Geometrical Transformations (for use with Angel Chapter 3)

CSCI 4631/5631

Translation

We've got a point P(x,y) and we want to translate it by (dx, dy)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix}$$

$$P' = P + T$$

Scaling

We've got a point P(x,y) and we want to stretch it by s_x along the x axis and by s_y along the y axis.

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

$$P' = S \cdot P$$

If $s_x = s_y$ this is called "uniform scaling" but if they are different, it's called "differential scaling"

Rotation

Points can be rotated about the origin through the angle θ.
Mathematically:

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

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

$$P' = R \cdot P$$

Positive angles are measured counterclockwise from x toward y

Homogenous Coordinates

We've got a point P(x,y) and we're going to represent it as the triplet P(x,y,W)

Why? $P' = P + T$

$P' = S \cdot P$

$P' = R \cdot P$

Homogenous Coordinates

In homogenous coordinates, points are the same if their x,y,W values are multiples of one another

$$\begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix}$$

At least one must be nonzero: (0,0,0) is not allowed

When W is nonzero: (x,y,W) divide through by W, and represent as (x/W , y/W, 1). This is known as “homogenizing”

Homogenous Coordinates Transformation

Now we can represent a translation within the context of a 3 x 3 transformation matrix:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = T(d_x, d_y) \bullet P$$

Translation Transformations

Translation matrices are additive:

$$\begin{bmatrix} 1 & 0 & dx1 \\ 0 & 1 & dy1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & dx2 \\ 0 & 1 & dy2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx1 + dx2 \\ 0 & 1 & dy1 + dy2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T(d_{x1},d_{y1}) \bullet T(d_{x2},d_{y2}) = T(d_{x1}+d_{x2} , d_{y1}+d_{y2})$$

Scaling Transformations

Scaling matrices are multiplicitive:

$$\begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x1} \bullet s_{x2} & 0 & 0 \\ 0 & s_{y1} \bullet s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S(x1,y1) \bullet S(x2,y2) = S(x1 \bullet x2,\ y1 \bullet y2)$$

Rotational Transformations

Rotation matrices are additive:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

too much to type!!!, but

$R(\theta) \bullet R(\phi) = R(\theta+\phi)$

Properties of rotation matrix: look at upper 2x2 submatrix rows as vectors.

Each is a unit vector
Each is perpendicular to the other
1st and 2nd vectors will be rotated by $R(\theta)$ to lie on positive x and y axes

Rotational Transformations

Rotation matrices are additive:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Same thing is true if we think of the column vectors.
A matrix with these properties is called “special orthogonal”

What does this mean? It preserves angles and lengths
also known as a “**rigid body transformation**”

Affine Transformations

Any arbitrary sequence of rotation and translation is going to be special orthogonal, and therefore, a rigid body transformation.

Once we add scaling operations to the mix, we are no longer special orthogonal, we are “affine”. Affine transformations preserve parallel lines, but angles and lengths may change.

$$\begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear transformations are an example of this. The matrix to the left shears along the x axis with a nonzero and b zero. It shears along the y axis with b nonzero and a zero.

Rotations and Scaling

Rotations: Since rotations happen about the origin, in order to rotate about some arbitrary point, you must first translate the primitive such that your rotation point is at the origin, rotate, then translate back

$$T(x_1, y_1) \bullet R(q) \bullet T(-x1, -y1)$$

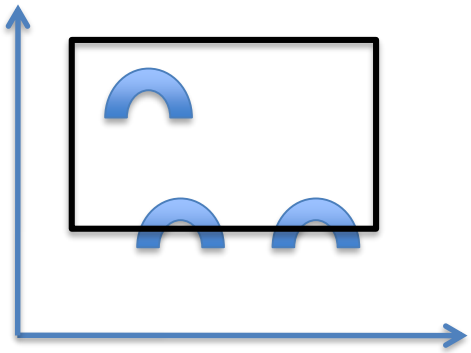
Also, since scaling operations affect the position of a primitive, you often want to scale around the center of the primitive, which also involves translating the center to the origin, scaling, and translating back again

$$T(x_1, y_1) \bullet S(s_x, s_y) \bullet T(-x1, -y1)$$

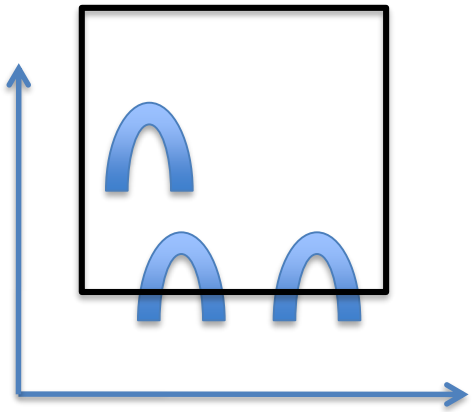
Window-To-Viewport Transformation

It is often the case that you need to transform between coordinate systems

Example: your primitives may live in the “world” coordinate system), and you need to transform your “window” to the world into the same *place* and *shape* as your viewport (i.e. the canvas you’re drawing onto)



world coordinates



screen coordinates

Efficiency

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

A generalized composition of R, T, and S matrices will look like the matrix to the left.

M • P takes 9 multiplies and 6 adds.

but, we can use the fact that the last row is fixed, and reduce this to 4 mults and 4 adds

Further, in interactive graphics, when we're rotating, we can take advantage of the fact that successive draws can be thought of as moving through small θ, so we can approximate cos θ as 1, and we get to 2 mults and 2 adds. This can lead to error buildup over successive calculations, however, so you need to be careful

3D Transformations (finally!)

$$\begin{bmatrix} x \\ y \\ z \\ W \end{bmatrix} = \begin{bmatrix} x/W \\ y/W \\ z/W \\ 1 \end{bmatrix}$$

We've been representing 2D points as 3-space vectors, so now we're going to represent 3D points as 4-space vectors (also known as quaternions)

Same properties as before, at least one element must be nonzero. We're working in a right-handed coordinate system where positive rotations are counterclockwise when looking from a positive axis toward the origin.

Translation in 3D

$$\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Easy – as an extension from the 2D case

Scaling in 3D

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Again easy – as an extension from the 2D case

Rotation About Z

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Again easy – as an extension from the 2D case

Rotation About X

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

looks like this…. and finally……

Rotation About Y

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

looks like this

Generalized Transformation Matrix in 3D

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here's a generalized representation of a composite rotation, scaling, and translation matrix.

For each of the rotation matrices mentioned in previous slides, the upper 3 x 3 matrix is special orthogonal, as are any combinations of these rotations, and combinations of rotations and translations.

Shear Transformation in 3D

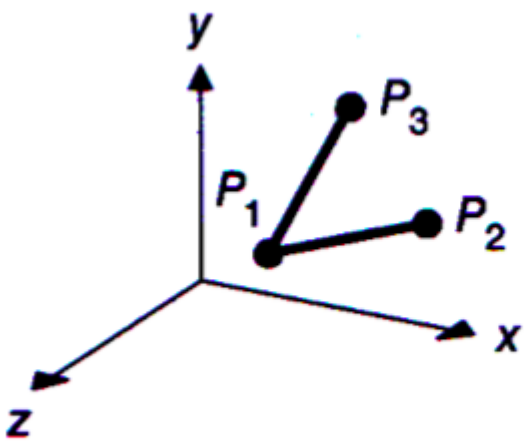
$$\begin{bmatrix} 1 & 0 & sh_x & 0 \\ 0 & 1 & sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here's a generalized representation of a composite rotation, scaling, and translation matrix.

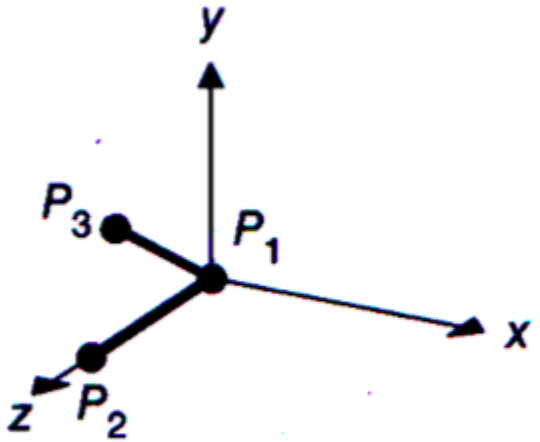
For each of the rotation matrices mentioned in previous slides, the upper 3 x 3 matrix is special orthogonal, as are any combinations of these rotations, and combinations of rotations and translations.

Composition of 3D Transforms

Let's suppose we want to transform P_1 P_2 and P_2 P_3 line segments from their original orientation (a) to that shown in (b)



(a) Initial position



(b) Final position

Composition of 3D Transforms

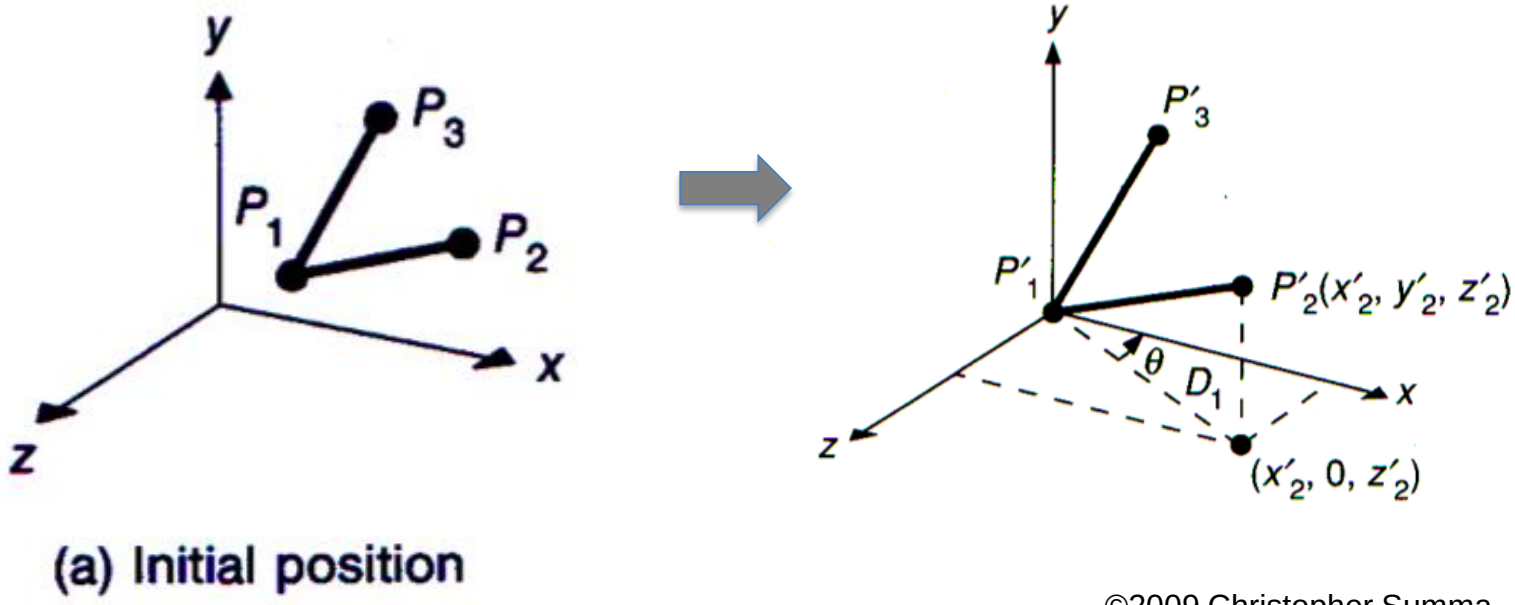
Let's suppose we want to transform P_1 P_2 and P_2 P_3 line segments from their original orientation (a) to that shown in (b)

Translate P1 to origin : this one is easy : $T(-x_1, -y_1, -z_1)$

Rotate about y such that P_1P_2 lies in (y,z) plane

Rotate about x such that P_1P_2 lies on z axis

Rotate about z such that P_1P_3 lies in (x,z) plane

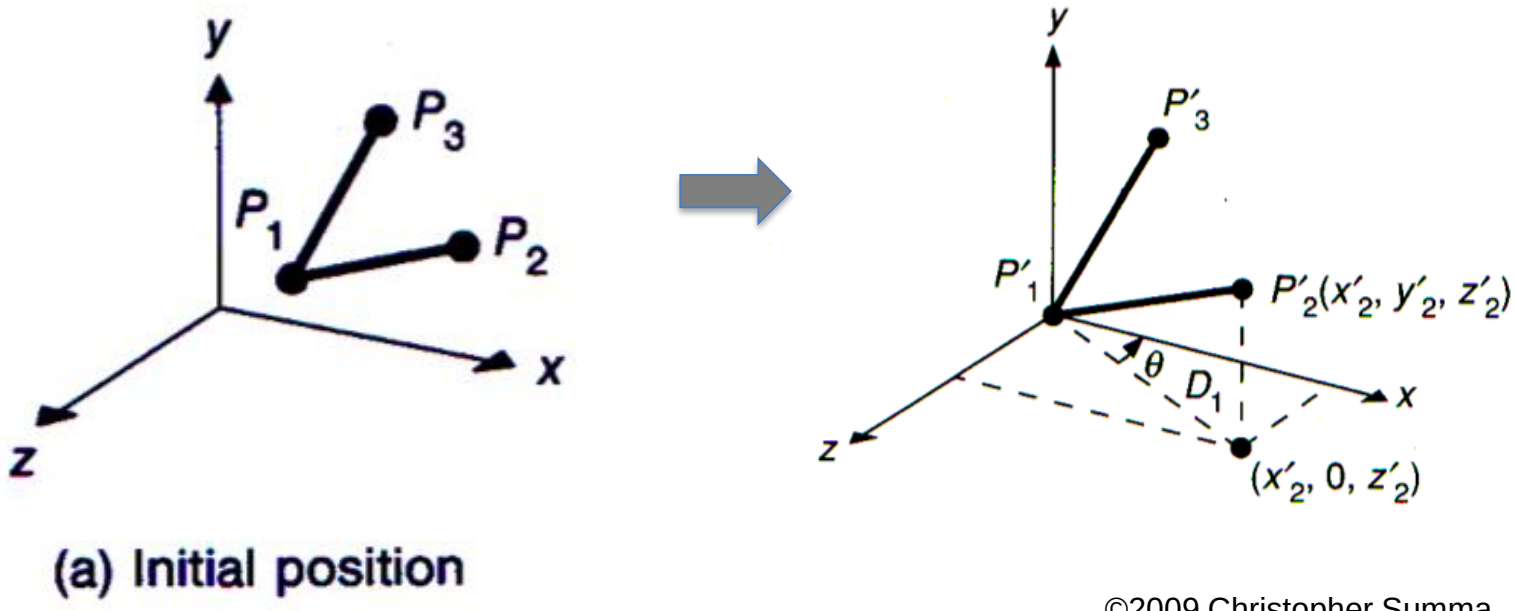


(a) Initial position

Composition of 3D Transforms

Let's suppose we want to transform P_1 P_2 and P_2 P_3 line segments from their original orientation (a) to that shown in (b)

Translate P_1 to origin : this one is easy : $T(-x_1, -y_1, -z_1)$



Composition of 3D Transforms

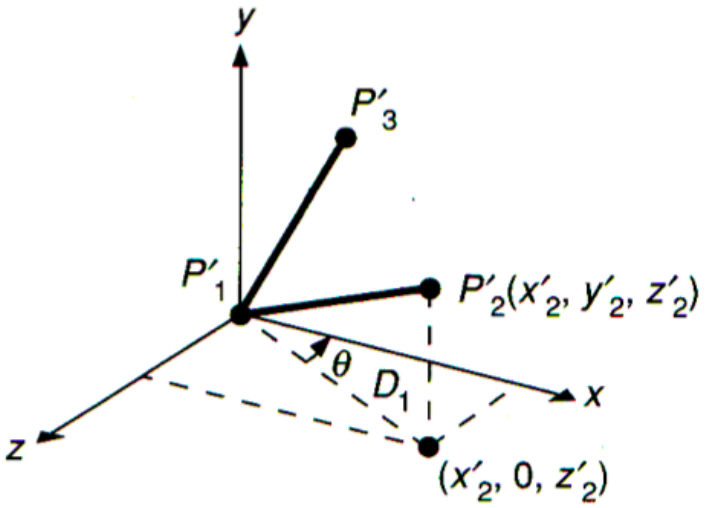
Let's suppose we want to transform P_1 P_2 and P_2 P_3 line segments from their original orientation (a) to that shown in (b)

Rotate about y such that P_1P_2 lies in (y,z) plane

rotate by $-(90-\theta) = (\theta-90)$

$\cos(\theta-90) = \sin\theta = z_2'/D_1 = (z_2-z_1)/D_1$

similarly, solve for $\sin(\theta-90)$, and substitute into rotation about y matrix



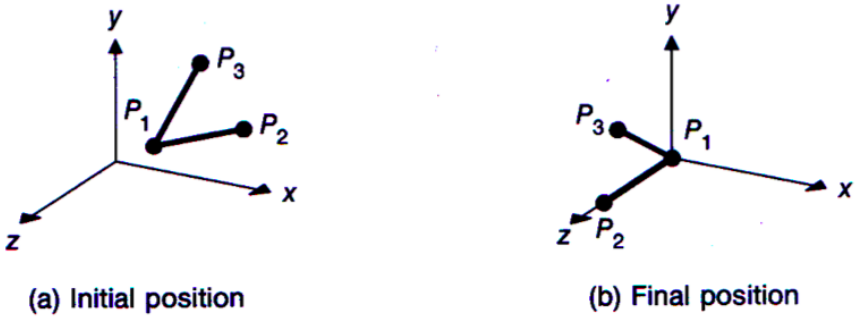
Composition of 3D Transforms

Let's suppose we want to transform P_1 P_2 and P_2 P_3 line segments from their original orientation (a) to that shown in (b)

Rotate about x such that P_1P_2 lies on z axis

Rotate about z such that P_1P_3 lies in (x,z) plane

These last 2 transforms use the same basic ideas, and rely on the fact that the rotations preserve lengths and angles between elements

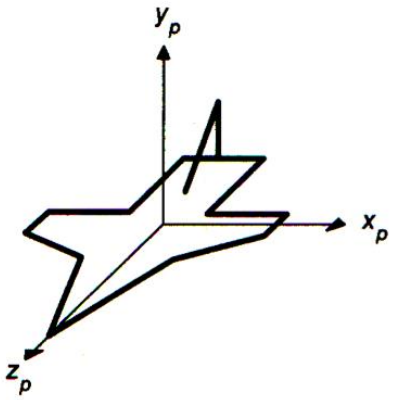


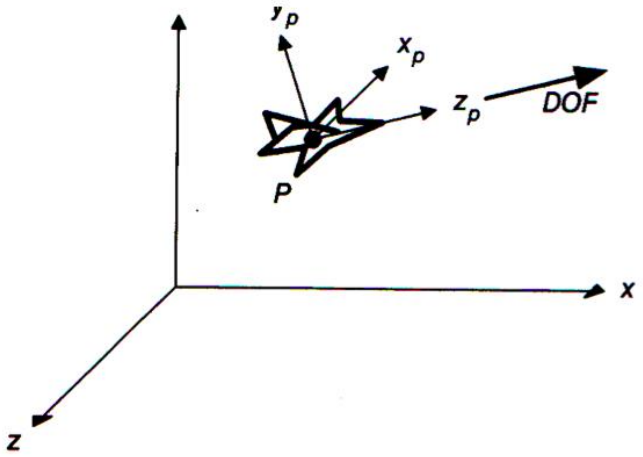
(a) Initial position

(b) Final position

Composition of 3D Transforms

Let's suppose we want to transform a local coordinate system to some other one, for example we've got a plane that we want to transform such that it is pointed in some direction DOF, and is not banked. We need a series of rotations to accomplish this.





Composition using properties of Orthogonal matrices

$$\begin{bmatrix} r_{1x} & r_{2x} & r_{3x} & 0 \\ r_{1y} & r_{2y} & r_{3y} & 0 \\ r_{1z} & r_{2z} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Unit row vectors of R rotate into the principal axes

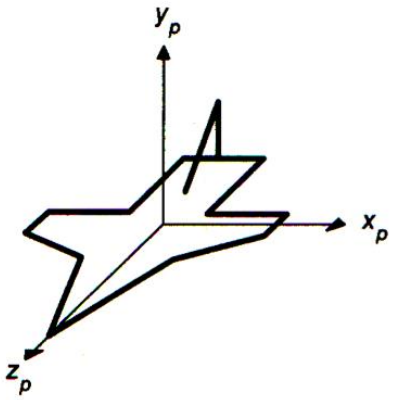
The rows of R represent the coordinates in the original space of unit vectors along the coordinate axes of the rotated space.

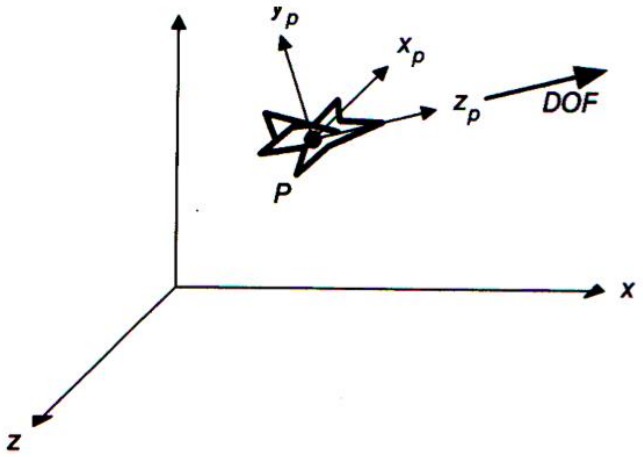
The columns of R represent the coordinates in the rotated space of unit vectors along the axes of the original space.

Determine in what direction each of x_p, y_p and z_p axes is heading, normalize them, and use them as column vectors in the rotation matrix.

Composition of 3D Transforms

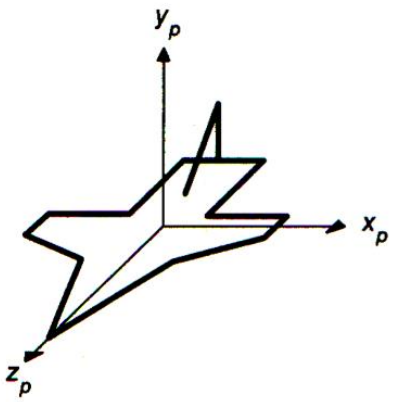
Let's suppose we want to transform a local coordinate system to some other one, for example we've got a plane that we want to transform such that it is pointed in some direction DOF, and is not banked. We need a series of rotations to accomplish this.

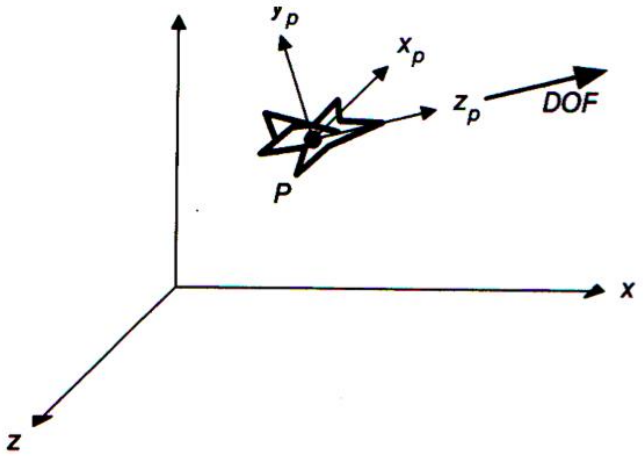




Composition of 3D Transforms

What we need is the z_p axis transformed in the DOF direction, with the x_p axis transformed into a horizontal vector perpendicular to DOF. Hmmm… y x DOF is in the right direction! Finally y_p should point in the direction that is orthogonal to x_p and z_p, or (DOF x (y x DOF))





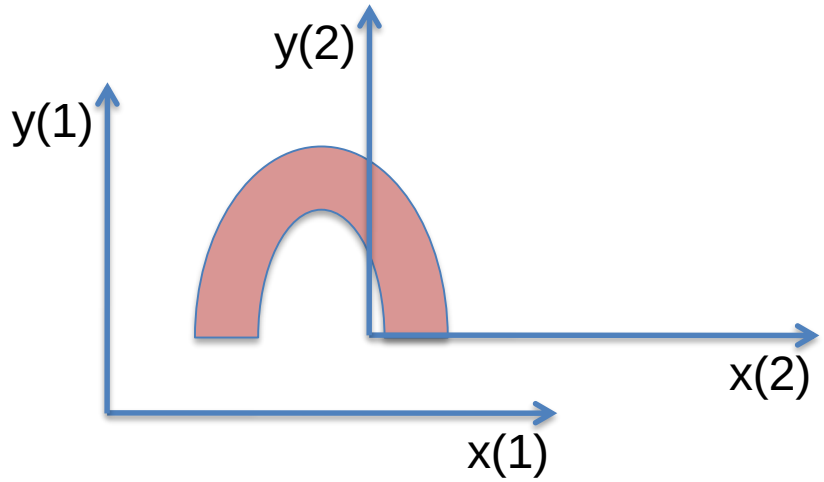
Composition of 3D Transforms

$$\begin{bmatrix} & & & 0 \\ |y \times DOF| & |DOF \times (y \times DOF)| & |DOF| & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Replacing the column vectors in the upper 3 x 3 matrix gives the proper rotation matrix

Transformation as a change in coordinate systems

It is often useful to think of your primitives as existing in their own coordinate system, and what you need to to is “map” them onto a different coordinate system (the screen perhaps) via transformations



Transformation as a change in coordinate systems

It is often useful to think of your primitives as existing in their own coordinate system, and what you need to to is “map” them onto a different coordinate system (the screen perhaps) via transformations

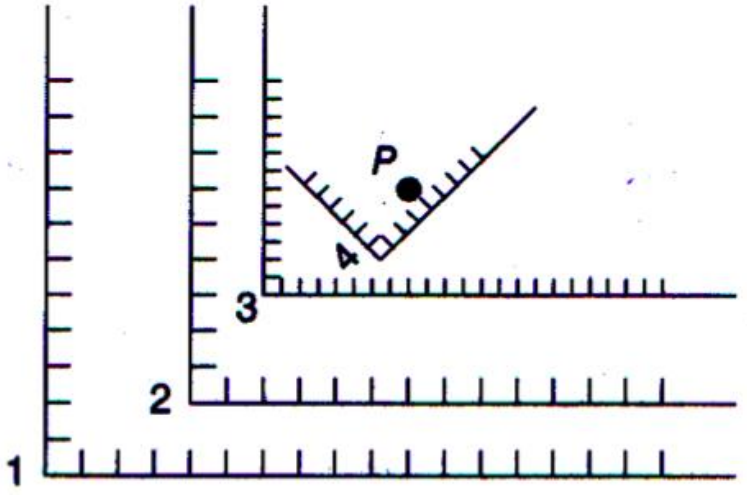
$P^{(i)} = M_{i \leftarrow j} \bullet P^{(j)}$ and $P^{(j)} = M_{j \leftarrow k} \bullet P^{(k)}$

$$P^{(i)} = M_{i \leftarrow k} \bullet P^{(k)}$$

$M_{1 \leftarrow 2} = T(4,2)$

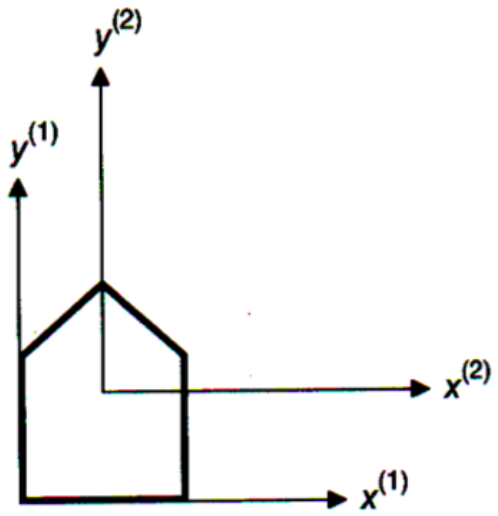
$M_{2 \leftarrow 3} = T(2,3) \bullet S(0.5, 0.5)$

$M_{3 \leftarrow 4} = T(6.7, 1.8) \bullet R(-45^{\circ})$



Transformation as a change in coordinate systems

It is often useful to think of your primitives as existing in their own coordinate system, and what you need to to is “map” them onto a different coordinate system (the screen perhaps) via transformations



$$M_{1\leftarrow 2} = T(x_1, y_1)$$

$$M_{2\leftarrow 1} = T(x_1, y_1)^{-1} = T(-x_1, -y_1)$$

Transformation as a change in coordinate systems

Suppose $Q^{(j)}$ is some transformation in coordinate system j, and we want to find some transformation $Q^{(i)}$ in coordinate system i.

Imagine applying $Q^{(j)}$ to points $P^{(j)}$ in system j. How to find the $Q^{(i)}$ in system i that will produce the same set of transformed points?

Mathematically: $Q^{(i)} \bullet P^{(i)} = M_{i \leftarrow j} \bullet Q^{(j)} \bullet P^{(j)}$

Substituting and Simplifying: $Q^{(i)} = M_{i \leftarrow j} \bullet Q^{(j)} \bullet M_{i \leftarrow j}^{-1}$

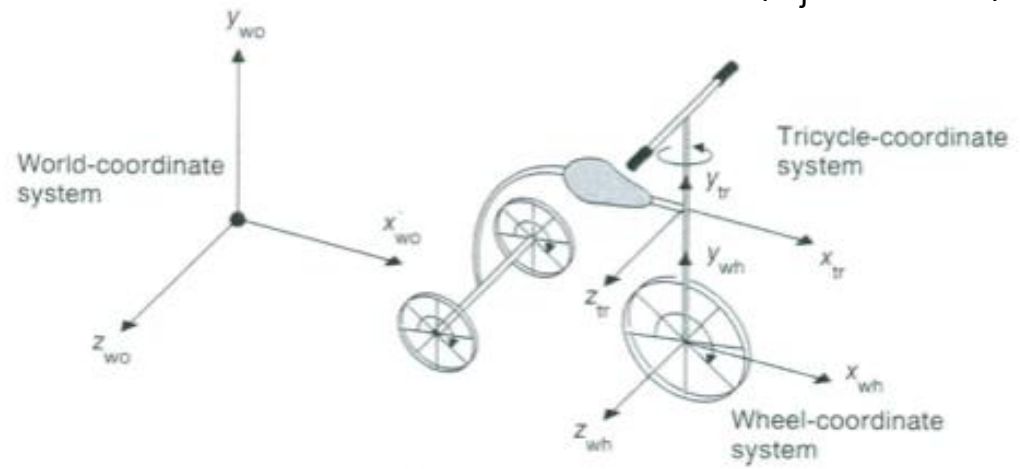


Fig. 5.26 A stylized tricycle with three coordinate systems.

Transformation as a change in coordinate systems

Moving coordinate systems – front wheel moves about z_{wh} – how does whole bike move in the world coordinate system?

Image a point on the wheel $P^{(wh)}$

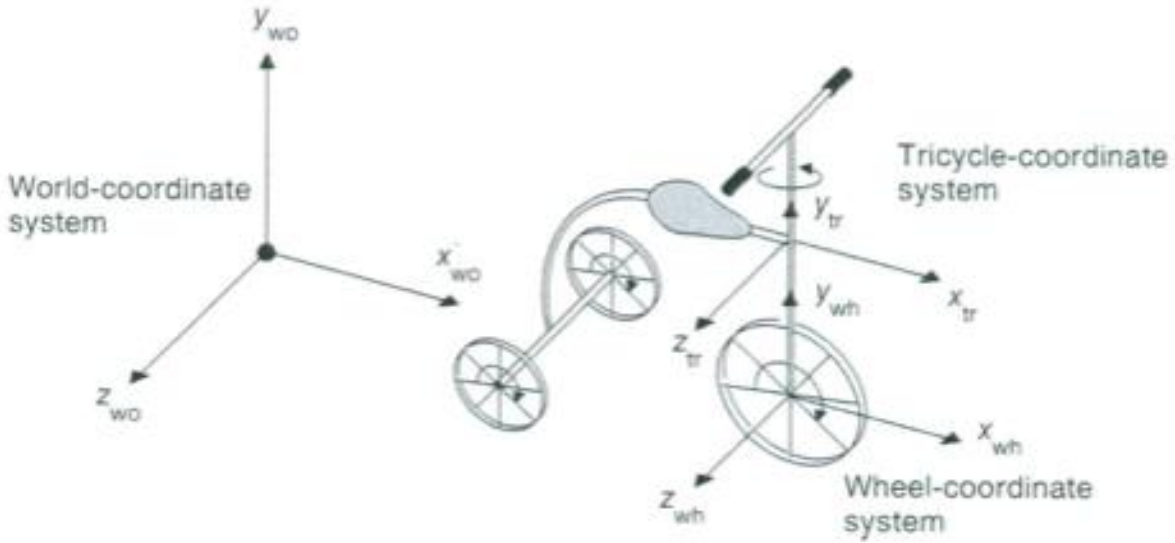


Fig. 5.26 A stylized tricycle with three coordinate systems.

Transformation as a change in coordinate systems

As wheel rotates through angle α about z_{wh} – a point on the wheel $P^{(wh)}$ moves through a distance αr (r is radius of wheel) and tricycle moves forward a distance αr

$P'^{(wh)} = T(\alpha r,0,0) \bullet R_z(\alpha) \bullet P^{(wh)}$ $\quad$ $P'^{(wh')} = R_z(\alpha) \bullet P^{(wh')}$

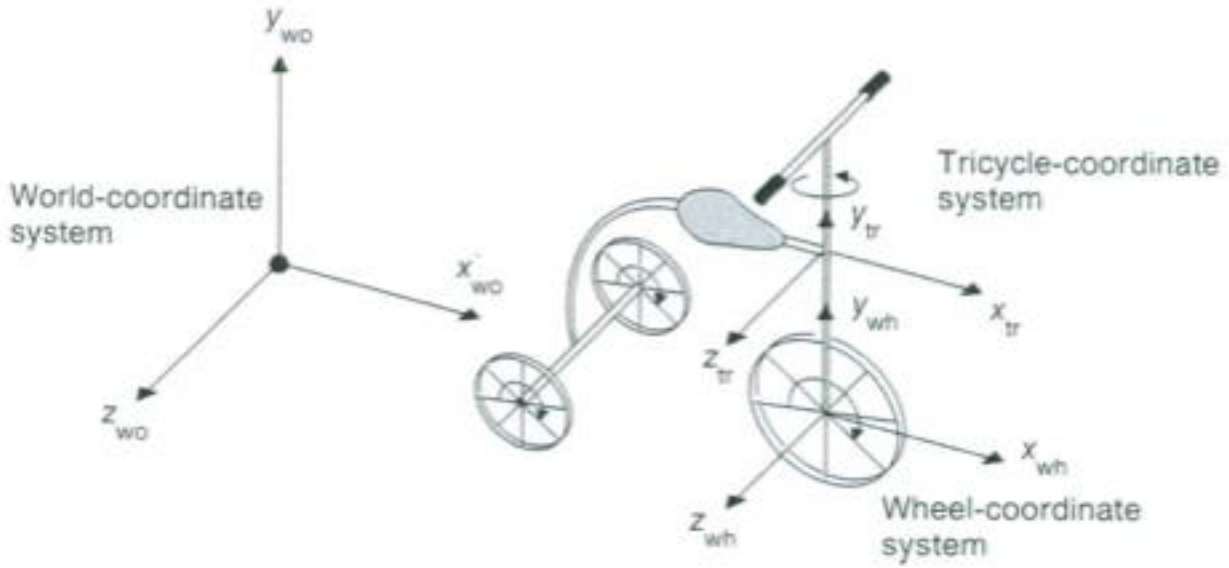


Fig. 5.26 A stylized tricycle with three coordinate systems.

Transformation as a change in coordinate systems

As wheel rotates through angle α about z_{wh} – a point on the wheel $P^{(wh)}$ moves through a distance αr (r is radius of wheel) and tricycle moves forward a distance αr

$$P'^{(wh)} = T(\alpha r,0,0) \bullet R_z(\alpha) \bullet P^{(wh)} \qquad P'^{(wh')} = R_z(\alpha) \bullet P^{(wh')}$$

Transforming to world coordinate system:

$$P^{(wo)} = M_{wo \leftarrow wh} \bullet P^{(wh)} = M_{wo \leftarrow tr} \bullet M_{tr \leftarrow wh} \bullet P^{(wh)}$$

$$P'^{(wo)} = M_{wo \leftarrow wh} \bullet P'^{(wh)} = M_{wo \leftarrow wh} \bullet T(\alpha r,0,0) \bullet R_z(\alpha) \bullet P^{(wh)}$$

$$P'^{(wo)} = M_{wo \leftarrow wh'} \bullet P'^{(wh')} = (M_{wo \leftarrow wh} \bullet M_{wh \leftarrow wh'}) \bullet (R_z(\alpha) \bullet P^{(wh)})$$