# Three dimensional affine transformation

## *Affine transformations*

- Affine transformations are those that preserve parallel lines.
- Most transformations we require are affine, the most important being:
  - Scaling
  - Rotation
  - Translation
- Other more complex transforms can be built from these.
- An example of a non-affine transformation:
  - Perspective projection (parallels not preserved).

## Definition

3D affine transformation refers to a mathematical operation that can be applied to 3D objects or points in space. It involves a combination of translation, rotation, scaling, and shearing. These transformations allow you to change the position, orientation, and size of objects in a 3D coordinate system. They are commonly used in computer graphics, animation, and virtual reality to manipulate and transform 3D objects.

# Three-Dimensional Affine Transformations

Affine transformations in three dimensions allow us to manipulate 3D objects by altering their position, orientation, and shape. A 3D point is expressed as:

$$P = X\vec{\boldsymbol{i}} + Y\vec{\boldsymbol{j}} + Z\vec{\boldsymbol{k}}$$

where

$$\vec{\boldsymbol{i}} = \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \vec{\boldsymbol{j}} = \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \vec{\boldsymbol{k}} = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

We use homogeneous coordinates and column vectors such that points are written as follows:

$$P = \begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

Generally, a 3D affine transformation is written in matrix form as:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

such that transforming point $P$ into point $Q$ with matrix $M$ is mathematically expressed as $Q = MP$ .

## Elementary Transformations

- Translation:

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Scaling:

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotation around the $x$ -axis:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotation around the $y$ -axis:

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotation around the $z$ -axis:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Properties of affine transformations

Here are some useful properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Midpoints map to midpoints (in fact, ratios are always preserved)


$$\text{ratio} = \frac{\|\mathbf{pq}\|}{\|\mathbf{qr}\|} = \frac{s}{t} = \frac{\|\mathbf{p'q'}\|}{\|\mathbf{q'r'}\|}$$

## Affine transformations in OpenGL

OpenGL maintains a "modelview" matrix that holds the current transformation **M.**

The modelview matrix is applied to points (usually vertices of polygons) before drawing.

It is modified by commands including:

- `glLoadIdentity()` $\mathbf{M} \leftarrow \mathbf{I}$
  - set **M** to identity
- `glTranslatef(`$t_x$`, `$t_y$`, `$t_z$`)` $\mathbf{M} \leftarrow \mathbf{MT}$
  - translate by ($t_x$, $t_y$, $t_z$)
- `glRotatef(θ, x, y, z)` $\mathbf{M} \leftarrow \mathbf{MR}$
  - rotate by angle $\theta$ about axis (x, y, z)
- `glScalef(`$s_x$`, `$s_y$`, `$s_z$`)` $\mathbf{M} \leftarrow \mathbf{MS}$
  - scale by ($s_x$, $s_y$, $s_z$)

Note that OpenGL adds transformations by *postmultiplication* of the modelview matrix.