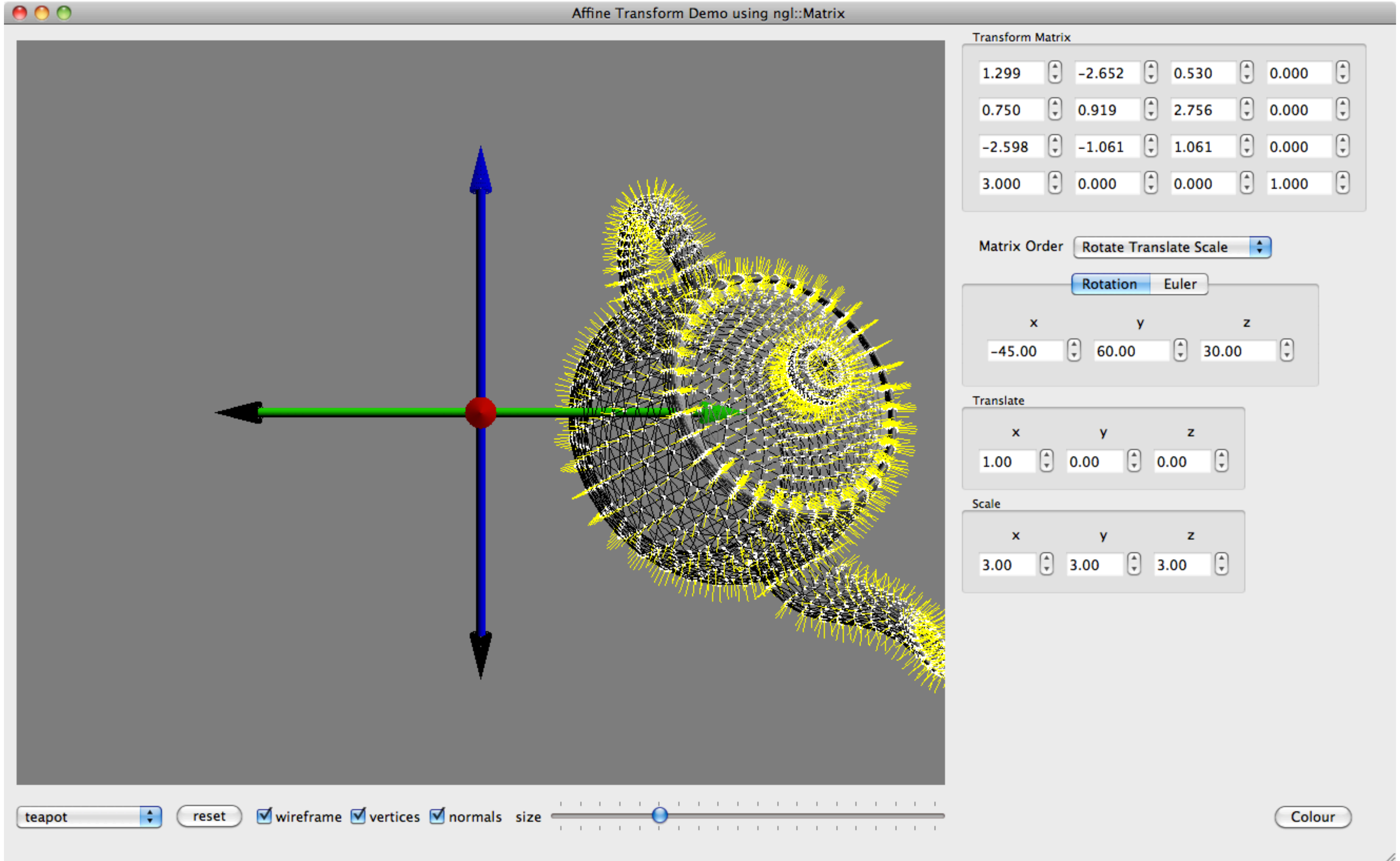
Matrices

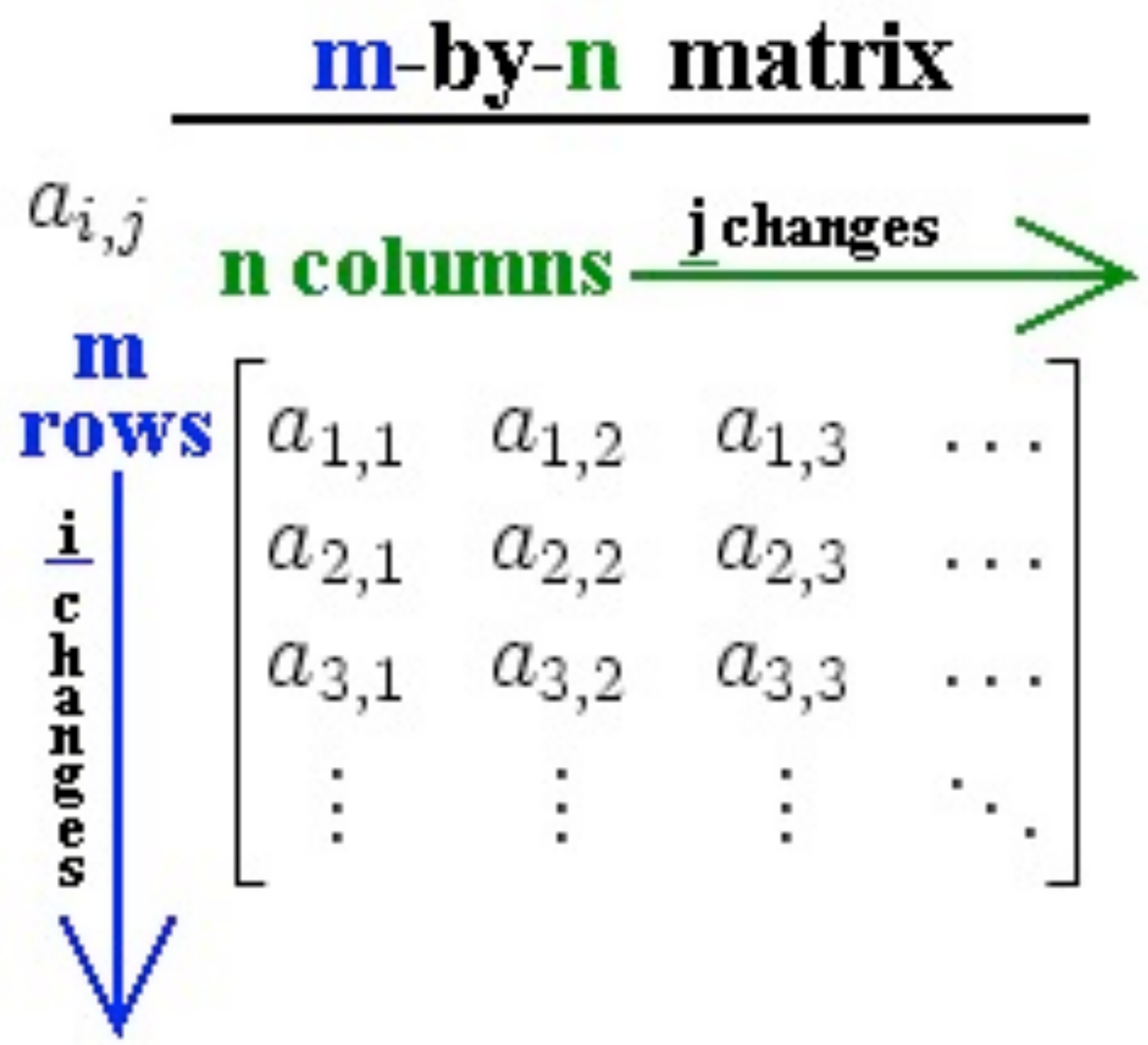
Matrices

- Matrices are convenient way of storing multiple quantities or functions
- They are stored in a table like structure where each element will contain a numeric value that can be the result of a complex equation or process
- In CG we tend to store all modelling transforms in matrix form a pass all vertices of a model through this matrix to scale, translate and rotate the model
- This is done by multiplying all vertices (stored as vector) by the matrix

Matrices in CG



Matrices



- In mathematics, a matrix (plural matrices) is a rectangular table of numbers or, more generally, of elements of a ring-like algebraic structure.
- The entries in the matrix are real or complex numbers or functions that result in real or complex numbers

Image from http://en.wikipedia.org/wiki/Image:Matrix.png

Definitions and Notations

- The horizontal lines in a matrix are called **rows** and the vertical lines are called **columns**.
- A matrix with m rows and n columns is called an m-by-n matrix (or mn matrix) and m and n are called its dimensions.
- The entry of a matrix A that lies in the i-th row and the j-th column is called the i, j entry or (i, j)-th entry of A.
- This is written as $A_{i,j}$ or $A[i, j]$.
- We often write $A := [a_{i,j}]_{m \times n}$ to define an $m \times n$ matrix A with each entry in the matrix $A[i, j]$ called a_{ij} for all $1 \leq i \leq m$ and $1 \leq j \leq n$.

Example

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

is a 4×4 matrix, the element $A[2,3]$ or $a_{2,3}$ is 7

Example

$$\begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix}$$

is a 4×4 matrix, the element $A[2,3]$ or $a_{2,3}$ is 7

General Matrix of the Form $A = m \times n$

- The general form of a matrix is represented as

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

- or as $A = [a_{ij}]_{m \times n}$ or simply as $A = [a_{ij}]$ if the **order** is known

Operations with Matrices

- A matrix has m rows and n columns and is said to be of **order** $m \times n$
- Thus the matrix

$$M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

- Has three rows and two columns and is of order 3×2 not to be confused with

$$N = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

- These matrices are related and N is the **transpose** of M or

$$N = M^T$$

Multiplication of a Matrix by a Number

- Multiplying a matrix by a scalar number is simply multiplying every element by the scalar

$$2\begin{bmatrix} 6 & 10 \\ 4 & 6 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 2\times 6 & 2\times 10 \\ 2\times 4 & 2\times 6 \\ 2\times 3 & 2\times 5 \end{bmatrix} = \begin{bmatrix} 12 & 20 \\ 8 & 12 \\ 6 & 10 \end{bmatrix}$$

- More generally, for any $\alpha \in \mathbb{R}$,

$$\alpha A = \alpha \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \vdots & a_{24} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} \alpha a_{11} & \alpha a_{12} & \cdots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \cdots & \alpha a_{24} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha a_{m1} & \alpha a_{m2} & \cdots & \alpha a_{mn} \end{bmatrix}$$

More Matrix Definitions

- A matrix of any order who's entries are all zero is called a **zero matrix** and this is represented by $\mathbf{O}$
- The matrix $(-A)$ is defined to be the matrix $(-1)A$
- Note that if $\alpha = 0$ then for any matrix A or αA is a zero matrix

Addition of Matrices

- If two matrices A and B are of the same order $m \times n$, then $A + B$ is the matrix defined by

$$A + B = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdot & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$$

Addition of Matrices

- For example let A B and C be matrices as shown below

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \; B = \begin{bmatrix} 2 & -1 & 3 \\ 4 & 2 & -1 \end{bmatrix} \; C = \begin{bmatrix} 2 & 7 \\ 5 & 2 \end{bmatrix}$$

Then

$$A + B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 2 & -1 & 3 \\ 4 & 2 & -1 \end{bmatrix} = \begin{bmatrix} 3 & 1 & 6 \\ 8 & 7 & 5 \end{bmatrix}$$

Neither $A + C$ nor $B + C$ can be found since C has a different order from that of A and B

Also note that $A + B = B + A$

Subtraction of Matrices

- To subtract elements of matrices A and B

$$C = A(-)B$$

- elements of matrix B are subtracted from their corresponding elements in matrix A and stored as elements of matrix C
- All the matrices must have the same number of elements.
- In the above example the $-$ sign is enclosed in parentheses to indicate subtraction of matrix elements
- Although for most operation - is sufficient unless we are looking at more advanced matrix theory (which we will not need)

Matrix subtraction

- The subtraction of matrix elements is defined for two matrices of the same dimension as follows

$$\begin{bmatrix} 1 & 3 \\ 1 & 0 \\ 1 & 2 \end{bmatrix} (-) \begin{bmatrix} 0 & 0 \\ 7 & 5 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1-0 & 3-0 \\ 1-7 & 0-5 \\ 1-2 & 2-1 \end{bmatrix}$$

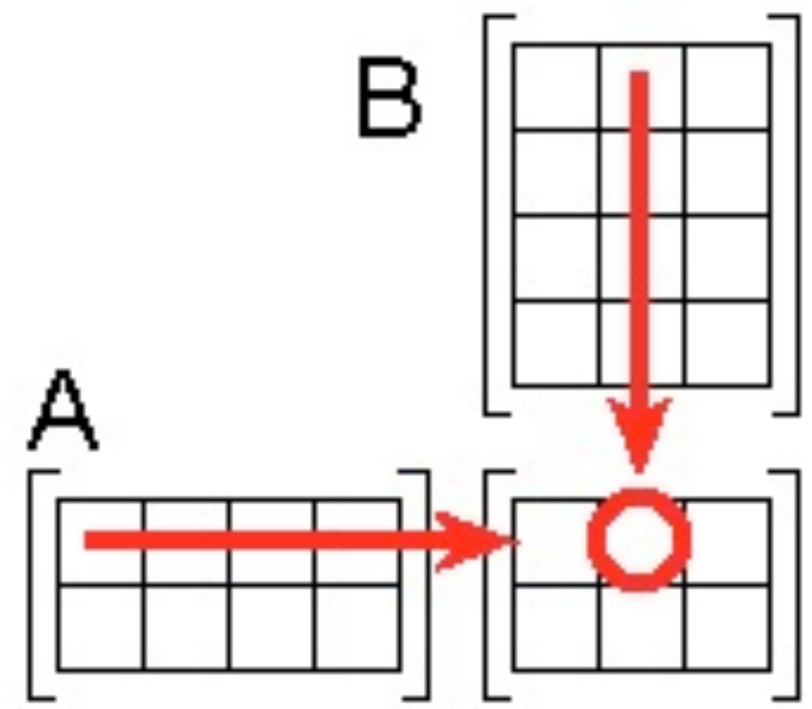
$$= \begin{bmatrix} 1 & 3 \\ -6 & -5 \\ -1 & 1 \end{bmatrix}$$

Matrix Multiplication

- The most common application of matrices in CG is matrix multiplication.
- It is defined between two matrices only if the number of columns of the first matrix is the same as the number of rows of the second matrix.
- If A is an m-by-n matrix and B is an n-by-p matrix, then their product $A \times B$ is an m-by-p matrix given by

$$(A \times B)_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{in} b_{nj}$$

Matrix Multiplication



- The figure shows how to calculate the $(AB)_{12}$ element of $A \times B$
- if A is a 2×4 matrix, and B is a 4×3 matrix.
- Elements from each matrix are paired off in the direction of the arrows; each pair is multiplied and the products are added.
- The location of the resulting number in AB corresponds to the row and column that were considered.

$$(A \times B)_{12} = \sum_{r=1}^{4} a_{1r} b_{r2} = a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42}$$

Matrix Multiplication

- If A is an $m \times n$ matrix and B is an $n \times q$ matrix then the product matrix AB is defined by

$$AB = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix} =$$

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix}$$

where

$$c_{ij} = [a_{i1} a_{i2} \ldots a_{in}] \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{nj} \end{bmatrix} = a_{i1} b_{1j} + a_{i2} b_{2j} + \ldots + a_{in} b_{nj}$$

for $i = 1, \ldots, m$, $j = 1, \ldots, q1$

Affine Transform matrix

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 4 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 2 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 4 & 1 \end{bmatrix}$$

Matrix Multiplication

For example

$$\begin{bmatrix} 6 & 10 \\ 4 & 6 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} 30 \\ 20 \end{bmatrix} = \begin{bmatrix} 6 \times 30 + 10 \times 20 \\ 4 \times 30 + 6 \times 20 \\ 3 \times 30 + 5 \times 20 \end{bmatrix} = \begin{bmatrix} 380 \\ 240 \\ 190 \end{bmatrix}$$

A is an $m \times n$ matrix and B is a $p \times q$ matrix, then the product AB can be formed *if and only if* $n = p$ and if it can be formed, it will be an $m \times q$ matrix.

Matrix Multiplication Properties

- All three notions of matrix multiplication are associative

$$A(BC) = (AB)C$$

- and distributive:

$$A(B + C) = AB + AC$$

- and

$$(A + B)C = AC + BC$$

- and compatible with scalar multiplication:

$$c(AB) = (aA)B = A(cB)$$

The Identity Matrix

- The $n \times n$**identity matrix** $I_n$is defined to be the $n \times n$matrix in which every element on the leading diagonal is a 1, and every other element is a zero. Thus,

$$I_1 = [1]\,, I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, I_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \ldots$$

- An important property of the identity matrix is the anything multiplied by the identity matrix remains the same

- For example

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

Summary of Properties of Matrices

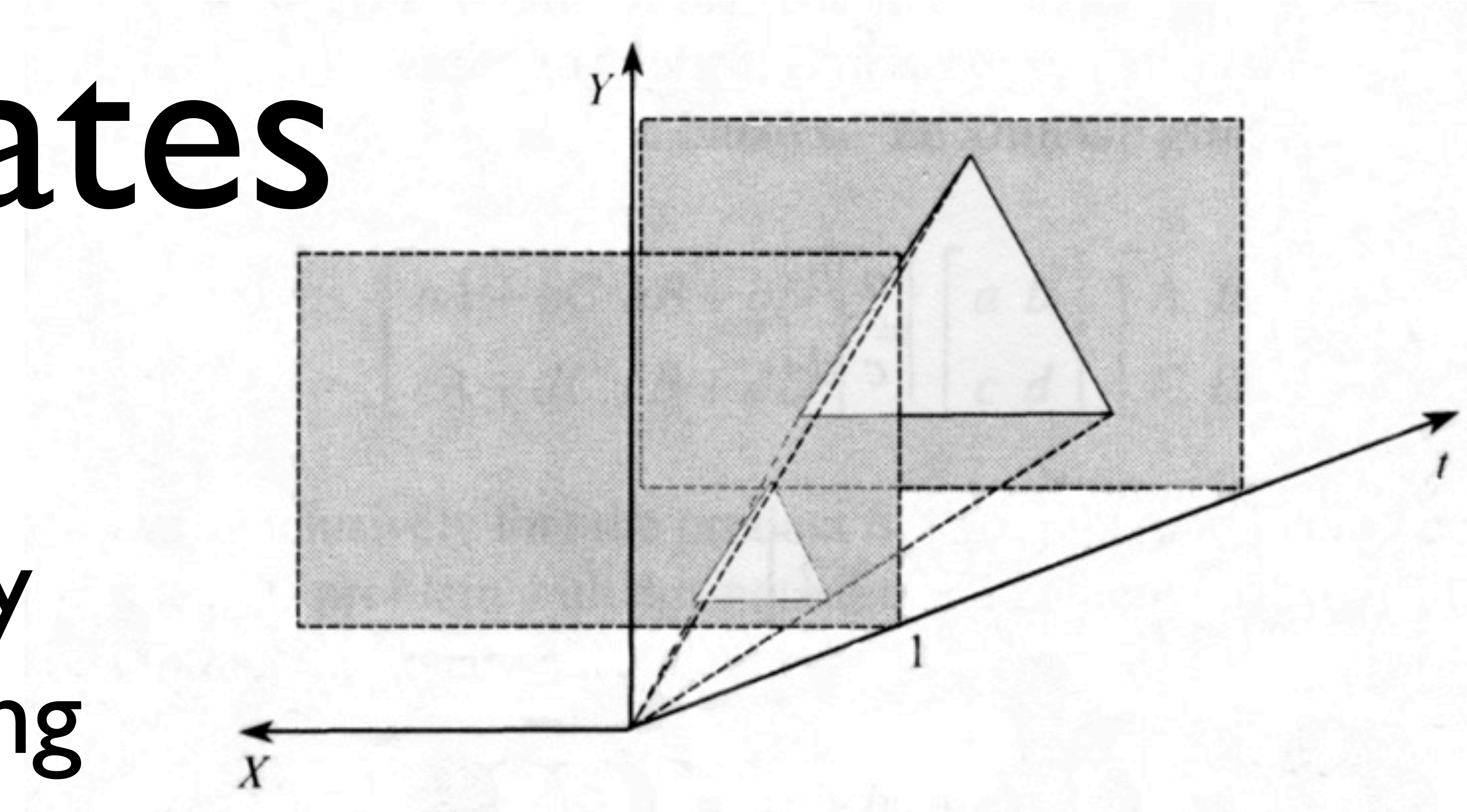
- The following properties are true whenever the operations of addition or multiplication are valid. If A, B and C are matrices and λ is a real number

$$
\begin{aligned}
&A + B = B + A \\
&A + (B + C) = (A + B) + C \\
&\lambda(A + B) = \lambda A + \lambda B \\
&AB \neq BA \text{ (in general)} \\
&A(BC) = (AB)C \\
&A(\lambda B) = (\lambda A)B = \lambda(AB) \\
&A(B + C) = AB + AC
\end{aligned}
$$

Co-ordinate systems and co-ordinate frames

- When representing Vectors and Points we use the same notation for example
- $\mathbf{v} = (3, 2, 7)$ is a vector whereas $P = (5, 3, 1)$
- This makes it seem as if vectors and Points are the same thing
- However Points have a location, but no size and direction
- whereas vectors have size and direction but no location
- What this means is that $\mathbf{v}$ has the components (3,2,7) in the underlying coordinate systems and Similarly P has the coordinates (5,3,1) in the underlying coordinate system.

Homogenous co-ordinates



- Homogenous co-ordinates were proposed by several mathematicians the most notable being Möbius
- Basically homogeneous co-ordinates define a point in a plane using three co-ordinates rather than two
- For a point (x,y) there exists a homogenous point (xt,yt,t) where t is an arbitrary number

Homogenous co-ordinates

- For example given a point (3,4)
- It may be expressed in homogenous co-ordinates as (6,8,2), because 3=6/2 and 4=8/2
- But the homogenous point (6,8,2) is not unique to (3,4); (12,16,4), (15,20,5) and (300,400,100) are also possible homogenous co-ordinates for (3,4)
- The reason why this co-ordinate system is called homogenous is because it is possible to transform functions such as f(x,y) into the form f(x/t,y/t) without disturbing the degree of the curve.

The homogeneous representation of a Point and a Vector

- It is useful to represent both points and vectors using the same set of basic underlying objects $[\mathbf{a}, \mathbf{b}, \mathbf{c}, \vartheta]$
- From the previous equations we see that the vector $\mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c}$ needs the four coefficients $[v_1, v_2, v_3, 0]$
- Whereas the point $P = p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c} + \vartheta$ needs the four coefficients $[p_1, p_2, p_3, 1]$
- The fourth component designates whether the object does or does not include ϑ.

The homogeneous representation of a Point and a Vector

- We can formally write any $\mathbf{v}$ and P using matrix notation multiplication as

$$v = [\mathbf{a}, \mathbf{b}, \mathbf{c}, \vartheta] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix} \text{ and } P = [\mathbf{a}, \mathbf{b}, \mathbf{c}, \vartheta] \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$

- Here the row matrix captures the nature of the coordinate frame, and the column vector captures the representation of the specific object of interest.
- Thus, vectors and points have different representations : vectors have 0 as a forth component whereas points have 1 as a fourth component.

Converting between co-ordinate systems

To go from ordinary to homogeneous coordinates,

if the object is a point, append a 1
if the object is a vector, append a 0

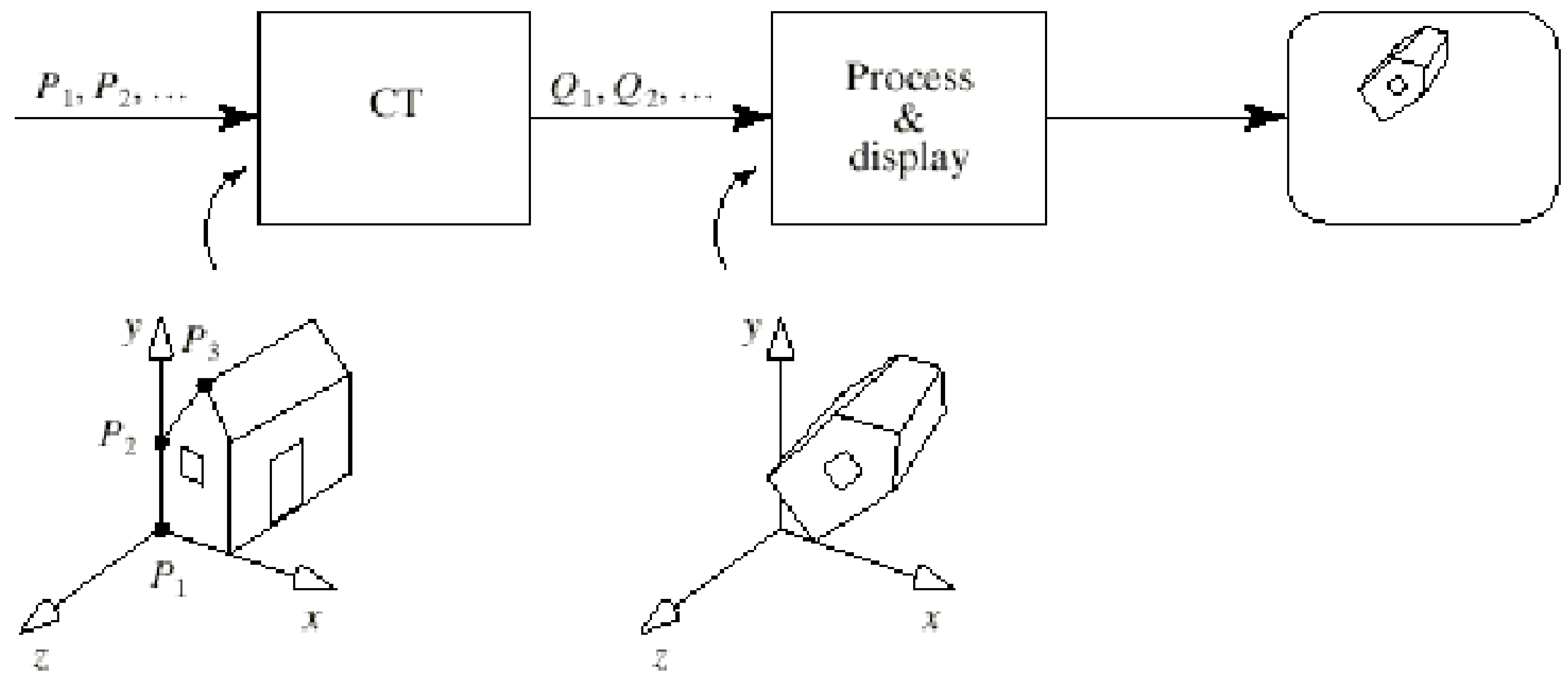
To go from homogeneous coordinates to ordinary coordinates,

if the object is a vector, its final coordinate is 0. So delete the 0
if the object is a point, its final coordinate is a 1. So delete the 1

Introduction to Affine Transforms

- Affine transforms are a fundamental computer graphics operation and are central to most graphics operations
- They also cause problems as it is difficult to get them right
- This is due to the difference between points and vectors and the fact that they do not transform in the same way.
- To overcome these problems we use homogeneous coordinates and an appropriate coordinate frame.

Transformation in the Graphics Pipeline



- Affine transforms fit into the graphics pipeline as shown
- First points are sent down the pipeline (p_1,p_2 ..)
- Next the points encounters the current transform (CT) which change them to a new position
- After this they are displayed in their new position

3D affine Transforms

- Any point can be expressed in the coordinate frame as $$\mathrm{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$
- Now using T() as a function to transform the points P to Q we use the matrix M as follows

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

3D affine Transforms

- Now using T() as a function to transform the points P to Q we use the matrix M as follows

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = M \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

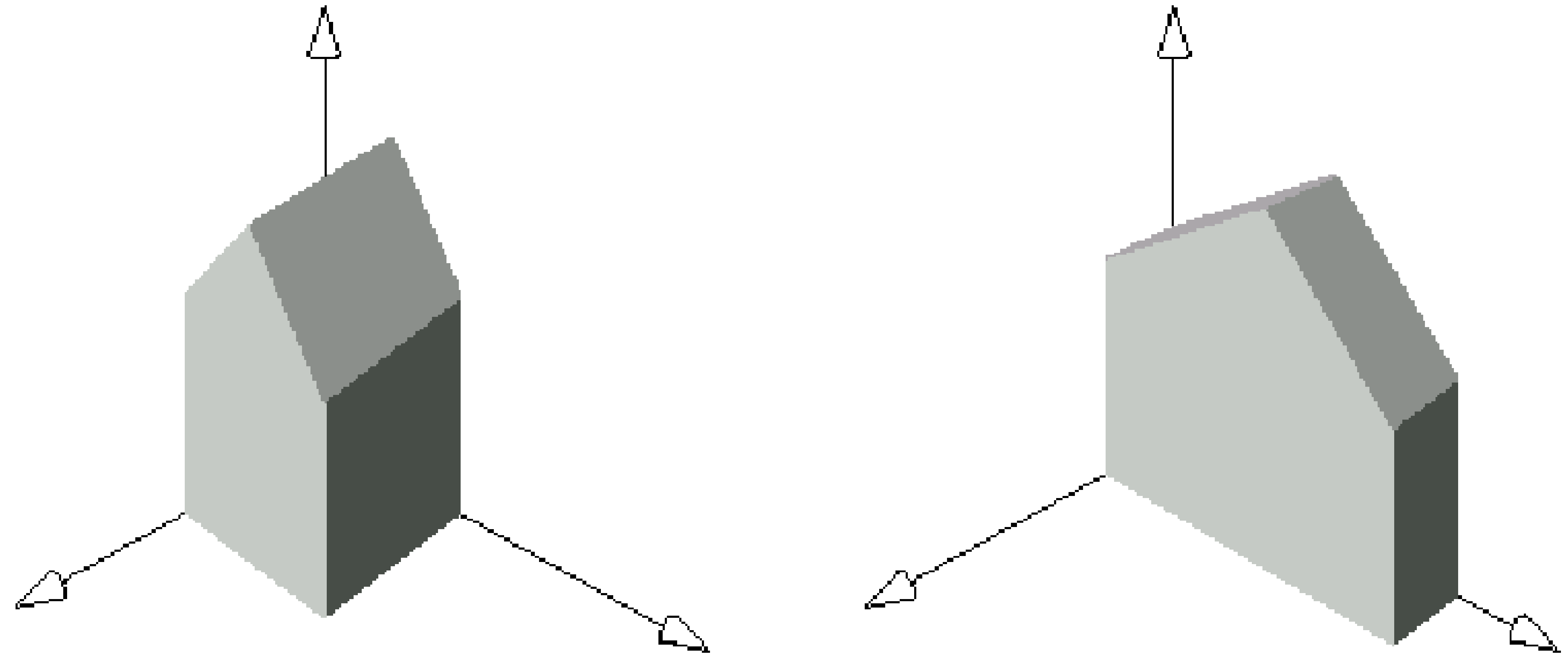
Translation

- For a pure translation the matrix has the following form

$$\begin{bmatrix} x^{'} \\ y^{'} \\ z^{'} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & m_{14} \\ 0 & 1 & 0 & m_{24} \\ 0 & 0 & 1 & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Where Q=MP is a shift in Q by the vector m=[m$_{14}, m_{24}, m_{34}$]

Scaling



- The algebra for 3D scaling is

$$x^{'} = s_x x$$
$$y^{'} = s_y y$$
$$z^{'} = s_z z$$

$$\begin{bmatrix} x^{'} \\ y^{'} \\ z^{'} \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- where S_x,S_y and S_z causes scaling of the corresponding coordinates.
- Scaling is about the origin as in the 2D as shown in the figure

Scaling about a point

- Given an arbitrary point (p_x, p_y, p_z) we can construct an algebra for scaling around a point thus

$$x^{'} = s_x(x - p_x) + p_x$$

$$y^{'} = s_y(x - p_y) + p_y$$

$$z^{'} = s_z(x - p_z) + p_z$$

- and in matrix form

$$\begin{bmatrix} x^{'} \\ y^{'} \\ z^{'} \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & p_x(1 - s_x) \\ 0 & s_y & 0 & p_y(1 - s_y) \\ 0 & 0 & s_z & p_z(1 - s_z) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Shearing

- 3D shears appear in greater variety compared to 2D versions
- The simplest shears are obtained by the identity matrix with one zero term replaced by some value as in

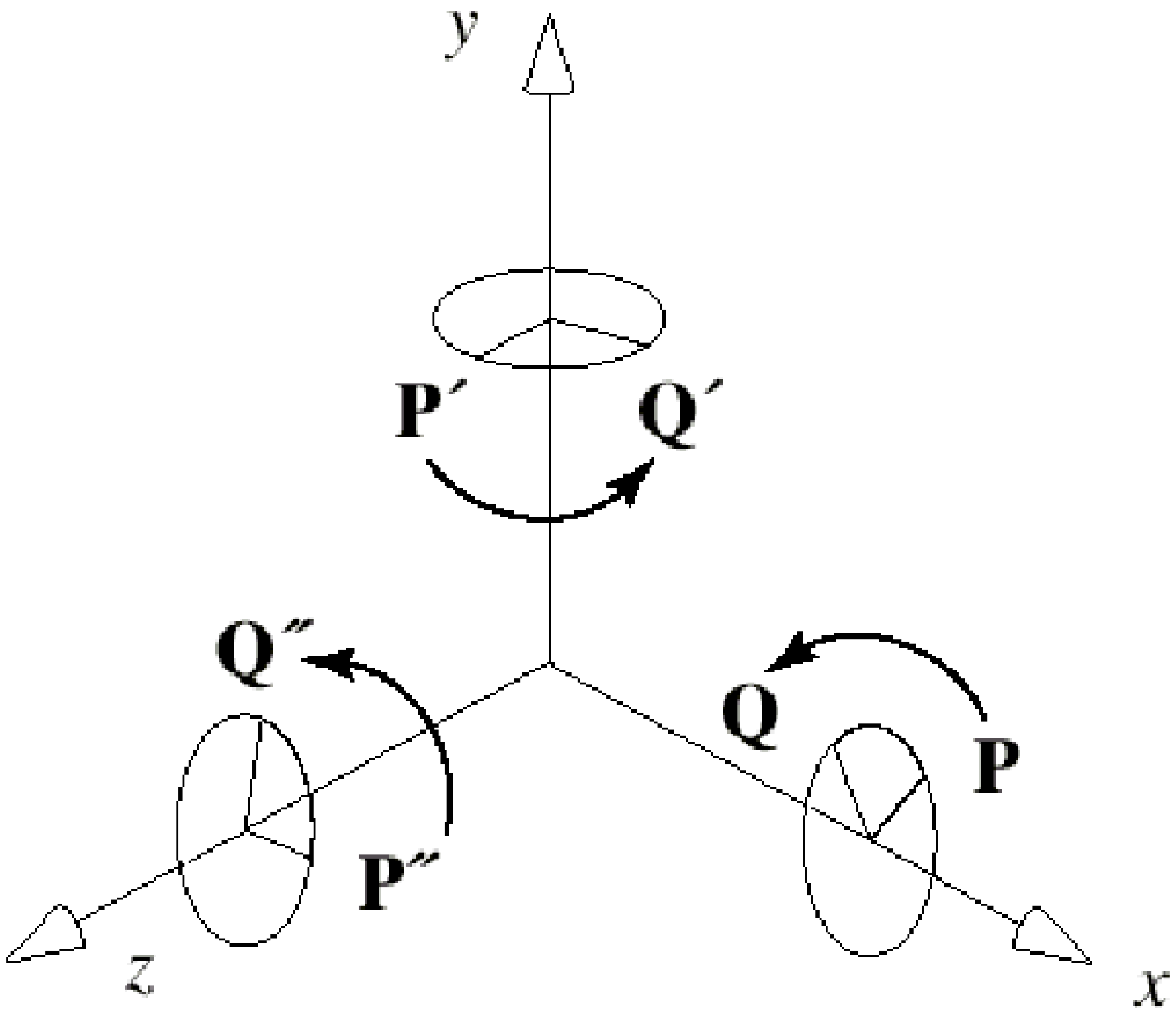
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ f & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- which produces $Q=(P_x, fP_x+P_y, P_z)$ this gives P_y offset by some amount proportional to P_x and the other components are unchanged.
- For interesting applications for shears see the paper by Barr in the accompanying papers.

Rotations

- Rotations in 3D are common in graphics (to rotate objects, cameras etc)
- In 3D we must specify an axis about which the rotations occurs, rather than just a single point
- One helpful approach is to decompose a rotation into a combination of simpler ones.

Elementary rotations about a coordinate axis



- The simplest rotation is a rotation about one of the coordinate axis.
- We call a rotation about the x-axis an "x-roll" about the y a "y-roll" and z a "z-roll"
- The figure shows the different "rolls" around the different axis

Elementary rotations about a coordinate axis

- The following three matrices represent transformations that rotate points through and angle β about a coordinate axis
- The angle is represented in radians

x-roll

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

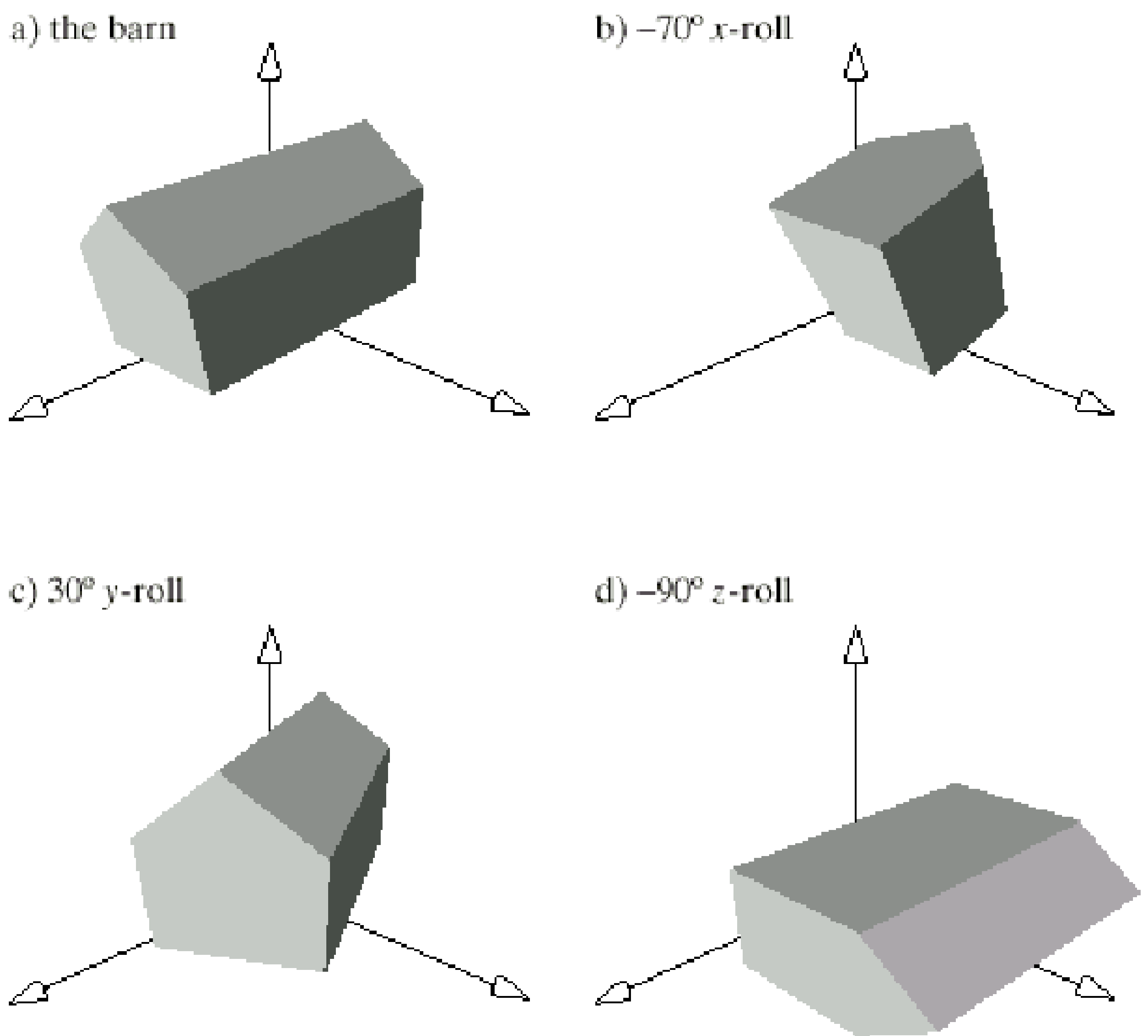
y-roll

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

z-roll

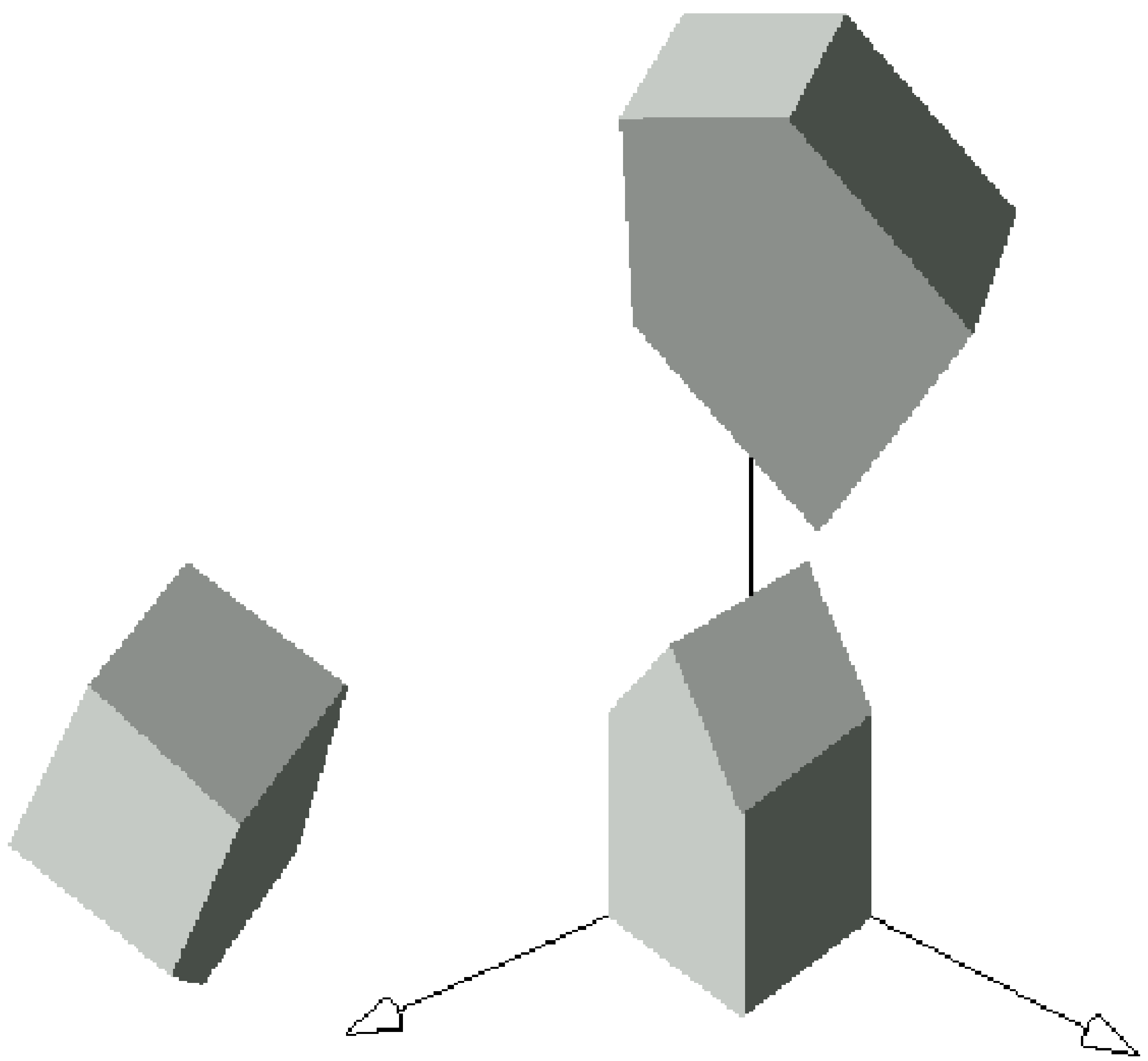
$$\mathrm{R}_z(\beta) = \begin{bmatrix} cos(\beta) & -sin(\beta) & 0 & 0 \\ sin(\beta) & cos(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Elementary rotations about a coordinate axis



- Note that 12 of the terms in each matrix are the zeros and ones of the identity matrix
- They occur in the row and column that correspond to the axis about which the rotation is being made
- These terms guarantee that the corresponding coordinate of the point being transformed will not be altered.
- An example of the different rolls are shown below

Composing 3D Affine Transforms



- Composing 3D affine transforms works the same way as in 2D
- We take the individual matrices for each rotation (M_1 and M_2) and then combine them by pre multiplying M_2 with M_1 to give $M=M_2M_1$
- Any number of affine transforms can be composed in this way, and a single matrix gives us the desired rotation.
- This is shown in the figure

Combining Rotations

- One of the most important distinctions between 2D and 3D transformations is the manner in which rotations combine
- In 2D two rotations R(β_1) and R(β_2) combine to produce R($\beta_1 + \beta_2$) and the order in which they combine make no difference
- In 3D the situation is more complex because rotations can be about different axes
- The order in which two rotations about different axes are performed does matter.
- 3D rotation matrices do not commute

3D rotations

- It is common to build a rotation in 3D by composing three elementary rotations
- An x-roll followed by a y-roll and then a z-roll
- Using the previous equations for the rolls we get

$$\mathrm{M}{=}\mathrm{R}_z(\beta_3)R_y(\beta_2)R_x(\beta_1)$$

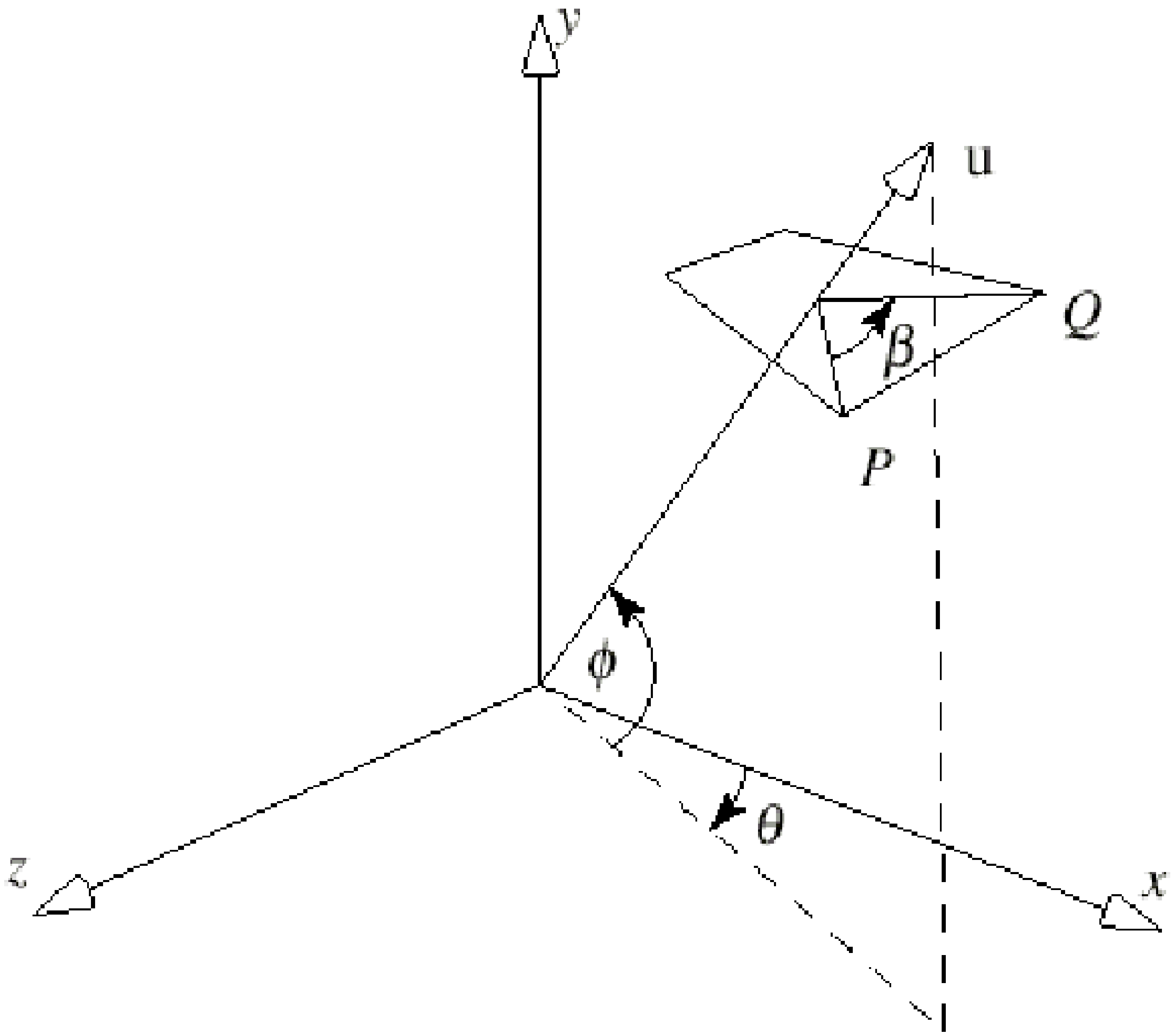
- In this context the angles β_1, β_2 and β_3 are called the **Euler** angles
- Euler's Theorem asserts that any 3D rotation can be obtained by three rolls about the x-, y- and z-axes
- so any rotation can be written as a particular product of 3 matrices for the appropriate choice of Euler angles.

Rotations about an Arbitrary Axis

- When using Euler angles we perform a sequence of x-,y- and z-rolls (rotations about coordinate axis)
- But it is much easier to work with rotations if we have a way to rotate about an axis that points in an arbitrary direction.
- Euler's Theorem states that every rotation can be represented as

> **Euler's Theorem** : Any rotation (or sequence of rotations) about a point is equivalent to a single rotation about some axis through that point

Euler



- The figure shows an axis represented by a vector **u** and an arbitrary point P that is to be rotated through angle β about **u** to produce the point Q
- Because **u** can have any direction it is difficult to find a single matrix that represents the rotation
- However we can do the rotation in one of two ways

The classic way

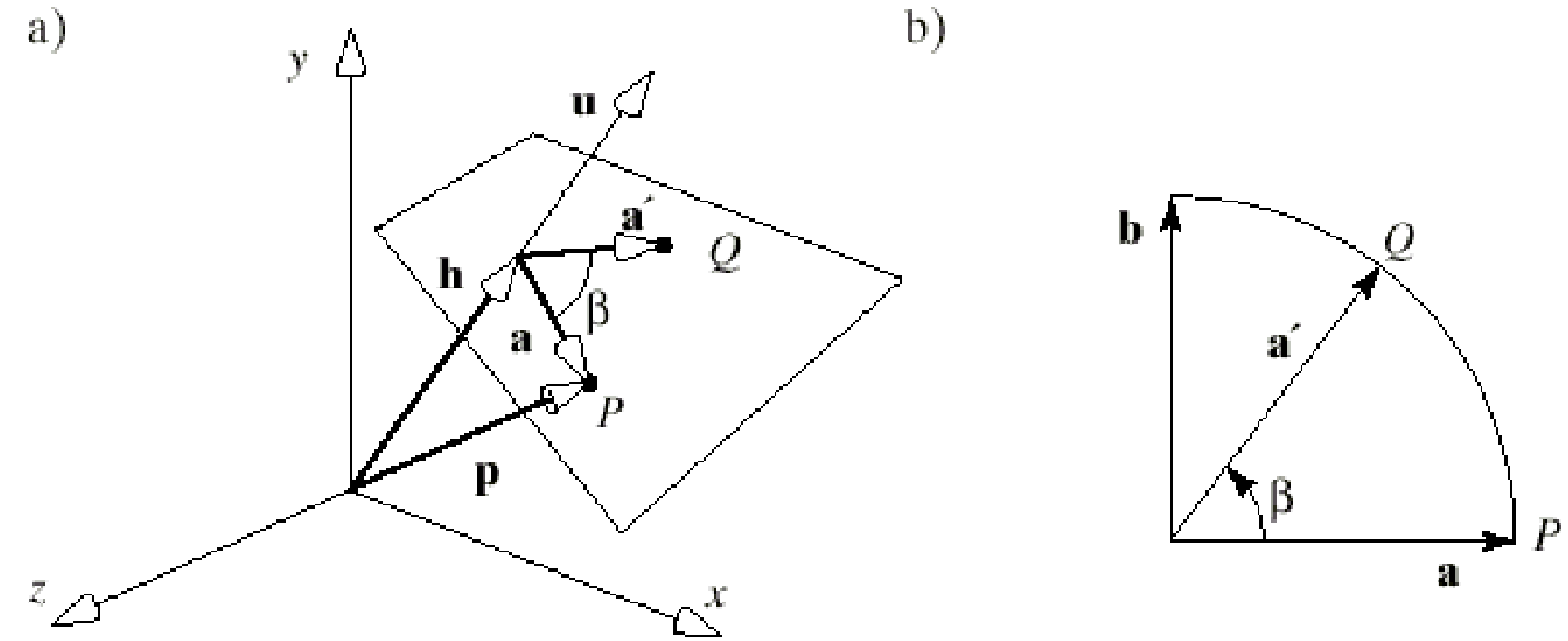
- We decompose the rotation into a sequence of known steps

  1. Perform two rotations so that **u** becomes aligned with the x-axis
  2. Do a z-roll through the angle β
  3. Undo the two alignment rotations to restore **u** to its original direction

- This method is similar to a rotation about a point in 2D The first step moves the point into the correct 2D location, we then do the rotation and finally replace into the original position (in the other axis no affected)

- The resultant equation is $\mathrm{R}_u(\beta) = R_y(-\theta)R_z(\phi)R_x(\beta)R_z(-\phi)R_y(\theta)$

The constructive way



- The above figure shows the axis of rotation **u** and the point P that we wish to rotate by β to make point Q
- As seen in figure b) the point Q is the linear combinations of the two vectors **a** and **b**
- Now we use cross products and dot products of the vectors to produce a final matrix as shown below

$$\mathrm{R}_u(\beta) = \begin{bmatrix} c+(1-c)u_x^2 & (1-c)u_y u_x - s u_z & (1-c)u_z u_x + s u_y & 0 \\ (1-c)u_x u_y + s u_z & c+(1-c)u_y^2 & (1-c)u_z u_y - s u_x & 0 \\ (1-c)u_x u_z - s u_y & (1-c)u_y u_z + s u_x & c+(1-c)u_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

References

- Basic Algebra and Geometry. Ann Hirst and David Singerman. Prentice Hall 2001
- Computer Graphics With OpenGL, F.S. Hill jr, Prentice Hall
- "Essential Mathematics for Computer Graphics fast" John VinceSpringer-Verlag London
- "Geometry for Computer Graphics: Formulae, Examples and Proofs" John Vince Springer-Verlag London 2004
- "Engineering Mathematics", K. A. Stroud, Macmillan 3rd Edition 1987
- http://en.wikipedia.org/wiki/Affine_geometry