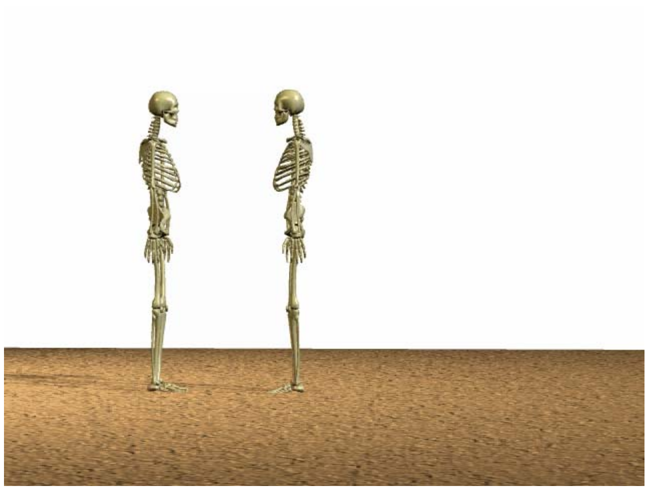
Affine Transformations in 3D



Affine Transformations in 3D

General form

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

Or:

$$Q = \mathbf{M}P$$

General Form

Rotation / Scaling / Shearing

Translation

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

Elementary 3D Affine Transformations

Translation

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

Scaling Around the Origin

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

Shear Around the Origin

Along x-axis

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

3D Rotation

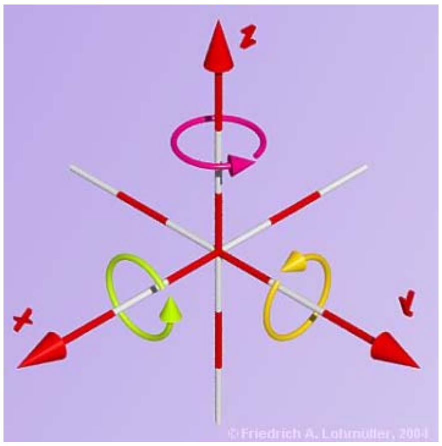
Various representations possible

Decomposition into axis rotations

- x-roll, y-roll, z-roll

Counterclockwise positive angle assumption

Three Axes to Rotate Around



Reminder: 2D Rotation

$$
\begin{aligned}
Q_x &= \cos\theta\, P_x - \sin\theta\, P_y \\
Q_y &= \sin\theta\, P_x + \cos\theta\, P_y
\end{aligned}
$$

In matrix form:

$$
\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}
$$

Or:

$$
Q = \mathbf{R}(\theta) P
$$

Z-Roll

$$
\boxed{\begin{aligned}
Q_x &= \cos\theta\, P_x - \sin\theta\, P_y \\
Q_y &= \sin\theta\, P_x + \cos\theta\, P_y
\end{aligned}}
$$

$$
\boxed{Q_z = P_z}
$$

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

X-Roll

Cyclic indexing

$$x \rightarrow \boxed{y \rightarrow z \rightarrow x} \rightarrow y$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \\ x \\ y \end{bmatrix}$$

$$Q_y = \cos\theta \, P_y - \sin\theta \, P_z$$
$$Q_z = \sin\theta \, P_y + \cos\theta \, P_z$$
$$Q_x = P_x$$

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

Y-Roll

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \\ x \\ y \end{bmatrix}$$

$$Q_z = \cos\theta \, P_z - \sin\theta \, P_x$$
$$Q_x = \sin\theta \, P_z + \cos\theta \, P_x$$
$$Q_y = P_y$$

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

Inversion of Transformations

Translation: $\mathbf{T}^{-1}(t_x,t_y,t_z) = \mathbf{T}(-t_x,-t_y,-t_z)$

Rotation: $\mathbf{R}^{-1}_{axis}(\theta) = \mathbf{R}_{axis}(-\theta)$

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

Shearing: $\mathbf{Sh}^{-1}(a) = \mathbf{Sh}(-a)$

Inverse of Rotations

Pure rotation only, no scaling or shear

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

$$\mathrm{M}^{-1} = \mathrm{M}^{T}$$

Since the rotation matrix **M** *is an orthonormal matrix*

Composition of 3D Affine Transformations

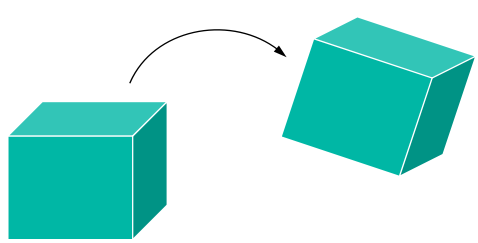
The composition of affine transformations is an affine transformation

Any 3D affine transformation can be performed as a series of elementary affine transformations

Rigid Body Transformations

Translations and rotations

Preserves lines, angles and distances



Composite 3D Rotation About the Origin

$$\mathbf{R}(\theta_1, \theta_2, \theta_3) = \mathbf{R}_z(\theta_3)\mathbf{R}_y(\theta_2)\mathbf{R}_x(\theta_1)$$

- *This is known as the “Euler angle” representation of 3D rotations*
- *The order of the rotation matrices is important !!*
- *Note: The Euler angle representation suffers from singularities*

Guerrilla CG Tutorial 13: The 3D Rotation Problem

Gimbal Lock

$$\mathbf{R}(\theta_1, \theta_2, \theta_3) = \mathbf{R}_z(\theta_3)\mathbf{R}_y(\theta_2)\mathbf{R}_x(\theta_1)$$

$$= \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 & 0 \\ 0 & \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Let $\theta_2 = 90^\circ$ $\quad(\sin(90^\circ) = 0,\ \cos(90^\circ) = 1)$:

$$\mathbf{R}(\theta_1, 90^\circ, \theta_3) = \mathbf{R}_z(\theta_3)\mathbf{R}_y(90^\circ)\mathbf{R}_x(\theta_1)$$

$$= \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 & 0 \\ 0 & \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \cos\theta_3\sin\theta_1 - \sin\theta_3\cos\theta_1 & \cos\theta_3\cos\theta_1 + \sin\theta_3\sin\theta_1 & 0 \\ 0 & \cos\theta_3\cos\theta_1 + \sin\theta_3\sin\theta_1 & -\cos\theta_3\sin\theta_1 + \sin\theta_3\cos\theta_1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Loss of a Rotational Degree of Freedom

$$\mathbf{R}(\theta_1, 90^\circ, \theta_3) = \begin{bmatrix} 0 & \cos\theta_3\sin\theta_1 - \sin\theta_3\cos\theta_1 & \cos\theta_3\cos\theta_1 + \sin\theta_3\sin\theta_1 & 0 \\ 0 & \cos\theta_3\cos\theta_1 + \sin\theta_3\sin\theta_1 & -\cos\theta_3\sin\theta_1 + \sin\theta_3\cos\theta_1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \sin(\theta_1 - \theta_3) & \cos(\theta_1 - \theta_3) & 0 \\ 0 & \cos(\theta_1 - \theta_3) & -\sin(\theta_1 - \theta_3) & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

where $\theta = \theta_1 - \theta_3$

Thus, the two remaining rotational degrees of freedom, θ_1 and θ_3, have collapsed into a single rotational degree of freedom θ, which is the difference of the two rotational angles

Guerrilla CG Tutorial:
14 – Euler (Gimbal Lock) Explained

There are Alternatives

It is often convenient to use other representations of 3D rotations that do not suffer from Gimbal Lock

- Advanced concepts
  - *Quaternions*
  - *Exponential Maps*

Rotation Around an Arbitrary Axis

Euler's theorem:

Any rotation or sequence of rotations around a point is equivalent to a single rotation around an axis that passes through the point

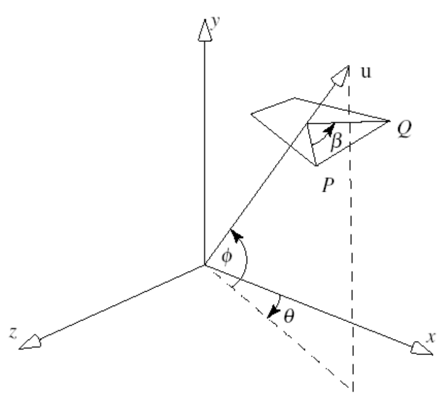
What does the matrix look like?

Rotation Around an Arbitrary Axis

Vector (axis): **u**

Rotation angle: β

Point: P



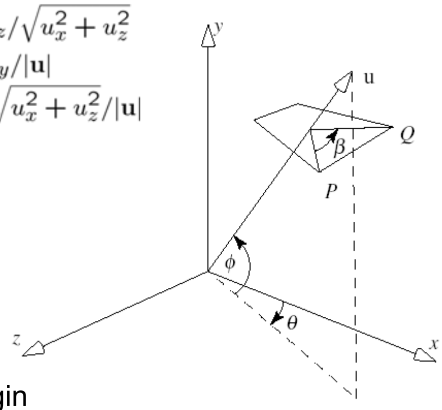
Method:

1. *Two rotations to align* **u** *with x-axis*
2. *Do x-roll by* β
3. *Undo the alignment*

Derivation

1. $\mathbf{R}_z(-\phi)\ \mathbf{R}_y(\theta)$
2. $\mathbf{R}_x(\beta)$
3. $\mathbf{R}_y(-\theta)\ \mathbf{R}_z(\phi)$

$$\cos(\theta) = u_x/\sqrt{u_x^2 + u_z^2}$$
$$\sin(\theta) = u_z/\sqrt{u_x^2 + u_z^2}$$
$$\sin(\phi) = u_y/|\mathbf{u}|$$
$$\cos(\phi) = \sqrt{u_x^2 + u_z^2}/|\mathbf{u}|$$



All together: $\mathbf{R}_\mathbf{u}(\beta)$ =

$$\boxed{\mathbf{R}_y(-\theta)\ \mathbf{R}_z(\phi)\ \mathbf{R}_x(\beta)\ \mathbf{R}_z(-\phi)\ \mathbf{R}_y(\theta)}$$

We should add translation too if the axis is not through the origin

Properties of Affine Transformations

1. ***Affine transformations are composed of elementary ones***
2. ***Preservation of affine combinations of points***
3. ***Preservation of lines and planes***
4. ***Preservation of parallelism of lines and planes***
5. ***Relative ratios are preserved***

Affine Combinations of Points

$$W = a_1 P_1 + a_2 P_2$$
$$T(W) = T(a_1 P_1 + a_2 P_2) = a_1 T(P_1) + a_2 T(P_2)$$

Proof: from linearity of matrix multiplication

$$\mathbf{M}W = \mathbf{M}(a_1 P_1 + a_2 P_2) = a_1 \mathbf{M}P_1 + a_2 \mathbf{M}P_2$$

Preservations of Lines and Planes

$$L(t) = (1-t)P_1 + tP_2$$
$$T(L) = (1-t)T(P_1) + tT(P_2) = (1-t)\mathbf{M}P_1 + t\mathbf{M}P_2$$

$$Pl(s,t) = (1-s-t)P_1 + tP_2 + sP_3$$
$$\begin{aligned} T(Pl) &= (1-s-t)T(P_1) + tT(P_2) + sT(P_3) \\ &= (1-s-t)\mathbf{M}P_1 + t\mathbf{M}P_2 + s\mathbf{M}P_3 \end{aligned}$$

Preservation of Parallelism

$$L(t) = P + t\mathbf{u}$$

$$\mathbf{M}L = \mathbf{M}(P + t\mathbf{u}) = \mathbf{M}P + \mathbf{M}(t\mathbf{u}) \rightarrow$$
$$\mathbf{M}L = \mathbf{M}P + t(\mathbf{M}\mathbf{u})$$

$\mathbf{Mu}$ independent of P

Similarly for planes

Transformations of Coordinate Systems

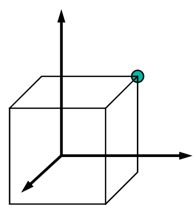
Coordinate systems consist of basis vectors and an origin (point)

They can be represented as affine matrices

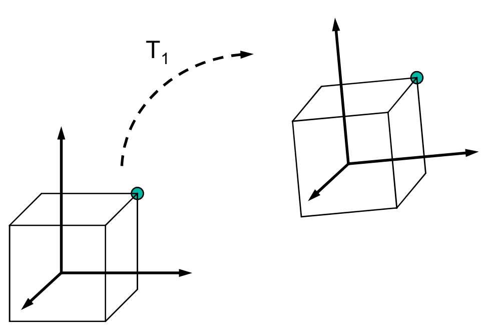
Therefore, we can transform them just like points and vectors

This provides an alternative way to think of transformations—
as changes of coordinate systems

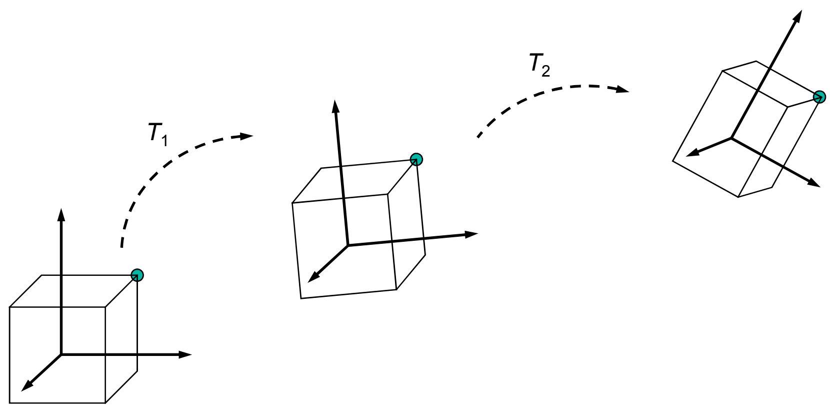
Transforming a Point by Transforming Coordinate Systems



Transforming a Point by Transforming Coordinate Systems

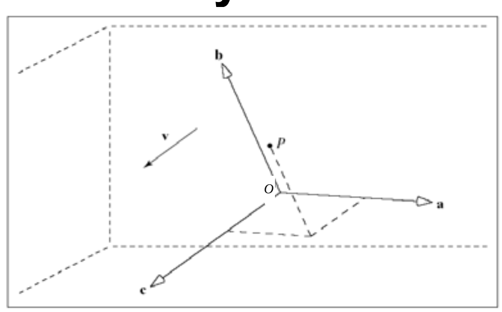


Transforming a Point by Transforming Coordinate Systems



Reminder: Coordinate Systems

Coordinate system:
O, **a**, **b**, **c**,



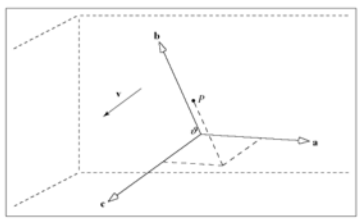
$$\mathbf{v} = [v_1 \;\; v_2 \;\; v_3]^T \rightarrow \mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c}$$

$$P = [p_1 \;\; p_2 \;\; p_3]^T \rightarrow P - O = p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c}$$
$$P = O + p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c}$$

Reminder: Coordinate Systems

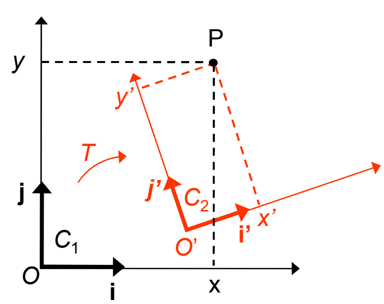
$$\mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c} \rightarrow \mathbf{v} = [\mathbf{a}\ \ \mathbf{b}\ \ \mathbf{c}\ \ O]\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}$$

$$P = O + p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c} \rightarrow P = [\mathbf{a}\ \ \mathbf{b}\ \ \mathbf{c}\ \ O]\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$



Transforming C_1 into C_2

What is the relationship between P in C_2 and P in C_1 if $T(C_1) \mapsto C_2$?



$$C_1 : P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$C_2 : P = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

$$O' = T(O),$$
$$\mathbf{i}' = T(\mathbf{i}),$$
$$\mathbf{j}' = T(\mathbf{j}),$$
$$\mathbf{k}' = T(\mathbf{k})$$

Derivation

By definition P is the linear combination of vectors $\mathbf{i}', \mathbf{j}', \mathbf{k}'$ and point O'.

$$P = x'\mathbf{i}' + y'\mathbf{j}' + z'\mathbf{k}' + O'$$

In coordinate system C_1:

$$P_{C_1} = x'\mathbf{i}'_{C_1} + y'\mathbf{j}'_{C_1} + z'\mathbf{k}'_{C_1} + O'_{C_1}$$

Derivation

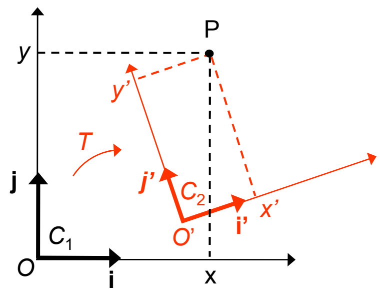
$$P_{C_1} = x'\mathbf{i}'_{C_1} + y'\mathbf{j}'_{C_1} + z'\mathbf{k}'_{C_1} + O'_{C_1}$$

We know that $[\mathbf{i}'_{C_1}, \mathbf{j}'_{C_1}, \mathbf{k}'_{C_1}, O'_{C_1}] = T([\mathbf{i}, \mathbf{j}, \mathbf{k}, O])$

$$\begin{aligned}
P_{C_1} &= x'T(\mathbf{i}) + y'T(\mathbf{j}) + z'T(\mathbf{k}) + T(O) \\
&= x'\mathrm{M}\mathbf{i} + y'\mathrm{M}\mathbf{j} + z'\mathrm{M}\mathbf{k} + \mathrm{M}O \\
&= x'\mathrm{M}\begin{bmatrix}1\\0\\0\\0\end{bmatrix} + y'\mathrm{M}\begin{bmatrix}0\\1\\0\\0\end{bmatrix} + z'\mathrm{M}\begin{bmatrix}0\\0\\1\\0\end{bmatrix} + \mathrm{M}\begin{bmatrix}0\\0\\0\\1\end{bmatrix} \\
&= \mathrm{M}\begin{bmatrix}x'\\0\\0\\0\end{bmatrix} + \mathrm{M}\begin{bmatrix}0\\y'\\0\\0\end{bmatrix} + \mathrm{M}\begin{bmatrix}0\\0\\z'\\0\end{bmatrix} + \mathrm{M}\begin{bmatrix}0\\0\\0\\1\end{bmatrix} \\
&= \mathrm{M}\left(\begin{bmatrix}x'\\0\\0\\0\end{bmatrix} + \begin{bmatrix}0\\y'\\0\\0\end{bmatrix} + \begin{bmatrix}0\\0\\z'\\0\end{bmatrix} + \begin{bmatrix}0\\0\\0\\1\end{bmatrix}\right) = \mathrm{M}\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}
\end{aligned}$$

P in C_1 *vs* P in C_2

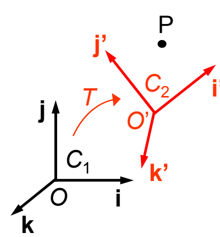
$C_1 \underset{T}{\mapsto} C_2$



$$P_{C_1} = \mathrm{M} P_{C_2}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathrm{M} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

Transformations as a Change of Basis



So, we know that

$$P_{C_1} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathrm{M} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M} P_{C_2}$$

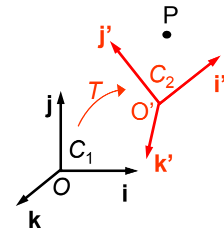
Now, what is M with respect to the basis vectors?

$$P_{C_2} = x'\mathbf{i}'_{C_2} + y'\mathbf{j}'_{C_2} + z'\mathbf{k}'_{C_2} + O'_{C_2} = x' \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + y' \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + z' \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$P_{C_1} = x'\mathbf{i}'_{C_1} + y'\mathbf{j}'_{C_1} + z'\mathbf{k}'_{C_1} + O'_{C_1} = x' \begin{bmatrix} i'_x \\ i'_y \\ i'_z \\ 0 \end{bmatrix} + y' \begin{bmatrix} j'_x \\ j'_y \\ j'_z \\ 0 \end{bmatrix} + z' \begin{bmatrix} k'_x \\ k'_y \\ k'_z \\ 0 \end{bmatrix} + \begin{bmatrix} O'_x \\ O'_y \\ O'_z \\ 1 \end{bmatrix}$$

$$P_{C_1} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} i'_x & j'_x & k'_x & O'_x \\ i'_y & j'_y & k'_y & O'_y \\ i'_z & j'_z & k'_z & O'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M} P_{C_2}$$

Transformations as a Change of Basis



$$P_{C_1} = \mathrm{M} P_{C_2}$$

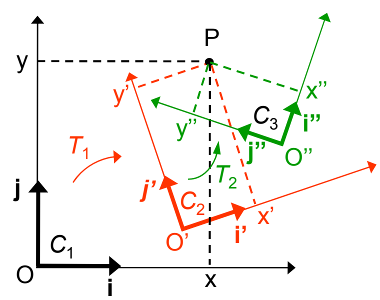
$$P_{C_1} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} i'_x & j'_x & k'_x & O'_x \\ i'_y & j'_y & k'_y & O'_y \\ i'_z & j'_z & k'_z & O'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M} P_{C_2}$$

That is:

We can view transformations as a change of coordinate system

Successive Transformations of the Coordinate System

$$C_1 \underset{T_1}{\mapsto} C_2 \underset{T_2}{\mapsto} C_3$$



Working backwards:

$$P_{C_2} = \mathrm{M}_2 P_{C_3} \rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M}_2 \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix}$$

$$P_{C_1} = \mathrm{M}_1 P_{C_2} \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathrm{M}_1 \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M}_1 \mathrm{M}_2 \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix}$$

Rule of Thumb

Transforming a point P:

Transformations: T_1, T_2, T_3

Matrix: $\mathbf{M} = \mathbf{M}_3\ \mathbf{M}_2\ \mathbf{M}_1$

Point is transformed by $\mathbf{M}P$

Each transformation happens with respect to the **same** coordinate system

Transforming a coordinate system:

Transformations: T_1, T_2, T_3 (not generally the same as the ones above)

Matrix: $\mathbf{M} = \mathbf{M}_1\ \mathbf{M}_2\ \mathbf{M}_3$

A point has coordinates $\mathbf{M}P$ in the original coordinate system

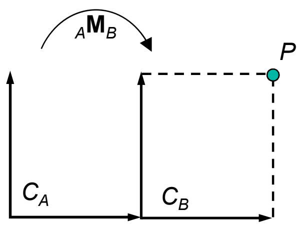
Each transformation happens with respect to **previous** coordinate system

Rule of Thumb

To find the transformation matrix that transforms P from C_A coordinates to C_B coordinates, we find a sequence of transformations that align C_B to C_A, accumulating matrices from left to right

Explanation of This Rule

Transformation **M**: ${}_A\mathbf{M}_B$



If we think coordinate systems, **M** takes C_A from the left and produces C_B on the right:

$$C_A \; {}_A\overrightarrow{\mathbf{M}}_B = C_B$$

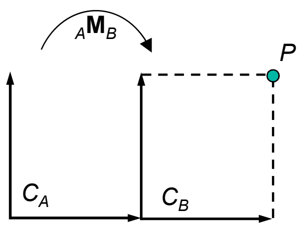
After this transformation we "talk" in C_B coordinates (right side).

If we think points, then we go the other way; **M** takes P_B on the right and produces the P_A coordinates on the left:

$$P_A = {}_A\overleftarrow{\mathbf{M}}_B \, P_B$$

Explanation of This Rule

Transformation **M**: ${}_A\mathbf{M}_B$



Consider this simple example, where to produce C_B we translate C_A by +1 along the x axis:

$P_A = (2,1)$ $\qquad$ $P_B = (1,1)$

If we move C_A by +1 in *x* to transform it into C_B then the *x* coordinate of *P* with respect to the new system is reduced by 1 (C_B is closer to P than C_A by 1).

So, if we want to transform the coordinates of *P* from C_B to C_A we need to add 1 in *x*. Exactly what we need to do to transform C_A to C_B.

Remember

Transformations are represented by affine matrices

$$\begin{array}{c} \begin{array}{cc} \text{Rotate/Scale/Shear} & \text{Translate} \end{array} \\ \mathrm{M} = \begin{bmatrix} \boxed{\begin{matrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{matrix}} & \boxed{\begin{matrix} m_{14} \\ m_{24} \\ m_{34} \end{matrix}} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix} \end{array}$$

Coordinate systems are represented by affine matrices too

$$\begin{array}{cccc} \text{Basis Vector 1} & \text{Basis Vector 2} & \text{Basis Vector 3} & \textit{Origin Point} \end{array}$$

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

Transforming Coordinate Systems vs Transforming Points: An Example

Let point $P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ wrt the *canonical* coordinate system $\mathbf{I} = [\mathbf{i} \quad \mathbf{j} \quad \mathbf{k} \quad O]$,

where

$$\mathbf{i} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{j} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{k} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad O = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix};$$

i.e, the point is represented as

$$\begin{aligned} P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} &= x\mathbf{i} + y\mathbf{j} + z\mathbf{k} + O \\ &= [\mathbf{i} \;\; \mathbf{j} \;\; \mathbf{k} \;\; O]\, P \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{I}P = P \end{aligned}$$

(Note: The canonical coordinate system is represented by the affine identity matrix $\mathbf{I}$)

Transforming Coordinate Systems vs Transforming Points: An Example

Now, let's transform point P to point P' by applying an affine transformation T_1 represented by the affine matrix $\mathbf{M}_1$; i.e.,

$$P' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathbf{M}_1 P \quad \Longrightarrow \quad P = \mathbf{M}_1^{-1} P'$$

Next, let's apply T_2 represented by $\mathbf{M}_2$ to transform P' to P''; i.e.,

$$P'' = \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix} = \mathbf{M}_2 P' \quad \Longrightarrow \quad P' = \mathbf{M}_2^{-1} P''$$

So,

$$\begin{aligned} P'' &= \mathbf{M}_2 P' \\ &= \mathbf{M}_2 \mathbf{M}_1 P \end{aligned}$$

However, from the coordinate system point of view:

$$\begin{aligned} P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} &= \mathbf{I}P \\ &= \mathbf{I}\mathbf{M}_1^{-1} P' \\ &= \mathbf{I}\mathbf{M}_1^{-1}\mathbf{M}_2^{-1} P'' \\ &= \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & O \end{bmatrix} \mathbf{M}_1^{-1}\mathbf{M}_2^{-1} P'' \end{aligned}$$

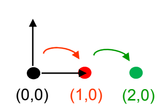
Transforming Coordinate Systems vs Transforming Points: An Example

For example, let $\mathbf{M}_1$ be a translation by $+1$ and let $\mathbf{M}_2$ be a scaling by $+2$, both in the $\mathbf{i}$ axis:

$$\boxed{\mathbf{M}_1} = \begin{bmatrix} 1 & 0 & 0 & +1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boxed{\mathbf{M}_2} = \begin{bmatrix} +2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

So,

$$\mathbf{M}_2\mathbf{M}_1 P = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 2x+2 \\ y \\ z \\ 1 \end{bmatrix} = P''$$

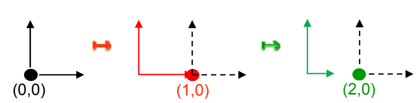


Now,

$$\mathbf{M}_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{M}_2^{-1} = \begin{bmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

So,

$$\begin{aligned} \mathbf{I}\mathbf{M}_1^{-1}\mathbf{M}_2^{-1} P'' &= \left((\mathbf{I}\mathbf{M}_1^{-1})\mathbf{M}_2^{-1} \right) P'' \\ &= \left(\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{M}_2^{-1} \right) P'' \\ &= \begin{bmatrix} 1/2 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2x+2 \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = P \end{aligned}$$



Transforming Coordinate Systems vs Transforming Points

In general, if we transform point P to Q by applying a series of n transformations, $\mathbf{M}_1$, followed by $\mathbf{M}_2$, ..., followed by $\mathbf{M}_n$; i.e.,

$$Q = \mathbf{M}_n \ldots \mathbf{M}_2 \mathbf{M}_1 P$$

then,

$$P = \mathbf{M}_1^{-1} \mathbf{M}_2^{-1} \ldots \mathbf{M}_n^{-1} Q.$$

This can be interpreted as the canonical coordinate system, represented by $\mathbf{I}$, being transformed by $\mathbf{M}_1^{-1}$, then being transformed by $\mathbf{M}_2^{-1}$, ..., then being transformed by $\mathbf{M}_n^{-1}$. On the LHS of the above equation, the coordinates of point P are relative to the canonical coordinate system $\mathbf{I}$, whereas the coordinates of point Q on the RHS are relative to the coordinate system represented by $\mathbf{M} = \mathbf{I}\ \mathbf{M}_1^{-1} \mathbf{M}_2^{-1} \ldots \mathbf{M}_n^{-1}$.