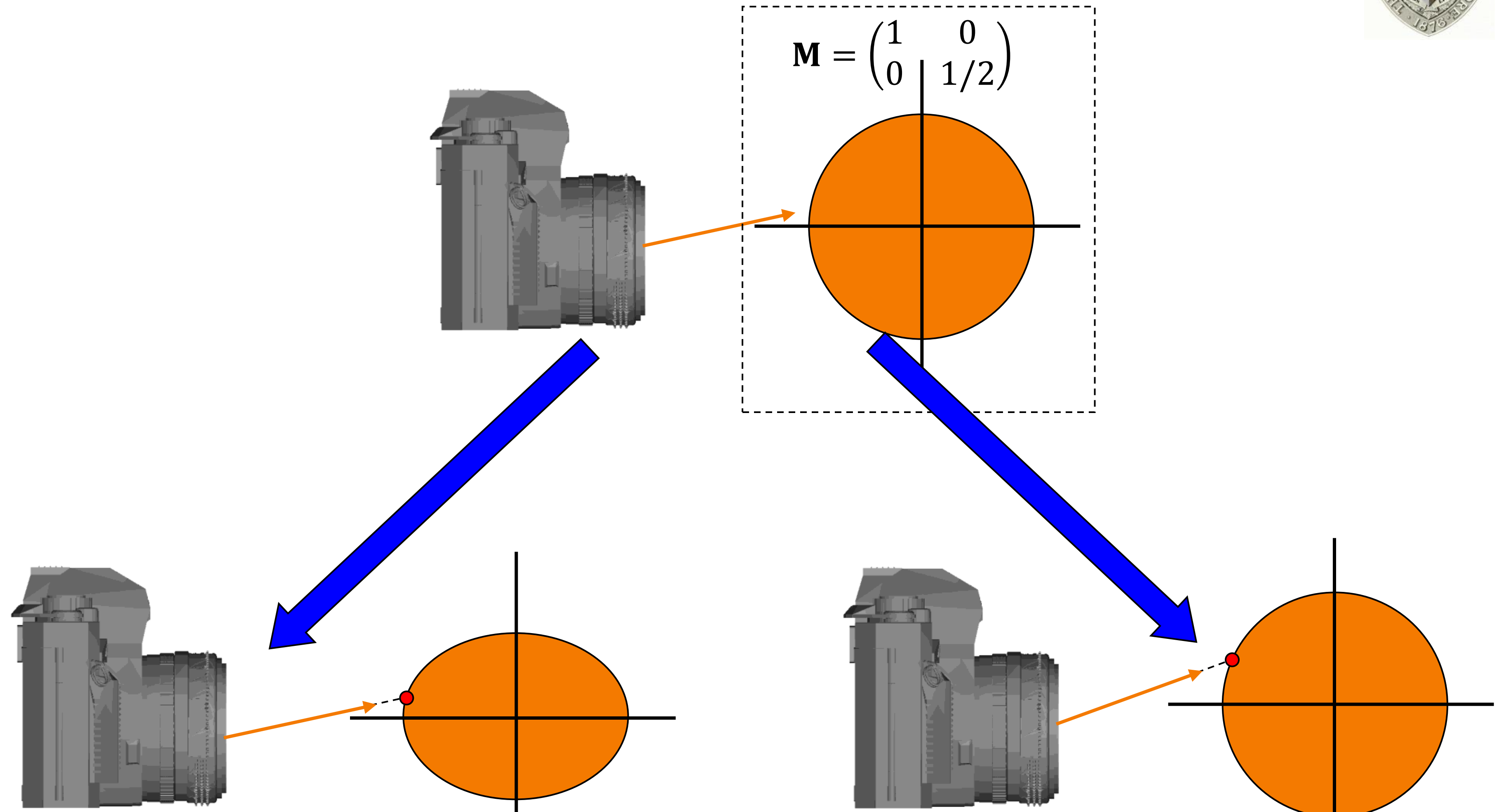
  Allows multiple instances of a single model – reducing storage and facilitating making consistent changes

- **Hierarchical Representation**

  Facilitates modifying the scene by having geometry moved with its context

  Accelerates ray-tracing by providing a hierarchy that can be used for bounding volume testing

Ray Casting



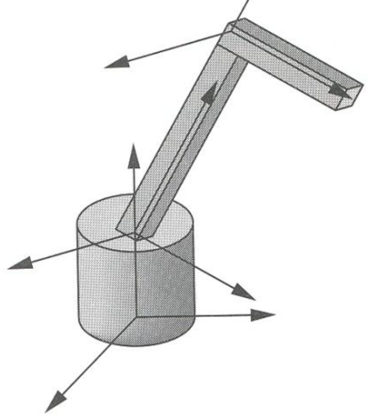
- Transform the shape ($\mathbf{M}$)
- Compute the intersection

- Transform the ray ($\mathbf{M}^{-1}$)
- Compute the intersection
- Transform the intersection ($\mathbf{M}$)

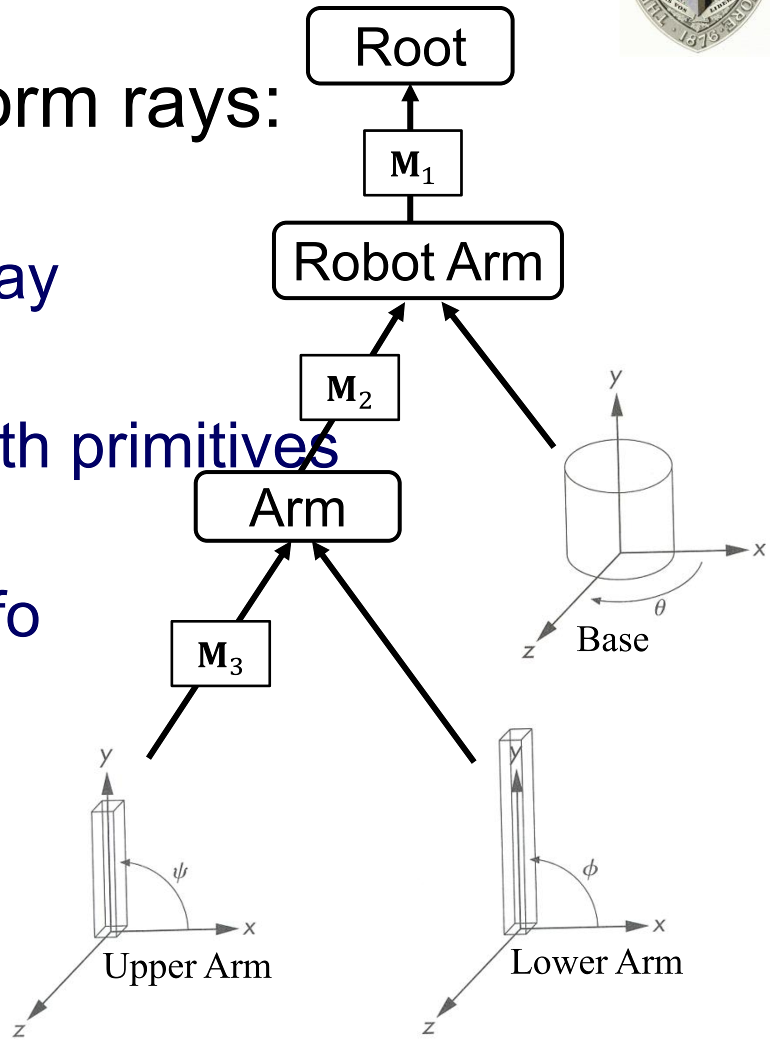
Ray Casting with Hierarchies

Iterating from the root, transform rays:

- world → model:
  - Apply inverse transform to ray
- model coordinates:
  - Intersect transformed ray with primitives
- model → world:
  - Apply transform to the hit info



Robot Arm

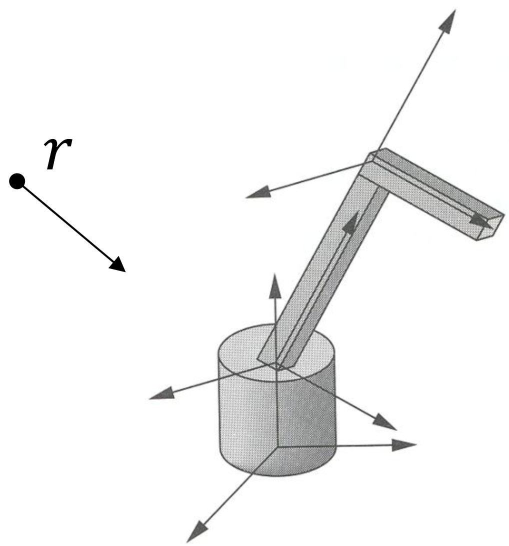


Angel Figures 8.8 & 8.9

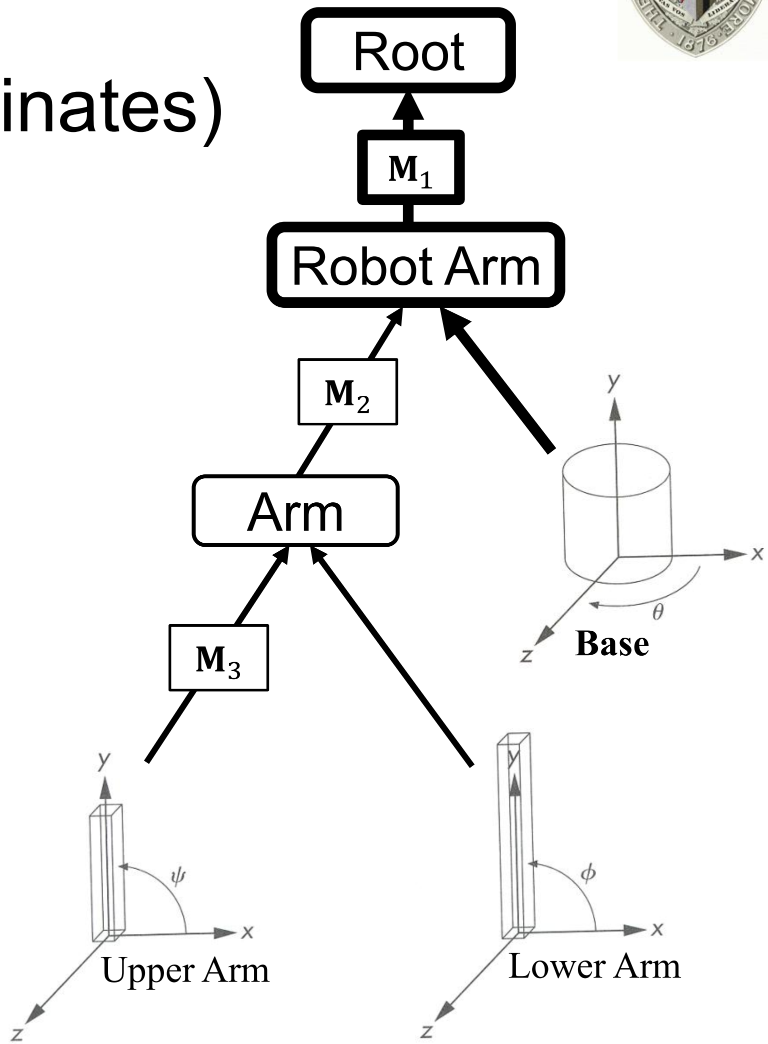
Ray Casting with Hierarchies

Given a ray r (in global coordinates)

- **Base**: Apply $\mathbf{M}_1^{-1}$ to the ray and test for intersection



Robot Arm

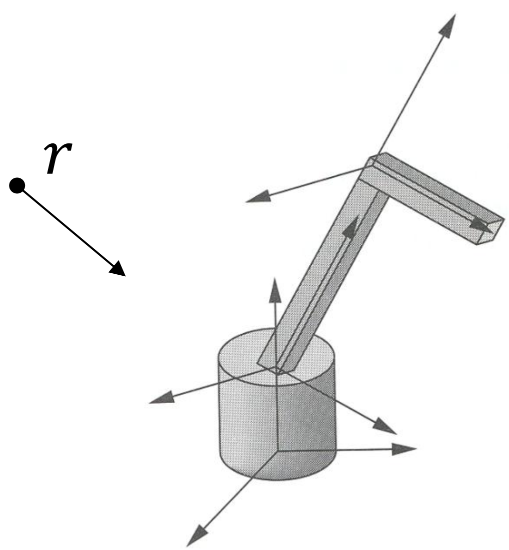


Angel Figures 8.8 & 8.9

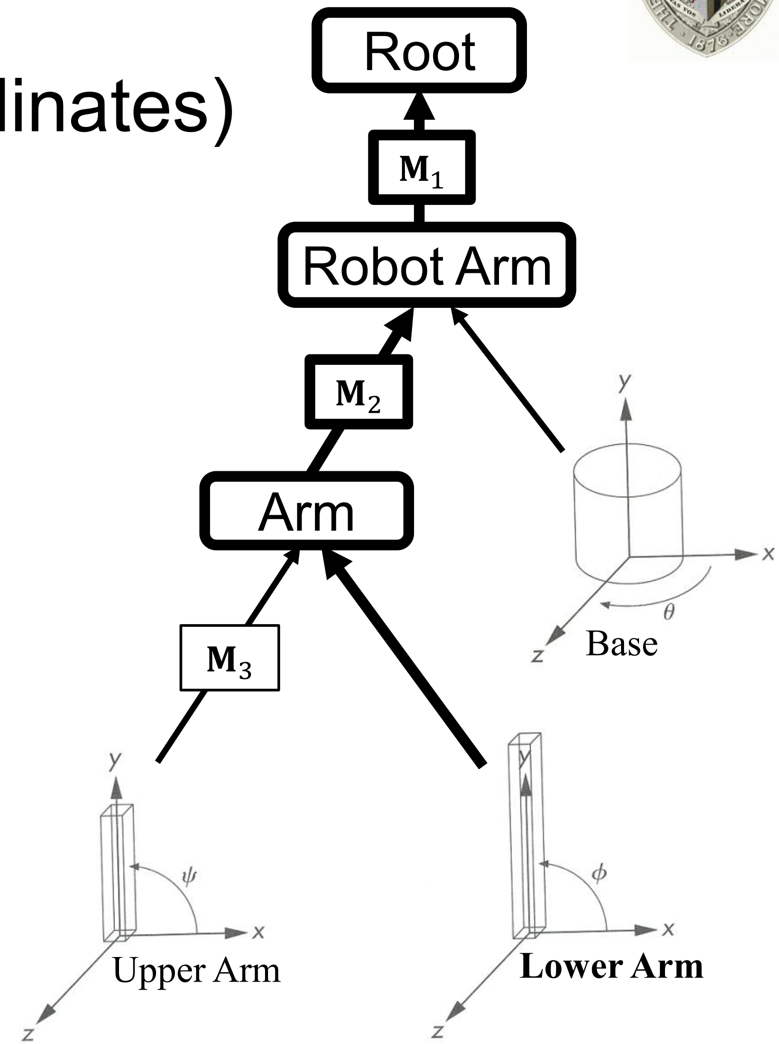
Ray Casting with Hierarchies

Given a ray r (in global coordinates)

- **Base**: Apply $\mathbf{M}_1^{-1}$ to the ray and test for intersection
- **Lower Arm**: Apply $(\mathbf{M}_1 \cdot \mathbf{M}_2)^{-1}$ to the ray and test for intersection



Robot Arm

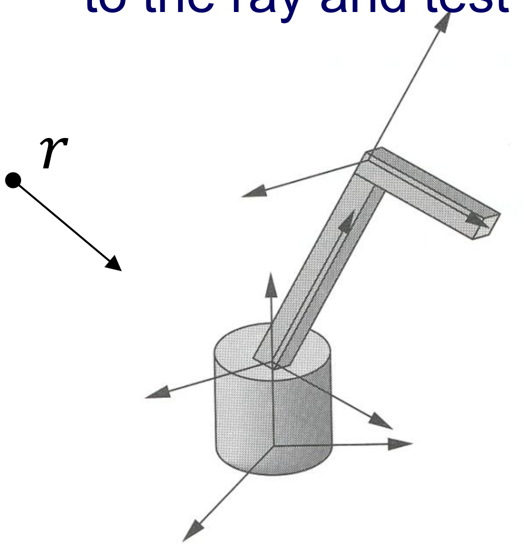


Angel Figures 8.8 & 8.9

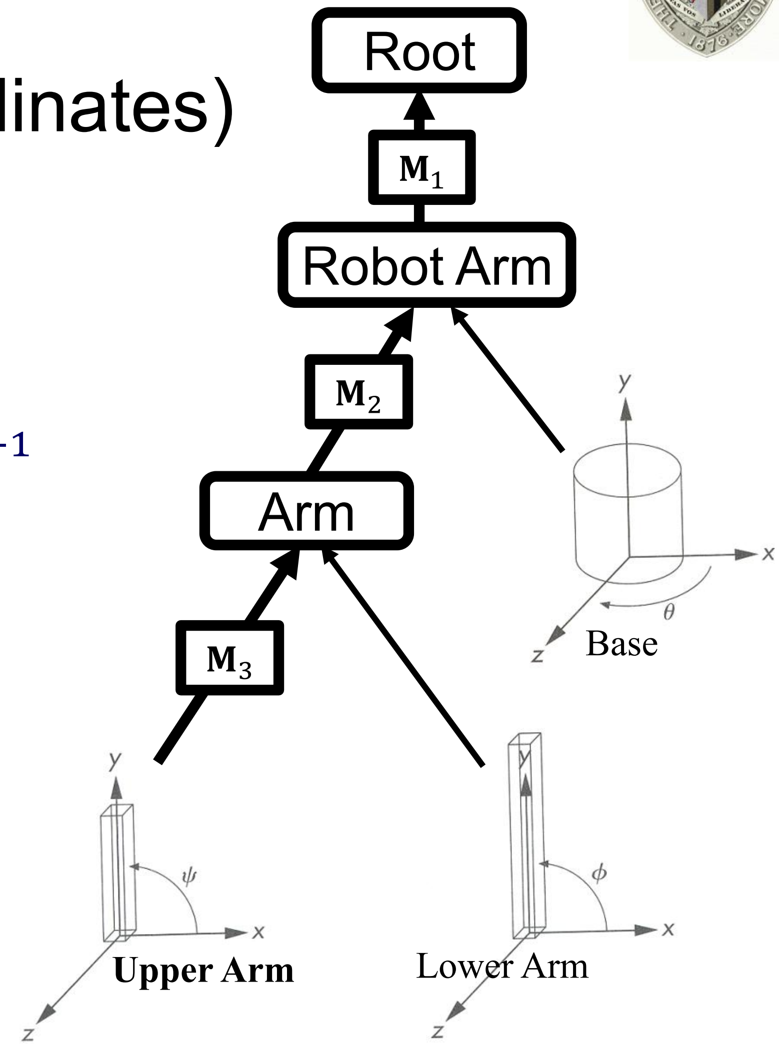
Ray Casting with Hierarchies

Given a ray r (in global coordinates)

- **Base**: Apply $\mathbf{M}_1^{-1}$ to the ray and test for intersection
- **Lower Arm**: Apply $(\mathbf{M}_1 \cdot \mathbf{M}_2)^{-1}$ to the ray and test for intersection
- **Upper Arm**: Apply $(\mathbf{M}_1 \cdot \mathbf{M}_2 \cdot \mathbf{M}_3)^{-1}$ to the ray and test for intersection



Robot Arm

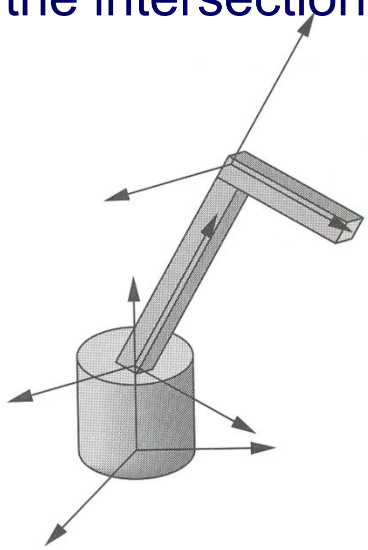


Angel Figures 8.8 & 8.9

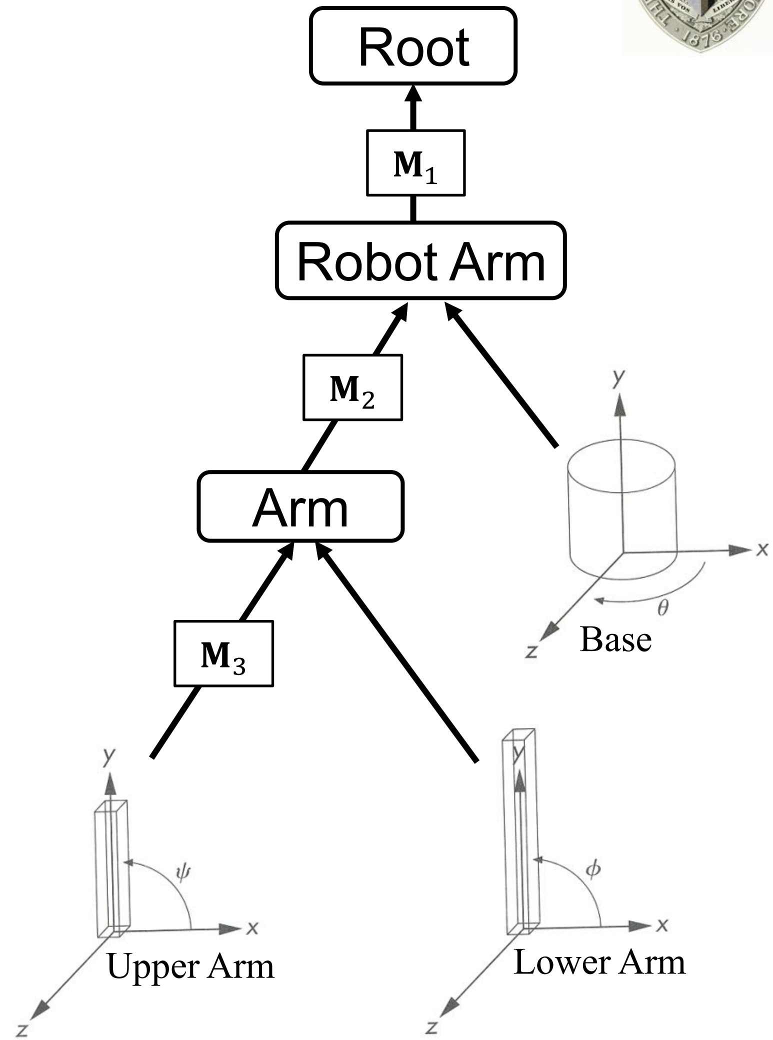
Ray Casting with Hierarchies

If there is an intersection:

- **Base**: Apply $\mathbf{M}_1$ to the intersection information
- **Lower Arm**: Apply $\mathbf{M}_1 \cdot \mathbf{M}_2$ to the intersection information
- **Upper Arm**: Apply $\mathbf{M}_1 \cdot \mathbf{M}_2 \cdot \mathbf{M}_3$ to the intersection information



Robot Arm



Angel Figures 8.8 & 8.9

Applying a Transformation

Position

Direction

Normal

$$\underset{\mathbf{M}}{\overset{\text{Affine}}{\begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}} = \underset{\mathbf{M}_T}{\overset{\text{Translate}}{\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}} \cdot \underset{\mathbf{M}_L}{\overset{\text{Linear}}{\begin{pmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}}$$

Applying a Transformation

Position

- Apply the full affine transformation:

$$\mathbf{p}' = \mathbf{M} \cdot \mathbf{p} = (\mathbf{M}_T \cdot \mathbf{M}_L) \cdot \mathbf{p}$$

Direction

Normal

$$\underbrace{\begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{M}}^{\text{Affine}} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{M}_T}^{\text{Translate}} \cdot \underbrace{\begin{pmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{M}_L}^{\text{Linear}}$$

Applying a Transformation

Position

Direction

- Apply the linear component of the transformation:

$$\vec{\mathbf{v}}' = \mathbf{M}_L \cdot \vec{\mathbf{v}}$$

Normal

$$\underset{\mathbf{M}}{\overset{\text{Affine}}{\begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}} = \underset{\mathbf{M}_T}{\overset{\text{Translate}}{\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}} \cdot \underset{\mathbf{M}_L}{\overset{\text{Linear}}{\begin{pmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}}$$

Applying a Transformation

Position

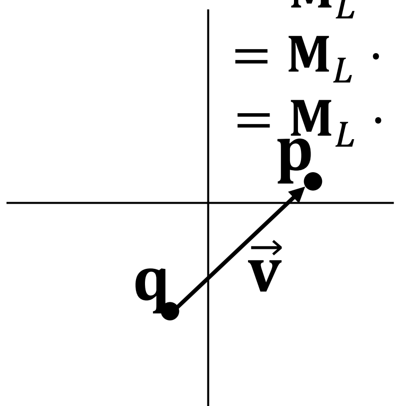
Direction

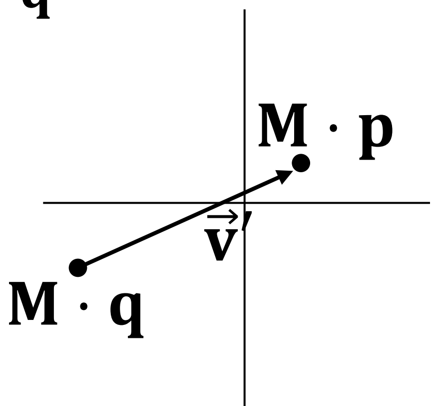
- Apply the linear component of the transformation:

$$\vec{\mathbf{v}}' = \mathbf{M}_L \cdot \vec{\mathbf{v}}$$

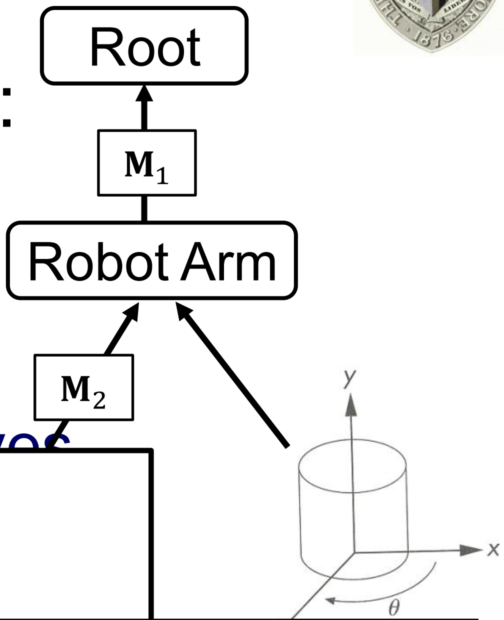
A direction $\vec{v}$ represents the difference between two positions: $\vec{\mathbf{v}} = \mathbf{p} - \mathbf{q}$.
The transformed direction is the difference of transformed positions:

$$\begin{aligned}\vec{\mathbf{v}}' &= \mathbf{M} \cdot \mathbf{p} - \mathbf{M} \cdot \mathbf{q} \\ &= (\mathbf{M}_L \cdot \mathbf{p} + \vec{\mathbf{t}}) - \left(\mathbf{M}_L \cdot \mathbf{q} + \vec{\mathbf{t}}\right) \\ &= \mathbf{M}_L \cdot \mathbf{p} - \mathbf{M}_L \cdot \mathbf{q} \\ &= \mathbf{M}_L \cdot (\mathbf{p} - \mathbf{q}) \\ &= \mathbf{M}_L \cdot \vec{\mathbf{v}}\end{aligned}$$





Ray Casting With Hierarchies



Iterating from the root, transform rays:

world → model:

Apply inverse transform to ray

model coordinates:

Intersect transformed ray with primitives

model –

Appl

Ray Transformation:

$$(\mathbf{p}, \vec{\mathbf{v}}) \to (\mathbf{M}^{-1} \cdot \mathbf{p}, \mathbf{M}_L^{-1} \cdot \vec{\mathbf{v}})$$

Note:

- When the ray direction is unit-length, time travelled along the ray is the same as distance travelled along the ray.
- Even if the original ray direction, $\vec{\mathbf{v}}$, was unit-length, the transformed ray direction, $\mathbf{M}_L^{-1} \cdot \vec{\mathbf{v}}$, may not be.

Robot Arm

Angel Figures 8.8 & 8.9

Ray Casting With Hierarchies

Iterating from the root, transform rays:

world → model:

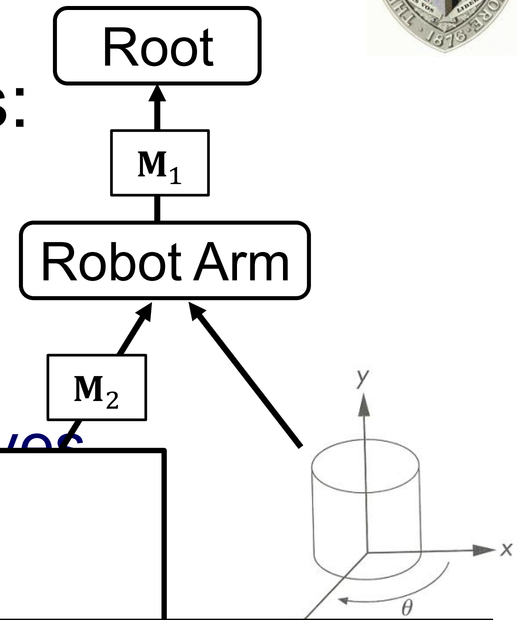
Apply inverse transform to ray

model coordinates:

Intersect transformed ray with primitives

model –

Apply



Ray Transformation:

$$(\mathbf{p},\vec{\mathbf{v}}) \to (\mathbf{M}^{-1}\cdot\mathbf{p}, \mathbf{M}_L^{-1}\cdot\vec{\mathbf{v}})$$

Note:

- When the ray direction is unit-length, time travelled along the ray is the same as distance travelled along the ray.
- E
tr

Recall:

For acceleration we sort bounding-box intersections by the time traveled along the ray, which is not the same as distance when the direction is not unit length.

.8 & 8.9

Applying a Transformation

Position

Direction

Normal

$$\vec{\mathbf{n}}' = ?$$

$$\underset{\mathbf{M}}{\overset{\text{Affine}}{\begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}} = \underset{\mathbf{M}_T}{\overset{\text{Translate}}{\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}} \cdot \underset{\mathbf{M}_L}{\overset{\text{Linear}}{\begin{pmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}}$$

Normal Transformation

Key Idea:
Normals describe a (perpendicularity) relationship to directions, not the directions themselves.

⇒ To transform a normal, we need to transform the relationship.

Normal Transformation

2D Motivating Example:

$$\underbrace{\begin{pmatrix}1 & 0 & 1\\0 & 2 & 1\\0 & 0 & 1\end{pmatrix}}_{\mathbf{M}} = \underbrace{\overset{\text{Translate}}{\begin{pmatrix}1 & 0 & 1\\0 & 1 & 1\\0 & 0 & 1\end{pmatrix}}}_{\mathbf{M}_T} \cdot \underbrace{\overset{\text{Scale}}{\begin{pmatrix}1 & 0 & 0\\0 & 2 & 0\\0 & 0 & 1\end{pmatrix}}}_{\mathbf{M}_L}$$

If $\vec{\mathbf{v}}$ is a direction in 2D, and $\vec{\mathbf{n}}$ is perpendicular to $\vec{\mathbf{v}}$, we want the transformed $\vec{\mathbf{n}}$ to be perpendicular to the transformed $\vec{\mathbf{v}}$:

$$\langle \vec{\mathbf{v}}, \vec{\mathbf{n}} \rangle = 0 \quad \Rightarrow \quad \langle \vec{\mathbf{v}}', \vec{\mathbf{n}}' \rangle = 0$$

$$\Updownarrow$$

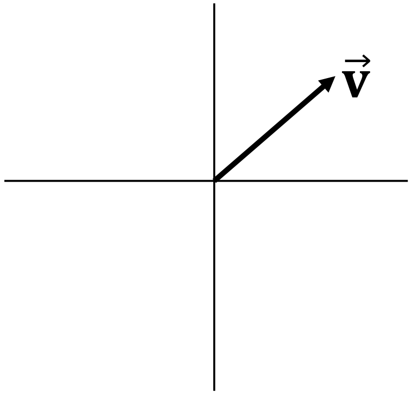
$$\langle \vec{\mathbf{v}}, \vec{\mathbf{n}} \rangle = 0 \quad \Rightarrow \quad \langle \mathbf{M}_L \cdot \vec{\mathbf{v}}, \vec{\mathbf{n}}' \rangle = 0$$

Normal Transformation

2D Motivating Example:

$$\underbrace{\begin{pmatrix}1 & 0 & 1\\0 & 2 & 1\\0 & 0 & 1\end{pmatrix}}_{\mathbf{M}} = \underbrace{\overset{\text{Translate}}{\begin{pmatrix}1 & 0 & 1\\0 & 1 & 1\\0 & 0 & 1\end{pmatrix}}}_{\mathbf{M}_T} \cdot \underbrace{\overset{\text{Scale}}{\begin{pmatrix}1 & 0 & 0\\0 & 2 & 0\\0 & 0 & 1\end{pmatrix}}}_{\mathbf{M}_L}$$

Say $\vec{\mathbf{v}} = (2,2)$

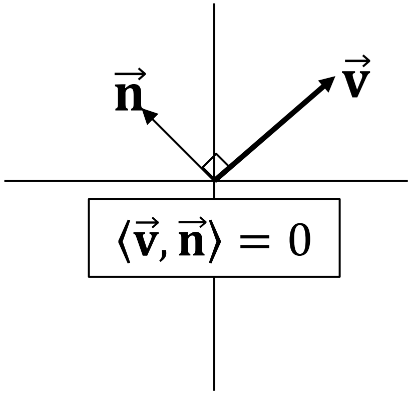


Normal Transformation

<u>2D Motivating Example</u>:

$$\underset{\mathbf{M}}{\begin{pmatrix}1 & 0 & 1\\ 0 & 2 & 1\\ 0 & 0 & 1\end{pmatrix}} = \underset{\mathbf{M}_T}{\overset{\text{Translate}}{\begin{pmatrix}1 & 0 & 1\\ 0 & 1 & 1\\ 0 & 0 & 1\end{pmatrix}}} \cdot \underset{\mathbf{M}_L}{\overset{\text{Scale}}{\begin{pmatrix}1 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 1\end{pmatrix}}}$$

Say $\vec{\mathbf{v}} = (2,2)\ldots$ then $\vec{\mathbf{n}} = (-\sqrt{.5}, \sqrt{.5})$



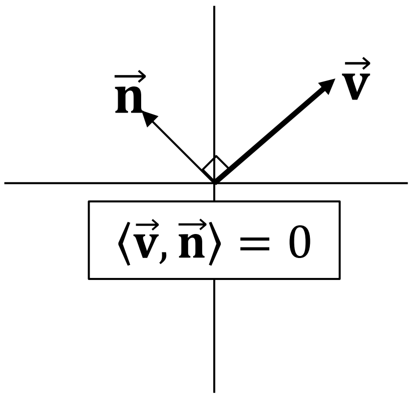
Normal Transformation

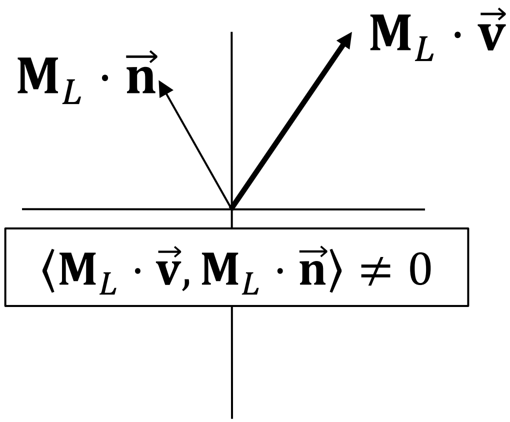
2D Motivating Example:

$$\underbrace{\begin{pmatrix}1 & 0 & 1\\0 & 2 & 1\\0 & 0 & 1\end{pmatrix}}_{\mathbf{M}} = \overbrace{\underbrace{\begin{pmatrix}1 & 0 & 1\\0 & 1 & 1\\0 & 0 & 1\end{pmatrix}}_{\mathbf{M}_T}}^{\text{Translate}} \cdot \overbrace{\underbrace{\begin{pmatrix}1 & 0 & 0\\0 & 2 & 0\\0 & 0 & 1\end{pmatrix}}_{\mathbf{M}_L}}^{\text{Scale}}$$

Say $\vec{\mathbf{v}} = (2,2)\ldots$ then $\vec{\mathbf{n}} = (-\sqrt{.5}, \sqrt{.5})$

Transforming $\mathbf{M}_L \cdot \vec{\mathbf{v}} = (2,4)$ and $\mathbf{M}_L \cdot \vec{\mathbf{n}} = (-\sqrt{.5}, \sqrt{2})$





Normal Transformation

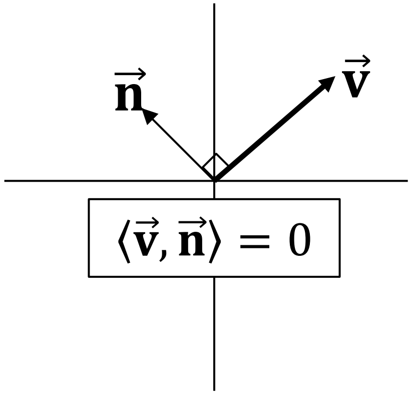
2D Motivating Example:

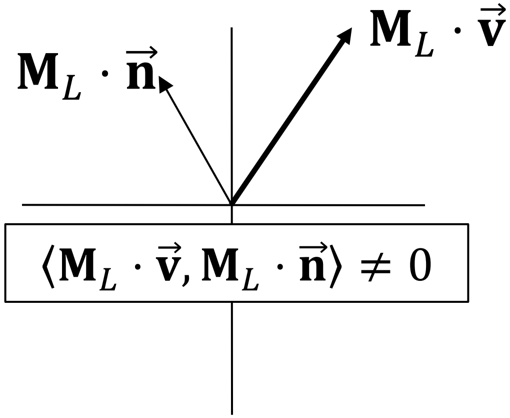
$$
\begin{array}{ccccc}
 & & \text{Translate} & & \text{Scale} \\
\begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix} & = & \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} & \cdot & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
\mathbf{M} & & \mathbf{M}_T & & \mathbf{M}_L
\end{array}
$$

Say $\vec{\mathbf{v}} = (2,2)\ldots$ then $\vec{\mathbf{n}} = (-\sqrt{.5}, \sqrt{.5})$

Transf $\sqrt{.5}, \sqrt{2})$

Transforming $\vec{\mathbf{n}}$ as a direction does not give a vector perpendicular to the transformed $\vec{\mathbf{v}}$!





Recall

<u>Transposes</u>:

The transpose of a matrix $\mathbf{M}$ is the matrix $\mathbf{M}^\top$ whose (i,j) -th coeff. is the (j,i) -th coeff. of $\mathbf{M}$:

$$\mathbf{M} = \begin{pmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{pmatrix} \quad \mathbf{M}^\top = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$$

<u>Recall</u>:

For matrix $\mathbf{M}$, the transpose of the transpose is $\mathbf{M}$:

$$(\mathbf{M}^\top)^\top = \mathbf{M}$$

Recall

Transposes:

The transpose of a matrix $\mathbf{M}$ is the matrix $\mathbf{M}^\top$ whose (i,j) -th coeff. is the (j,i) -th coeff. of $\mathbf{M}$:

$$\mathbf{M} = \begin{pmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{pmatrix} \quad \mathbf{M}^\top = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$$

Recall:

For matrices $\mathbf{M}$ and $\mathbf{N}$, the transpose of the product is the reversed product of the transposes:

$$(\mathbf{M} \cdot \mathbf{N})^\top = \mathbf{N}^\top \cdot \mathbf{M}^\top$$

Recall

Dot-Products:

The dot product of two vectors $\vec{\mathbf{v}} = \left(v_x, v_y, v_z\right)^\top$ and $\vec{\mathbf{w}} = \left(w_x, w_y, w_z\right)^\top$ is obtained by summing the product of the coefficients:

$$\langle \vec{\mathbf{v}}, \vec{\mathbf{w}} \rangle = v_x \cdot w_x + v_y \cdot w_y + v_z \cdot w_z$$

We can also express this as a matrix product:

$$\langle \vec{\mathbf{v}}, \vec{\mathbf{w}} \rangle = \vec{\mathbf{v}}^\top \cdot \vec{\mathbf{w}} = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix} \cdot \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix}$$

$$\langle \vec{\mathbf{v}}, \vec{\mathbf{w}} \rangle = \vec{\mathbf{v}}^\top \cdot \vec{\mathbf{w}}$$

Recall

<u>Orthogonal Matrices</u>:

A matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ is **orthogonal** if it preserves the dot-product:

$$\langle \mathbf{R} \cdot \vec{\mathbf{v}}, \mathbf{R} \cdot \vec{\mathbf{w}} \rangle = \langle \vec{\mathbf{v}}, \vec{\mathbf{w}} \rangle \quad \forall \vec{\mathbf{v}}, \vec{\mathbf{w}} \in \mathbb{R}^3$$

$$\Updownarrow$$

$$\vec{\mathbf{v}}^\top \cdot \mathbf{R}^\top \cdot \mathbf{R} \cdot \vec{\mathbf{w}} = \vec{\mathbf{v}}^\top \cdot \vec{\mathbf{w}} \qquad \forall \vec{\mathbf{v}}, \vec{\mathbf{w}} \in \mathbb{R}^3$$

$$\Updownarrow$$

$$\mathbf{R}^\top \cdot \mathbf{R} = \mathbf{Id}.$$

$$\Updownarrow$$

$$\mathbf{R}^\top = \mathbf{R}^{-1}$$

$\langle \vec{\mathbf{v}}, \vec{\mathbf{w}} \rangle = \vec{\mathbf{v}}^\top \cdot \vec{\mathbf{w}}$

Recall

Transposes and Dot-Products:

If $\mathbf{M}$ is a matrix, and $\vec{\mathbf{v}}$ and $\vec{\mathbf{w}}$ are vectors, then:

$$\begin{aligned}
\langle \vec{\mathbf{v}}, \mathbf{M} \cdot \vec{\mathbf{w}} \rangle &= \vec{\mathbf{v}}^\top \cdot (\mathbf{M} \cdot \vec{\mathbf{w}}) \\
&= (\vec{\mathbf{v}}^\top \cdot \mathbf{M}) \cdot \vec{\mathbf{w}} \\
&= \left((\vec{\mathbf{v}}^\top \cdot \mathbf{M})^\top \right)^\top \cdot \vec{\mathbf{w}} \\
&= (\mathbf{M}^\top \cdot \vec{\mathbf{v}})^\top \cdot \vec{\mathbf{w}} \\
&= \langle \mathbf{M}^\top \vec{\mathbf{v}}, \vec{\mathbf{w}} \rangle
\end{aligned}$$

$\langle \vec{\mathbf{v}}, \mathbf{M} \cdot \vec{\mathbf{w}} \rangle = \langle \mathbf{M}^\top \cdot \vec{\mathbf{v}}, \vec{\mathbf{w}} \rangle$

Normal Transformation

A normal $\vec{\mathbf{n}}$ is defined by having a fixed (zero) dot-product with some direction vector(s) $\vec{\mathbf{v}}$.

We need the dot-product of the transformed normal $\vec{\mathbf{n}}'$ with the transformed direction(s) to not change:

$$\begin{aligned} \langle \vec{\mathbf{n}}, \vec{\mathbf{v}} \rangle &= \langle \vec{\mathbf{n}}', \mathbf{M}_L \cdot \vec{\mathbf{v}} \rangle \\ &= \langle \mathbf{M}_L^\top \cdot \vec{\mathbf{n}}', \vec{\mathbf{v}} \rangle \end{aligned}$$

$$\Updownarrow$$

$$\vec{\mathbf{n}} = \mathbf{M}_L^\top \cdot \vec{\mathbf{n}}'$$

$$\Updownarrow$$

$$\vec{\mathbf{n}}' = (\mathbf{M}_L^\top)^{-1} \cdot \vec{\mathbf{n}}$$

If the linear transformation $\mathbf{M}_L$ is orthogonal (i.e. no scaling) then $(\mathbf{M}_L^\top)^{-1} = (\mathbf{M}_L^{-1})^{-1} = \mathbf{M}_L$.

Applying a Transformation

Position

$$\mathbf{p}' = \mathbf{M} \cdot \mathbf{p}$$

Direction

$$\vec{\mathbf{v}}' = \mathbf{M}_L \cdot \vec{\mathbf{v}}$$

Normal

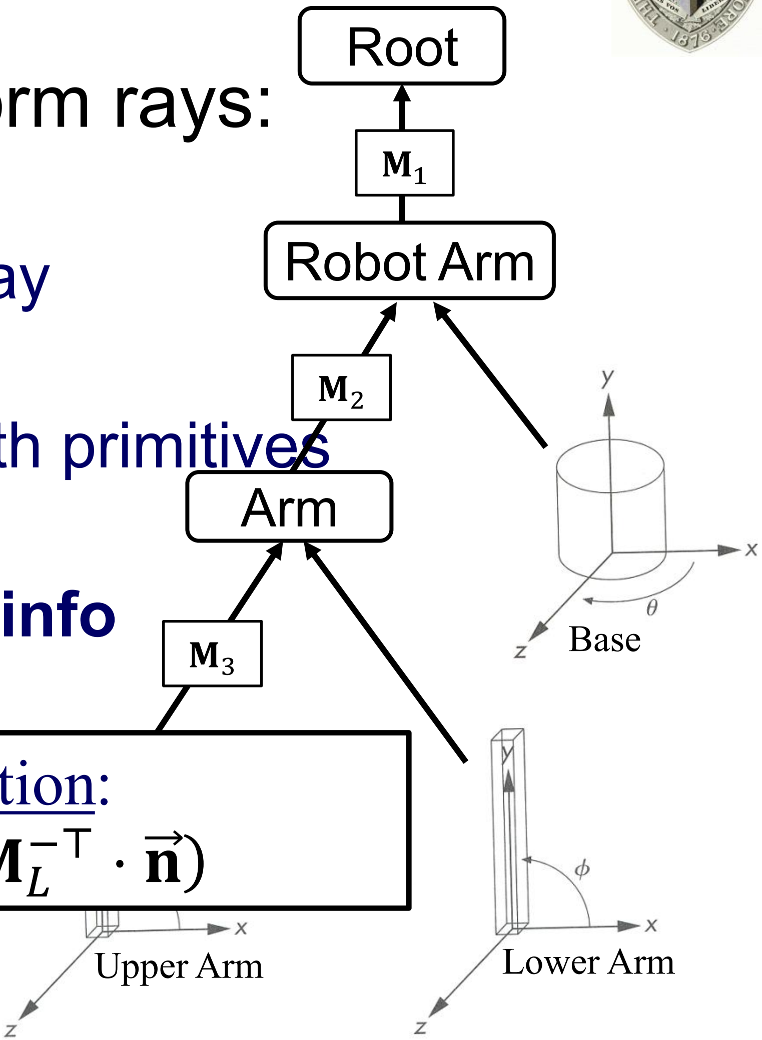
$$\vec{\mathbf{n}}' = \mathbf{M}_L^{-\top} \cdot \vec{\mathbf{n}}$$

$$\underset{\mathbf{M}}{\overset{\text{Affine}}{\begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}} = \underset{\mathbf{M}_T}{\overset{\text{Translate}}{\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}} \cdot \underset{\mathbf{M}_L}{\overset{\text{Linear}}{\begin{pmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}}$$

Ray Casting With Hierarchies

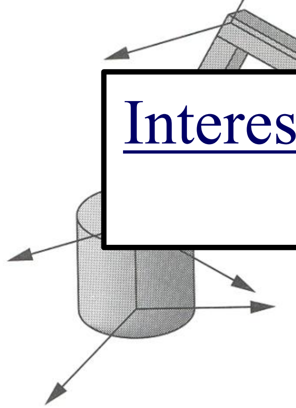
Iterating from the root, transform rays:

- world → model:
  - Apply inverse transform to ray
- model coordinates:
  - Intersect transformed ray with primitives
- **model → world:**
  - **Apply transform to the hit info**



Interesection Transformation:

$$(\mathbf{p},\vec{\mathbf{n}}) \rightarrow (\mathbf{M}\cdot\mathbf{p},\mathbf{M}_L^{-\top}\cdot\vec{\mathbf{n}})$$



Robot Arm

Angel Figures 8.8 & 8.9

Overview

Transformation Hierarchies
- Scene graphs
- Ray casting

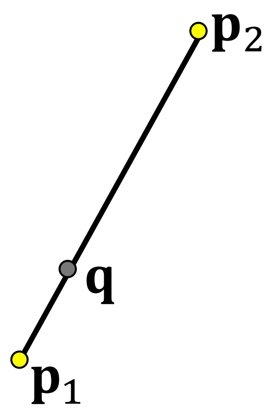
Barycentric Coordinates

Barycentric Coordinates

Recall:

Given vertices $\mathbf{p}_1$ and $\mathbf{p}_2$, a point $\mathbf{q}$ on the line segment between the vertices is the (non-negatively) weighted average of $\mathbf{p}_1$ and $\mathbf{p}_2$:

$$LS = \{\alpha\mathbf{p}_1 + \beta\mathbf{p}_2 \mid \alpha + \beta = 1, \alpha, \beta \geq 0\}$$

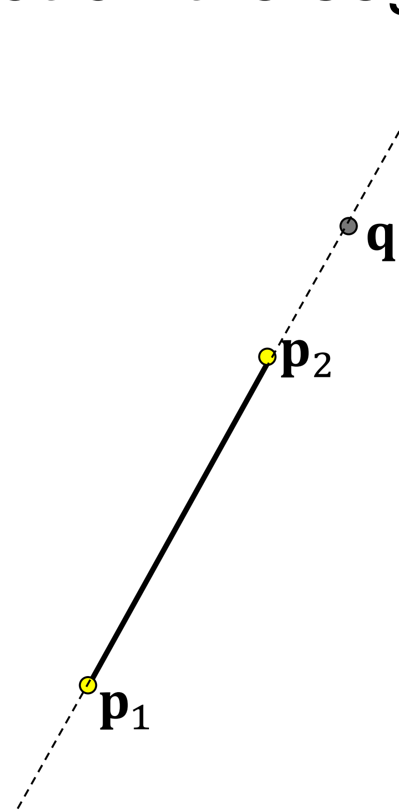


$$\boxed{\begin{gathered}\alpha + \beta = 1 \\ \alpha, \beta \geq 0\end{gathered}}$$

Barycentric Coordinates

<u>Recall</u>:

If the weights sum to one but are not positive, $\mathbf{q}$ is on the line but not on the segment (extrapolation)

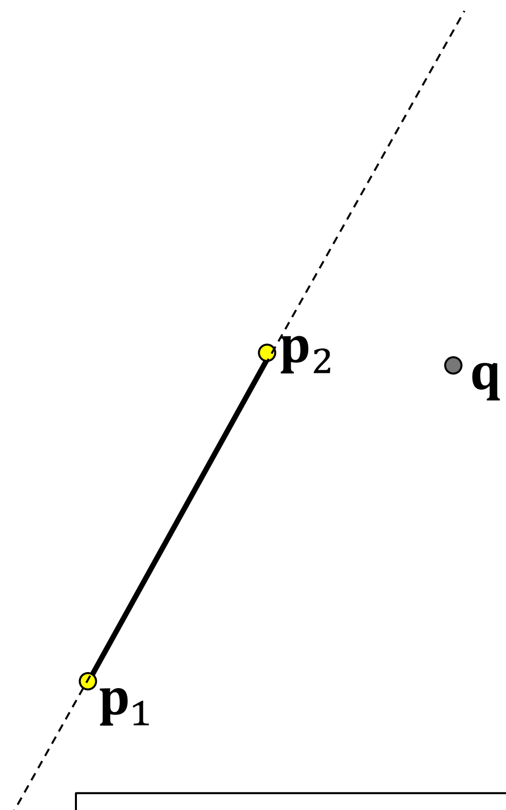


$$\boxed{\begin{array}{c}\alpha+\beta=1\\ \alpha<0\end{array}}$$

Barycentric Coordinates

<u>Recall</u>:

If the weights do not sum to one, $\mathbf{q}$ will not (in general*) be on the line



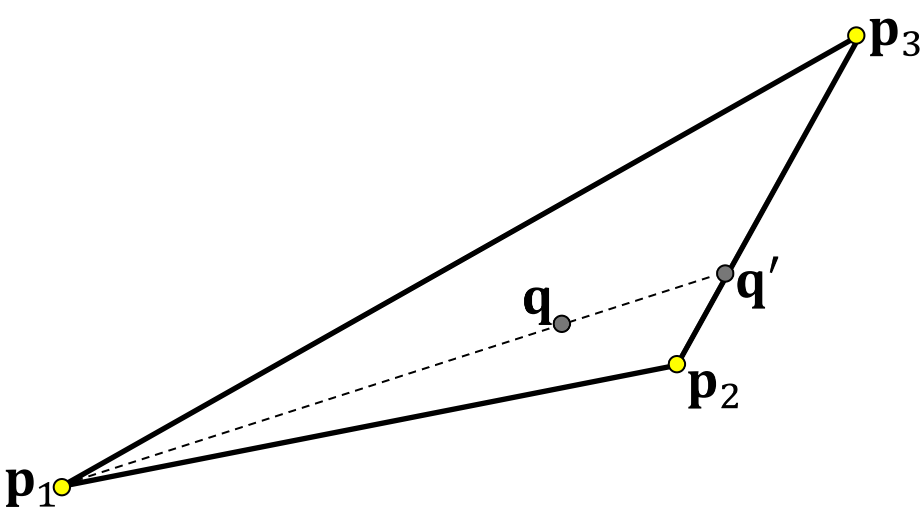
$\boxed{\alpha + \beta \neq 1}$

*unless the line through $\mathbf{p}_1$ and $\mathbf{p}_2$ passes through the origin

Barycentric Coordinates

A triangle is defined by three (non-collinear) vertices

Note:
A point $\mathbf{q}$ is inside the triangle if and only if $\mathbf{q}$ is on the line segment between (w.l.o.g*) $\mathbf{p}_1$ and a point $\mathbf{q}'$ on the opposite edge $\overline{\mathbf{p}_2\mathbf{p}_3}$.



*w.l.o.g $\equiv$ without loss of generality

Inside Triangle ⇒ Positive Weights

Claim:

Any point $\mathbf{q}$ inside the triangle, can be expressed as:

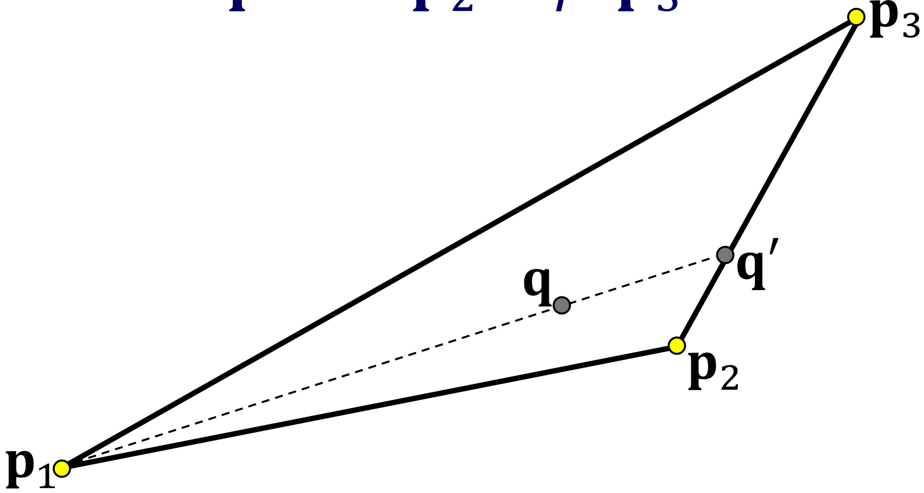
$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

Proof:

The point $\mathbf{q}$ is between $\mathbf{p}_1$ and $\mathbf{q}'$ on edge $\overline{\mathbf{p}_2\mathbf{p}_3}$:

$\Rightarrow \mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{q}'$ with $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$

$\Rightarrow \mathbf{q}' = \alpha'\mathbf{p}_2 + \beta'\mathbf{p}_3$ with $\alpha' + \beta' = 1$ and $\alpha', \beta' \geq 0$



Inside Triangle $\Rightarrow$ Positive Weights

Claim:

Any point $\mathbf{q}$ inside the triangle, can be expressed as:

$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

Proof:

The point $\mathbf{q}$ is between $\mathbf{p}_1$ and $\mathbf{q}'$ on edge $\overline{\mathbf{p}_2\mathbf{p}_3}$:

$\Rightarrow \mathbf{q} = \alpha\mathbf{p}_1 + \beta\boxed{\mathbf{q}'}$ with $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$

$\Rightarrow \boxed{\mathbf{q}' = \alpha'\mathbf{p}_2 + \beta'\mathbf{p}_3}$ with $\alpha' + \beta' = 1$ and $\alpha', \beta' \geq 0$

This gives $\mathbf{q}$ as the linear sum:

- $\mathbf{q} = \alpha\mathbf{p}_1 + (\beta\alpha')\mathbf{p}_2 + (\beta\beta')\mathbf{p}_3$

Inside Triangle ⇒ Positive Weights

Claim:

Any point $\mathbf{q}$ inside the triangle, can be expressed as:

$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

Proof:

The point $\mathbf{q}$ is between $\mathbf{p}_1$ and $\mathbf{q}'$ on edge $\overline{\mathbf{p}_2\mathbf{p}_3}$:

$\Rightarrow \mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{q}'$ with $\alpha + \beta = 1$ and $\boxed{\alpha, \beta \geq 0}$

$\Rightarrow \mathbf{q}' = \alpha'\mathbf{p}_2 + \beta'\mathbf{p}_3$ with $\alpha' + \beta' = 1$ and $\boxed{\alpha', \beta' \geq 0}$

This gives $\mathbf{q}$ as the linear sum:

- $\mathbf{q} = \alpha\mathbf{p}_1 + (\beta\alpha')\mathbf{p}_2 + (\beta\beta')\mathbf{p}_3$
  - » Positive weights: α , $\beta\alpha'$, $\beta\beta' \geq 0$

Inside Triangle $\Rightarrow$ Positive Weights

Claim:

Any point $\mathbf{q}$ inside the triangle, can be expressed as:

$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

Proof:

The point $\mathbf{q}$ is between $\mathbf{p}_1$ and $\mathbf{q}'$ on edge $\overline{\mathbf{p}_2\mathbf{p}_3}$:

$\Rightarrow \mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{q}'$ with $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$

$\Rightarrow \mathbf{q}' = \alpha'\mathbf{p}_2 + \beta'\mathbf{p}_3$ with $\alpha' + \beta' = 1$ and $\alpha', \beta' \geq 0$

This gives $\mathbf{q}$ as the linear sum:

- $\mathbf{q} = \alpha\mathbf{p}_1 + (\beta\alpha')\mathbf{p}_2 + (\beta\beta')\mathbf{p}_3$
  - » Positive weights: α, $\beta\alpha'$, $\beta\beta' \geq 0$
  - » Summing to one: $\alpha + \beta\alpha' + \beta\beta'$

Inside Triangle ⇒ Positive Weights

Claim:

Any point $\mathbf{q}$ inside the triangle, can be expressed as:

$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

Proof:

The point $\mathbf{q}$ is between $\mathbf{p}_1$ and $\mathbf{q}'$ on edge $\overline{\mathbf{p}_2\mathbf{p}_3}$:

$\Rightarrow \mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{q}'$ with $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$

$\Rightarrow \mathbf{q}' = \alpha'\mathbf{p}_2 + \beta'\mathbf{p}_3$ with $\alpha' + \beta' = 1$ and $\alpha', \beta' \geq 0$

This gives $\mathbf{q}$ as the linear sum:

- $\mathbf{q} = \alpha\mathbf{p}_1 + (\beta\alpha')\mathbf{p}_2 + (\beta\beta')\mathbf{p}_3$
  - » Positive weights: α , $\beta\alpha'$, $\beta\beta' \geq 0$
  - » Summing to one:
    $\alpha + \beta\alpha' + \beta\beta' = \alpha + \beta(\alpha' + \beta')$

Inside Triangle ⇒ Positive Weights

Claim:

Any point $\mathbf{q}$ inside the triangle, can be expressed as:

$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

Proof:

The point $\mathbf{q}$ is between $\mathbf{p}_1$ and $\mathbf{q}'$ on edge $\overline{\mathbf{p}_2\mathbf{p}_3}$:

$\Rightarrow \mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{q}'$ with $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$

$\Rightarrow \mathbf{q}' = \alpha'\mathbf{p}_2 + \beta'\mathbf{p}_3$ with $\boxed{\alpha' + \beta' = 1}$ and $\alpha', \beta' \geq 0$

This gives $\mathbf{q}$ as the linear sum:

- $\mathbf{q} = \alpha\mathbf{p}_1 + (\beta\alpha')\mathbf{p}_2 + (\beta\beta')\mathbf{p}_3$
  - » Positive weights: α , $\beta\alpha'$, $\beta\beta' \geq 0$
  - » Summing to one:
    $\alpha + \beta\alpha' + \beta\beta' = \alpha + \beta(\alpha' + \beta') = \alpha + \beta$

Inside Triangle $\Rightarrow$ Positive Weights

Claim:

Any point $\mathbf{q}$ inside the triangle, can be expressed as:

$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

Proof:

The point $\mathbf{q}$ is between $\mathbf{p}_1$ and $\mathbf{q}'$ on edge $\overline{\mathbf{p}_2\mathbf{p}_3}$:

$\Rightarrow \mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{q}'$ with $\boxed{\alpha + \beta = 1}$ and $\alpha, \beta \geq 0$

$\Rightarrow \mathbf{q}' = \alpha'\mathbf{p}_2 + \beta'\mathbf{p}_3$ with $\alpha' + \beta' = 1$ and $\alpha', \beta' \geq 0$

This gives $\mathbf{q}$ as the linear sum:

- $\mathbf{q} = \alpha\mathbf{p}_1 + (\beta\alpha')\mathbf{p}_2 + (\beta\beta')\mathbf{p}_3$
  - » Positive weights: α , $\beta\alpha'$, $\beta\beta' \geq 0$
  - » Summing to one:
    $\alpha + \beta\alpha' + \beta\beta' = \alpha + \beta(\alpha' + \beta') = \alpha + \beta = 1$

Inside Triangle $\Leftarrow$ Positive Weights

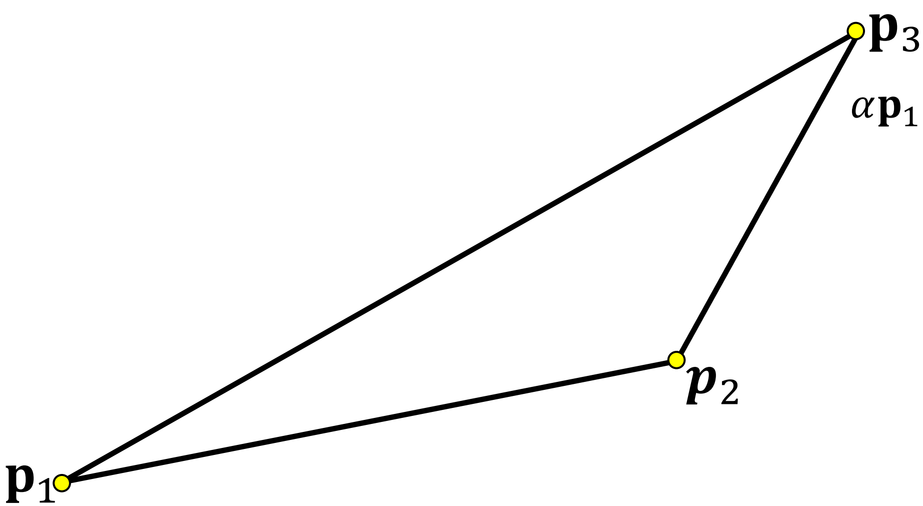
Claim:

If a point $\mathbf{q}$ can be expressed as:

$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 \quad \text{with } \boxed{\alpha + \beta + \gamma = 1} \text{ and } \alpha, \beta, \gamma \geq 0$$

then $\mathbf{q}$ is in the triangle.

Proof:



$$\begin{aligned}\alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 &= \alpha\mathbf{p}_1 + (1-\alpha)\frac{\beta\mathbf{p}_2 + \gamma\mathbf{p}_3}{1-\alpha} \\ &= \alpha\mathbf{p}_1 + (1-\alpha)\frac{\beta\mathbf{p}_2 + \gamma\mathbf{p}_3}{\beta+\gamma}\end{aligned}$$

Inside Triangle $\Leftarrow$ Positive Weights

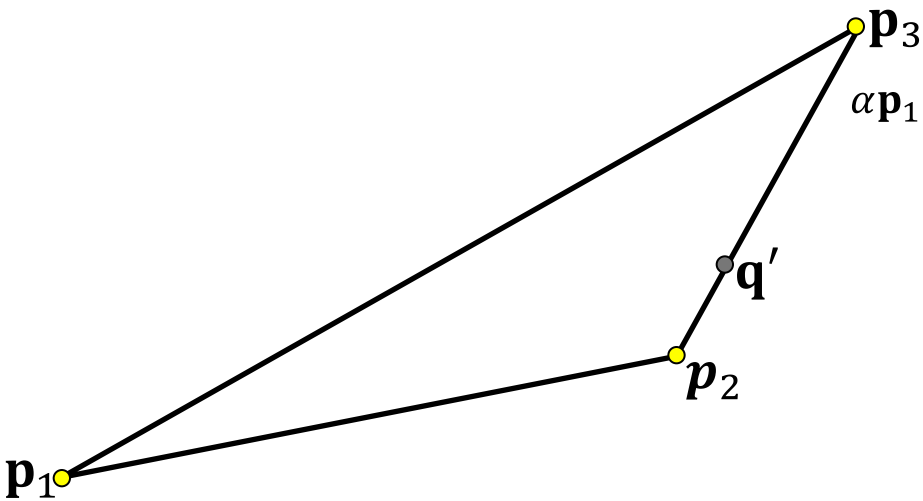
<u>Claim</u>:

If a point $\mathbf{q}$ can be expressed as:

$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

then $\mathbf{q}$ is in the triangle.

<u>Proof</u>:



$$\alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 = \alpha\mathbf{p}_1 + (1-\alpha)\frac{\beta\mathbf{p}_2 + \gamma\mathbf{p}_3}{1-\alpha}$$

$$= \alpha\mathbf{p}_1 + (1-\alpha)\boxed{\frac{\beta\mathbf{p}_2 + \gamma\mathbf{p}_3}{\beta+\gamma}}$$

A point $\mathbf{q}'$ on the segment between $\mathbf{p}_2$ and $\mathbf{p}_3$

Inside Triangle $\Leftarrow$ Positive Weights

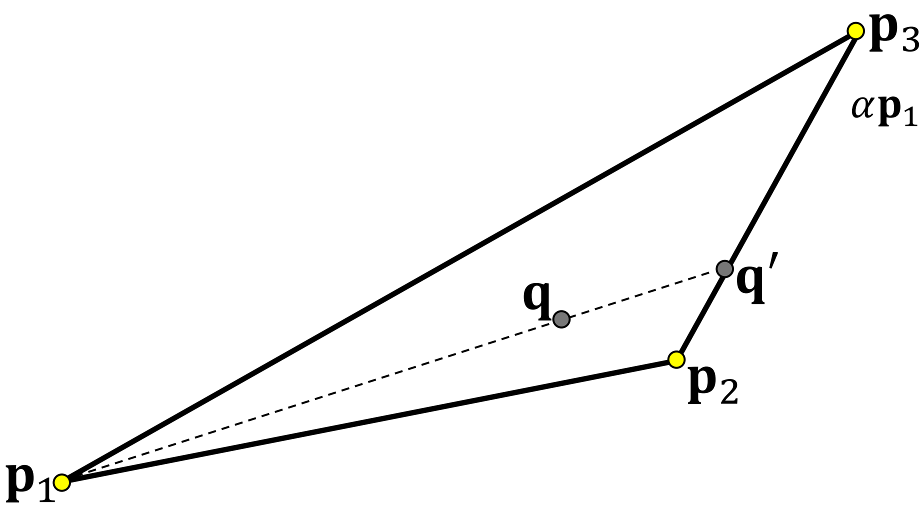
Claim:

If a point $\mathbf{q}$ can be expressed as:

$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

then $\mathbf{q}$ is in the triangle.

Proof:



$$\alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 = \alpha\mathbf{p}_1 + (1-\alpha)\frac{\beta\mathbf{p}_2 + \gamma\mathbf{p}_3}{1-\alpha}$$

$$= \boxed{\alpha\mathbf{p}_1 + (1-\alpha)\frac{\beta\mathbf{p}_2 + \gamma\mathbf{p}_3}{\beta+\gamma}}$$

A point $\mathbf{q}$ on the segment between $\mathbf{p}_1$ and $\mathbf{q}'$

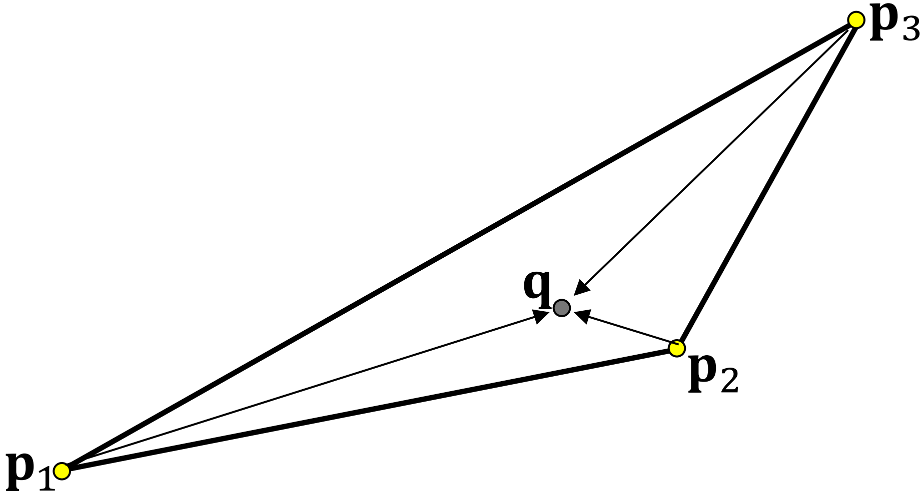
Barycentric Coordinates

The barycentric coordinates of a point $\mathbf{q}$:

$$\mathbf{q} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3$$

let us express $\mathbf{q}$ as the weighted average of the triangle vertices.

The weights α, β, and γ tell us, relatively, how close the point $\mathbf{q}$ is to each of $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$.



Barycentric Coordinates

Barycentric coordinates are needed in:

- Ray-tracing, to test for intersection
- Rendering, to interpolate triangle information

Barycentric Coordinates

Barycentric coordinates are needed in:

- Ray-tracing, to test for intersection
- Rendering, to interpolate triangle information

```
Float TriangleIntersect( Ray r , Triangle tgl )
{
    Plane p = PlaneContaining( tgl );
    float t = IntersectionDistance( r, p );
    if( t<0 ) return ∞;
    else
    {
        ( α , β , γ ) = BarycentricCoordinates( r(t) , tgl );
        if( α<0  or β<0 or γ<0 ) return ∞;
        else return t;
    }
}
```

Barycentric Coordinates

Barycentric coordinates are needed in:

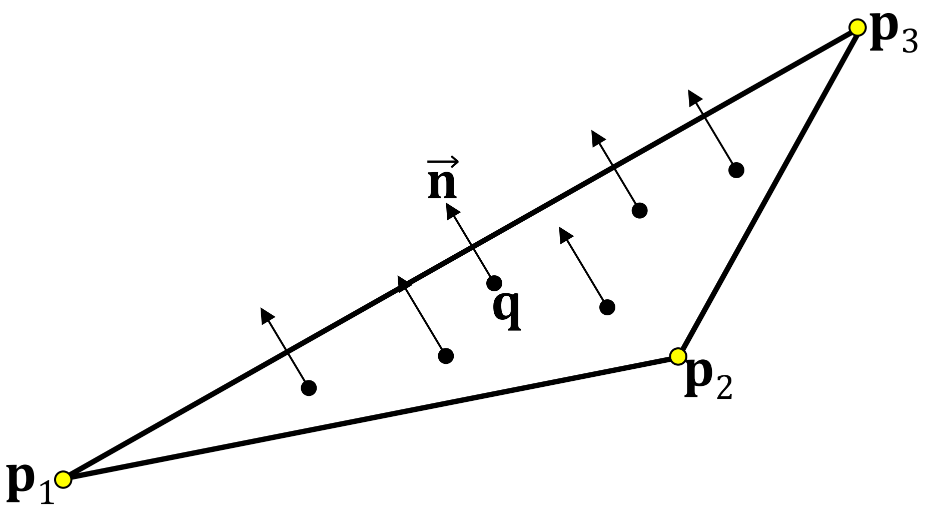
- Ray-tracing, to test for intersection
- Rendering, to interpolate triangle information

  Information (e.g. colors, normals, etc.) is typically defined at vertices rather than triangles

Barycentric Coordinates

We can associate the same **geometric** normal to every point on the face of a triangle by computing:

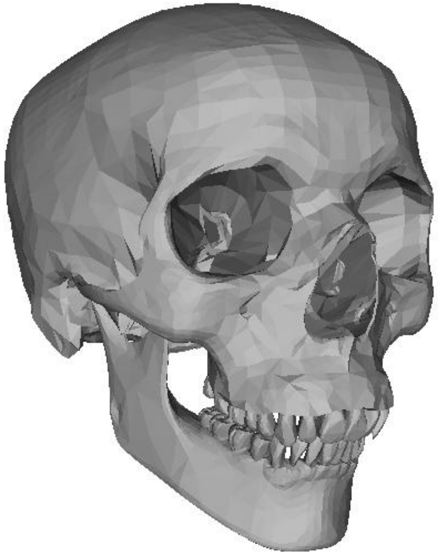
$$\vec{\mathbf{n}} = \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{\|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)\|}$$



Barycentric Coordinates

We can associate the same **geometric** normal to every point on the face of a triangle by computing:

$$\vec{\mathbf{n}} = \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{\|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)\|}$$



Triangle Normals

Results in flat shading across the faces

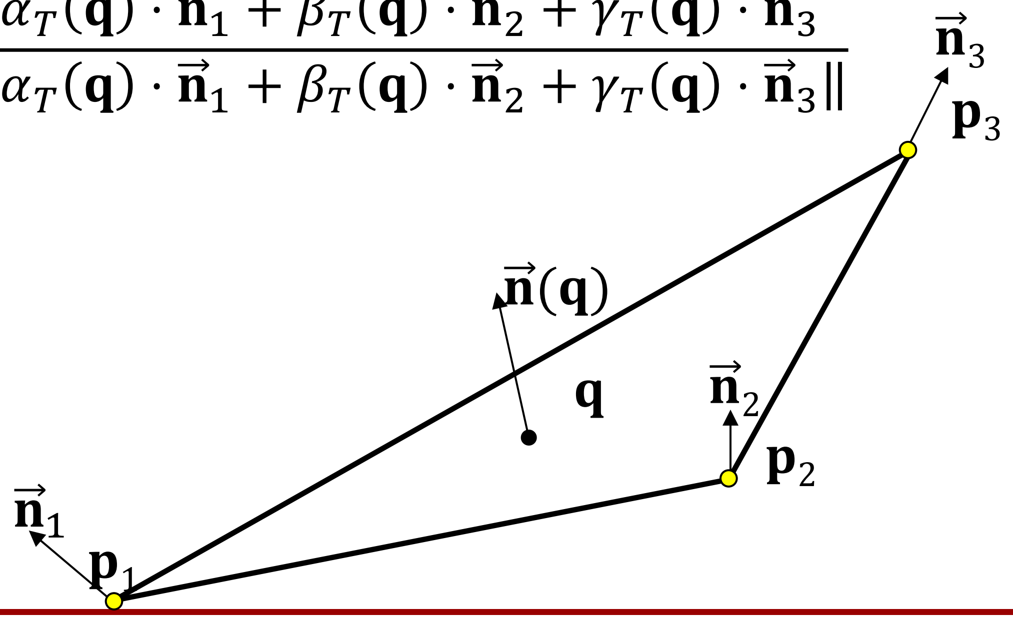
Barycentric Coordinates

Instead, we can associate different **rendering** normals to the vertices:

$$T = \big((\mathbf{p}_1, \vec{\mathbf{n}}_1), (\mathbf{p}_2, \vec{\mathbf{n}}_2), (\mathbf{p}_3, \vec{\mathbf{n}}_3)\big)$$

⇒ The normal at a point $\mathbf{q}$ in the triangle is the interpolation of the normals at the vertices:

$$\vec{\mathbf{n}}(\mathbf{q}) = \frac{\alpha_T(\mathbf{q}) \cdot \vec{\mathbf{n}}_1 + \beta_T(\mathbf{q}) \cdot \vec{\mathbf{n}}_2 + \gamma_T(\mathbf{q}) \cdot \vec{\mathbf{n}}_3}{\|\alpha_T(\mathbf{q}) \cdot \vec{\mathbf{n}}_1 + \beta_T(\mathbf{q}) \cdot \vec{\mathbf{n}}_2 + \gamma_T(\mathbf{q}) \cdot \vec{\mathbf{n}}_3\|}$$

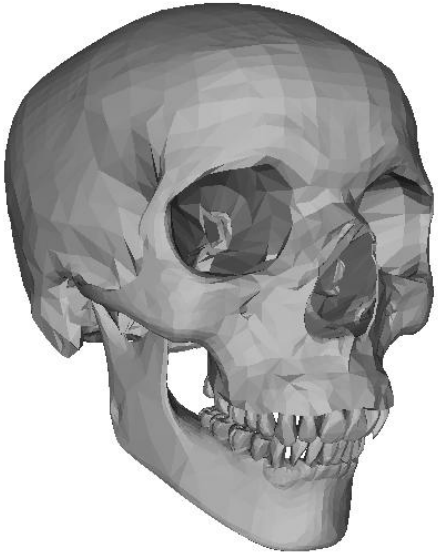


Barycentric Coordinates

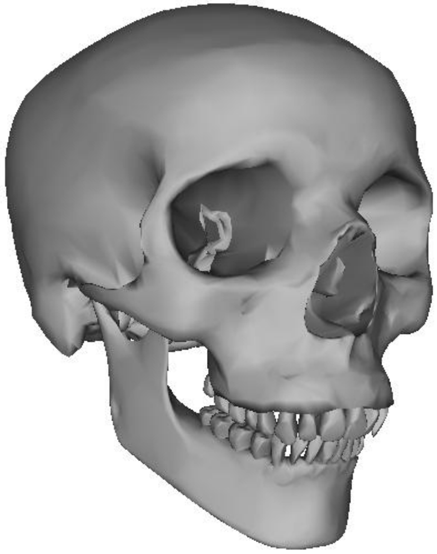
Instead, we can associate different **rendering** normals to the vertices:

$$T = \left((\mathbf{p}_1, \vec{\mathbf{n}}_1), (\mathbf{p}_2, \vec{\mathbf{n}}_2), (\mathbf{p}_3, \vec{\mathbf{n}}_3)\right)$$

⇒ The normal we use to render at $\mathbf{q}$ is the interpolation of the normals from the vertices:



Triangle Normals



Interpolated Point Normals

Barycentric Coordinates

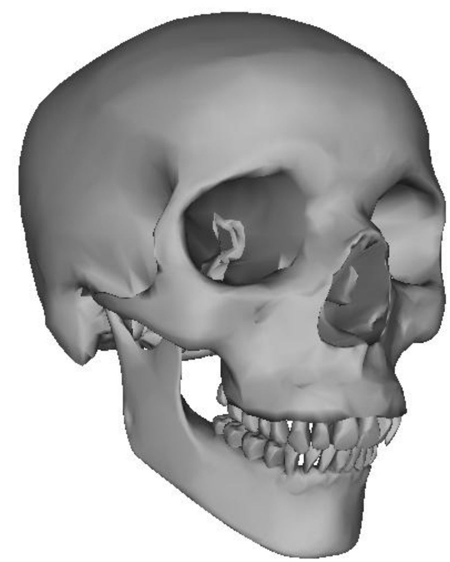
Instead, we can associate different **rendering** nor

> Note:
>
> Don’t confuse the two normals
>
> - Geometric normal (for intersections)
> - Rendering normal (for rendering)

⇒ T

i

Triangle Normals



Interpolated Point Normals