Computer Graphics

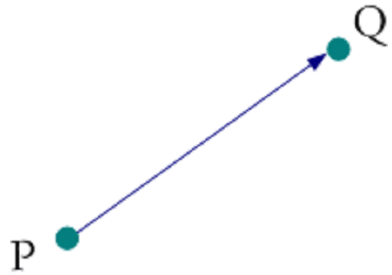
Affine Transformations

Hyewon SEO

Contents

- **Affine Transformations in 2D and 3D**
  - **Matrix form**
  - **Homogeneous representation**
- **Standard transformations**
  - **Rotation (2D/3D)**
  - **Translation (2D/3D)**
  - **Scaling (2D/3D)**
  - **Shear (2D/3D)**
- **Combination of transformations**
- **Change of coordinate systems**

Affine space



- V = Q – P
- Q = V + P
  - Addition between a vector and a point!!

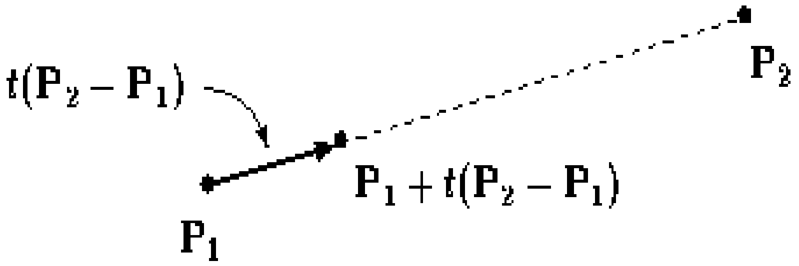
- Affine space
  - Extension of the vector space by treating the vector and the point ‘homogeneously’
- Affine operations
  - Addition of two vectors
  - Multiplication of a scalar and a vector (or a point but with a constraint)
  - Addition of a vector and a point

Affine combination

- **Let P_1 and P_2 be points in an affine space. Consider the expression:**

$$\mathbf{P} = \mathbf{P}_1 + t(\mathbf{P}_2 - \mathbf{P}_1)$$

$$\mathbf{P} = (1-t)\mathbf{P}_1 + t\mathbf{P}_2$$



- **Affine combination of two points P_1 and P_2**

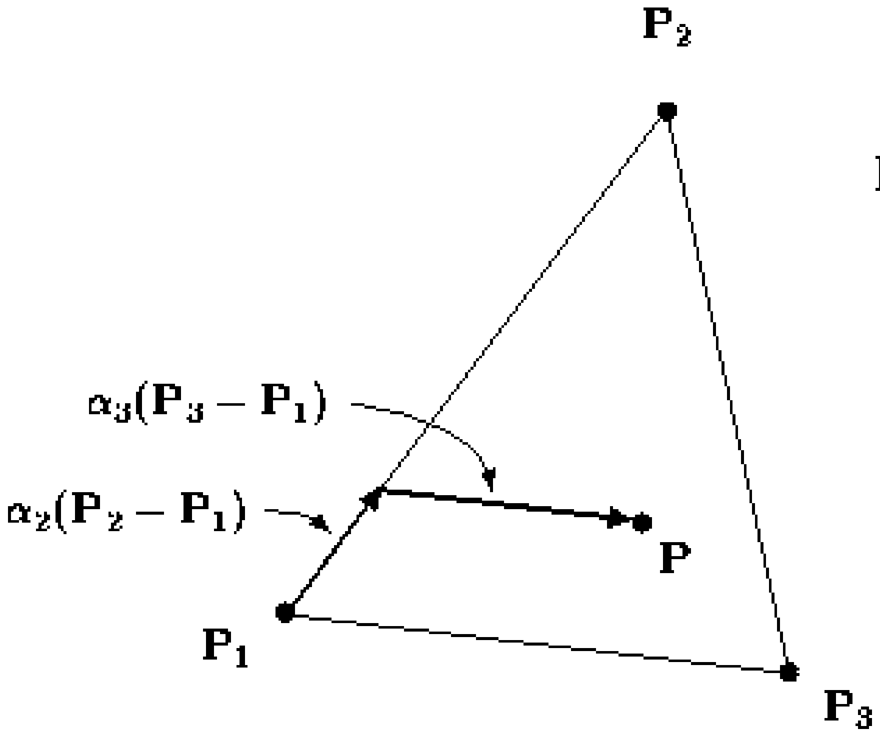
$$\mathbf{P} = \alpha_1\mathbf{P}_1 + \alpha_2\mathbf{P}_2\,, \quad \alpha_1 + \alpha_2 = 1$$

- **Generalization:**

$$\sum_i \alpha_i P_i = P_0 + \underbrace{\sum_i \alpha_i (P_i - P_0)}_{v}$$

$$\alpha_1 + \alpha_2 + \cdots + \alpha_n = 1$$

An example – affine combination



$$\mathbf{P} = \alpha_1 \mathbf{P_1} + \alpha_2 \mathbf{P_2} + \alpha_3 \mathbf{P_3}$$

$$\alpha_1 = \alpha_2 = \frac{1}{4}, \quad \alpha_3 = \frac{1}{2}$$

(i) $0 \leq \alpha 1, \alpha 2, \alpha 3 \leq 1$

(ii) α_i **< 0 or** α_i **>1**

(iii) α_i **= 0**

Affine Transformations

- Map from one affine space to another, $\Re^n \rightarrow \Re^n$
  (n **:dimension of the space) that** **preserves affine combinations**

$$X(\sum \alpha_i P_i) = \sum \alpha_i X(P_i)$$

  - Preserves lines and poly-lines
  - maps parallel lines to parallel lines

- Importance of transformation in 3D modeling:
  - Moving objects in the space: adjust their position and orientation and scale in the space.
  - Specifying parent/child relationships (skeleton)

Affine Transformations

- **General form of affine transformation:**
  - **a linear transformation followed by a translation.**

$$p' = M.p + t$$

  - p**: a point in the space**
  - t**: translation vector**
  - M**: matrix of the linear transformation**

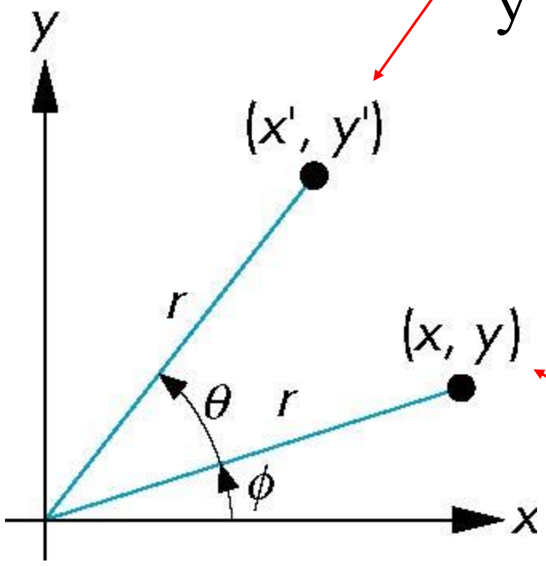
- **Rotation, translation, scaling, shear**

Rotation (2D)

- **Consider rotation about the origin by θ degrees**

$$x' = r \cos(\phi + \theta)$$
$$y' = r \sin(\phi + \theta)$$

$$\boxed{\begin{aligned} x' &= x \cos\theta - y \sin\theta \\ y' &= x \sin\theta + y \cos\theta \end{aligned}}$$



$$x = r \cos\phi$$
$$y = r \sin\phi$$

2D Rotation

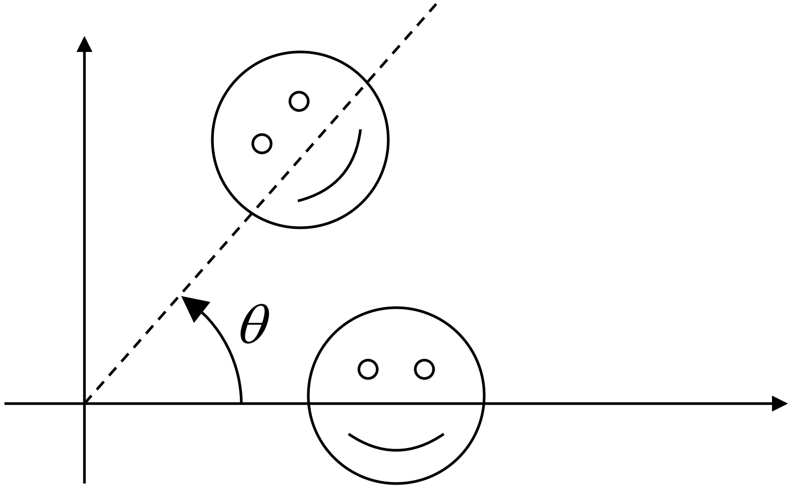
General form of rotation:

$$p' = M . p$$

Rotation in 2D:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

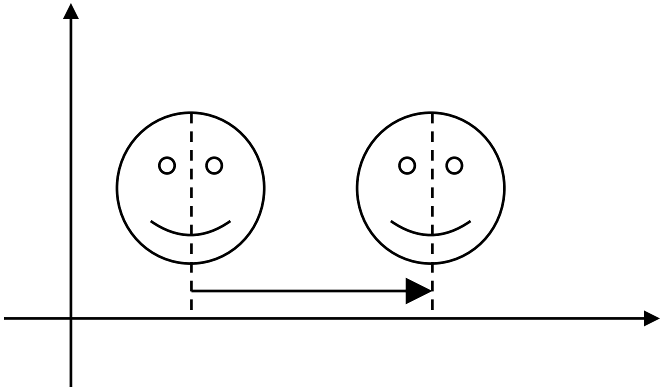
Example:



Translation in 2D

- **Point translation**

$$p' = p + t$$



- **Matrix form?**
- **Is it Affine?**
- **Preserves lines?**
- **Does it work for points and vectors?**

Translation in 2D -- answers

- **No 2x2 matrix notation**
- **It is Affine**
- **Different interpretations for vectors and points**
- **We need a new representation**

Translation in 2D

How can we write $p' = p + t$ in the matrix form?

The solution: homogeneous coordinate system.

$$p' = p + t$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_y \\ 0 & 1 & t_x \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Additional coordinate to handle the translation

Translation in 2D

Matrix form of affine transformation:

$$p' = M . p + t$$

with the homogeneous coordinate system:

$$\begin{pmatrix} p' \\ 1 \end{pmatrix} = \begin{pmatrix} M & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix}$$

Homogeneous coordinate system

Homogeneous Representation

shape	point	vector
Previous notation 2D and 3D	$\begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$
homogeneous 2D, 3D	$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$	$\begin{bmatrix} v_1 \\ v_2 \\ 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}$

Homogeneous coordinate system in 2D and 3D

Useful representation

- **Matrix notation for translation**
  - **seamless integration of translation, rotation, scaling, and shear.**

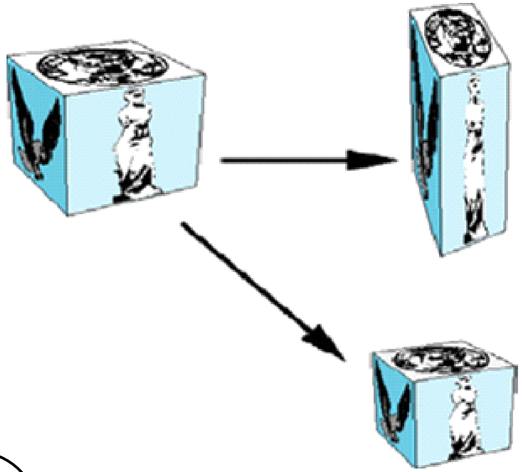
- **Separation between points and vectors:**
  - **Points:** $p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$**, they are moved by translation**
  - **Vectors:** $v = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$**, invariant under translation**

Homogeneous Coordinates and Computer Graphics

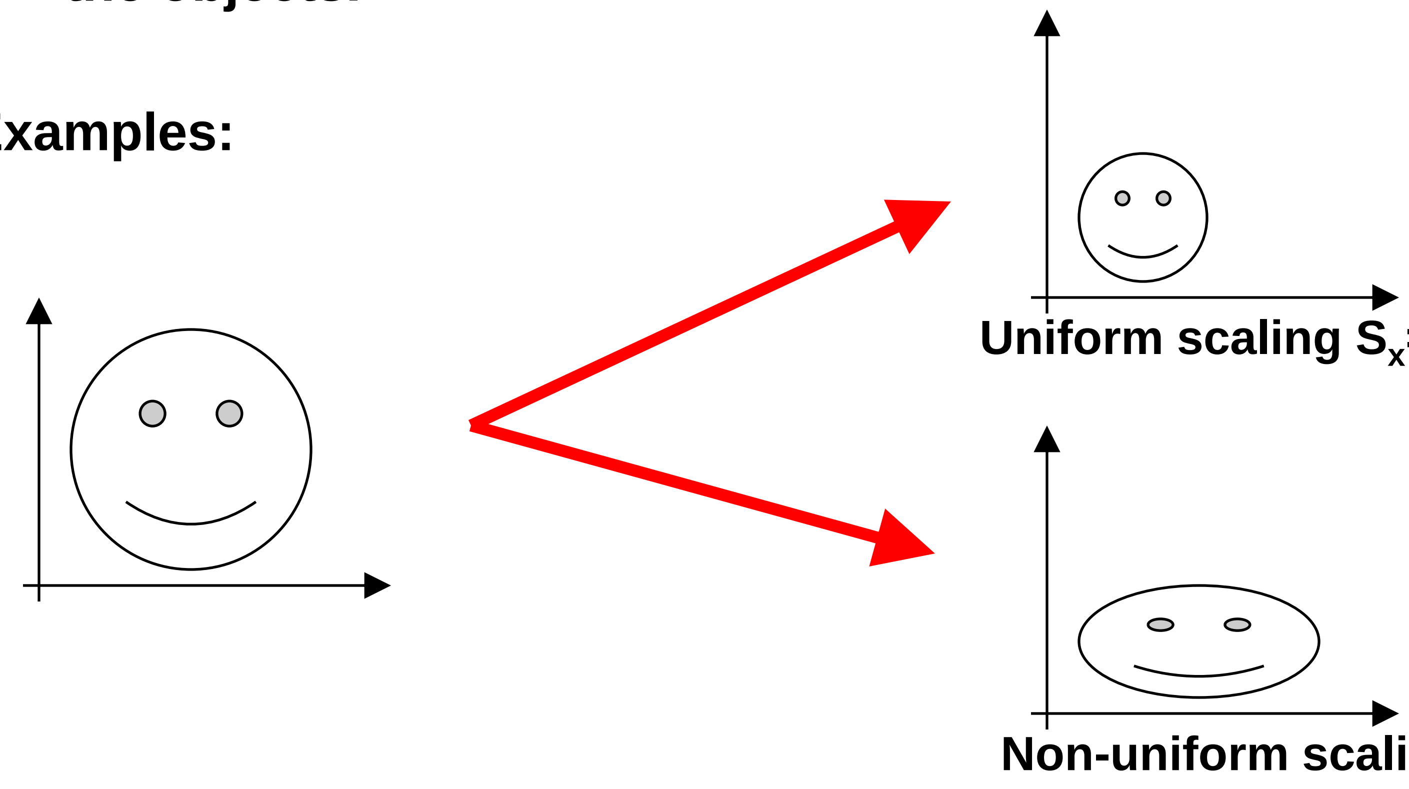
- **Homogeneous coordinates are key to all computer graphics systems**
  - **All standard transformations (rotation, translation, scaling) can be implemented by matrix multiplications with 4 x 4 matrices**
  - **Hardware pipeline works with 4 dimensional representations**
  - **For orthographic viewing, we can maintain $w=0$ for vectors and $w=1$ for points**
  - **For perspective we need a *perspective division***

Scaling

- **Change the size of the object**
- **Achieved by applying a scaling factor S_x and S_y to the vertices coordinates of the objects.**



Examples:



Scale Transformations in 2D

- **Scaling relative to *(0,0)*:**

$$x' = s_x x$$
$$y' = s_y y$$

- **Matrix form:**

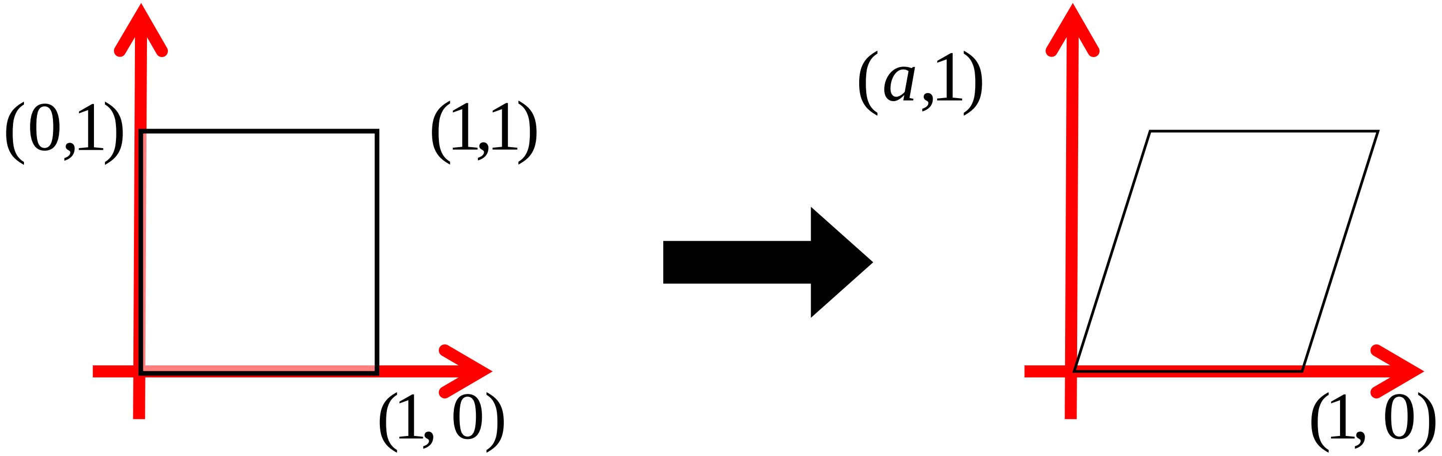
$$\begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} s_x \cdot x \\ s_y \cdot y \end{pmatrix}$$

- **Scaling relative to (x_f, y_f):** $x' = x S_x + (1 - S_x) x_f$
  $y' = y S_y + (1 - S_y) y_f$

2D Shear Transformation

- **Along *X-axis*:**

$$Sh_x = \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + ay \\ y \end{pmatrix}$$



Inverses

- **Although we could compute inverse matrices by general formula, we can use simple geometric observations**

  - **Translation:** $T^{-1}(d_x, d_y, d_z) = T(-d_x, -d_y, -d_z)$

  - **Rotation:** $R^{-1}(\theta) = R(-\theta)$
    - **Holds for any rotation matrix**
    - **Note that since cos(-q) = cos(q) and sin(-q)=-sin(q)**

    $$R^{-1}(\theta) = R^{T}(\theta)$$

  - **Scaling:** $S^{-1}(s_x, s_y, s_z) = S(1/s_x, 1/s_y, 1/s_z)$

2D Rotating About an Arbitrary Point

- **Pivot point: V_x, V_y**

- **Steps**
  1. **Translate through ____**
  2. **Rotate about the origin through angle ____**
  3. **Translated back through ____**

- **The matrix form:**

$$\begin{bmatrix} 1 & 0 & V_x \\ 0 & 1 & V_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -V_x \\ 0 & 1 & -V_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & dx \\ \sin\theta & \cos\theta & dy \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{where} \quad \begin{aligned} dx &= -\cos\theta V_x + \sin\theta VY + V_x \\ dy &= -\sin\theta V_x - \cos\theta V_y + V_y \end{aligned}$$

Translation and Scaling in 3D

Translation:

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Scaling

$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Shear Transformation in 3D

- **Along x-direction**

- **2D matrix form**

$$\begin{bmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **3D:** $\begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$

- **it is an Affine map**

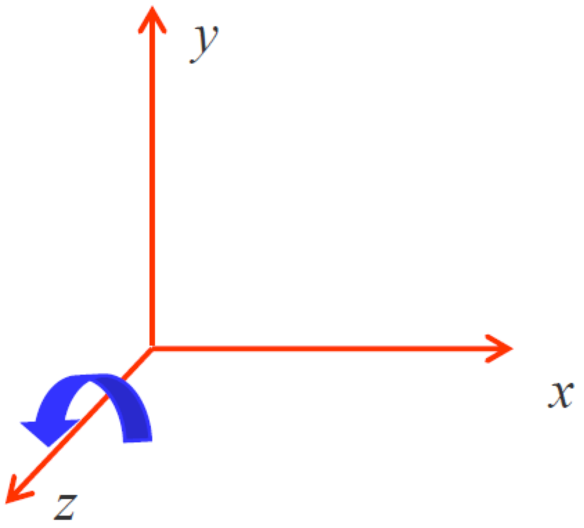
Rotation in 3D

- Rotation about a coordinate axis
- z-axis here: A simple extension of planar rotation

- Matrix form

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

c=cos(θ), s= sin(θ)

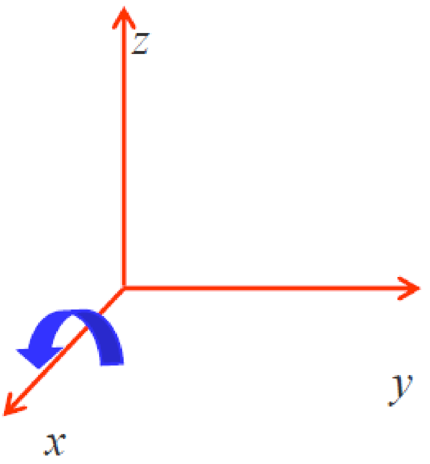


$$P' = R_z(\theta).P$$

Rotation about x-axis and y-axis

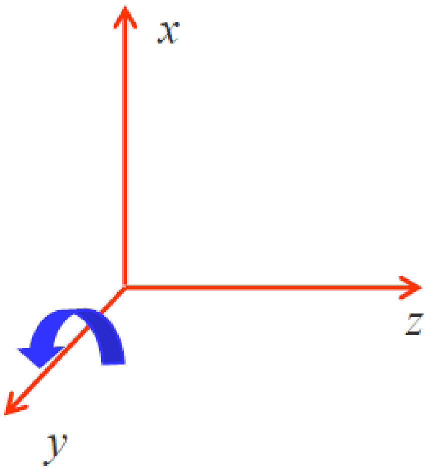
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_x(\theta).P$$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_y(\theta).P$$



Composing Affine Transformations

- **Composing or concatenating the transformation**
- **Concatenation of two Affine transformations is also Affine**
- **Functional form and matrix form**

$$P'' = T_2(P') = T_2(T_1(P)) = (T_2T_1)(P) = T(P)$$
$$P'' = M_2P' = M_2M_1P = MP$$
$$M = M_2M_1$$

- **Reverse order!!**

Composing Affine Transformations

- **Matrices are a convenient and efficient way to represent a sequence of transformations**
  - **Efficiency with premultiplication**
  - **Matrix multiplication is associative**

    p' = (T * (R * (S***p**)))

    p' = (T*R*S) * **p**

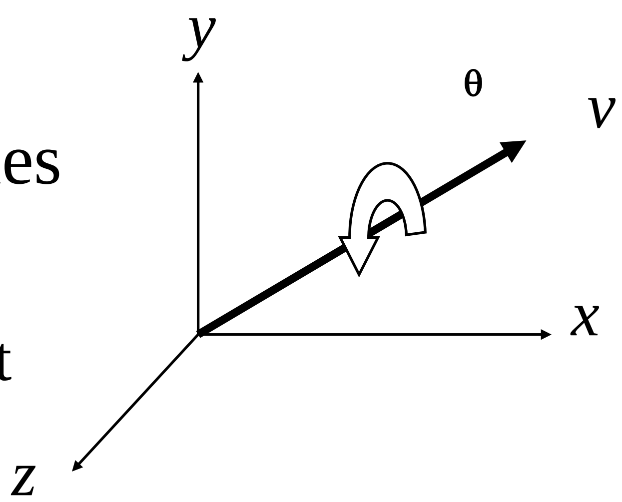
- **Be aware: order of transformations matters**
  - **Matrix multiplication is not commutative**

    p' = T * R * S * p

General Rotation About the Origin

A rotation by θ about an arbitrary axis can be decomposed into the concatenation of rotations about the *x*, *y*, and *z* axes.

$$\mathbf{R}_v(\theta) = \mathbf{R}_z(\theta_z)\,\mathbf{R}_y(\theta_y)\,\mathbf{R}_x(\theta_x)$$



$\theta_x\ \theta_y\ \theta_z$ are called the Euler angles

Note that rotations do not commute.
We can use rotations in another order but with different angles

Rotation About a Fixed Point other than the Origin

1. Move fixed point to origin
2. Rotate
3. Move fixed point back

$M = T(\mathbf{p_f})\, R(\theta)\, T(-\mathbf{p_f})$