Lecture 8: Coordinate Frame Transformations
# Breakdown of Lectures

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Introduction &amp; Image Formation</td>
</tr>
<tr>
<td>2.</td>
<td>Programming with OpenGL</td>
</tr>
<tr>
<td>3.</td>
<td>OpenGL: Pipeline Architecture</td>
</tr>
<tr>
<td>4.</td>
<td>OpenGL: An Example Program</td>
</tr>
<tr>
<td>5.</td>
<td>Vertex and Fragment Shaders 1</td>
</tr>
<tr>
<td>6.</td>
<td>Vertex and Fragment Shaders 2</td>
</tr>
<tr>
<td>7.</td>
<td>Representation and Coordinate Systems</td>
</tr>
<tr>
<td>8.</td>
<td>Coordinate Frame Transformations</td>
</tr>
<tr>
<td>9.</td>
<td>Transformations and Homogeneous Coordinates</td>
</tr>
<tr>
<td>10.</td>
<td>Input, Interaction and Callbacks</td>
</tr>
<tr>
<td>11.</td>
<td>More on Callbacks</td>
</tr>
<tr>
<td>12.</td>
<td>Mid-semester Test</td>
</tr>
<tr>
<td></td>
<td>Study break</td>
</tr>
<tr>
<td>13.</td>
<td>3D Hidden Surface Removal</td>
</tr>
<tr>
<td>14.</td>
<td>Mid term-test solution and project discussion</td>
</tr>
<tr>
<td>15.</td>
<td>Computer Viewing</td>
</tr>
<tr>
<td>16.</td>
<td>Shading</td>
</tr>
<tr>
<td>17.</td>
<td>Shading Models</td>
</tr>
<tr>
<td>18.</td>
<td>Shading in OpenGL</td>
</tr>
<tr>
<td>19.</td>
<td>Texture Mapping</td>
</tr>
<tr>
<td>20.</td>
<td>Texture Mapping in OpenGL</td>
</tr>
<tr>
<td>21.</td>
<td>Hierarchical Modelling</td>
</tr>
<tr>
<td>22.</td>
<td>3D Modelling: Subdivision Surfaces</td>
</tr>
<tr>
<td>23.</td>
<td>Animation Fundamentals and Quaternions</td>
</tr>
<tr>
<td>24.</td>
<td>Skinning</td>
</tr>
</tbody>
</table>
Content

• Learn how to define and change coordinate frames
• Derive homogeneous coordinate transformation matrices
• Introduce standard transformations
  — Rotation, Translation, Scaling, Shear
Coordinate Frame

- Basis vectors alone cannot represent points
- We can add a single point, the *origin*, to the basis vectors to form a *coordinate frame*
Representation in a Coordinate Frame

• A coordinate system (or coordinate frame) is determined by \( (\mathbf{P}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) \)

• Within this coordinate frame, every vector \( \mathbf{v} \) can be written as

\[
\mathbf{v} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3
\]

Every point can be written as

\[
\mathbf{P} = \mathbf{P}_0 + \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + \beta_3 \mathbf{v}_3
\]

for some \( \alpha_1, \alpha_2, \alpha_3, \) and \( \beta_1, \beta_2, \beta_3 \)
Homogeneous Coordinates

- Consider the point $P$ and the vector $v$, where

$$P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3$$
$$v = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$$

- They appear to have similar representations:

$$P = [\beta_1, \beta_2, \beta_3]^T, \quad v = [\alpha_1, \alpha_2, \alpha_3]^T$$

which confuses the point with the vector.

A vector has no position.

Vector can be placed anywhere.

point: fixed
A Single Representation

• Assuming $0 \cdot \mathbf{P} = \mathbf{0}$ and $1 \cdot \mathbf{P} = \mathbf{P}$, we can write

$$\mathbf{v} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 + 0 \cdot \mathbf{P}_0$$

$$\mathbf{P} = \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + \beta_3 \mathbf{v}_3 + \mathbf{P}_0 = \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + \beta_3 \mathbf{v}_3 + 1 \cdot \mathbf{P}_0$$

• Thus we obtain the four-dimensional **homogeneous coordinate** representation

$$\mathbf{v} = [\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad 0]^T \mathbf{v}$$

$$\mathbf{P} = [\beta_1 \quad \beta_2 \quad \beta_3 \quad 1]^T \mathbf{v}$$
Homogeneous Coordinates

• The homogeneous coordinate form for a three dimensional point $[x \ y \ z]^T$ is given as

$p = [x \ y \ z \ 1]^T \rightarrow [wx \ wy \ wz \ w]^T = [x' \ y' \ z' \ w]^T$

• We return to a three dimensional point (for $w \neq 0$) by

$$x \leftarrow x'/w$$
$$y \leftarrow y'/w$$
$$z \leftarrow z'/w$$

• If $w = 0$, the representation is that of a vector

• Homogeneous coordinates replace points in three dimensions by lines through the origin in four dimensions

• For $w = 1$, the representation of a point is $[x \ y \ z \ 1]^T$
Homogeneous Coordinates and Computer Graphics

• Homogeneous coordinates are key to all computer graphics systems
  – All standard transformations (rotation, translation, scaling) can be implemented with matrix multiplications using 4 x 4 matrices
  – Hardware pipeline works with 4 dimensional representations
  – For **orthographic viewing**, we can maintain $w = 0$ for vectors and $w = 1$ for points
  – For **perspective** we need a **perspective division**
Representing the Second Basis in Terms of the First

- How can we relate \( \mathbf{u} \) with \( \mathbf{v} \)?
- Each of the basis vectors \( \mathbf{u}_1, \mathbf{u}_2, \) and \( \mathbf{u}_3 \) are vectors that can be represented in terms of the first set of basis vectors,
  i.e.,

\[
\begin{align*}
\mathbf{u}_1 &= \gamma_{11} \mathbf{v}_1 + \gamma_{12} \mathbf{v}_2 + \gamma_{13} \mathbf{v}_3 \\
\mathbf{u}_2 &= \gamma_{21} \mathbf{v}_1 + \gamma_{22} \mathbf{v}_2 + \gamma_{23} \mathbf{v}_3 \\
\mathbf{u}_3 &= \gamma_{31} \mathbf{v}_1 + \gamma_{32} \mathbf{v}_2 + \gamma_{33} \mathbf{v}_3
\end{align*}
\]

for some \( \gamma_{11}, \ldots, \gamma_{33} \)
Representing the Second Basis in Terms of the First (cont.)

- \( \mathbf{u}_1 = \gamma_{11} \mathbf{v}_1 + \gamma_{12} \mathbf{v}_2 + \gamma_{13} \mathbf{v}_3 \) can be written as:

\[
\mathbf{u}_1 = [\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3] \begin{bmatrix} \gamma_{11} \\ \gamma_{12} \\ \gamma_{13} \end{bmatrix} = \mathbf{V} \begin{bmatrix} \gamma_{11} \\ \gamma_{12} \\ \gamma_{13} \end{bmatrix}
\]

- Similarly, \( \mathbf{u}_2 = \gamma_{21} \mathbf{v}_1 + \gamma_{22} \mathbf{v}_2 + \gamma_{23} \mathbf{v}_3 \) and \( \mathbf{u}_3 = \gamma_{31} \mathbf{v}_1 + \gamma_{32} \mathbf{v}_2 + \gamma_{33} \mathbf{v}_3 \) can be written as:

\[
\mathbf{u}_2 = \mathbf{V} \begin{bmatrix} \gamma_{21} \\ \gamma_{22} \\ \gamma_{23} \end{bmatrix}
\]

\[
\mathbf{u}_3 = \mathbf{V} \begin{bmatrix} \gamma_{31} \\ \gamma_{32} \\ \gamma_{33} \end{bmatrix}
\]
Representing the Second Basis in Terms of the First (cont.)

- We can put the terms \( \gamma_{11}, \ldots, \gamma_{33} \) into a \( 3 \times 3 \) matrix:

\[
M = \begin{bmatrix}
\gamma_{11} & \gamma_{12} & \gamma_{13} \\
\gamma_{21} & \gamma_{22} & \gamma_{23} \\
\gamma_{31} & \gamma_{32} & \gamma_{33}
\end{bmatrix}
\]

then we have:

\[
[\mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{u}_3] = \mathbf{V} M^T
\]

That is,

\[
\mathbf{U} = \mathbf{V} M^T
\]

The superscript T denotes matrix transpose
The same vector $\mathbf{w}$ represented in two coordinate systems

- We can write

$$\mathbf{w} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3$$

$$\mathbf{w} = \beta_1 \mathbf{u}_1 + \beta_2 \mathbf{u}_2 + \beta_3 \mathbf{u}_3$$

as follows:

$$\mathbf{w} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \mathbf{V} \mathbf{a}$$

$$\mathbf{w} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \mathbf{U} \mathbf{b}$$

Let's call this $3 \times 3$ matrix $\mathbf{V}$

Each $\mathbf{v}_i$ is a column vector of 3 components
Representing the Second Basis in Terms of the First (cont.)

- In this example, we have \( w = V a \) and \( w = U b \).
- So

\[
V a = U b
\]

- With \( U = V M^T \), we have

\[
V a = V M^T b \\
\Rightarrow a = M^T b
\]

- Thus, \( a \) and \( b \) are related by \( M^T \)
Change of Coordinate Frames

• We can apply a similar process in homogeneous coordinates to the representations of both points and vectors.

Consider two coordinate frames:

\((P_0, v_1, v_2, v_3)\)
\((Q_0, u_1, u_2, u_3)\)

• Any point or vector can be represented in either coordinate frame.
• We can represent \((Q_0, u_1, u_2, u_3)\) in terms of \((P_0, v_1, v_2, v_3)\)
Representing One Coordinate Frame in Terms of the Other

- We can extend what we did with the change of basis vectors:

\[
\begin{align*}
\mathbf{u}_1 &= \gamma_{11} \mathbf{v}_1 + \gamma_{12} \mathbf{v}_2 + \gamma_{13} \mathbf{v}_3 \\
\mathbf{u}_2 &= \gamma_{21} \mathbf{v}_1 + \gamma_{22} \mathbf{v}_2 + \gamma_{23} \mathbf{v}_3 \\
\mathbf{u}_3 &= \gamma_{31} \mathbf{v}_1 + \gamma_{32} \mathbf{v}_2 + \gamma_{33} \mathbf{v}_3 \\
\mathbf{Q}_0 &= \gamma_{41} \mathbf{v}_1 + \gamma_{42} \mathbf{v}_2 + \gamma_{43} \mathbf{v}_3 + \mathbf{P}_0
\end{align*}
\]

by replacing the $3 \times 3$ matrix $\mathbf{M}$ by a $4 \times 4$ matrix as follows:

\[
\mathbf{M} = \begin{bmatrix}
\gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\
\gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\
\gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\
\gamma_{41} & \gamma_{42} & \gamma_{43} & 1
\end{bmatrix}
\]
Working with Representations

- Within the two coordinate frames any point or vector has a representation of the same form:
  \[ \mathbf{a} = [\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4] \] in the first frame
  \[ \mathbf{b} = [\beta_1 \quad \beta_2 \quad \beta_3 \quad \beta_4] \] in the second frame
  where \( \alpha_4 = \beta_4 = 1 \) for points and \( \alpha_4 = \beta_4 = 0 \) for vectors and
  \[ \mathbf{a} = \mathbf{M}^T \mathbf{b} \]

- The matrix \( \mathbf{M}^T \) is \( 4 \times 4 \) and specifies an affine transformation in homogeneous coordinates
Transformations in Graphics pipeline

We had considered the following coordinate systems:

1. Object (or model) coordinates
2. World coordinates
3. Eye (or camera) coordinates
4. Clip coordinates
5. Normalized device coordinates
6. Window (or screen) coordinates

Can be combined in a model-view transform

Affine transform brings representations in the eye-frame

The six frames are w.r.t. immediate-mode rendering.
Moving the Camera

Camera and object frame in default positions

\[ \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

model-view matrix
Moving the Camera

Camera frame is fixed, we are placing object frame relative to the object frame.

\[
A = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & -d \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

- The application programmer works in the object/world coordinates (a.k.a. application frame)
The World and Camera Coordinate Frames

• When we work with representations, we work with $n$-tuples or arrays of scalars
• Changes in coordinate frame are then defined by $4 \times 4$ matrices
• In OpenGL, the base frame that we start with is the world frame
• Eventually we represent entities in the camera frame by changing the world representation using the model-view matrix
• Initially these frames are the same (i.e, $M=I$)
An example

We consider two reference frames that have basis vector relation

\[ u_1 = v_1, \]
\[ u_2 = v_1 + v_2, \]
\[ u_3 = v_1 + v_2 + v_3. \]

Let's say the reference point does not change, so

\[ Q_0 = P_0. \]

Our matrix \( M^T \) would be:

\[
M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
\[
M^T = \begin{bmatrix}
1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Only accounting for rotation
An example

Now, we want our frames to have different reference point....

Let's say, to the point \( Q_0 \) that has the following representation in the original system.

\[
Q_0 = P_0 + v_1 + 2v_2 + 3v_3,
\]

The \( M^T \) for such a setting will be:

\[
M^T = \begin{bmatrix}
1 & 1 & 1 & 1 \\
0 & 1 & 1 & 2 \\
0 & 0 & 1 & 3 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Also accounting for translation
A Few Common Transformations

• **Rigid transformation:** The $4 \times 4$ matrix has the form:

$$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$$

where $R$ is a $3 \times 3$ rotation matrix and $t \in \mathbb{R}^{3\times1}$ is a translation vector. Rigid transformation preserves everything (*angle* (this means the *shape*), *length*, *area*, etc).

• **Similarity transformation:** The matrix has the form:

$$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} \text{ or } \begin{bmatrix} R & t \\ 0^T & s' \end{bmatrix}$$

where $s, s' \neq 1$. Similarity transformation preserves *angle*, ratios of *lengths* and of *areas*.
A Few Common Transformations (cont.)

- **Affine transformation:** The $4 \times 4$ matrix has the form:
  \[
  \begin{bmatrix}
  A & t \\
  0^\intercal & 1
  \end{bmatrix}
  \]
  where $A$ can be any $3 \times 3$ non-singular matrix and $t \in \mathbb{R}^3$ is a translation vector. Affine transformation preserves *parallelism*, ratios of lengths.

- **Perspective transformation:** The matrix can be any non-singular $4 \times 4$ matrix. Perspective transformation matrix preserves *cross ratios* (i.e., ratio of ratios of lengths).
A Few Common Transformations (cont.)

• Rigid transformation is equivalent to a change in coordinate frames. It has 6 degrees of freedom (dof) i.e. 3 rotations + 3 translations (along each of the three axes)

• Similarity transformation has 7 dof (an additional scaling)

• Affine transformation has 12 dof
  – 3 rotations + 3 translations + 3 scaling + 3 shear
General Transformations

• A transformation maps points to other points and/or vectors to other vectors
Pipeline Implementation

$T$ \hspace{1cm} (from application program)

$u$ \hspace{1cm} $T(u)$ \hspace{1cm} $v$ \hspace{1cm} $T(v)$

transformation \hspace{1cm} rasterizer

vertices \hspace{1cm} (before transformation) \hspace{1cm} vertices \hspace{1cm} (after transformation)
Further Reading


• Sec 3.7 to 3.9