Objectives

• Learn how to define and change coordinate frames
• Introduce standard transformations
  – Rotation, Translation, Scaling, Shear
• Derive homogeneous coordinate transformation matrices
• Learn to build arbitrary transformation matrices from simple transformations
Coordinate Frame

- Basis vectors alone cannot represent a point.
- We can add a single point, the *origin*, to the basis vectors to form a *coordinate frame*.
- Recall the affine space.

![Coordinate Frame Diagram](image-url)
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

• Since \(0 \cdot \mathbf{P} = \mathbf{0}\) and \(1 \cdot \mathbf{P} = \mathbf{P}\) then 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

\[
\mathbf{v} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \end{bmatrix}^T \mathbf{v} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \end{bmatrix}
\]

\[
\mathbf{P} = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & 1 \end{bmatrix}^T \mathbf{v}
\]
Homogeneous Coordinates

• The homogeneous coordinate form for a three dimensional point \([x\ y\ z]^T\) is given as
  \[
p = [x\ y\ z\ 1]^T = [wx\ wy\ wz\ w]^T = [x'\ y'\ z'\ w]^T
  \]

• 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

• Note that 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 \( \mathbf{a} \) with \( \mathbf{b} \)?

• 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.,

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

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

- 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:

$$[u_1 \quad u_2 \quad u_3] = VM^T$$

That is,

$$U = VM^T$$

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:

$$(\mathbf{v}_1 \mathbf{v}_2 \mathbf{v}_3) \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \mathbf{V} \mathbf{a}$$
$$(\mathbf{u}_1 \mathbf{u}_2 \mathbf{u}_3) \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \mathbf{U} \mathbf{b}$$

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:

\[ \mathbf{a} = [\alpha_1 \ \alpha_2 \ \alpha_3 \ \alpha_4] \] in the first frame
\[ \mathbf{b} = [\beta_1 \ \beta_2 \ \beta_3 \ \beta_4] \] 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
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 $t \in \mathbb{R}^3$ 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*.
• **Affine transformation:** The $4 \times 4$ matrix has the form:

\[
\begin{bmatrix}
A & t \\
0^T & 1
\end{bmatrix}
\]

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 because the 4 elements on the last row of the matrix are fixed
  - 3 rotations + 3 translations + 3 scaling + 3 shear.

- Perspective transformation has 11 dof [?? Check]
  - 3 rotations + 3 translations + 5 intrinsic camera parameters.
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, $M = I$)
Moving the Camera

• If objects are on both sides of $z = 0$, we must move the camera coordinate frame, e.g.,

$$
M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & -d \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
$$
General Transformations

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

\[ v = T(u) \]

\[ Q = T(P) \]
Affine Transformations

• Preserving parallel lines
• Characteristic of many physically important transformations
  – Rigid body transformations: rotation, translation
  – Scaling, shear
• Importance in graphics is that we need only transform endpoints of line segments and let implementation draw line segment between the transformed endpoints
Pipeline Implementation

\[ T \text{ (from application program)} \]

\[ \begin{align*}
T(u) & \\
T(v) & \\
\end{align*} \]

vertices (before transformation) \[ \rightarrow \]
vertices (after transformation) \[ \rightarrow \]
pixels

frame buffer
Notation

• We will be working with both coordinate-free representations of transformations and representations within a particular frame
  – $P, Q, R$: points in an affine space
  – $p, q, r$: representations of points
    - array of 4 scalars in homogeneous coordinates
  – $u, v, w$: representations of vectors in an affine space
    - array of 4 scalars in homogeneous coordinates
• $\alpha, \beta, \gamma$: scalars
Translation

• Move (translate, displace) a point to a new location

• Displacement determined by a vector \( \mathbf{d} \)
  – Three degrees of freedom

\[
P' = P + \mathbf{d}
\]
How many ways?

- Although we can move a point to a new location in an infinite number of ways, when we move many points (of a rigid object) there is usually only one way.

![Diagram showing object and translation vector](image.png)
Translation using Representations

- Using the homogeneous coordinate representation in some frame
  - \( \mathbf{p} = [x \ y \ z \ 1]^T \)
  - \( \mathbf{p}' = [x' \ y' \ z' \ 1]^T \)
  - \( \mathbf{d} = [d_x \ d_y \ d_z \ 0]^T \)
- Hence \( \mathbf{p}' = \mathbf{p} + \mathbf{d} \) or
  - \( x' = x + d_x \)
  - \( y' = y + d_y \)
  - \( z' = z + d_z \)

Note that this expression is in four dimensions and expresses point = vector + point
Translation Matrix

- We can also express translation using a $4 \times 4$ matrix $T$ in homogeneous coordinates

\[ p' = Tp \]

where

\[ T = T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

- This form is better for implementation because all affine transformations can be expressed this way and multiple transformations can be concatenated together
Further Reading


• Sec 3.7 to 3.9