GEOMETRIC OBJECTS AND TRANSFORMATION

Lecture 2

Comp3080 Computer Graphics

HKBU

GEOMETRIC OBJECTS

OBJECTIVES

- Introduce the elements of geometry
  - Scalars
  - Vectors
  - Points
- Develop mathematical operations among them in a coordinate-free manner
- Define basic primitives
  - Line segments
  - Polygons

BASIC ELEMENTS

- Geometry is the study of the relationships among objects in an n-dimensional space
  - In computer graphics, we are interested in objects that exist in three dimensions
- Want a minimum set of primitives from which we can build more sophisticated objects
- We will need three basic elements
  - Scalars
  - Vectors
  - Points

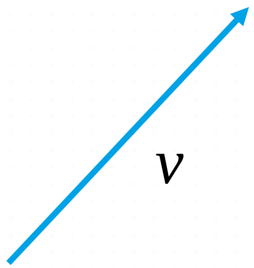
COORDINATE-FREE GEOMETRY

- When we learned simple geometry, most of us started with a Cartesian approach
  - Points were at locations in space $\mathbf{p} = (x, y, z)$
  - We derived results by algebraic manipulations involving these coordinates
- This approach was nonphysical
  - Physically, points exist regardless of the location of an arbitrary coordinate system
  - Most geometric results are independent of the coordinate system
  - Example Euclidean geometry: two triangles are identical if two corresponding sides and the angle between them are identical

VECTORS

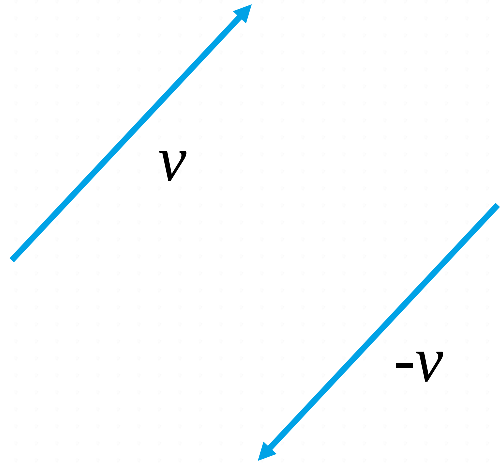
- Physical definition: a vector is a quantity with two attributes
  - Direction
  - Magnitude

- Examples include
  - Force
  - Velocity
  - Directed line segments
    - Most important example for graphics

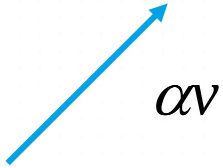


VECTOR OPERATIONS

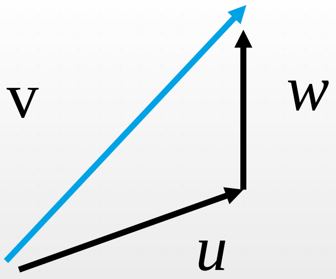
- Every vector has an inverse
  - Same magnitude but points in opposite direction



- Every vector can be multiplied by a scalar



- There is a zero vector
  - Zero magnitude, undefined orientation
- The sum of any two vectors is a vector
  - Use head-to-tail axiom



LINEAR VECTOR SPACES

- Mathematical system for manipulating vectors

- Operations
  - Scalar-vector multiplication: $u = \alpha v$
  - Vector-vector addition: $w = u + v$

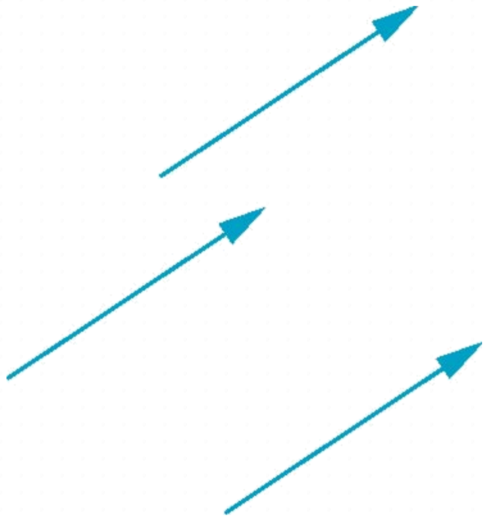
- Expressions such as

$$v = u + 2w - 3r$$

make sense in a vector space

VECTORS LACK POSITION

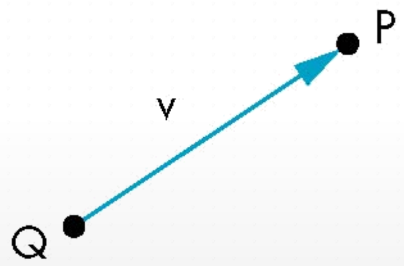
- These vectors are identical
  - Same length and magnitude



- Vectors spaces insufficient for geometry
  - Need points

POINTS

- Location in space
- Operations allowed between points and vectors
  - Point-point subtraction yields a vector $v = P - Q$
  - Equivalent to point-vector addition $P = v + Q$

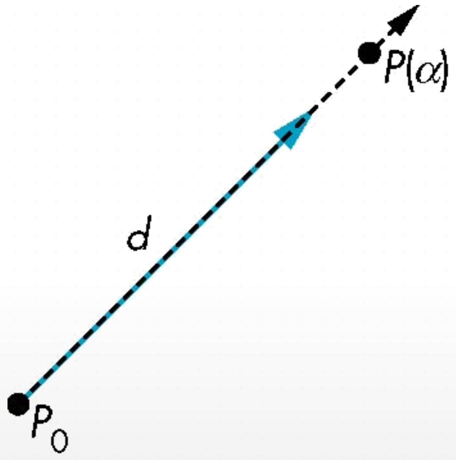


AFFINE SPACES

- Point + a vector space
- Operations
  - Vector-vector addition
  - Scalar-vector multiplication
  - Point-vector addition
  - Scalar-scalar operations
- For any point define
  - $1 \bullet P = P$
  - $0 \bullet P = \mathbf{0}$ (zero vector)

LINES

- Consider all points of the form
  - $P(\alpha) = P_0 + \alpha\,\mathbf{d}$
  - Set of all points that pass through P_0 in the direction of the vector $\mathbf{d}$



PARAMETRIC FORM

- This form is known as the parametric form of the line
  - More robust and general than other forms
  - Extends to curves and surfaces

- Two-dimensional forms
  - Explicit: $y = mx + h$
  - Implicit: $ax + by + c = 0$
  - Parametric:

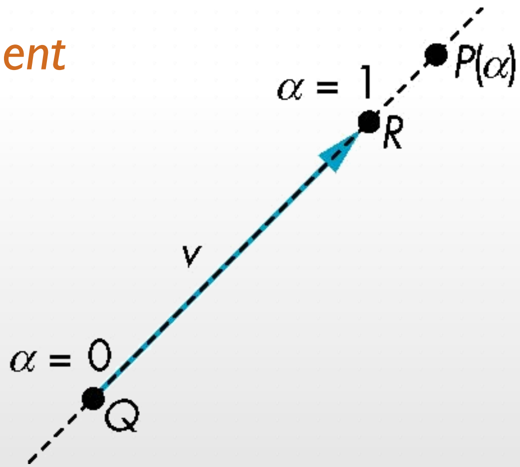
$$x(\alpha) = \alpha\, x_0 + (1 - \alpha)x_1$$
$$y(\alpha) = \alpha\, y_0 + (1 - \alpha)y_1$$

RAYS AND LINE SEGMENTS

- If $\alpha \geq 0$, then $P(\alpha)$ is the *ray* leaving P_0 in the direction $\mathbf{d}$
- If we use two points to define v, then

$$P(\alpha) = Q + \alpha(R - Q) = Q + \alpha v$$
$$= \alpha R + (1 - \alpha)Q$$

- For $0 \leq \alpha \leq 1$ we get all the points on the *line segment* joining R and Q



CURVES AND SURFACES

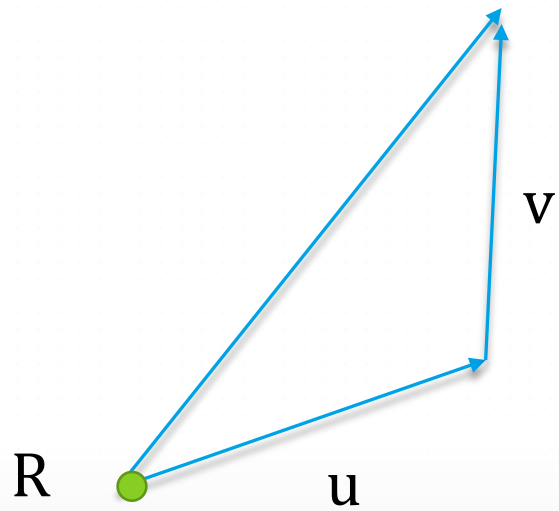
- Curves are one parameter entities of the form $P(\alpha)$ where the function is nonlinear
- Surfaces are formed from two-parameter functions $P(\alpha, \beta)$
  - Linear functions give planes and polygons

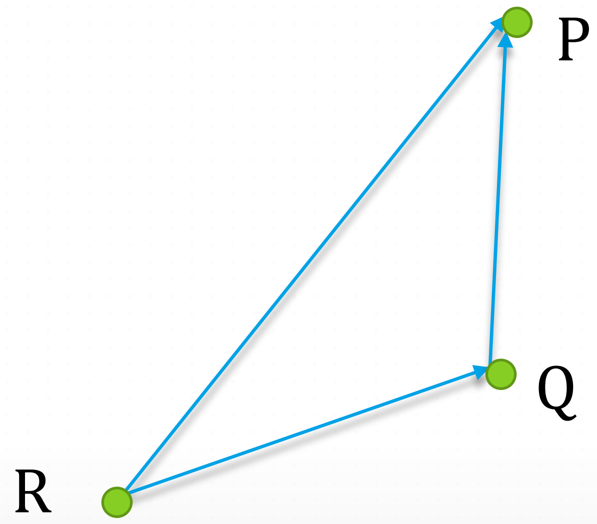
$P(\alpha)$

$P(\alpha, \beta)$

PLANES

- A plane can be defined by a point and two vectors or by three points





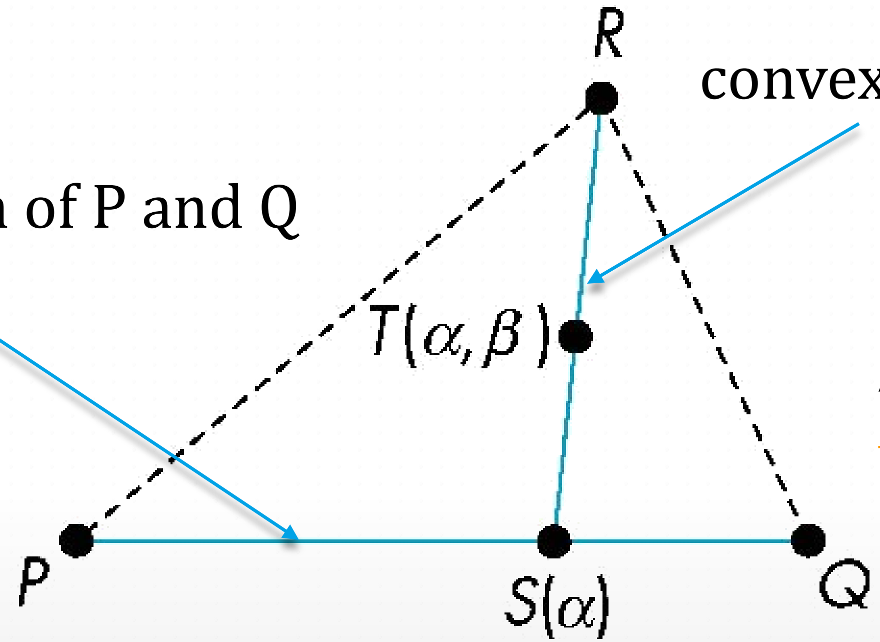
$P(\alpha, \beta) = R + \alpha\, u + \beta\, v$ $\qquad$ $P(\alpha, \beta) = R + \alpha(Q\text{-}R) + \beta(P\text{-}Q)$

TRIANGLES

$$T(\alpha, \beta) = R + \alpha(Q-R) + \beta(P-Q)$$

convex sum of $S(\alpha)$ and R

convex sum of P and Q

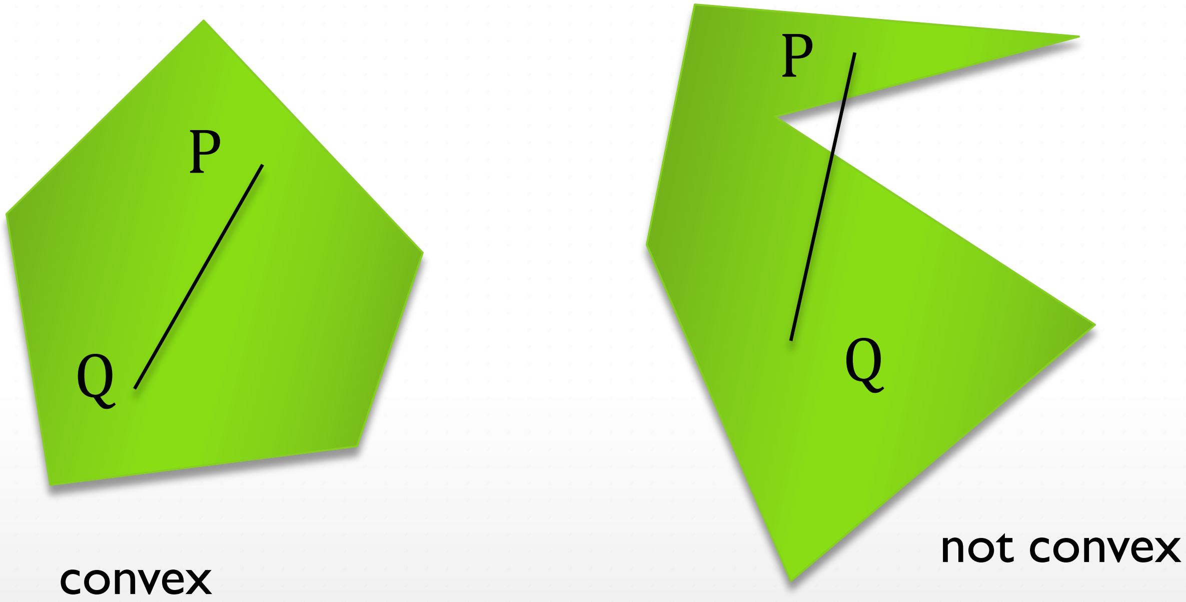


Convex sum:
All coefficients are non-negative and sum up to 1.

for $0 \leq \alpha, \beta \leq 1$, we get all points in triangle

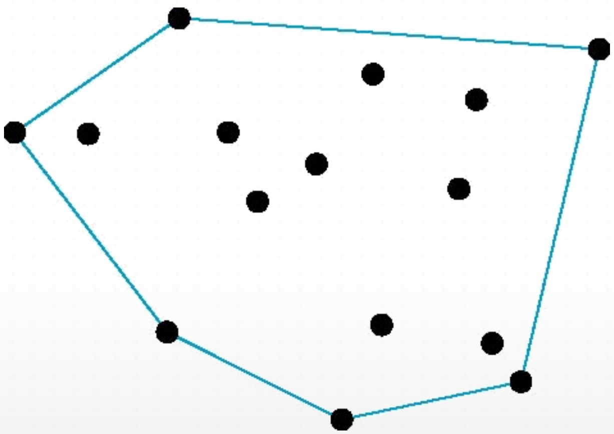
CONVEXITY

- An object is *convex* iff for any two points in the object, all points on the line segment between these points are also in the object



CONVEX HULL

- Smallest convex object containing $P_1, P_2, \ldots P_n$
- Formed by "shrink wrapping" points



AFFINE SUMS

- Consider the "sum"

$$P = \alpha_1 P_1 + \alpha_2 P_2 + \ldots + \alpha_n P_n$$

- Can show by induction that this sum makes sense iff

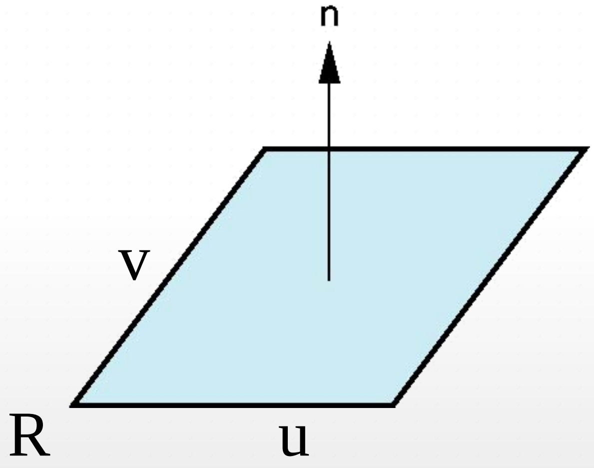
$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = 1$$

  in which case we have the *affine sum* of the points $P_1, P_2, \ldots P_n$

- If, in addition, $\alpha_i \geq 0$, we have the *convex hull* of $P_1, P_2, \ldots P_n$

NORMALS

- Every plane has a vector *n* normal (perpendicular, orthogonal) to it
- From point-two vector form $P(\alpha, \beta) = R + \alpha u + \beta v$, we know we can use the cross product to find $n = u \times v$.



REPRESENTATION

OBJECTIVES

- Introduce concepts such as dimension and basis
- Introduce coordinate systems for representing vectors spaces and frames for representing affine spaces
- Discuss change of frames and bases
- Introduce homogeneous coordinates

LINEAR INDEPENDENCE

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

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = 0 \qquad \text{iff} \qquad \alpha_1 = \alpha_2 = \ldots = 0$$

- 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, as 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

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

where the $\{\alpha_i\}$ are unique.

REPRESENTATION

- Until now we have been able to work with 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

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

EXAMPLE

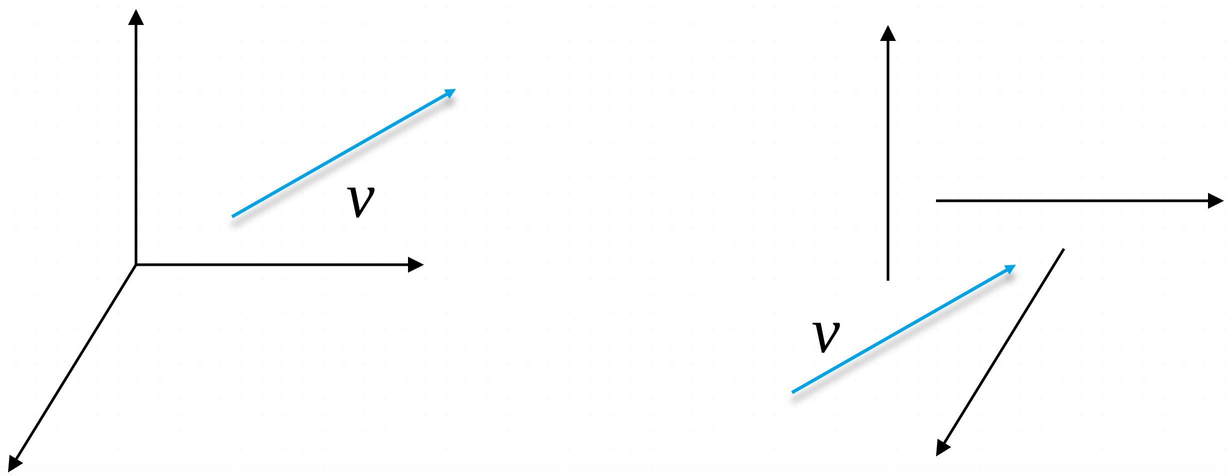
$$v = 2v_1 + 3v_2 - 4v_3$$

$$\mathbf{a} = [2\ 3\ -4]^T$$

- Note that this representation is with respect to a particular basis.
- For example, in OpenGL we 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

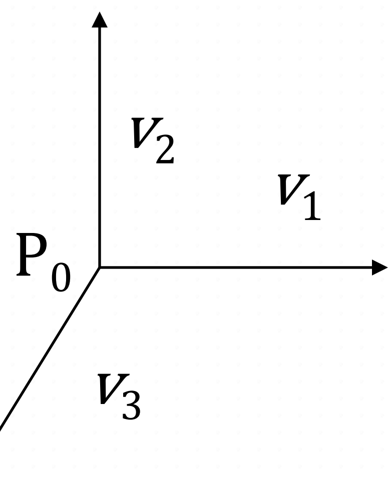
- Which is correct?



- 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

$$P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \ldots + \beta_n v_n$$
$$v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$$

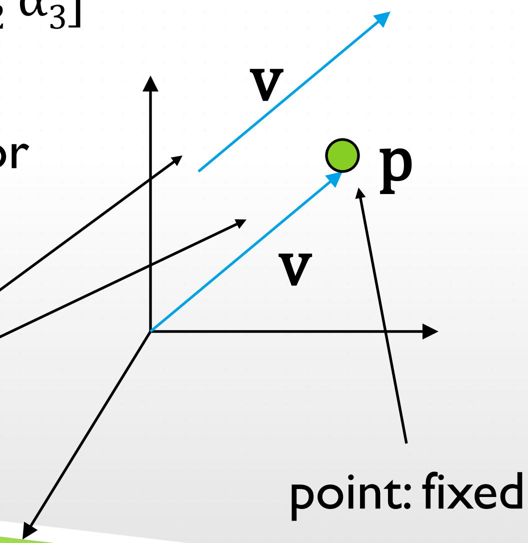
- They appear to have the similar representations

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

which confuses the point with the vector

- A vector has no position

Vector can be placed anywhere



A SINGLE REPRESENTATION

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

$$\mathrm{v} = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 = [\alpha_1\ \alpha_2\ \alpha_3\ 0]\,[v_1\ \ v_2\ \ v_3\ \ \mathrm{P}_0]^{\mathrm{T}}$$

$$\mathrm{P} = \mathrm{P}_0 + \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 = [\beta_1\ \beta_2\ \beta_3\ 1]\,[v_1\ \ v_2\ \ v_3\ \ \mathrm{P}_0]^{\mathrm{T}}$$

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

$$\mathbf{v} = [\alpha_1\ \alpha_2\ \alpha_3\ 0]^{\mathrm{T}} \qquad \mathbf{p} = [\beta_1\ \beta_2\ \beta_3\ 1]^{\mathrm{T}}$$

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*

CHANGE OF COORDINATE SYSTEMS

- Consider two representations of the same vector with respect to two different bases.

- The representations are

$$\mathbf{a} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

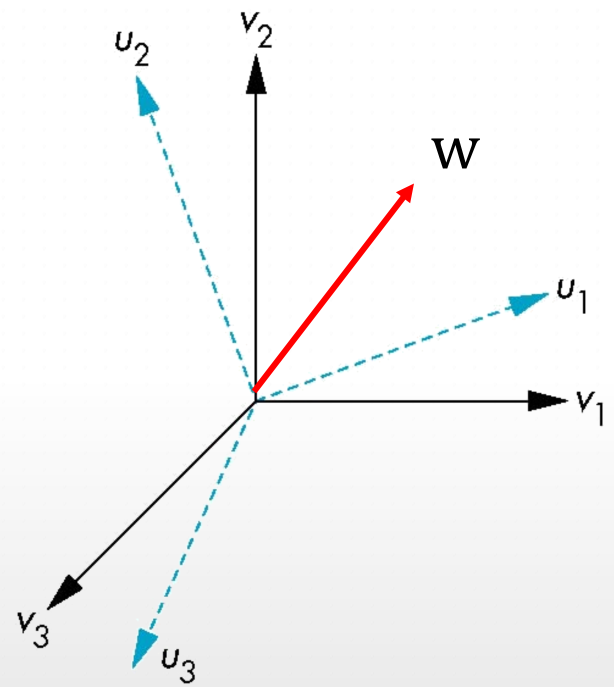
where

$$\begin{aligned} \mathbf{w} &= \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 = \mathbf{a}^{\mathrm{T}}\mathbf{v} = [\alpha_1\ \alpha_2\ \alpha_3]\,[v_1\ v_2\ v_3]^{\mathrm{T}} \\ &= \beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3 = \mathbf{b}^{\mathrm{T}}\mathbf{u} = [\beta_1\ \beta_2\ \beta_3]\,[u_1\ u_2\ u_3]^{\mathrm{T}} \end{aligned}$$

REPRESENTING SECOND BASIS IN TERMS OF FIRST

- Each of the basis vectors, $\mathrm{u}_1, \mathrm{u}_2, \mathrm{u}_3$, are vectors that can be represented in terms of the first basis

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



MATRIX FORM

- The coefficients define a 3 x 3 matrix

$$\mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix} \qquad \mathbf{u} = \mathbf{M}\mathbf{v}$$

$$\mathbf{w} = \mathbf{a}^T\mathbf{v} = \mathbf{b}^T\mathbf{u} = \mathbf{b}^T\mathbf{M}\mathbf{v} \quad => \quad \mathbf{a}^T = \mathbf{b}^T\mathbf{M}$$

and hence the bases can be related by $\mathbf{a} = \mathbf{M}^T\mathbf{b}$

CHANGE OF BASES

- Suppose that we have a vector $\boldsymbol{w}$ whose representation in some basis is

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

- We can denote the three basis vectors as v_1, v_2 and v_3. Hence,

$$\boldsymbol{w} = v_1 + 2v_2 + 3v_3$$

- Now suppose that we want to make a new basis from the three vectors v_1, v_2 and v_3 where

$$u_1 = v_1,$$
$$u_2 = v_1 + v_2$$
$$u_3 = v_1 + v_2 + v_3$$

- The matrix is

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

CHANGE OF BASES

$\mathbf{a} = \mathbf{M}^T\mathbf{b}$

$\mathbf{b} = (\mathbf{M}^T)^{-1}\mathbf{a}$

$$(\mathbf{M}^T)^{-1} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{b} = (\mathbf{M}^T)^{-1}\mathbf{a}$

$$= \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$= \begin{bmatrix} -1 \\ -1 \\ 3 \end{bmatrix}$$

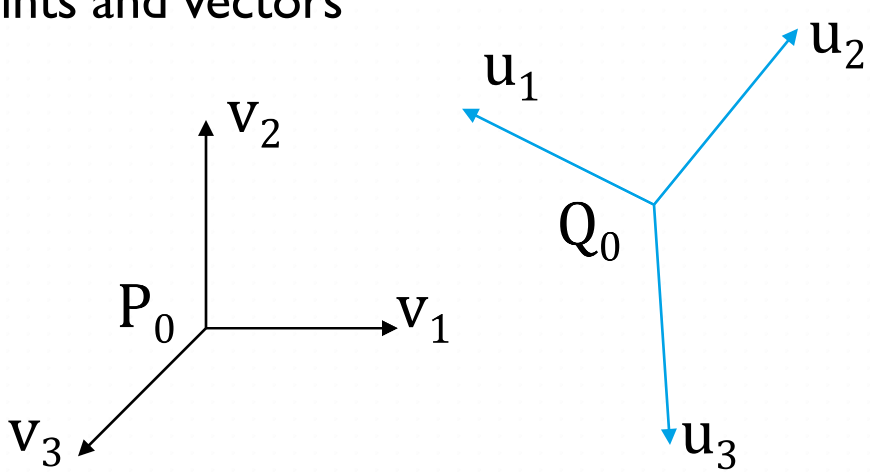
Thus,

$$\boldsymbol{w} = -u_1 - u_2 + 3u_3$$

CHANGE OF FRAMES

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

Consider two 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 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 FRAME IN TERMS OF THE OTHER

- Extending what we did with change of bases

$$
\begin{aligned}
u_1 &= \gamma_{11} v_1 + \gamma_{12} v_2 + \gamma_{13} v_3 \\
u_2 &= \gamma_{21} v_1 + \gamma_{22} v_2 + \gamma_{23} v_3 \\
u_3 &= \gamma_{31} v_1 + \gamma_{32} v_2 + \gamma_{33} v_3 \\
Q_0 &= \gamma_{41} v_1 + \gamma_{42} v_2 + \gamma_{43} v_3 \\
&\quad + P_0
\end{aligned}
$$

- Defining a 4 x 4 matrix

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

WORKING WITH REPRESENTATIONS

- Within the two frames any point or vector has a representation of the same form

  $\mathbf{a} = [\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4]^T$ in the first frame

  $\mathbf{b} = [\beta_1\ \beta_2\ \beta_3\ \beta_4]^T$ in the second frame

  where $\alpha_4 = \beta_4 = 1$ for points and $\alpha_4 = \beta_4 = 0$ for vectors

- The matrix $\mathbf{M}$ is 4 x 4 and specifies an affine transformation in homogeneous coordinates

$$\mathbf{a} = \mathbf{M}^T\mathbf{b}$$

CHANGE OF FRAMES

- Suppose that we have a vector w whose representation in some basis is

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

- We can denote the three basis vectors as v_1, v_2 and v_3. Hence,

$$w = v_1 + 2v_2 + 3v_3$$

- Now suppose that we want to make a new frame where

$$u_1 = v_1,$$
$$u_2 = v_1 + v_2$$
$$u_3 = v_1 + v_2 + v_3$$
$$Q_0 = P_0 + v_1 + 2v_2 + 3v_3$$

- The matrix is

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 1 \end{bmatrix}$$

CHANGE OF FRAMES (VECTOR)

$$\mathbf{a} = \mathbf{M}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{b} = (\mathbf{M}^{\mathrm{T}})^{-1}\mathbf{a}$$

$$(\mathbf{M}^{\mathrm{T}})^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -1 & 0 & 1 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{b} = (\mathbf{M}^{\mathrm{T}})^{-1}\mathbf{a}$$

$$= \begin{bmatrix} 1 & -1 & 0 & 1 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 3 \\ 0 \end{bmatrix}$$

Thus, for vector we still have

$$\boldsymbol{w} = -u_1 - u_2 + 3u_3$$

CHANGE OF FRAMES (POINT)

A point on the old frame

$$\mathbf{p} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

$$\mathbf{p'} = (\mathbf{M}^T)^{-1}\mathbf{p}$$

$$= \begin{bmatrix} 1 & -1 & 0 & 1 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Thus, point's representation has been changed.

AFFINE TRANSFORMATIONS

- Every linear transformation is equivalent to a change in frames
- Every affine transformation preserves lines
- However, an affine transformation has only 12 *degrees of freedom* because 4 of the elements in the matrix are fixed and are a subset of all possible 4 x 4 linear transformations

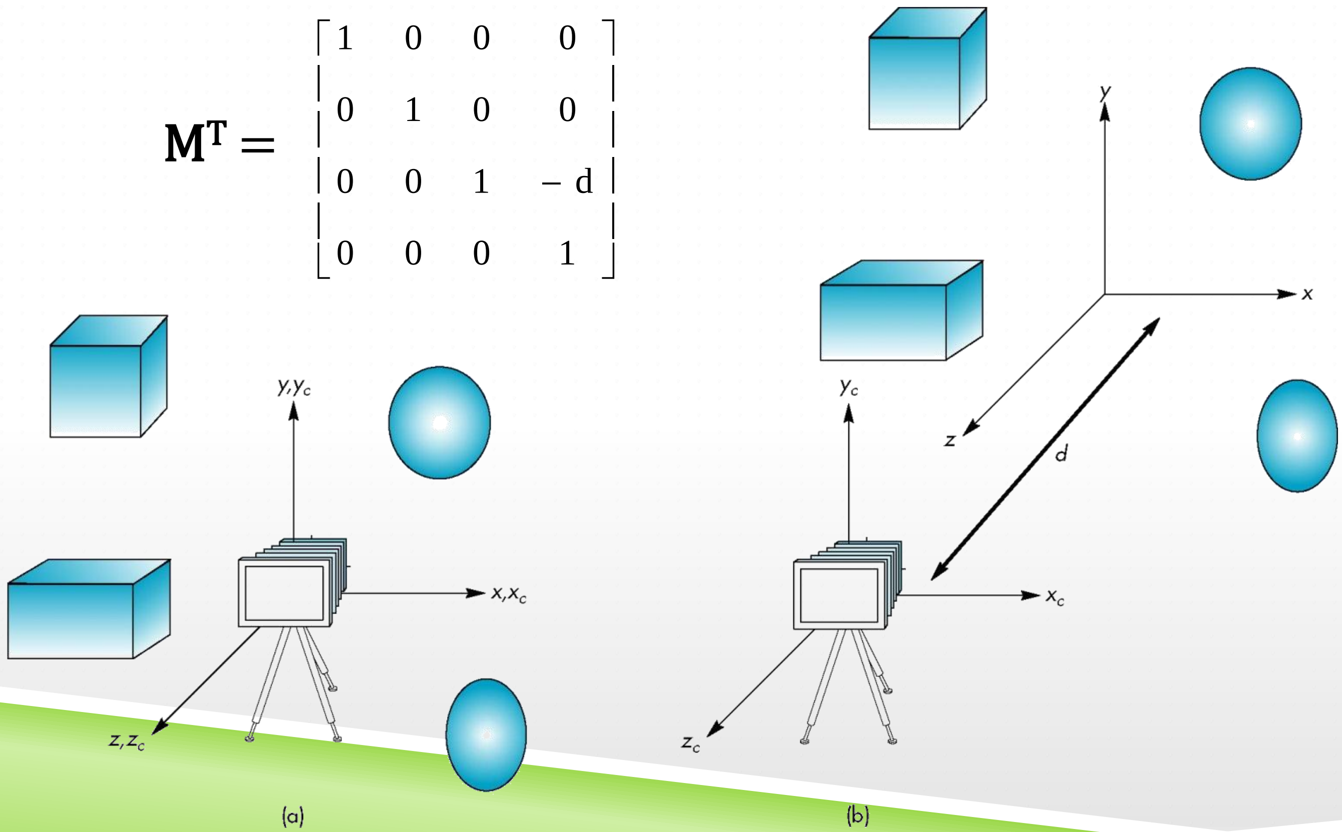
THE WORLD AND CAMERA FRAMES

- When we work with representations, we work with n-tuples or arrays of scalars
- Changes in frame are then defined by 4 x 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 ($\mathbf{M} = \mathbf{I}$)

MOVING THE CAMERA

- If objects are on both sides of $z = 0$, we must move camera frame

$$\mathbf{M}^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



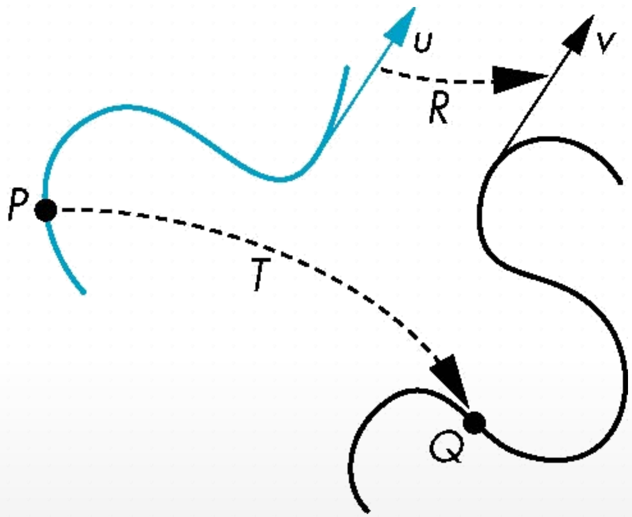
TRANSFORMATIONS

OBJECTIVES

- Introduce standard transformations
  - Rotation
  - Translation
  - Scaling
  - Shear
- Derive homogeneous coordinate transformation matrices
- Learn to build arbitrary transformation matrices from simple transformations

GENERAL TRANSFORMATIONS

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



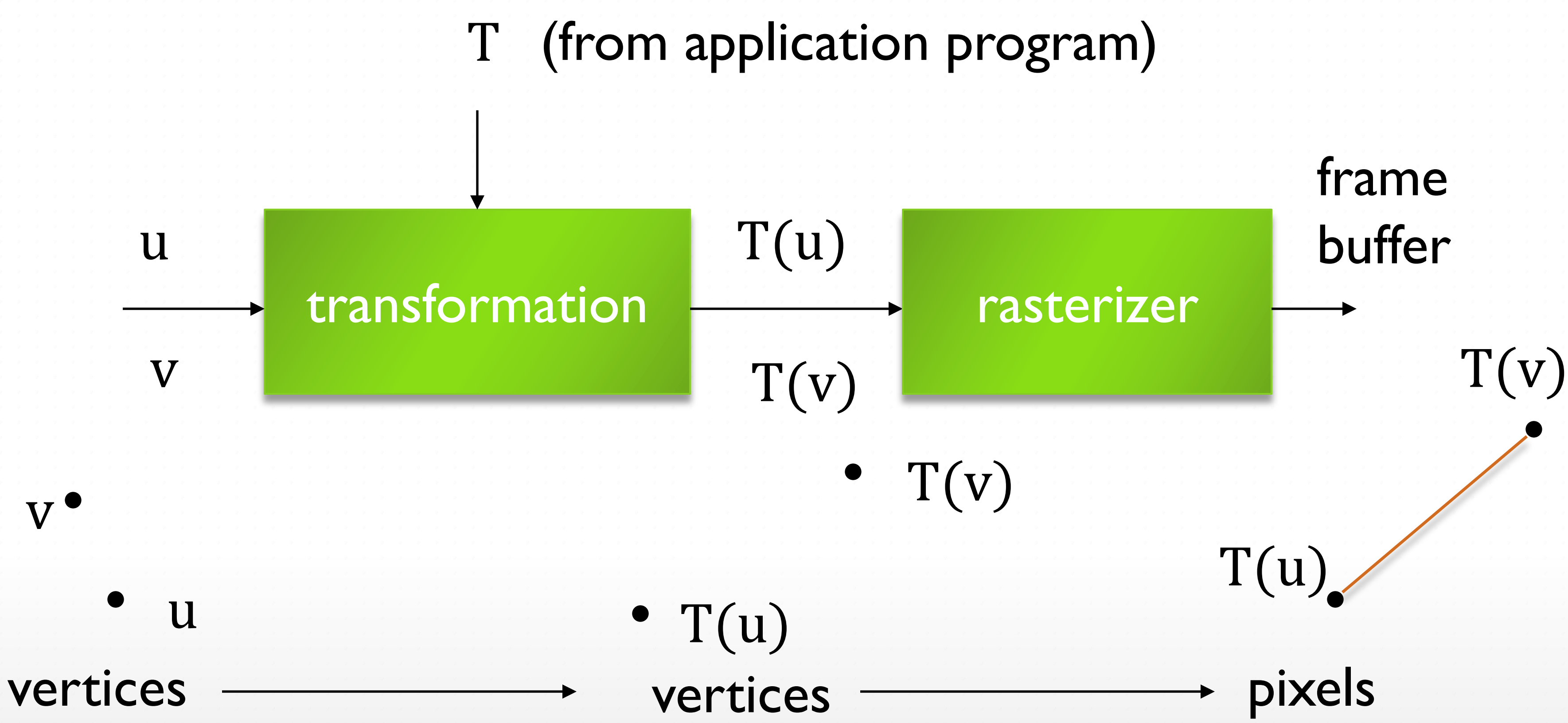
$$\mathbf{v} = \mathrm{T}(\mathbf{u})$$

$$\mathrm{Q} = \mathrm{T}(\mathrm{P})$$

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

PIPELINE IMPLEMENTATION



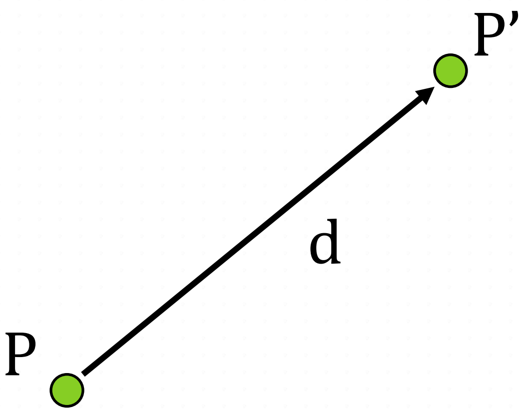
NOTATION

- 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
u, v, w:	vectors in an affine space
α, β, γ:	scalars
p, **q**, **r**:	representations of points-array of 4 scalars in homogeneous coordinates
u, **v**, **w**:	representations of points-array of 4 scalars in homogeneous coordinates

TRANSLATION

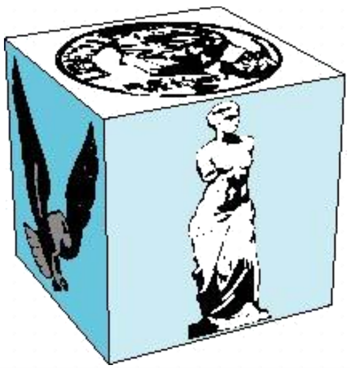
- Move (translate, displace) a point to a new location



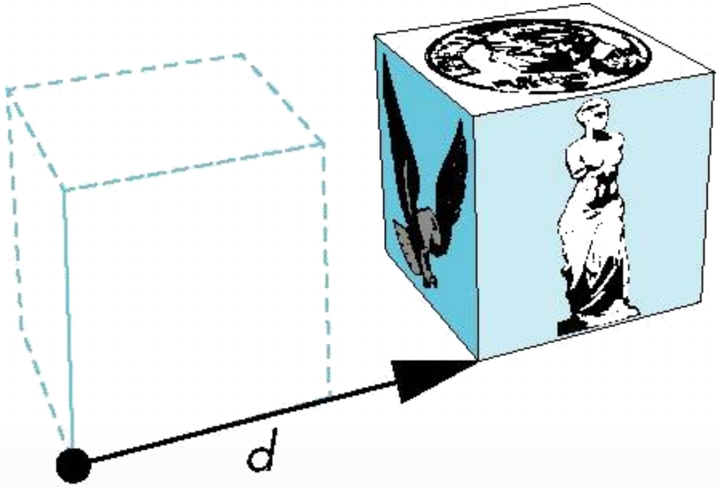
- Displacement determined by a vector d
  - Three degrees of freedom
  - $P' = P + d$

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

$$\mathbf{p} = [x\ y\ z\ 1]^T$$
$$\mathbf{p}' = [x'\ y'\ z'\ 1]^T$$
$$\mathbf{d} = [d_x\ d_y\ d_z\ 0]^T$$

- 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 x 4 matrix $\mathbf{T}$ ($=\mathbf{M}^T$) in homogeneous coordinates $\mathbf{p}' = \mathbf{T}\mathbf{p}$ where

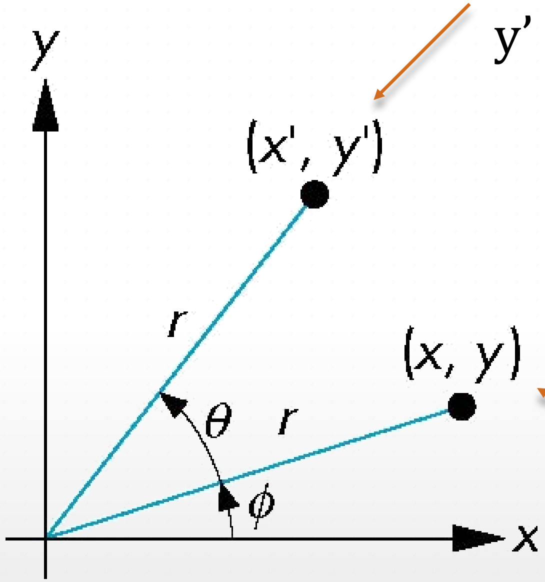
$$\mathbf{T} = \mathbf{T}(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 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

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



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

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

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

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

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

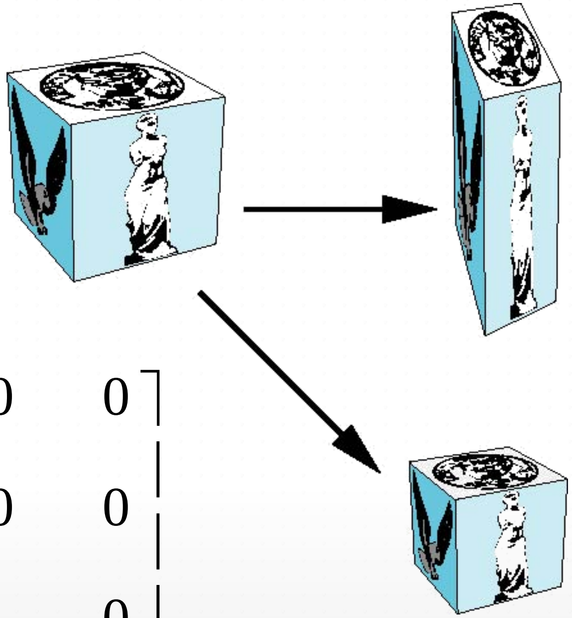
$$\mathbf{R} = \mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

SCALING

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

$$\mathbf{S} = \mathbf{S}(s_x, s_y, s_z) = \quad \begin{aligned} x' &= s_x x \\ y' &= s_y x \\ z' &= s_z x \end{aligned}$$



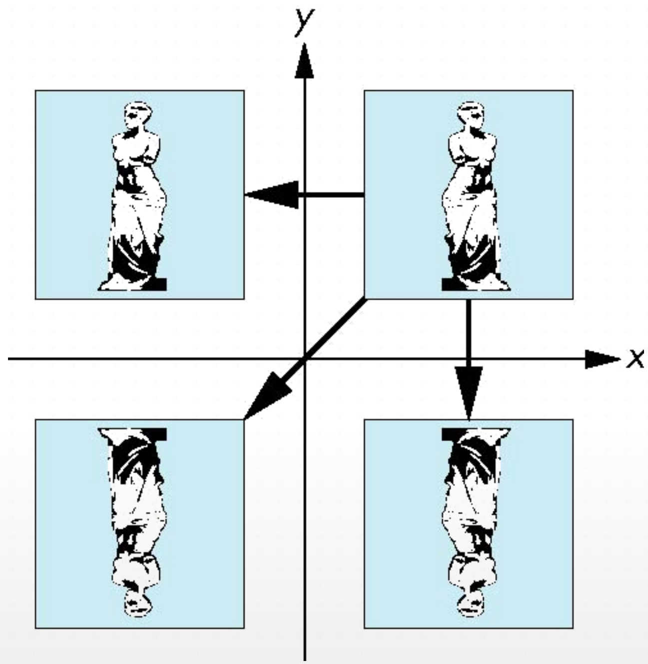
$$\mathbf{p}' = \mathbf{S}\mathbf{p} \qquad \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

ORDER OF TRANSFORMATIONS

- Note that matrix on the right is the first applied
- Mathematically, the following are equivalent

$$p' = \mathbf{ABCp} = \mathbf{A(B(Cp))}$$

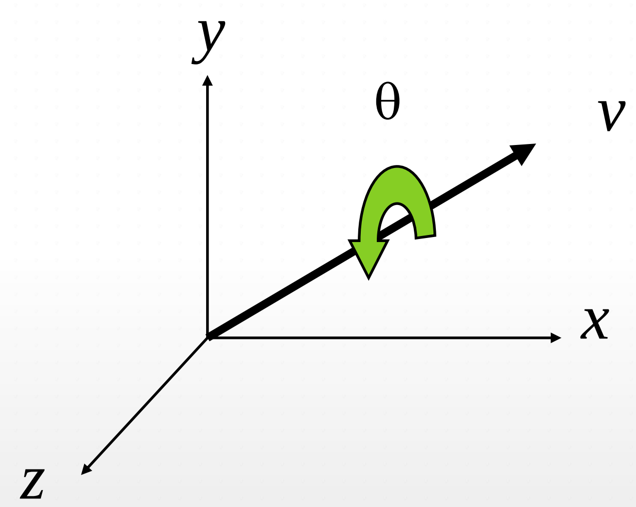
GENERAL ROTATION ABOUT THE ORIGIN

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

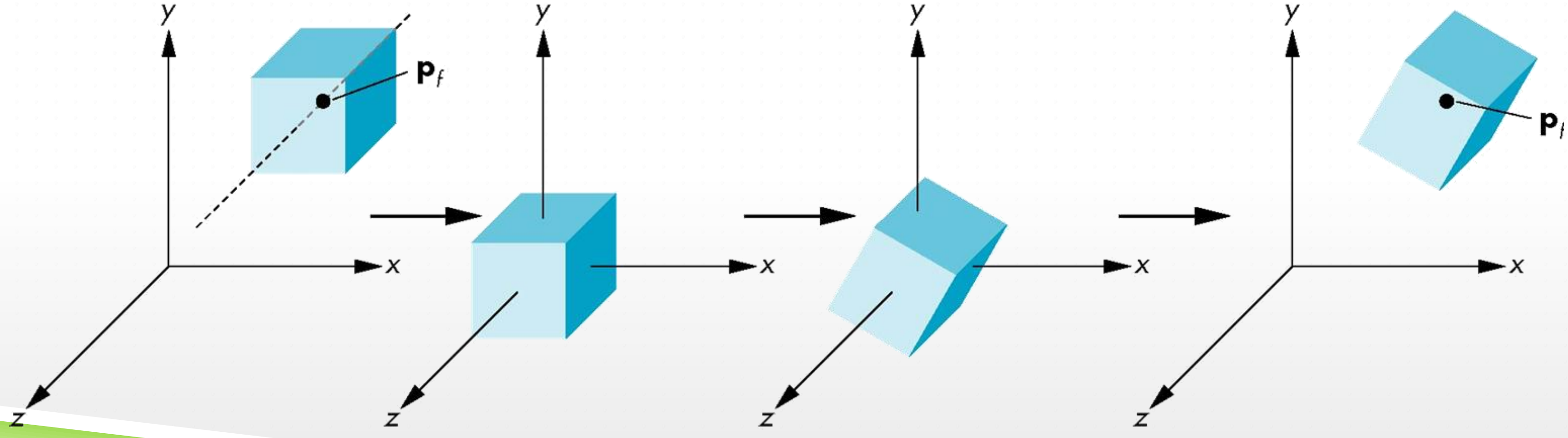
Note that rotations do not commute
We can use rotations in another order but with different angles



ROTATION ABOUT A FIXED POINT OTHER THAN THE ORIGIN

- Move fixed point to origin
- Rotate
- Move fixed point back

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



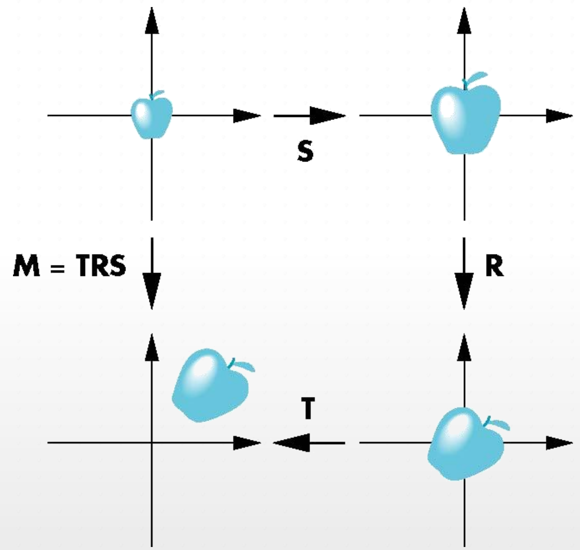
INSTANCING

- 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

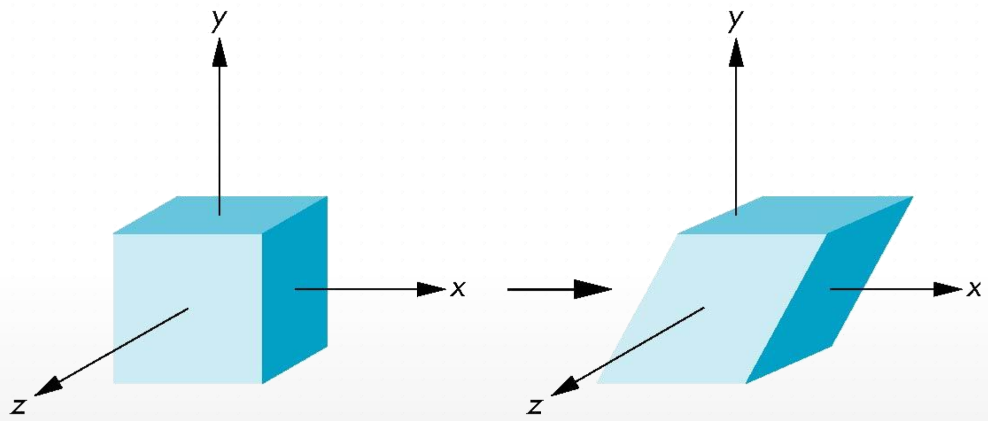
Orient

Locate



SHEAR

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



SHEAR MATRIX

Consider simple shear along x axis

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

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