TRANSFORMATIONS

OUTLINE

- Introduce coordinate systems
- Introduce homogeneous coordinates
- Standard transformations
  - Translation
  - Rotation
  - Scaling
  - Shear

LINEAR INDEPENDENCE

- A set of vectors $v_1, v_2, \ldots, v_n$ is ***linearly independent*** if

  $\alpha_1 v_1 + \alpha_2 v_2 + .. \alpha_n v_n = 0$ iff $\alpha_1 = \alpha_2 = \ldots = 0$

- If a set of vectors is linearly independent, we cannot represent one in terms of the others
- If a set of vectors is linearly dependent, at least one can be written in terms of the others

DIMENSION

- In a vector space, the maximum number of linearly independent vectors is fixed and is called the ***dimension*** of the space
- In an *n*-dimensional space, any set of n linearly independent vectors form a ***basis*** for the space
- Given a basis $v_1, v_2, \ldots, v_n$, any vector v can be written as

$$v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$$

REPRESENTATION

- Until now we have been talking about geometric entities without using any frame of reference, such as a coordinate system
- Need a frame of reference to relate points and objects to our physical world.
  - For example, where is a point? Can't answer without a reference system
  - World coordinates
  - Camera coordinates

COORDINATE SYSTEMS

- Consider a basis $v_1, v_2, \ldots, v_n$
- A vector is written $v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$
- The list of scalars $\{\alpha_1, \alpha_2, \ldots \alpha_n\}$ is the ***representation*** of v with respect to the given basis
- We can write the representation as a row or column array of scalars

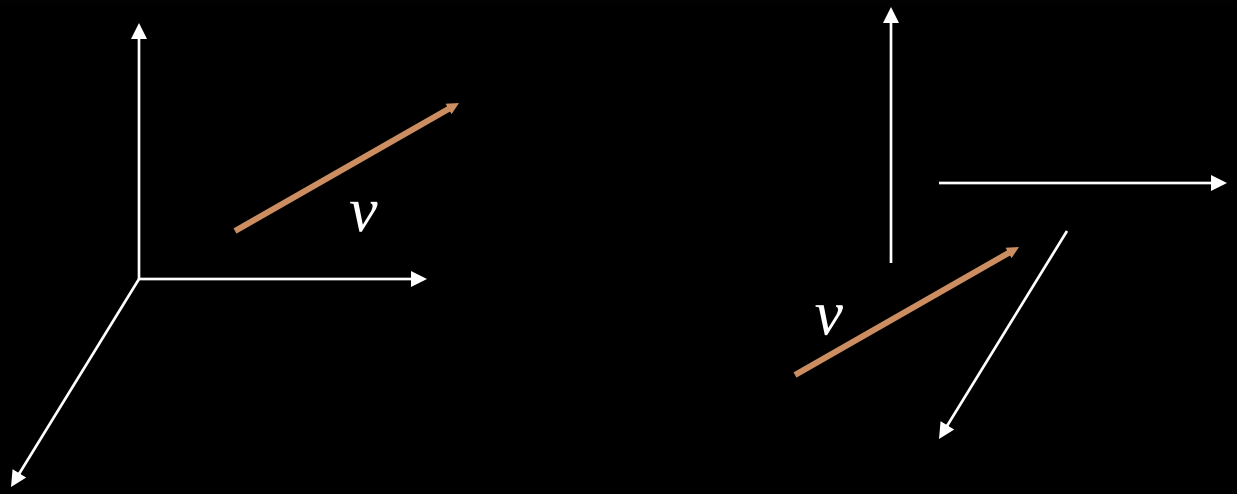
$$\mathbf{a} = [\alpha_1 \quad \alpha_2 \quad \ldots \alpha_n]^T = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ \alpha_n \end{bmatrix}$$

EXAMPLE

- $v=2v_1+3v_2-4v_3$
- $\mathbf{a}=[2, 3, -4]^T$
- Note that this representation is with respect to a particular basis
- For example, in OpenGL we will start by representing vectors using the object basis but later the system needs a representation in terms of the camera or eye basis

COORDINATE SYSTEMS

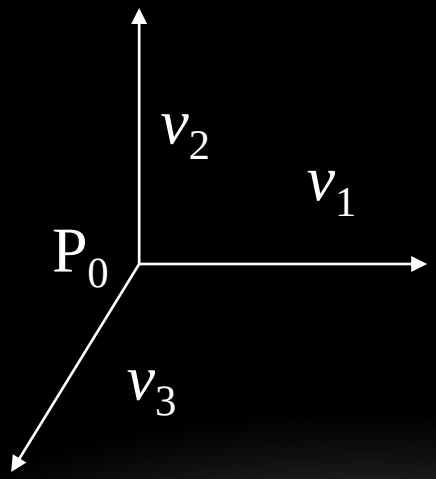
- Which is correct?



- Both are because vectors have no fixed location

FRAMES

- A coordinate system is insufficient to represent points
- If we work in an affine space we can add a single point, the ***origin***, to the basis vectors to form a ***frame***



REPRESENTATION IN A FRAME

- Frame determined by (P_0, v_1, v_2, v_3)
- Within this frame, every vector can be written as

$$v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$$

- Every point can be written as

$$P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \ldots + \beta_n v_n$$

CONFUSING POINTS AND VECTORS

Consider the point and the vector

$$P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \ldots + \beta_n v_n$$

$$v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$$

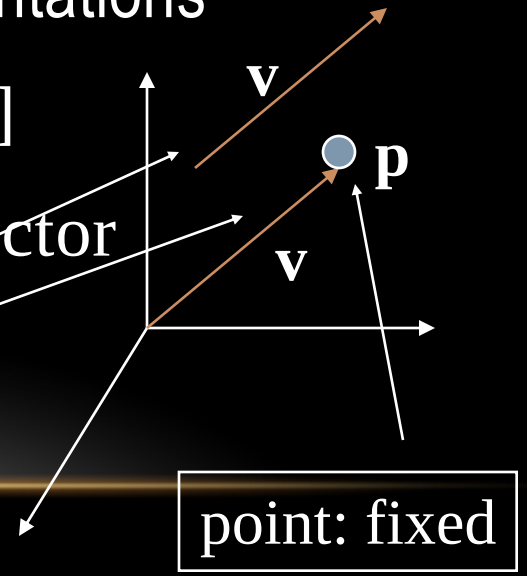
They appear to have the similar representations

$$\mathbf{p} = [\beta_1\ \beta_2\ \beta_3] \qquad \mathbf{v} = [\alpha_1\ \alpha_2\ \alpha_3]$$

which confuses the point with the vector

A vector has no position

Vector can be placed anywhere



A SINGLE REPRESENTATION

If we define $0 \bullet P = \mathbf{0}$ and $1 \bullet P = P$ then we can write

$$\begin{aligned} v &= \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 \\ &= [\alpha_1 \; \alpha_2 \; \alpha_3 \; 0 \,] \, [v_1 \; v_2 \; v_3 \; P_0]^T \\ P &= P_0 + \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 \\ &= [\beta_1 \; \beta_2 \; \beta_3 \; 1 \,] \, [v_1 \; v_2 \; v_3 \; P_0]^T \end{aligned}$$

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

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

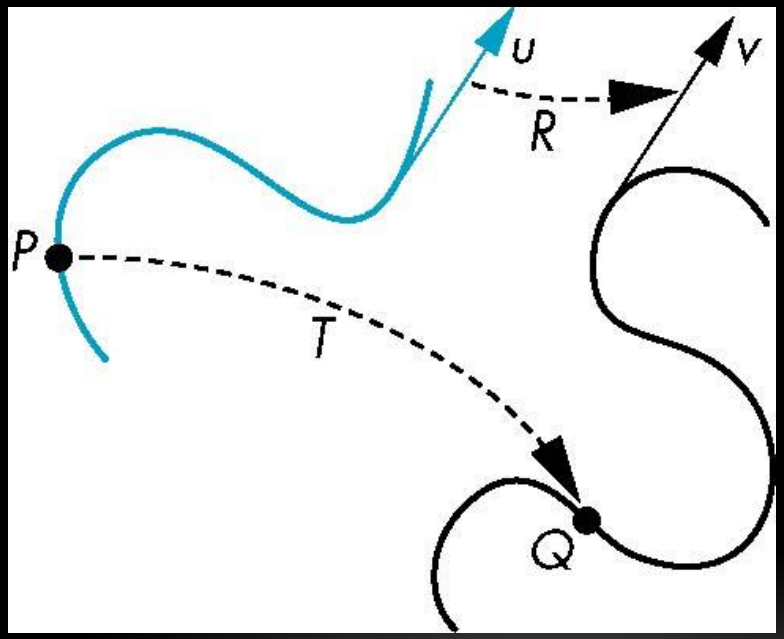
$$\mathbf{p} = [\beta_1 \; \beta_2 \; \beta_3 \; 1 \,]^T$$

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*

GENERAL TRANSFORMATIONS

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



v=T(u)

AFFINE TRANSFORMATIONS

- Line preserving
- 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

SIDE NOTE: DEFINITION OF AFFINE

af·fine

/əˈfīn/

noun

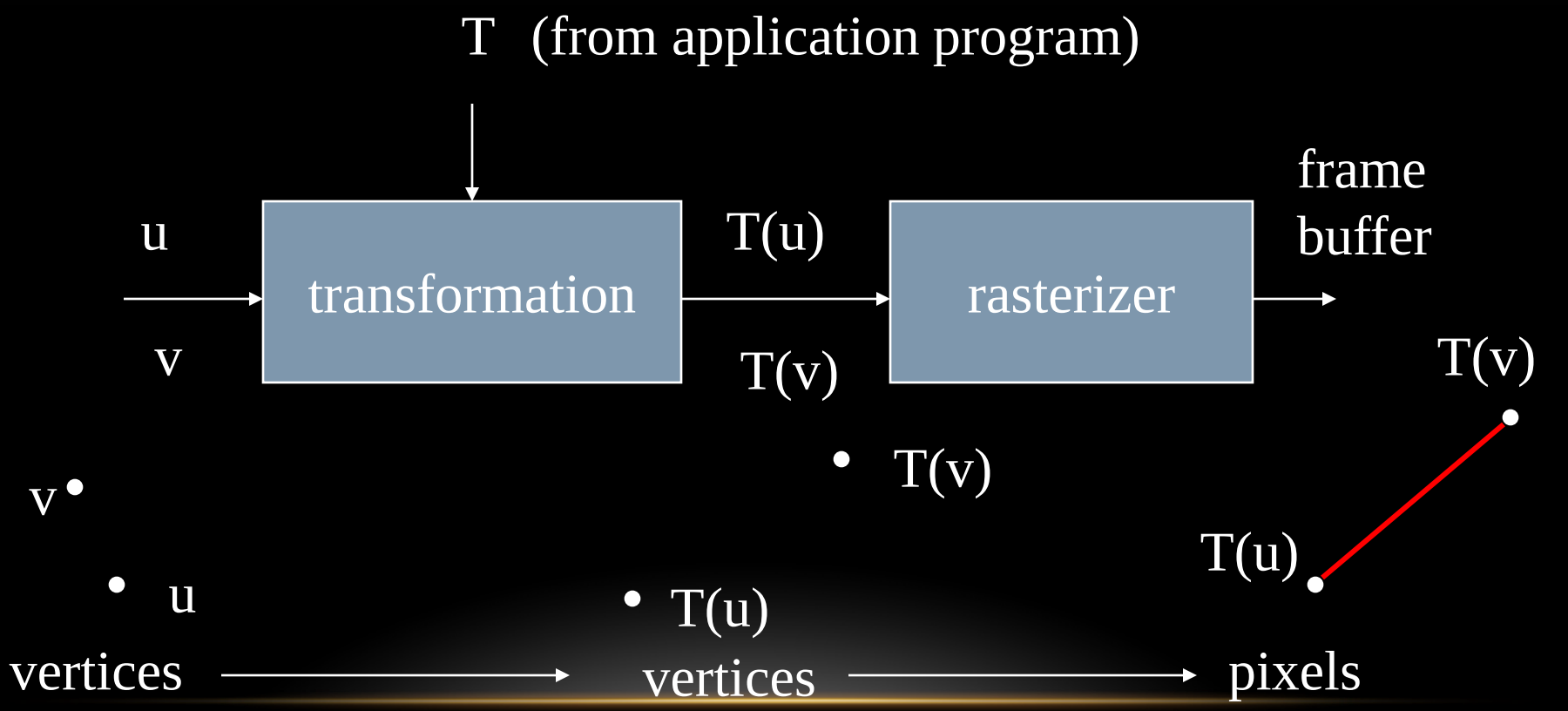
1. a relative by marriage

adjective

1. allowing for or preserving parallel relationships

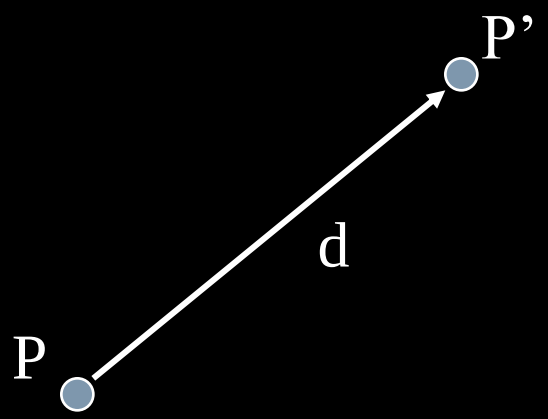
Powered by Oxford Dictionaries

PIPELINE IMPLEMENTATION



TRANSLATION

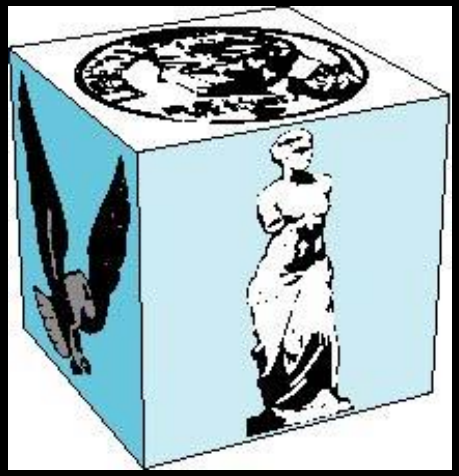
- Move (translate, displace) a point to a new location



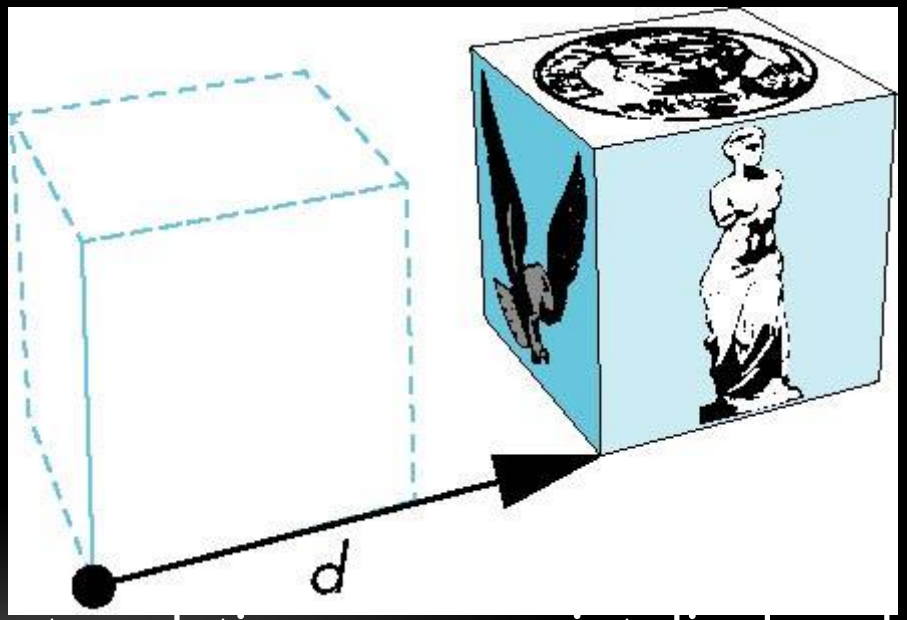
- Displacement determined by a vector d
  - P'=P+d

HOW MANY WAYS?

Although we can move a point to a new location in infinite ways, when we move many points there is usually only one way



object



translation: every point displaced by same vector

TRANSLATION USING REPRESENTATIONS

Using the homogeneous coordinate representation in some frame

$\mathbf{p}=[\ x\ y\ z\ 1]^T$

$\mathbf{p}'=[x'\ y'\ z'\ 1]^T$

$\mathbf{d}=[dx\ dy\ dz\ 0]^T$

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 x 4 matrix $\mathbf{T}$ in homogeneous coordinates

$\mathbf{p}'=\mathbf{T}\mathbf{p}$ where

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

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

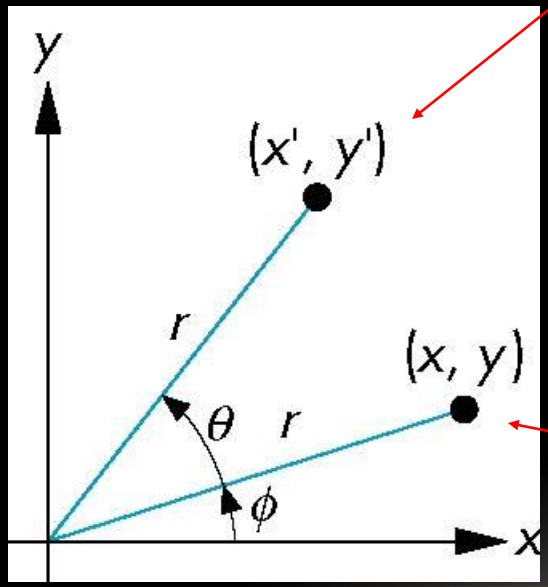
ROTATION (2D)

Consider rotation about the origin by θ degrees

- radius stays the same, angle increases by θ

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

$$x' = x \cos\theta - y \sin\theta$$
$$y' = x \sin\theta + y \cos\theta$$



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

ROTATION ABOUT THE Z AXIS

- Rotation about z axis in three dimensions leaves all points with the same z
  - Equivalent to rotation in two dimensions in planes of constant z

$$x' = x \cos\theta - y \sin\theta$$
$$y' = x \sin\theta + y \cos\theta$$
$$z' = z$$

  - or in homogeneous coordinates

$$\mathbf{p}' = \mathbf{R}_{\mathbf{Z}}(\theta)\mathbf{p}$$

ROTATION MATRIX

$$\mathbf{R} = \mathbf{R}_Z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ROTATION ABOUT X AND Y AXES

- Same argument as for rotation about *z* axis
  - For rotation about *x* axis, *x* is unchanged
  - For rotation about *y* axis, *y* is unchanged

$$\mathbf{R} = \mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R} = \mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

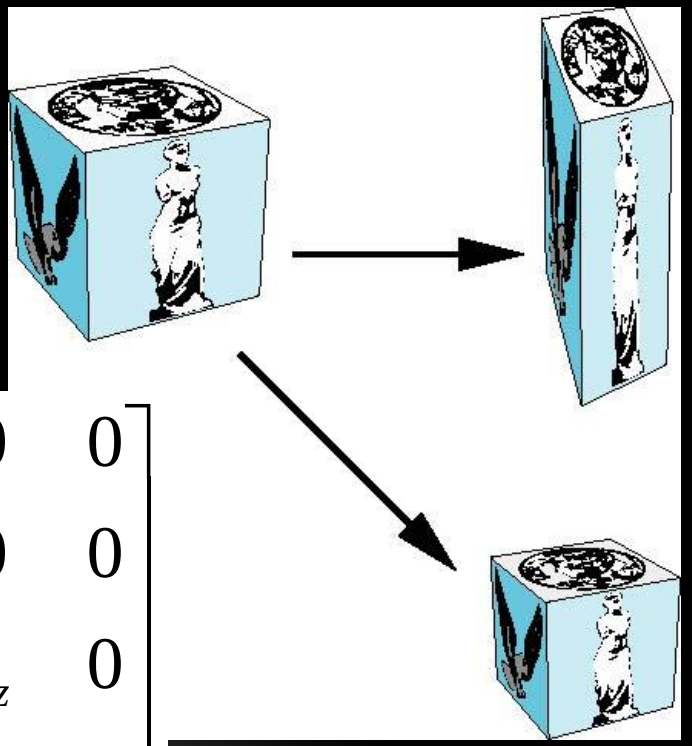
SCALING

Expand or contract along each axis (fixed point of origin)

$$x' = s_x x$$
$$y' = s_y x$$
$$z' = s_z x$$

$$\mathbf{p}' = \mathbf{S}\mathbf{p}$$

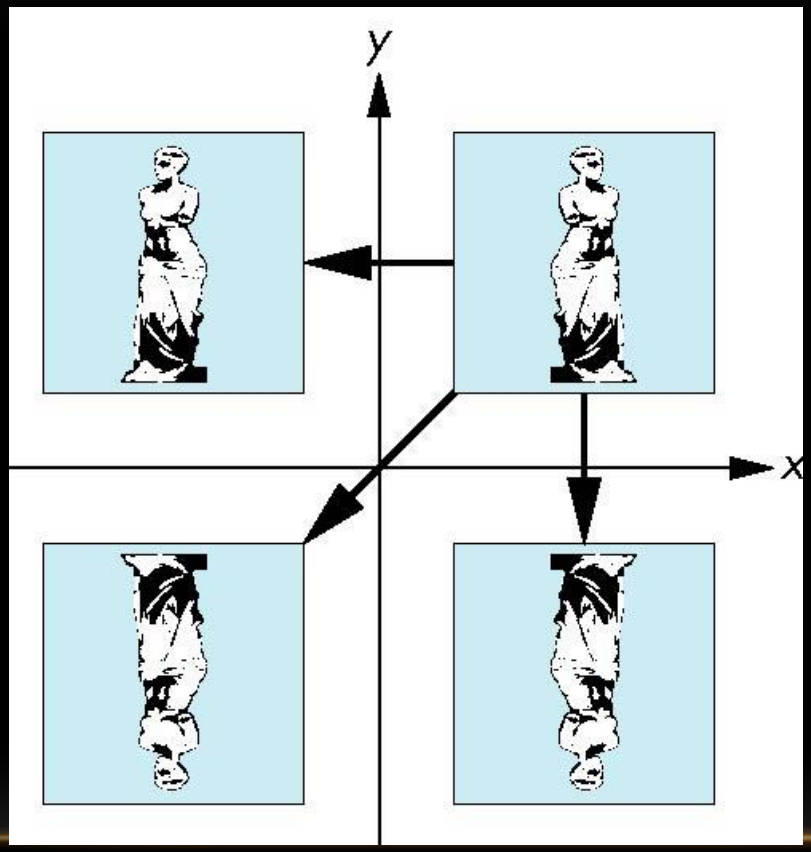
$$\mathbf{S} = \mathbf{S}(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



REFLECTION

corresponds to negative scale factors

$s_x = -1\ s_y = 1$



original

$s_x = -1\ s_y = -1$

$s_x = 1\ s_y = -1$

INVERSES

- Although we could compute inverse matrices by general formulas, we can use simple geometric observations
  - Translation: $\mathbf{T}^{-1}(d_x, d_y, d_z) = \mathbf{T}(-d_x, -d_y, -d_z)$
  - Rotation: $\mathbf{R}^{-1}(\theta) = \mathbf{R}(-\theta)$
    - Holds for any rotation matrix
    - Note that since $\cos(-\theta) = \cos(\theta)$ and $\sin(-\theta) = -\sin(\theta)$

    $\mathbf{R}^{-1}(\theta) = \mathbf{R}^{T}(\theta)$
  - Scaling: $\mathbf{S}^{-1}(s_x, s_y, s_z) = \mathbf{S}(1/s_x, 1/s_y, 1/s_z)$

CONCATENATION

- We can form arbitrary affine transformation matrices by multiplying together rotation, translation, and scaling matrices
- Because the same transformation is applied to many vertices, the cost of forming a matrix $\mathbf{M}=\mathbf{ABCD}$ is not significant compared to the cost of computing $\mathbf{Mp}$ for many vertices $\mathbf{p}$
- The difficult part is how to form a desired transformation from the specifications in the application

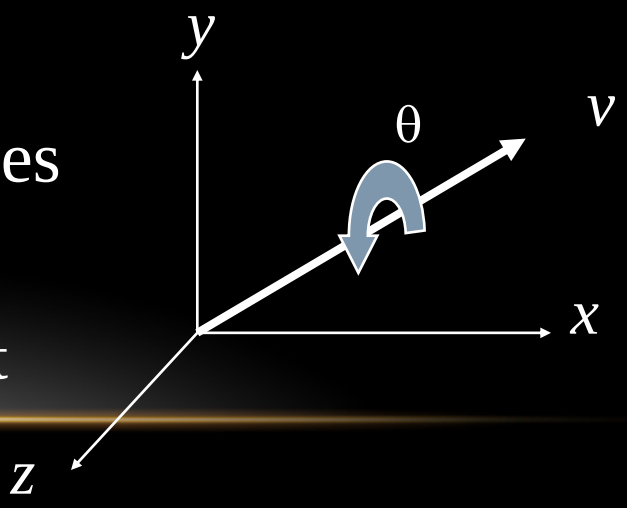
GENERAL ROTATION ABOUT THE ORIGIN

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

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

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

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



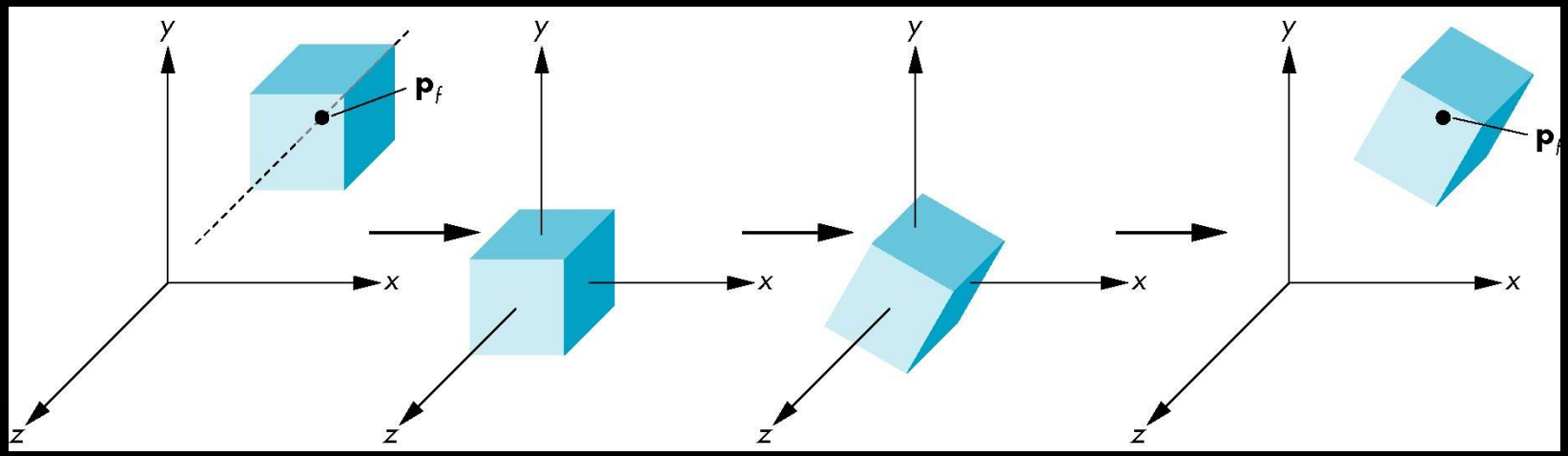
ROTATION ABOUT A FIXED POINT OTHER THAN THE ORIGIN

Move fixed point to origin

Rotate

Move fixed point back

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



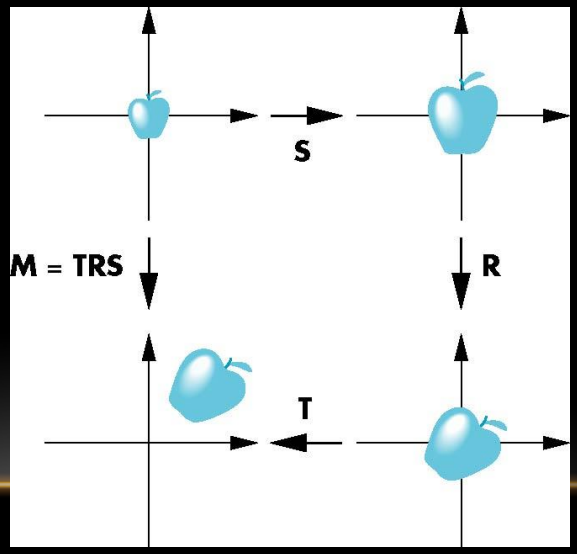
INSTANCING

- In modeling, we often start with a simple object centered at the origin, oriented with the axis, and at a standard size
- We apply an *instance transformation* to its vertices to

Scale

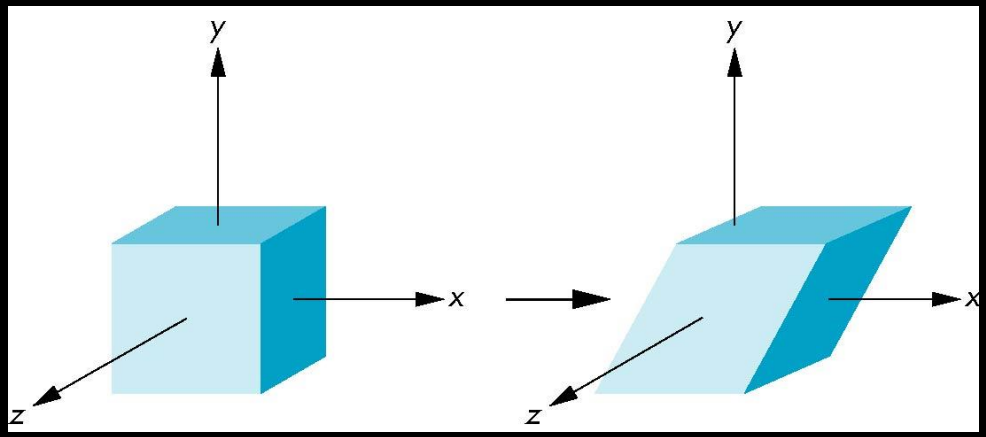
Orient

Locate



SHEAR

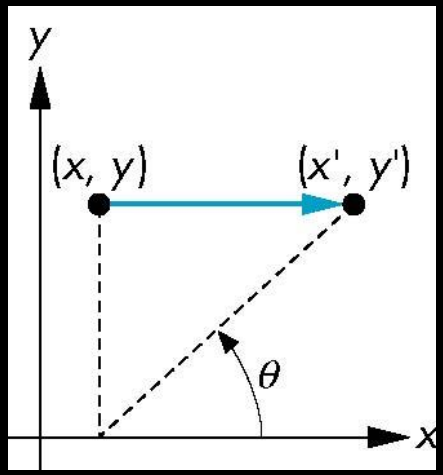
- Helpful to add one more basic transformation
- Equivalent to pulling faces in opposite directions



SHEAR MATRIX

Consider simple shear along x axis

$$x' = x + y \cot \theta$$
$$y' = y$$
$$z' = z$$



$$\mathbf{H}(\theta) = \begin{bmatrix} 1 & \cot\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

SUMMARY

- Introduce coordinate systems
- Introduce homogeneous coordinates
- Standard transformations
  - Translation
  - Rotation
  - Scaling
  - Shear