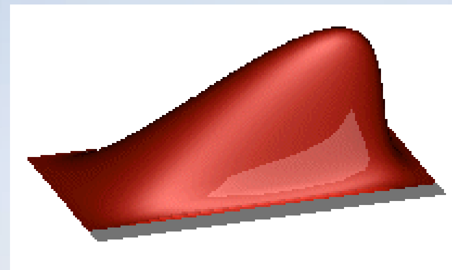
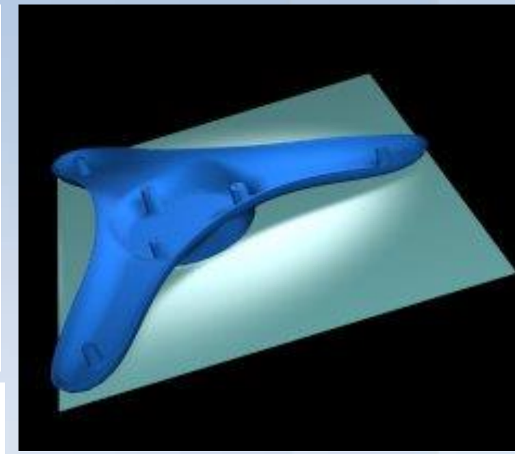


Lectures 1234567
Computer Aided Design

Hikmet Kocabaş, Prof., PhD.
Istanbul Technical University

Lectures 1234567

- Lecture 1 Intro. to Computer Graphic Systems
- Lecture 2 Geometry
- Lecture 3 Vector Algebra
- Lecture 4 Transformations
- Lecture 5 Curves
- Lecture 6 Surface Modeling
- Lecture 7 Solid Modeling

Vector Algebra and Transformations

- Source book:
- Geometric Modeling : A First Course 1995-1999 by Aristides A. G. Requicha

- Computer Aided Geometric Design, Thomas W. Sederberg, 2003.

- Motions and Projections
- Points and Vectors

Computer Aided Geometric Design

CAGD is a young field. The first work in this field began in the mid 1960s. The term computer aided geometric design was coined in 1974 by R.E. Barnhill and R.F. Riesenfeld in connection with a conference at the University of Utah.

Points, Vectors and Coordinate Systems

Consider the simple problem of writing a computer program which finds the area of any triangle. We must first decide how to uniquely describe the triangle. One way might be to provide the lengths l_1, l_2, l_3 of the three sides, from which Heron's formula yields

$$Area = \sqrt{s(s - l_1)(s - l_2)(s - l_3)}$$

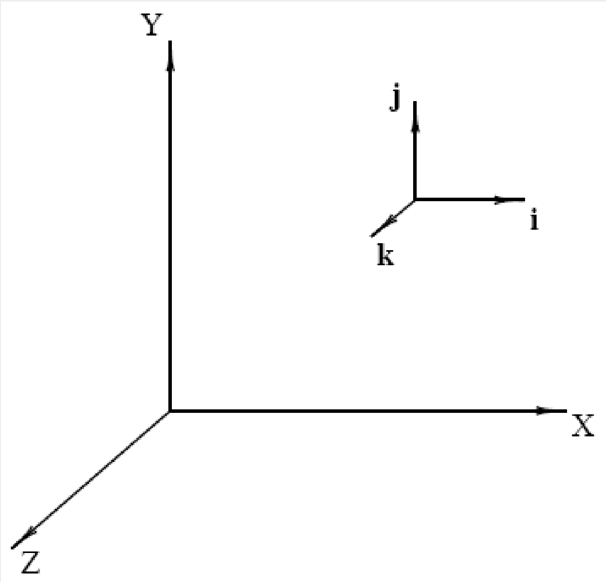
$$s = \frac{l_1 + l_2 + l_3}{2}$$

Points, Vectors and Coordinate Systems

An alternate way to describe the triangle is in terms of its vertices. But while the lengths of the sides of a triangle are independent of its position, we can specify the vertices to our computer program only with reference to some coordinate system — which can be defined simply as any method for representing points with numbers.

Points, Vectors and Coordinate Systems

Then, each vertex of our triangle could be described in terms of its respective distance from the two walls containing the origin and from the floor. These distances are the Cartesian coordinates (x, y, z) of the vertex with respect to the coordinate system we defined.



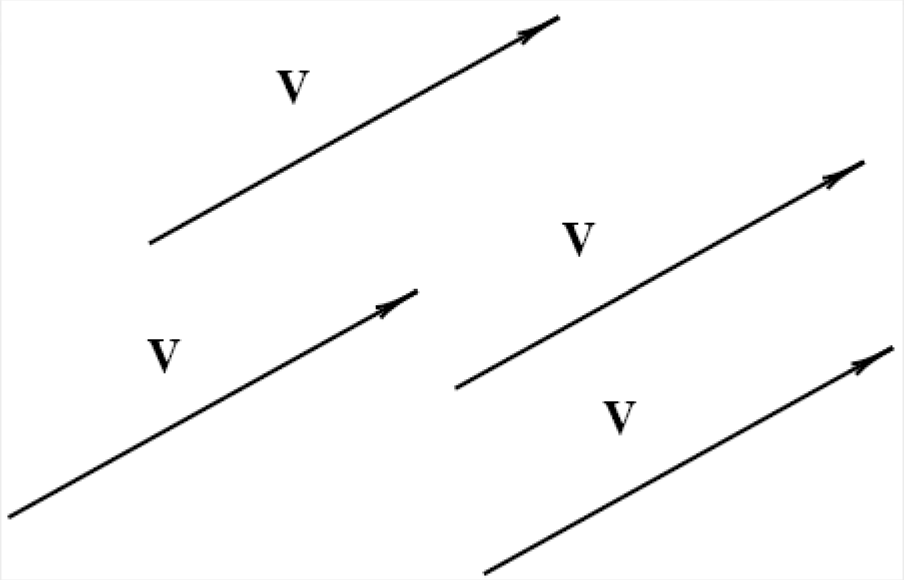
Unit Vectors

A *unit vector is a vector whose length equals unity.*

Vectors

A vector can be pictured as a line segment of definite length with an arrow on one end.

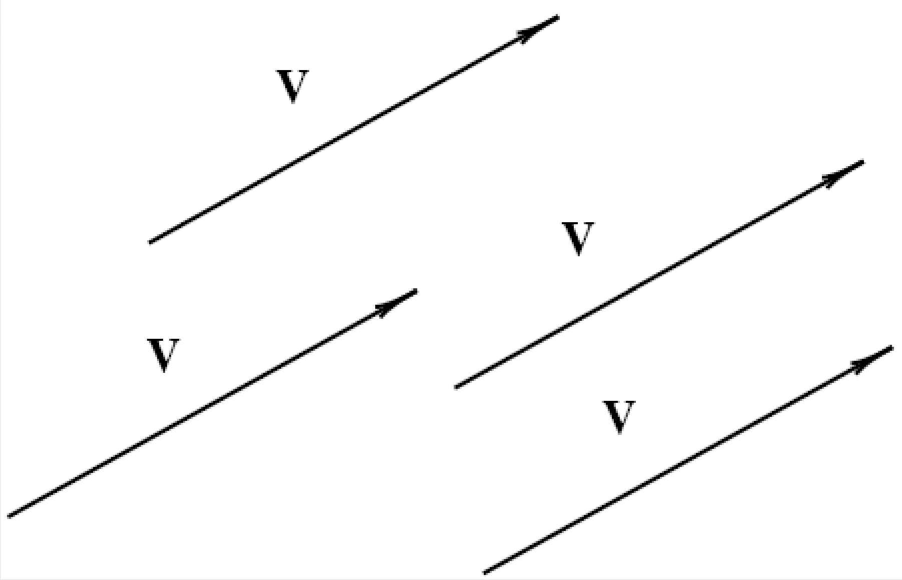
We will call the end with the arrow the tip or head and the other end the tail.



Vectors

Two vectors are equivalent if they have the same length, are parallel, and point in the same direction (have the same sense) as shown in Figure.

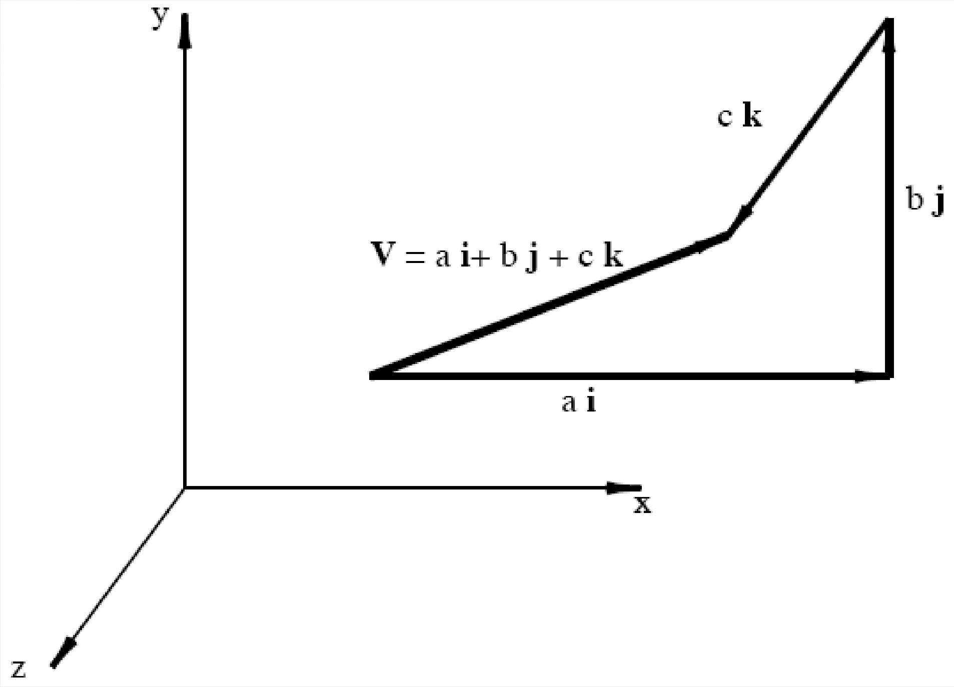
Equivalent Vectors



Vectors

For a given coordinate system, we can describe a three-dimensional vector in the form (a, b, c) where a (or b or c) is the distance in the x (or y or z) direction from the tail to the tip of the vector.

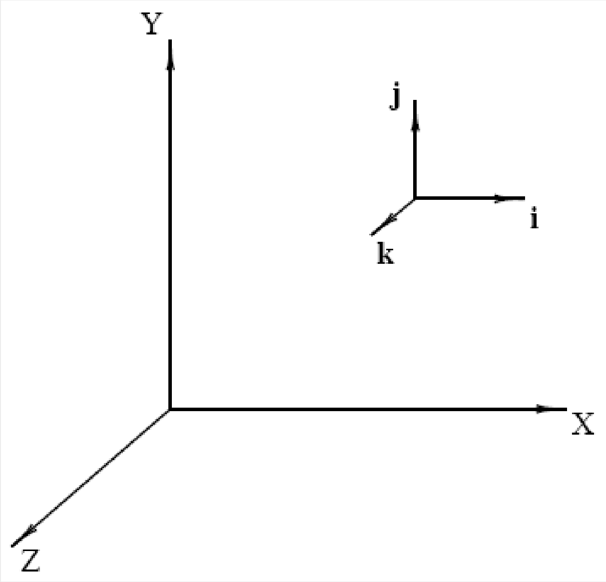
Vector in Component Form



Unit vectors

The symbols i, j, and k denote vectors of “unit length” (based on the unit of measurement of the coordinate system) which point in the positive x, y, and z directions respectively (see Figure). Unit vectors allow us to express a vector in component form

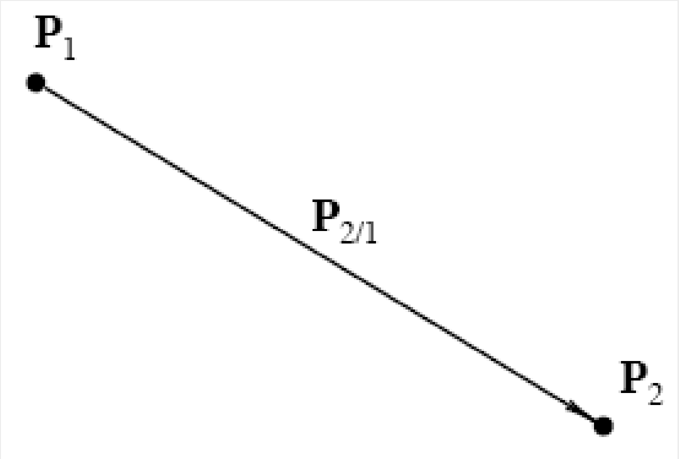
P = (a, b, c) = ai + bj + ck



Unit Vectors

Points and Vectors

An expression such as (x, y, z) can be called a triple of numbers. A triple can signify either a point or a vector. Relative Position Vectors Given two points P_1 and P_2, we can define $P_{2/1} = P_2 - P_1$
as the vector pointing from P_1 to P_2. This notation $P_{2/1}$ is widely used in engineering mechanics,
and can be read “the position of point P_2 relative to P_1” (see Figure).



The distance between two points

In a Euclidean space we define the distance between two points p and q as the norm of the vector p – q.

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

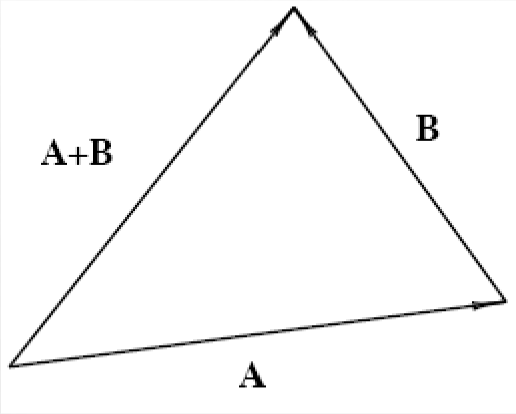
Because points correspond to vectors, for a fixed origin, and vectors correspond to column matrices, for a fixed basis, there is also a one-to-one orrespondence between points and column matrices. A pair (origin, basis) is called a *frame or coordinate system. For a fixed* frame, points correspond to column matrices.

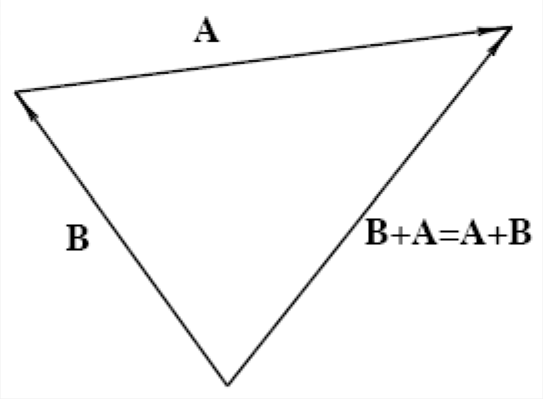
Vector Algebra

Given two vectors P1 = (x1, y1, z1) and P2 = (x2, y2, z2), the following operations are defined:

Addition:

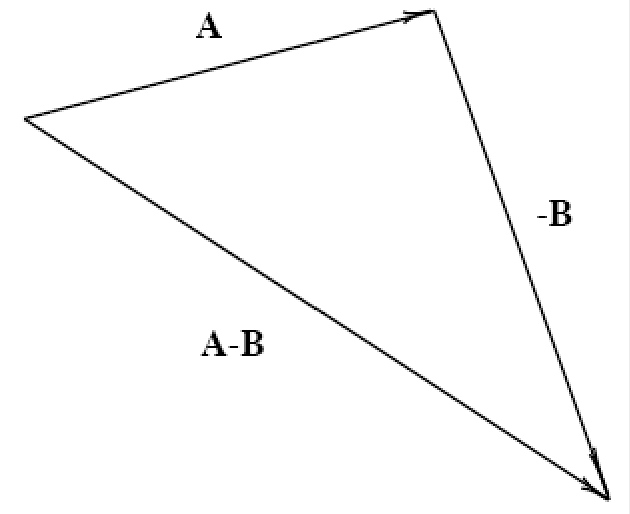
P1 + P2 = P2 + P1 = (x1 + x2, y1 + y2, z1 + z2)





Subtraction:

P1 – P2 = (x1 – x2, y1 – y2, z1 – z2)



Vector Algebra

Using matrix notation a Vector can be written as

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad E = \begin{bmatrix} \boldsymbol{e_1} & \boldsymbol{e}_2 & \dots & \boldsymbol{e}_n \end{bmatrix}.$$

$x = x_1 e_1 + x_2 e_2 + \ldots + x_n e_n$

$x = EX.$

The correspondence between vectors and matrices preserves addition and multiplication by a scalar. The matrix *Z that* corresponds to the sum of two vectors *z = x + y is the sum*

$$Z = X + Y = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix}.$$

Vector Algebra

For multiplication by a scalar,
if *z = a x, then Z = a X, or*

$$Z = \begin{bmatrix} ax_1 \\ ax_2 \\ \vdots \\ ax_n \end{bmatrix}.$$

The *inner or dot product, denoted x . y, is another useful operation defined on vectors. It* produces a scalar given two vector arguments. It is defined formally by a set of axioms. The square root of the inner product of a vector with itself is the *norm or length of the* vector, denoted $\|x\| = \sqrt{x.x}\,.$

Vector Algebra

Scalar multiplication: $cP1 = (cx_1, cy_1, cz_1)$

Length of a Vector: $|\mathbf{P}_1| = \sqrt{x_1^2 + y_1^2 + z_1^2}$

The length of x in an orthonormal basis becomes

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

Dot Product: The dot product of two vectors is defined

$P1 \cdot P2 = |P1||P2| \cos\theta$

where θ is the angle between the two vectors.

Vector Algebra

Two vectors are *orthogonal if their dot* product is zero. The cosine of the angle between two vectors is given by

$$\cos\theta = \frac{x.y}{\|x\|.\|y\|}$$

The most convenient bases are the *orthonormal bases, composed of unit vectors that are* pairwise orthogonal. In an orthonormal basis the inner product of two vectors is

$\boldsymbol{x.y = X^T Y = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n}$ **,**

where the superscript denotes matrix transposition, obtained by interchanging rows with columns.

Vector Algebra

Since the unit vectors i, j, k are mutually perpendicular,

$i \cdot i = j \cdot j = k \cdot k = 1$

$i \cdot j = i \cdot k = j \cdot k = 0.$

Since the dot product obeys the distributive law

$P1 \cdot (P2 + P3) = P1 \cdot P2 + P1 \cdot P3,$

we can easily derive the very useful equation

$$P1 \cdot P2 = (x_1 i + y_1 j + z_1 k) \cdot (x_2 i + y_2 j + z_2 k)$$

$$= (x_1 * x_2 + y_1 * y_2 + z_1 * z_2)$$

Vector Algebra

The dot product allows us to easily compute the angle between any two vectors. From the dot product equation

$$\theta = \cos^{-1}\left(\frac{\mathbf{P}_1 \cdot \mathbf{P}_2}{|\mathbf{P}_1||\mathbf{P}_2|}\right)$$

Example. Find the angle between vectors (1, 2, 4) and (3,−4, 2).

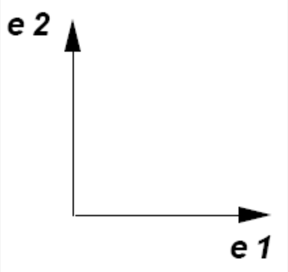
$$\begin{aligned}\theta &= \cos^{-1}\left(\frac{\mathbf{P}_1 \cdot \mathbf{P}_2}{|\mathbf{P}_1||\mathbf{P}_2|}\right)\\ &= \cos^{-1}\left(\frac{(1,2,4)\cdot(3,-4,2)}{|(1,2,4)||(3,-4,2)|}\right)\\ &= \cos^{-1}\left(\frac{3}{\sqrt{21}\sqrt{29}}\right)\\ &\approx 83.02^\circ\end{aligned}$$

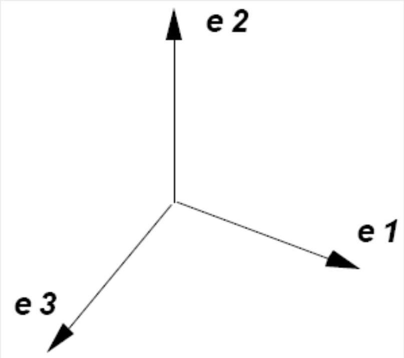
Vector product

Finally, there is an additional operation on vectors, called the *vector product (also known* as *cross, or exterior product), that is very useful, especially in 3-D. Here we define it in* terms of components in a right-handed, orthonormal, 3-D basis:

$x \times y = (x_2y_3 - x_3y_2)e_1 + (x_3y_1 - x_1y_3)e_2 + (x_1y_2 - x_2y_1)e_3$

The result of a cross product is not truly a vector, and its definition depends on the orientation or handedness of a basis.





Vector product

The cross product of two parallel vectors is zero. For two non-parallel vectors, *x and y* , the cross-product $x \times y$ *is perpendicular to both x and y . In particular, if E is a righthanded* orthonormal basis in 3-D, then

$e_1 \times e_2 = e_3$

$e_2 \times e_3 = e_1$

$e_3 \times e_1 = e_2$

Vector Algebra

Cross Product: The cross product $P_1 \times P_2$ is a vector whose magnitude is

$|P_1 \times P_2| = |P_1||P_2| \sin\theta$

(where again θ is the angle between P_1 and P_2), and whose direction is mutually perpendicular to P_1 and P_2 with a sense defined by the right hand rule as follows. Point your fingers in the direction of P_1 and orient your hand such that when you close your fist your fingers pass through the direction of P_2. Then your right thumb points in the sense of $P_1 \times P_2$.

Cross Product

From this basic definition, one can verify that

$P_1 \times P_2 = -P_2 \times P_1$,

$i \times j = k$, $j \times k = i$, $k \times i = j$

$j \times i = -k$, $k \times j = -i$, $i \times k = -j$.

Since the cross product obeys the distributive law

$P_1 \times (P_2 + P_3) = P_1 \times P_2 + P_1 \times P_3$,

we can derive the important relation

$$\begin{aligned}
\mathbf{P}_1 \times \mathbf{P}_2 &= (x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k}) \times (x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k}) \\
&= (y_1 z_2 - y_2 z_1, x_2 z_1 - x_1 z_2, x_1 y_2 - x_2 y_1) \\
&= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix}
\end{aligned}$$

Cross Product, Area of a Triangle

Cross products have many important uses. For example, finding a vector which is mutually perpendicular to two other vectors. Also, the cross product provides a straightforward method for finding the area of a triangle which is defined by three points P_1, P_2, P_3 in space.

$$Area = \frac{1}{2}|\mathbf{P}_{1/2}||\mathbf{P}_{1/3}|\sin\theta_1 = \frac{1}{2}|\mathbf{P}_{1/2} \times \mathbf{P}_{1/3}|$$

Cross Product, Area of a Triangle

For example, the area of a triangle with vertices P1 = (1, 1, 1), P2 = (2, 4, 5), P3 = (3, 2, 6) is

$$
\begin{aligned}
Area &= \frac{1}{2}|\mathbf{P}_{1/2} \times \mathbf{P}_{1/3}| \\
&= \frac{1}{2}|(1,3,4) \times (2,1,5)| \\
&= \frac{1}{2}|(11,3,-5)| = \frac{1}{2}\sqrt{11^2 + 3^2 + (-5)^2} \\
&\approx 6.225
\end{aligned}
$$

Points vs. Vectors

A point is a geometric entity which connotes position, whereas a vector connotes direction and magnitude. From a purely mathematical viewpoint, there are good arguments for carefully distinguishing between triples that refer to points and triples that signify vectors. However, no problem arises if we recognize that a triple connoting a point can be interpreted as a vector from the origin to the point. Thus, we could call a point an absolute position vector and the difference between two points a relative position vector.

Homogeneous Coordinates

The homogeneous Cartesian coordinates (X, Y,W) of a point are defined x=X/W; y=Y/W.
Homogeneous coordinates are useful, among other things, for expressing points at infinity: The point (X,Y,0) is an infinite distance from the origin (or from any finite point, for that matter) in the direction X_i+Y_j.
Obviously, the homogeneous coordinates of a point are only unique to within a scale factor. For example, the point (x, y) = (2, 3) has homogeneous coordinates (X, Y,W)=(2, 3, 1), or (4, 6, 2), or in general, (2W,3W,W).
The point (X, Y,W) = (0, 0, 0) is undefined.

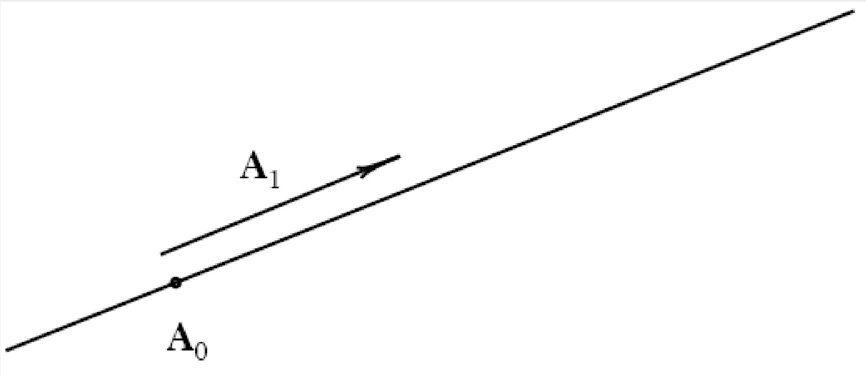
Lines

A line can be defined using either a parametric equation or an implicit equation.

Parametric equations of lines

Linear parametric equation. A line can be written in parametric form as follows:

$x = a_0 + a_1 t$; $y = b_0 + b_1 t$



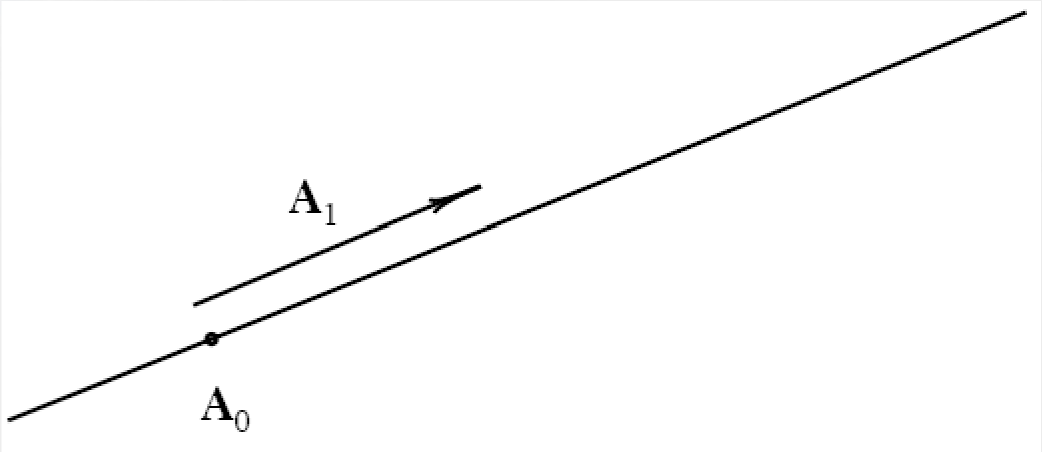
In vector form,

$$\mathbf{P}(t) = \left\{ \begin{array}{c} x(t) \\ y(t) \end{array} \right\} = \left\{ \begin{array}{c} a_0 + a_1 t \\ b_0 + b_1 t \end{array} \right\} = \mathbf{A}_0 + \mathbf{A}_1 t.$$

Lines

In this equation, A_0 is a point on the line and A_1 is the direction of the line (see Figure)

Line given by $A_0 + A_1t$.



Affine parametric equation of a line. A straight line can also be expressed

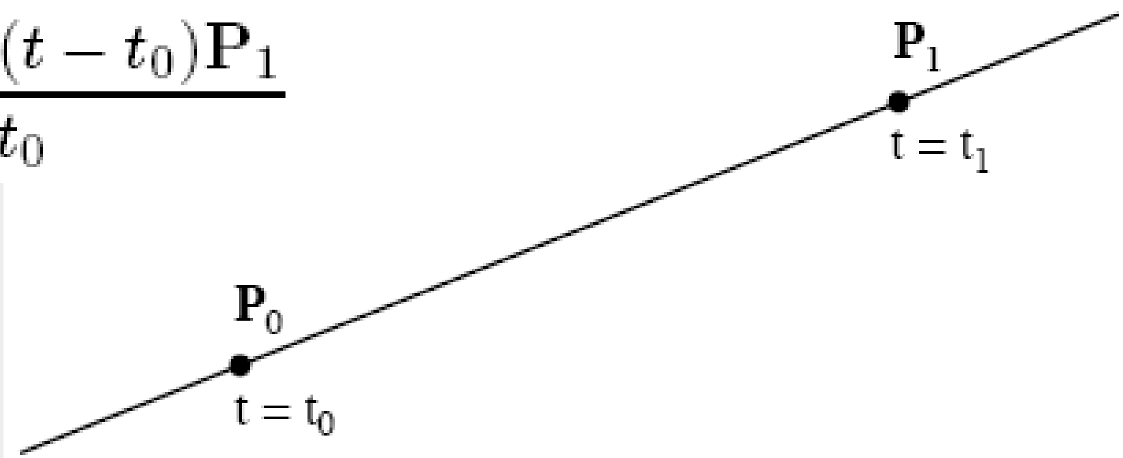
$$\mathbf{P}(t) = \frac{(t_1 - t)\mathbf{P}_0 + (t - t_0)\mathbf{P}_1}{t_1 - t_0}$$

Lines

where P_0 and P_1 are two points on the line and to and t1 are any parameter values. Note that $P(t_0) = P_0$ and $P(t_1) = P_1$.
Note in Figure that the line segment P_0–P_1 is defined by restricting the parameter: $t_0 \leq t \leq t_1$.

$$\mathbf{P}(t) = \frac{(t_1 - t)\mathbf{P}_0 + (t - t_0)\mathbf{P}_1}{t_1 - t_0}$$

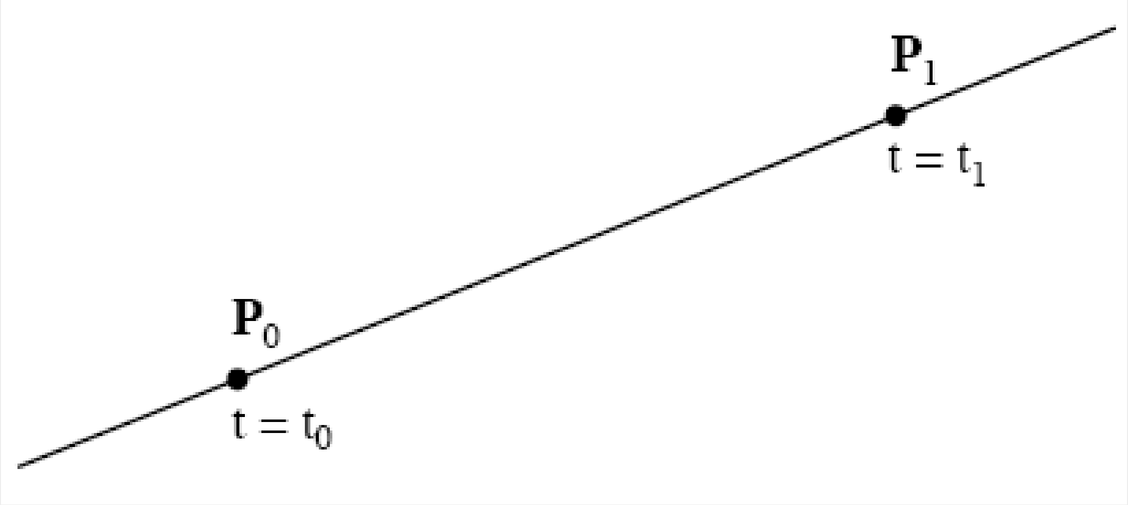
Line given by P(t)



Lines

$$\mathbf{P}(t) = \frac{(t_1 - t)\mathbf{P}_0 + (t - t_0)\mathbf{P}_1}{t_1 - t_0}$$

Sometimes this is expressed by saying that the line segment is the portion of the line in the parameter interval or domain $[t_0, t_1]$. We will soon see that the line in Figure is actually a degree one Bezier curve. Most commonly, we have $t_0 = 0$ and $t_1 = 1$ in which case $P(t) = (1 - t)P_0 + tP_1$.



Transformations

Moving, sizing, and deforming objects are fundamental operations in geometric modeling. Since objects are sets of points, what we need are *transformations that map points onto* other points.

The following subsections discuss linear and affine transformations, which are the simplest and most commonly used in geometric modeling. For simplicity we assume a fixed origin, and make no distinction between points and vectors.

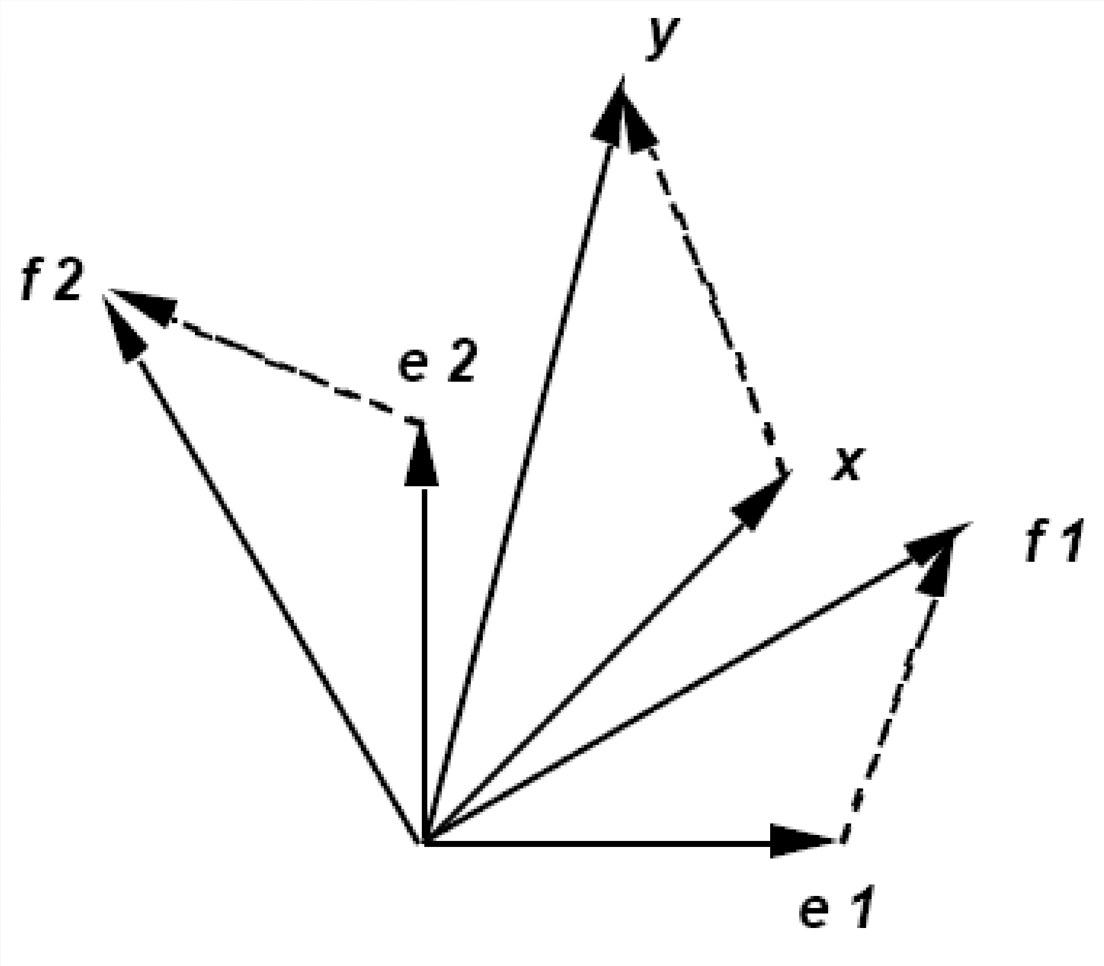
Linear Transformations

A transformation *T from a vector space onto itself is linear if it distributes over linear*
combinations, i.e.,
$T(ax + by) = aT(x) + bT(y).$
Suppose we have two bases
$E = [e_1 \dots e_n]$
$F = [f_1 \dots f_n]$
and we want a linear transformation that maps the vectors of *E onto the vectors of F (see* Figure):
$Tef(e_i) = f_i,\ i = 1,\dots,n.$

Linear Transformations

Transforming
a basis *E*
and a vector ***x.***

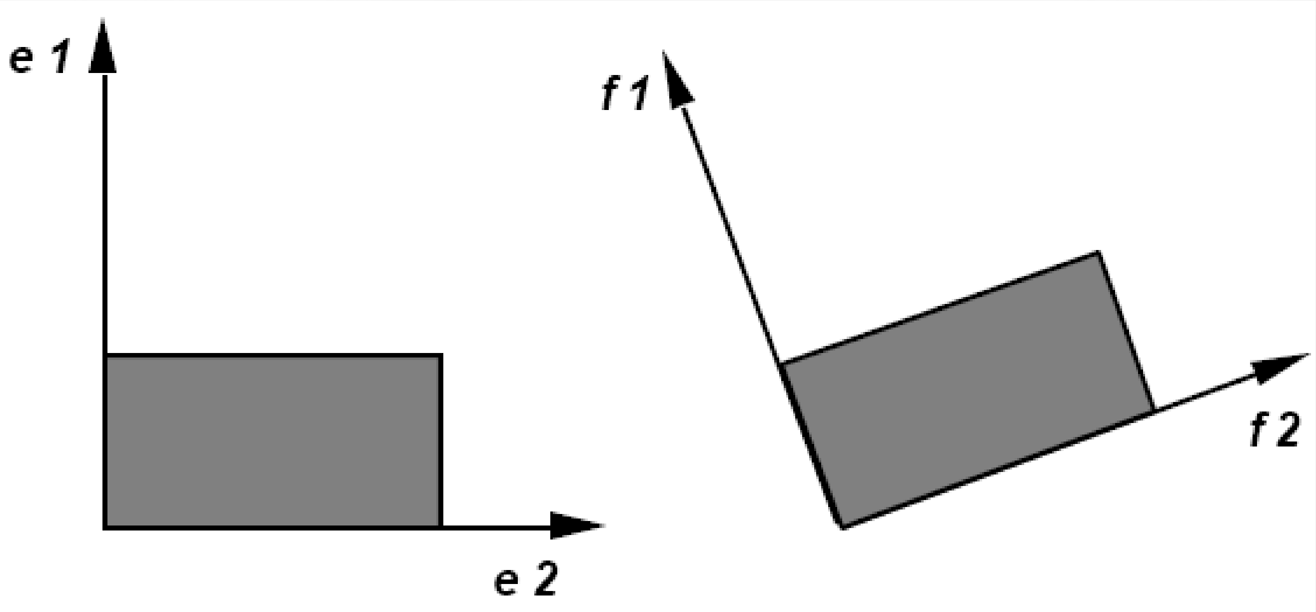
Let *x be a vector*
and y its
transformed
version
$y = Tef(x)$.



Linear Transformations

Consider the rectangle shown on the left in Figure, and suppose we want to orient it such that it aligns with the rectangle on the right.

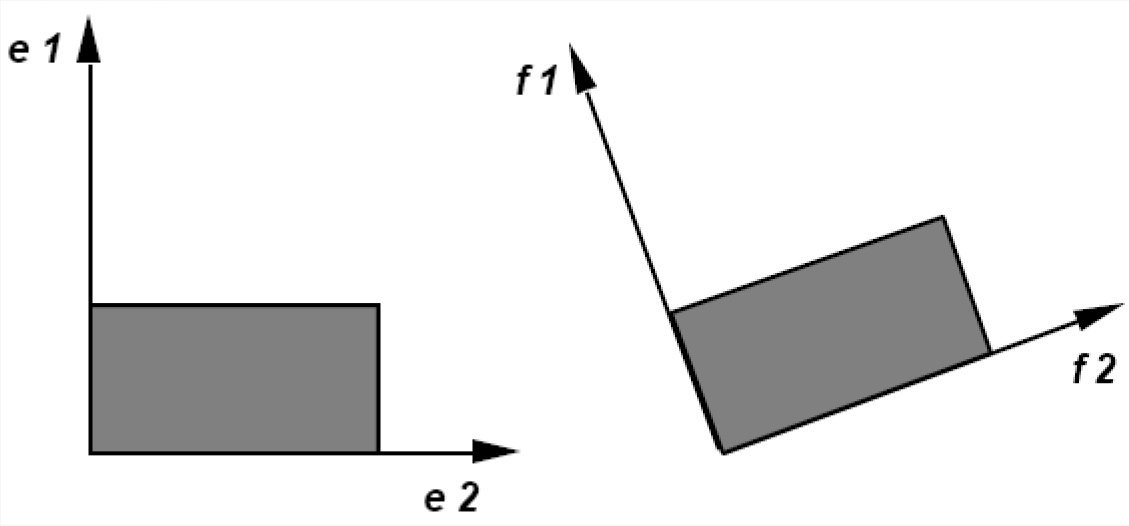
Orienting an object by frame alignment



Linear Transformations

All we need is a transformation with a matrix whose columns are the components of the vectors *F in the basis E. (In 2-D it* is easy to determine the required transformation by other means, but in a 3-D example it would not be as easy.)

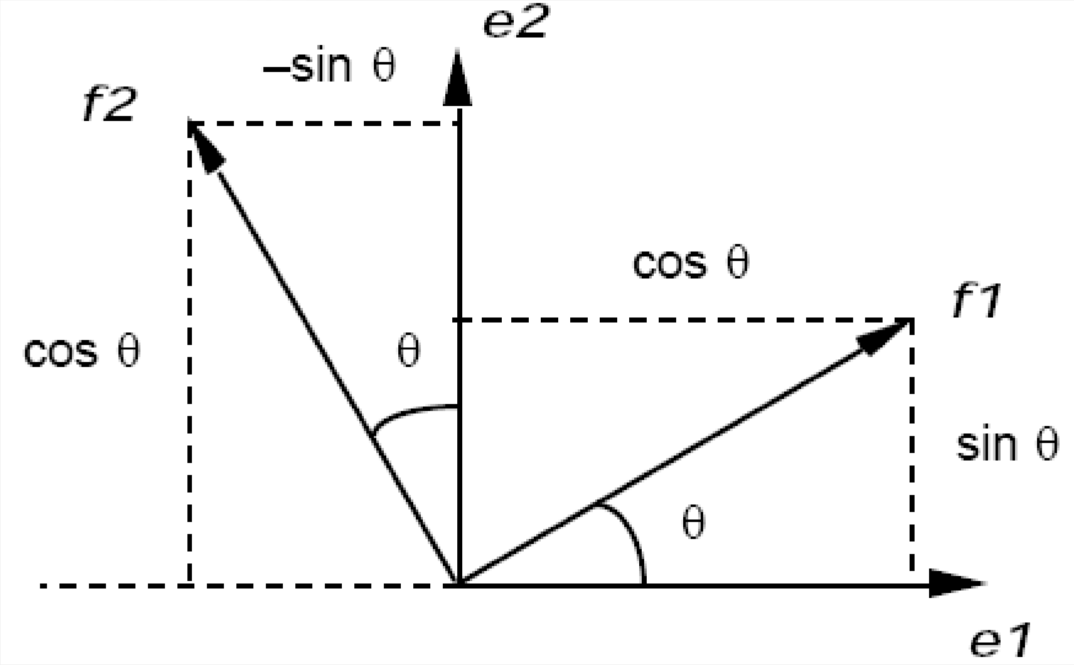
Orienting an object by frame alignment



Linear Transformations

For a specific example, let us determine the matrix that corresponds to a counterclockwise rotation by an angle . The components of the vectors *F are easy to calculate from* elementary trigonometry, as shown in Figure.

Derivation of a rotation matrix



Linear Transformations

We obtain $F_1^e = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \quad F_2^e = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$

and therefore the rotation matrix is

$$M^e = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Composition of successive transformations corresponds to matrix multiplication

$$\boldsymbol{T}_2(\boldsymbol{T}_1(\boldsymbol{x})) \xleftrightarrow{E} M_2^e M_1^e X^e$$

Linear Transformations

And the inverse transformation, which maps a basis *F onto a basis E corresponds to the* inverse matrix

$$M_{fe}^{e} = {M_{ef}^{e}}^{-1}$$

Matrix multiplication is not commutative, i.e., in general *AB≠BA for arbitrary square* matrices *A and B. The inverse of a matrix product reverses the order of the matrices:*

$$(AB)^{-1} = B^{-1}A^{-1}$$

Note that some linear transformations do not map a basis onto another.

Linear Transformations

Here we investigate several interesting linear transformations in 2-D. The results apply also to 3-D, with minor and obvious modifications.

Scaling – Consider the transformation with matrix

$$\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

To study its effect on a vector we multiply the corresponding matrices $\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax \\ by \end{bmatrix}$

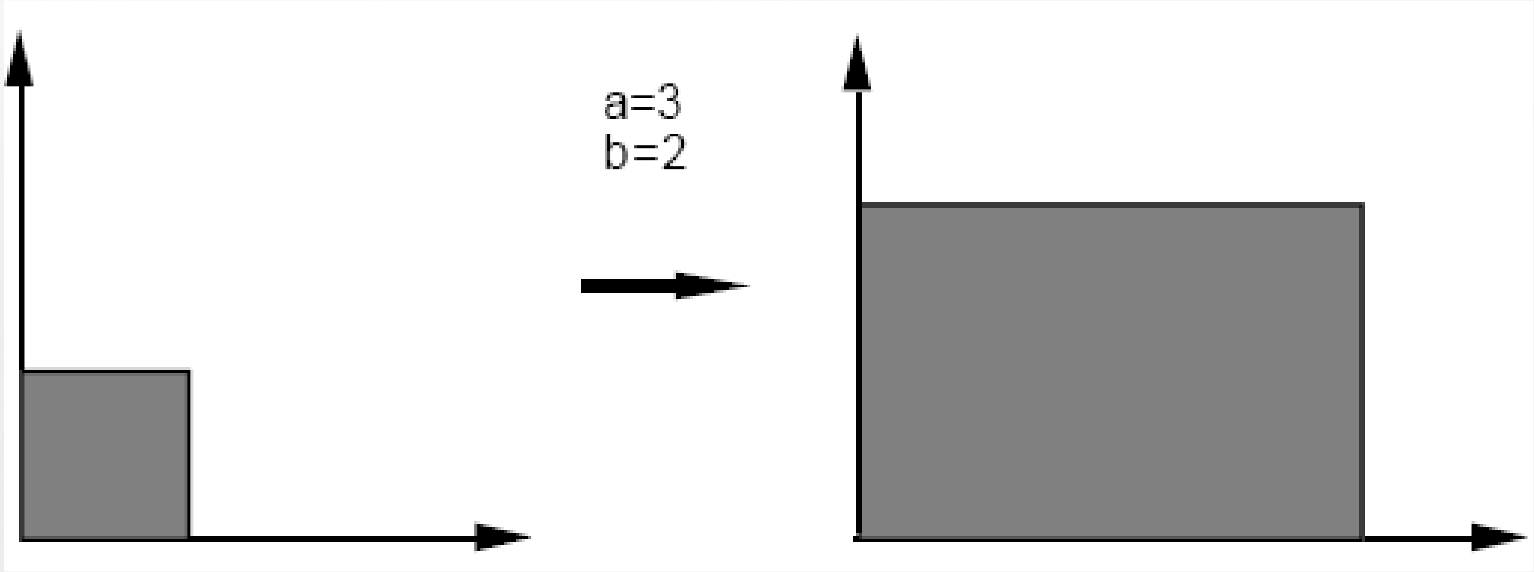
Linear Transformations

Here we are denoting the components of a 2-D vector by the customary *x and y. The result* is a scaling by factors *a and b along the x and y axes. If a = b the scaling is uniform or* isotropic and alters the size of an object but not its shape. If both scale factors equal unity, the transformation is the identity and does not modify the object.

$$\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax \\ by \end{bmatrix}$$

Linear Transformations

Figure illustrates anisotropic scaling by its effect on a square located at the origin.



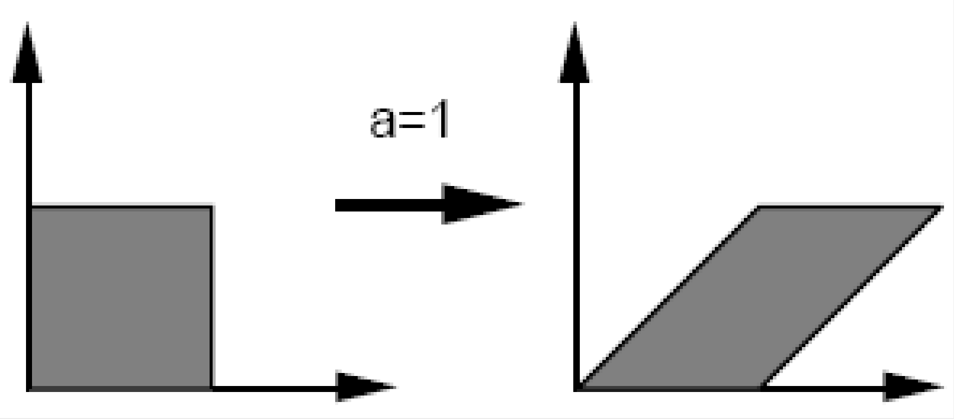
Non-uniform scaling

Linear Transformations

Shear – Now let one of the off-diagonal elements of the matrix be non-zero. The result is a shear, with the following matrix, and with the effect shown in Figure.

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

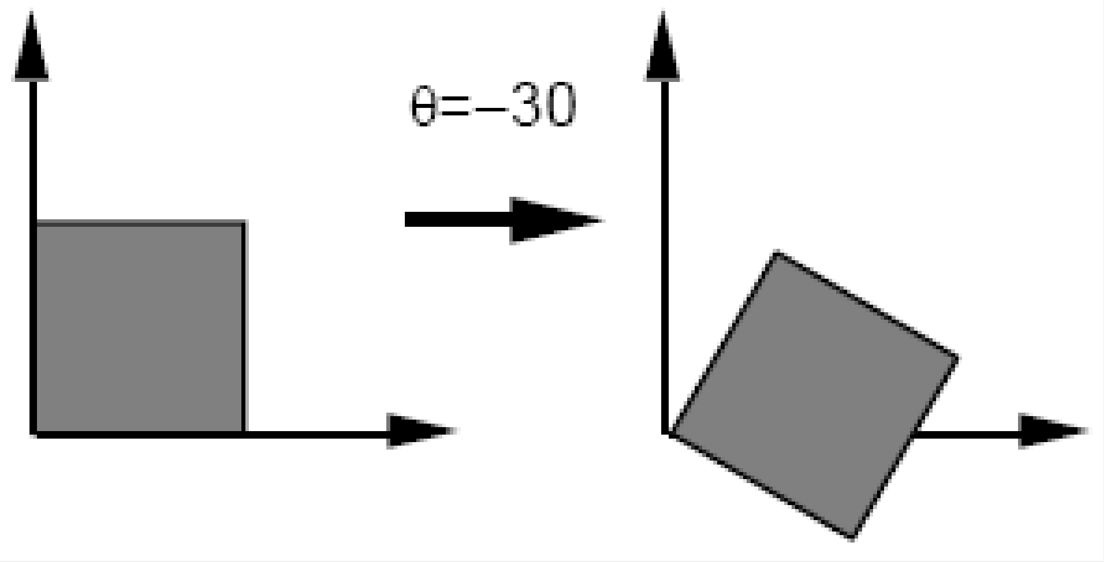
Shear



Linear Transformations

Rotation – As we saw earlier, the matrix is

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



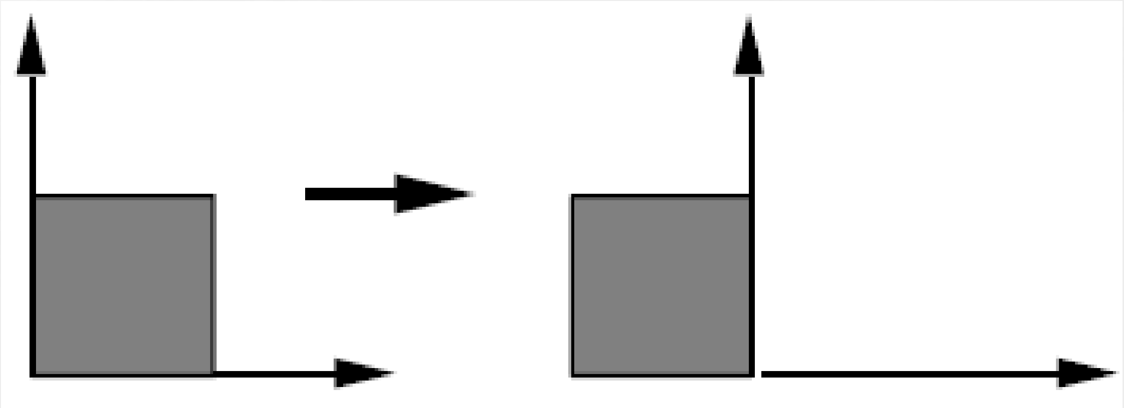
Rotation

Linear Transformations

Reflection – Scalings with negative factors produce reflections. A reflection about the x axis is shown below.

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

Reflection about the vertical axis



Reflections about the horizontal axis, or about the origin can be constructed similarly.

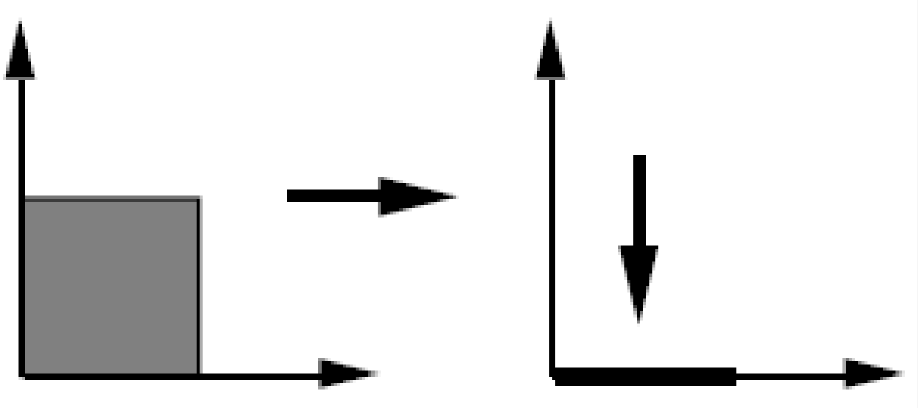
Linear Transformations

Orthographic projection – Consider now

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix}$$

This transformation zeroes the *y component and does not affect the x component. It* corresponds to a perpendicular or orthographic projection on the *x axis.*

Orthographic projection on the horizontal axis

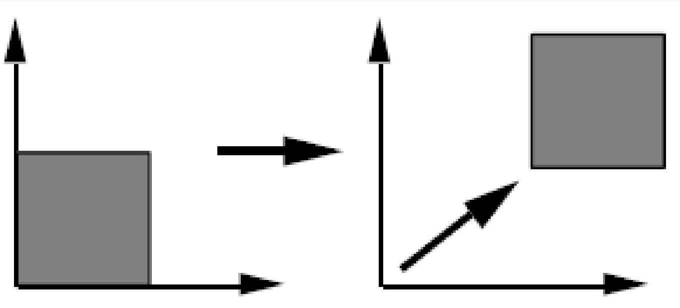


Linear Transformations

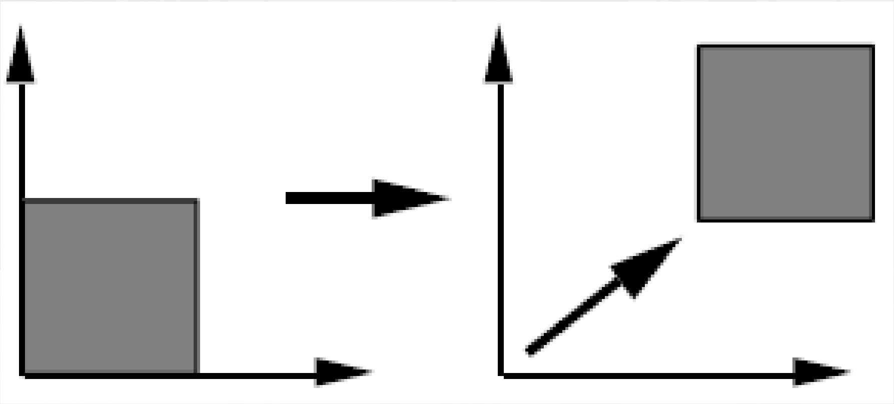
Orthographic projection does not map a basis onto another basis. It is called a *singular* transformation, and cannot be inverted. The projection causes a loss of information about the *y components of the vectors. Knowledge of the x component is insufficient to recover a* vector, because many vectors project on the same point of the *x axis.*

Rigid Motions

A translation Δ is a mapping that associates to each vector *x the sum* $x + \delta$*, where* δ *is a* constant vector. Translations are not linear transformations and cannot be computed by matrix multiplication as we have been doing (but see Section below). The components of a translated vector $y = x + \delta$ *are* $Y = X + D$, where *D is the column matrix that corresponds to the translation vector. A translation is* shown in Figure.



Rigid Motions



Translation

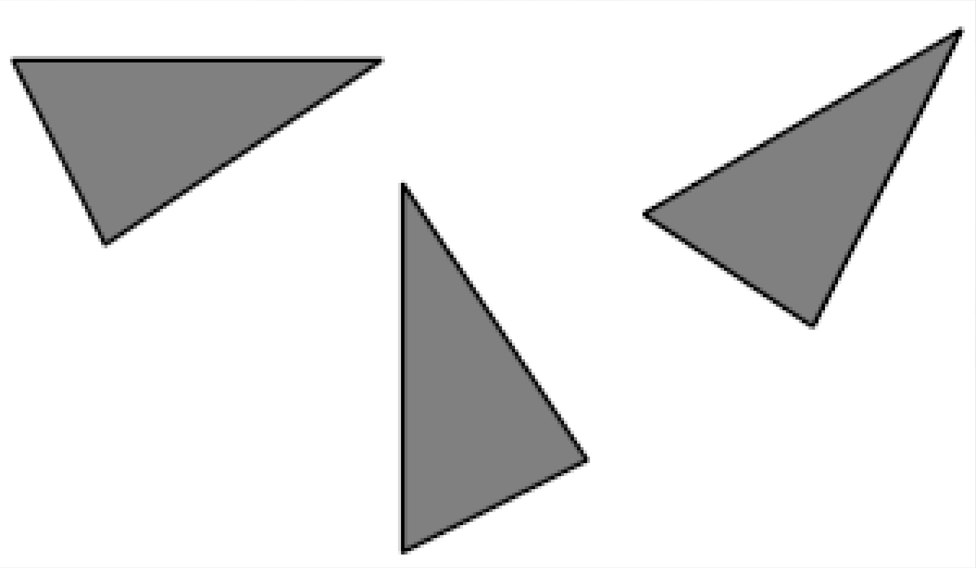
Compositions of translations and linear ransformations are called *affine transformations.* Both translations and linear transformations are practically important, and their nonuniform behavior with respect to components is computationally inconvenient. Separate procedures must be written for dealing with translations and linear transformations, and they cannot be composed by matrix multiplication. (We will see later that both transformations can be treated uniformly if we introduce homogeneous coordinates.)

Rigid Motions

Typically, in geometric modeling we do not want to change the shape of a transformed object. Transformations are applied primarily to locate and orient objects. Transformations that preserve distance are called *isometries (from the Greek, meaning "same measure").* Isometries that also preserve the signed angles between vectors are called in this course *rigid motions. (This is not entirely standard terminology; some texts consider "rigid* motions" and "isometries" as synonyms.) It can be shown that rigid motions are affine and must be *compositions of translations and rotations.*

Rigid Motions

Figure shows several congruent triangles in the plane.



Instances of a triangle

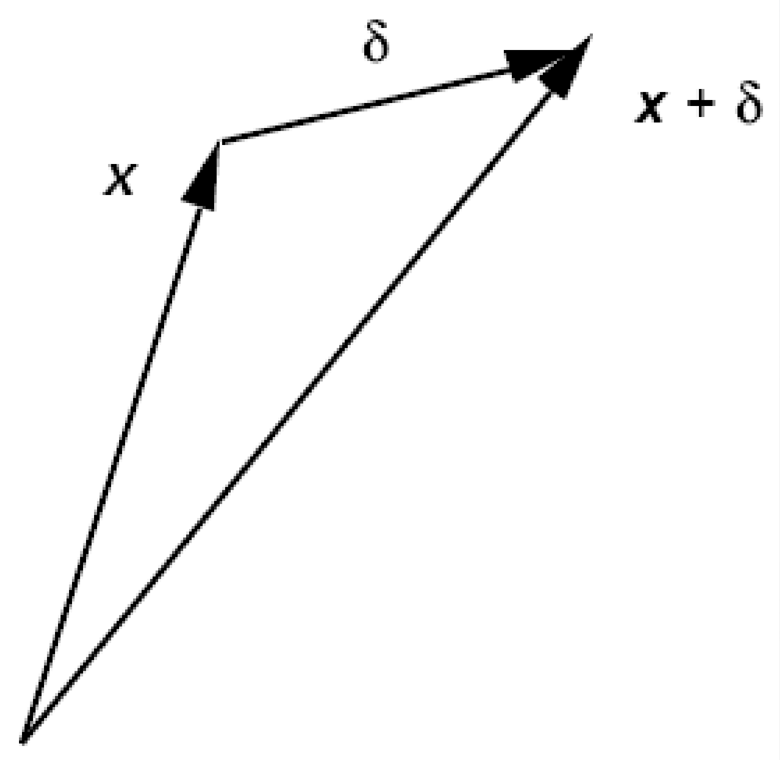
The matrix that corresponds to a rotation in an orthonormal basis is a special case of a so called *orthogonal matrix. These matrices can be inverted easily, by transposition:*

$$M_{orth}^{-1} = M_{orth}^{t}$$

Rigid Motions

Free and Applied Vectors

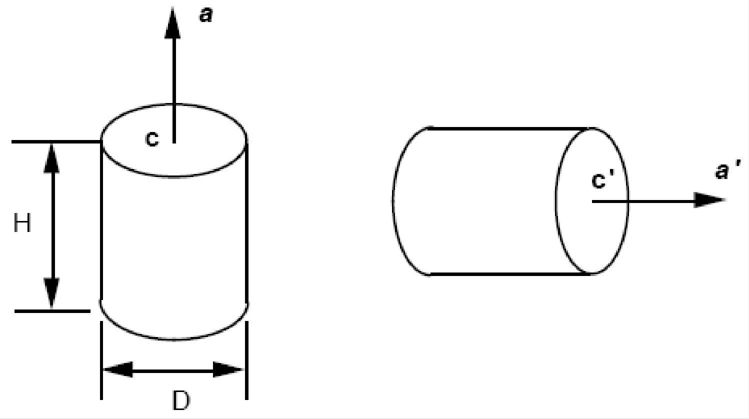
Translation by vector addition



We defined translation of a vector *x as the addition to x of a vector* , *as shown in Figure.* Vector translation does not correspond to the intuitive notion of translation of an “arrow” by translating its endpoints, without changing the length or the direction of the arrow.

Rigid Motions

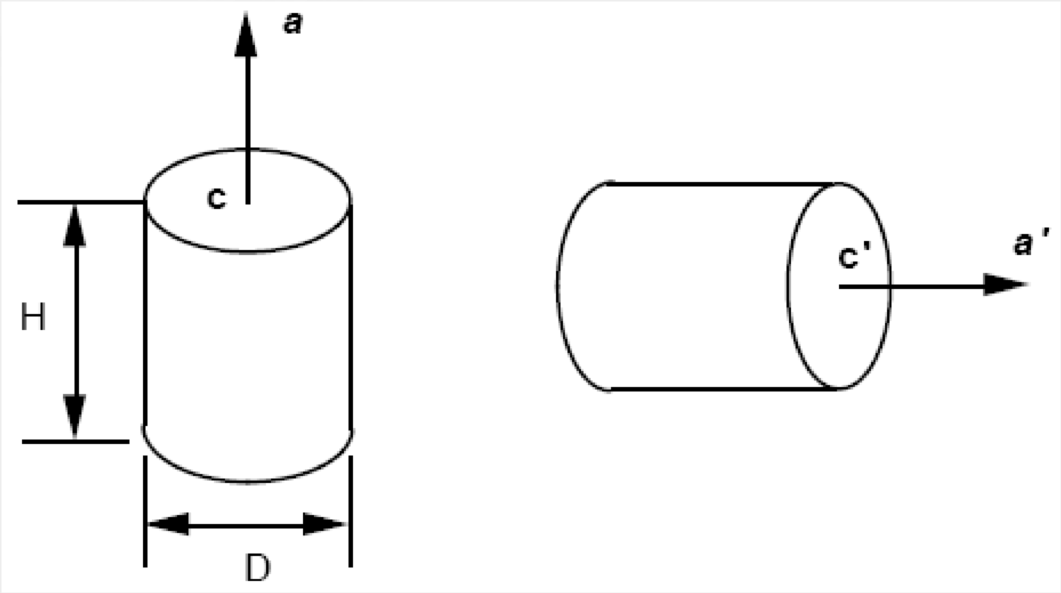
Consider the right, circular cylinder shown on the left in Figure 2.3.2. The cylinder is characterized completely by two scalar parameters—its diameter *D and height H—plus a* point c—the center of a base—and a vector *a along the cylinder's axis. Suppose now that* we want to move the cylinder to a different location and orientation, shown on the right in the figure.



Rigid Motions

Mathematically, moving the cylinder corresponds to applying a rigid motion T to it. How can we compute the values c' and a' *that characterize the cylinder after the* application of T*? Clearly* $c' = T(c)$*. But* $a' \neq T(a)$ *because* T *has a translational* component.

Moving
a cylinder



Rigid Motions

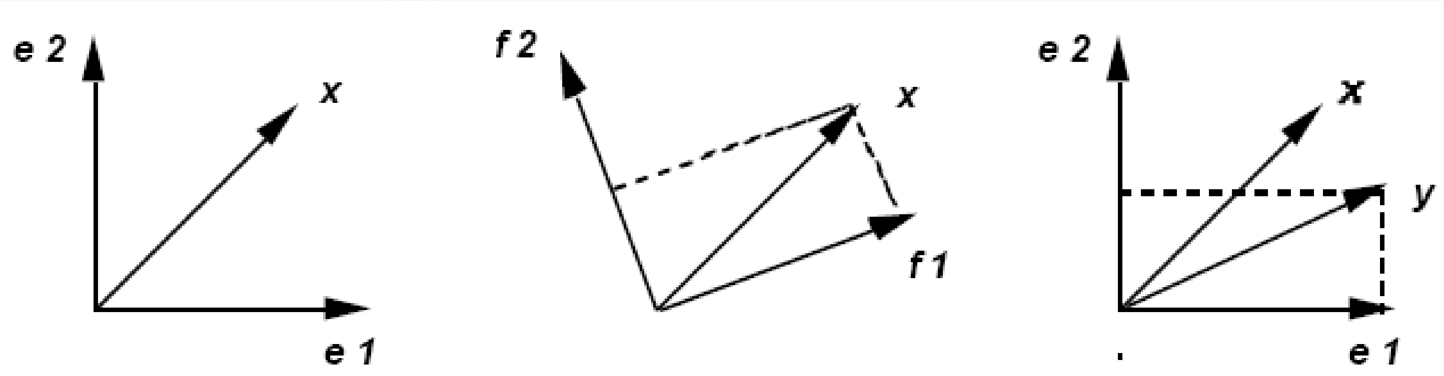
The cylinder in our example can be described by scalars *D and H, point c, and free vector a. Intuitively, it is helpful to think of a as being attached to the point c. This notion may be* formalized by defining yet another entity, called an *applied vector, which consists of a pair* (p, *x), where p is a point and x a free vector. Equivalently, we can define an applied* vector as a pair of endpoints (p, q) with q = p + *x. An applied vector is transformed by* applying a transformation to both endpoints. In Figure the pair (c, *a) is an applied* vector, which transforms as shown on the right in the figure.

Rigid Motions

Free and applied vectors are used extensively in geometric modeling. For example, the normal direction to a surface is often represented by a free vector plus the point at which the normal is calculated, i.e., by an applied vector. (Point information is unnecessary for planar surfaces, which have a single, constant normal.) Tangential directions for curves are treated similarly.

Vector transformation

The need for inversion has a simple geometric interpretation, illustrated by the example of Figure. Consider a vector *x in base E, on the left in the figure. If we rotate the basis* by an angle to obtain basis *F, as shown in the center of the figure, the components of x* change.



Vector transformation versus change of basis

Homogeneous Coordinates

We begin this section with a pragmatic view of homogeneous coordinate methods. We then explain them geometrically, and finally show how they can be used to compute perspective projections.

Transformations in Homogeneous Coordinates
Translations and linear transformations can be treated more uniformly if we introduce a different system of coordinates, called *homogeneous coordinates. For simplicity we work* in 2-D, but generalizations to 3-D or *n-D are straightforward.*

Homogeneous Coordinates

We continue to make no distinction between points and ordinary vectors. Suppose that we have a vector *x with* components *X , and want to apply to it a linear transformation with matrix M, so as to* obtain another vector *y with components Y. We introduce an additional component and* associate with the vector *x the column matrix*

$$X^* = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

The elements of *X* are called homogeneous coordinates. We also add a third row and* column to the linear transformation matrices as follows

$$X^* = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ 1 \end{bmatrix} \qquad M^* = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This matrix can be written in block format as

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

Homogeneous Coordinates

where *M is the usual 2 by 2 linear transformation matrix, and the two-zero row and column* are both denoted by 0. Multiplying the matrices

$$M^* X^* = \begin{bmatrix} M & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = \begin{bmatrix} MX \\ 1 \end{bmatrix} = \begin{bmatrix} Y \\ 1 \end{bmatrix} = Y^*$$

shows how to evaluate the effects of a linear transformation in the new, augmented-matrix format. Scalings, shears, rotations, and so on, can be achieved by replacing *M in the 3 by* 3 matrix above by the various matrices we discussed earlier.

Homogeneous Coordinates

Let us now investigate what happens if the elements of the third column of the matrix become non-zero. Consider

$$M^{*} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}$$

and apply it to a generic vector:

$$Y^{*} = M^{*} X^{*} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + a \\ y + b \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

This is precisely the result of translating *x by a vector with components (a, b). Therefore* we have found a method for computing both translations and linear transformations by matrix multiplication. In particular, rigid motions in the plane are associated with 3 by 3 matrices. In 3-D they correspond to 4 by 4 matrices.

For reference, the three matrices that correspond to rotations about the *x, y and z axes are:*

Homogeneous Coordinates

$$x:\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & -\sin\theta & 0\\ 0 & \sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \quad y:\begin{bmatrix}\cos\theta & 0 & \sin\theta & 0\\ 0 & 1 & 0 & 0\\ -\sin\theta & 0 & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \quad z:\begin{bmatrix}\cos\theta & -\sin\theta & 0 & 0\\ \sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Uniform treatment of translations and rotations is computationally important. It implies that we only need one procedure to implement both, and that matrix-multiplication hardware can be used for both. We will see later that homogeneous coordinates also can deal with projections, which are needed for displaying objects.

Homogeneous Coordinates

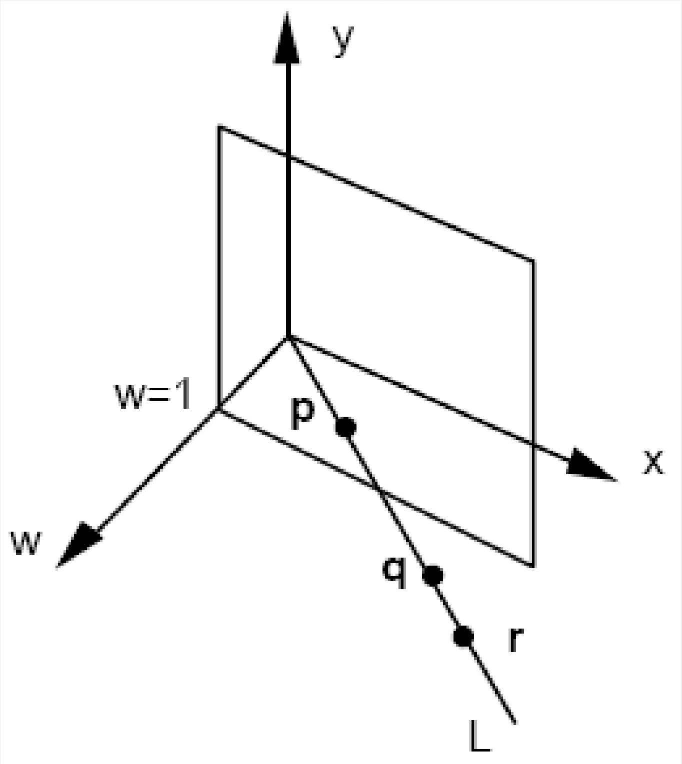
Geometric Interpretation

Homogeneous-coordinate methods were introduced above as convenient “recipes”. But they have a rich body of mathematics and geometric intuition underlying them. Here we explore it briefly. First we generalize slightly, and write the homogeneous coordinates of an Euclidean point p as

$$P^* = \begin{bmatrix} x \\ y \\ w \end{bmatrix}.$$

Homogeneous Coordinates

We have increased the dimension of our space by one. In addition, since we identify points at *w=1 with Euclidean points, we have placed the standard Euclidean plane at w=1. Figure* illustrates this construction.



The Euclidean plane imbedded in an auxiliary 3-space

Homogeneous Coordinates

In particular, if *w≠1 and is not zero, we can always scale all the* components so as to *normalize the coordinates:*

$$\begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}$$

The set of all lines through the origin of our auxiliary 3-D space is called the *projective plane. The elements of the projective plane are called projective points.*

Homogeneous Coordinates

Perspective

Thus far we have only used homogeneous-coordinate matrices with a last row whose offdiagonal elements are null. Let us now investigate wnat happens when they are non-null. Consider the product

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/d & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 - x/d \end{bmatrix}$$

Homogeneous Coordinates

Perspective

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/d & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 - x/d \end{bmatrix}$$

(We use a –1/*d term for reasons that will be obvious soon.) The result is no longer on the w=1 plane. Normalizing it we obtain*

$$\begin{bmatrix} \dfrac{x}{1 - x/d} \\ \dfrac{y}{1 - x/d} \\ 1 \end{bmatrix}$$

Perspective

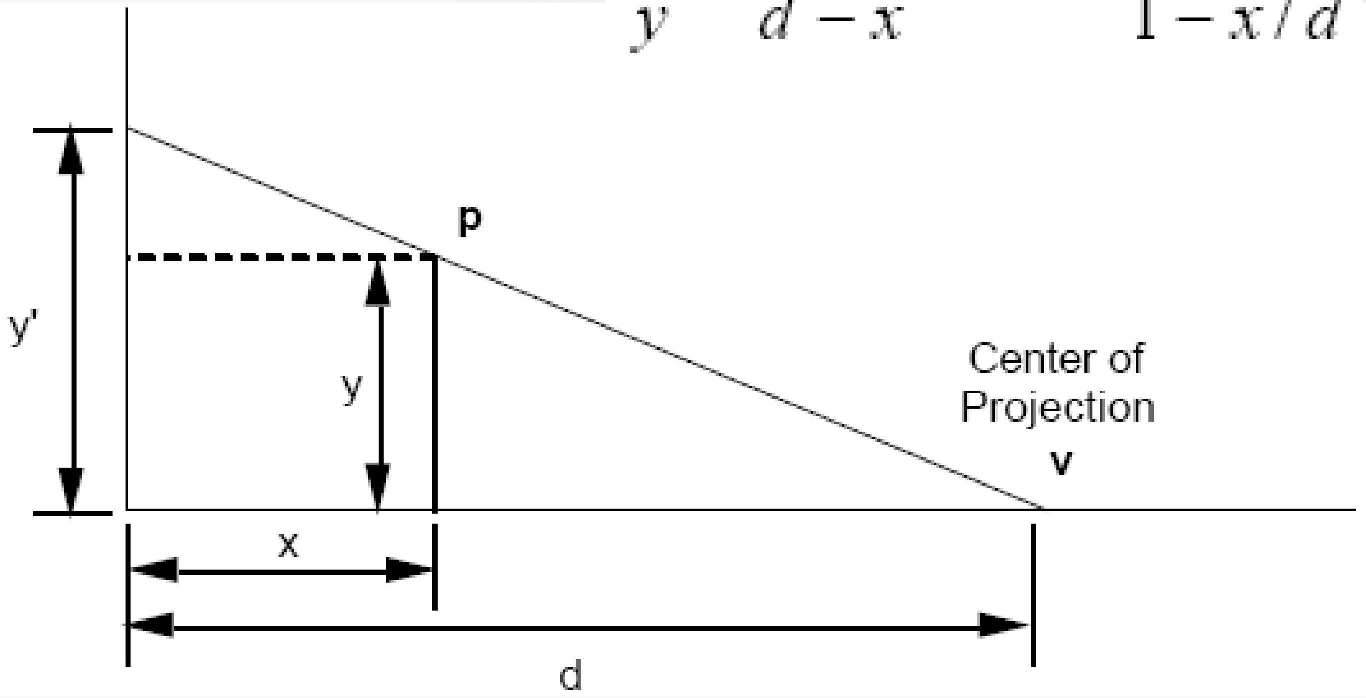
What is the physical meaning of this transformation? We will answer this question with the help of Figure, which shows how to project a point on the *y axis of the Euclidean* plane from a center of projection v lying on the *x axis at x=d. By similarity of triangles*

$$\frac{y'}{y} = \frac{d}{d-x}, \quad y' = \frac{y}{1-x/d}$$

This is precisely the *y coordinate we computed above by matrix multiplication.*

Perspective

$$\frac{y'}{y} = \frac{d}{d-x}, \quad y' = \frac{y}{1-x/d}$$



Central projection of a 2-D point on the vertical axis

Perspective

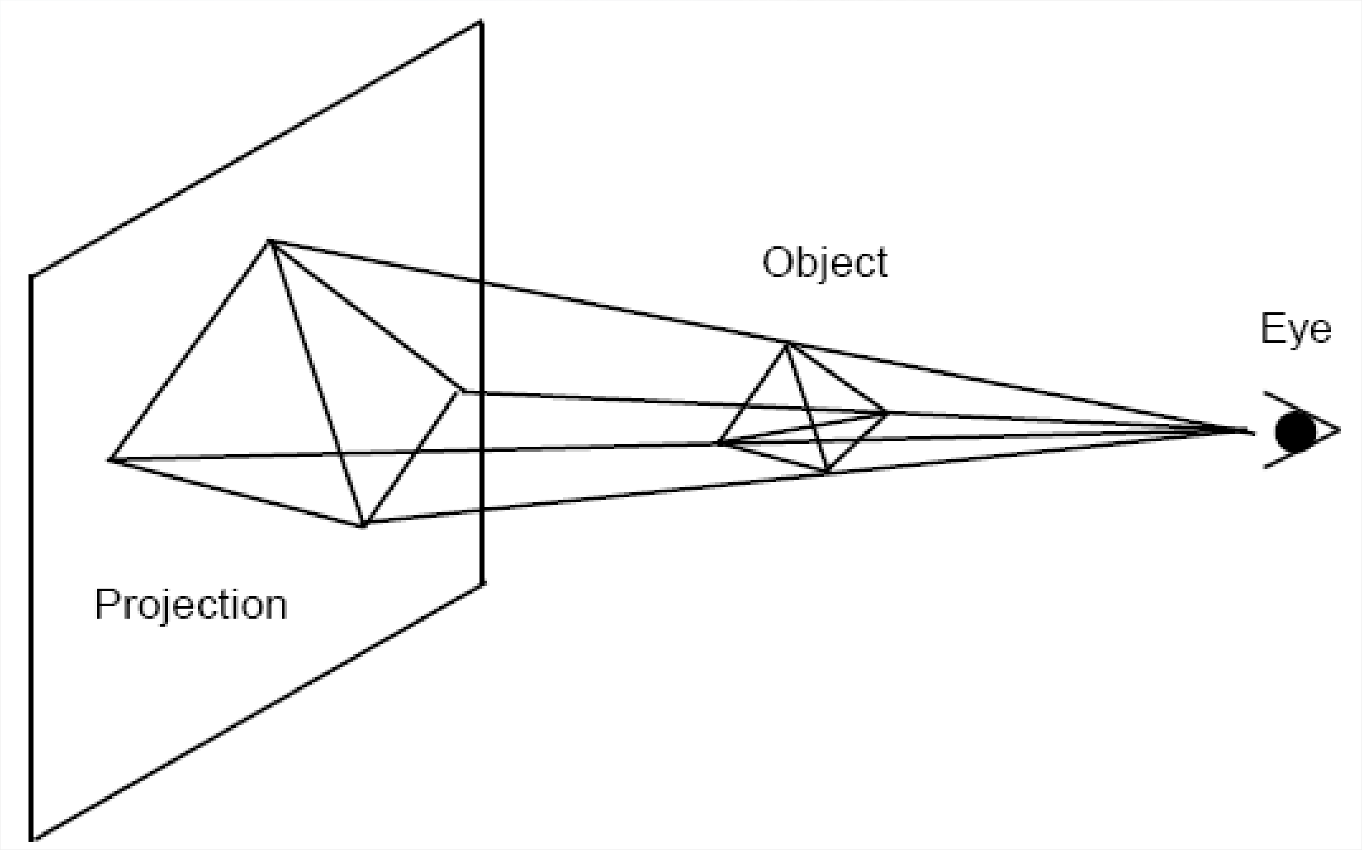
In 3-D, an analogous argument shows that multiplication by the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 1 \end{bmatrix}$$

provides us with the *x and y coordinates of the projection of a point on the xy plane, from a* center of projection on the *z axis at z=d. Projection on a plane is a fundamental operation* for the generation of a display—see Figure.

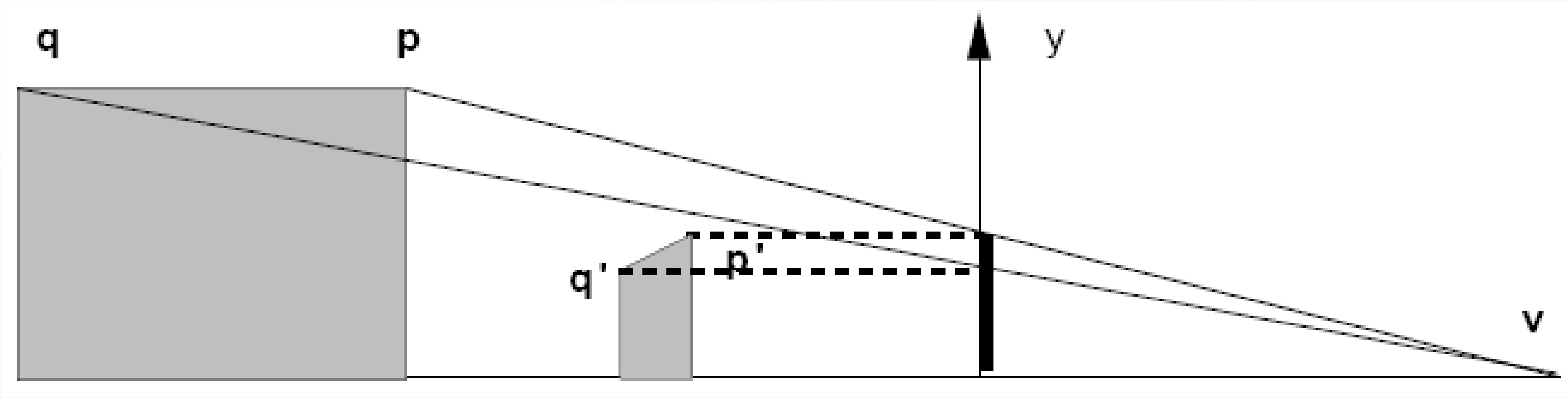
Perspective

Drawing an object by projecting it on a plane



Perspective

Perspective transformation applied to a 2-D solid



Perspective

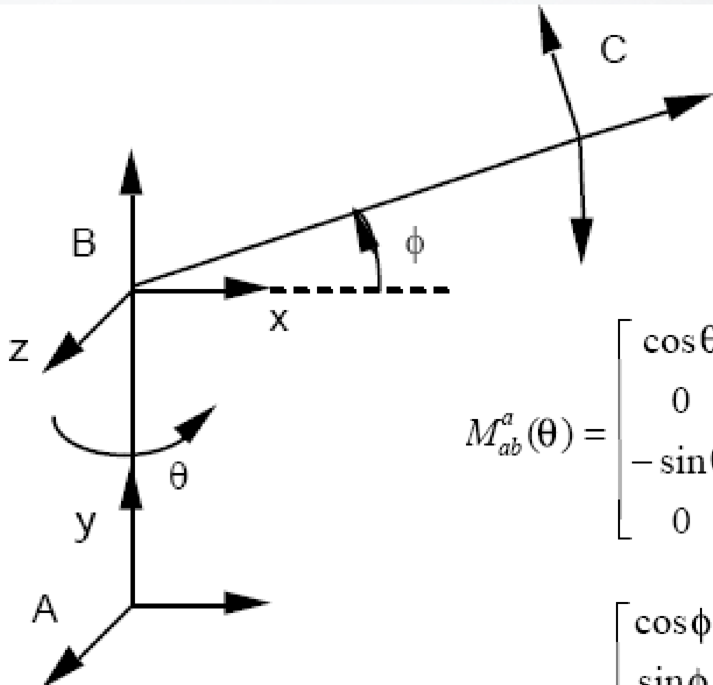
In 3-D the perspective transformation produces a deformed 3-D object, which must be projected orthographically onto the *xy plane to generate the desired 2-D image. Computing* a planar projection involves matrix multiplication, followed by normalization and orthographic projection. This latter involves essentially no computation, since it amounts to ignoring the *z coordinate. But normalization is relatively expensive, because it requires a* division.

Applications in Robotics and Simulation

A robotic manipulator is a kinematic chain, i.e., a collection of solid bodies—called *links—connected at joints. The most common joints are the revolute joint, which* corresponds to rotational motion between two links, and the *prismatic joint, which* corresponds to a translation. Most of the industrial robot "arms" in use today have only revolute joints. Figure shows an idealized robot with two links and two revolute joints.

Applications in Robotics and Simulation

Stick-figure model for a 2-link robot



$$X^a = M^a_{ab} X^b = M^a_{ab} M^b_{bc} X^c$$

$$X^a = M^a_{ab}(\theta) M^b_{bc}(\phi) X^c$$

$$M^a_{ab}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & L_1 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M^b_{bc}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & L_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & L_2\cos\phi \\ \sin\phi & \cos\phi & 0 & L_2\sin\phi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

References

- *CAD/CAM Theory and Practice* , Ibrahim Zeid, McGraw Hill , 1991

- *Mathematical Elements for Computer Graphics*, Rogers, D.F., Adams, J.A., McGraw Hill, 1990.

- Computer Aided Geometric Design, Thomas W. Sederberg, 2003.