Scene Graphs and Barycentric Coordinates

Michael Kazhdan

(601.457/657)

HB Ch. 5
FvDFH Ch. 5
Last Time

• 2D Transformations
  ◦ Basic 2D transformations
  ◦ Matrix representation
  ◦ Matrix composition

• 3D Transformations
  ◦ Basic 3D transformations
  ◦ Same as 2D
Homogeneous Coordinates

- Add a 4\(^{th}\) coordinate to every 3D point
  - \((x, y, z, w)\) represents a point at location \(
  \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)\)
  - \((x, y, z, 0)\) represents the (unsigned) direction \(
  \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}}\right)\)

- Represent transformations by \(4 \times 4\) 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 \(3 \times 3\) 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
Transformation Example 1

- 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
Transformation Example 1

Building

Floor 1    Floor 2    Floor 3    Floor 4    Floor 5

Office 1    Office N

Bookshelf 1    Desk 1    Desk 2    Chair 1    Chair K
Transformation Example 1

Building

Floor 1

Floor 2

Floor 3

Floor 4

Floor 5

Office 1

Office N

Bookshelf 1

Desk 1

Desk 2

Chair 1

Chair K

Instances

Definitions

Bookshelf

Desk

Chair
The transformation applied to a node of the scene graph is the composition of transformations to the root.
Transformation Example 2

- Well-suited for articulated characters

Rose et al. `96
Scene Graphs

• **Instancing:**
  Allow us to have multiple instances of a single model – reducing model storage size and making it easier to make consistent changes

• **Local Modeling:**
  Allow us to model objects in local coordinates and then place them into a global frame – particularly important for animation

• **Hierarchical Representation:**
  Accelerate ray-tracing by providing a hierarchy that can be used for bounding volume testing
Ray Casting with Hierarchies

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

\[
M = \begin{pmatrix}
1 & 0 \\
0 & 1/2 \\
\end{pmatrix}
\]
Ray Casting with Hierarchies

Transform rays/intersections, not primitives

- For each node ...
  - Global-to-Local: Transform ray by matrix inverse
  - Local: Intersect transformed ray with primitives
  - Local-to-Global: Transform hit info by matrix
Ray Casting with Hierarchies

• Given a ray $r$ (in global coordinates)
  ○ **Base**: “Apply” the inverse of $M_1$ to the ray and test for intersection
Ray Casting with Hierarchies

• Given a ray $r$ (in global coordinates)
  ○ **Base**: “Apply” the inverse of $M_1$ to the ray and test for intersection
  ○ **Lower Arm**: “Apply” the inverse of $M_1 \circ M_2$ to the ray and test for intersection
Ray Casting with Hierarchies

- Given a ray (in global coordinates)
  - **Base**: “Apply” the inverse of $M_1$ to the ray and test for intersection
  - **Lower Arm**: “Apply” the inverse of $M_1 \circ M_2$ to the ray and test for intersection
  - **Upper Arm**: “Apply” the inverse of $M_1 \circ M_2 \circ M_3$ to the ray and test for intersection
Ray Casting with Hierarchies

• If there is an intersection:
  - **Base**: “Apply” $M_1$ to the intersection information
  - **Lower Arm**: “Apply” $M_1 \circ M_2$ to the intersection information
  - **Upper Arm**: “Apply” $M_1 \circ M_2 \circ M_3$ to the intersection information

Angel Figures 8.8 & 8.9
Applying a Transformation

- Position
- Direction
- Normal

\[
\begin{align*}
&M = \begin{pmatrix}
a & b & c & t_x \\
d & e & f & t_y \\
g & h & i & t_z \\
0 & 0 & 0 & 1
\end{pmatrix}, & M_T = \begin{pmatrix}
1 & 0 & 0 & t_x \\
0 & 1 & 0 & t_y \\
0 & 0 & 1 & t_z \\
0 & 0 & 0 & 1
\end{pmatrix}, & M_L = \begin{pmatrix}
a & b & c & 0 \\
d & e & f & 0 \\
g & h & i & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\end{align*}
\]
Applying a Transformation

- Position
  - Apply the full affine transformation:
    \[ p' = M(p) = (M_T \cdot M_L)(p) \]

- Direction

- Normal

\[
\begin{pmatrix}
  a & b & c & t_x \\
  d & e & f & t_y \\
  g & h & i & t_z \\
  0 & 0 & 0 & 1 \\
\end{pmatrix} = \begin{pmatrix}
  1 & 0 & 0 & t_x \\
  0 & 1 & 0 & t_y \\
  0 & 0 & 1 & t_z \\
  0 & 0 & 0 & 1 \\
\end{pmatrix} \cdot \begin{pmatrix}
  a & b & c & 0 \\
  d & e & f & 0 \\
  g & h & i & 0 \\
  0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

\[
M = \begin{pmatrix}
  a & b & c & t_x \\
  d & e & f & t_y \\
  g & h & i & t_z \\
  0 & 0 & 0 & 1 \\
\end{pmatrix}, \quad M_T = \begin{pmatrix}
  1 & 0 & 0 & t_x \\
  0 & 1 & 0 & t_y \\
  0 & 0 & 1 & t_z \\
  0 & 0 & 0 & 1 \\
\end{pmatrix}, \quad M_L = \begin{pmatrix}
  a & b & c & 0 \\
  d & e & f & 0 \\
  g & h & i & 0 \\
  0 & 0 & 0 & 1 \\
\end{pmatrix}
\]
# Applying a Transformation

- **Position**
- **Direction**
  - Apply the linear component of the transformation:
    \[ \hat{\mathbf{v}}' = M_L(\hat{\mathbf{v}}) \]
- **Normal**

<table>
<thead>
<tr>
<th>Affine</th>
<th>Translate</th>
<th>Linear</th>
</tr>
</thead>
</table>
| \[
\begin{pmatrix}
    a & b & c & t_x \\
    d & e & f & t_y \\
    g & h & i & t_z \\
    0 & 0 & 0 & 1
\end{pmatrix}
\] | \[
\begin{pmatrix}
    1 & 0 & 0 & t_x \\
    0 & 1 & 0 & t_y \\
    0 & 0 & 1 & t_z \\
    0 & 0 & 0 & 1
\end{pmatrix}
\] | \[
\begin{pmatrix}
    a & b & c & 0 \\
    d & e & f & 0 \\
    g & h & i & 0 \\
    0 & 0 & 0 & 1
\end{pmatrix}
\] |
Applying a Transformation

- Position
- Direction
  - Apply the linear component of the transformation:
    \[
    \vec{v}' = M_L(\vec{v})
    \]

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

\[
\vec{v}' = M_L(p) - M_L(q)
= (M_L(p) + \vec{t}) - (M_L(q) + \vec{t}')
= M_L(p) - M_L(q)
= M_L(p - q)
= M_L(\vec{v})
\]

\[
q \quad \vec{v} \quad p
\]

\[
M(q) \quad \vec{v}' \quad M(p)
\]
Ray Casting With Hierarchies

Transform rays/intersections, not primitives

- For each node ...
  - Global-to-Local: Transform ray by matrix inverse
  - Local-to-Global: Intersect transformed ray with primitives
  - Local: Transform hit info by matrix

Ray Transformation:

\[(p, \vec{v}) \rightarrow \left(M^{-1}(p), M_L^{-1}(\vec{v})\right)\]

Robot Arm

Robot Arm

Robot Arm

Root

\[M_1\]

\[M_2\]

Base

Upper Arm

Lower Arm

Angel Figures 8.8 & 8.9
Ray Casting With Hierarchies

Transform rays/intersections, not primitives

- For each node ... 
  - Global-to-Local: Transform ray by matrix inverse
  - Local-to-Global: Intersect transformed ray with primitives
  - Local-to-Global: Transform hit info by matrix

Ray Transformation:

\[(p, \vec{v}) \rightarrow \left( M^{-1}(p), M_L^{-1}(\vec{v}) \right) \]

Note:
Even if the original ray direction, \( \vec{v} \), was unit-length, the transformed ray direction may not be.

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.
Applying a Transformation

- Position
- Direction
- Normal

\[ \hat{n}' = ? \]

\[
\begin{pmatrix}
a & b & c & t_x \\
d & e & f & t_y \\
g & h & i & t_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & t_x \\
0 & 1 & 0 & t_y \\
0 & 0 & 1 & t_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
a & b & c & 0 \\
d & e & f & 0 \\
g & h & i & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
Normal Transformation

**Key Idea:**
A normal describes a (perpendicularity) relationship to a direction, not the direction itself.

⇒ To transform a normal, we must transform the relationship.
Normal Transformation

2D Example:

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

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

$$\updownarrow$$

$$\langle \vec{v}, \vec{n} \rangle = 0 \Rightarrow \langle M_L(\vec{v}), \vec{n}' \rangle = 0$$
Normal Transformation

2D Example:

\[
\begin{pmatrix}
1 & 0 & 1 \\
0 & 2 & 1 \\
0 & 0 & 1
\end{pmatrix} = \begin{pmatrix}
1 & 0 & 1 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{pmatrix} \cdot \begin{pmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

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

2D Example:

\[
\begin{pmatrix}
1 & 0 & 1 \\
0 & 2 & 1 \\
0 & 0 & 1
\end{pmatrix}
= \begin{pmatrix}
1 & 0 & 1 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{pmatrix}
\cdot \begin{pmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

\( M \cdot M_T \cdot M_L \)

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

\[ \langle \vec{v}, \vec{n} \rangle = 0 \]
Normal Transformation

2D Example:

\[
\begin{pmatrix}
1 & 0 & 1 \\
0 & 2 & 1 \\
0 & 0 & 1
\end{pmatrix} = \begin{pmatrix}
1 & 0 & 1 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{pmatrix} \cdot \begin{pmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

\(M\) \(M_T\) \(M_L\)

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

Transforming \(M_L(\vec{v}) = (2,4)\) and \(M_L(\vec{n}) = (-\sqrt{5}, \sqrt{2})\)

\[\langle \vec{v}, \vec{n} \rangle = 0\]

\[\langle M_L(\vec{v}), M_L(\vec{n}) \rangle \neq 0\]
Normal Transformation

2D Example:

Say \( \vec{v} = (2, 2) \) then \( \vec{n} = (-1, \sqrt{5}, 1, \sqrt{5}) \).

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

\[
\begin{pmatrix}
1 & 0 & 1 \\
0 & 2 & 1 \\
0 & 0 & 1
\end{pmatrix} = \begin{pmatrix}
1 & 0 & 1 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{pmatrix} \cdot \begin{pmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

\[
M \quad M_T \quad M_L
\]
Recall

Transposes:

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

  $M = \begin{pmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{pmatrix}$,\hspace{1cm} $M^\top = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$

Recall:

- For matrix $M$, the transpose of the transpose is $M$:

  $(M^\top)^\top = M$
Recall

Transposes:

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

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

Recall:

• For matrices $M$ and $N$, the transpose of the product is the reversed product of the transposes:

$$(M \cdot N)^\top = N^\top \cdot M^\top$$
Recall

**Dot-Products:**

- The dot product of two vectors \( \vec{v} = (v_x, v_y, v_z)^\top \) and \( \vec{w} = (w_x, w_y, w_z)^\top \) is obtained by summing the product of the coefficients:
  \[
  \langle \vec{v}, \vec{w} \rangle = v_x \cdot w_x + v_y \cdot w_y + v_z \cdot w_z
  \]
- We can also express this as a matrix product:
  \[
  \langle \vec{v}, \vec{w} \rangle = \vec{v}^\top \cdot \vec{w} = (v_x \ v_y \ v_z) \cdot \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix}
  \]
Recall

Transposes and Dot-Products:

• If \( M \) is a matrix, and \( \vec{v} \) and \( \vec{w} \) are vectors, then:

\[
\langle \vec{v}, M\vec{w} \rangle = \vec{v}^\top \cdot (M \cdot \vec{w})
= (\vec{v}^\top \cdot M) \cdot \vec{w}
= (M^\top \cdot \vec{v})^\top \cdot \vec{w}
= \langle M^\top \vec{v}, \vec{w} \rangle
\]
Applying a Transformation

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

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

$$\langle \vec{n}, \vec{v} \rangle = \langle \vec{n}', M_L \vec{v} \rangle$$

$$= \langle M_L^\top \vec{n}', \vec{v} \rangle$$

$\uparrow$

$$\vec{n} = M_L^\top \vec{n}'$$

$\Downarrow$

$$\vec{n}' = (M_L^\top)^{-1} \vec{n}$$

Note that if the linear transformation $M_L$ is orthogonal (i.e. no scaling) then $(M_L^\top)^{-1} = M_L$. 
Applying a Transformation

- Position
  \[ p' = M(p) \]

- Direction
  \[ \vec{v}' = M_L(\vec{v}) \]

- Normal
  \[ \vec{n}' = (M_L^T)^{-1}(\vec{n}) \]

Affine
\[
\begin{pmatrix}
a & b & c & t_x \\
d & e & f & t_y \\
g & h & i & t_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

Translate
\[
\begin{pmatrix}
1 & 0 & 0 & t_x \\
0 & 1 & 0 & t_y \\
0 & 0 & 1 & t_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

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

Transform rays/intersections, not primitives

- For each node ...
  - Global-to-Local: Transform ray by matrix inverse
  - Local: Intersect transformed ray with primitives
  - Local-to-Global: Transform hit info by matrix

**Intersection Transformation:**

\[(p, \vec{n}) \rightarrow \left( M(p), M_L^{-T}(\vec{n}) \right)\]
Overview

• Transformation Hierarchies
  ◦ Scene graphs
  ◦ Ray casting

• Barycentric Coordinates
Barycentric Coordinates

Recall:

Given vertices $p_1$ and $p_2$, a point $q$ on the line segment between the vertices is the (non-negatively) weighted average of $p_1$ and $p_2$:

$$LS = \{ \alpha p_1 + \beta p_2 \mid \alpha + \beta = 1, \alpha, \beta \geq 0 \}$$
Barycentric Coordinates

Recall:

If the weights sum to one but are not positive, \( q \) is on the line but not on the line segment

\[
\alpha + \beta = 1
\]

\[
\alpha < 0
\]
Recall:

If the weights don’t sum to one, $q$ is not on the line unless the line through $p_1$ and $p_2$ passes through the origin.
Barycentric Coordinates

A triangle is defined by three non-collinear vertices.

Note:
A point $q$ is in the triangle if and only if $q$ is on the line segment between (without loss of generality) $p_1$ and a point $q'$ on edge $p_2p_3$. 
Claim:
Any point $q$ in the triangle, can be expressed as:

$$q = \alpha p_1 + \beta p_2 + \gamma p_3 \text{ with } \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

Proof:
The point $q$ is between $p_1$ and $q'$ on edge $p_2p_3$:

$$\Rightarrow q = \alpha p_1 + \beta q' \text{ with } \alpha + \beta = 1 \text{ and } \alpha, \beta \geq 0$$

$$\Rightarrow q' = \alpha' p_2 + \beta' p_3 \text{ with } \alpha' + \beta' = 1 \text{ and } \alpha', \beta' \geq 0$$
In Triangle ⇒ Positive Weights

Claim:
Any point $q$ in the triangle, can be expressed as:

$q = \alpha p_1 + \beta p_2 + \gamma p_3$ with $\alpha + \beta + \gamma = 1$ and $\alpha, \beta, \gamma \geq 0$

Proof:
The point $q$ is between $p_1$ and $q'$ on edge $p_2p_3$:

$\Rightarrow q = \alpha p_1 + \beta q'$ with $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$

$\Rightarrow q' = \alpha' p_2 + \beta' p_3$ with $\alpha' + \beta' = 1$ and $\alpha', \beta' \geq 0$

Putting this together we get $q$ as the linear sum:

$\circ q = \alpha p_1 + (\beta \alpha')p_2 + (\beta \beta')p_3$
In Triangle ⇒ Positive Weights

Claim:
Any point \( q \) in the triangle, can be expressed as:
\[
q = \alpha p_1 + \beta p_2 + \gamma p_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \quad \text{and} \quad \alpha, \beta, \gamma \geq 0
\]

Proof:
The point \( q \) is between \( p_1 \) and \( q' \) on edge \( p_2p_3 \):
\[
\Rightarrow q = \alpha p_1 + \beta q' \quad \text{with} \quad \alpha + \beta = 1 \quad \text{and} \quad \alpha, \beta \geq 0
\]
\[
\Rightarrow q' = \alpha' p_2 + \beta' p_3 \quad \text{with} \quad \alpha' + \beta' = 1 \quad \text{and} \quad \alpha', \beta' \geq 0
\]

Putting this together we get \( q \) as the linear sum:
\[
q = \alpha p_1 + (\beta \alpha') p_2 + (\beta \beta') p_3
\]
» Positive weights: \( \alpha, \beta \alpha', \beta \beta' \geq 0 \)
In Triangle $\Rightarrow$ Positive Weights

Claim:
Any point $q$ in the triangle, can be expressed as:
\[ q = \alpha p_1 + \beta p_2 + \gamma p_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0 \]

Proof:
The point $q$ is between $p_1$ and $q'$ on edge $p_2p_3$:
\[ \Rightarrow q = \alpha p_1 + \beta q' \quad \text{with} \quad \alpha + \beta = 1 \text{ and } \alpha, \beta \geq 0 \]
\[ \Rightarrow q' = \alpha' p_2 + \beta' p_3 \quad \text{with} \quad \alpha' + \beta' = 1 \text{ and } \alpha', \beta' \geq 0 \]

Putting this together we get $q$ as the linear sum:
\[ q = \alpha p_1 + (\beta \alpha')p_2 + (\beta \beta')p_3 \]
» Positive weights: $\alpha, \beta \alpha', \beta \beta' \geq 0$
» Summing to one:
\[ \alpha + \beta \alpha' + \beta \beta' \]
In Triangle $\Rightarrow$ Positive Weights

Claim:
Any point $q$ in the triangle, can be expressed as:

\[ q = \alpha p_1 + \beta p_2 + \gamma p_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \quad \text{and} \quad \alpha, \beta, \gamma \geq 0 \]

Proof:
The point $q$ is between $p_1$ and $q'$ on edge $p_2p_3$:

\[ q = \alpha p_1 + \beta q' \quad \text{with} \quad \alpha + \beta = 1 \quad \text{and} \quad \alpha, \beta \geq 0 \]

\[ q' = \alpha' p_2 + \beta' p_3 \quad \text{with} \quad \alpha' + \beta' = 1 \quad \text{and} \quad \alpha', \beta' \geq 0 \]

Putting this together we get $q$ as the linear sum:

- $q = \alpha p_1 + (\beta \alpha') p_2 + (\beta \beta') p_3$
  - » Positive weights: $\alpha, \beta \alpha', \beta \beta' \geq 0$
  - » Summing to one:
    \[ \alpha + \beta \alpha' + \beta \beta' = \alpha + \beta (\alpha' + \beta') \]
In Triangle ⇒ Positive Weights

Claim:
Any point $q$ in the triangle, can be expressed as:

$$q = \alpha p_1 + \beta p_2 + \gamma p_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \text{ and } \alpha, \beta, \gamma \geq 0$$

Proof:
The point $q$ is between $p_1$ and $q'$ on edge $p_2p_3$:

$$\Rightarrow q = \alpha p_1 + \beta q' \quad \text{with} \quad \alpha + \beta = 1 \text{ and } \alpha, \beta \geq 0$$

$$\Rightarrow q' = \alpha' p_2 + \beta' p_3 \quad \text{with} \quad \alpha' + \beta' = 1 \text{ and } \alpha', \beta' \geq 0$$

Putting this together we get $q$ as the linear sum:

$\circ\; q = \alpha p_1 + (\beta \alpha')p_2 + (\beta \beta') p_3$

- Positive weights: $\alpha, \beta \alpha', \beta \beta' \geq 0$
- Summing to one:
  $$\alpha + \beta \alpha' + \beta \beta' = \alpha + \beta (\alpha' + \beta') = \alpha + \beta$$
In Triangle ⇒ Positive Weights

Claim:
Any point \( q \) in the triangle, can be expressed as:
\[ q = \alpha p_1 + \beta p_2 + \gamma p_3 \quad \text{with} \quad \alpha + \beta + \gamma = 1 \quad \text{and} \quad \alpha, \beta, \gamma \geq 0 \]

Proof:
The point \( q \) is between \( p_1 \) and \( q' \) on edge \( p_2p_3 \):
\[ \Rightarrow q = \alpha p_1 + \beta q' \quad \text{with} \quad \alpha + \beta = 1 \quad \text{and} \quad \alpha, \beta \geq 0 \]
\[ \Rightarrow q' = \alpha' p_2 + \beta' p_3 \quad \text{with} \quad \alpha' + \beta' = 1 \quad \text{and} \quad \alpha', \beta' \geq 0 \]

Putting this together we get \( q \) as the linear sum:
- \[ q = \alpha p_1 + (\beta \alpha')p_2 + (\beta \beta')p_3 \]
  » Positive weights: \( \alpha , \beta \alpha' , \beta \beta' \geq 0 \)
  » Summing to one:
  \[ \alpha + \beta \alpha' + \beta \beta' = \alpha + \beta (\alpha' + \beta') = \alpha + \beta = 1 \]
Claim:
If a point $q$ can be expressed as:

$$q = \alpha p_1 + \beta p_2 + \gamma p_3$$
with $\alpha + \beta + \gamma = 1$ and $\alpha, \beta, \gamma \geq 0$

then $q$ is in the triangle.

Proof:

A point $q'$ on the segment between $p_2$ and $p_3$
Positive Weights $\Rightarrow$ In Triangle

Claim:
If a point $q$ can be expressed as:

\[ q = \alpha p_1 + \beta p_2 + \gamma p_3 \]

with $\alpha + \beta + \gamma = 1$ and $\alpha, \beta, \gamma \geq 0$ then $q$ is in the triangle.

Proof:

A point $q$ on the segment between $p_1$ and $q'$
Barycentric Coordinates

The barycentric coordinates of a point \( q \): \[
q = \alpha p_1 + \beta p_2 + \gamma p_3
\]
let us express \( q \) as the weighted average of the triangle vertices.

- The weights \( \alpha \), \( \beta \), and \( \gamma \) tell us, relatively, how close the point \( q \) is to \( p_1 \), \( p_2 \), and \( p_3 \) (respectively).
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

```c
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
  ○ In 3D models, information is often associated with vertices rather than triangles (e.g. color, normals, etc.)
Barycentric Coordinates

For example:

- We can associate the same normal/color to every point on the face of a triangle by computing:

\[
\mathbf{n} = \frac{\mathbf{p}_2 - \mathbf{p}_1 \times \mathbf{p}_3 - \mathbf{p}_1}{\| \mathbf{p}_2 - \mathbf{p}_1 \times \mathbf{p}_3 - \mathbf{p}_1 \|}
\]
Barycentric Coordinates

For example:

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

  \[ \vec{n} = \frac{(p_2 - p_1) \times (p_3 - p_1)}{\| (p_2 - p_1) \times (p_3 - p_1) \|} \]

This gives rise to flat shading/coloring across the faces.
Barycentric Coordinates

Instead:

- We can associate a normal with each vertex:
  
  \[ T = ((p_1, \vec{n}_1), (p_2, \vec{n}_2), (p_3, \vec{n}_3)) \]

\[ \Rightarrow \text{The normal at a point } q \text{ in the triangle is the interpolation of the normals at the vertices:} \]

\[ \vec{n}(q) = \frac{\alpha_q \vec{n}_1 + \beta_q \vec{n}_2 + \gamma_q \vec{n}_3}{\| \alpha_q \vec{n}_1 + \beta_q \vec{n}_2 + \gamma_q \vec{n}_3 \|} \]
Barycentric Coordinates

Instead:

- We can associate a normal with each vertex:
  \[ T = (p_1, \vec{n}_1), (p_2, \vec{n}_2), (p_3, \vec{n}_3) \]

⇒ The normal at a point \( q \) in the triangle is the interpolation of the normals at the vertices:
Barycentric Coordinates

Instead:

- We can associate normals to every vertex:
  \[ T = p_1, n_1, p_2, n_2, p_3, n_3 \]

  ⇒ The normal at a point \( q \) in the triangle is the interpolation of the normals at the vertices:

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

interpolation of the normals at the vertices:

Triangle Normals

Interpolated Point Normals