전산 선박 설계

Part 4. 3차원 형상변환을 이용한 로봇의 정/역기구학

2006.9.

서울대학교 조선해양공학과
이규열

Advanced
Ship
Design
Automation
Laboratory

목 차

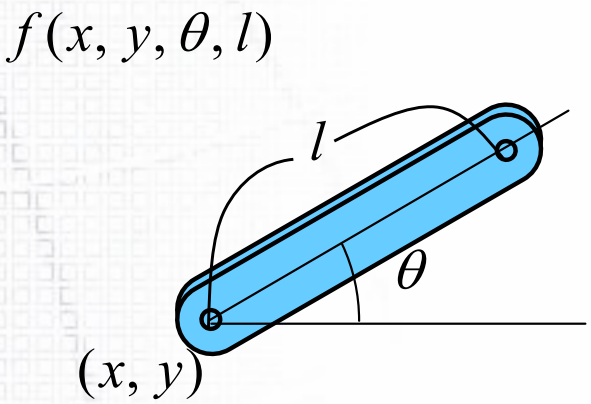
Ch 1. 자유도(degree of freedom)

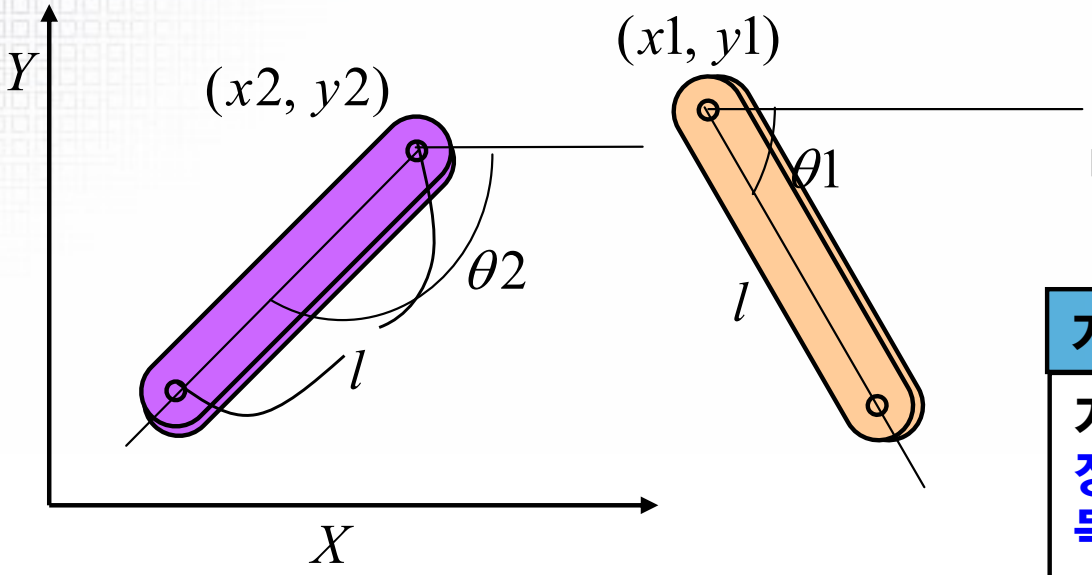
1.1 하나의 강체의 자유도
1.2 두 개 이상의 강체로 이루어진 물체의 자유도
1.3 연습 문제

Advanced
Ship
Design
Automation
Laboratory

1.1 하나의 강체의 자유도(1)

자유도가 3인 평면 위의 강체





■ **평면상의 물체의 위치와 자세를 표현하기 위해서는**
1. 물체의 형상정보(l)와
2. 기준점의 위치(x, y)
3. 기준방위로부터의 회전(θ)
이 필요하다.

■ **그러나 형상정보(l)는 고정된 상수 값(물체가 강체이므로)이므로 변수라 보지 않는다.**

■ **결국, 왼쪽의 강체의 위치와 자세는 3개의 변수(x, y, θ)로 정의 됨**

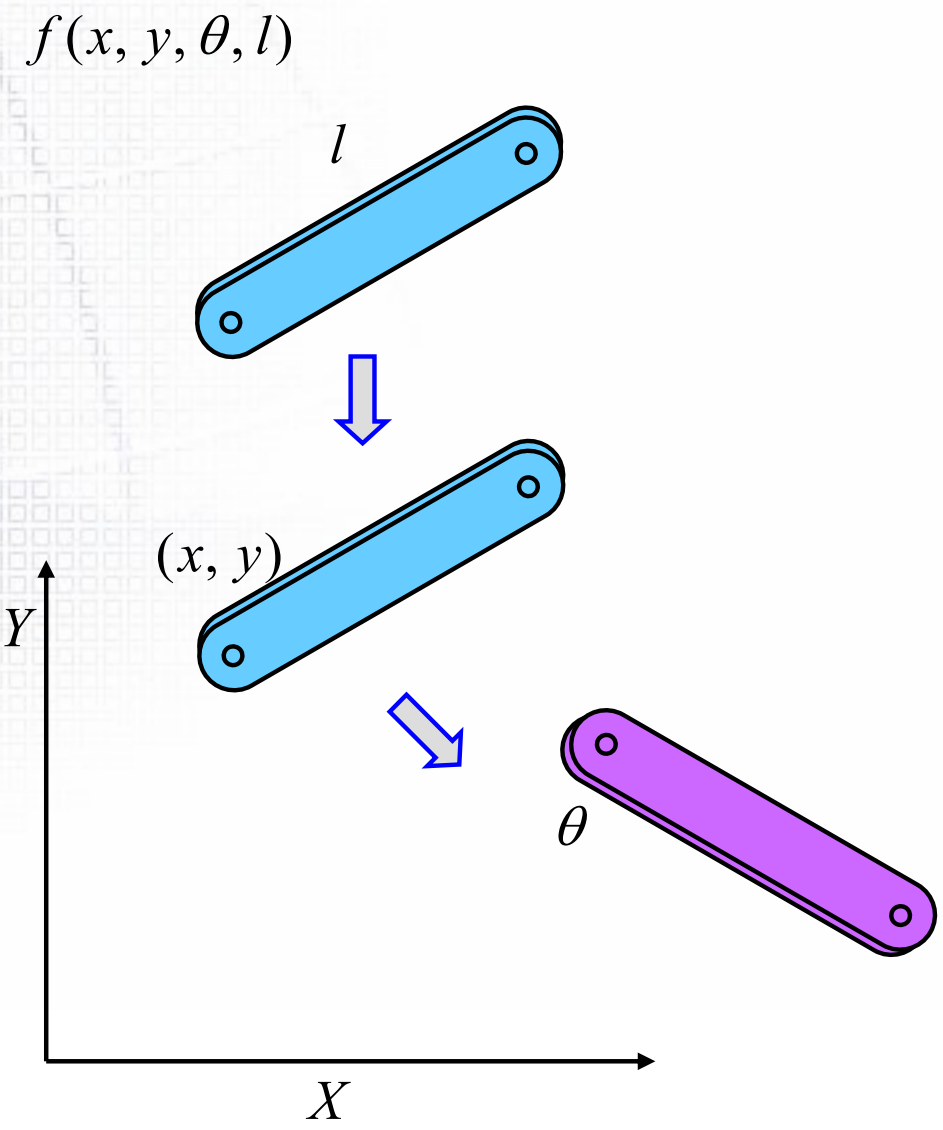
■ **이때, 평면 내에서 "이 강체는 자유도가 3이다"라고 말함**

기구(mechanism)의 자유도 정의

기구를 이루는 모든 링크들의 위치와 자세를 정의하기 위해 필요한 독립변수의 최소 개수

1.1 하나의 강체의 자유도(2)

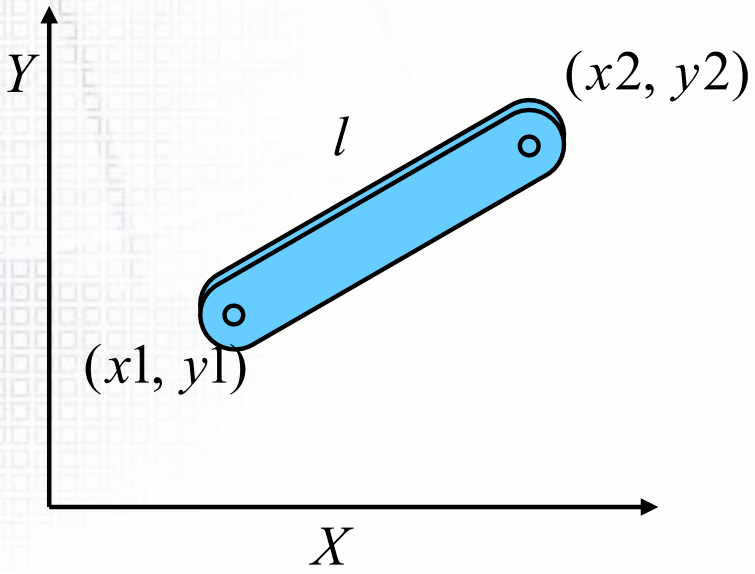
자유도가 3인 평면 위의 강체



- 한 강체가2차원 평면상에 있을 때, 강체의 위치와 방향을 기술하기 위해서는
- 강체의 **평행이동**한 모습과
- 강체의 **회전**한 모습을 모두 표현 할 수 있어야 하므로
- x, y, theta의 3가지 정보가 필요하다.
- 결국, 왼쪽의 강체의 위치와 자세는 3개의 변수(x, y, θ)로 정의 됨
- 평면 내에서 하나의 강체의 자유도가 3임을 확인할 수 있다

1.1 하나의 강체의 자유도(3)

자유도가 3인 평면 위의 강체



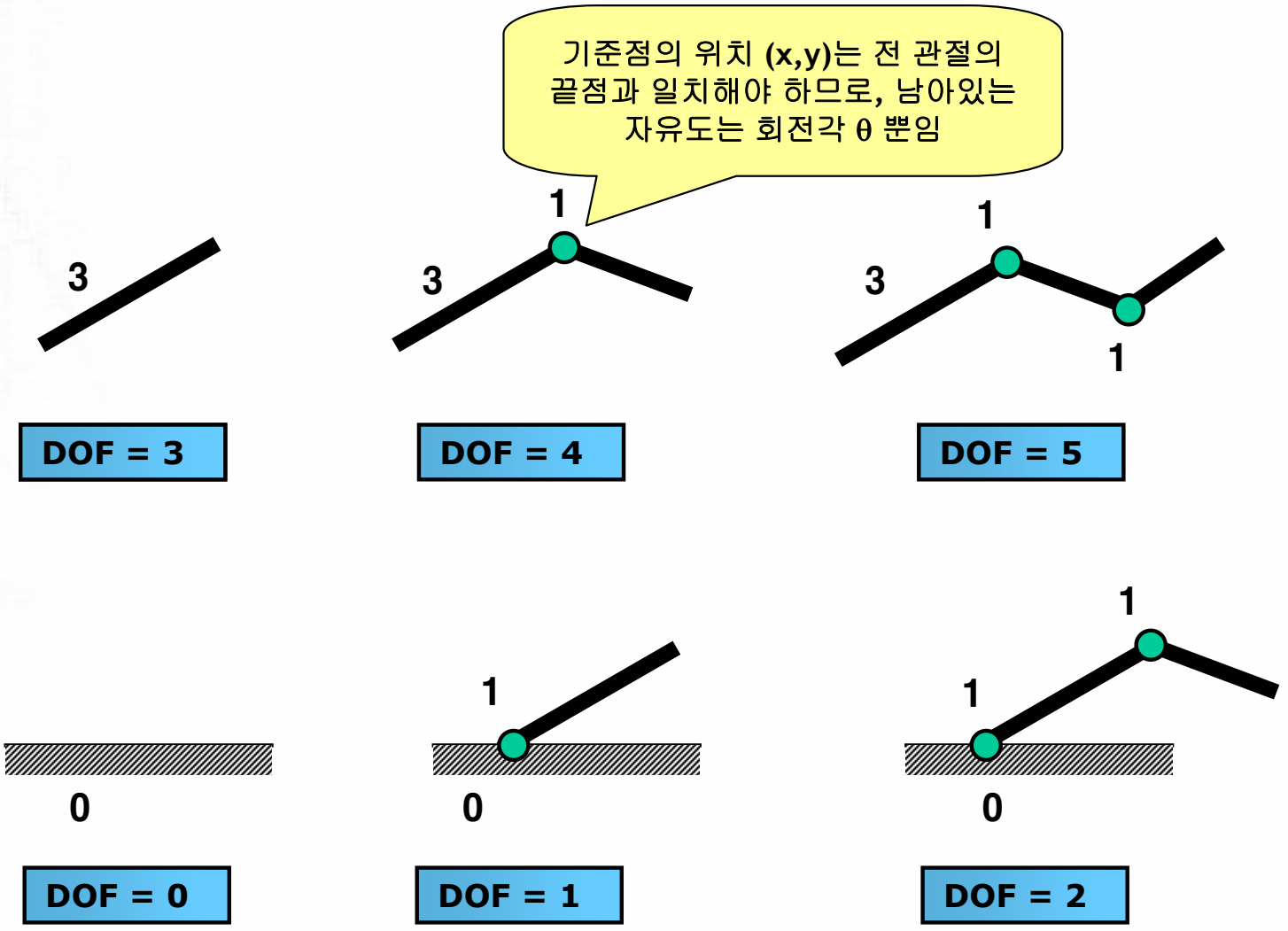
- 만약, 왼쪽의 강체를 두 점으로 표현하면 물체의 위치와 방향을 나타내는데 4개의 독립변수가 필요하지 않을까?

$$y2 = y1 + \sqrt{l^2 - (x2 - x1)^2}$$

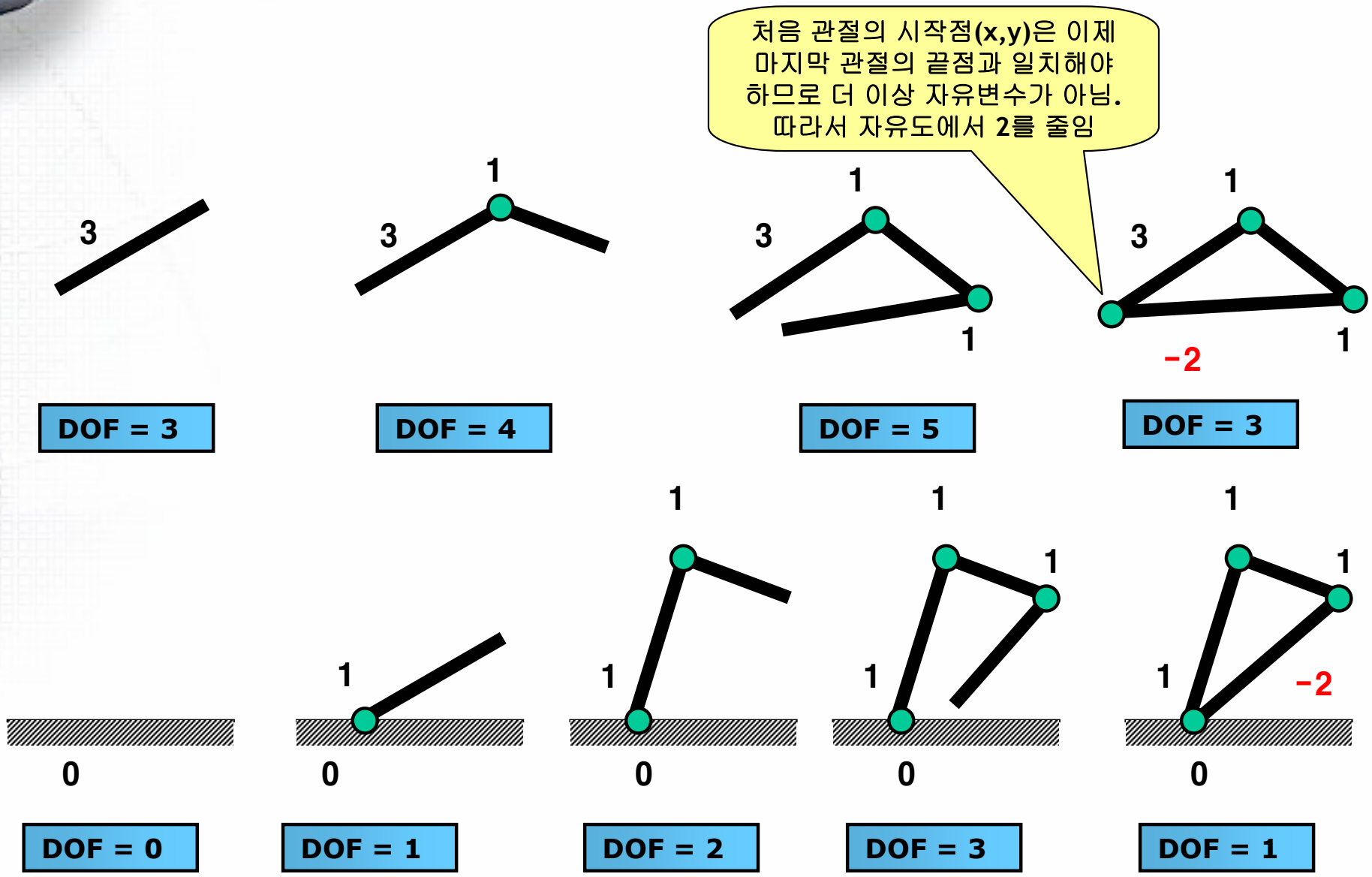
$$l = \sqrt{(x2 - x1)^2 - (y2 - y1)^2}$$

- 그러나, y2가 x1, x2, y1으로써 결정할 수 있는 종속변수이므로,
- 결국, 왼쪽의 강체의 위치와 자세는 3개의 변수(*x1*, *y1*, x2)로 정의 된다고 할 수 있다.
- 따라서, 평면 내에서 하나의 강체의 자유도는 3이다

1.2 두 개 이상의 강체로 이루어진 물체의 자유도 (1)

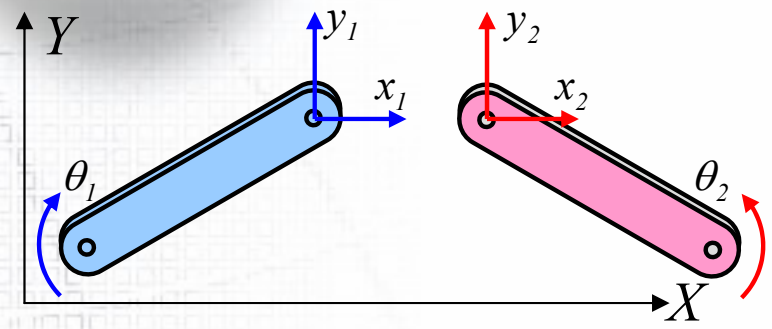


1.2 두 개 이상의 강체로 이루어진 물체의 자유도 (2)

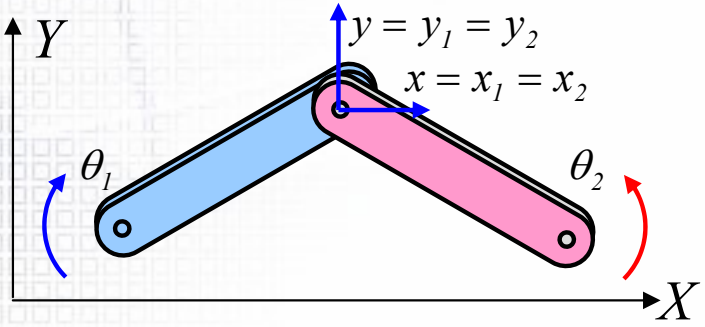


1.2 두 개 이상의 강체로 이루어진 물체의 자유도 (3)

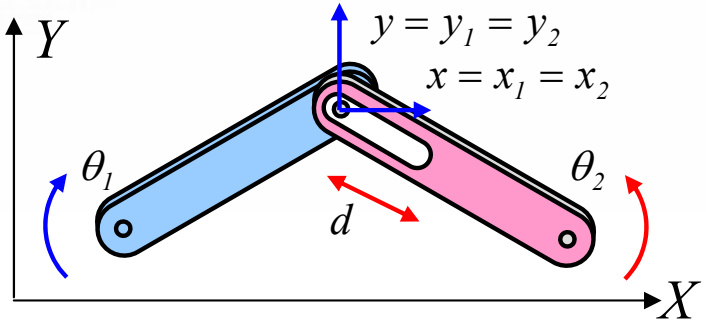
연결되지 않은 두 링크 (DOF = 6)



1DOF의 관절로 연결된 두 링크 (DOF = 4)



2DOF의 관절로 연결된 두 링크 (DOF = 5)



2차원 평면에서의 자유도 (Gruebler Formula)

- 각각의 강체는 3DOF를 가짐
- 1DOF의 관절로 연결할 경우, 전체의 자유도에서 2DOF(3-1)가 감소
- 2DOF의 관절로 연결할 경우,전체의 자유도에서 1DOF(3-2)가 감소
- 지면은 한 개이며, 0DOF를 가짐

■관절의 자유도: 두 링크의 상대적인 자세와 위치를 표현하기 위한 독립변수의 최소개수

$$\therefore DOF = 3(N-1-J) + \sum_{i=1}^{J} F_i$$

$$DOF = 3N - 3G - \sum_{i=1}^{J}(3 - F_i)$$

$$= 3(N-1) - \sum_{i=1}^{J}(3 - F_i)$$

$$= 3(N-1) - 3J - \sum_{i=1}^{J}(3 - F_i)$$

$$= 3(N-1-J) + \sum_{i=1}^{J} F_i$$

N: 링크의 수 (지면 포함)
G: 고정링크(지면)의 수
J: 관절의 개수
F_i: i번째 관절의 자유도

3차원 평면에서의 자유도 (kutzbach Formula)

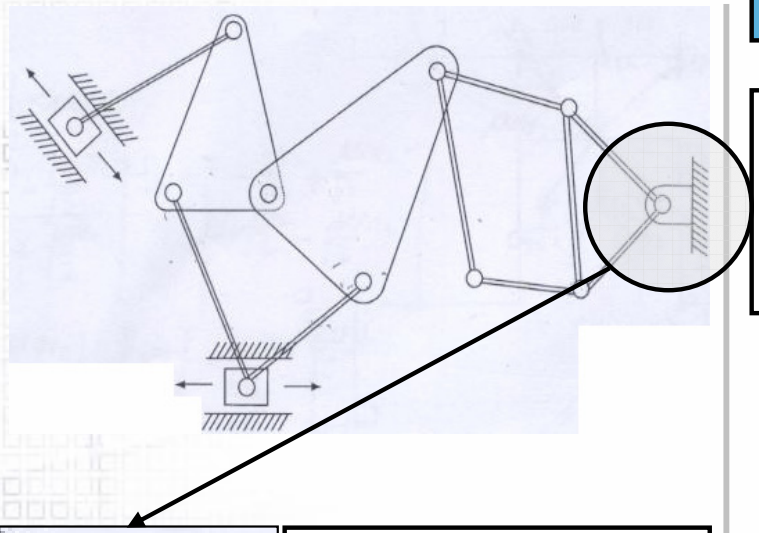
- 각각의 강체는 6DOF를 가짐

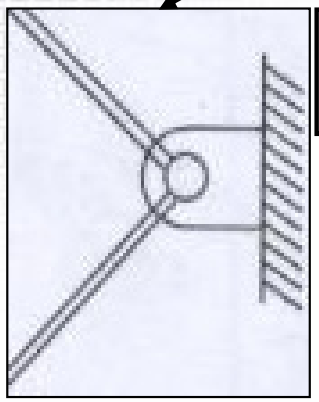
$$\therefore DOF = 6(N-1-J) + \sum_{i=1}^{J} F_i$$

N: 링크의 수 (지면 포함)
G: 고정링크(지면)의 수
J: 관절의 개수
F_i: i번째 관절의 자유도

1.3 연습 문제 (1)

문제1) 그루블러의 공식을 이용하여 다음 기구의 자유도를 구하여라.





위치만 같고, 두 개의 관절이 존재

풀이1

N: 1+11

J: 15

F_i: 13(1)+2(2)=17

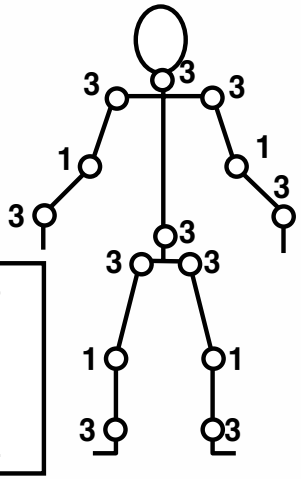
$$DOF = 3(N-1-J)+\sum_{i=1}^{J} F_i$$

$$= 3(12-1-15)+[13(1)+2(2)]$$

$$= -12+17 = 4(DOF)$$

1.3 연습 문제 (2)

**문제2) 다음은 골프를 하는 사람과 휴면 모델의 개략적인 그림이다.
주어진 모델을 참고하여 골프 하는 사람의 자유도를 구하여라.**



두 다리는 항상 그라운드에 고정되어 있음
골프 그립과 손 사이에서 슬립은 생기지 않음
동그라미는 조인트를 나타냄
동그라미 옆 숫자는 해당 조인트의 자유도

풀이1

N: 16

J: 14

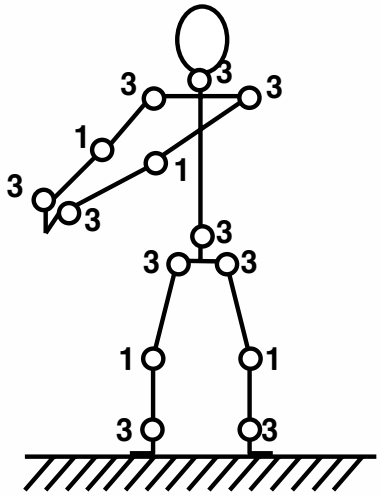
F_i: 10(3)+1(4)=34

?

$$DOF = 6(N-1-J)+\sum_{i=1}^{J} F_i$$

$$= 6(16-1-14)+[10(3)+4(1)] = 40(DOF)$$

풀이2



두 다리는 항상 그라운드에 고정되어 있음
골프 그립과 손 사이에서 슬립은 생기지 않음
동그라미는 조인트를 나타냄
동그라미 옆 숫자는 해당 조인트의 자유도

N: 13

J: 14

F_i: 10(3)+1(4)=34

$$DOF = 6(N-1-J)+\sum_{i=1}^{J} F_i$$

$$= 6(13-1-14)+[10(3)+4(1)]$$

$$= -12+34 = 22(DOF)$$

Ch 2. 공간표시와 3차원 형상변환

2.1 공간상의 점과 벡터의 표현

2.2 원점이 일치하는 두 좌표계간의 매핑변환 문제

2.3 원점이 일치하지 않는 두 좌표계간의 매핑변환 문제

Advanced
Ship
Design
Automation
Laboratory

2.1 공간상의 점과 벡터의 표현

2.1.1 공간상의 점의 표현

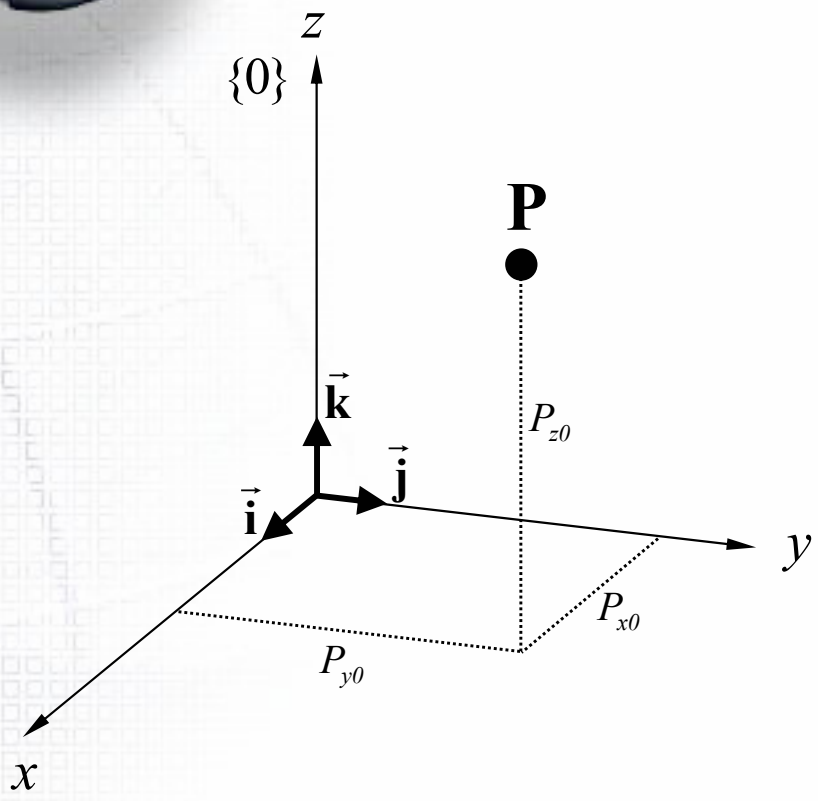
2.1.2 공간상의 벡터의 표현

2.1.3 동차좌표를 이용한 점의 평행이동변환

2.1.4 동차좌표를 이용한 벡터의 평행이동변환

2.1.5 동차좌표를 이용한 점과 벡터의 회전변환

2.1.1 공간상의 점의 표현(1)



■ 공간상의 점을 표현하기 위해서는 기준 좌표계를 결정해야 함

■ 점 **P** 는 다음과 같이 기준좌표계 $\{0\}$ 의 세개의 단위벡터 **i , j, k**로 표현가능

$$\mathbf{P} = P_{x0}\vec{\mathbf{i}} + P_{y0}\vec{\mathbf{j}} + P_{z0}\vec{\mathbf{k}}$$

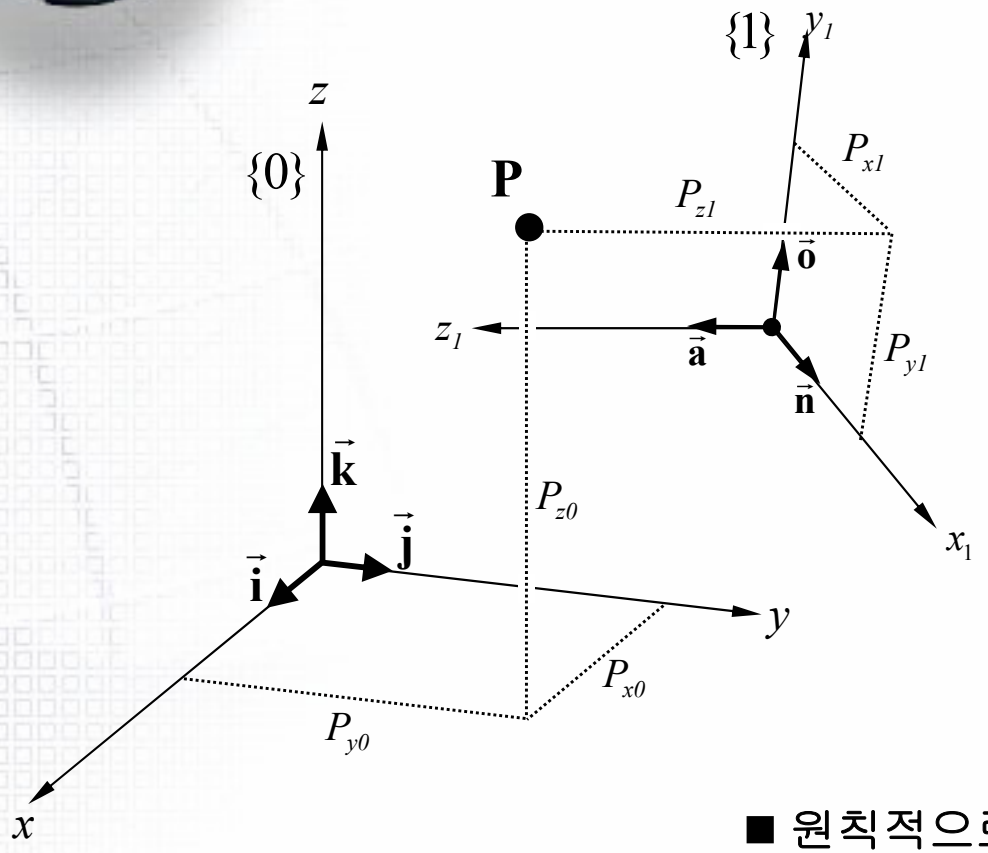
$$= \begin{bmatrix} \vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} \end{bmatrix} \begin{bmatrix} P_{x0} \\ P_{y0} \\ P_{z0} \end{bmatrix}$$

■ 이때 P_x, P_y, P_z를 점 **P**의 $\{0\}$ 좌표계에 대한 좌표라고 하고 다음과 같이 간단하게 표현할 수 있다

$$\mathbf{P} = P_{x0}\vec{\mathbf{i}} + P_{y0}\vec{\mathbf{j}} + P_{z0}\vec{\mathbf{k}} \quad \Leftrightarrow \quad \mathbf{P}_{\{0\}} = \begin{bmatrix} P_{x0} \\ P_{y0} \\ P_{z0} \end{bmatrix}$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

2.1.1 공간상의 점의 표현 (2)



■ 점 **P**를 어느 좌표계를 기준으로 표현하느냐에 따라서 좌표값이 달라짐

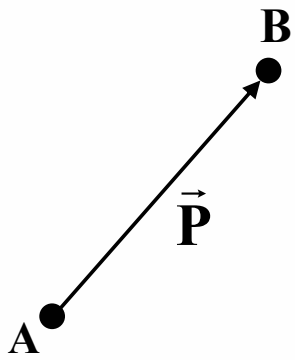
$$\mathbf{P} = P_{x0}\vec{\mathbf{i}} + P_{y0}\vec{\mathbf{j}} + P_{z0}\vec{\mathbf{k}} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix}P_{x0}\\ P_{y0}\\ P_{z0}\end{bmatrix}$$

$$= P_{x1}\vec{\mathbf{n}} + P_{y1}\vec{\mathbf{o}} + P_{z1}\vec{\mathbf{a}} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\begin{bmatrix}P_{x1}\\ P_{y1}\\ P_{z1}\end{bmatrix}$$

■ 원칙적으로, 점 **P**를 좌표값으로만 표현할 때에는 어느 좌표계를 기준으로 한 값인지 표시해주어야 함

$$\mathbf{P}_{\{0\}} = \begin{bmatrix}P_{x0}\\ P_{y0}\\ P_{z0}\end{bmatrix} \quad \Leftrightarrow \quad \mathbf{P}_{\{1\}} = \begin{bmatrix}P_{x1}\\ P_{y1}\\ P_{z1}\end{bmatrix}$$

2.1.2 공간상의 벡터의 표현



■ 벡터가 점 **A**에서 시작하고 점 **B**에서 끝난다면, 다음과 같이 표현할 수 있다

$$\vec{\mathbf{P}} = \mathbf{B} - \mathbf{A}$$
$$= (B_{x0} - A_{x0})\vec{\mathbf{i}} + (B_{y0} - A_{y0})\vec{\mathbf{j}} + (B_{z0} - A_{z0})\vec{\mathbf{k}}$$

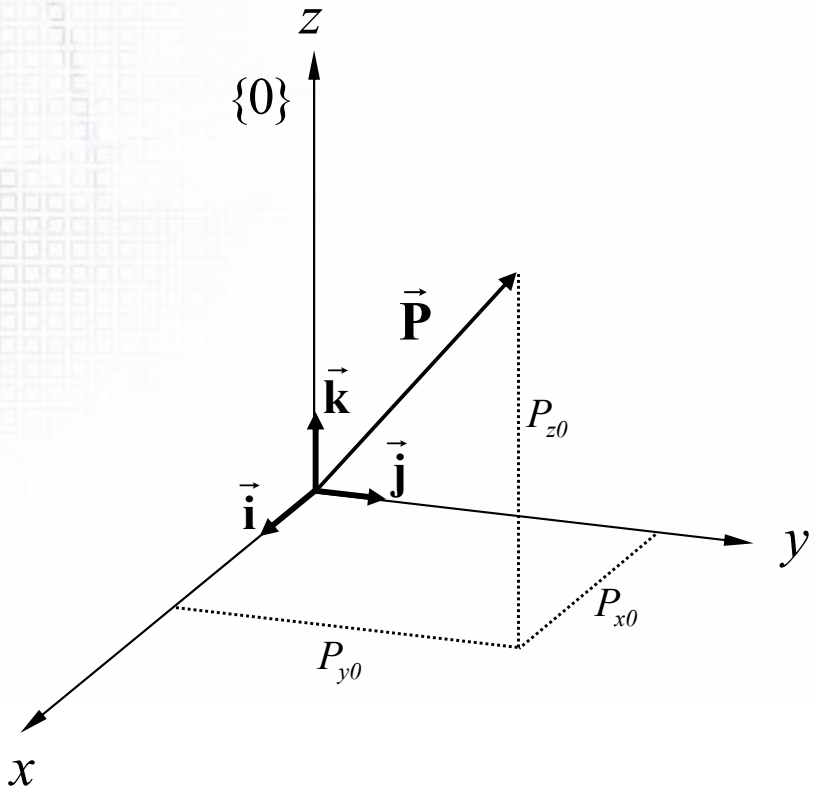
■ 벡터 **P**가 원점에서 시작한다면 다음과 같이 표현할 수 있다

$$\vec{\mathbf{P}} = P_{x0}\vec{\mathbf{i}} + P_{y0}\vec{\mathbf{j}} + P_{z0}\vec{\mathbf{k}} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix}P_{x0}\\ P_{y0}\\ P_{z0}\end{bmatrix},$$

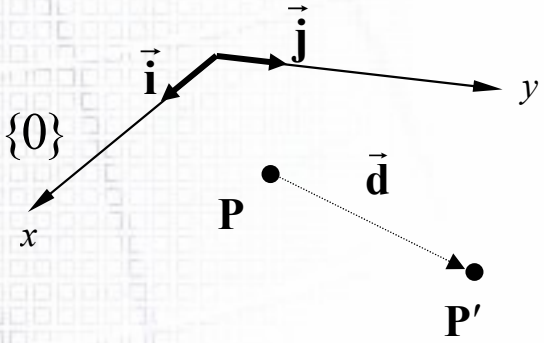
$$where \quad P_{x0} = (B_{x0} - A_{x0}),\ P_{y0} = (B_{y0} - A_{y0}),\ P_{z0} = (B_{z0} - A_{z0})$$

■ 벡터 **P**를 다음과 같이 좌표로도 표현할 수 있다
점과 동일하게 표현되므로 주의가 필요하다

$$\vec{\mathbf{P}}_{\{0\}} = \begin{bmatrix}P_{x0}\\ P_{y0}\\ P_{z0}\end{bmatrix}$$



2.1.3 동차좌표를 이용한 점의 평행이동변환 (1)



■ 점 **P**를 벡터 **d**만큼 이동하여 새로운 점 **P′**를 생성하는 변환을 생각해보자

$$\mathbf{P} = P_{x0}\vec{\mathbf{i}} + P_{y0}\vec{\mathbf{j}} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}}\end{bmatrix}\begin{bmatrix}P_{x0}\\ P_{y0}\end{bmatrix},$$

$$\vec{\mathbf{d}} = d_{x0}\vec{\mathbf{i}} + d_{y0}\vec{\mathbf{j}} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}}\end{bmatrix}\begin{bmatrix}d_{x0}\\ d_{y0}\end{bmatrix},$$

$$\mathbf{P}' = \mathrm{P} + \vec{\mathrm{d}}$$

$$= \begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}}\end{bmatrix}\begin{bmatrix}P_x\\ P_y\end{bmatrix} + \begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}}\end{bmatrix}\begin{bmatrix}d_x\\ d_y\end{bmatrix}$$

$$= \begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}}\end{bmatrix}\begin{bmatrix}P_x + d_x\\ P_y + d_y\end{bmatrix},$$

■ 반복적으로 나타나는 단위벡터 **i, j**를 제거하여 좌표만으로 표현하면 다음과 같다

행렬간의 곱으로 표시할 수는 없을까?

$$\mathbf{P}_{xy} = \begin{bmatrix}P_x\\ P_y\end{bmatrix}, \qquad \vec{\mathrm{d}}_{xy} = \begin{bmatrix}d_x\\ d_y\end{bmatrix},$$

$$\mathbf{P}'_{xy} = \mathbf{P}_{xy} + \vec{\mathrm{d}}_{xy} = \begin{bmatrix}P_x\\ P_y\end{bmatrix} + \begin{bmatrix}d_x\\ d_y\end{bmatrix} = \begin{bmatrix}P_x + d_x\\ P_y + d_y\end{bmatrix}$$

$$P_x + d_x = \begin{bmatrix}1 & 0 & d_x\end{bmatrix}\begin{bmatrix}P_x\\ P_y\\ 1\end{bmatrix} \qquad P_y + d_y = \begin{bmatrix}0 & 1 & d_y\end{bmatrix}\begin{bmatrix}P_x\\ P_y\\ 1\end{bmatrix}$$

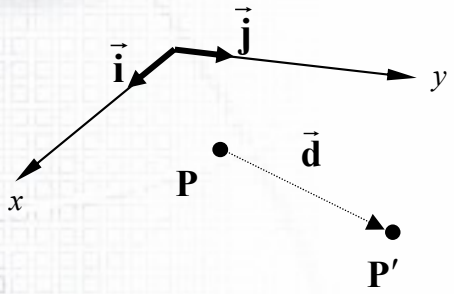
$$\therefore \begin{bmatrix}P_x + d_x\\ P_y + d_y\end{bmatrix} = \begin{bmatrix}1 & 0 & d_x\\ 0 & 1 & d_y\end{bmatrix}\begin{bmatrix}P_x\\ P_y\\ 1\end{bmatrix}$$

$$1 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}\begin{bmatrix}P_x\\ P_y\\ 1\end{bmatrix}$$

$$\therefore \begin{bmatrix}P_x + d_x\\ P_y + d_y\\ 1\end{bmatrix} = \begin{bmatrix}1 & 0 & d_x\\ 0 & 1 & d_y\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}P_x\\ P_y\\ 1\end{bmatrix}$$

2.1.3 동차좌표를 이용한 점의 평행이동변환 (2)

■ 이제 2차원 좌표에 단위벡터를 표시하여 점으로 표시해보자



직교좌표계

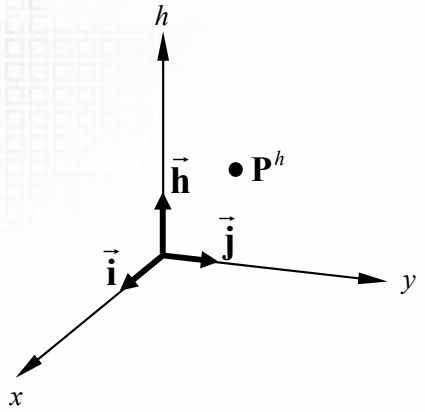
$$\mathbf{P}_{xy}=\begin{bmatrix}P_x\\P_y\end{bmatrix},\quad \vec{\mathrm{d}}_{xy}=\begin{bmatrix}d_x\\d_y\end{bmatrix},$$

$$\mathrm{P}'_{xy}=\mathbf{P}_{xy}+\vec{\mathrm{d}}_{xy}=\begin{bmatrix}P_x\\P_y\end{bmatrix}+\begin{bmatrix}d_x\\d_y\end{bmatrix}=\begin{bmatrix}P_x+d_x\\P_y+d_y\end{bmatrix}$$

$$\mathbf{P}=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}}\end{bmatrix}\begin{bmatrix}P_x\\P_y\end{bmatrix}=P_x\vec{\mathrm{i}}+P_y\vec{\mathrm{j}},$$

$$\vec{\mathrm{d}}=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}}\end{bmatrix}\begin{bmatrix}d_x\\d_y\end{bmatrix}=d_x\vec{\mathrm{i}}+d_y\vec{\mathrm{j}},$$

$$\mathbf{P}'=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}}\end{bmatrix}\begin{bmatrix}P_x+d_x\\P_y+d_y\end{bmatrix}=(P_x+d_x)\vec{\mathrm{i}}+(P_y+d_y)\vec{\mathrm{j}}$$



동차좌표계

점의 좌표 = 변환 행렬 × 점의 좌표

$$\begin{bmatrix}P_x+d_x\\P_y+d_y\\1\end{bmatrix}=\begin{bmatrix}1 & 0 & d_x\\0 & 1 & d_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}P_x\\P_y\\1\end{bmatrix}$$

$$\mathbf{P}=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}} & ?\end{bmatrix}\begin{bmatrix}P_x\\P_y\\1\end{bmatrix}=P_x\vec{\mathrm{i}}+P_y\vec{\mathrm{j}}+1\cdot ?$$

$$P_x+d_x=\begin{bmatrix}1 & 0 & d\end{bmatrix}\begin{bmatrix}P_x\\P_y\\1_x\end{bmatrix}$$

$$P_y+d_y=\begin{bmatrix}0 & 1 & d_y\end{bmatrix}\begin{bmatrix}P_x\\P_y\\1\end{bmatrix}$$

$$1=\begin{bmatrix}0 & 0 & 1\end{bmatrix}\begin{bmatrix}P_x\\P_y\\1\end{bmatrix}$$

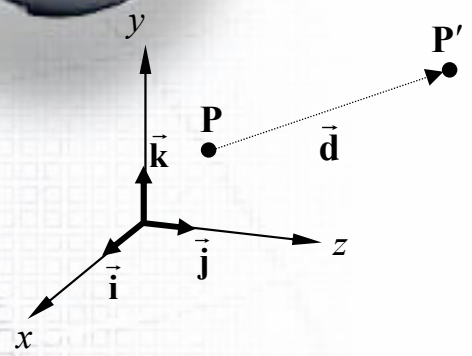
$$\mathbf{P}^h=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}} & \vec{\mathrm{h}}\end{bmatrix}\begin{bmatrix}P_x\\P_y\\1\end{bmatrix}=P_x\vec{\mathrm{i}}+P_y\vec{\mathrm{j}}+1\cdot\vec{\mathrm{h}}$$

점 P의 동차좌표계에서의 표현

$$\Leftrightarrow\quad \mathbf{P}=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}}\end{bmatrix}\begin{bmatrix}P_x\\P_y\end{bmatrix}=P_x\vec{\mathrm{i}}+P_y\vec{\mathrm{j}},$$

점 P의 직교좌표계에서의 표현

2.1.3 동차좌표를 이용한 점의 평행이동변환 (3)



직교좌표계

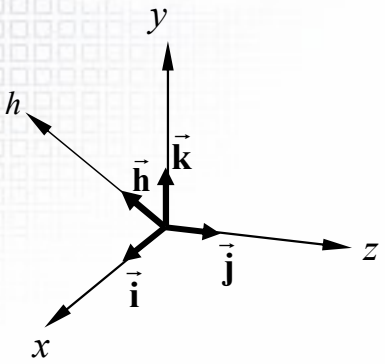
■ 이제 3차원 좌표에 단위벡터를 표시하여 점으로 표시해보자

$$\mathbf{P}_{xyz}=\begin{bmatrix}P_x\\P_y\\P_z\end{bmatrix},\quad \vec{\mathrm{d}}_{xyz}=\begin{bmatrix}d_x\\d_y\\d_z\end{bmatrix}$$

$$\mathbf{P}'_{xyz}=\mathbf{P}_{xyz}+\vec{\mathrm{d}}_{xyz}=\begin{bmatrix}P_x\\P_y\\P_z\end{bmatrix}+\begin{bmatrix}d_x\\d_y\\d_z\end{bmatrix}=\begin{bmatrix}P_x+d_x\\P_y+d_y\\P_z+d_z\end{bmatrix}$$

$$\mathbf{P}=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}} & \vec{\mathrm{k}}\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\end{bmatrix}=P_x\vec{\mathrm{i}}+P_y\vec{\mathrm{j}}+P_z\vec{\mathrm{k}}$$

$$\mathbf{P}'=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}} & \vec{\mathrm{k}}\end{bmatrix}\begin{bmatrix}P_x+d_x\\P_y+d_y\\P_z+d_z\end{bmatrix}=(P_x+d_x)\vec{\mathrm{i}}+(P_y+d_y)\vec{\mathrm{j}}+(P_z+d_z)\vec{\mathrm{k}}$$



동차좌표계

점의 좌표 $\begin{bmatrix}P_x+d_x\\P_y+d_y\\P_z+d_z\\1\end{bmatrix}=\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}$ 점의 좌표

변환 행렬

$$P_x+d_x=\begin{bmatrix}1&0&0&d_x\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}\qquad 1=\begin{bmatrix}0&0&0&1\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}$$

$$P_y+d_y=\begin{bmatrix}0&1&0&d_y\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}$$

$$P_z+d_z=\begin{bmatrix}0&0&1&d_z\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}$$

$$\mathbf{P}=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}} & \vec{\mathrm{k}} & ?\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}=P_x\vec{\mathrm{i}}+P_y\vec{\mathrm{j}}+P_z\vec{\mathrm{k}}+1\cdot ?$$

$$\mathbf{P}^h=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}} & \vec{\mathrm{k}} & \vec{\mathrm{h}}\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}=P_x\vec{\mathrm{i}}+P_y\vec{\mathrm{j}}+P_z\vec{\mathrm{k}}+1\cdot\vec{\mathrm{h}}$$

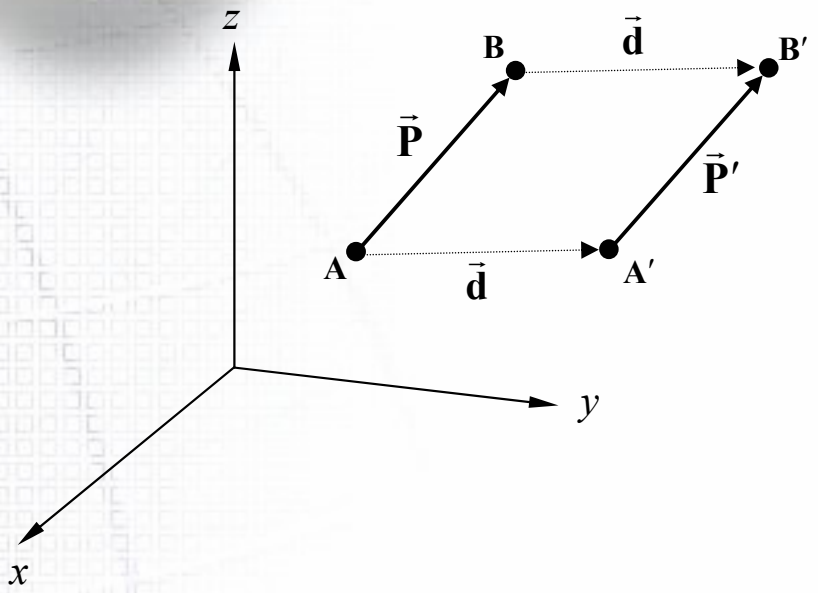
점 P의 동차좌표계 표현

$\Leftrightarrow$

$$\mathbf{P}=\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}} & \vec{\mathrm{k}}\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\end{bmatrix}=P_x\vec{\mathrm{i}}+P_y\vec{\mathrm{j}}+P_z\vec{\mathrm{k}}$$

점 P의 직교좌표계 표현

2.1.4 동차좌표를 이용한 벡터의 평행이동변환 (1)
- 벡터의 평행이동변환을 동차좌표로 표현할 수 있을까?



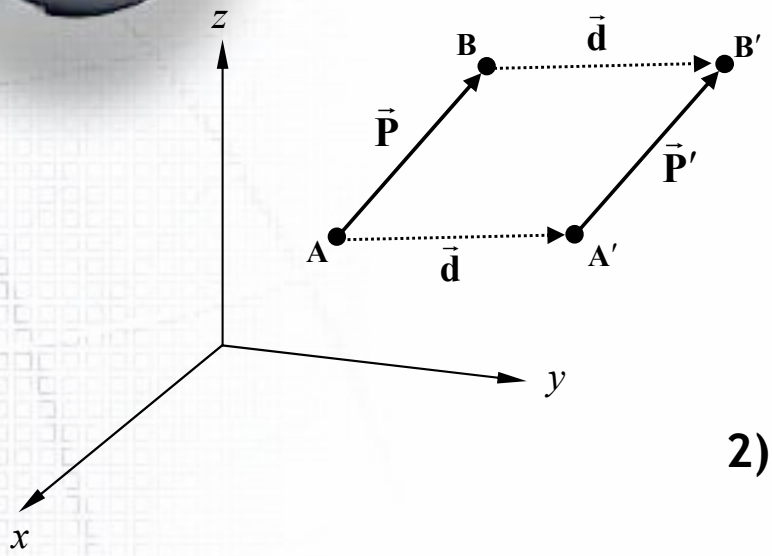
■ 일반적인 직교좌표를 이용한 벡터의 평행이동 표현
→ 벡터는 평행이동 후에도 변하지 않음

$$\vec{\mathbf{P}}_{xyz} = \mathrm{B}_{xyz} - \mathrm{A}_{xyz} = \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} - \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} = \begin{bmatrix} B_x - A_x \\ B_y - A_y \\ B_z - A_z \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

$$\vec{\mathbf{P}}'_{xyz} = \mathrm{B}'_{xyz} - \mathrm{A}'_{xyz} = \left(\mathrm{B}_{xyz} + \mathrm{d}_{xyz}\right) - \left(\mathrm{A}_{xyz} + \mathrm{d}_{xyz}\right) = \begin{bmatrix} B_x + d_x \\ B_y + d_y \\ B_z + d_z \end{bmatrix} - \begin{bmatrix} A_x + d_x \\ A_y + d_y \\ A_z + d_z \end{bmatrix} = \begin{bmatrix} B_x - A_x \\ B_y - A_y \\ B_z - A_z \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \vec{\mathbf{P}}_{xyz}$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

2.1.4 동차좌표를 이용한 벡터의 평행이동변환 (2)



■ 동차좌표를 이용한 벡터의 평행이동 표현

1) 벡터 $\vec{\mathrm{P}}_{xyz} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$ 의 동차좌표 표현이 $\vec{\mathrm{P}}^{h}_{xyz} = \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$ 라면,

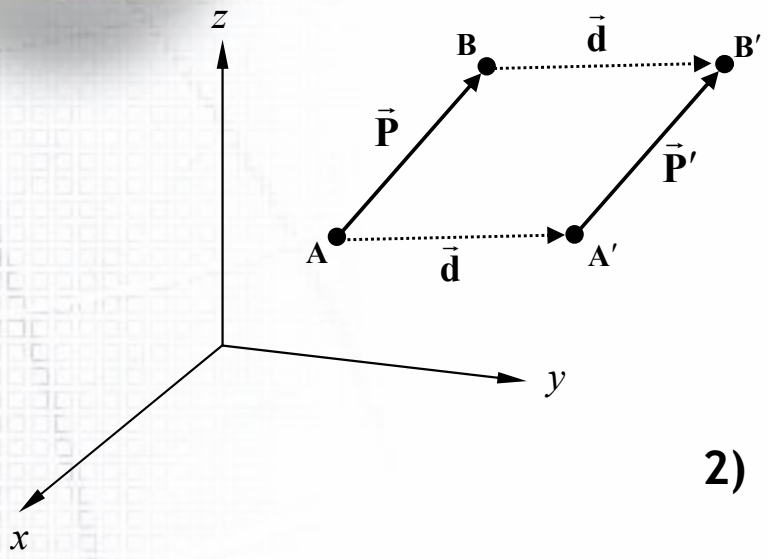
2) 평행이동 후의 벡터는 동차좌표의 평행이동변환 행렬을 이용하여 다음과 같이 계산할 수 있다

$$\vec{\mathrm{P}}'^{h}_{xyz} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} P_x + d_x \\ P_x + d_y \\ P_x + d_z \\ 1 \end{bmatrix}$$

$$\therefore \vec{\mathrm{P}}'_{xyz} = \begin{bmatrix} P_x + d_x \\ P_x + d_y \\ P_x + d_z \end{bmatrix} \neq \vec{\mathrm{P}}_{xyz}$$

3) 이는 잘못된 결과로, 이러한 결과를 얻게 된 이유는
1)에서 가정한 벡터의 동차좌표 표현이 잘못되었기 때문이다

2.1.4 동차좌표를 이용한 벡터의 평행이동변환 (3)



■ 동차좌표를 이용한 벡터의 평행이동 표현

1) 벡터 $\vec{\mathbf{P}}_{xyz} = \begin{bmatrix} P_x & P_y & P_z \end{bmatrix}^T$ 의 동차좌표 표현은

$\vec{\mathbf{P}}^h_{xyz} = \begin{bmatrix} P_x & P_y & P_z & 1 \end{bmatrix}^T$ 이 아니라

$\vec{\mathbf{P}}^h_{xyz} = \begin{bmatrix} P_x & P_y & P_z & 0 \end{bmatrix}^T$ 이다

2) 평행이동 후의 벡터는 동차좌표의 평행이동변환 행렬을 이용하여 다음과 같이 계산할 수 있다

$$\vec{\mathbf{P}}'^h_{xyz} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 0 \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \\ P_z \\ 0 \end{bmatrix}$$

→ 평행이동변환 후에도 결과는 벡터(P_h=0)이며, 각 성분값은 변화가 없음을 알 수 있다

$$\therefore \vec{\mathbf{P}}'_{xyz} = \begin{bmatrix} P_x \\ P_x \\ P_x \end{bmatrix} = \vec{\mathbf{P}}_{xyz}$$

[정리] 동차좌표를 이용한 점과 벡터의 평행이동변환

■ 점의 동차좌표 표현

$$\mathrm{P}_{xyz}=\begin{bmatrix}P_x\\P_y\\P_z\end{bmatrix} \quad \Leftrightarrow \quad \mathrm{P}^h_{xyz}=\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}$$

■ 벡터의 동차좌표 표현*

$$\vec{\mathrm{P}}_{xyz}=\begin{bmatrix}P_x\\P_y\\P_z\end{bmatrix} \quad \Leftrightarrow \quad \vec{\mathrm{P}}^h_{xyz}=\begin{bmatrix}P_x\\P_y\\P_z\\0\end{bmatrix}$$

■ 점과 벡터의 평행이동

$$\mathrm{P}'^h_{xyz}=\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}\cdot\mathrm{P}^h_{xyz}$$

$$=\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}=\begin{bmatrix}P_x+d_x\\P_y+d_y\\P_z+d_z\\1\end{bmatrix}$$

$$\vec{\mathrm{P}}'^h_{xyz}=\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}\cdot\vec{\mathrm{P}}^h_{xyz}$$

$$=\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}P_x\\P_y\\P_z\\0\end{bmatrix}=\begin{bmatrix}P_x\\P_y\\P_z\\0\end{bmatrix}=\vec{\mathrm{P}}^h_{xyz}$$

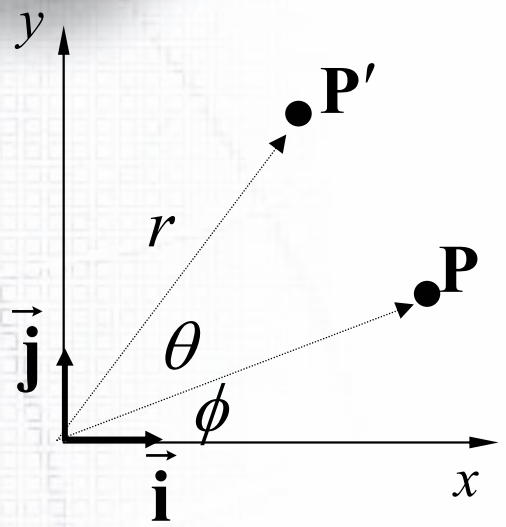
벡터는 평행이동변환 후에도 변화가 없음

변환 행렬

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

* 점과 벡터는 직교좌표에서는 표현방법이 동일하지만, 동차좌표에서는 표현방법이 다르기 때문에 점과 벡터의 구분이 가능함

2.1.5 동차좌표를 이용한 점과 벡터의 회전변환 (1)



■ 점 P를 x축에 대하여 theta 만큼 회전하여 새로운 점 P′를 생성하는 변환을 생각해보자

$$\mathbf{P} = P_x\vec{\mathrm{i}} + P_y\vec{\mathrm{j}} = \begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}}\end{bmatrix}\begin{bmatrix}P_x \\ P_y\end{bmatrix},$$

$$\mathrm{P}' = \mathrm{P}'_x\vec{\mathrm{i}} + \mathrm{P}'_y\vec{\mathrm{j}} = \begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}}\end{bmatrix}\begin{bmatrix}\mathrm{P}'_x \\ \mathrm{P}'_y\end{bmatrix},$$

■ 점 P를 원점을 시점으로 하는 벡터라고 생각한다면, 벡터의 회전변환도 점의 회전변환과 동일하게 계산할 수 있음을 알 수 있다

$$P'_x = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta \qquad P_x = r\cos\phi$$
$$P'_y = r\sin(\phi+\theta) = r\sin\phi\cos\theta + r\cos\phi\sin\theta \qquad P_y = r\sin\phi$$

$$P'_x = r\cos(\phi+\theta) = P_x\cos\theta - P_y\sin\theta$$
$$P'_y = r\sin(\phi+\theta) = P_x\cos\theta + P_y\sin\theta = P_x\sin\theta + P_y\cos\theta$$

$$\begin{bmatrix}P'_x \\ P'_y \\ 1\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}P_x \\ P_y \\ 1\end{bmatrix}$$

$$Rot(z,\theta)$$

$$\begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}} & \vec{\mathrm{h}}\end{bmatrix}\begin{bmatrix}P'_x \\ P'_y \\ 1\end{bmatrix} = \begin{bmatrix}\vec{\mathrm{i}} & \vec{\mathrm{j}} & \vec{\mathrm{h}}\end{bmatrix}\begin{bmatrix}\mathbf{cos}\theta & -\sin\theta & 0 \\ \sin\theta & \mathbf{cos}\theta & 0 \\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}P_x \\ P_y \\ 1\end{bmatrix}$$

점의 회전변환은 점의 왼쪽에 곱한다

2.1.5 동차좌표를 이용한 점과 벡터의 회전변환 (2)

■ 점 **P**가 **3**차원 공간상의 점 또는 벡터라면, 각 축에 대한 회전변환은 다음과 같다

■**Z**축 회전

$$\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}\begin{bmatrix}P'_x\\P'_y\\P'_z\\1\end{bmatrix}=\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}\begin{bmatrix}\cos\theta & -\sin\theta & 0 & 0\\ \sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}$$

$$Rot(z,\theta)$$

■**Y**축 회전

$$\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}\begin{bmatrix}P'_x\\P'_y\\P'_z\\1\end{bmatrix}=\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}\begin{bmatrix}\cos\theta & 0 & \sin\theta & 0\\ 0 & 1 & 0 & 0\\ -\sin\theta & 0 & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}$$

$$Rot(y,\theta)$$

■**X**축 회전

$$\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}\begin{bmatrix}P'_x\\P'_y\\P'_z\\1\end{bmatrix}=\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & -\sin\theta & 0\\ 0 & \sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}P_x\\P_y\\P_z\\1\end{bmatrix}$$

$$Rot(x,\theta)$$

점과 벡터의
평행이동변환, 회전변환은
한 좌표계 내에서 좌표만 변하는 것이다

@SDAL Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

2.2 원점이 일치하는 두 좌표계 간의 매핑변환 문제

2.2.1 원점이 일치하는 두 좌표계간의 매핑변환 문제

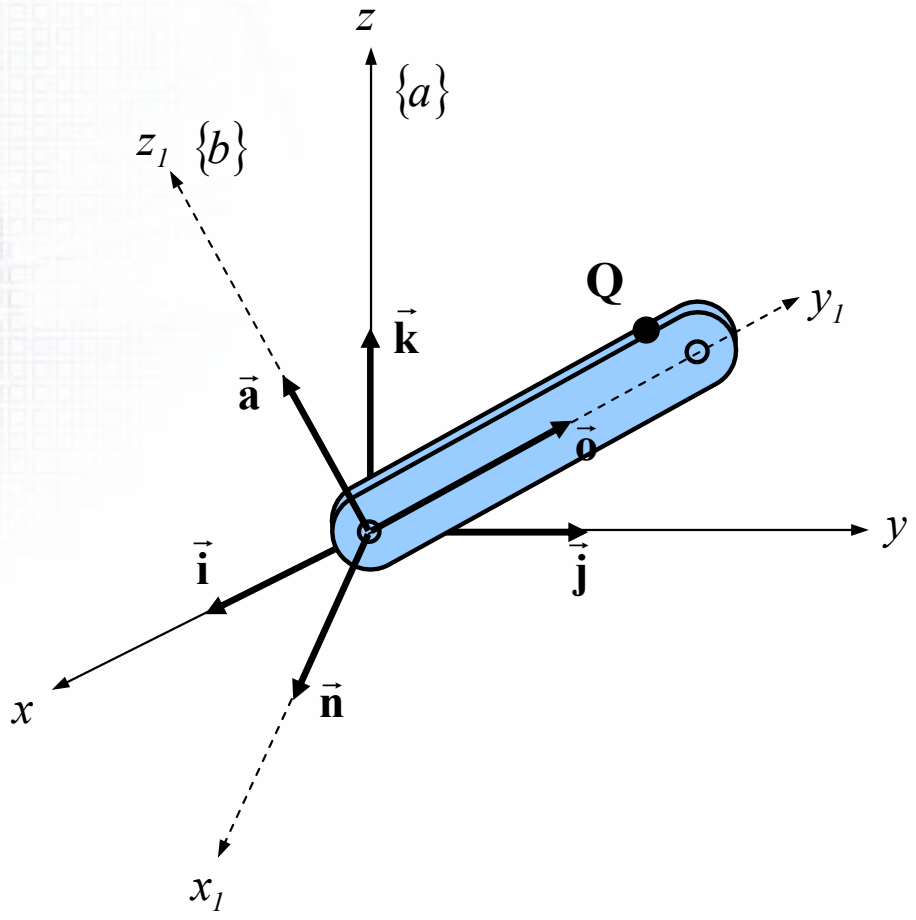
2.2.2 원점이 일피하는 두 좌표계산의 매핑변환 행렬

2.2.3 원점이 일치하는 두 좌표계간의 매핑변환 행렬의 성질

2.2.4 원점이 일치하는 세 개 이상의 좌표계 사이의 매핑변환 문제

2.2.1 원점이 일치하는 두 좌표계간의 매핑변환문제

- 원점이 일치하는 두 개의 좌표계 $\{a\}$ 와 $\{b\}$ 가 있을 때,
- 한 좌표계상의 어느 점의 좌표값을 알 때,
  다른 좌표계에서는 어떠한 좌표값에 해당되는 지를 계산하는 문제



$$\mathbf{Q} = Q_{xb}\,\vec{\mathbf{n}} + Q_{yb}\,\vec{\mathbf{o}} + Q_{zb}\,\vec{\mathbf{a}} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\begin{bmatrix}Q_{xb}\\ Q_{yb}\\ Q_{zb}\end{bmatrix}$$

$$= \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix}Q_{xa}\\ Q_{ya}\\ Q_{za}\end{bmatrix} = Q_{xa}\,\vec{\mathbf{i}} + Q_{ya}\,\vec{\mathbf{j}} + Q_{za}\,\vec{\mathbf{k}}$$

$$\mathbf{Q}_b = \begin{bmatrix}Q_{xb}\\ Q_{yb}\\ Q_{zb}\end{bmatrix} \quad \Leftrightarrow \quad \mathbf{Q}_a = \begin{bmatrix}Q_{xa}\\ Q_{ya}\\ Q_{za}\end{bmatrix}$$

2.2.2 원점이 일치하는 두 좌표계간의 매핑변환 행렬 (1)

■ 만약, 좌표계 {*b*} 의 단위벡터 **n, o, a**가 다음과 같이
좌표계 {*a*} 의 단위벡터 **i, j, k**로 표현된다면,

$$\vec{\mathbf{n}} = n_x\,\vec{\mathbf{i}} + n_y\,\vec{\mathbf{j}} + n_z\,\vec{\mathbf{k}}$$
$$\vec{\mathbf{o}} = o_x\,\vec{\mathbf{i}} + o_y\,\vec{\mathbf{j}} + o_z\,\vec{\mathbf{k}}$$
$$\vec{\mathbf{a}} = a_x\,\vec{\mathbf{i}} + a_y\,\vec{\mathbf{j}} + a_z\,\vec{\mathbf{k}}$$

$$\begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\cdot \mathbf{R}_{ab}$$

■ 행렬 **R**은 두 좌표계 사이의 방향관계를 표현하고 있다

→ 두 좌표계의 원점은 일치하고 방향만 다르므로
한 좌표계를 원점을 중심으로 **회전**시키면 항상 두 좌표계가 일치하도록
만들 수 있다

→ 좌표계 {*a*}의 방향을 나타내는 단위벡터 **i, j, k**가 행렬 **R**에 의해서
좌표계 {*b*}의 방향을 나타내는 단위벡터 **n, o, a**와 일치하게 되므로
행렬 **R**을 좌표계 {*a*}에서 좌표계 {*b*}로의 '**회전변환행렬**' 이라고 부르고
'$\mathbf{R}_{ab}$'라고 표기한다

2.2.2 원점이 일치하는 두 좌표계간의 매핑변환 행렬 (2)

■ 좌표계 {*a*}에서 좌표계 {*b*}로의 회전 변환행렬 $\mathbf{R}_{ab}$가 다음과 같이 주어졌다면,

$$\vec{\mathbf{n}} = n_x\,\vec{\mathbf{i}} + n_y\,\vec{\mathbf{j}} + n_z\,\vec{\mathbf{k}}$$
$$\vec{\mathbf{o}} = o_x\,\vec{\mathbf{i}} + o_y\,\vec{\mathbf{j}} + o_z\,\vec{\mathbf{k}}$$
$$\vec{\mathbf{a}} = a_x\,\vec{\mathbf{i}} + a_y\,\vec{\mathbf{j}} + a_z\,\vec{\mathbf{k}}$$

$$\begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\cdot \mathbf{R}_{ab}$$

■ 좌표계 {*a*}에서 좌표계 {*b*}로의 회전 변환행렬 $\mathbf{R}_{ab}$ 는
점**Q**의 좌표계 {*b*}에 대한 좌표값 $\mathbf{Q}_b=(Q_{xb}, Q_{yb}, Q_{zb})$를
좌표계 {*a*}에 대한 좌표값 $\mathbf{Q}_a=(Q_{xa}, Q_{ya}, Q_{za})$ 로 대응시키는 '**매핑변환행렬**' 이다

$$\mathbf{Q} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix}\begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} Q_{xa} \\ Q_{ya} \\ Q_{za} \end{bmatrix}$$

$$\therefore \begin{bmatrix} Q_{xa} \\ Q_{ya} \\ Q_{za} \end{bmatrix} = \mathbf{R}_{ab}\cdot\begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \end{bmatrix}$$

2.2.3 원점이 일치하는 두 좌표계간의 매핑변환 행렬의 성질 (1)

■ 좌표계 $\{a\}$에서 좌표계 $\{b\}$로의 회전 변환행렬 $\mathbf{R}_{ab}$ 의
(= 좌표계 $\{b\}$의 좌표값을 좌표계 $\{a\}$의 좌표값으로 대응시키는 매핑변환행렬)

$$\vec{\mathbf{n}} = n_x\,\vec{\mathbf{i}} + n_y\,\vec{\mathbf{j}} + n_z\,\vec{\mathbf{k}}$$
$$\vec{\mathbf{o}} = o_x\,\vec{\mathbf{i}} + o_y\,\vec{\mathbf{j}} + o_z\,\vec{\mathbf{k}}$$
$$\vec{\mathbf{a}} = a_x\,\vec{\mathbf{i}} + a_y\,\vec{\mathbf{j}} + a_z\,\vec{\mathbf{k}}$$

$$\begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\cdot \mathbf{R}_{ab}$$

■ 행렬의 원소 9개는 서로 독립적이지 않고 다음의 식을 만족시킨다

1. 각 벡터들은 단위벡터 임

$$\|\vec{\mathbf{n}}\|^2 = \|\vec{\mathbf{o}}\|^2 = \|\vec{\mathbf{a}}\|^2 = 1$$

2. 각 벡터간 직교관계 임

$$\vec{\mathbf{n}}\cdot\vec{\mathbf{o}} = \vec{\mathbf{o}}\cdot\vec{\mathbf{a}} = \vec{\mathbf{a}}\cdot\vec{\mathbf{n}} = 0$$

3. 전치행렬은 자기자신의 역행렬이다

$$(\mathbf{R}_{ab})^{\mathrm{T}}\mathbf{R}_{ab} = \mathbf{I} \qquad \therefore (\mathbf{R}_{ab})^{-1} = (\mathbf{R}_{ab})^{\mathrm{T}}$$

4. $\mathbf{R}_{ab}$의 역변환인 $\mathbf{R}_{ba}$는 $\mathbf{R}_{ab}$의 전치행렬을 이용하여 구할 수 있다

$$(\mathbf{R}_{ab})^{-1} = (\mathbf{R}_{ab})^{-\mathrm{T}} = \mathbf{R}_{ba}$$

■ 9개의 unknown과
6개의 constraint이 있음

회전행렬은 3개의 독립변수를
가지고 있음

2.2.3 원점이 일치하는 두 좌표계간의 매핑변환 행렬의 성질 (2)

3. 전치행렬은 자기자신의 역행렬이다

$$\left(\mathbf{R}_{ab}\right)^{\mathrm{T}}\mathbf{R}_{ab}=\mathbf{I} \qquad \therefore \left(\mathbf{R}_{ab}\right)^{-1}=\left(\mathbf{R}_{ab}\right)^{\mathrm{T}}$$

[증명]

$$let \quad \mathbf{R}_{ab}=\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix}=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix},$$

1. $\|\vec{\mathbf{n}}\|^2=\|\vec{\mathbf{o}}\|^2=\|\vec{\mathbf{a}}\|^2=1$

2. $\vec{\mathbf{n}}\cdot\vec{\mathbf{o}}=\vec{\mathbf{n}}\cdot\vec{\mathbf{a}}=\vec{\mathbf{o}}\cdot\vec{\mathbf{a}}=0$

$$\left(\mathbf{R}_{ab}\right)^{T}\mathbf{R}_{ab}=\begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$=\begin{bmatrix} r_{11}^2+r_{21}^2+r_{31}^2 & r_{11}r_{12}+r_{21}r_{22}+r_{31}r_{32} & r_{11}r_{13}+r_{21}r_{23}+r_{31}r_{33} \\ r_{12}r_{11}+r_{22}r_{21}+r_{32}r_{31} & r_{12}^2+r_{22}^2+r_{32}^2 & r_{12}r_{13}+r_{22}r_{23}+r_{32}r_{33} \\ r_{13}r_{11}+r_{23}r_{21}+r_{33}r_{31} & r_{13}r_{12}+r_{23}r_{22}+r_{33}r_{32} & r_{13}^2+r_{23}^2+r_{33}^2 \end{bmatrix}$$

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

$$\because r_{11}^2+r_{21}^2+r_{31}^2=r_{12}^2+r_{22}^2+r_{32}^2=r_{13}^2+r_{23}^2+r_{33}^2=1$$

2.2.3 원점이 일치하는 두 좌표계간의 매핑변환 행렬의 성질 (3)

4. $\mathbf{R}_{ab}$의 역변환인 $\mathbf{R}_{ba}$는 $\mathbf{R}_{ab}$의 전치행렬을 이용하여 구할 수 있다

$$(\mathbf{R}_{ab})^{-1} = (\mathbf{R}_{ab})^{-T} = \mathbf{R}_{ba}$$

■ 좌표계 $\{a\}$에서 좌표계 $\{b\}$로의 회전 변환행렬 $\mathbf{R}_{ab}$의
(= 좌표계 $\{b\}$의 좌표값을 좌표계 $\{a\}$의 좌표값으로 대응시키는 매핑변환행렬)

$$\vec{\mathbf{n}} = n_x\,\vec{\mathbf{i}} + n_y\,\vec{\mathbf{j}} + n_z\,\vec{\mathbf{k}}$$
$$\vec{\mathbf{o}} = o_x\,\vec{\mathbf{i}} + o_y\,\vec{\mathbf{j}} + o_z\,\vec{\mathbf{k}}$$
$$\vec{\mathbf{a}} = a_x\,\vec{\mathbf{i}} + a_y\,\vec{\mathbf{j}} + a_z\,\vec{\mathbf{k}}$$

$$\begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\cdot\mathbf{R}_{ab}$$

$$\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}(\mathbf{R}_{ab})^{-1} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}(\mathbf{R}_{ab})^{T} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\begin{bmatrix} n_x & n_y & n_z \\ o_x & o_y & o_z \\ a_x & a_y & a_z \end{bmatrix} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\cdot\mathbf{R}_{ba}$$

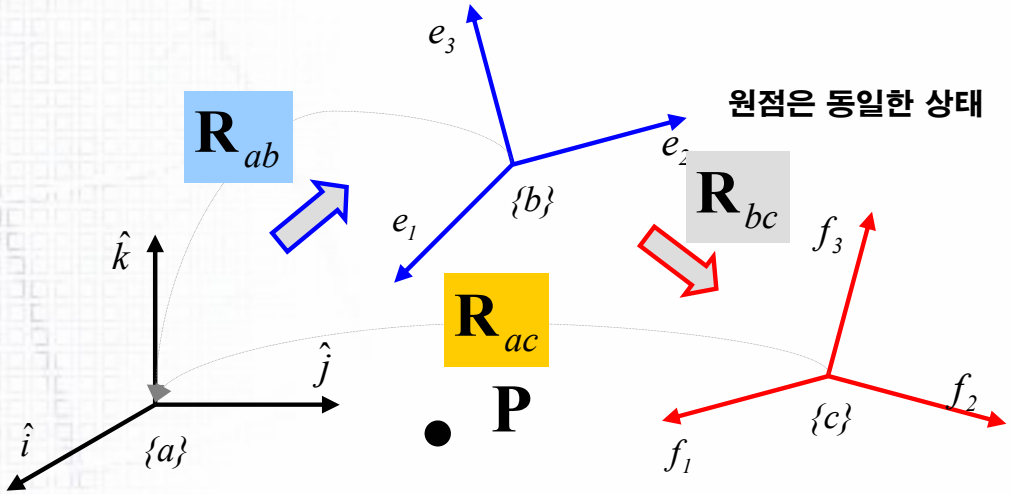
■ 행렬 $\mathbf{R}_{ba}$는 점Q의 좌표계 $\{a\}$에 대한 좌표값 $\mathbf{Q}_a=(Q_{xa}, Q_{ya}, Q_{za})$를
좌표계 $\{b\}$에 대한 좌표값 $\mathbf{Q}_b=(Q_{xb}, Q_{yb}, Q_{zb})$로 대응시키는 **'매핑변환행렬'** 이다

$$\mathbf{Q} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} Q_{xa} \\ Q_{ya} \\ Q_{za} \end{bmatrix} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\begin{bmatrix} n_x & n_y & n_z \\ o_x & o_y & o_z \\ a_x & a_y & a_z \end{bmatrix}\begin{bmatrix} Q_{xa} \\ Q_{ya} \\ Q_{za} \end{bmatrix} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \end{bmatrix}$$

$$\therefore \begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \end{bmatrix} = \mathbf{R}_{ba}\cdot\begin{bmatrix} Q_{xa} \\ Q_{ya} \\ Q_{za} \end{bmatrix}$$

2.2.4 원점이 일치하는 세 개 이상의 좌표계 사이의 매핑변환문제 (1)

세 개의 좌표계사이의 회전 변환



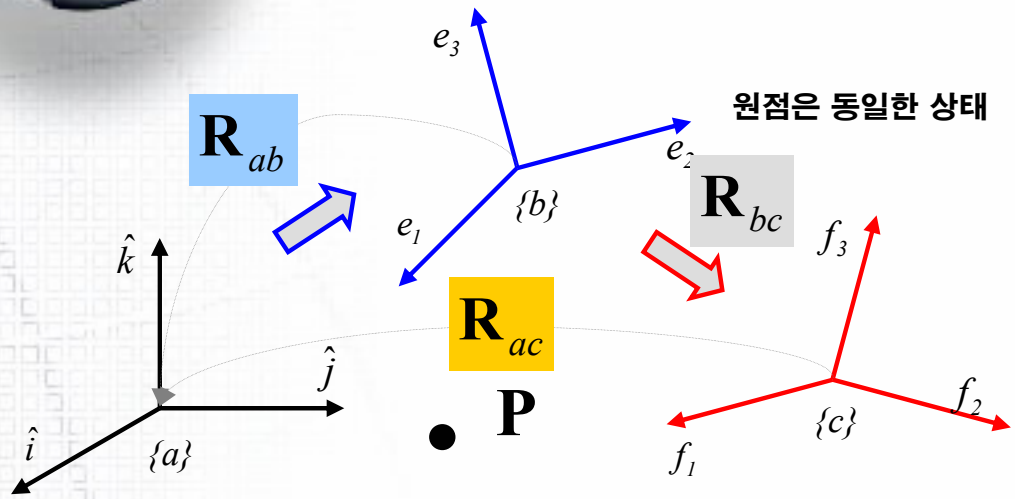
■ 좌표계 $\{a\}$를 행렬 $\mathbf{R}_{ab}$ 만큼 회전하여 좌표계 $\{b\}$를 생성하고,
다시 좌표계 $\{b\}$를 행렬 $\mathbf{R}_{bc}$ 만큼 회전하여 좌표계 $\{c\}$를 생성하였을 때,

■ 좌표계 $\{b\}$의 좌표를 좌표계$\{a\}$의 좌표로 대응시키는 매핑변환행렬은 $\mathbf{R}_{ab}$,
좌표계 $\{c\}$의 좌표를 좌표계$\{b\}$의 좌표로 대응시키는 매핑변환행렬은 $\mathbf{R}_{bc}$ 이다

■ 좌표계 $\{c\}$의 좌표를 좌표계$\{a\}$의 좌표로 대응시키는 매핑변환행렬 $\mathbf{R}_{ac}$ 는
다음과 같이 구할 수 있다

$$\mathbf{R}_{ac} = \mathbf{R}_{ab} \cdot \mathbf{R}_{bc}$$

2.2.4 원점이 일치하는 세 개 이상의 좌표계 사이의 매핑변환문제 (2)



$$\mathbf{R}_{ac} = \mathbf{R}_{ab} \cdot \mathbf{R}_{bc}$$

$$\begin{bmatrix} P_{xa} \\ P_{ya} \\ P_{ya} \end{bmatrix} = \mathbf{R}_{ac} \cdot \begin{bmatrix} P_{xc} \\ P_{yc} \\ P_{yc} \end{bmatrix} = \mathbf{R}_{ab} \cdot \mathbf{R}_{bc} \cdot \begin{bmatrix} P_{xc} \\ P_{yc} \\ P_{yc} \end{bmatrix}$$

[증명]

$$[e_1 \quad e_2 \quad e_3] = [i \quad j \quad k]\,\mathbf{R}_{ab}$$

$$[f_1 \quad f_2 \quad f_3] = [e_1 \quad e_2 \quad e_3]\,\mathbf{R}_{bc}$$

$$\mathbf{P} = [f_1 \quad f_2 \quad f_3]\begin{bmatrix} P_{xc} \\ P_{yc} \\ P_{zc} \end{bmatrix} = [e_1 \quad e_2 \quad e_3]\,\mathbf{R}_{bc}\begin{bmatrix} P_{xc} \\ P_{yc} \\ P_{zc} \end{bmatrix}$$

$$= [i \quad j \quad k]\,\mathbf{R}_{ab}\mathbf{R}_{bc}\begin{bmatrix} P_{xc} \\ P_{yc} \\ P_{zc} \end{bmatrix}$$

$$= [i \quad j \quad k]\,\mathbf{R}_{ac}\begin{bmatrix} P_{xc} \\ P_{yc} \\ P_{zc} \end{bmatrix}$$

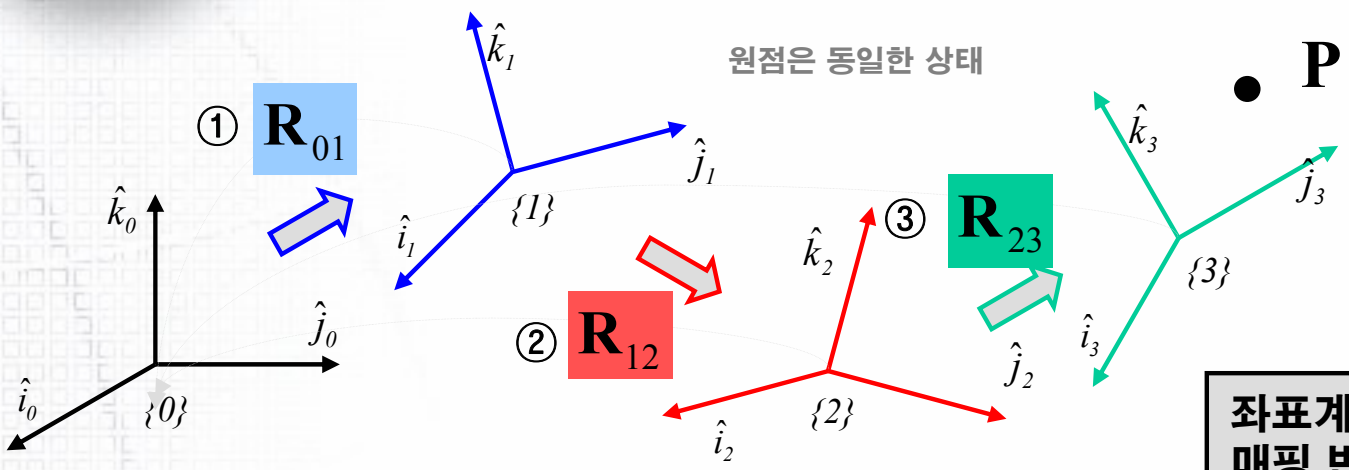
$$\therefore [f_1 \quad f_2 \quad f_3] = [i \quad j \quad k]\,\mathbf{R}_{ab}\mathbf{R}_{bc} = [i \quad j \quad k]\,\mathbf{R}_{ac}$$

$$\therefore \mathbf{R}_{ac} = \mathbf{R}_{ab} \cdot \mathbf{R}_{bc}$$

2.2.5 원점이 일치하는 세 개 이상의 좌표계 사이의 매핑변환문제 (3)

네 개의 좌표계사이의 회전 변환

점 P의 절대위치는 변하지 않는다.



$$\mathbf{R}_{03} = \mathbf{R}_{01} \cdot \mathbf{R}_{12} \cdot \mathbf{R}_{23}$$

$$\begin{bmatrix} P_{x0} \\ P_{y0} \\ P_{y0} \end{bmatrix} \overset{①}{\overset{②}{\overset{③}{=}}} \mathbf{R}_{01} \cdot \mathbf{R}_{12} \cdot \mathbf{R}_{23} \cdot \begin{bmatrix} P_{x3} \\ P_{y3} \\ P_{y3} \end{bmatrix}$$

좌표계의 연속된 회전변환에 따른 매핑 변환행렬은 각 좌표계 회전변환 행렬을 순서대로으로 누적해 곱한 것과 같다

$$[\vec{i}_1 \quad \vec{j}_1 \quad \vec{k}_1] = [\vec{i}_0 \quad \vec{j}_0 \quad \vec{k}_0]\,\mathbf{R}_{01}$$

$$[\vec{i}_2 \quad \vec{j}_2 \quad \vec{k}_2] = [\vec{i}_1 \quad \vec{j}_1 \quad \vec{k}_1]\,\mathbf{R}_{12}$$

$$[\vec{i}_3 \quad \vec{j}_3 \quad \vec{k}_3] = [\vec{i}_2 \quad \vec{j}_2 \quad \vec{k}_2]\,\mathbf{R}_{23}$$

$$[\vec{i}_3 \quad \vec{j}_3 \quad \vec{k}_3] \begin{bmatrix} P_{x_3} \\ P_{y_3} \\ P_{z_3} \end{bmatrix} = [\vec{i}_2 \quad \vec{j}_2 \quad \vec{k}_2]\mathbf{R}_{23} \begin{bmatrix} P_{x_3} \\ P_{y_3} \\ P_{z_3} \end{bmatrix}$$

$$= [\vec{i}_1 \quad \vec{j}_1 \quad \vec{k}_1]\mathbf{R}_{12}\mathbf{R}_{23} \begin{bmatrix} P_{x_3} \\ P_{y_3} \\ P_{z_3} \end{bmatrix}$$

$$= [\vec{i}_0 \quad \vec{j}_0 \quad \vec{k}_0]\mathbf{R}_{01}\mathbf{R}_{12}\mathbf{R}_{23} \begin{bmatrix} P_{x_3} \\ P_{y_3} \\ P_{z_3} \end{bmatrix}$$

$$\therefore \mathbf{R}_{03} = \mathbf{R}_{01} \cdot \mathbf{R}_{12} \cdot \mathbf{R}_{23}$$

2.3 원점이 일치하지 않는 두 좌표계간의 매핑변환 문제

2.3.1 원점이 일치하지 않는 두 좌표계간의 매핑변환 문제

2.3.2 좌표계 원점의 동차좌표 표현

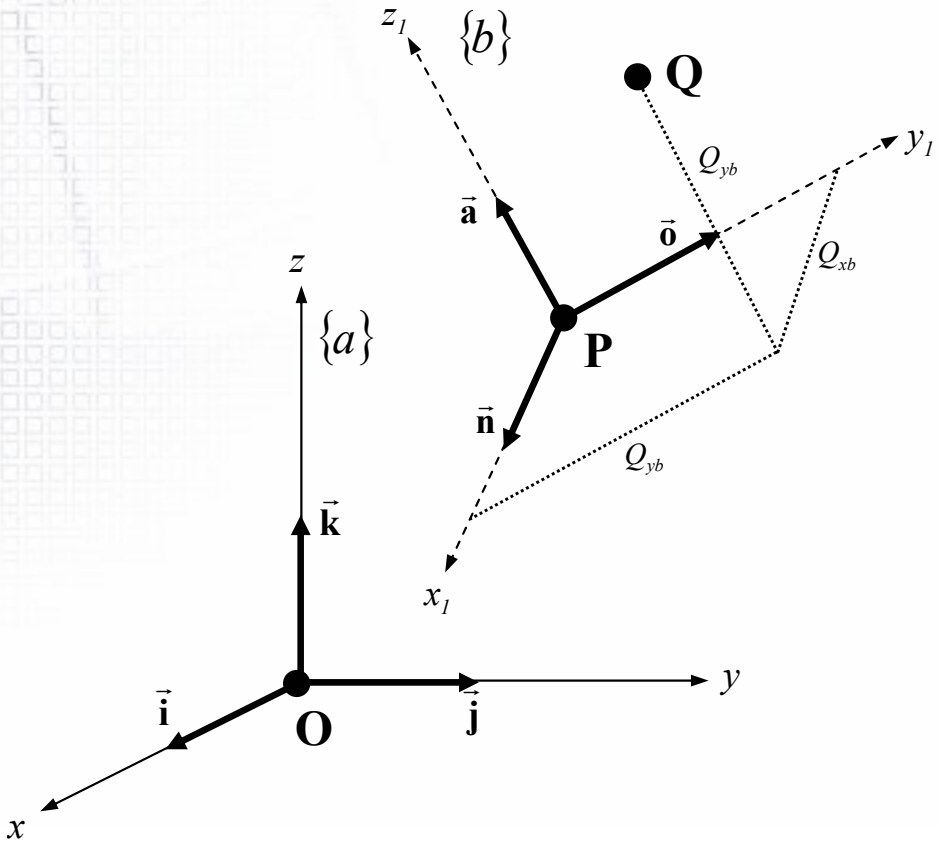
2.3.3 원점이 일치하지 않는 두 좌표계간의 매핑변환 행렬

2.3.4 원점이 일치하지 않는 세 개 이상의 좌표계 사이의 매핑변환 문제

Advanced
Ship
Design
Automation
Laboratory

2.3.1 원점이 일치하지 않는 두 좌표계간의 매핑변환문제

- 원점이 일치하지 않는 두 개의 좌표계 $\{a\}$와 $\{b\}$ 가 있을 때,
- 한 좌표계상의 어느 점의 좌표값을 알 때,
  다른 좌표계에서는 어떠한 좌표값에 해당되는 지를 계산하는 문제



$$\mathbf{P} = P_{xa}\,\vec{\mathbf{i}} + P_{ya}\,\vec{\mathbf{j}} + P_{za}\,\vec{\mathbf{k}} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix}P_{xa}\\ P_{ya}\\ P_{za}\end{bmatrix}$$

$$\mathbf{Q} = Q_{xb}\,\vec{\mathbf{n}} + Q_{yb}\,\vec{\mathbf{o}} + Q_{zb}\,\vec{\mathbf{a}} + \mathbf{P}$$

$$= \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\begin{bmatrix}Q_{xb}\\ Q_{yb}\\ Q_{zb}\end{bmatrix} + \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix}P_{xa}\\ P_{ya}\\ P_{za}\end{bmatrix}$$

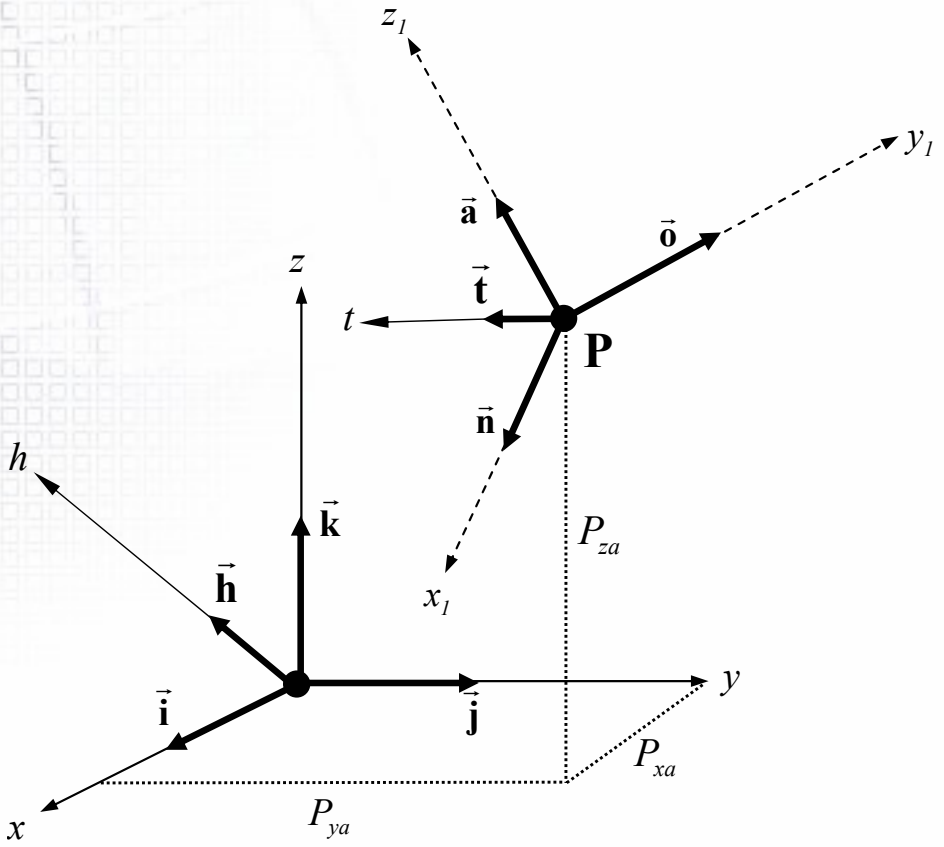
$$= \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix}Q_{xa}\\ Q_{ya}\\ Q_{za}\end{bmatrix}$$

$$= Q_{xa}\,\vec{\mathbf{i}} + Q_{ya}\,\vec{\mathbf{j}} + Q_{za}\,\vec{\mathbf{k}}$$

$$\mathbf{Q}_b = \begin{bmatrix}Q_{xb} & Q_{yb} & Q_{zb}\end{bmatrix}^{\mathrm{T}} \Leftrightarrow \mathbf{Q}_a = \begin{bmatrix}Q_{xa} & Q_{ya} & Q_{za}\end{bmatrix}^{\mathrm{T}}$$

2.3.2 좌표계 원점의 동차좌표 표현

■ 원점이 일치하지 않는 두 개의 좌표계 $\{a\}$와 $\{b\}$ 가 있을 때,
좌표계 $\{b\}$ 의 원점 **P**를 동차좌표로 표현해 보자



■ 점 **P**를 좌표계 $\{a\}$ 의 동차좌표로 표현하면,

$$\mathbf{P} = P_{xa}\,\vec{\mathbf{i}} + P_{ya}\,\vec{\mathbf{j}} + P_{za}\,\vec{\mathbf{k}} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix}P_{xa}\\ P_{ya}\\ P_{za}\end{bmatrix}$$

$$\mathbf{P}^h = P_{xa}\,\vec{\mathbf{i}} + P_{ya}\,\vec{\mathbf{j}} + P_{za}\,\vec{\mathbf{k}} + 1\cdot\vec{\mathbf{h}} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}\begin{bmatrix}P_{xa}\\ P_{ya}\\ P_{za}\\ 1\end{bmatrix}$$

■ 점 **P**은 좌표계 $\{b\}$ 의 원점이므로 좌표값은 **(0,0,0)** 이고, 이를 동차좌표로 표현하면,

$$\mathbf{P} = 0\cdot\vec{\mathbf{n}} + 0\cdot\vec{\mathbf{o}} + 0\cdot\vec{\mathbf{a}} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}$$

$$\mathbf{P}^h = 0\cdot\vec{\mathbf{n}} + 0\cdot\vec{\mathbf{o}} + 0\cdot\vec{\mathbf{a}} + 1\cdot\vec{\mathbf{t}} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}} & \vec{\mathbf{t}}\end{bmatrix}\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{1}\end{bmatrix} = \vec{\mathbf{t}}$$

2.3.3 원점이 일치하지 않는 두 좌표계 사이의 매핑변환행렬 (1)

■ 좌표계 {*b*} 의 단위벡터 **n, o, a**가 다음과 같이 좌표계 {*a*} 의 단위벡터 **i, j, k**로 표현된다면,

$$\vec{\mathbf{n}} = n_x\,\vec{\mathbf{i}} + n_y\,\vec{\mathbf{j}} + n_z\,\vec{\mathbf{k}}$$
$$\vec{\mathbf{o}} = o_x\,\vec{\mathbf{i}} + o_y\,\vec{\mathbf{j}} + o_z\,\vec{\mathbf{k}}$$
$$\vec{\mathbf{a}} = a_x\,\vec{\mathbf{i}} + a_y\,\vec{\mathbf{j}} + a_z\,\vec{\mathbf{k}}$$
$$\mathbf{P} = P_{xa}\,\vec{\mathbf{i}} + P_{ya}\,\vec{\mathbf{j}} + P_{za}\,\vec{\mathbf{k}}$$

$$\begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \qquad \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\begin{bmatrix} n_x & n_y & n_z \\ o_x & o_y & o_z \\ a_x & a_y & a_z \end{bmatrix}$$

■ 점**Q**의 좌표계 {*b*} 에 대한 좌표값 $\mathbf{Q}_b$가 주어졌을 때, 좌표계 {*a*} 에 대한 좌표값 $\mathbf{Q}_a$는 **?**

$$\mathbf{Q} = Q_{xb}\,\vec{\mathbf{n}} + Q_{yb}\,\vec{\mathbf{o}} + Q_{zb}\,\vec{\mathbf{a}} + \mathbf{P}$$
$$= \begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}}\end{bmatrix}\begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \end{bmatrix} + \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} P_{xa} \\ P_{ya} \\ P_{za} \end{bmatrix}$$
$$= \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix}\begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \end{bmatrix} + \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} P_{xa} \\ P_{ya} \\ P_{za} \end{bmatrix}$$
$$= \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\end{bmatrix}\begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}\begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \end{bmatrix}$$

행렬과의 곱으로 나타낼 수 없음

→ 매핑변환 행렬을 구할 수 없음

$$\mathbf{P}'_a = \mathbf{P}_a + \vec{\mathbf{d}}_a = \begin{bmatrix} P_{xa} + d_{xa} \\ P_{ya} + d_{ya} \\ P_{za} + d_{za} \end{bmatrix}$$
$$= \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}\begin{bmatrix} P_{xa} \\ P_{ya} \\ P_{za} \end{bmatrix}$$

2.3.3 원점이 일치하지 않는 두 좌표계 사이의 매핑변환행렬 (2)

■ 좌표계 $\{b\}$ 의 단위벡터 **n, o, a**와 원점 **P**가
동차좌표계 $\{a\}$ 의 단위벡터 **i, j, k , h** 로 표현된다면,

$$\vec{\mathbf{n}} = n_x\,\vec{\mathbf{i}} + n_y\,\vec{\mathbf{j}} + n_z\,\vec{\mathbf{k}} + 0\cdot\vec{\mathbf{h}}$$
$$\vec{\mathbf{o}} = o_x\,\vec{\mathbf{i}} + o_y\,\vec{\mathbf{j}} + o_z\,\vec{\mathbf{k}} + 0\cdot\vec{\mathbf{h}}$$
$$\vec{\mathbf{a}} = a_x\,\vec{\mathbf{i}} + a_y\,\vec{\mathbf{j}} + a_z\,\vec{\mathbf{k}} + 0\cdot\vec{\mathbf{h}}$$
$$\mathbf{P} = P_{xa}\,\vec{\mathbf{i}} + P_{ya}\,\vec{\mathbf{j}} + P_{za}\,\vec{\mathbf{k}} + 1\cdot\vec{\mathbf{h}}$$
$$= 0\cdot\vec{\mathbf{n}} + 0\cdot\vec{\mathbf{o}} + 0\cdot\vec{\mathbf{a}} + 1\cdot\vec{\mathbf{t}}$$

$$\left[\vec{\mathbf{n}} \quad \vec{\mathbf{o}} \quad \vec{\mathbf{a}} \quad \vec{\mathbf{t}}\right] = \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \begin{bmatrix} n_x & o_x & a_x & P_{xa} \\ n_y & o_y & a_y & P_{ya} \\ n_z & o_z & a_z & P_{za} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \cdot \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & P_{xa} \\ 0 & 1 & 0 & P_{ya} \\ 0 & 0 & 1 & P_{za} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \cdot \mathbf{R}_{ab}$$

■ 행렬 **R**은 두 동차좌표계 사이의 위치와 방향관계를 표현하고 있다

→ 한 좌표계를 원점을 중심으로 **회전**한 후, 두 좌표계의 원점이 일치하도록 **평행**이동하면, 두 좌표계가 일치하도록 만들 수 있다

→ 행렬 **R**을 좌표계 $\{a\}$에서 좌표계 $\{b\}$로의 '**변환행렬**' 이라고 부르고 '$\mathbf{R}_{ab}$'라고 표기한다

2.3.3 원점이 일치하지 않는 두 좌표계 사이의 매핑변환행렬 (3)

■ 좌표계 {*b*} 의 단위벡터 **n, o, a**와 원점 **P**가 동차좌표계 {*a*} 의 단위벡터 **i, j, k , h** 로 표현된다면,

$$\vec{\mathbf{n}} = n_x\,\vec{\mathbf{i}} + n_y\,\vec{\mathbf{j}} + n_z\,\vec{\mathbf{k}} + 0\cdot\vec{\mathbf{h}}$$
$$\vec{\mathbf{o}} = o_x\,\vec{\mathbf{i}} + o_y\,\vec{\mathbf{j}} + o_z\,\vec{\mathbf{k}} + 0\cdot\vec{\mathbf{h}}$$
$$\vec{\mathbf{a}} = a_x\,\vec{\mathbf{i}} + a_y\,\vec{\mathbf{j}} + a_z\,\vec{\mathbf{k}} + 0\cdot\vec{\mathbf{h}}$$
$$\mathbf{P} = P_{xa}\,\vec{\mathbf{i}} + P_{ya}\,\vec{\mathbf{j}} + P_{za}\,\vec{\mathbf{k}} + 1\cdot\vec{\mathbf{h}}$$
$$= 0\cdot\vec{\mathbf{n}} + 0\cdot\vec{\mathbf{o}} + 0\cdot\vec{\mathbf{a}} + 1\cdot\vec{\mathbf{t}}$$

$$\left[\vec{\mathbf{n}} \quad \vec{\mathbf{o}} \quad \vec{\mathbf{a}} \quad \vec{\mathbf{t}}\right] = \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \begin{bmatrix} n_x & o_x & a_x & P_{xa} \\ n_y & o_y & a_y & P_{ya} \\ n_z & o_z & a_z & P_{za} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \cdot \mathbf{R}_{ab}$$

■ 좌표계 {*a*}에서 좌표계 {*b*}로의 변환행렬 $\mathbf{R}_{ab}$ 는
점**Q**의 좌표계 {*b*}에 대한 동차좌표값 $\mathbf{Q}_b=(Q_{xb}, Q_{yb}, Q_{zb}, 1)$를
좌표계 {*a*}에 대한 동차좌표값 $\mathbf{Q}_a=(Q_{xa}, Q_{ya}, Q_{za}, 1)$ 로 대응시키는 '**매핑변환행렬**' 이다

$$\mathbf{Q} = Q_{xb}\,\vec{\mathbf{n}} + Q_{yb}\,\vec{\mathbf{o}} + Q_{zb}\,\vec{\mathbf{a}} + \mathbf{P}$$
$$= \left[\vec{\mathbf{n}} \quad \vec{\mathbf{o}} \quad \vec{\mathbf{a}} \quad \vec{\mathbf{t}}\right] \begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \\ 1 \end{bmatrix}$$
$$= \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \begin{bmatrix} n_x & o_x & a_x & P_{xa} \\ n_y & o_y & a_y & P_{ya} \\ n_z & o_z & a_z & P_{za} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \\ 1 \end{bmatrix} = \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \begin{bmatrix} Q_{xa} \\ Q_{ya} \\ Q_{za} \\ 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} Q_{xa} \\ Q_{ya} \\ Q_{za} \\ 1 \end{bmatrix} = \mathbf{R}_{ab} \cdot \begin{bmatrix} Q_{xb} \\ Q_{yb} \\ Q_{zb} \\ 1 \end{bmatrix}$$

2.3.3 원점이 일치하지 않는 두 좌표계 사이의 매핑변환행렬 (4)

■ 좌표계 {*a*}에서 좌표계 {*b*}로의 변환행렬 $\mathbf{R}_{ab}$ 의 **역변환**인 $\mathbf{R}_{ba}$ 는
점Q의 좌표계 {*a*}에 대한 동차좌표값 $\mathbf{Q}_a=(Q_{xa}, Q_{ya}, Q_{za}, 1)$를
좌표계 {*b*}에 대한 동차좌표값 $\mathbf{Q}_b=(Q_{xb}, Q_{yb}, Q_{zb}, 1)$ 로 대응시키는 '**매핑변환행렬**' 이다

$$\begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}} & \vec{\mathbf{t}}\end{bmatrix}=\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}\begin{bmatrix}n_x & o_x & a_x & P_{xa}\\ n_y & o_y & a_y & P_{ya}\\ n_z & o_z & a_z & P_{za}\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$=\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}\cdot\mathbf{R}_{ab}$$

$$\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}=\begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}} & \vec{\mathbf{t}}\end{bmatrix}\begin{bmatrix}n_x & o_x & a_x & P_{xa}\\ n_y & o_y & a_y & P_{ya}\\ n_z & o_z & a_z & P_{za}\\ 0 & 0 & 0 & 1\end{bmatrix}^{-1}$$

$$=\begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}} & \vec{\mathbf{t}}\end{bmatrix}\cdot\mathbf{R}_{ba}$$

$$\therefore \mathbf{R}_{ba}=\left(\mathbf{R}_{ab}\right)^{-1}$$

$$\mathbf{Q}=Q_{xa}\,\vec{\mathbf{i}}+Q_{ya}\,\vec{\mathbf{j}}+Q_{za}\,\vec{\mathbf{k}}+\mathbf{P}$$

$$=\begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix}\begin{bmatrix}Q_{xa}\\ Q_{ya}\\ Q_{za}\\ 1\end{bmatrix}$$

$$=\begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}} & \vec{\mathbf{t}}\end{bmatrix}\mathbf{R}_{ba}\begin{bmatrix}Q_{xa}\\ Q_{ya}\\ Q_{za}\\ 1\end{bmatrix}=\begin{bmatrix}\vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}} & \vec{\mathbf{t}}\end{bmatrix}\begin{bmatrix}Q_{xb}\\ Q_{yb}\\ Q_{zb}\\ 1\end{bmatrix}$$

$$\therefore\begin{bmatrix}Q_{xb}\\ Q_{yb}\\ Q_{zb}\\ 1\end{bmatrix}=\mathbf{R}_{ba}\cdot\begin{bmatrix}Q_{xa}\\ Q_{ya}\\ Q_{za}\\ 1\end{bmatrix}$$

2.3.3 원점이 일치하지 않는 두 좌표계 사이의 매핑변환행렬 (5)

$$\therefore \mathbf{R}_{ba} = \left(\mathbf{R}_{ab}\right)^{-1}$$

$$= \begin{bmatrix} n_x & o_x & a_x & P_{xa} \\ n_y & o_y & a_y & P_{ya} \\ n_z & o_z & a_z & P_{za} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \left(\begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & P_{xa} \\ 0 & 1 & 0 & P_{ya} \\ 0 & 0 & 1 & P_{za} \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)^{-1}$$

$$= \begin{bmatrix} 1 & 0 & 0 & P_{xa} \\ 0 & 1 & 0 & P_{ya} \\ 0 & 0 & 1 & P_{za} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} 1 & 0 & 0 & -P_{xa} \\ 0 & 1 & 0 & -P_{ya} \\ 0 & 0 & 1 & -P_{za} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} n_x & n_y & n_z & 0 \\ o_x & o_y & o_z & 0 \\ a_x & a_y & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} n_x & n_y & n_z & A \\ o_x & o_y & o_z & B \\ a_x & a_y & a_z & C \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

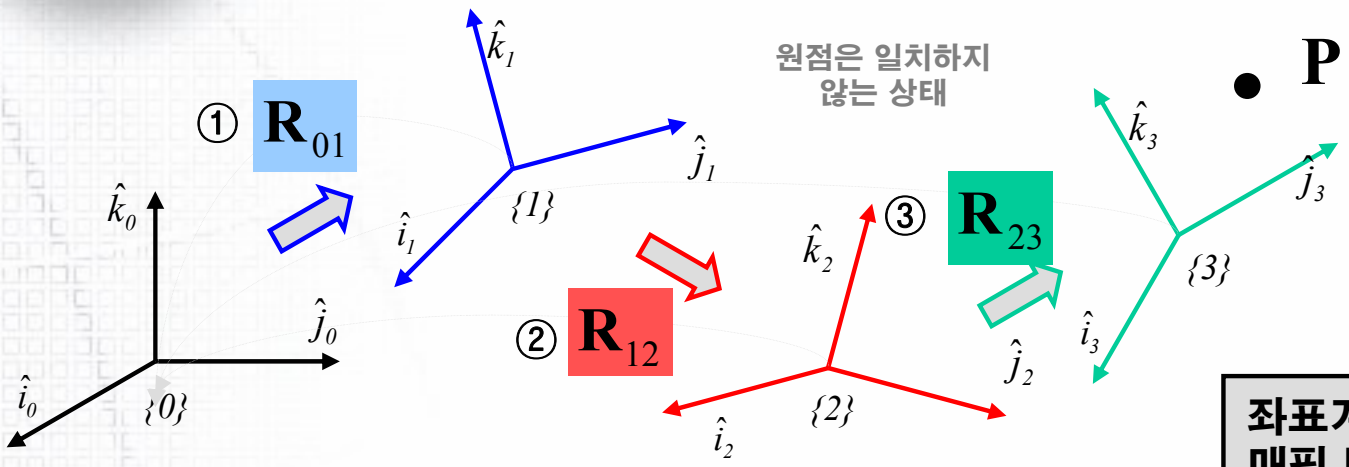
$$A = -P_{xa} \cdot n_x - P_{ya} \cdot o_x - P_{za} \cdot a_x,$$
$$B = -P_{xa} \cdot n_y - P_{ya} \cdot o_y - P_{za} \cdot a_y,$$
$$C = -P_{xa} \cdot n_z - P_{ya} \cdot o_z - P_{za} \cdot a_z$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

2.3.4 원점이 일치하지 않는 세 개 이상의 좌표계 사이의 매핑변환문제

네 개의 좌표계사이의 회전 변환

점 P의 절대위치는 변하지 않는다.



$$\mathbf{R}_{03} = \mathbf{R}_{01} \cdot \mathbf{R}_{12} \cdot \mathbf{R}_{23}$$

$$\begin{bmatrix} P_{x0} \\ P_{y0} \\ P_{z0} \\ 1 \end{bmatrix} = \overset{①}{\mathbf{R}_{01}} \cdot \overset{②}{\mathbf{R}_{12}} \cdot \overset{③}{\mathbf{R}_{23}} \cdot \begin{bmatrix} P_{x3} \\ P_{y3} \\ P_{z3} \\ 1 \end{bmatrix}$$

좌표계의 연속된 변환에 따른 매핑 변환행렬은 각 좌표계 변환행렬을 순서대로으로 누적해 곱한 것과 같다

$$\begin{bmatrix} \vec{\mathbf{i}}_1 & \vec{\mathbf{j}}_1 & \vec{\mathbf{k}}_1 & \vec{\mathbf{h}}_1 \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{i}}_0 & \vec{\mathbf{j}}_0 & \vec{\mathbf{k}}_0 & \vec{\mathbf{h}}_0 \end{bmatrix} \mathbf{R}_{01}$$
$$\begin{bmatrix} \vec{\mathbf{i}}_2 & \vec{\mathbf{j}}_2 & \vec{\mathbf{k}}_2 & \vec{\mathbf{h}}_2 \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{i}}_1 & \vec{\mathbf{j}}_1 & \vec{\mathbf{k}}_1 & \vec{\mathbf{h}}_1 \end{bmatrix} \mathbf{R}_{12}$$
$$\begin{bmatrix} \vec{\mathbf{i}}_3 & \vec{\mathbf{j}}_3 & \vec{\mathbf{k}}_3 & \vec{\mathbf{h}}_3 \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{i}}_2 & \vec{\mathbf{j}}_2 & \vec{\mathbf{k}}_2 & \vec{\mathbf{h}}_2 \end{bmatrix} \mathbf{R}_{23}$$

$$\mathbf{P}_{\{3\}} = \begin{bmatrix} \vec{\mathbf{i}}_3 & \vec{\mathbf{j}}_3 & \vec{\mathbf{k}}_3 & \vec{\mathbf{h}}_3 \end{bmatrix} \begin{bmatrix} P_{x_3} \\ P_{y_3} \\ P_{z_3} \\ 1 \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{i}}_2 & \vec{\mathbf{j}}_2 & \vec{\mathbf{k}}_2 & \vec{\mathbf{h}}_2 \end{bmatrix} \mathbf{R}_{23} \begin{bmatrix} P_{x_3} \\ P_{y_3} \\ P_{z_3} \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \vec{\mathbf{i}}_1 & \vec{\mathbf{j}}_1 & \vec{\mathbf{k}}_1 & \vec{\mathbf{h}}_1 \end{bmatrix} \mathbf{R}_{12}\mathbf{R}_{23} \begin{bmatrix} P_{x_3} \\ P_{y_3} \\ P_{z_3} \\ 1 \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{i}}_0 & \vec{\mathbf{j}}_0 & \vec{\mathbf{k}}_0 & \vec{\mathbf{h}}_0 \end{bmatrix} \mathbf{R}_{01}\mathbf{R}_{12}\mathbf{R}_{23} \begin{bmatrix} P_{x_3} \\ P_{y_3} \\ P_{z_3} \\ 1 \end{bmatrix} = \mathbf{P}_{\{0\}}$$

$$\therefore \mathbf{R}_{03} = \mathbf{R}_{01} \cdot \mathbf{R}_{12} \cdot \mathbf{R}_{23}$$

Ch 3. 3차원 형상변환을 이용한 로봇의 정/역기구학

3.1 기본 이론

3.2 3차원 형상변환을 이용한 로봇의 정/역기구학

Advanced
Ship
Design
Automation
Laboratory

3.1 기본 이론

3.1.1 기구학

3.1.2 연속적으로 움직이는 좌표계 사이의 매핑변환 행렬

3.1.3 고정된 기준좌표계에 대한 매핑변환행렬

Advanced
Ship
Design
Automation
Laboratory

3.1.1 기구학 (機構學, Theory of mechanism, Robotics, Kinematics)

■ 기계(로봇)를 구성하는 각 요소들이 상호 연계되어 이루어지는 위치와 방향을 파악하는 방법

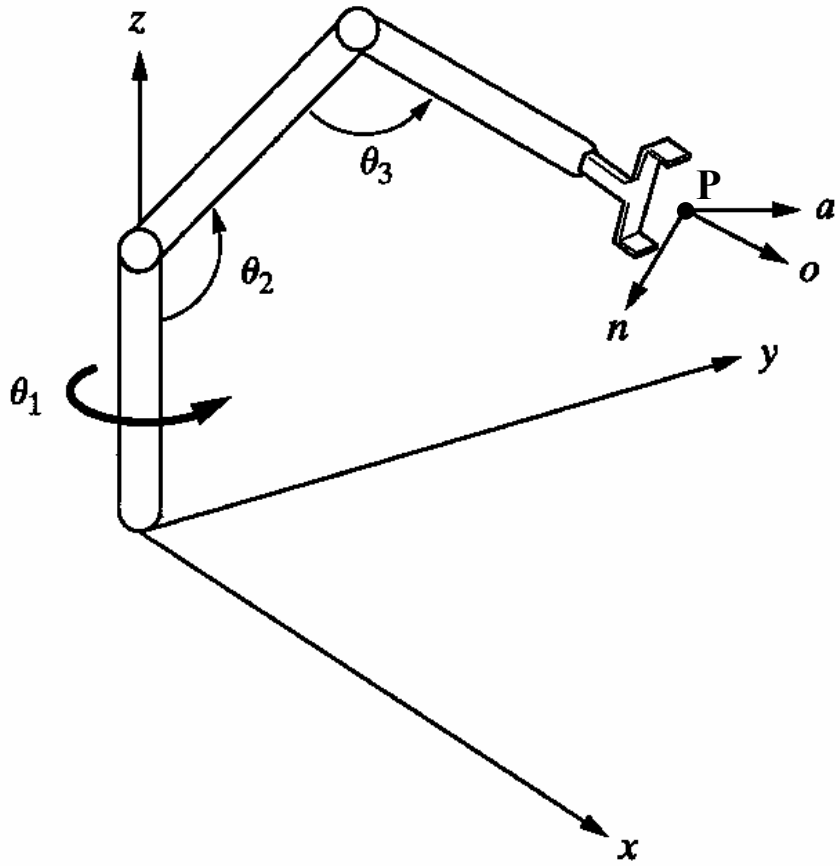
오른쪽 그림과 같은 **3**축 로봇에 대해서

■ 정기구학(Forward Kinematics)

- 로봇의 **3**축의 각도가 주어졌을 때, **end effect**(로봇의끝점)의 위치와 방향을 계산하는 방법
- **Given:** $\theta_1, \theta_2, \theta_3$ → **Find:** $\mathbf{P}, \mathbf{n}, \mathbf{o}, \mathbf{a}^*$

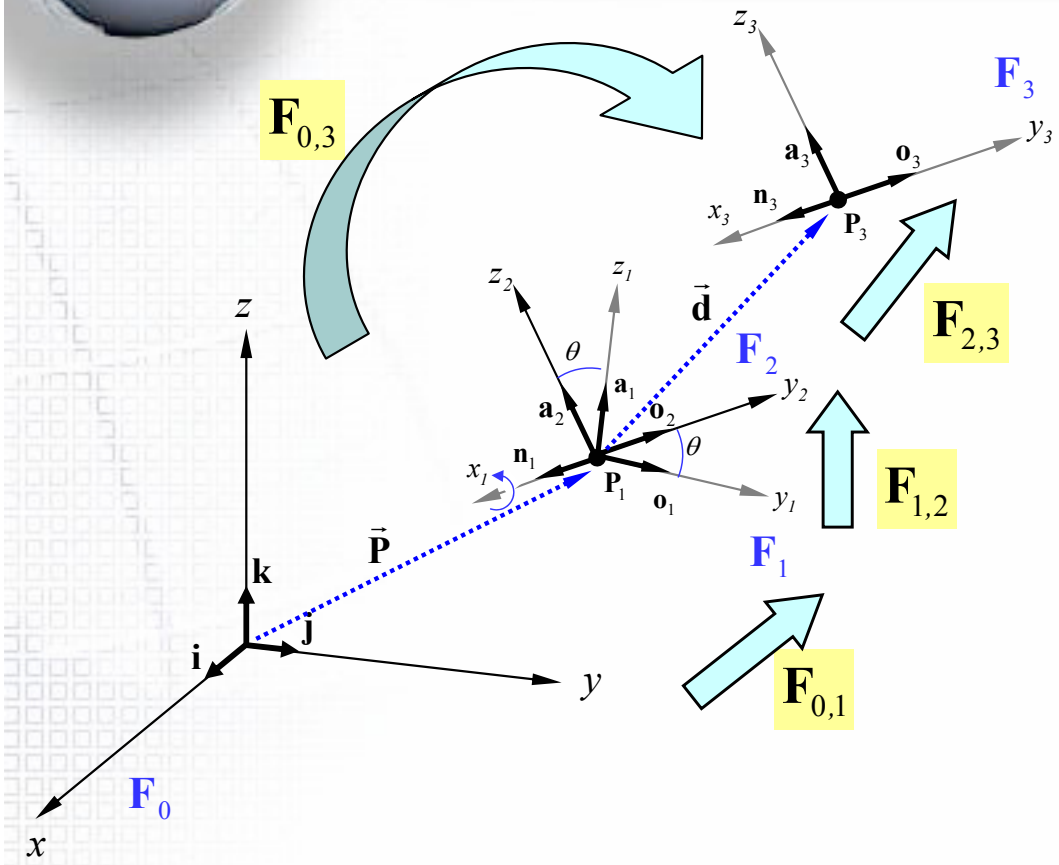
■ 역기구학(Inverse Kinematics)

- **End effect**의 위치와 방향이 주어졌을 때, 로봇의 **3**축의 각도를 계산하는 방법
- **Given:** $\mathbf{P}, \mathbf{n}, \mathbf{o}, \mathbf{a}$ → **Find:** $\theta_1, \theta_2, \theta_3$



참고: **6**축 로봇

3.1.2 연속적으로 움직이는 좌표계 사이의 변환행렬 (1)



- 좌표계 $\mathbf{F}_0$ 을 좌표계 $\mathbf{F}_1$ 로 변환하는 행렬 $\mathbf{F}_{0,1}$

$$\begin{bmatrix}\vec{\mathbf{n}}_1 & \vec{\mathbf{o}}_1 & \vec{\mathbf{a}}_1 & \vec{\mathbf{t}}_1\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix} \begin{bmatrix}1 & 0 & 0 & P_{x0}\\ 0 & 1 & 0 & P_{y0}\\ 0 & 0 & 1 & P_{z0}\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$= \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix} \cdot \mathbf{F}_{0,1}$$

$$= \begin{bmatrix}\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} & \vec{\mathbf{h}}\end{bmatrix} \cdot Trans(P_{x0}, P_{y0}, P_{z0})$$

- 좌표계 $\mathbf{F}_1$ 을 좌표계 $\mathbf{F}_2$ 로 변환하는 행렬 $\mathbf{F}_{1,2}$

$$\begin{bmatrix}\vec{\mathbf{n}}_2 & \vec{\mathbf{o}}_2 & \vec{\mathbf{a}}_2 & \vec{\mathbf{t}}_2\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{n}}_1 & \vec{\mathbf{o}}_1 & \vec{\mathbf{a}}_1 & \vec{\mathbf{t}}_1\end{bmatrix} \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & -\sin\theta & 0\\ 0 & \sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$= \begin{bmatrix}\vec{\mathbf{n}}_1 & \vec{\mathbf{o}}_1 & \vec{\mathbf{a}}_1 & \vec{\mathbf{t}}_1\end{bmatrix} \cdot \mathbf{F}_{1,2}$$

$$= \begin{bmatrix}\vec{\mathbf{n}}_1 & \vec{\mathbf{o}}_1 & \vec{\mathbf{a}}_1 & \vec{\mathbf{t}}_1\end{bmatrix} \cdot Rot(x, \theta)$$

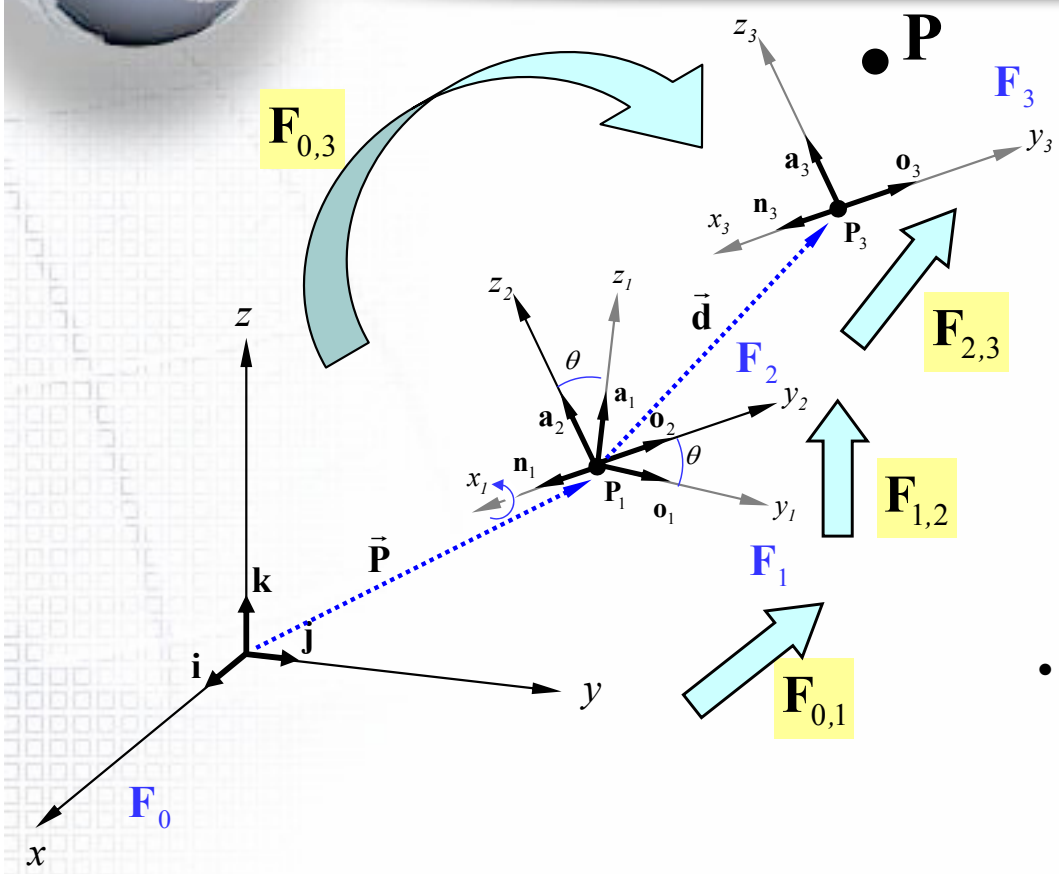
- 좌표계 $\mathbf{F}_2$ 을 좌표계 $\mathbf{F}_3$ 으로 변환하는 행렬 $\mathbf{F}_{2,3}$

$$\begin{bmatrix}\vec{\mathbf{n}}_3 & \vec{\mathbf{o}}_3 & \vec{\mathbf{a}}_3 & \vec{\mathbf{t}}_3\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{n}}_2 & \vec{\mathbf{o}}_2 & \vec{\mathbf{a}}_2 & \vec{\mathbf{t}}_2\end{bmatrix} \begin{bmatrix}1 & 0 & 0 & d_{x2}\\ 0 & 1 & 0 & d_{y2}\\ 0 & 0 & 1 & d_{y2}\\ 0 & 0 & 0 & d_{y2}\end{bmatrix}$$

$$= \begin{bmatrix}\vec{\mathbf{n}}_2 & \vec{\mathbf{o}}_2 & \vec{\mathbf{a}}_2 & \vec{\mathbf{t}}_2\end{bmatrix} \cdot \mathbf{F}_{2,3}$$

$$= \begin{bmatrix}\vec{\mathbf{n}}_2 & \vec{\mathbf{o}}_2 & \vec{\mathbf{a}}_2 & \vec{\mathbf{t}}_2\end{bmatrix} \cdot Trans(d_{x2}, d_{y2}, d_{z2})$$

3.1.2 연속적으로 움직이는 좌표계 사이의 변환행렬 (2)



$$[\vec{n}_1 \quad \vec{o}_1 \quad \vec{a}_1 \quad \vec{t}_1] = [\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}] \cdot F_{0,1}$$
$$= [\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}] \cdot Trans(P_{x0}, P_{y0}, P_{z0})$$

$$[\vec{n}_2 \quad \vec{o}_2 \quad \vec{a}_2 \quad \vec{t}_2] = [\vec{n}_1 \quad \vec{o}_1 \quad \vec{a}_1 \quad \vec{t}_1] \cdot F_{1,2}$$
$$= [\vec{n}_1 \quad \vec{o}_1 \quad \vec{a}_1 \quad \vec{t}_1] \cdot Rot(x, \theta)$$

$$[\vec{n}_3 \quad \vec{o}_3 \quad \vec{a}_3 \quad \vec{t}_3] = [\vec{n}_2 \quad \vec{o}_2 \quad \vec{a}_2 \quad \vec{t}_2] \cdot F_{2,3}$$
$$= [\vec{n}_2 \quad \vec{o}_2 \quad \vec{a}_2 \quad \vec{t}_2] \cdot Trans(d_{x2}, d_{y2}, d_{z2})$$

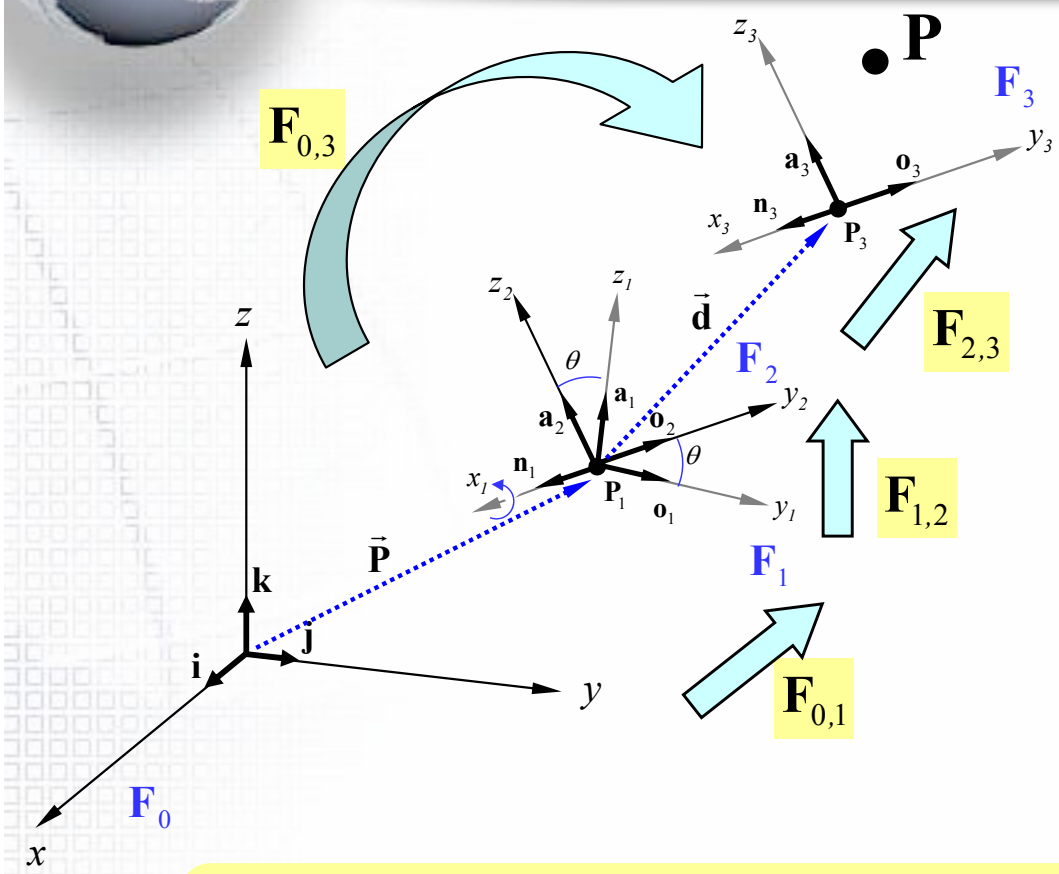
- 좌표계 F_3 을 좌표계 F_0 로 변환하는 행렬 $F_{0,3}$

$$[\vec{n}_3 \quad \vec{o}_3 \quad \vec{a}_3 \quad \vec{t}_3] = [\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}] \cdot F_{0,3}$$

$$[\vec{n}_3 \quad \vec{o}_3 \quad \vec{a}_3 \quad \vec{t}_3] = [\vec{n}_2 \quad \vec{o}_2 \quad \vec{a}_2 \quad \vec{t}_2] \cdot F_{2,3}$$
$$= [\vec{n}_1 \quad \vec{o}_1 \quad \vec{a}_1 \quad \vec{t}_1] \cdot F_{1,2} \cdot F_{2,3}$$
$$= [\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}] \cdot F_{0,1} \cdot F_{1,2} \cdot F_{2,3}$$

$$\therefore F_{0,3} = F_{0,1} \cdot F_{1,2} \cdot F_{2,3}$$
$$= Trans(P_{x0}, P_{y0}, P_{z0}) \cdot Rot(x, \theta) \cdot Trans(d_{x2}, d_{y2}, d_{z2})$$

3.1.2 연속적으로 움직이는 좌표계 사이의 변환행렬 (3)



$$\left[\vec{\mathbf{n}}_3 \quad \vec{\mathbf{o}}_3 \quad \vec{\mathbf{a}}_3 \quad \vec{\mathbf{t}}_3\right] = \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \cdot \mathbf{F}_{0,3}$$
$$= \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \cdot \mathbf{F}_{0,1} \cdot \mathbf{F}_{1,2} \cdot \mathbf{F}_{2,3}$$

$$\therefore \mathbf{F}_{0,3} = \mathbf{F}_{0,1} \cdot \mathbf{F}_{1,2} \cdot \mathbf{F}_{2,3}$$
$$= Trans(P_{x0}, P_{y0}, P_{z0}) \cdot Rot(x, \theta) \cdot Trans(d_{x2}, d_{y2}, d_{z2})$$

$$\left[\vec{\mathbf{n}}_3 \quad \vec{\mathbf{o}}_3 \quad \vec{\mathbf{a}}_3 \quad \vec{\mathbf{t}}_3\right] \begin{bmatrix} P_{x3} \\ P_{y3} \\ P_{z3} \\ 1 \end{bmatrix} = \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \cdot \mathbf{F}_{0,1} \cdot \mathbf{F}_{1,2} \cdot \mathbf{F}_{2,3} \begin{bmatrix} P_{x3} \\ P_{y3} \\ P_{z3} \\ 1 \end{bmatrix}$$

$$= \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \begin{bmatrix} P_{x0} \\ P_{y0} \\ P_{z0} \\ 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} P_{x0} \\ P_{y0} \\ P_{z0} \\ 1 \end{bmatrix} = \mathbf{F}_{0,1} \cdot \mathbf{F}_{1,2} \cdot \mathbf{F}_{2,3} \begin{bmatrix} P_{x3} \\ P_{y3} \\ P_{z3} \\ 1 \end{bmatrix}$$
$$= Trans(P_{x0}, P_{y0}, P_{z0}) \cdot Rot(x, \theta) \cdot Trans(d_{x2}, d_{y2}, d_{z2}) \begin{bmatrix} P_{x3} \\ P_{y3} \\ P_{z3} \\ 1 \end{bmatrix}$$

좌표계의 연속된 변환에 따른 매핑 변환행렬은 각 좌표계 변환행렬을 순서대로으로 누적해 곱한 것과 같다

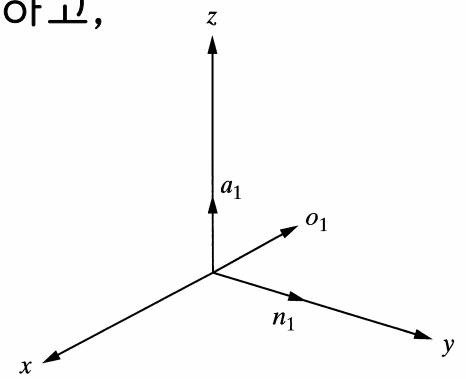
3.1.2 연속적으로 움직이는 좌표계 사이의 변환행렬 (4)

■ 예제 풀이

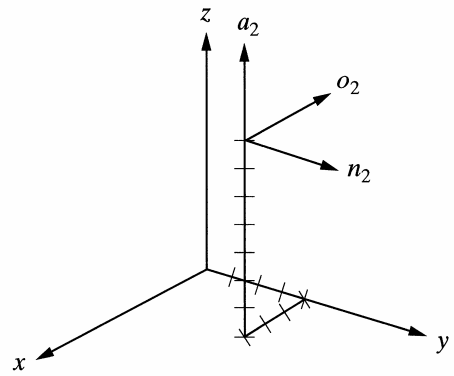
- 좌표계 *noa*는 처음에 좌표계 *xyz*와 일치하고 있다.
  움직이는 좌표계 *noa*에 대해서 다음과 같은 변환을 차례대로 수행하였을 때,
  변환이 끝난 좌표계 *noa* 상의 점 $P_{noa}(7,3,2)$의 좌표계 *xyz*에 대한 좌표값
  P_{xyz}를 계산하여라.
  (1) a축에 대해서 90° 만큼 회전하고,
  (2) n, o, a축을 따라서 [4, −3, 7] 만큼 이동하고,
  (3) o축에 대해서 90° 만큼 회전하라

$$\mathbf{F}_{0,3} = \mathbf{F}_{0,1}\cdot\mathbf{F}_{1,2}\cdot\mathbf{F}_{2,3}$$
$$= \mathbf{Rot}(y,90°)\cdot\mathbf{Trans}(4,-3,7)\cdot\mathbf{Rot}(z,90°)$$
$$= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

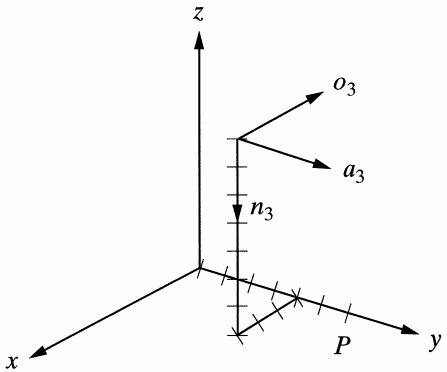
$$\mathbf{F}_{3,0}\cdot\begin{bmatrix} 7 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ 0 \\ 1 \end{bmatrix} \qquad \therefore\ \mathbf{P}_{xyz} = \begin{bmatrix} 0 \\ 6 \\ 0 \end{bmatrix}$$



After the first transformation

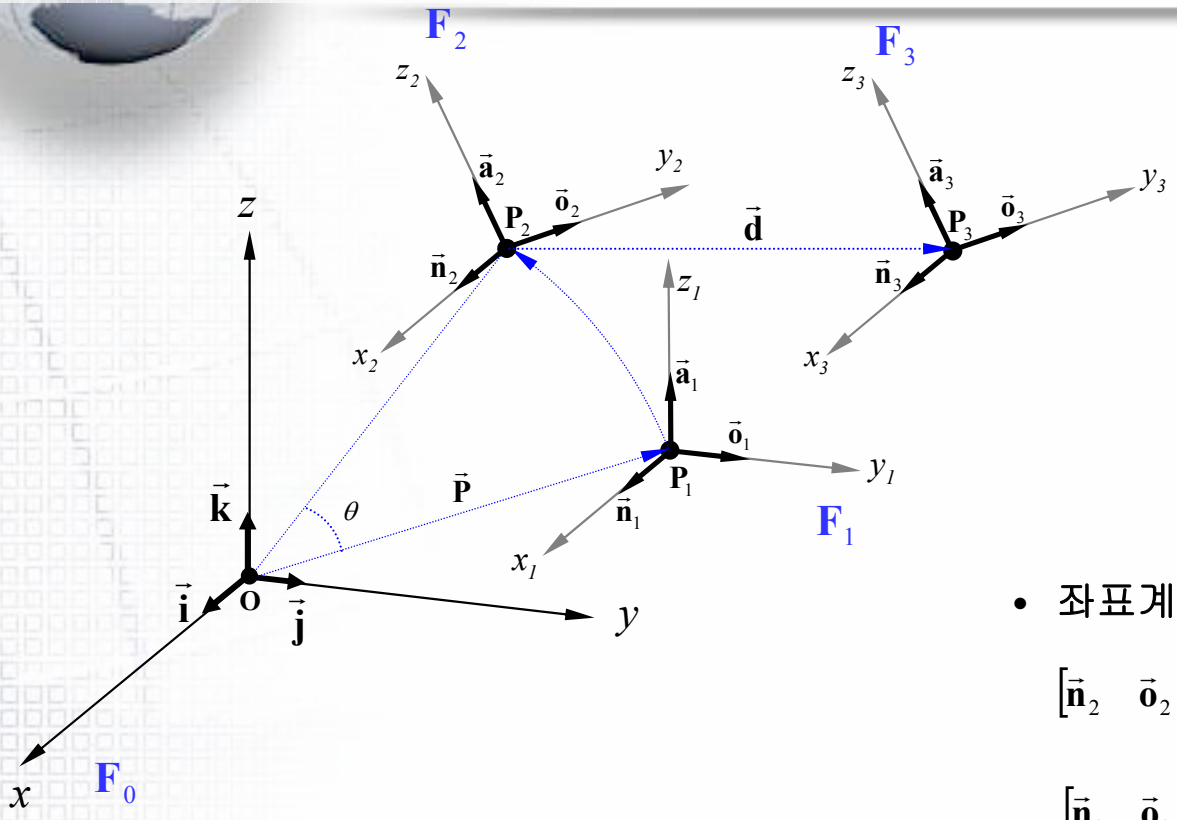


After the second transformation



After the third transformation

3.1.3 고정된 기준 좌표계에 대한 변환행렬 (1)



- 좌표계 F_0 의 i, j, k, h 를 P 만큼 평행이동하는 변환

$$\left[\vec{n}_1 \quad \vec{o}_1 \quad \vec{a}_1 \quad \vec{t}_1\right] = \left[\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}\right] \begin{bmatrix} 1 & 0 & 0 & P_{x0} \\ 0 & 1 & 0 & P_{y0} \\ 0 & 0 & 1 & P_{z0} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \left[\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}\right] \cdot Trans(P_{x0}, P_{y0}, P_{z0})$$

- 좌표계 F_1 를 좌표계 F_0의 x축에 대해서 회전하는 변환

$$\left[\vec{n}_2 \quad \vec{o}_2 \quad \vec{a}_2 \quad \vec{t}_2\right] = \left[\vec{n}_1 \quad \vec{o}_1 \quad \vec{a}_1 \quad \vec{t}_1\right] \cdot ?$$

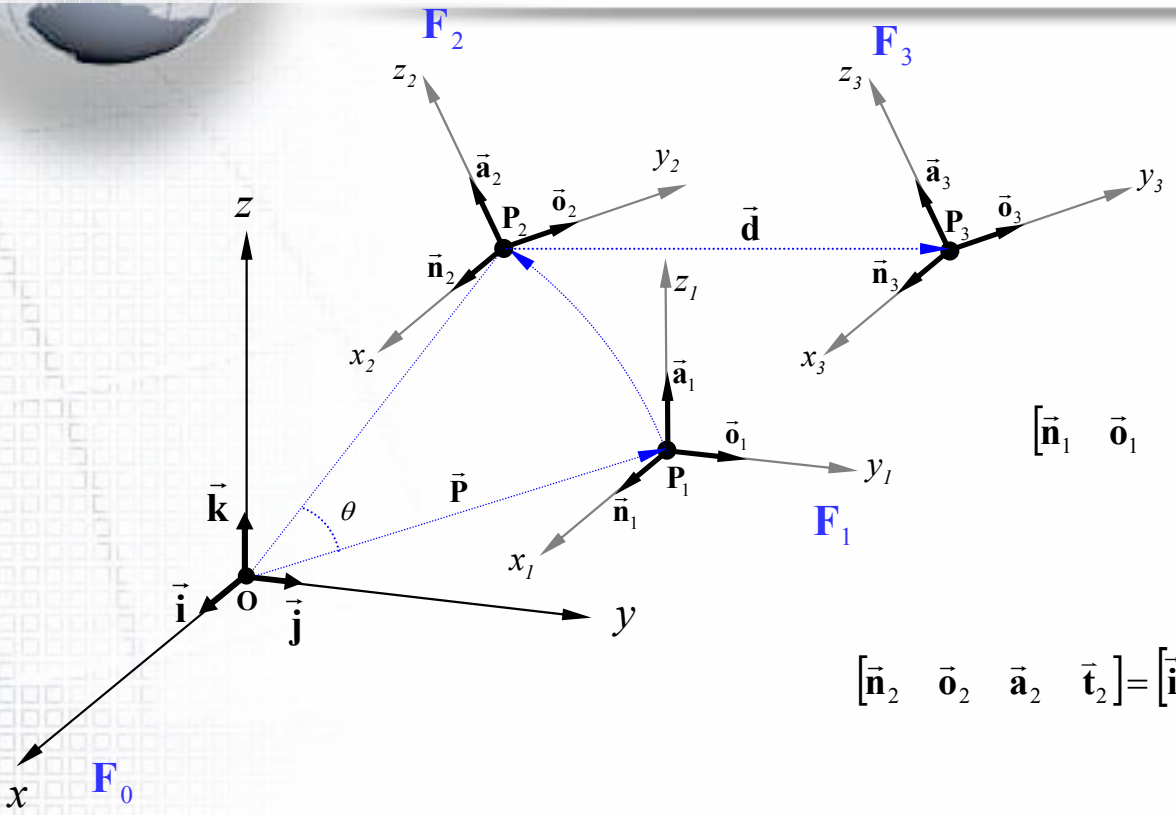
$$\left[\vec{n}_1 \quad \vec{o}_1 \quad \vec{a}_1 \quad \vec{t}_1\right] = \left[\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}\right] \cdot Trans(P_{x0}, P_{y0}, P_{z0}) \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \left[\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}\right] \cdot \begin{bmatrix} n_{x0} & o_{x0} & a_{x0} & P_{x0} \\ n_{y0} & o_{y0} & a_{y0} & P_{y0} \\ n_{z0} & o_{z0} & a_{z0} & P_{z0} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\therefore \left[\vec{n}_2 \quad \vec{o}_2 \quad \vec{a}_2 \quad \vec{t}_2\right] = \left[\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}\right] \cdot Rot(x, \theta) \cdot \begin{bmatrix} n_{x0} & o_{x0} & a_{x0} & P_{x0} \\ n_{y0} & o_{y0} & a_{y0} & P_{y0} \\ n_{z0} & o_{z0} & a_{z0} & P_{z0} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \left[\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}\right] \cdot Rot(x, \theta) \cdot Trans(P_{x0}, P_{y0}, P_{z0}) \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.1.3 고정된 기준 좌표계에 대한 변환행렬 (2)



$$[\vec{n}_1 \quad \vec{o}_1 \quad \vec{a}_1 \quad \vec{t}_1] = [\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}] \cdot Trans(P_{x0}, P_{y0}, P_{z0}) \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

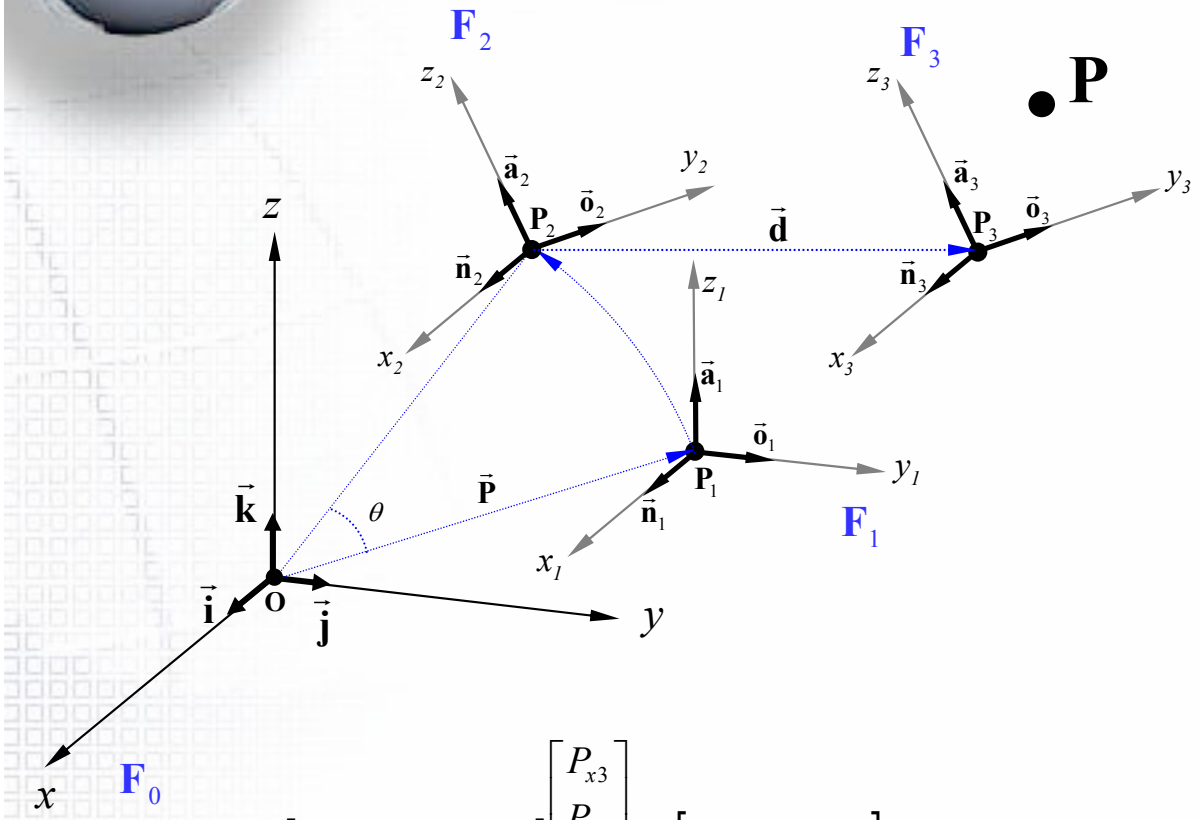
$$[\vec{n}_2 \quad \vec{o}_2 \quad \vec{a}_2 \quad \vec{t}_2] = [\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}] \cdot Rot(x,\theta) \cdot Trans(P_{x0}, P_{y0}, P_{z0}) \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 좌표계 $\mathbf{F}_2$ 를 좌표계 $\mathbf{F}_0$의 좌표로 표현된 벡터 **d**만큼 평행이동 하는 변환

$$[\vec{n}_3 \quad \vec{o}_3 \quad \vec{a}_3 \quad \vec{t}_3] = [\vec{n}_2 \quad \vec{o}_2 \quad \vec{a}_2 \quad \vec{t}_2] \cdot ?$$

$$\therefore [\vec{n}_3 \quad \vec{o}_3 \quad \vec{a}_3 \quad \vec{t}_3] = [\vec{i} \quad \vec{j} \quad \vec{k} \quad \vec{h}] \cdot Trans(d_{x0}, d_{y0}, d_{z0}) \cdot Rot(x,\theta) \cdot Trans(P_{x0}, P_{y0}, P_{z0})$$

3.1.3 고정된 기준 좌표계에 대한 변환행렬 (3)



$$\left[\vec{\mathbf{n}}_3 \quad \vec{\mathbf{o}}_3 \quad \vec{\mathbf{a}}_3 \quad \vec{\mathbf{t}}_3\right]\begin{bmatrix} P_{x3} \\ P_{y3} \\ P_{z3} \\ 1 \end{bmatrix} = \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right] \cdot Trans(d_{x0}, d_{y0}, d_{z0}) \cdot Rot(x,\theta) \cdot Trans(P_{x0}, P_{y0}, P_{z0}) \cdot \begin{bmatrix} P_{x3} \\ P_{y3} \\ P_{z3} \\ 1 \end{bmatrix} = \left[\vec{\mathbf{i}} \quad \vec{\mathbf{j}} \quad \vec{\mathbf{k}} \quad \vec{\mathbf{h}}\right]\begin{bmatrix} P_{x0} \\ P_{y0} \\ P_{z0} \\ 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} P_{x0} \\ P_{y0} \\ P_{z0} \\ 1 \end{bmatrix} = Trans(d_{x0}, d_{y0}, d_{z0}) \cdot Rot(x,\theta) \cdot Trans(P_{x0}, P_{y0}, P_{z0}) \cdot \begin{bmatrix} P_{x3} \\ P_{y3} \\ P_{z3} \\ 1 \end{bmatrix}$$

고정된 기준좌표계에 대한 변환에 따른 매핑 변환행렬은 각 좌표계 변환행렬을 역순으로 누적해 곱한 것과 같다

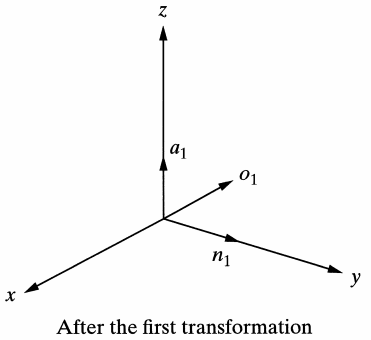
3.1.3 고정된 기준 좌표계에 대한 변환행렬 (4)

■ 예제 풀이

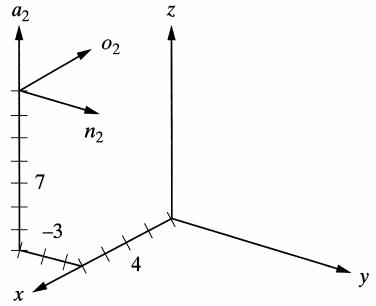
- 좌표계 *noa*는 처음에 좌표계 *xyz*와 일치하고 있다.
  고정 좌표계 *xyz*에 대해서 다음과 같은 변환을 차례대로 수행하였을 때,
  변환이 끝난 좌표계 *noa* 상의 점 $P_{noa}(7,3,2)$의 좌표계 *xyz*에 대한 좌표값 P_{xyz}를 계산하여라.
  (1) z축에 대해서 90° 만큼 회전하고,
  (2) x, y, z축을 따라서 [4, −3, 7] 만큼 이동하고,
  (3) y축에 대해서 90° 만큼 회전하라

$$\mathbf{Rot}_{\mathbf{F}_0}(y,90^\circ)\cdot\mathbf{Trans}_{\mathbf{F}_0}(4,-3,7)\cdot\mathbf{Rot}_{\mathbf{F}_0}(z,90^\circ)\cdot\begin{bmatrix}7\\3\\2\\1\end{bmatrix}=\begin{bmatrix}9\\4\\-1\\1\end{bmatrix}$$

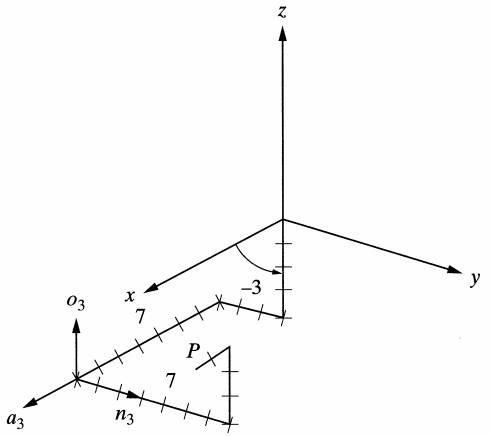
$$\therefore\ \mathbf{P}_{xyz}=\begin{bmatrix}9\\4\\-1\end{bmatrix}$$



After the first transformation



After the second transformation



After the third transformation

3.2 3차원 형상변환을 이용한 로봇의 정/역기구학

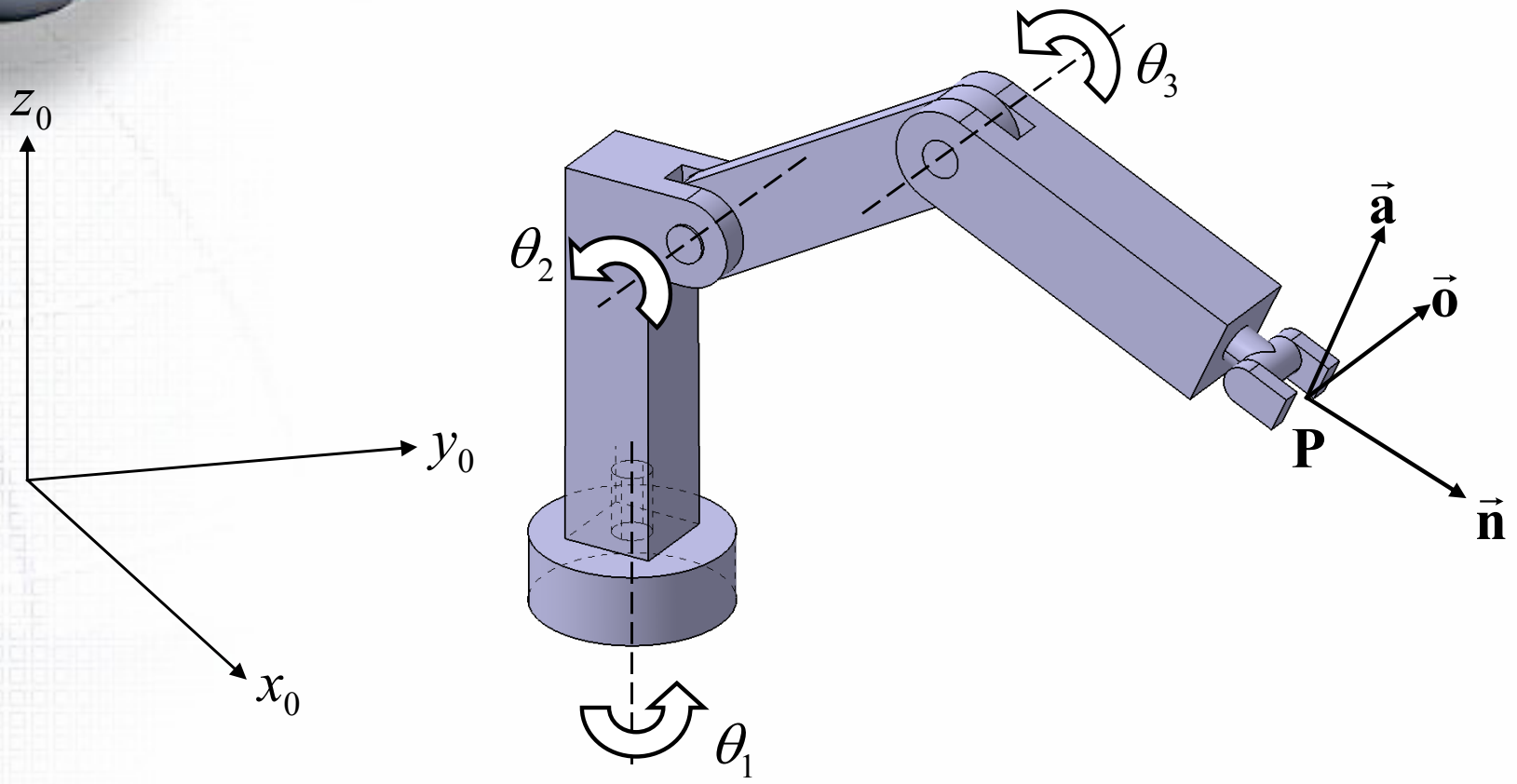
3.2.1 3축 로봇의 형상

3.2.2 3축 로봇의 관절별 좌표계 설정

3.2.3 로봇 축의 각이 주어 졌을때의 정기구학식

3.2.4 정기구학식을 역으로 계산하는 역기구학식

3.2.1 3축 로봇의 형상



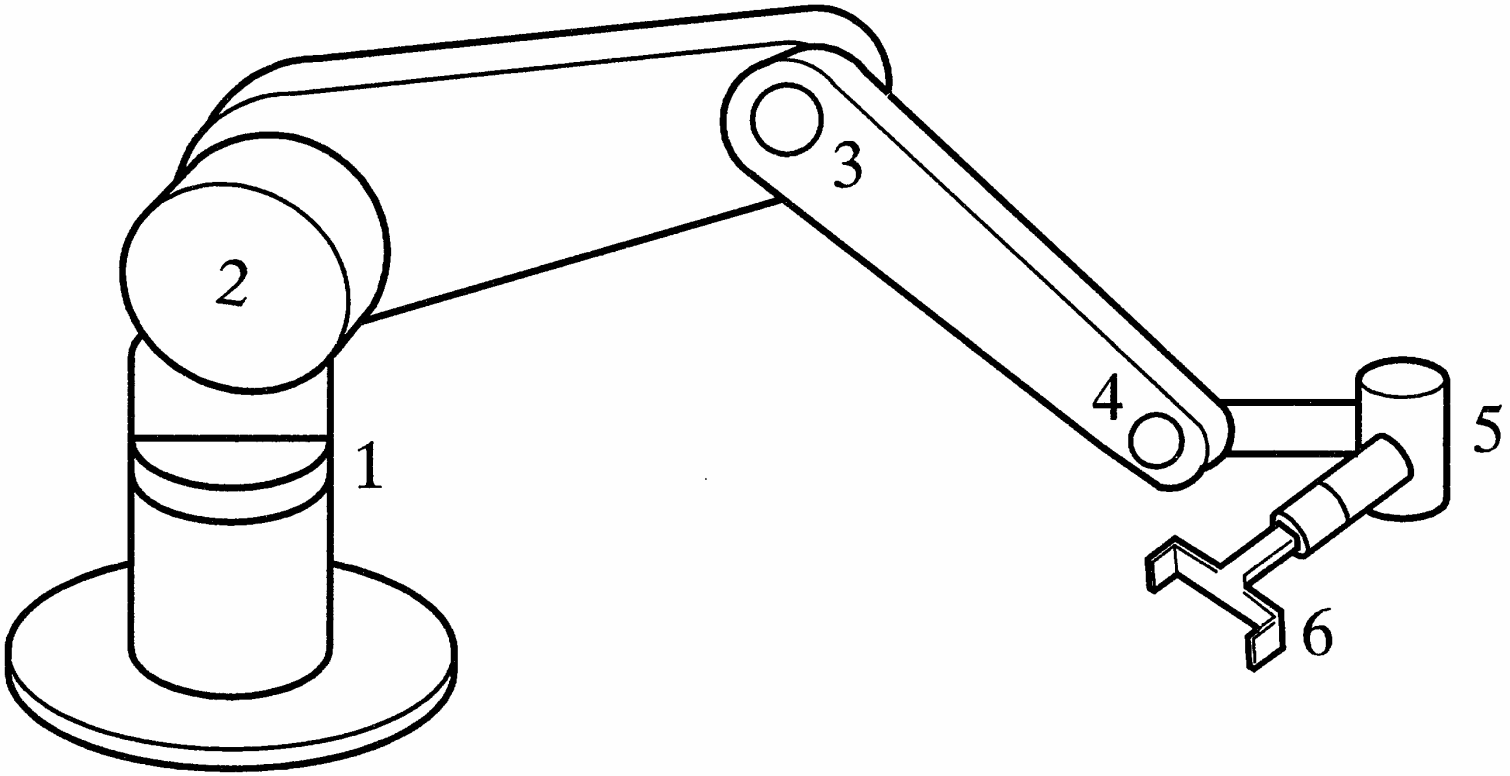
■ 정기구학(Forward Kinematics)

- 로봇의 3축의 각도가 주어졌을 때, end effect(로봇의끝점)의 위치와 방향을 기준 좌표계의 좌표로 표현하는 것
- Given: $\theta_1, \theta_2, \theta_3$ → Find: **P, n, o, a***

■ 역기구학(Inverse Kinematics)

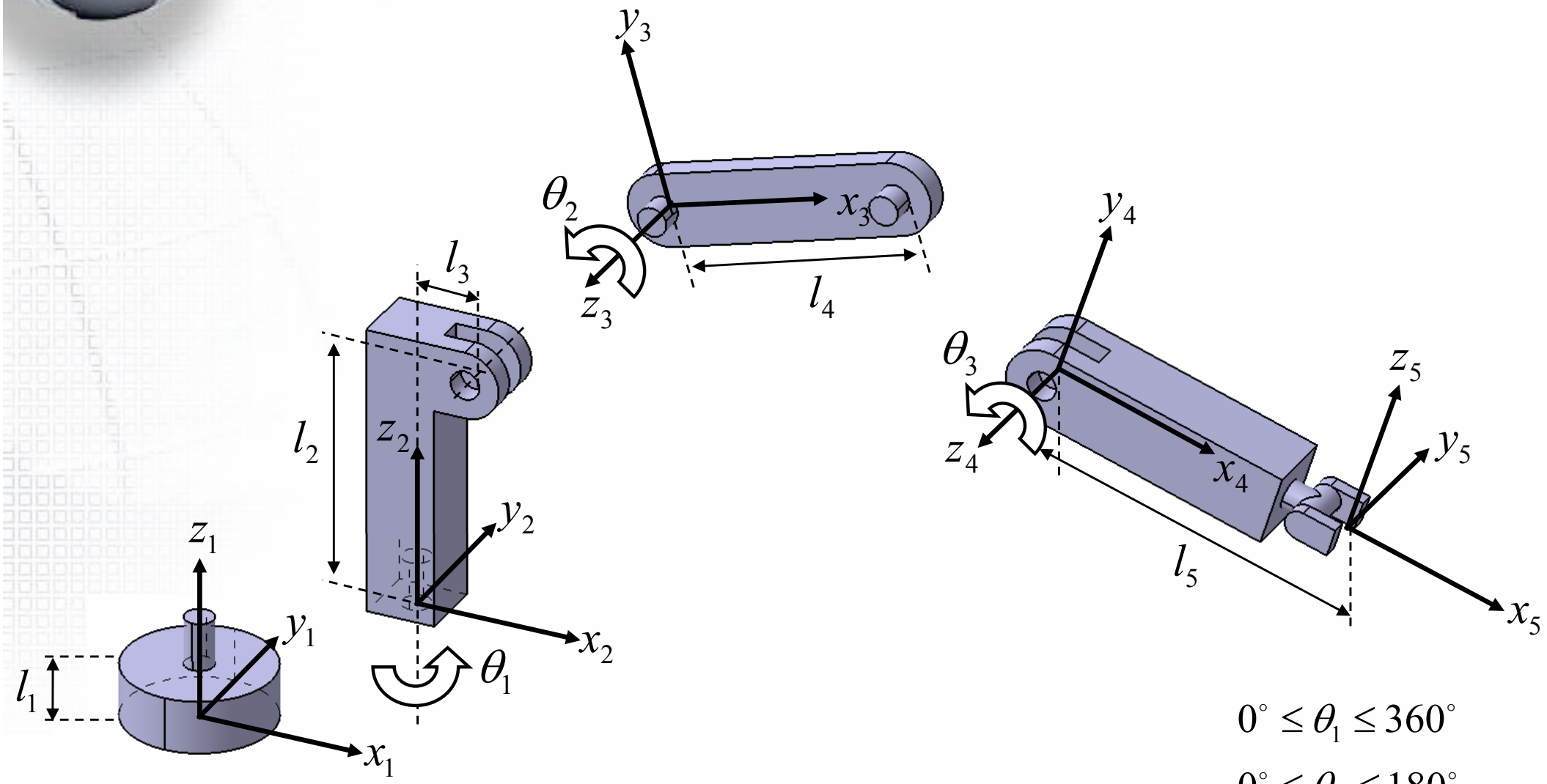
- End effect의 위치와 방향이 기준 좌표계의 좌표로 주어졌을 때, 로봇의 3축의 각도를 계산하는 것
- Given: **P, n, o, a** → Find: $\theta_1, \theta_2, \theta_3$

6축 관절 로봇의 예



@SDAL
Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

3.2.2 3축 로봇의 관절(Link)별 좌표계 설정

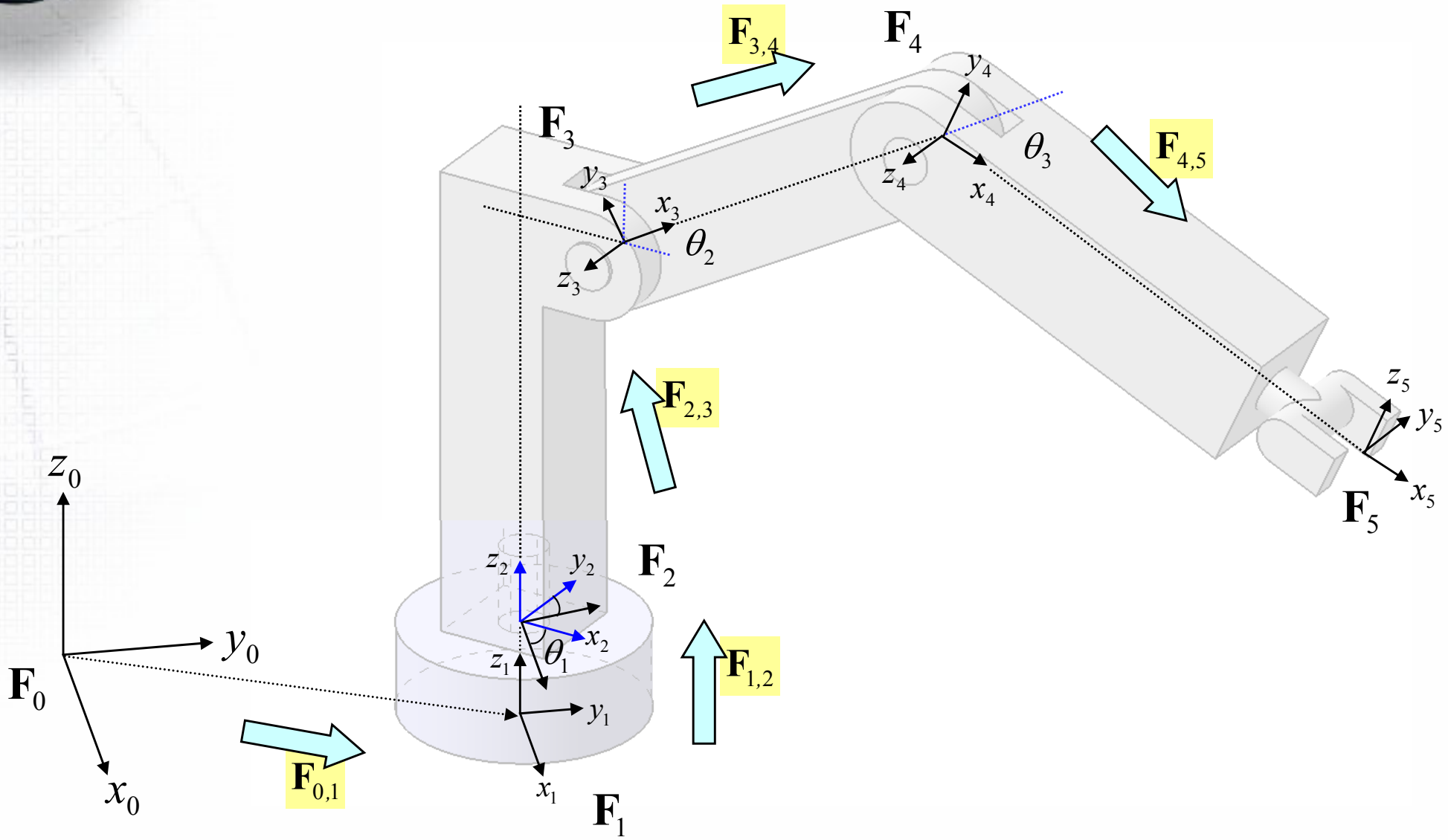


$$0^\circ \le \theta_1 \le 360^\circ$$

$$0^\circ \le \theta_2 \le 180^\circ$$

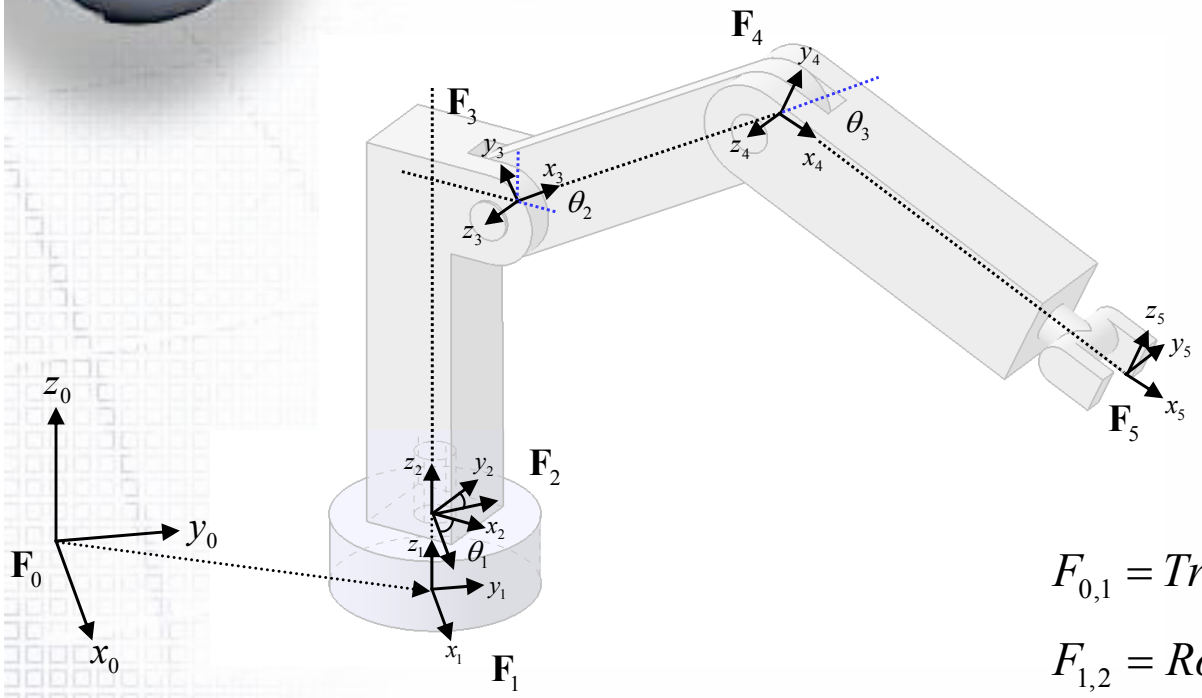
$$-180^\circ \le \theta_3 \le 180^\circ$$

3.2.3 로봇 축의 각이 θ_1, θ_2, θ_3 으로 주어졌을 때의 정기구학식 (1)



$$\therefore\ \mathbf{F}_{0,5} = \mathbf{F}_{0,1} \cdot \mathbf{F}_{1,2} \cdot \mathbf{F}_{2,3} \cdot \mathbf{F}_{3,4} \cdot \mathbf{F}_{4,5}$$

3.2.3 로봇 축의 각이 θ_1, θ_2, θ_3 으로 주어졌을 때의 정기구학식 (2)



$$F_{0,1} = Trans(\vec{d})$$

$$F_{1,2} = Rot(z,\theta_1) \cdot Trans(z,l_1)$$

$$F_{2,3} = Rot(z,\theta_2) \cdot Trans(x,l_3) \cdot Rot(x,90^\circ) \cdot Trans(z,l_2)$$

$$F_{3,4} = Trans(x,l_4) \cdot Rot(z,\theta_3)$$

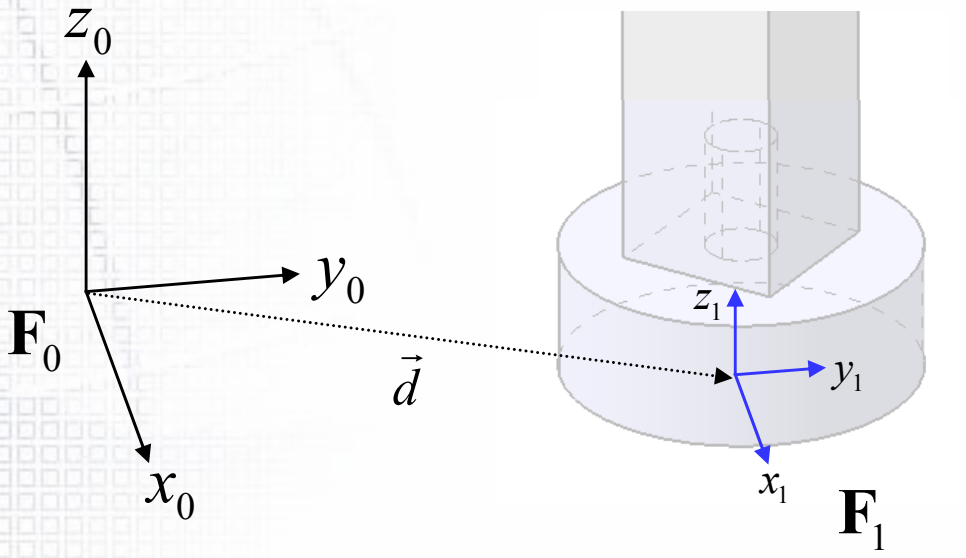
$$F_{4,5} = Rot(x,-90^\circ) \cdot Trans(x,l_5)$$

$$F_{0,5} = F_{0,1} \cdot F_{1,2} \cdot F_{2,3} \cdot F_{3,4} \cdot F_{4,5}$$

$$F_{0,5} = Trans(\vec{d}) \cdot Trans(z,l_1) \cdot Rot(z,\theta_1) \cdot Trans(z,l_2) \cdot Rot(x,90^\circ) \cdot Trans(z,l_3) \cdot Rot(z,\theta_2) \cdot$$
$$Trans(x,l_4) \cdot Rot(z,\theta_3) \cdot Trans(x,l_5) \cdot Rot(x,-90^\circ)$$

3.2.3 로봇 축의 각이 $\theta_1, \theta_2, \theta_3$ 으로 주어졌을 때의 정기구학식 (3)

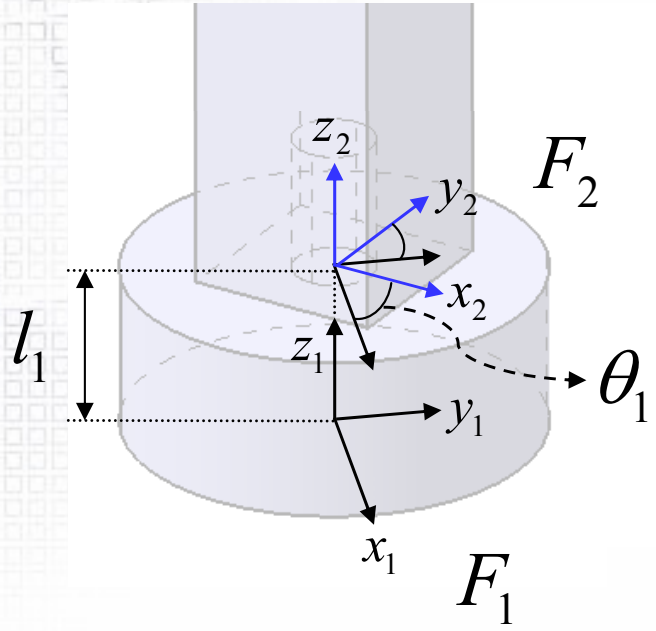
$$\boxed{F_0} \Longrightarrow \boxed{F_1}$$



$$Trans(\vec{d}) = \begin{bmatrix} 1 & 0 & 0 & d_{x0} \\ 0 & 1 & 0 & d_{y0} \\ 0 & 0 & 1 & d_{z0} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$F_{0,1} = Trans(\vec{d}) = \begin{bmatrix} 1 & 0 & 0 & d_{x0} \\ 0 & 1 & 0 & d_{y0} \\ 0 & 0 & 1 & d_{z0} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.2.3 로봇 축의 각이 $\theta_1, \theta_2, \theta_3$ 으로 주어졌을 때의 정기구학식 (4)



$$\boxed{F_1} \Longrightarrow \boxed{F_2}$$

$$Trans(z,l_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

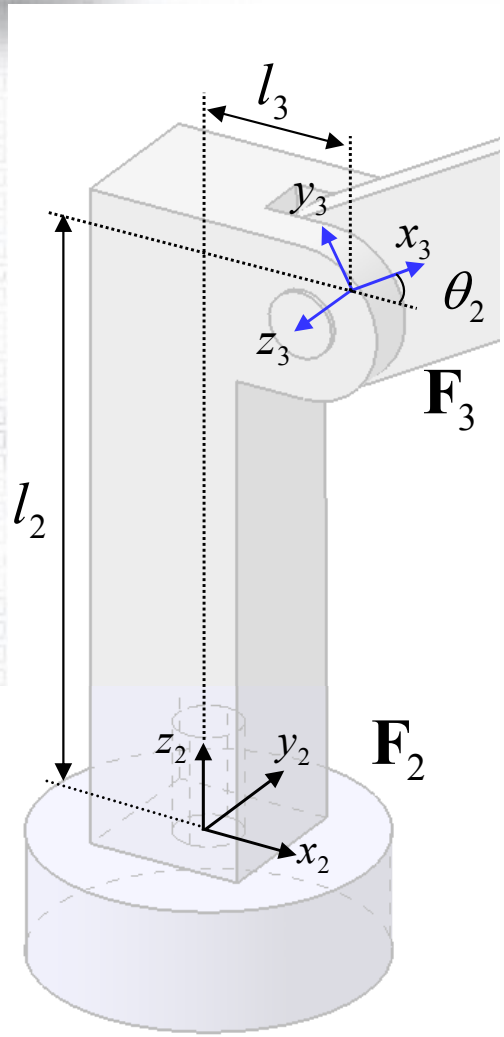
$$Rot(z,\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$F_{1,2} = Rot(z,\theta_1)\cdot Trans(z,l_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

or

$$F_{1,2} = Trans(z,l_1)\cdot Rot(z,\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.2.3 로봇 축의 각이 $\theta_1, \theta_2, \theta_3$ 으로 주어졌을 때의 정기구학식 (5)



$$\boxed{F_2} \Longrightarrow \boxed{F_3}$$

$$Trans(z,l_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(x,90^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 90^\circ & -\sin 90^\circ & 0 \\ 0 & \sin 90^\circ & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trans(x,l_3) = \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(z,\theta_2) = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

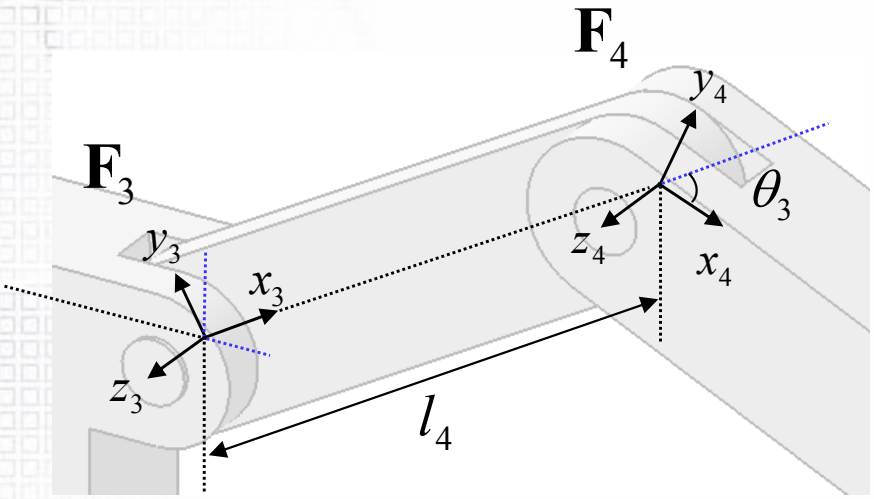
$$F_{2,3} = Trans(z,l_2)\cdot Rot(x,90^\circ)\cdot Trans(x,l_3)\cdot Rot(z,\theta_2) = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & l_3 \\ 0 & 0 & -1 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$or \quad F_{2,3} = Rot(x,90^\circ)\cdot Trans(z,l_2)\cdot Trans(x,l_3)\cdot Rot(z,\theta_2)$$

$$= Rot(x,90^\circ)\cdot Trans(z,l_2)\cdot Rot(z,\theta_2)\cdot Trans(x,l_3)$$

3.2.3 로봇 축의 각이 θ_1, θ_2, θ_3 으로 주어졌을 때의 정기구학식 (6)

$$\boxed{F_3} \Longrightarrow \boxed{F_4}$$



$$Trans(x, l_4) = \begin{bmatrix} 1 & 0 & 0 & l_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

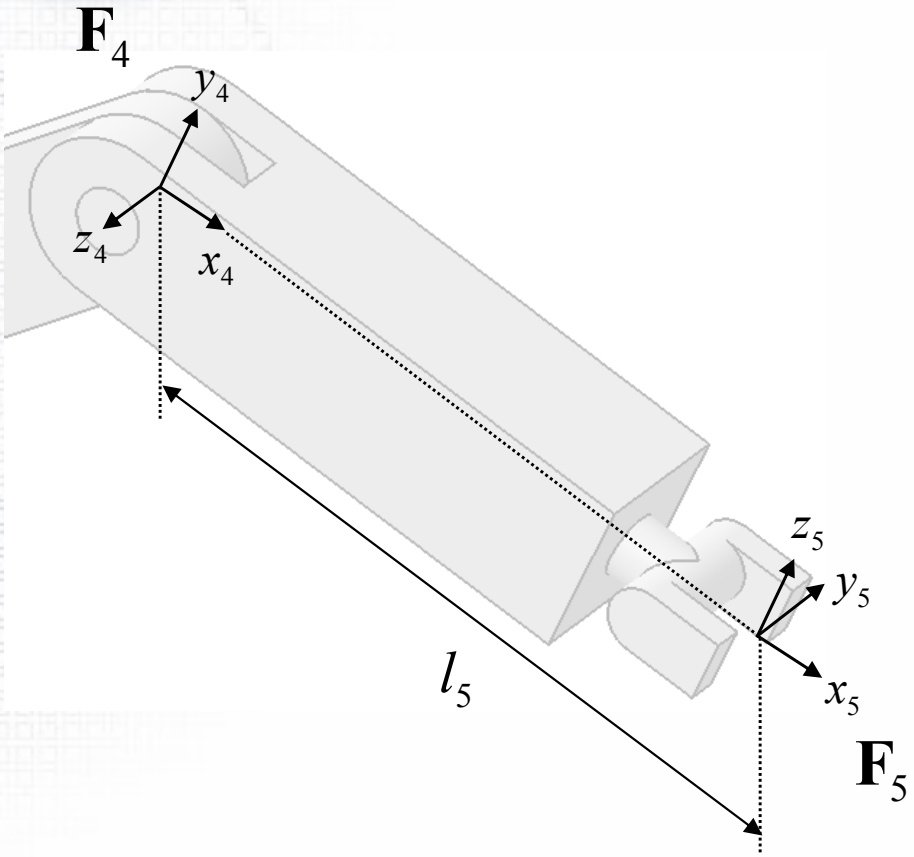
$$Rot(z, \theta_3) = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$F_{3,4} = Trans(x, l_4) \cdot Rot(z, \theta_3) = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & l_4 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

3.2.3 로봇 축의 각이 $\theta_1, \theta_2, \theta_3$ 으로 주어졌을 때의 정기구학식 (7)

$$\boxed{F_4} \Longrightarrow \boxed{F_5}$$



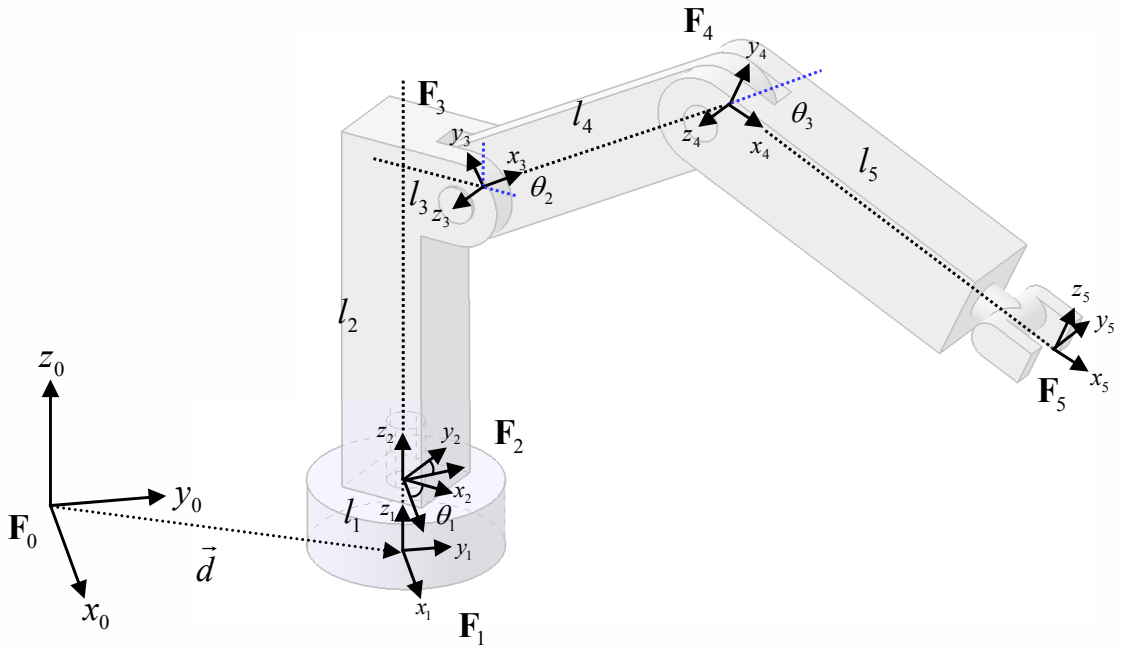
$$Trans(x, l_5) = \begin{bmatrix} 1 & 0 & 0 & l_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(x, -90^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-90^\circ) & -\sin(-90^\circ) & 0 \\ 0 & \sin(-90^\circ) & \cos(-90^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$F_{4,5} = Trans(x, l_5) \cdot Rot(x, -90^\circ) = \begin{bmatrix} 1 & 0 & 0 & l_5 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

3.2.3 로봇 축의 각이 θ_1, θ_2, θ_3 으로 주어졌을 때의 정기구학식 (8)



$$F_{0,5} = F_{0,1}F_{1,2}F_{2,3}F_{3,4}F_{4,5} =$$

$$\begin{bmatrix} C_1C_2C_3 - C_1S_2S_3 & -S_1 & -C_1C_2S_3 - C_1S_2C_3 & l_5(C_1C_2C_3 - C_1S_2S_3) + l_4C_1C_2 + l_3C_1 + d_x \\ S_1C_2C_3 - S_1S_2S_3 & C_1 & -S_1C_2S_3 - S_1S_2C_3 & l_5(S_1C_2C_3 - S_1S_2S_3) + l_4S_1C_2 + l_3S_1 + d_y \\ S_2C_3 + C_2S_3 & 0 & -S_2S_3 + C_2C_3 & l_5(S_2C_3 + C_2S_3) + l_4S_2 + l_2 + l_1 + d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(S_i = \sin\theta_i \ , \ C_i = \cos\theta_i)$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

3.2.4 정기구학식을 역으로 계산하는 역기구학식 (1)

$$F_{0,5} = F_{0,1}F_{1,2}F_{2,3}F_{3,4}F_{4,5} =$$

$$\begin{bmatrix} C_1C_2C_3 - C_1S_2S_3 & -S_1 & -C_1C_2S_3 - C_1S_2C_3 & l_5(C_1C_2C_3 - C_1S_2S_3) + l_4C_1C_2 + l_3C_1 + d_x \\ S_1C_2C_3 - S_1S_2S_3 & C_1 & -S_1C_2S_3 - S_1S_2C_3 & l_5(S_1C_2C_3 - S_1S_2S_3) + l_4S_1C_2 + l_3S_1 + d_y \\ S_2C_3 + C_2S_3 & 0 & -S_2S_3 + C_2C_3 & l_5(S_2C_3 + C_2S_3) + l_4S_2 + l_2 + l_1 + d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(S_i = \sin\theta_i \ , \ C_i = \cos\theta_i)$$

$$= \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \Leftarrow \text{주어진 값}$$

$$o_x = -\sin\theta_1 \ , \ o_y = \cos\theta_1$$

$$\sin\theta_1 = -o_x \ , \ \cos\theta_1 = o_y$$

$$\tan\theta_1 = \frac{-o_x}{o_y}$$

$$\therefore \theta_1 = \arctan\left(\frac{-o_x}{o_y}\right) \quad \text{(rad)}$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

3.2.4 정기구학식을 역으로 계산하는 역기구학식 (2)

$$F_{0,5} = F_{0,1}F_{1,2}F_{2,3}F_{3,4}F_{4,5} =$$

$$\begin{bmatrix} C_1C_2C_3 - C_1S_2S_3 & -S_1 & -C_1C_2S_3 - C_1S_2C_3 & l_5(C_1C_2C_3 - C_1S_2S_3) + l_4C_1C_2 + l_3C_1 + d_x \\ S_1C_2C_3 - S_1S_2S_3 & C_1 & -S_1C_2S_3 - S_1S_2C_3 & l_5(S_1C_2C_3 - S_1S_2S_3) + l_4S_1C_2 + l_3S_1 + d_y \\ S_2C_3 + C_2S_3 & 0 & -S_2S_3 + C_2C_3 & l_5(S_2C_3 + C_2S_3) + l_4S_2 + l_2 + l_1 + d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$(S_i = \sin\theta_i \ , \ C_i = \cos\theta_i)$

양변에 $F_{0,1}$과 $F_{1,2}$의 역행렬을 곱한다.

$$F_{1,2}^{-1} \times F_{0,1}^{-1} \times F_{0,5} =$$

$$\begin{bmatrix} -\sin\theta_2\sin\theta_3 + \cos\theta_2\cos\theta_3 & 0 & -\cos\theta_2\sin\theta_3 - \sin\theta_2\cos\theta_3 & l_5(\cos\theta_2\cos\theta_3 - \sin\theta_2\sin\theta_3) + l_4\cos\theta_2 + l_3 \\ 0 & 1 & 0 & 0 \\ \sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3 & 0 & -\sin\theta_2\sin\theta_3 + \cos\theta_2\cos\theta_3 & l_5(\sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3) + l_4\sin\theta_2 + l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} n_x\cos\theta_1 - o_x\sin\theta_1 & n_x\sin\theta_1 + o_x\cos\theta_1 & a_x & -l_1a_x - d_xn_x - d_yo_x - d_za_x + p_x \\ n_y\cos\theta_1 - o_y\sin\theta_1 & n_y\sin\theta_1 + o_y\cos\theta_1 & a_y & -l_1a_y - d_xn_y - d_yo_y - d_za_y + p_y \\ n_z\cos\theta_1 - o_z\sin\theta_1 & n_z\sin\theta_1 + o_z\cos\theta_1 & a_z & -l_1a_z - d_xn_z - d_yo_z - d_za_z + p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.2.4 정기구학식을 역으로 계산하는 역기구학식 (3)

$$F_{1,2}^{-1} \times F_{0,1}^{-1} \times F_{0,5} =$$

$$\begin{bmatrix} -\sin\theta_2\sin\theta_3+\cos\theta_2\cos\theta_3 & 0 & -\cos\theta_2\sin\theta_3-\sin\theta_2\cos\theta_3 & l_5(\cos\theta_2\cos\theta_3-\sin\theta_2\sin\theta_3)+l_4\cos\theta_2+l_3 \\ 0 & 1 & 0 & 0 \\ \sin\theta_2\cos\theta_3+\cos\theta_2\sin\theta_3 & 0 & -\sin\theta_2\sin\theta_3+\cos\theta_2\cos\theta_3 & l_5(\sin\theta_2\cos\theta_3+\cos\theta_2\sin\theta_3)+l_4\sin\theta_2+l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} n_x\cos\theta_1-o_x\sin\theta_1 & n_x\sin\theta_1+o_x\cos\theta_1 & a_x & -l_1a_x-d_xn_x-d_yo_x-d_za_x+p_x \\ n_y\cos\theta_1-o_y\sin\theta_1 & n_y\sin\theta_1+o_y\cos\theta_1 & a_y & -l_1a_y-d_xn_y-d_yo_y-d_za_y+p_y \\ n_z\cos\theta_1-o_z\sin\theta_1 & n_z\sin\theta_1+o_z\cos\theta_1 & a_z & -l_1a_z-d_xn_z-d_yo_z-d_za_z+p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$l_5(\cos\theta_2\cos\theta_3-\sin\theta_2\sin\theta_3)+l_4\cos\theta_2 = -l_1a_x-d_xn_x-d_yo_x-d_za_x+p_x-l_3$ 에서 우변을 a 로치환.

$l_5(\sin\theta_2\cos\theta_3+\cos\theta_2\sin\theta_3)+l_4\sin\theta_2 = -l_1a_z-d_xn_z-d_yo_z-d_za_z+p_z-l_2$ 에서 우변을 b 로치환 ($\because$ 이미알고있는값)

$\cos\theta_2\cos\theta_3-\sin\theta_2\sin\theta_3 = n_x\cos\theta_1-o_x\sin\theta_1$ 에서 우변을 c 로치환

$\sin\theta_2\cos\theta_3+\cos\theta_2\sin\theta_3 = n_z\cos\theta_1-o_z\sin\theta_1$ 에서 우변을 d 로치환 ($\because$ 이미알고있는값)

$$l_5(\cos\theta_2\cos\theta_3-\sin\theta_2\sin\theta_3)+l_4\cos\theta_2 = a \qquad l_5(\sin\theta_2\cos\theta_3+\cos\theta_2\sin\theta_3)+l_4\sin\theta_2 = b$$

$$\cos\theta_2\cos\theta_3-\sin\theta_2\sin\theta_3 = c \qquad \sin\theta_2\cos\theta_3+\cos\theta_2\sin\theta_3 = d$$

$$l_5c+l_4\cos\theta_2 = a \qquad l_5d+l_4\sin\theta_2 = b$$

$$l_4\cos\theta_2 = a-l_5c \qquad l_4\sin\theta_2 = b-l_5d$$

$$\frac{l_4\sin\theta_2}{l_4\cos\theta_2} = \frac{b-l_5d}{a-l_5c} \quad \Rightarrow \quad \tan\theta_2 = \frac{b-l_5d}{a-l_5c}$$

$$\theta_2 = \arctan\left(\frac{b-l_5d}{a-l_5c}\right)$$

$$\therefore \theta_2 = \arctan\left(\frac{b-l_5d}{a-l_5c}\right) \quad \text{(rad)}$$

$$\begin{pmatrix} a = -l_1a_x-d_xn_x-d_yo_x-d_za_x+p_x-l_3 \\ b = -l_1a_z-d_xn_z-d_yo_z-d_za_z+p_z-l_2 \\ c = n_x\cos\theta_1-o_x\sin\theta_1 \\ d = n_z\cos\theta_1-o_z\sin\theta_1 \end{pmatrix}$$

3.2.4 정기구학식을 역으로 계산하는 역기구학식 (4)

$$F_{1,2}^{-1} \times F_{0,1}^{-1} \times F_{0,5} =$$

$$\begin{bmatrix} -\sin\theta_2\sin\theta_3+\cos\theta_2\cos\theta_3 & 0 & -\cos\theta_2\sin\theta_3-\sin\theta_2\cos\theta_3 & l_5(\cos\theta_2\cos\theta_3-\sin\theta_2\sin\theta_3)+l_4\cos\theta_2+l_3 \\ 0 & 1 & 0 & 0 \\ \sin\theta_2\cos\theta_3+\cos\theta_2\sin\theta_3 & 0 & -\sin\theta_2\sin\theta_3+\cos\theta_2\cos\theta_3 & l_5(\sin\theta_2\cos\theta_3+\cos\theta_2\sin\theta_3)+l_4\sin\theta_2+l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} n_x\cos\theta_1-o_x\sin\theta_1 & n_x\sin\theta_1+o_x\cos\theta_1 & a_x & -l_1a_x-d_xn_x-d_yo_x-d_za_x+p_x \\ n_y\cos\theta_1-o_y\sin\theta_1 & n_y\sin\theta_1+o_y\cos\theta_1 & a_y & -l_1a_y-d_xn_y-d_yo_y-d_za_y+p_y \\ n_z\cos\theta_1-o_z\sin\theta_1 & n_z\sin\theta_1+o_z\cos\theta_1 & a_z & -l_1a_z-d_xn_z-d_yo_z-d_za_z+p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$-\sin\theta_2\sin\theta_3+\cos\theta_2\cos\theta_3 = n_x\cos\theta_1-o_x\sin\theta_1$ $\quad$ $\sin\theta_2\cos\theta_3+\cos\theta_2\sin\theta_3 = n_z\cos\theta_1-o_z\sin\theta_1$

삼각함수의 덧셈정리에 의해

$\cos\theta_2\cos\theta_3-\sin\theta_2\sin\theta_3 = \cos(\theta_2+\theta_3)$

$\sin\theta_2\cos\theta_3+\cos\theta_2\sin\theta_3 = \sin(\theta_2+\theta_3)$

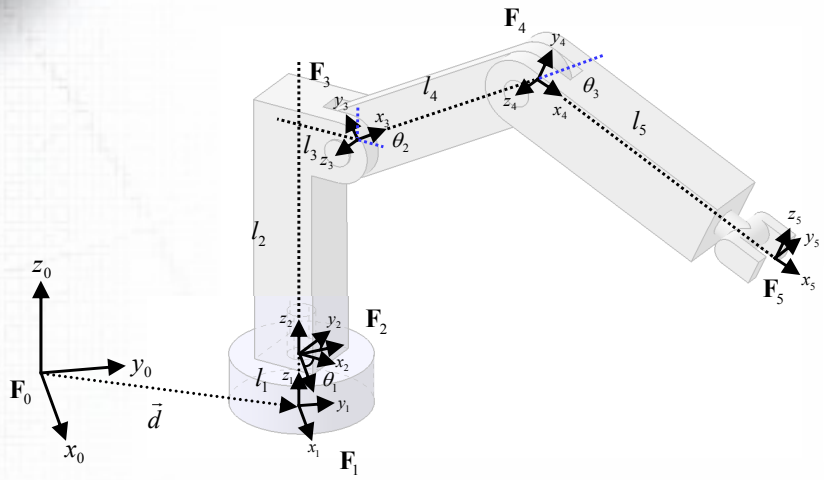
$\cos(\theta_2+\theta_3) = n_x\cos\theta_1-o_x\sin\theta_1$ $\quad$ $\sin(\theta_2+\theta_3) = n_z\cos\theta_1-o_z\sin\theta_1$

$$\tan(\theta_2+\theta_3) = \frac{n_z\cos\theta_1-o_z\sin\theta_1}{n_x\cos\theta_1-o_x\sin\theta_1}$$

$$\theta_2+\theta_3 = \arctan\left(\frac{n_z\cos\theta_1-o_z\sin\theta_1}{n_x\cos\theta_1-o_x\sin\theta_1}\right)$$

$$\therefore \theta_3 = \arctan\left(\frac{n_z\cos\theta_1-o_z\sin\theta_1}{n_x\cos\theta_1-o_x\sin\theta_1}\right)-\theta_2 \quad (\text{rad})$$

3.2.4 정기구학식을 역으로 계산하는 역기구학식 (5)



$$\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\theta_1 = \arctan\left(\frac{-o_x}{o_y}\right) \quad \text{(rad)}$$

$$\theta_2 = \arctan\left(\frac{b - l_5 d}{a - l_5 c}\right) \quad \text{(rad)}$$

$$\begin{pmatrix} a = -l_1 a_x - d_x n_x - d_y o_x - d_z a_x + p_x - l_3 \\ b = -l_1 a_z - d_x n_z - d_y o_z - d_z a_z + p_z - l_2 \\ c = n_x \cos\theta_1 - o_x \sin\theta_1 \\ d = n_z \cos\theta_1 - o_z \sin\theta_1 \end{pmatrix}$$

$$\theta_3 = \arctan\left(\frac{n_z \cos\theta_1 - o_z \sin\theta_1}{n_x \cos\theta_1 - o_x \sin\theta_1}\right) - \theta_2 \quad \text{(rad)}$$

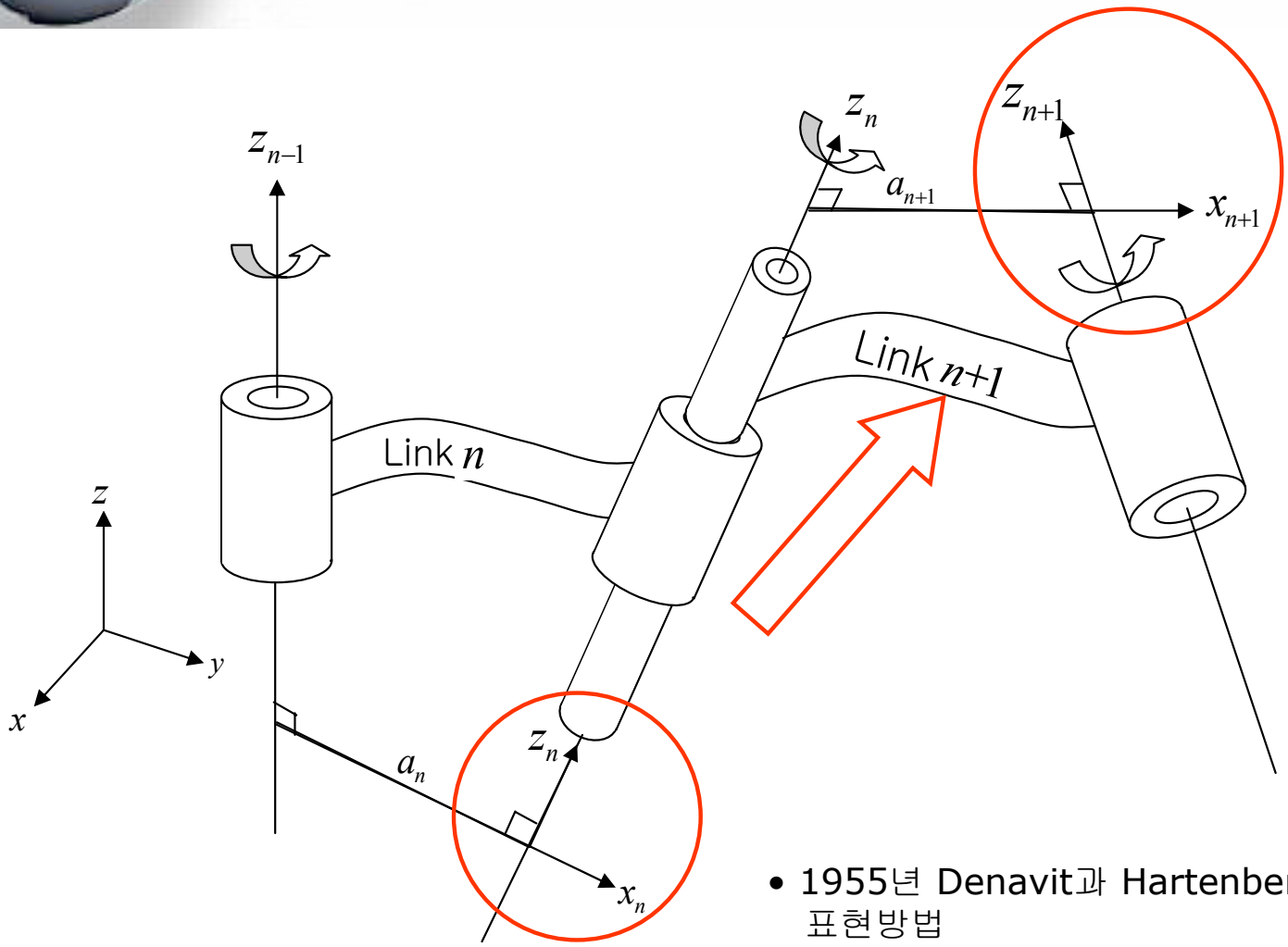
Ch 4. Denavit-Hartenberg 표기법을 이용한 로봇의 정/역기구학

4.1 기본 이론

4.2 3차원 형상변환을 이용한 로봇의 정/역기구학

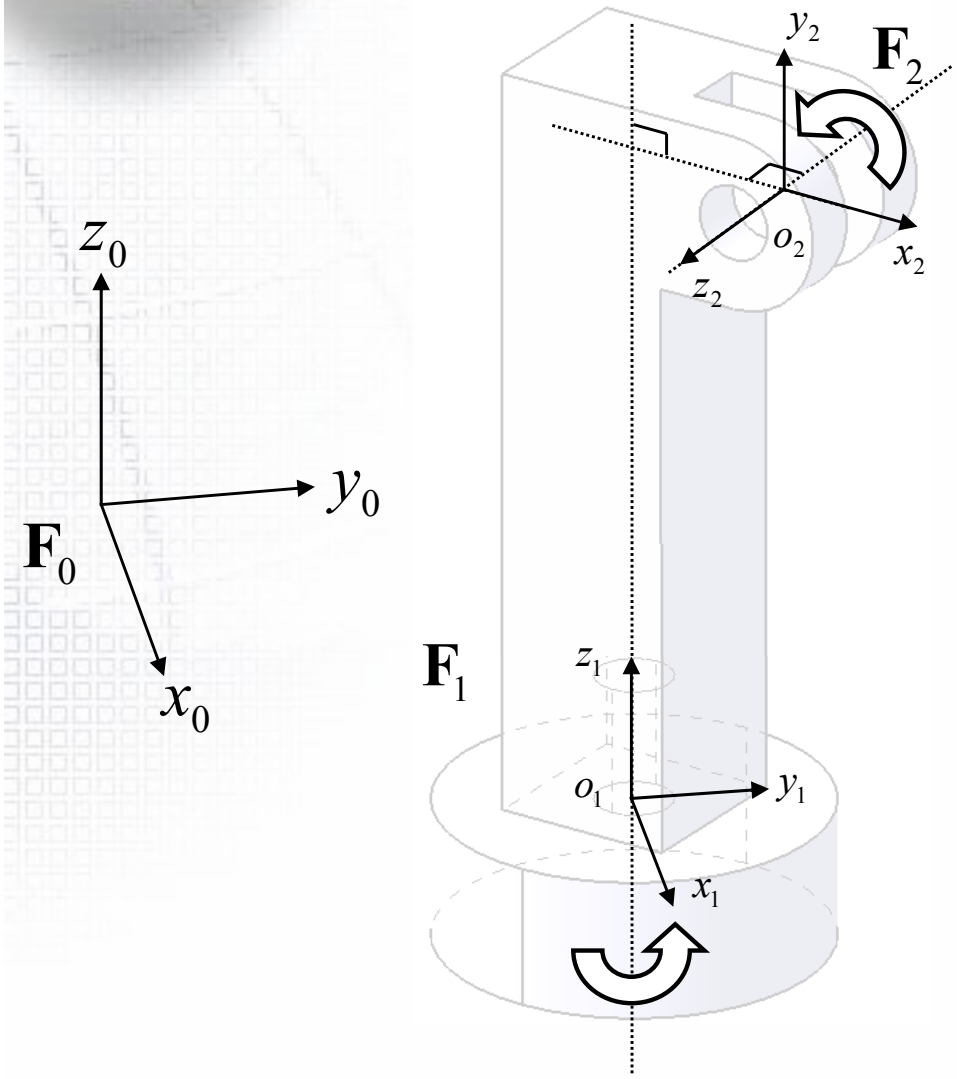
Advanced
Ship
Design
Automation
Laboratory

4.1 Denavit-Hartenberg 표기법



- 1955년 Denavit과 Hartenberg가 제안한 로봇 기구학식 표현방법
- 로봇을 관절(회전축, 병진축)과 관절을 잇는 링크로 단순화함
- 각 관절에 지역 기준좌표계를 설정한 후, 각 좌표계사이의 변환이 z축에 대한 Rotation, Translation, x축에 대한 Translation, Rotation의 4개의 변환으로 표현될 수 있음을 밝힘

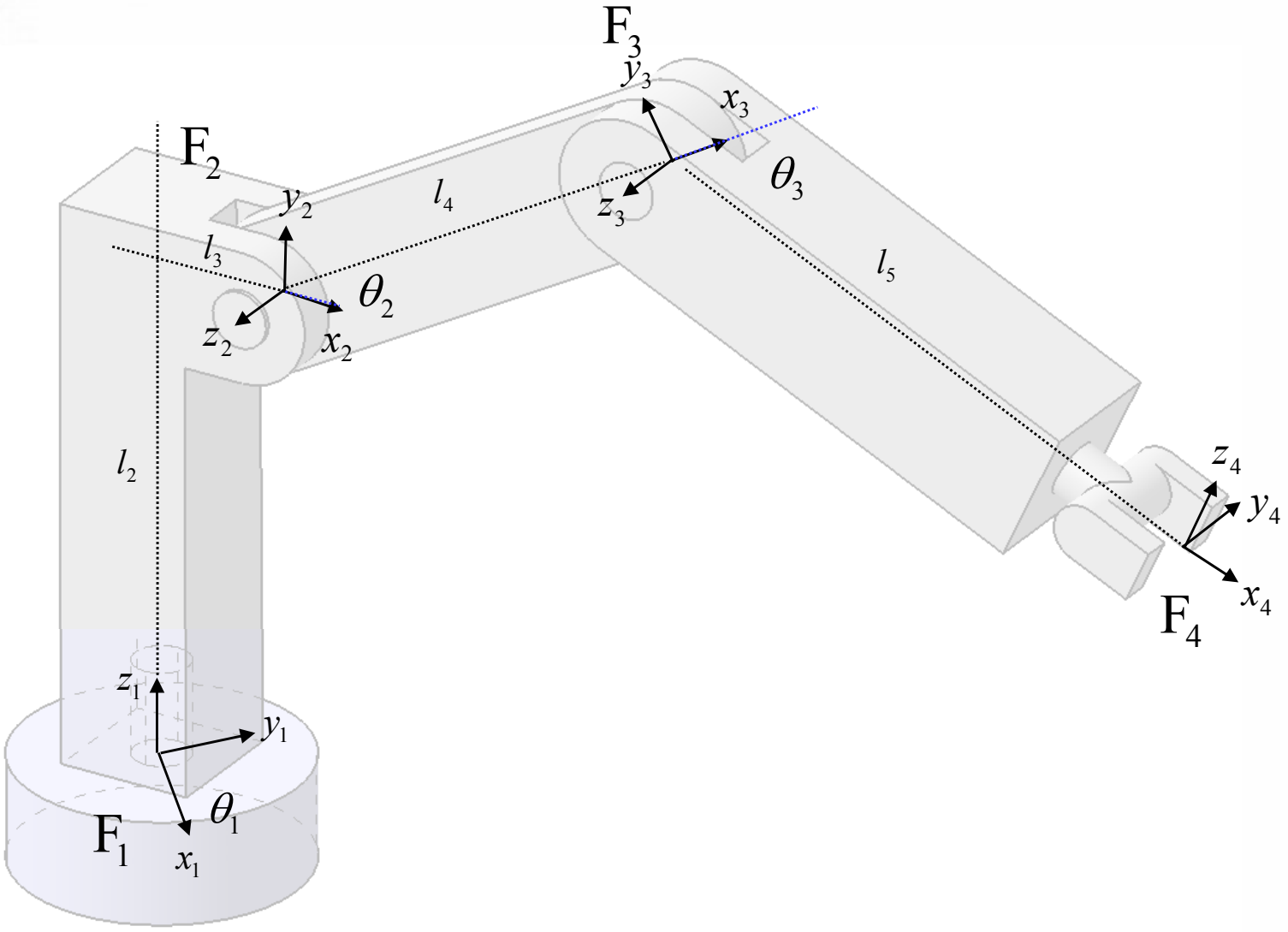
4.2 좌표계 설정 방법



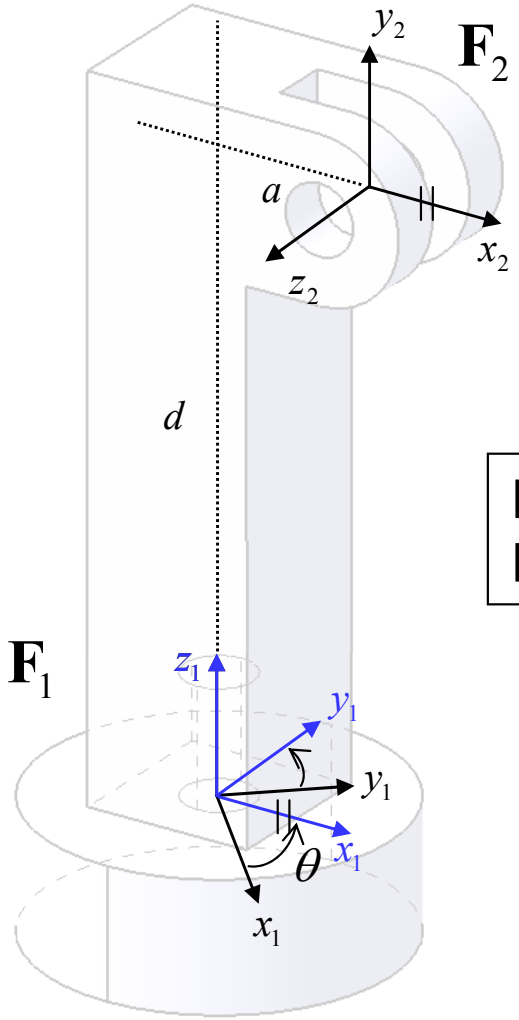
1. 회전 또는 병진관절을 z_i축으로 표시한다
2. z_i, z_{i+1}축 사이의 최단 거리를 잇는다.
3. 최단 거리의 연장선이 x_{i+1}축이 된다.*
4. 오른손 좌표계 법칙에 의해 y_i축을 표시할 수 있다.
5. 원점과 좌표계 번호를 표시한다.
6. 로봇의 기저(base)의 x_i축은 World 좌표계(F_0)의 x_0축과 평행하게 설정한다.
7. 로봇의 끝점(end effect)에서는 주로 링크의 방향을 z축으로 설정한다. 간혹 x축으로 설정하는 경우도 있다.

(* z축이 교차할 경우에는 z_i와 z_{i+1}의 외적을 x_{i+1}로 표시한다.)

4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(1)



4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(2)

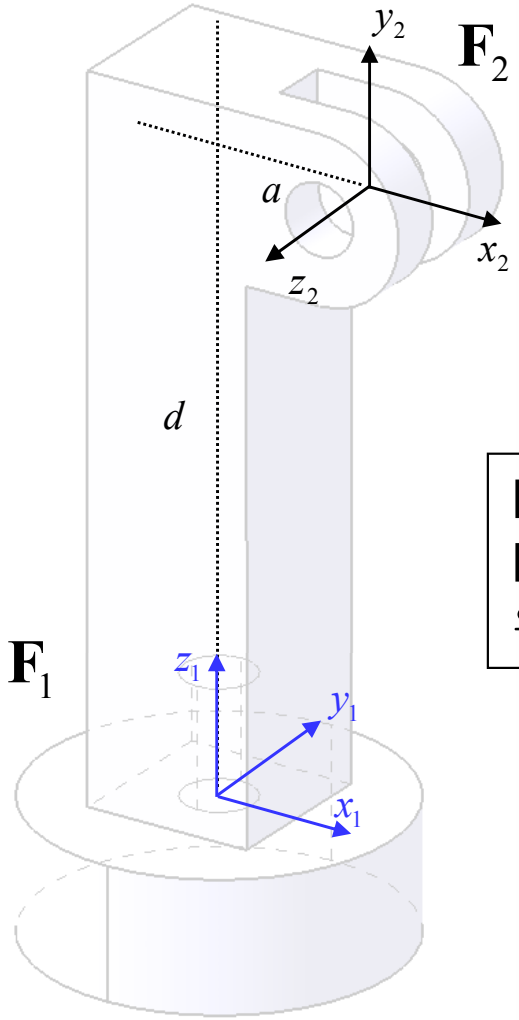


F_1을 z_1축을 중심으로 회전시켜,
F_1의 x_1과 F_2의 x_2를 평행하게 한다.

$$F_{1,2} =$$

SDAL Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(3)

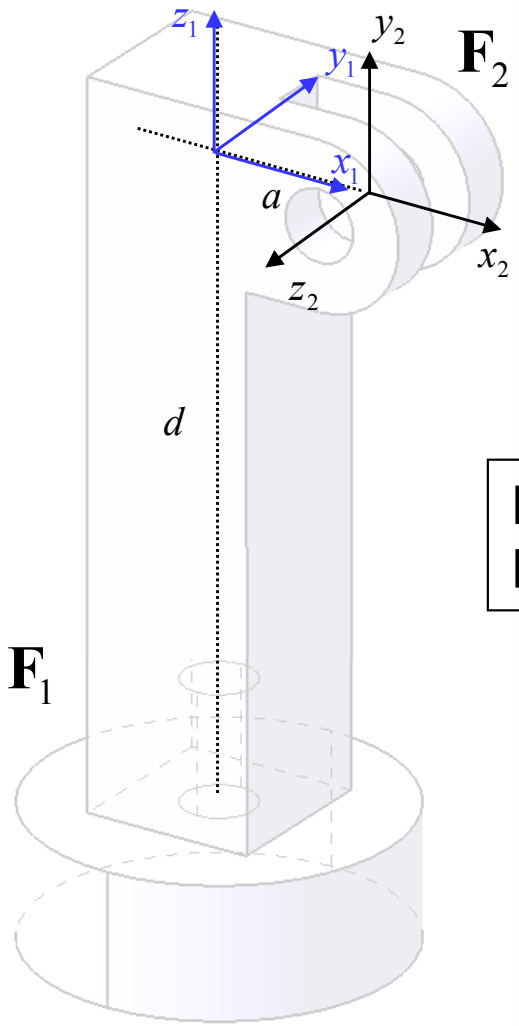


F_1을 z_1축을 기준으로 평행 이동시켜, F_1의 x_1과 F_2의 x_2를 일직선 상에 위치시킨다.

$$F_{1,2} = Rot(z, \theta_1)$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(4)

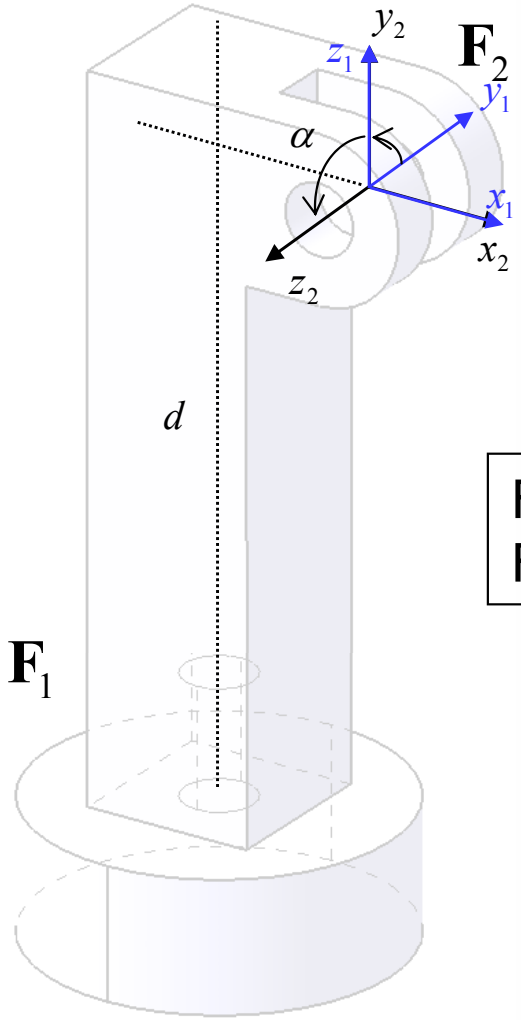


F_1을 x_1축을 기준으로 평행 이동시켜,
F_1과 F_2의 원점을 일치시킨다.

$$F_{1,2} = Rot(z,\theta_1) \times Trans(z,l_2)$$

Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(5)

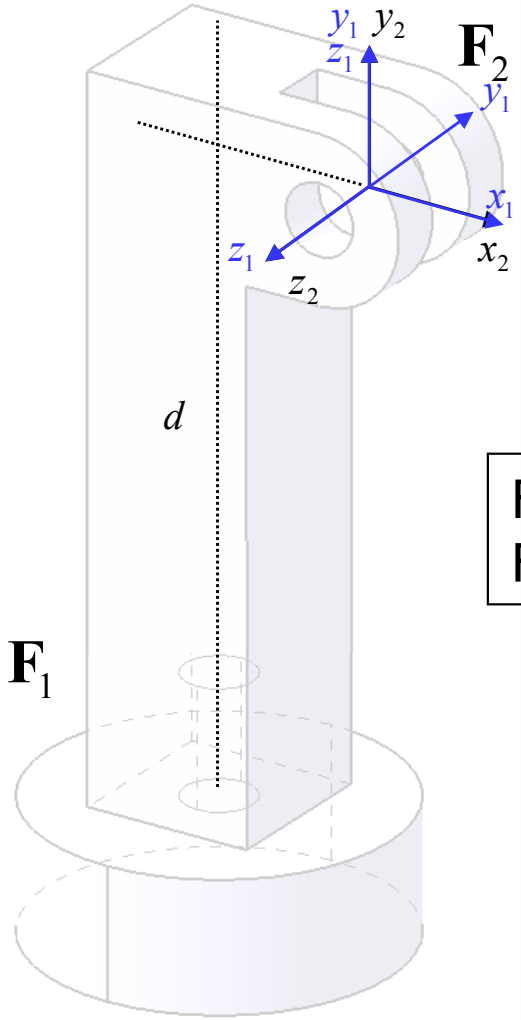


$\mathbf{F}_1$을 x_1축을 중심으로 회전시켜,
$\mathbf{F}_1$의 z_1과 $\mathbf{F}_2$의 z_2를 일치시킨다.

$$F_{1,2} = Rot(z,\theta_1) \times Trans(z,l_2) \times Trans(x,l_3)$$

Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(6)



F_1을 x_1축을 중심으로 회전시켜,
F_1의 z_1과 F_2의 z_2를 일치시킨다.

$$F_{1,2} = Rot(z,\theta_1) \times Trans(z,l_2) \times Trans(x,l_3)$$

SDAL
Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(7)

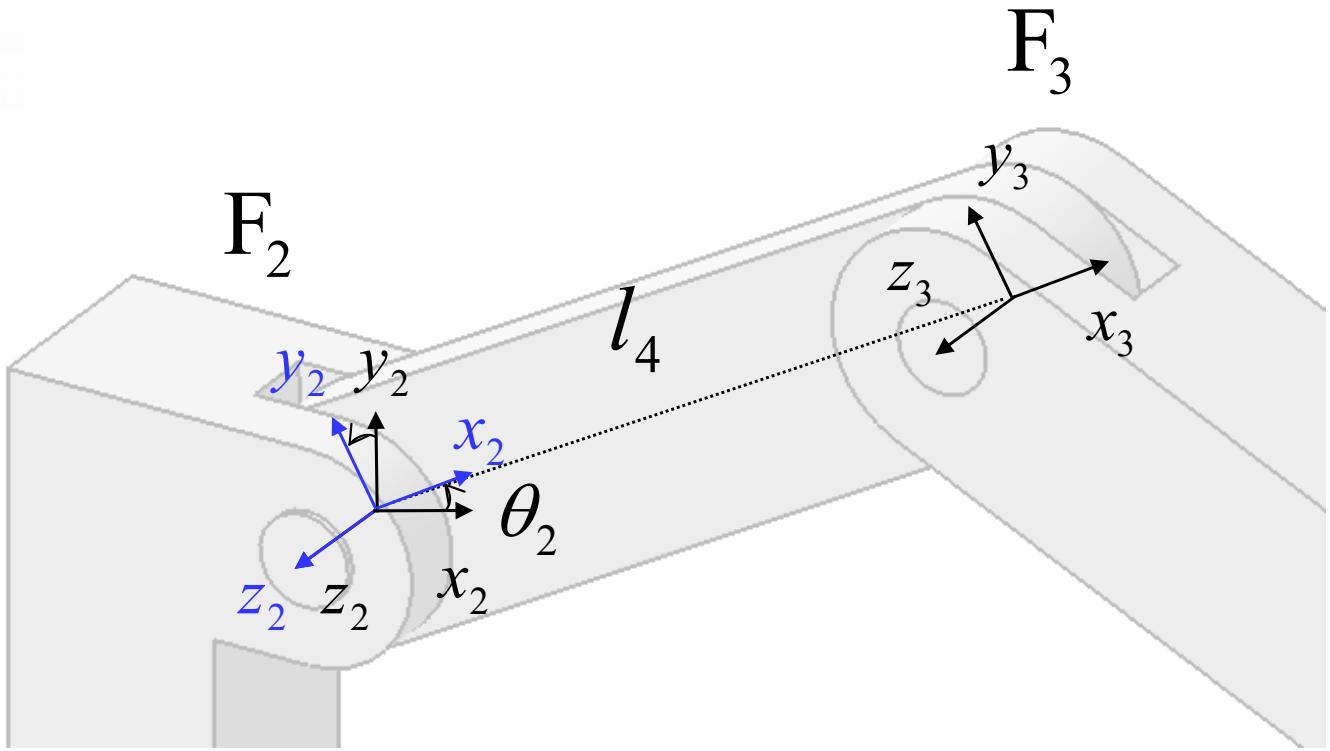
$$F_{i,i+1} = Rot(z,\theta) \times Trans(z,d) \times Trans(x,a) \times Rot(x,\alpha)$$

$$= \begin{bmatrix} C\alpha & -S\alpha & 0 & 0 \\ S\alpha & C\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\theta & -S\theta C\alpha & S\theta S\alpha & aC\theta \\ S\theta & C\theta C\alpha & -C\theta S\alpha & aS\theta \\ 0 & S\alpha & C\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

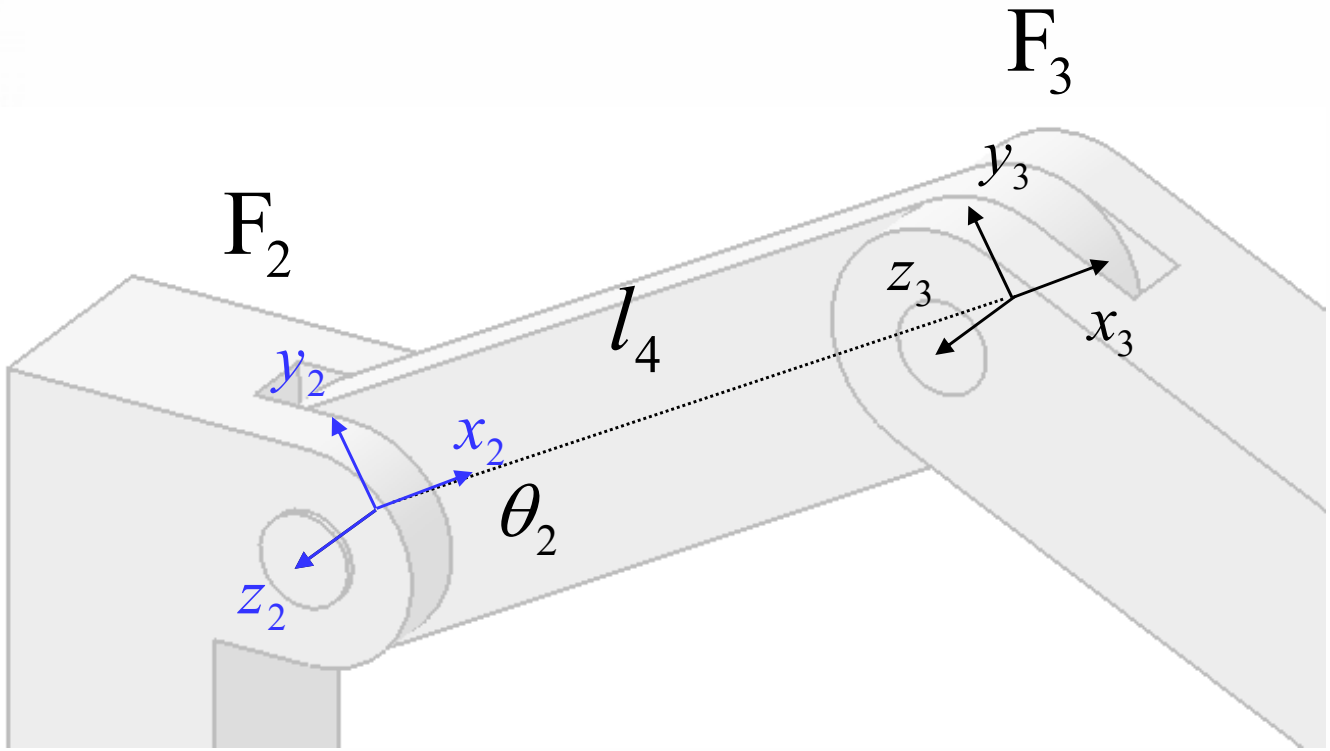
4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(8)



$$F_{2,3} =$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

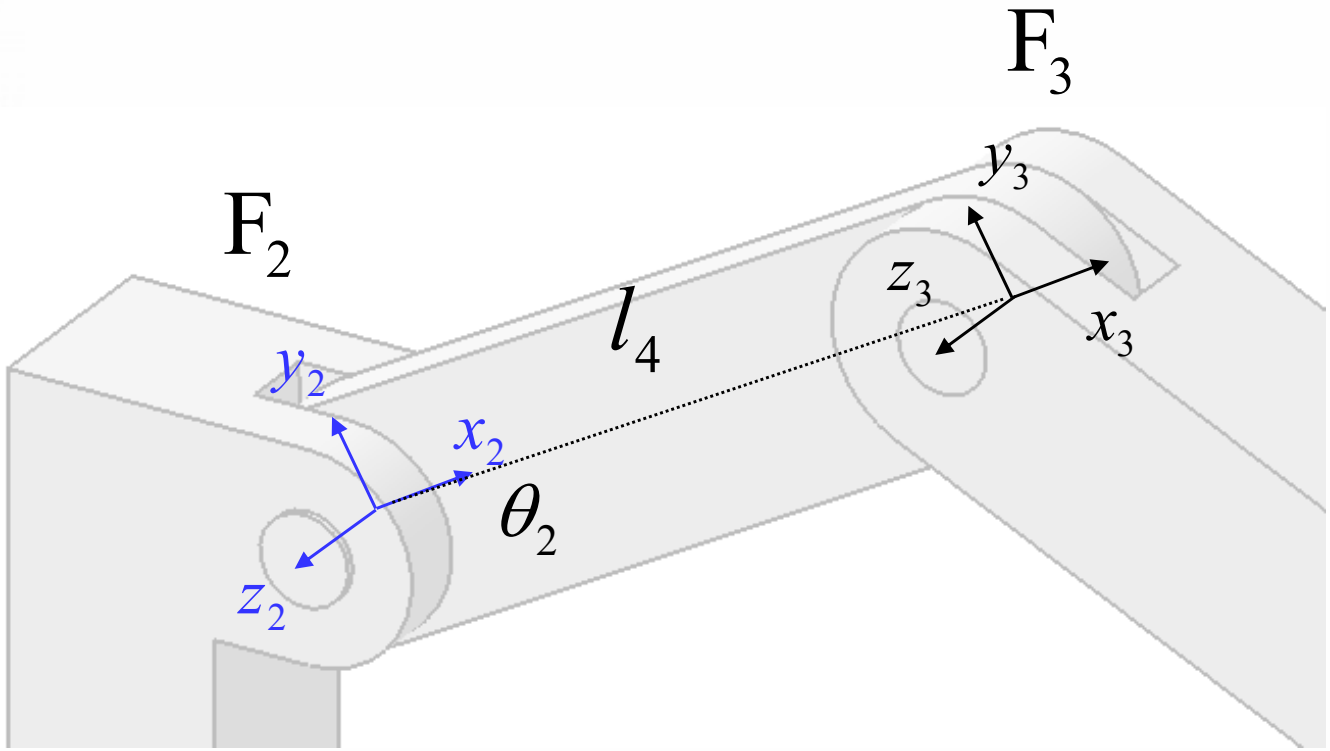
4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(9)



$$F_{2,3} = Rot(z, \theta_2)$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

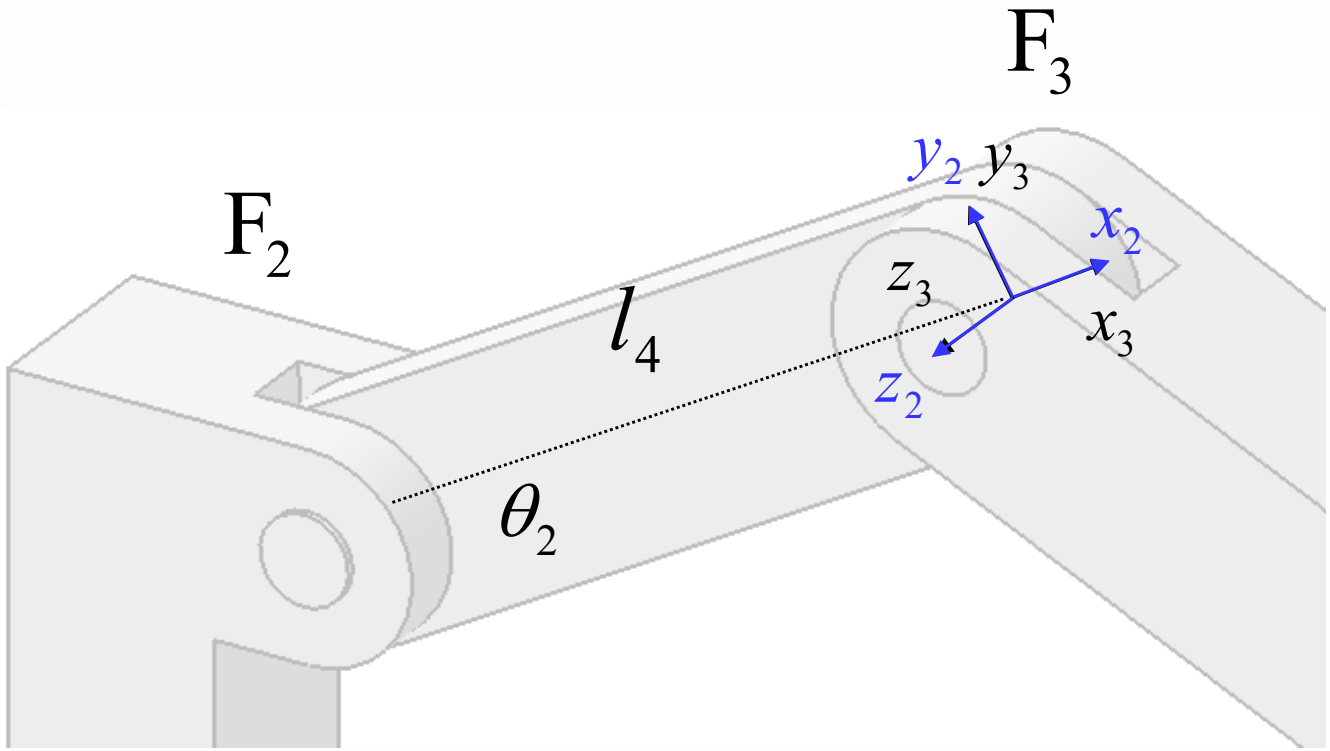
4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(10)



$$F_{2,3} = Rot(z,\theta_2) \times Trans(z,0)$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

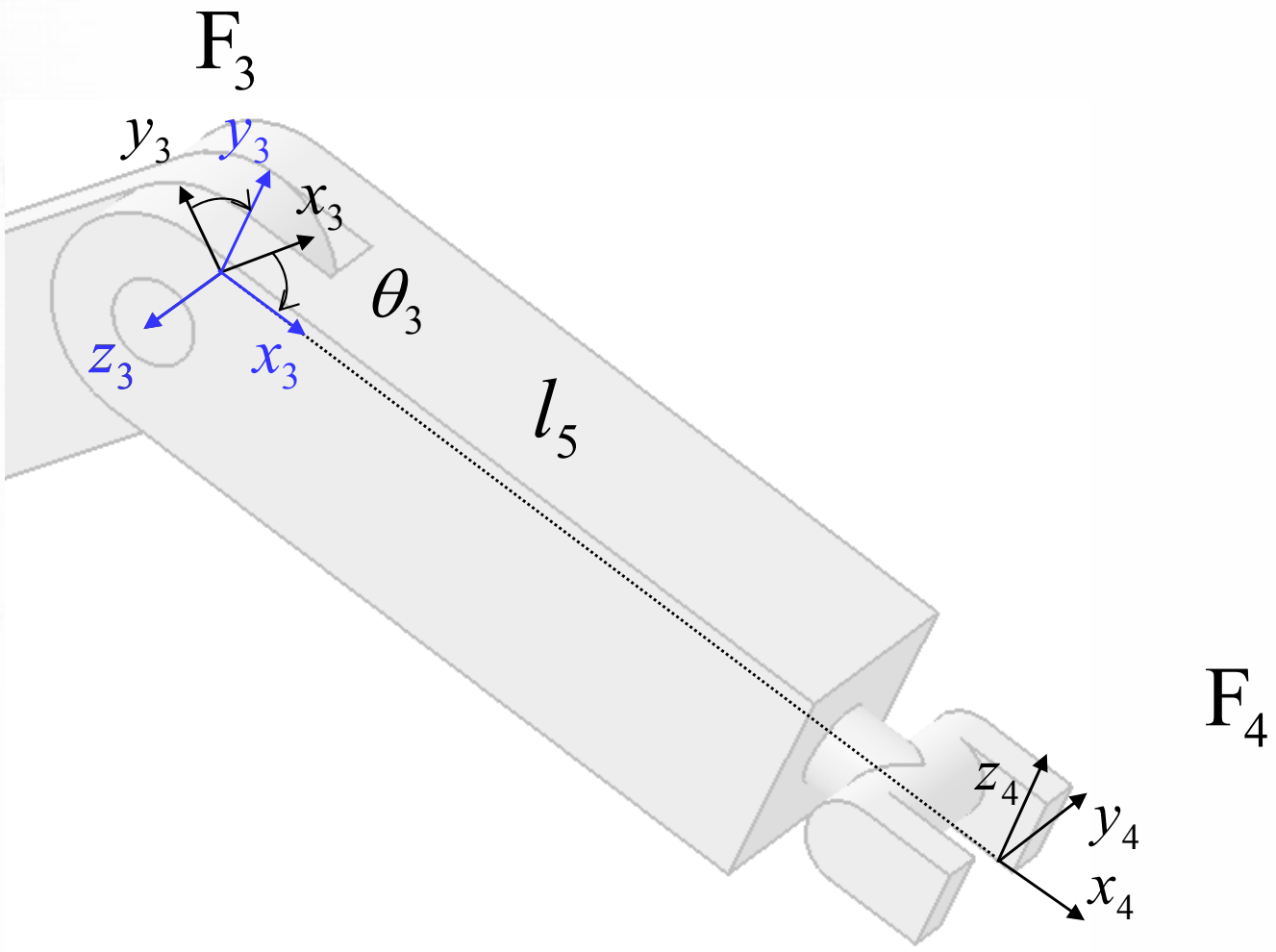
4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(11)



$$F_{2,3} = Rot(z,\theta_2) \times Trans(z,0) \times Trans(x,l_4)$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

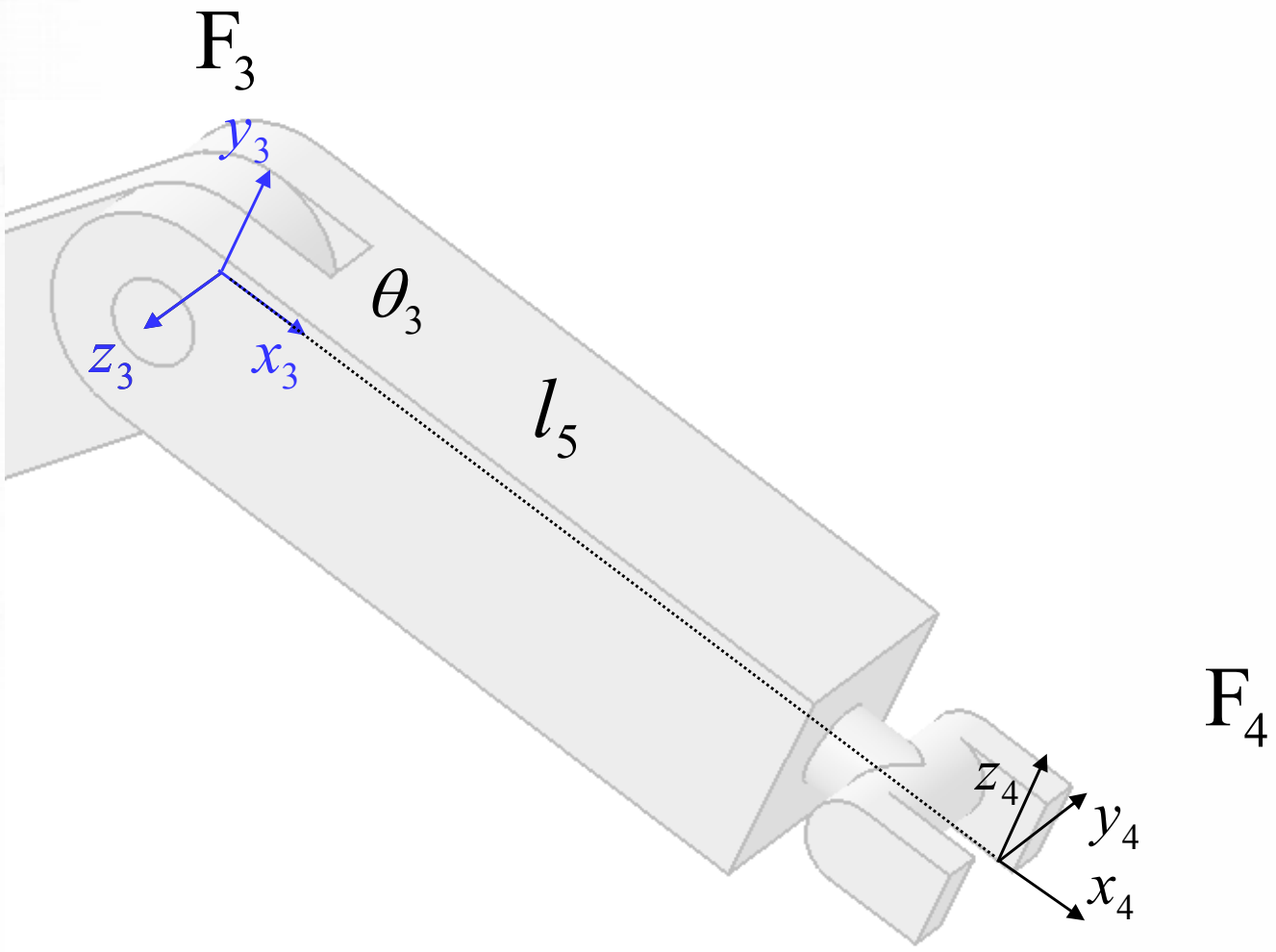
4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(12)



$$F_{3,4} =$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

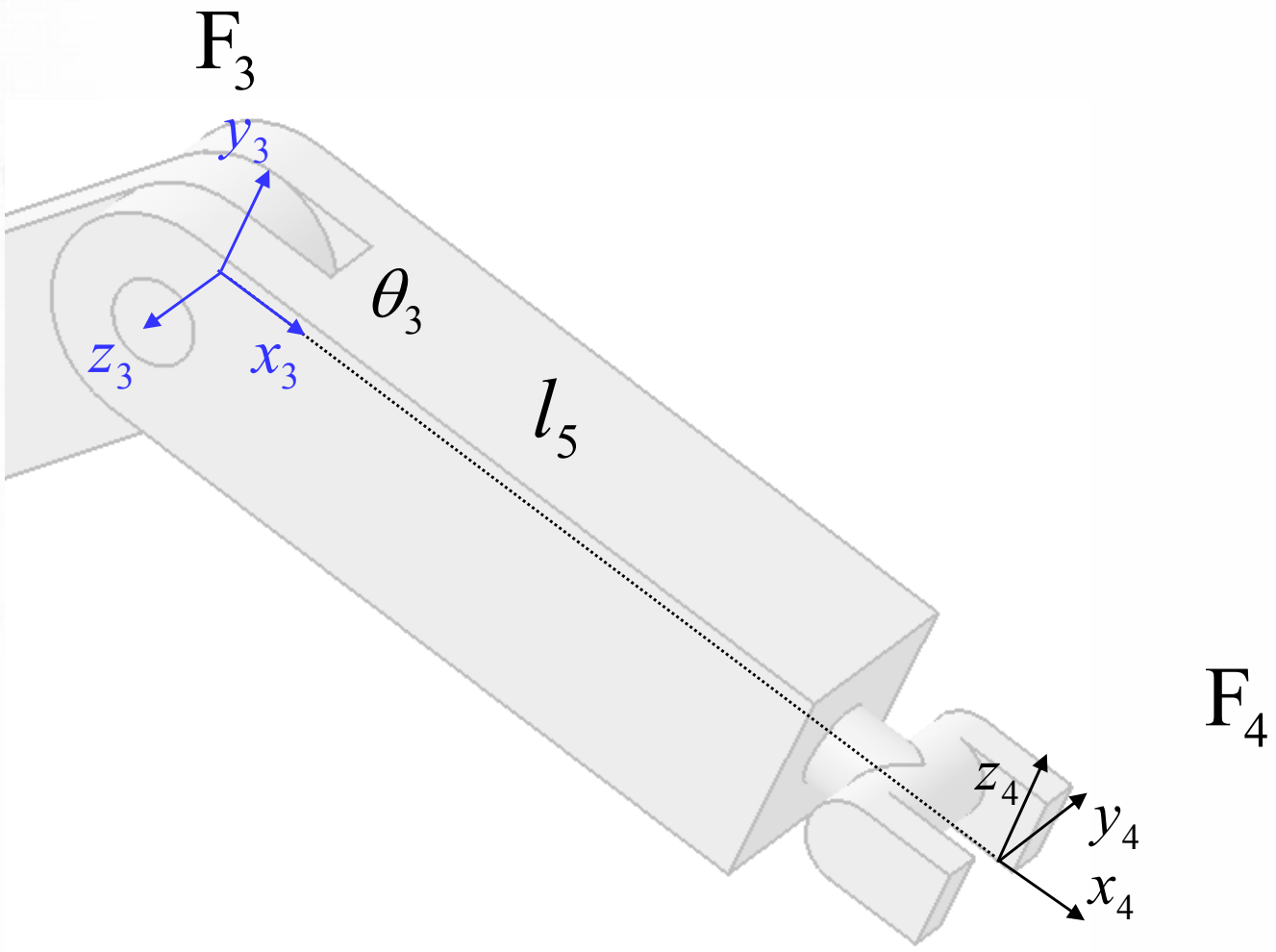
4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(13)



$$F_{3,4} = Rot(z, \theta_3)$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

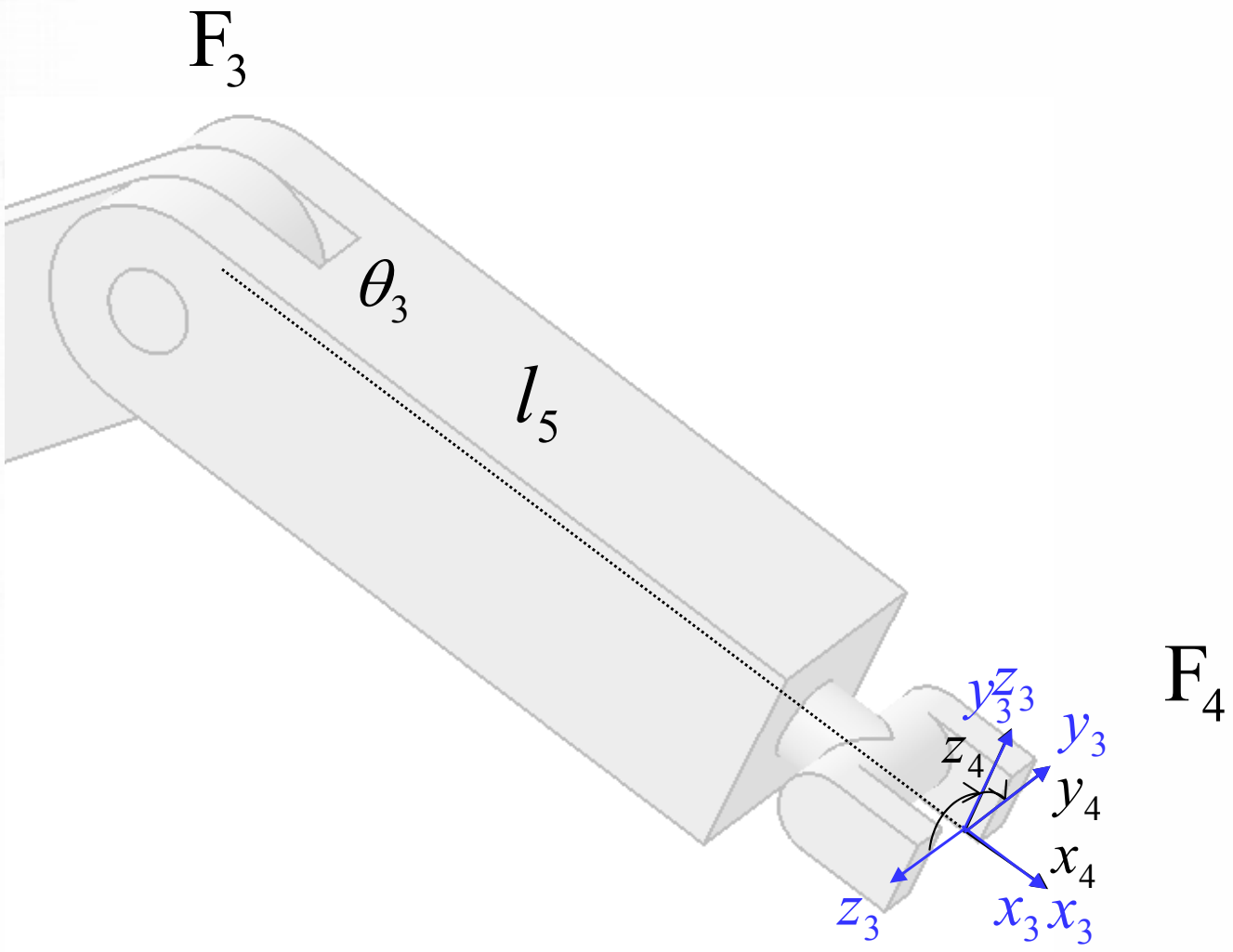
4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(14)



$$F_{3,4} = Rot(z,\theta_3) \times Trans(z,0)$$

SDAL
Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

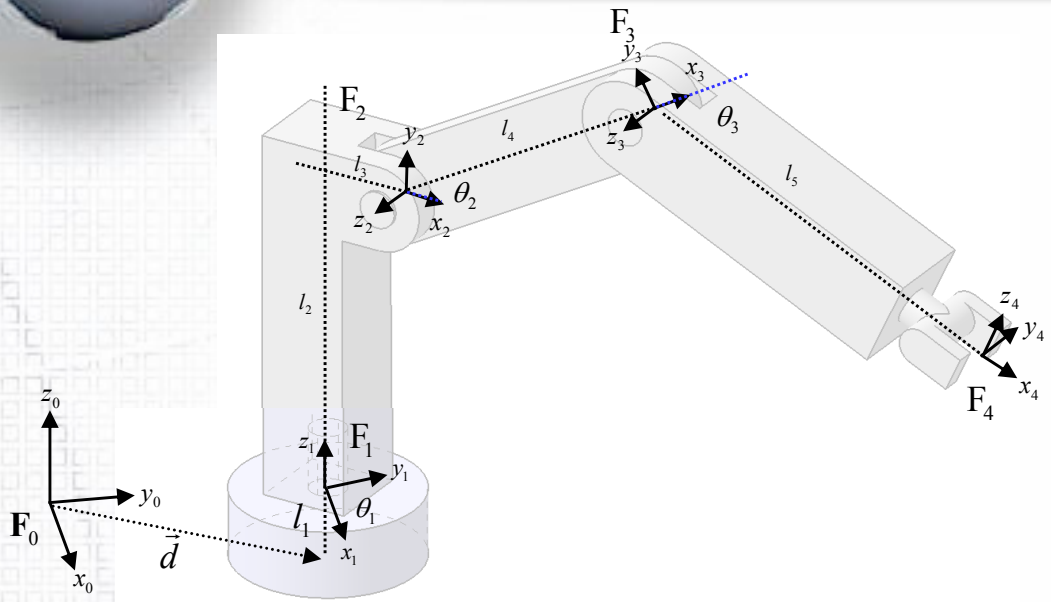
4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(15)



$$F_{3,4} = Rot(z,\theta_3) \times Trans(z,0) \times Trans(x,l_5)$$

SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

4.3 D-H 표현법을 이용한 끝단(End Effect)의 위치와 방향을 계산하는 정기구학 계산식(16)



	θ	d	a	α
1	θ_1	l_2	l_3	90°
2	θ_2	0	l_4	0°
3	θ_3	0	l_5	-90°

$$F_{0,1} = Trans(\vec{d}) \times Trans(z, l_1)$$

$$F_{1,2} = Rot(z, \theta_1) \times Trans(z, l_2) \times Trans(x, l_3) \times Rot(x, 90^\circ)$$

$$F_{2,3} = Rot(z, \theta_2) \times Trans(z, 0) \times Trans(x, l_4) \times Rot(x, 0)$$

$$F_{3,4} = Rot(z, \theta_3) \times Trans(z, 0) \times Trans(x, l_5) \times Rot(x, -90^\circ)$$

$$\therefore \; F_{0,4} = F_{0,1} \times F_{1,2} \times F_{2,3} \times F_{3,4}$$

3차원 형상변환을 이용한 정기구학 식과 D-H 표현법을 이용한 정기구학 식은 동일하다. 따라서, 역기구학식을 계산하는 방법도 동일하다

Ch 5. 로봇의 3차원 가시화

5.1 3차원 가시화 개요

5.2 3차원 뷰잉 파이프라인

5.3 3차원 가시화 프로그램 작성

* Ch 5.에서는 Ch 1~4와는 달리 행벡터를 이용하여 설명함

Advanced
Ship
Design
Automation
Laboratory

5.1 3차원 가시화 개요

5.1.1 3차원 가시화의 개념

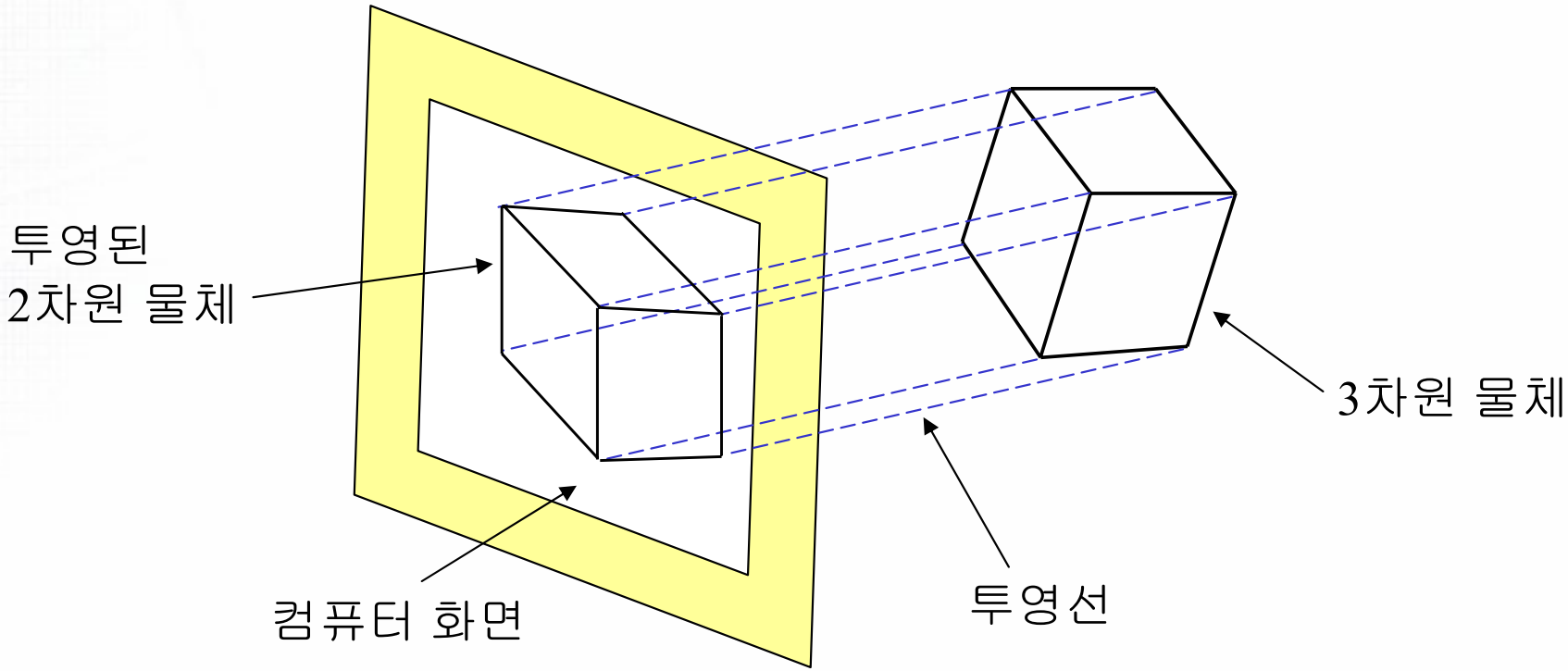
5.1.2 투영 방법

Advanced
Ship
Design
Automation
Laboratory

5.1.1 3차원 가시화의 개념

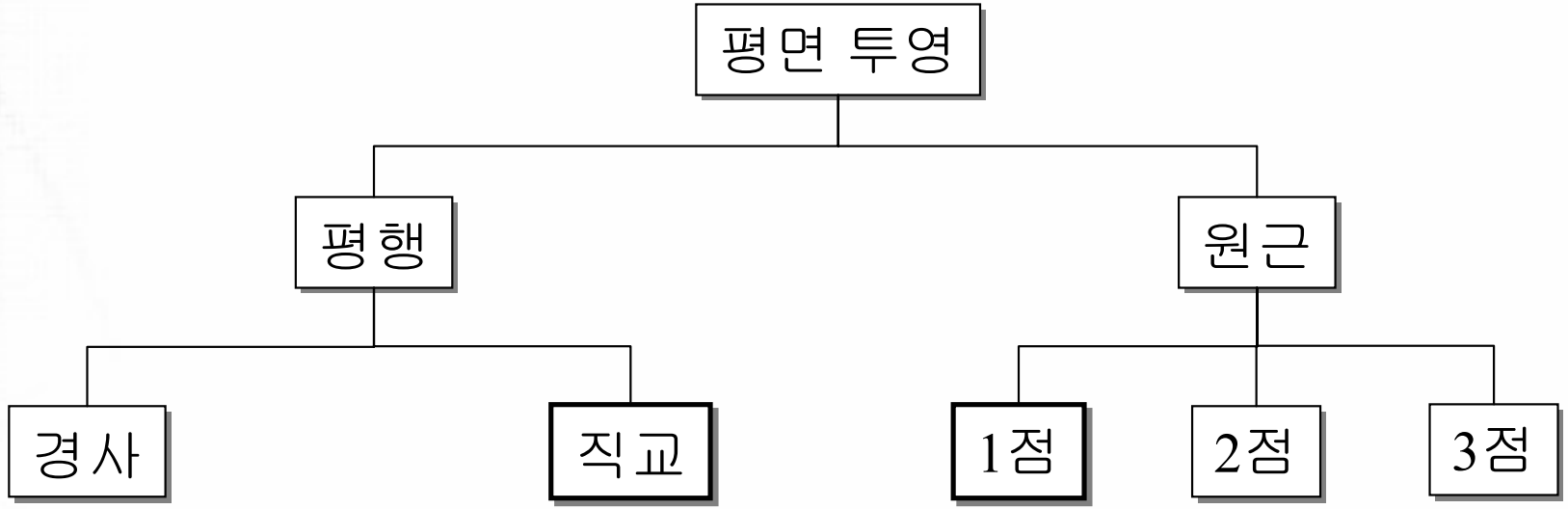
■ 3-Dimensional Computer Graphics [Visualization]

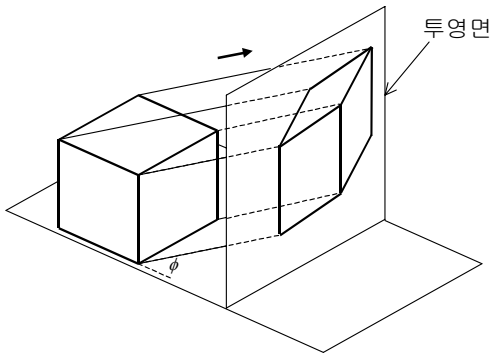
➢ 실세계의 3차원 물체를 컴퓨터 화면으로 투영(projection)하여 2차원으로 가시화하는 것

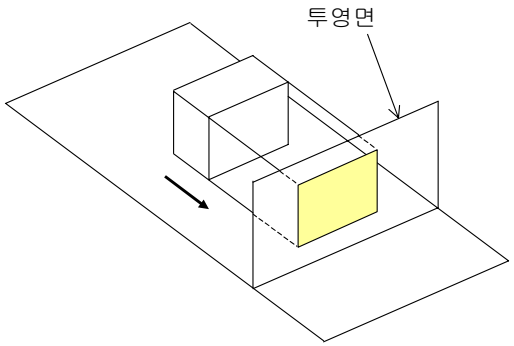


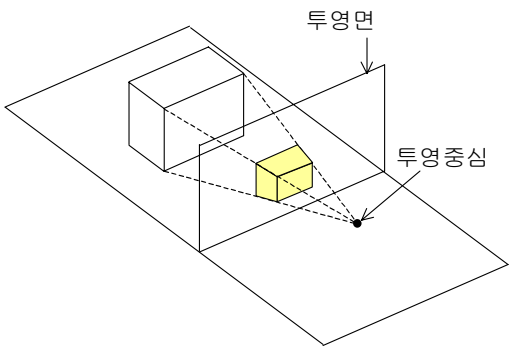
5.1.2 투영 방법

■ 투영의 종류



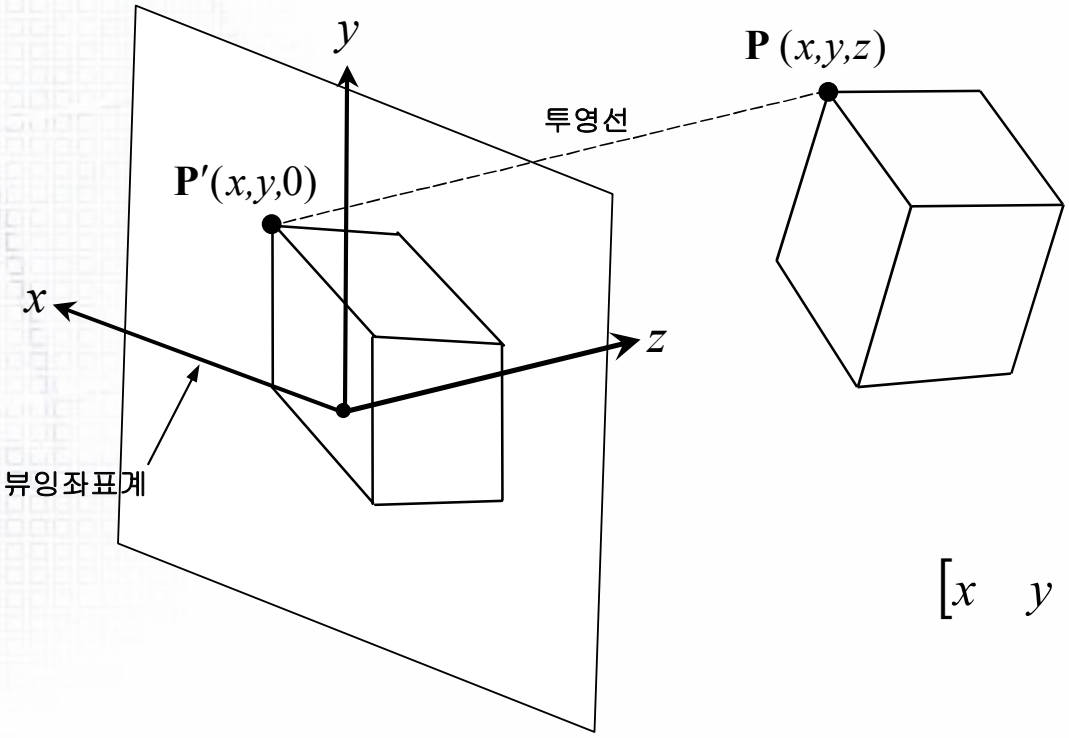






5.1.2.1 직교 투영(orthogonal projection)

■ 투영면과 xy 평면이 일치하도록 좌표계 xyz 를 잡으면, 임의의 점 $\mathbf{P}(x, y, z)$를 투영면에 직교투영한 점 $\mathbf{P}'$ 는 다음과 같이 나타낼 수 있다



주어진 점을 xy 평면에 직교투영하는 변환행렬

$$[x \quad y \quad 0 \quad 1] = [x \quad y \quad z \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

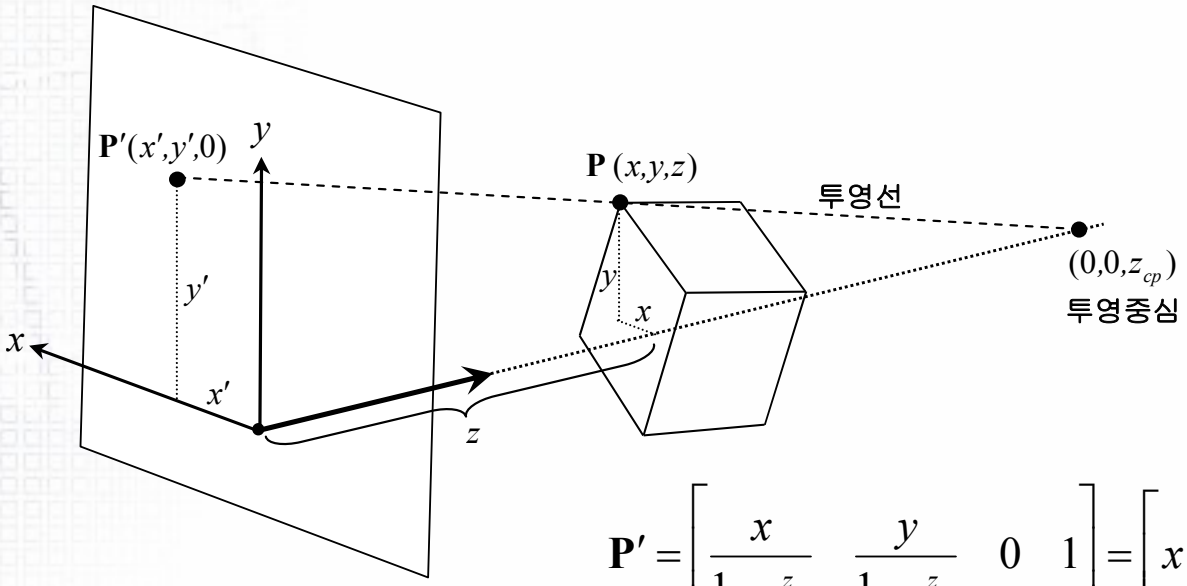
■ 특징

- 모든 투영선은 z축에 평행
- 계산이 간단하지만 원근감이 없어서 시각적 현실감이 부족함

5.1.2.2 원근 투영(perspective projection)

■ 원근투영: 임의의 점 **P**를 투영중심을 지나는 선을 따라서 투영하는 방법

■ 투영면과 xy 평면이 일치하도록 좌표계 xyz 를 잡고 투영중심이 $(0,0,z_{cp})$ 라면, 임의의 점 $P(x,y,z)$를 투영면에 1점 원근투영한 점 P' 는 다음과 같다



$$\frac{x'}{x} = \frac{z_{cp}}{z_{cp}-z} \Rightarrow x' = \frac{x}{1-\frac{z}{z_{cp}}}, \qquad \frac{y'}{y} = \frac{z_{cp}}{z_{cp}-z} \Rightarrow y' = \frac{y}{1-\frac{z}{z_{cp}}}$$

$$\mathbf{P}' = \begin{bmatrix} \frac{x}{1-\frac{z}{z_{cp}}} & \frac{y}{1-\frac{z}{z_{cp}}} & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 0 & 1-\frac{z}{z_{cp}} \end{bmatrix}$$

$$= \begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{z_{cp}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

주어진 점을 $(0,0,z_{cp})$ 점을 중심으로 xy평면에 원근투영하는 변환행렬

■ 특징

➢ 시각적 현실감이 뛰어나지만 물체의 정확한 치수를 유지하지 못하는 단점이 있다

5.2 3차원 뷰잉 연산

5.2.1 3차원 뷰잉 파이프라인

5.2.2 뷰잉 좌표계의 설정 방법

5.2.3 뷰잉 변환 행렬

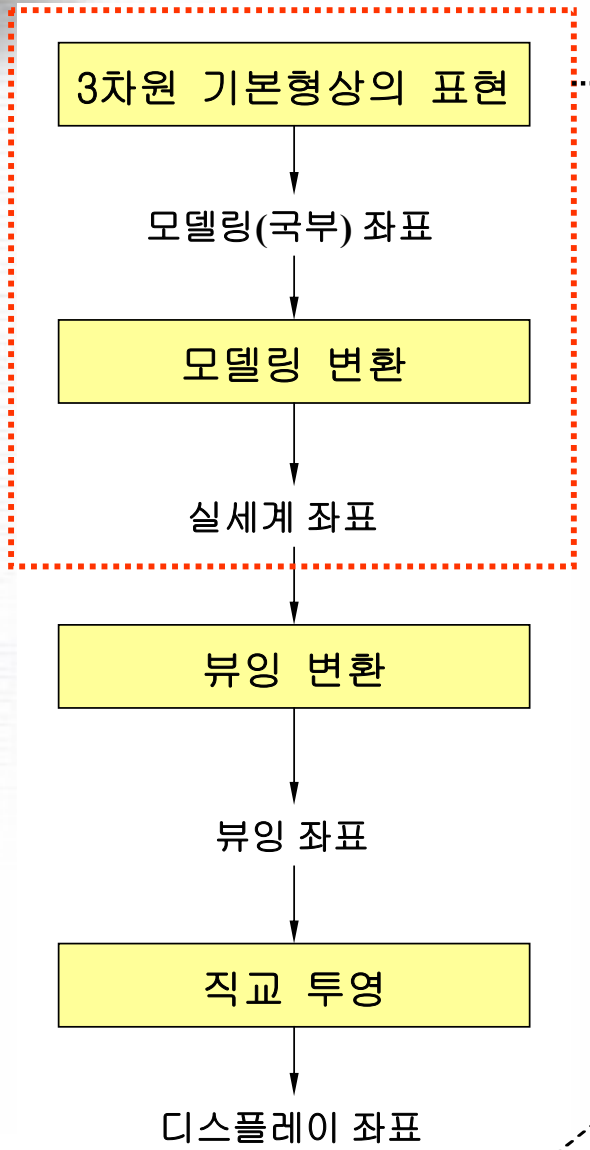
5.2.4 투영 변환 행렬

5.2.5 Viewing to Screen 변환 행렬

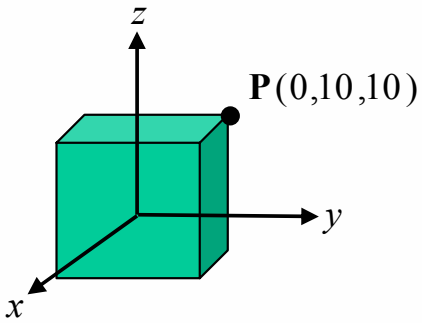
5.2.6 OPenGL Sample Code

Advanced
Ship
Design
Automation
Laboratory

5.2.1 3차원 뷰잉 파이프라인 (1)

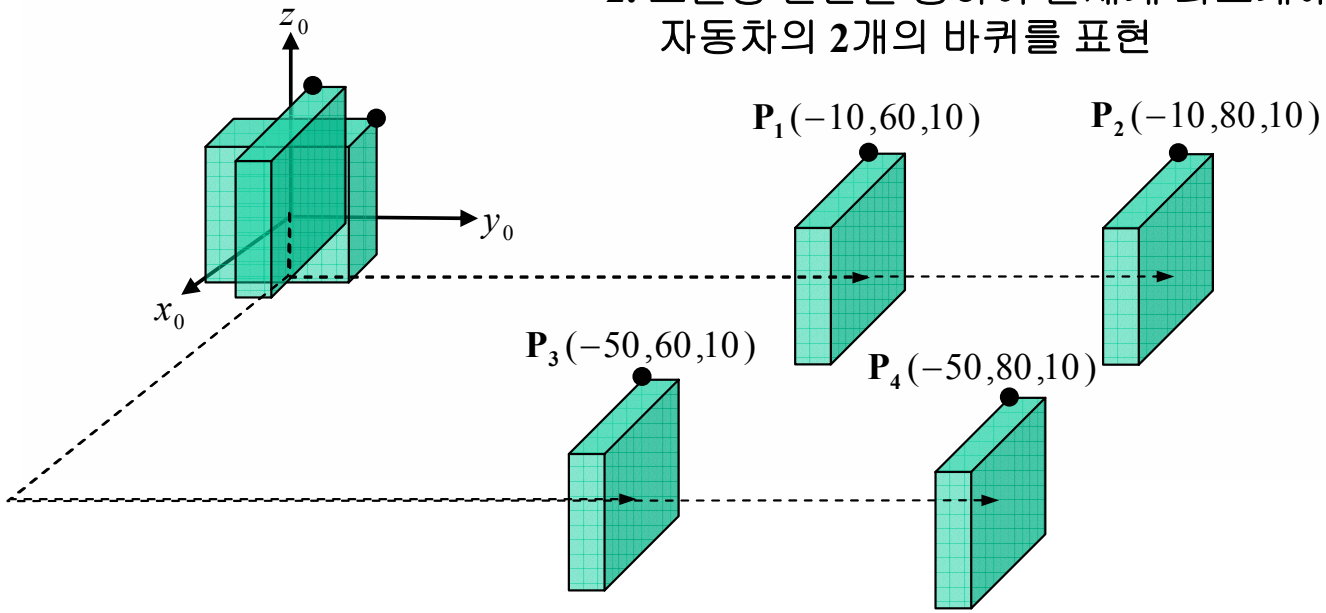


물체를 직접 실세계 좌표계에 표현할 수도 있지만,
자주 반복적으로 사용되는 물체의 경우
물체를 기준으로 한 모델링 (국부) 좌표계로 표현한 후
변환을 통하여 원하는 형상을 표현할 수 있다.
이 때의 변환을 모델링 변환이라고 함

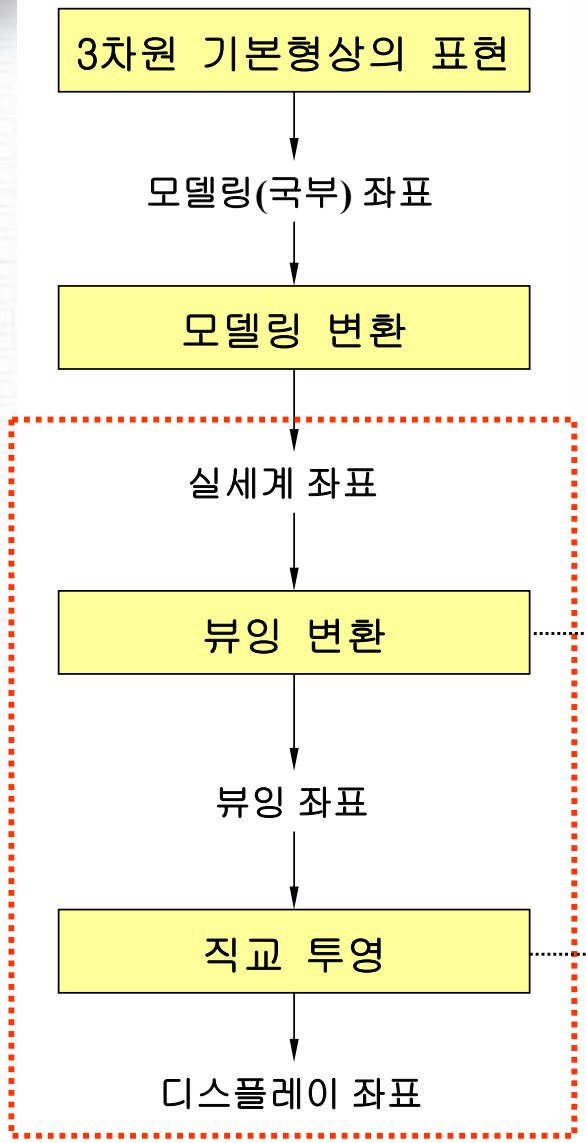


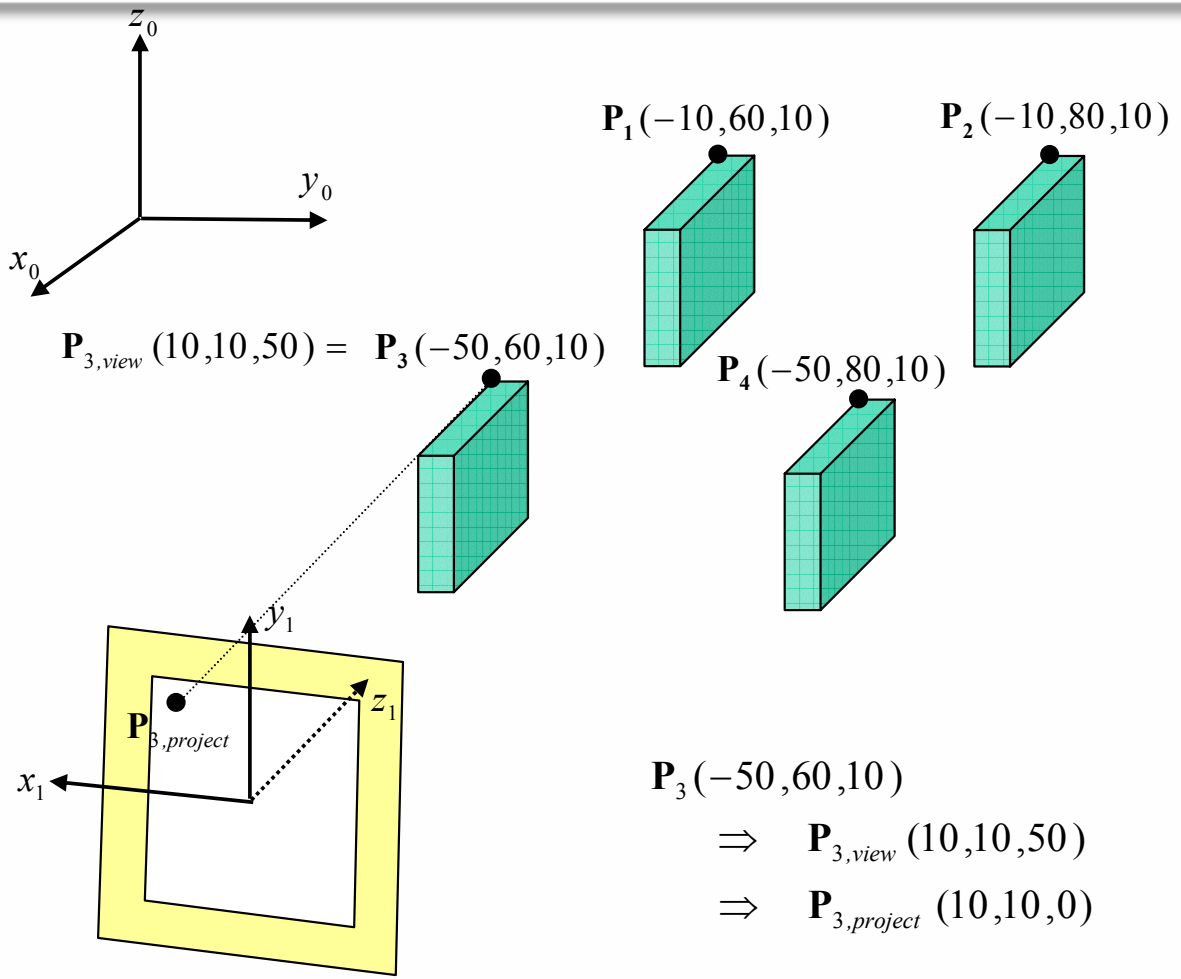
1. 모델링 좌표계를 기준으로
자동차의 바퀴를 정사각형으로 표현

2. 모델링 변환을 통하여 실세계 좌표계에
자동차의 **2**개의 바퀴를 표현



5.2.1 3차원 뷰잉 파이프라인 (2)





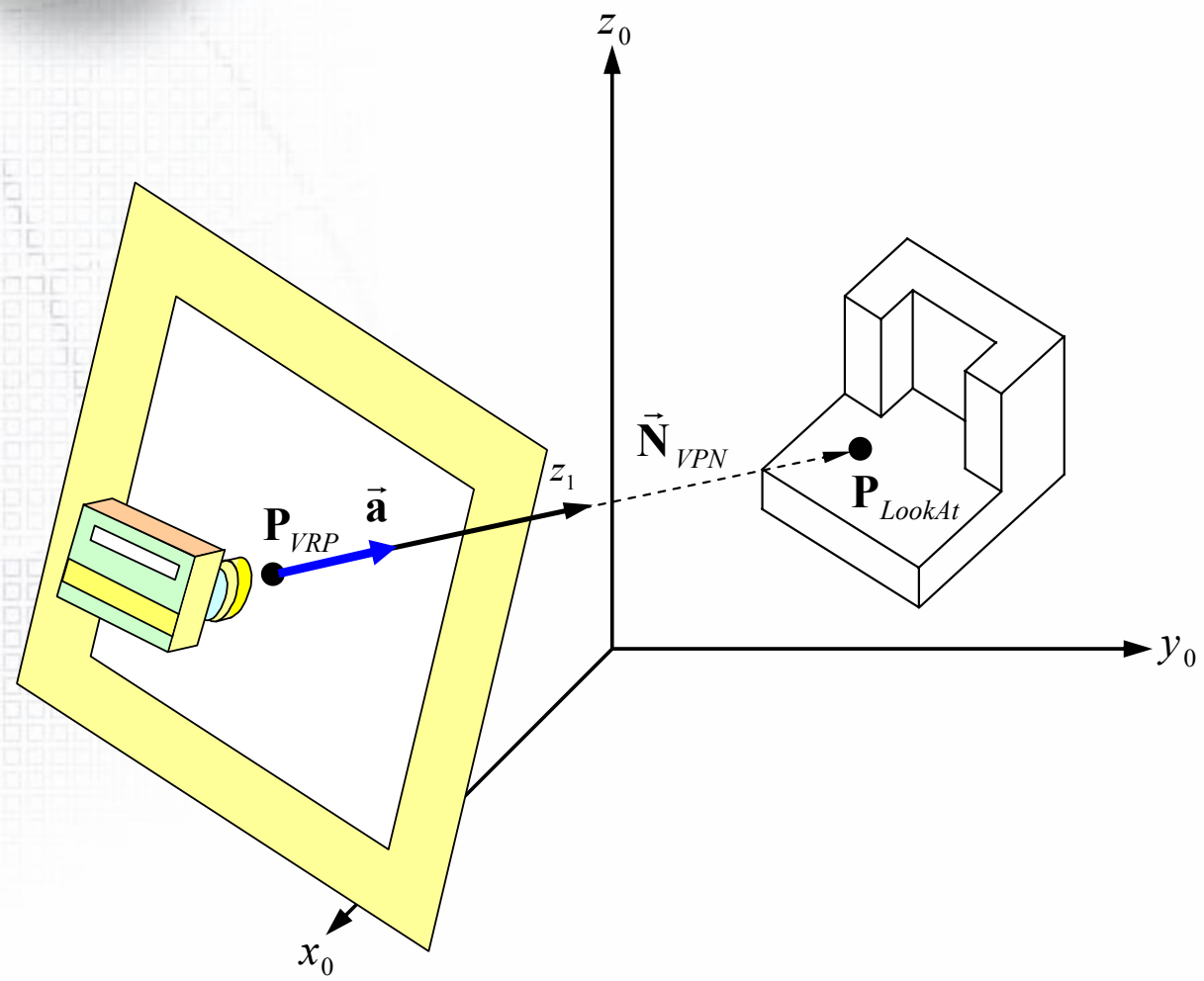
$\mathbf{P}_3(-50,60,10)$

$\Rightarrow \quad \mathbf{P}_{3,view}(10,10,50)$

$\Rightarrow \quad \mathbf{P}_{3,project}(10,10,0)$

실세계 좌표로 표현된 3차원 물체로부터
2차원 모니터에 투영된 좌표를 구하기 위해서는
그후, 뷰잉좌표에 직교 투영변환을 통하여
모니터 상의 디스플레이 좌표를 구할 수 있다.
이를 뷰잉변환이라고 한다.

5.2.2 뷰잉 좌표계의 설정방법 (1)



1. 유저로부터 카메라의 위치 $\mathbf{P}_{VRP}$와 바라보는 점의 위치 $\mathbf{P}_{LookAt}$ 를 실세계 좌표로 입력받는다.

2. 이 정보로 부터 카메라가 물체를 바라보는 방향 $\mathbf{N}_{VPN}$과 투영면을 결정할 수 있다.

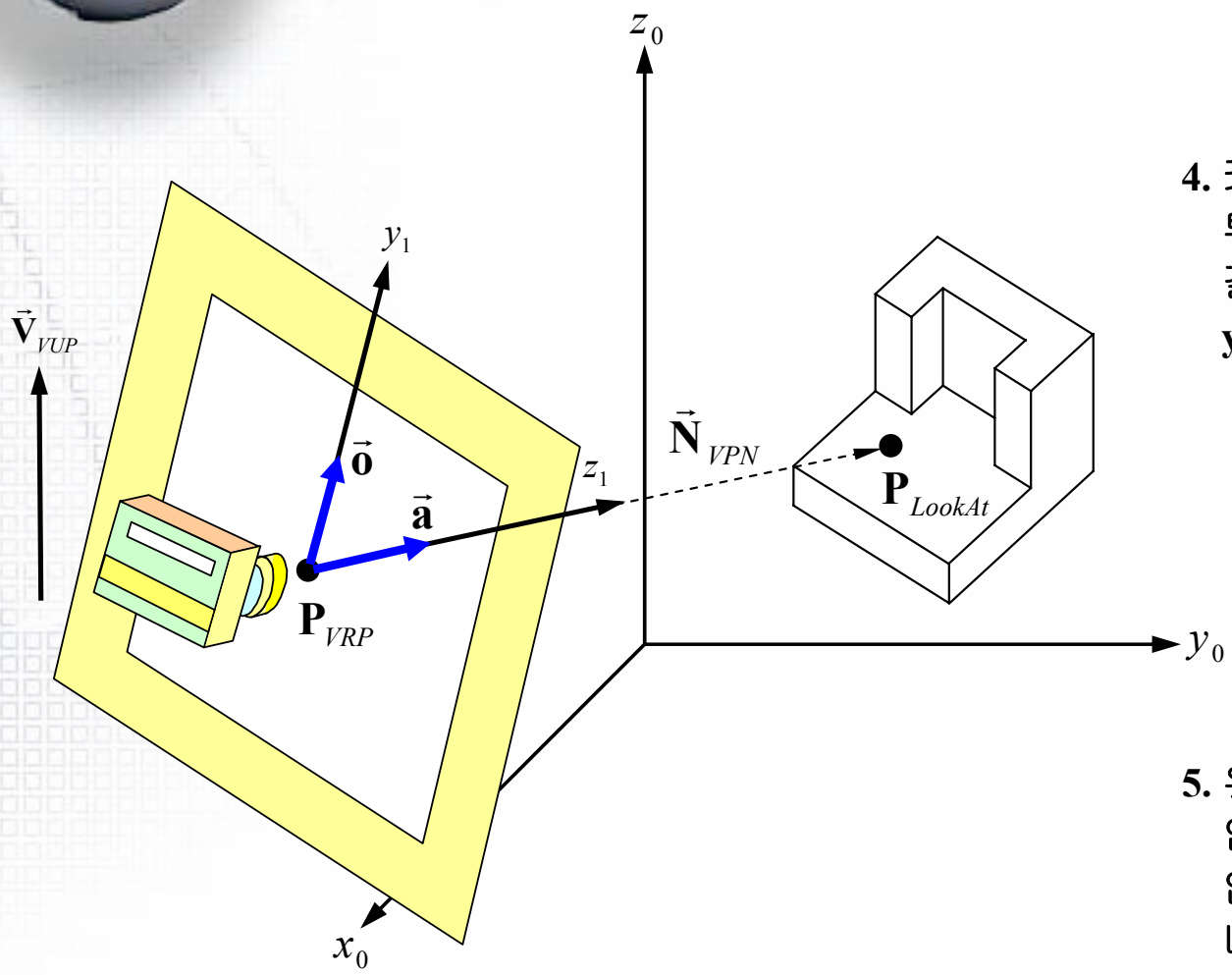
$$\vec{\mathbf{N}}_{VPN} = \mathbf{P}_{LookAt} - \mathbf{P}_{VRP}$$

3. $\mathbf{P}_{VRP}$ 를 뷰잉좌표계의 원점으로 하고, $\mathbf{N}_{VPN}$ 을 뷰잉좌표계의 z축으로 결정한다. 그러면 z축의 단위벡터 a는 다음과 같이 계산할 수 있다

$$\vec{\mathbf{a}} = \frac{\vec{\mathbf{N}}_{VPN}}{\left|\vec{\mathbf{N}}_{VPN}\right|}$$

* VRP: View Reference Point, VPN: View Plane Normal

5.2.2 뷰잉 좌표계의 설정방법 (2)



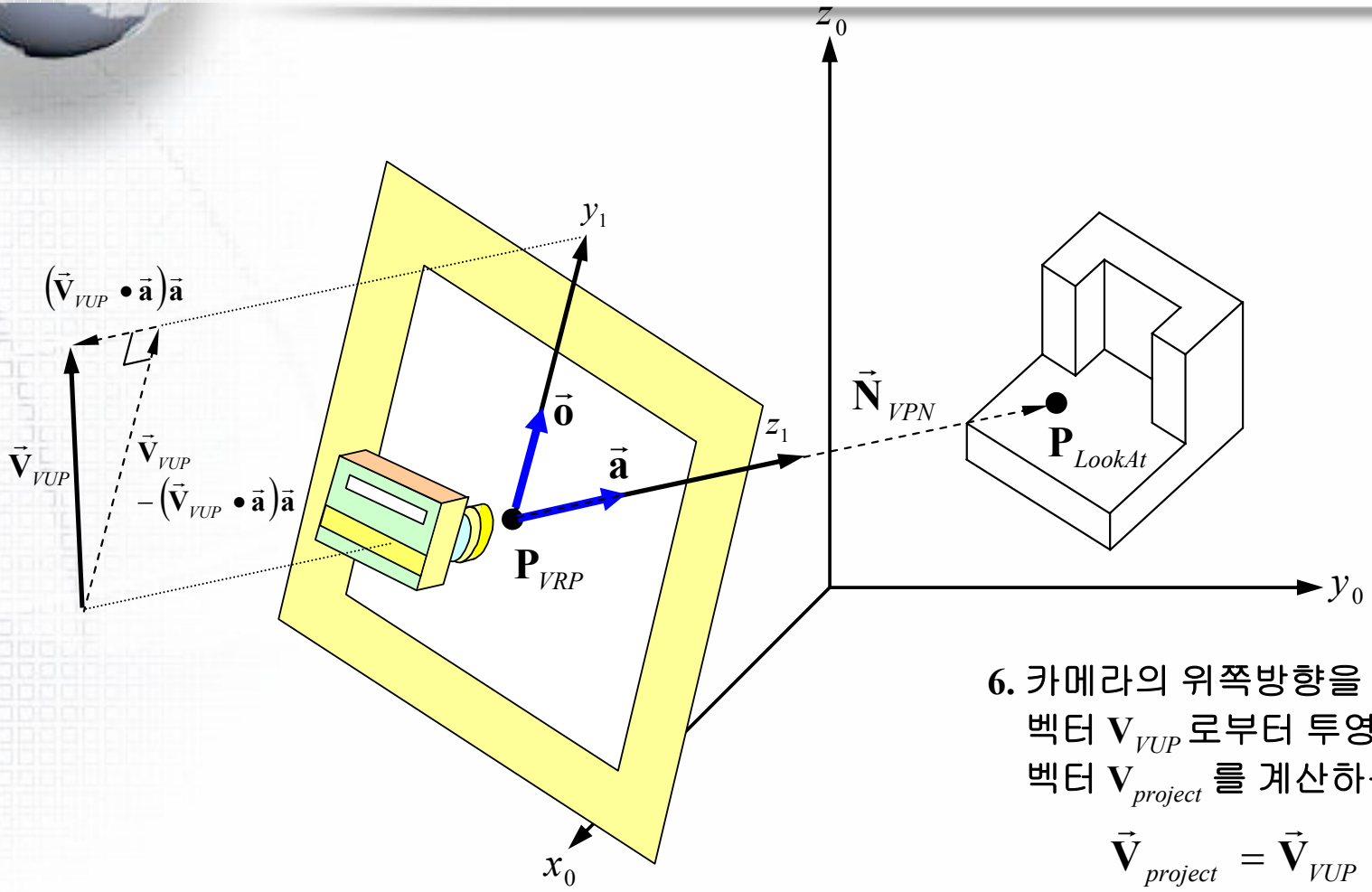
4. 카메라는 $\mathbf{P}_{VRP}$를 중심으로 회전할 수 있으므로 투영면상에 존재하는 카메라의 위쪽방향을 결정해야 한다. 이 때, 이 방향을 뷰잉좌표계의 **y**축으로 결정하고 단위벡터를 **o**라고 하자.

5. 유저가 정확하게 투영면상의 벡터를 입력하기 어려우므로 투영면상에는 존재하지 않을지라도 카메라의 위쪽방향을 쉽게 나타낼 수 있는 개략적인 벡터 $\mathbf{V}_{VUP}$를 입력 받는다.

ex) $\vec{\mathbf{V}}_{VUP} = (0, 0, 1)$

* VUP: View-Up Vector

5.2.2 뷰잉 좌표계의 설정방법 (3)



6. 카메라의 위쪽방향을 나타내는 개략적인 벡터 $\mathbf{V}_{VUP}$ 로부터 투영면상에 존재하는 벡터 $\mathbf{V}_{project}$ 를 계산하는 방법은 다음과 같다.

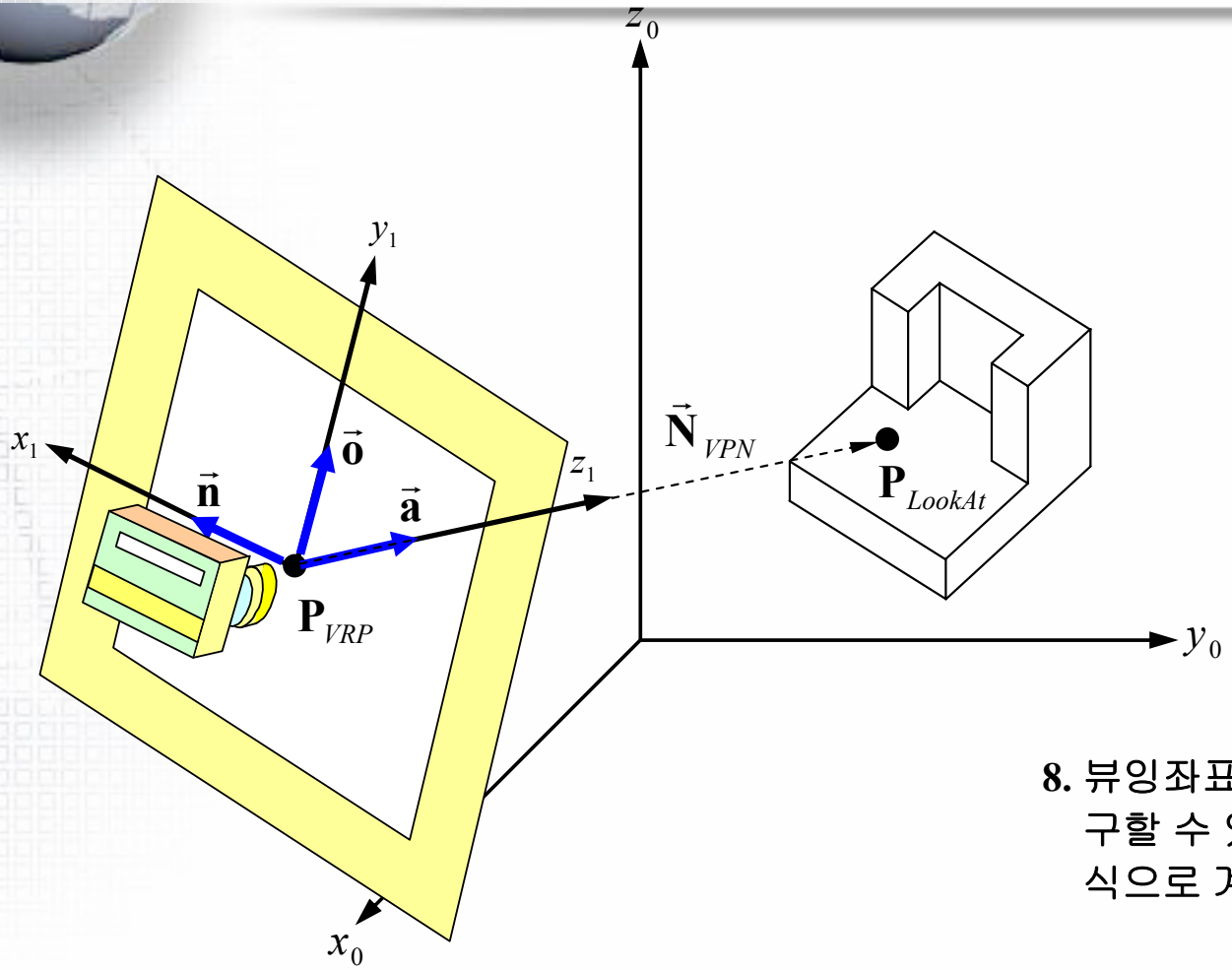
$$\vec{\mathbf{V}}_{project} = \vec{\mathbf{V}}_{VUP} - (\vec{\mathbf{V}}_{VUP} \bullet \vec{\mathbf{a}})\vec{\mathbf{a}}$$

7. 그러면, 뷰잉좌표계의 **y**축 방향의 단위벡터 **o**는 다음과 같이 계산할 수 있다.

$$\vec{\mathbf{o}} = \frac{\vec{\mathbf{V}}_{project}}{\left|\vec{\mathbf{V}}_{project}\right|}$$

* VUP: View-Up Vector

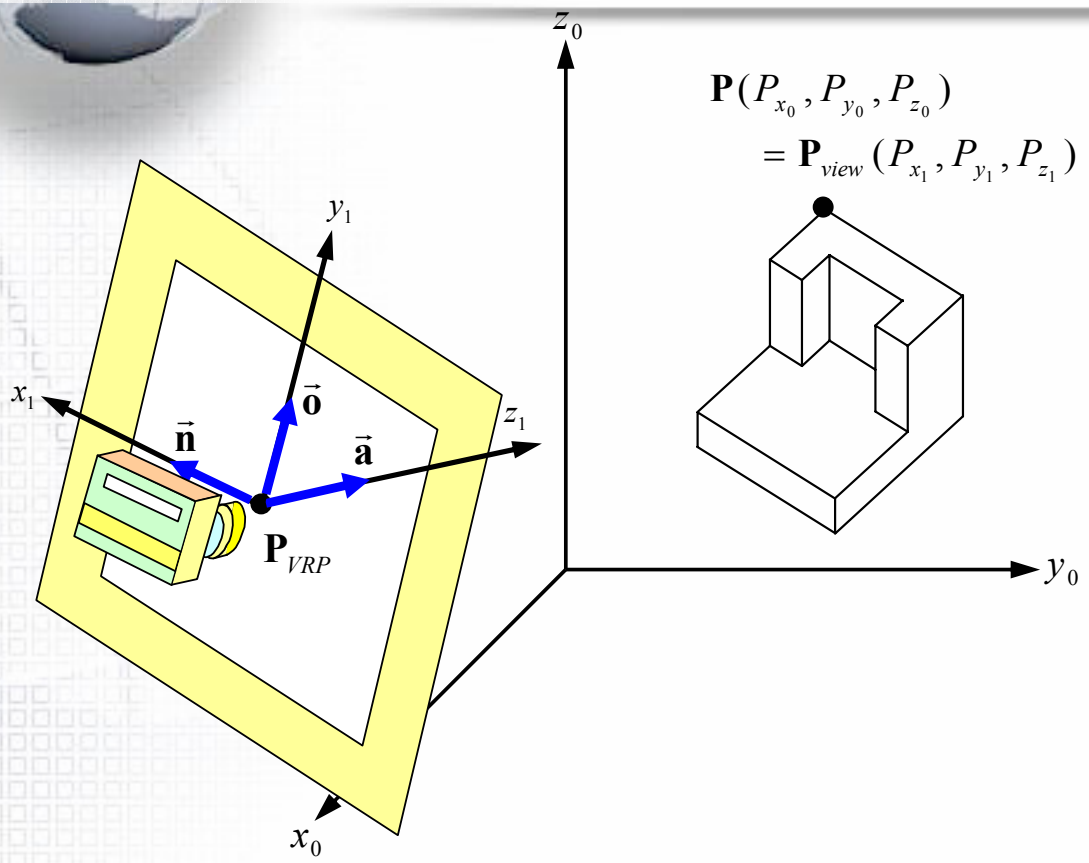
5.2.2 뷰잉 좌표계의 설정방법 (4)



8. 뷰잉좌표계의 **x**축의 방향은 **y**축과 **z**축을 외적하면 구할 수 있다. **x**축 방향의 단위벡터 **n**은 다음의 식으로 계산할 수 있다.

$$\vec{\mathbf{n}} = \vec{\mathbf{o}} \times \vec{\mathbf{a}}$$

5.2.3 뷰잉 변환 행렬



$$\mathbf{P}(P_{x_0}, P_{y_0}, P_{z_0}) = \mathbf{P}_{view}(P_{x_1}, P_{y_1}, P_{z_1})$$

$$\mathbf{P}_{view} = \begin{bmatrix} P_{x_1} & P_{y_1} & P_{z_1} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} P_{x_0} & P_{y_0} & P_{z_0} & 1 \end{bmatrix} \begin{bmatrix} n_{x_0} & o_{x_0} & a_{x_0} & 0 \\ n_{y_0} & o_{y_0} & a_{y_0} & 0 \\ n_{z_0} & o_{z_0} & a_{z_0} & 0 \\ A & B & C & 1 \end{bmatrix}$$

$$\mathbf{F}_{1,0} = \begin{bmatrix} \vec{\mathbf{n}}_{x_0y_0z_0} \\ \vec{\mathbf{o}}_{x_0y_0z_0} \\ \vec{\mathbf{a}}_{x_0y_0z_0} \\ \mathbf{P}_{VRP} \end{bmatrix} = \begin{bmatrix} n_{x_0} & n_{y_0} & n_{z_0} & 0 \\ o_{x_0} & o_{y_0} & o_{z_0} & 0 \\ a_{x_0} & a_{y_0} & a_{z_0} & 0 \\ P_{x_0} & P_{y_0} & P_{z_0} & 1 \end{bmatrix}$$

뷰잉좌표계상의 점을 실세계표계상의 좌표로 매핑하는 변환행렬

$$\mathbf{F}_{0,1} = \mathbf{F}_{0,1}^{-1} = \begin{bmatrix} n_{x_0} & n_{y_0} & n_{z_0} & 0 \\ o_{x_0} & o_{y_0} & o_{z_0} & 0 \\ a_{x_0} & a_{y_0} & a_{z_0} & 0 \\ P_{x_0} & P_{y_0} & P_{z_0} & 1 \end{bmatrix}^{-1}$$

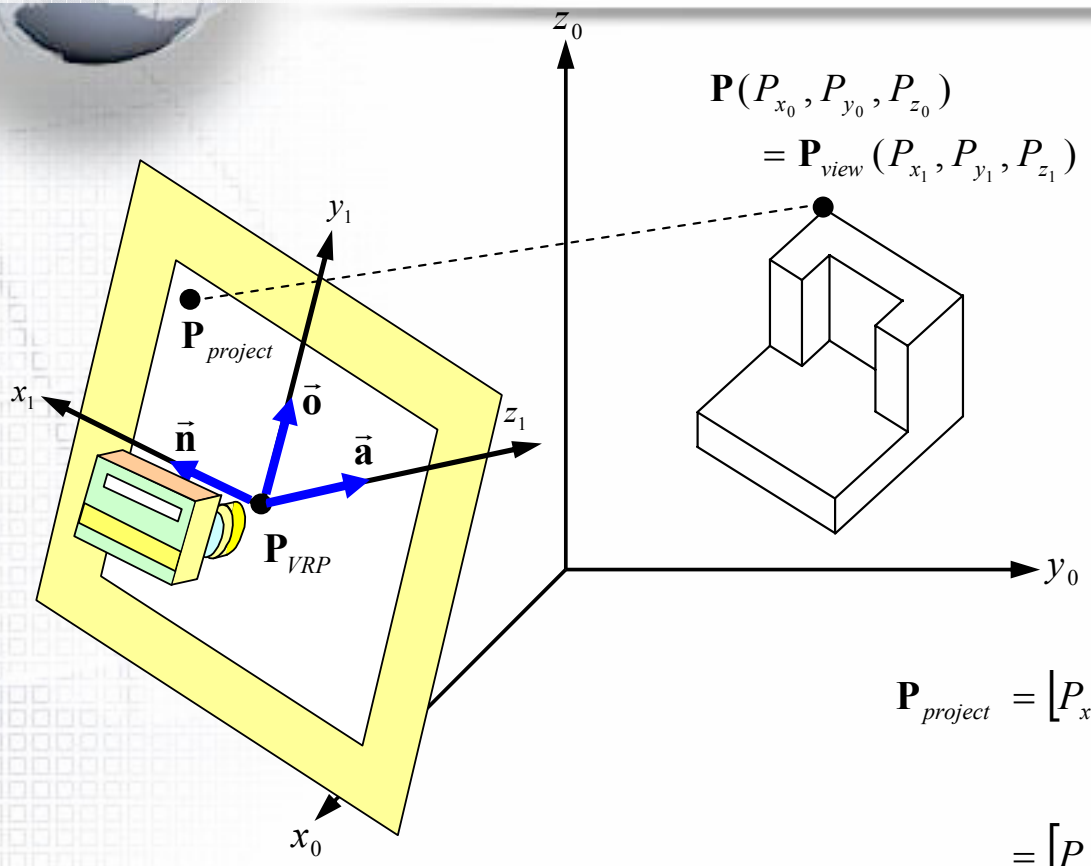
$$= \left(\begin{bmatrix} n_{x_0} & n_{y_0} & n_{z_0} & 0 \\ o_{x_0} & o_{y_0} & o_{z_0} & 0 \\ a_{x_0} & a_{y_0} & a_{z_0} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ P_{x_0} & P_{y_0} & P_{z_0} & 1 \end{bmatrix} \right)^{-1}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -P_{x_0} & -P_{y_0} & -P_{z_0} & 1 \end{bmatrix} \cdot \begin{bmatrix} n_{x_0} & o_{x_0} & a_{x_0} & 0 \\ n_{y_0} & o_{y_0} & a_{y_0} & 0 \\ n_{z_0} & o_{z_0} & a_{z_0} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} n_{x_0} & o_{x_0} & a_{x_0} & 0 \\ n_{y_0} & o_{y_0} & a_{y_0} & 0 \\ n_{z_0} & o_{z_0} & a_{z_0} & 0 \\ A & B & C & 1 \end{bmatrix}$$

실세계좌표계상의 점을 뷰잉좌표계상의 좌표로 매핑하는 변환행렬

5.2.4 투영 변환 행렬



$$\mathbf{T}_{project} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

주어진 점을 xy 평면에
직교투영하는 변환행렬

$$\mathbf{P}_{project} = \begin{bmatrix} P_{x_1} & P_{y_1} & 0 & 1 \end{bmatrix}$$

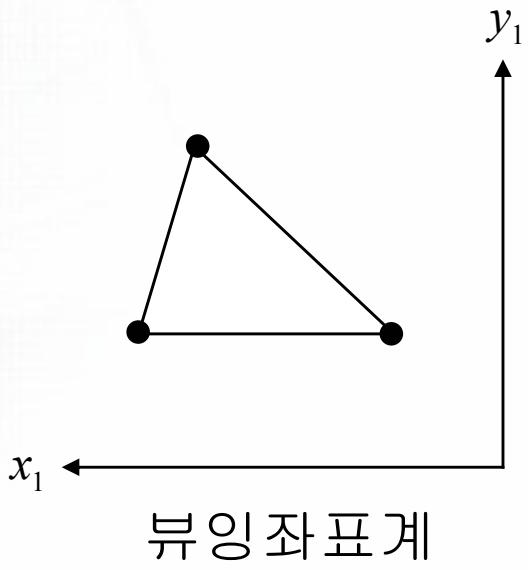
$$= \begin{bmatrix} P_{x_1} & P_{y_1} & P_{z_1} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

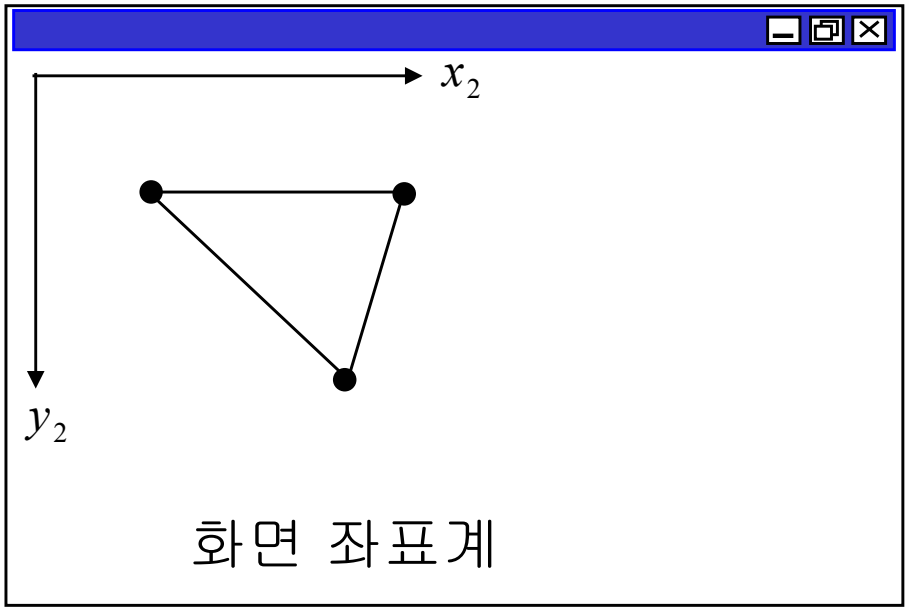
$$= \underbrace{\begin{bmatrix} P_{x_0} & P_{y_0} & P_{z_0} & 1 \end{bmatrix}}_{\mathbf{P}^h} \begin{bmatrix} n_{x_0} & o_{x_0} & a_{x_0} & 0 \\ n_{y_0} & o_{y_0} & a_{y_0} & 0 \\ n_{z_0} & o_{z_0} & a_{z_0} & 0 \\ A & B & C & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

실세계좌표계상의 3차원 점을
모니터상의 2차원 점으로 변환하는 행렬

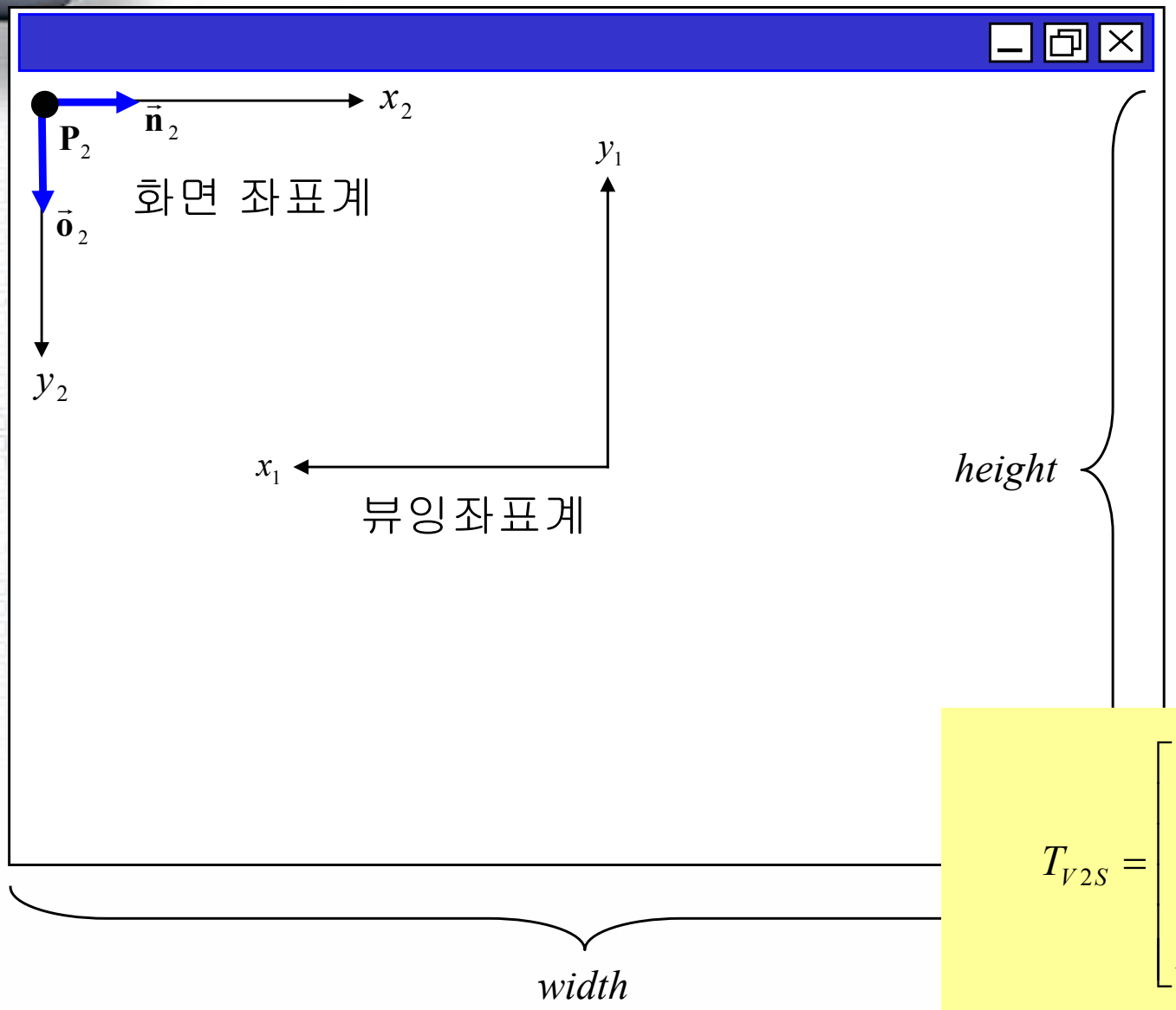
5.2.5 Viewing to Screen 변환 행렬 (1)

- 뷰잉좌표계와 화면좌표계는 x, y 좌표값의 부호가 반대임
- 뷰잉좌표계와 화면좌표계는 원점이 일치하지 않음





5.2.5 Viewing to Screen 변환 행렬 (2)



$$\vec{\mathbf{n}} = \vec{\mathbf{o}} \times \vec{\mathbf{a}}$$

$$\vec{\mathbf{n}}_2 = (-1,0,0)$$

$$\vec{\mathbf{o}}_2 = (0,-1,0)$$

$$\vec{\mathbf{a}}_2 = (0,0,1)$$

$$\mathbf{P}_2 = (\frac{width}{2}, \frac{height}{2}, 0)$$

$$T_{V2S} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ P_{2,x} & P_{2,y} & 0 & 1 \end{bmatrix}$$

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

5.2.6 OpenGL Sample Code

```
Vector VP(0,2,24); //View Point
Vector VS(0,2,0); //View Site
Vector UP(0,1,0); //Up Vector

void display()
{
        glMatrixMode(GL_MODELVIEW); //모델좌표계를 불러옴
        glLoadIdentity(); // 모든 변환 행렬을 초기화
        gluLookAt(VP.x,VP.y,VP.z,VS.x,VS.y,VS.z,UP.x,UP.y,UP.z);
                                                //뷰잉 좌표계를 설정

        glMatrixMode(GL_PROJECTION); //뷰잉 볼륨을 설정하기 위한 전처리
        gluPerspective(45.0, (double)width / (double)height, plane, 1000);
                                                //뷰잉 볼륨을 설정

        glMatrixMode(GL_MODELVIEW);
        glScalef(scale_x,scale_y,scale_z); //확대 비율 설정

        // 그리는 코드 //

        glutSwapBuffers(); //화면에 업데이트
}
```


@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

5.3 3차원 가시화 프로그램 작성

5.3.1 3차원 가시화 클래스 작성 예

5.3.2 3차원 가시화 클래스를 이용한 프로그램 작성 예

5.3.3 MoveTo, LineTo, Line, TextOut 함수 추가

5.3.4 Quadrangle, Polygon, Cube 함수 추가

5.3.5 로봇의 3차원 가시화 프로그램 작성 예

5.3.1 3차원 가시화 클래스 작성 예 (1)

```
#ifndef _Visualization_h_
#define _Visualization_h_

#include "Vector.h"
#include "Matrix3D.h"

class Visualization {
public:
   Visualization();
   ~Visualization();

   // Member Variables
   Vector m_ViewPoint, m_ViewSite, m_UpVector;
   double m_XPanning, m_YPanning;
   double m_Scale;
   Matrix3D  m_Final,
             m_Mapping, m_Projection,
             m_ScaleM, m_Display;
```

```
// Member Function
   void SetViewCoordinate( Vector ViewPoint,
                           Vector ViewSite,
                           Vector UpVector);
   void SetScale(double Scale);
   void SetPanning(double x, double y);
   void CalcMatrix();

   Vector CalcPoint(Vector global_point);
};

#endif
```

5.3.1 3차원 가시화 클래스 작성 예 (2)

```
Visualization::Visualization() {
   m_XPanning = m_YPanning = 0;    m_Scale = 1;
   m_Projection.Set( 1, 0, 0, 0,
                     0, 1, 0, 0,
                     0, 0, 0, 0,
                     0, 0, 0, 1);
}

void Visualization::SetViewCoordinate( Vector ViewPoint, Vector ViewSite, Vector UpVector) {
   m_ViewPoint = ViewPoint; m_ViewSite = ViewSite; m_UpVector = UpVector;
   Vector n, o, a;      Vector P = m_ViewPoint;

   a = m_ViewSite - m_ViewPoint;    a.Normalize();
   o = m_UpVector -(m_UpVector%a)*a;     o.Normalize();
   n = o*a;   n.Normalize();

    Matrix3D   InverseTrans, InverseRot;
    InverseTrans.Set(…..); InverseRot.Set(….);   m_Mapping = InverseTrans * InverseRot;
    CalcMatrix();
}
```

5.3.1 3차원 가시화 클래스 작성 예 (3)

```
void Visualization::SetPanning(double x, double y){
        m_XPanning = x; m_YPanning = y;
        m_Display.Set( -1, 0, 0, 0,
                        0, -1, 0, 0,
                        0,  0, -1, 0,
                        x,  y, 0, 1);
         CalcMatrix();
}

void Visualization::CalcMatrix()
{
            m_Final = m_Mapping*m_Projection*m_ScaleM*m_Display;
}

Vector Visualization::CalcPoint(Vector global_point)
{
            Vector temp = global_point*m_Final;
            return temp;
}
```

5.3.2 3차원 가시화 클래스를 이용한 프로그램 작성 예 (1)

```
#include "Visualization.h"
#include "Vector.h"

class CVisualView : public CView
{
protected: // create from serialization only
        CVisualView();
        DECLARE_DYNCREATE(CVisualView)

// Attributes
public:
        CVisualDoc* GetDocument();

        Visualization m_Vis;
        Vector m_origin, m_x, m_y, m_z;

        .....
};
```

5.3.2 3차원 가시화 클래스를 이용한 프로그램 작성 예 (2)

```
void C……View::OnSize(UINT nType, int cx, int cy)
{
        CView::OnSize(nType, cx, cy);

        // TODO: Add your message handler code here
        CRect rect;
        GetClientRect(&rect);

        int width = rect.Width();
        int height = rect.Height();

        m_Vis.SetPanning(  width / 2., height / 2 .);
}
```

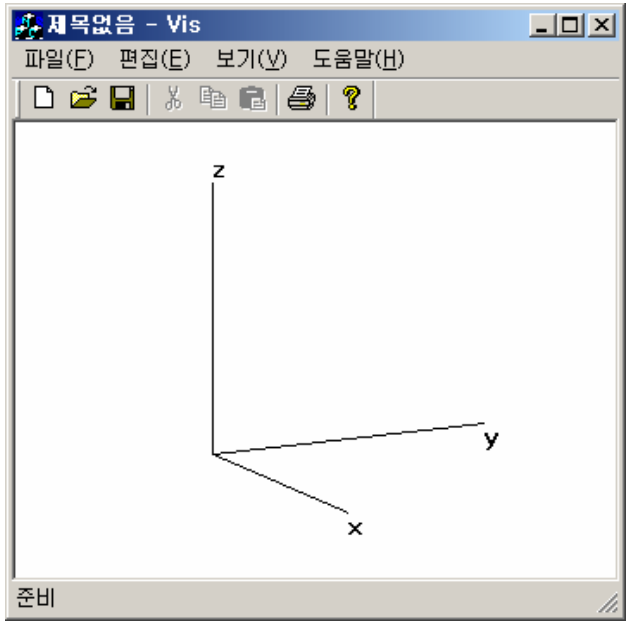
5.3.2 3차원 가시화 클래스를 이용한 프로그램 작성 예 (3)

```
Vector origin(0,0,0);  Vector x(100,0,0);  Vector y(0,100,0);  Vector z(0,0,100);

temp = m_Vis.CalcPoint(origin);
pDC->MoveTo(temp.x, temp.y);
temp = m_Vis.CalcPoint(x);
pDC->LineTo(temp.x, temp.y);
pDC->TextOut(temp.x, temp.y, "x");

temp = m_Vis.CalcPoint(origin);
pDC->MoveTo(temp.x, temp.y);
temp = m_Vis.CalcPoint(y);
pDC->LineTo(temp.x, temp.y);
pDC->TextOut(temp.x, temp.y, "y");

temp = m_Vis.CalcPoint(origin);
pDC->MoveTo(temp.x, temp.y);
temp = m_Vis.CalcPoint(z);
pDC->LineTo(temp.x, temp.y);
pDC->TextOut(temp.x, temp.y, "z");
```

사용법이 너무 번거로움

```
m_Vis.MoveTo(pDC, origin);
m_Vis.LineTo(pDC, x);
```

3차원 가시화 클래스 내부에서 처리하게 할 수는 없을까**?**



5.3.3 MoveTo, LineTo, Line, TextOut 함수 추가 (1)

```
class Visualization {
public:
   ..................
   // Drawing Functions
   void MoveTo( CDC* pDC, Vector point );
   void LineTo( CDC* pDC, Vector point );
   void Line( CDC* pDC, Vector point1,
                              Vector point2 );
   void TextOut(CDC* pDC, Vector point,
                              CString text);
..................
};
```

```
void Visualization::MoveTo(CDC* pDC, Vector point) {
    Vector temp = CalcPoint(point);
    pDC->MoveTo(temp.x, temp.y);
}
void Visualization::LineTo(CDC* pDC, Vector point) {
    Vector temp = CalcPoint(point);
    pDC->LineTo(temp.x, temp.y);
}
void Visualization::Line(CDC* pDC, Vector point1, Vector point2) {
    MoveTo(pDC, point1);
    LineTo(pDC, point2);
}
void Visualization::TextOut(CDC* pDC, Vector point, CString text) {
    Vector temp = CalcPoint(point);
    pDC->TextOut(temp.x, temp.y, text);
}
```

```
temp = m_Vis.CalcPoint(origin);
pDC->MoveTo(temp.x, temp.y);
temp = m_Vis.CalcPoint(x);
pDC->LineTo(temp.x, temp.y);
```

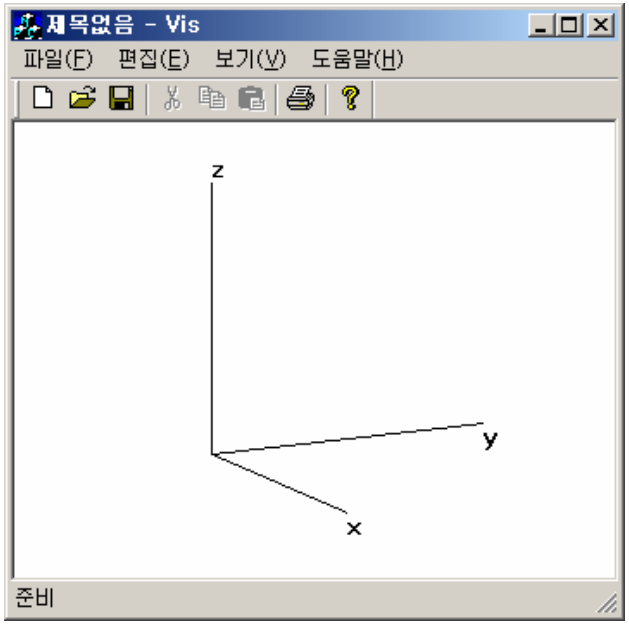
```
m_Vis.MoveTo(pDC, origin);
m_Vis.LineTo(pDC, x);
```

```
m_Vis.Line(pDC, origin, x);
m_Vis.TextOut(pDC, x, "x");
```

5.3.3 MoveTo, LineTo, Line, TextOut 함수 추가 (2)

```
temp = m_Vis.CalcPoint(origin);
pDC->MoveTo(temp.x, temp.y);
temp = m_Vis.CalcPoint(x);
pDC->LineTo(temp.x, temp.y);
pDC->TextOut(temp.x, temp.y, "x");

temp = m_Vis.CalcPoint(origin);
pDC->MoveTo(temp.x, temp.y);
temp = m_Vis.CalcPoint(y);
pDC->LineTo(temp.x, temp.y);
pDC->TextOut(temp.x, temp.y, "y");

temp = m_Vis.CalcPoint(origin);
pDC->MoveTo(temp.x, temp.y);
temp = m_Vis.CalcPoint(z);
pDC->LineTo(temp.x, temp.y);
pDC->TextOut(temp.x, temp.y, "z");
```

```
m_Vis.Line(pDC, origin, x);
m_Vis.TextOut(pDC, x, "x");

m_Vis.Line(pDC, origin, y);
m_Vis.TextOut(pDC, y, "y");

m_Vis.Line(pDC, origin, z);
m_Vis.TextOut(pDC, z, "z");
```



5.3.4 Quadrangle, Polygon, Cube 함수 추가 (1)

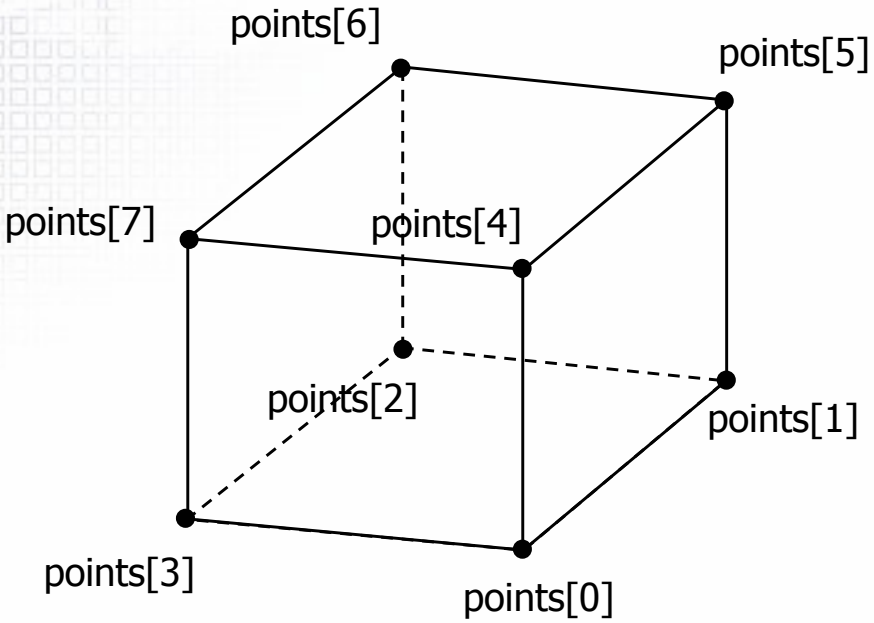
```
class Visualization {
public:
   ..................
   // Drawing Functions
   void Quadrangle(CDC* pDC, Vector point1,
                   Vector point2, Vector point3,
                   Vector point4);
   void Polygon(CDC* pDC, int nPoint, Vector* points);
..................
};
```

```
void Visualization::Quadrangle(CDC* pDC, Vector point1,
                   Vector point2, Vector point3, Vector point4){
   MoveTo(pDC, point1);
   LineTo(pDC, point2);
   LineTo(pDC, point3);
   LineTo(pDC, point4);
   LineTo(pDC, point1);
}
void Visualization::Polygon(CDC* pDC, int nPoint, Vector* points)
{
   MoveTo(pDC, points[0]);
   for(int i = 1; i < nPoint; i++){
      LineTo(pDC, points[i]);
   }
   LineTo(pDC, points[0]);
}
```

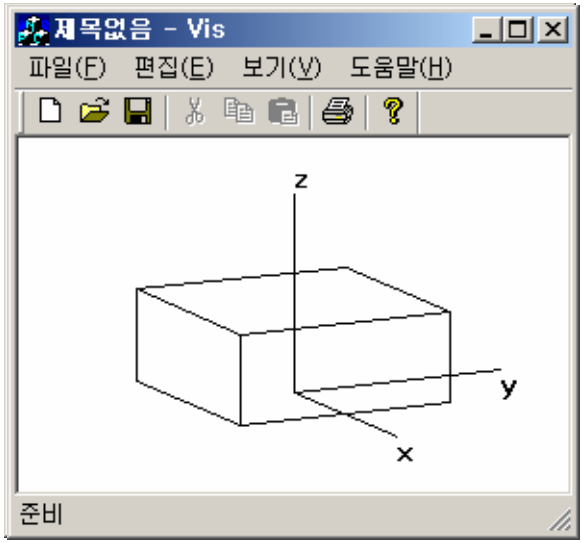
5.3.4 Quadrangle, Polygon, Cube 함수 추가 (2)

```
class Visualization {
public:
   ..................
   // Drawing Functions
   void Cube(CDC* pDC, Vector* points);
..................
};
```

```
void Visualization::Cube(CDC* pDC, Vector* points) {
   Quadrangle(pDC, points[0], points[1], points[2], points[3]);
   Quadrangle(pDC, points[4], points[5], points[6], points[7]);
   Quadrangle(pDC, points[0], points[4], points[7], points[3]);
   Quadrangle(pDC, points[3], points[7], points[6], points[2]);
   Quadrangle(pDC, points[2], points[6], points[5], points[1]);
   Quadrangle(pDC, points[1], points[5], points[4], points[0]);
}
```




```
Vector points[8];
points[0].Set(10,10,0);
points[1].Set(0,10,0);
..............
points[7].Set(10,0,10);

m_Vis.Cube(pDC, points);
```



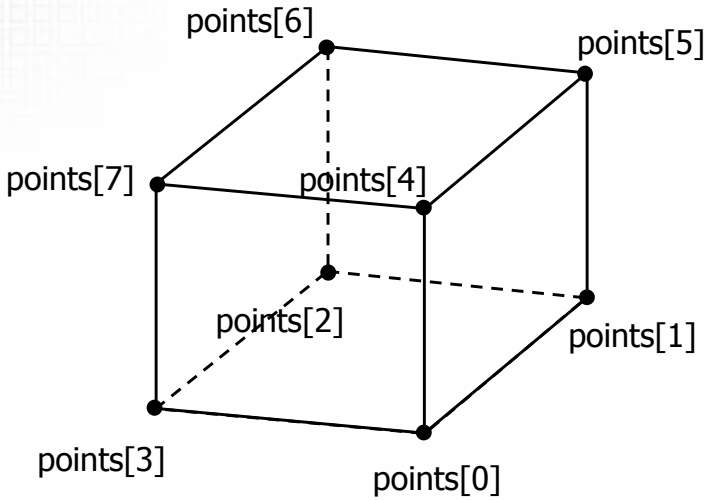
SDAL
Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

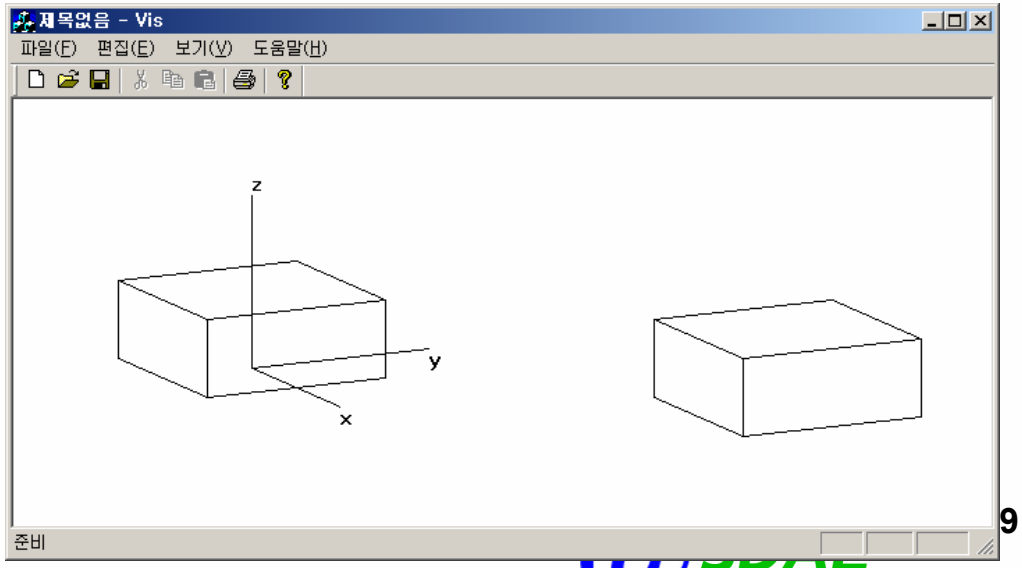
5.3.5 로봇의 3차원 가시화 프로그램 작성 예 (1)

```
void Visualization::CalcMatrix() {
        m_Final =  m_Modeling*  m_Mapping*m_Projection*m_ScaleM*m_Display;
}

Vector Visualization::CalcPoint(Vector global_point) {
        Vector temp = global_point*m_Final;
        return temp;
}

void Visualization::SetIdentityModelingMatrix() {
        m_Modeling.Set(1,0,0,0,    0,1,0,0,    0,0,1,0,    0,0,0,1);
         CalcMatrix();
}
void Visualization::SetTranslationMatrix(double x, double y, double z) {
        Matrix3D Trans;
        Trans.Set(1,0,0,0,   0,1,0,0,  0,0,1,0,  x,y,z,1);

        m_Modeling = Trans * m_Modeling;
        CalcMatrix();
}
```

5.3.5 로봇의 3차원 가시화 프로그램 작성 예 (2)

```
Vector points[8];
points[0].Set(10,10,0);
points[1].Set(0,10,0);
.....
points[7].Set(10,0,10);

m_Vis.SetIdentityModelingMatrix();
m_Vis.Cube(pDC, points);

m_Vis.SetTranslationMatrix(200, 200, 0);
m_Vis.Cube(pDC, points);
```





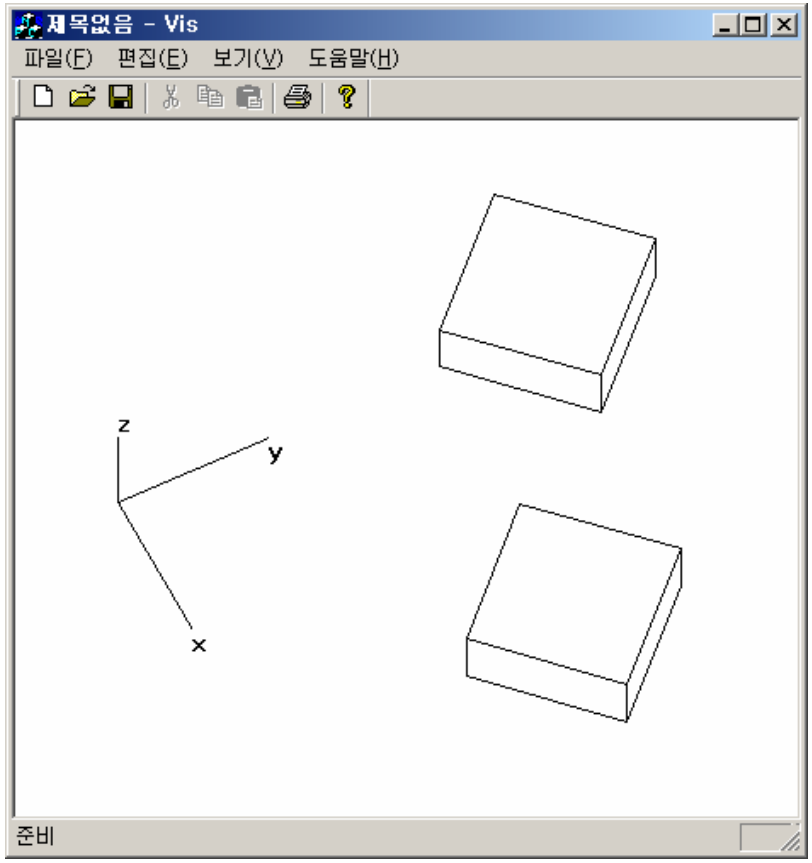
5.3.5 로봇의 3차원 가시화 프로그램 작성 예 (3)

```
#ifndef M_PI
#define M_PI       3.14159265358979323846
#endif

void Visualization::SetRotationMatrix_X(double theta)
{
        double theta_rad = theta*M_PI/180.;
        double c = cos(theta_rad);  double s = sin(theta_rad);
        Matrix3D RotX;
        RotX.Set(1,0,0,0,
                 0,c,s,0,
                 0,-s,c,0,
                 0,0,0,1);

         m_Modeling = RotX * m_Modeling;
         CalcMatrix();
}
```

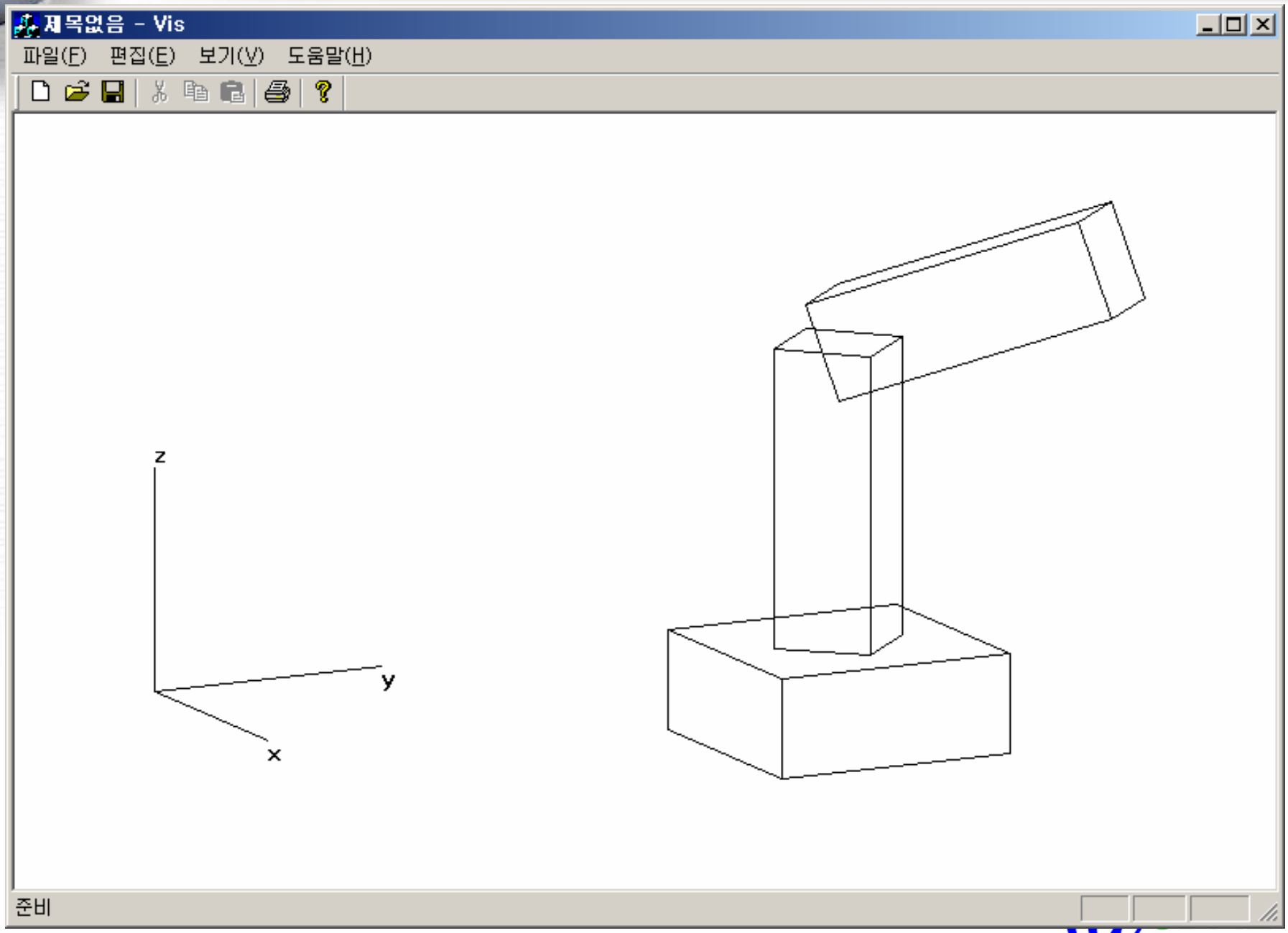

@SDAL Advanced Ship Design Automation Lab. http://asdal.snu.ac.kr

5.3.5 로봇의 3차원 가시화 프로그램 작성 예 (4)

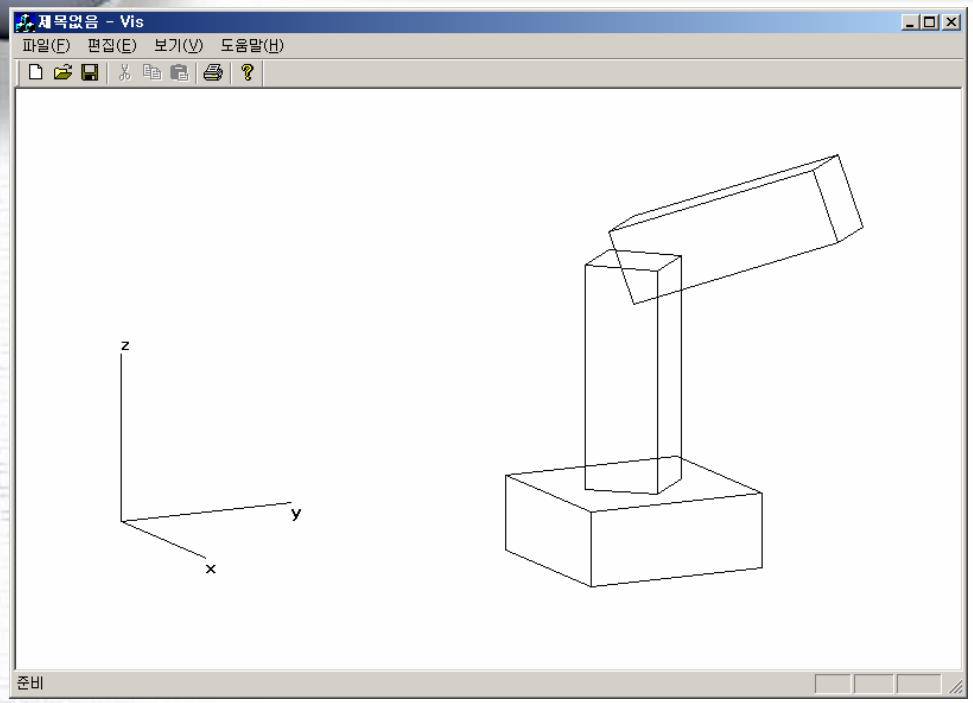
```
Vector points[8];
points[0].Set(10,10,0);
points[1].Set(0,10,0);
.....
points[7].Set(10,0,10);

m_Vis.SetIdentityModelingMatrix();
m_Vis.SetTranslationMatrix(200, 200, 0);
m_Vis.SetRotationMatrix_Z(45);
m_Vis.Cube(pDC, points);

m_Vis.SetIdentityModelingMatrix();
m_Vis.SetRotationMatrix_Z(45);
m_Vis.SetTranslationMatrix(200, 200, 0);
m_Vis.Cube(pDC, points);
```

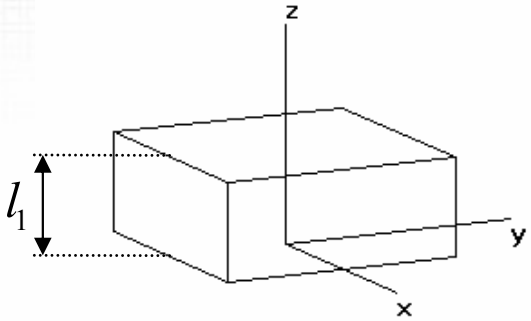


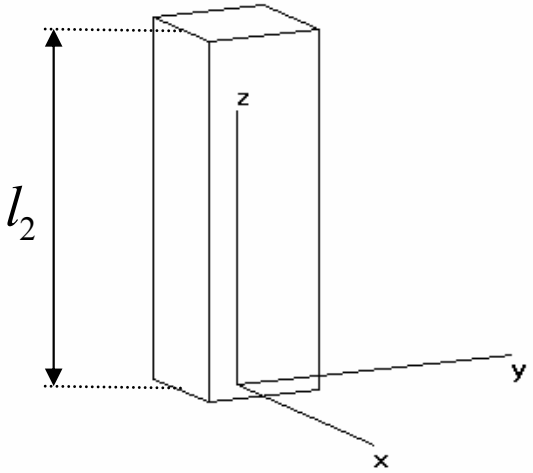
5.3.5 로봇의 3차원 가시화 프로그램 작성 예 (5)

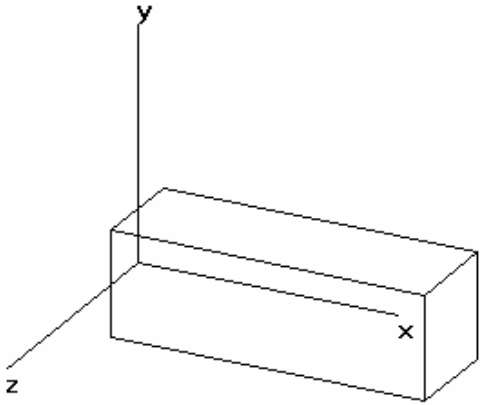


5.3.5 로봇의 3차원 가시화 프로그램 작성 예 (6)



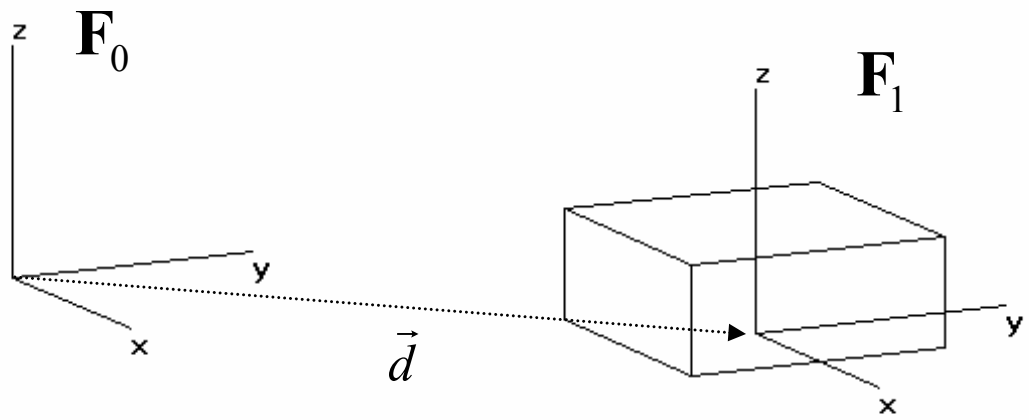






@SDAL Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

5.3.5 로봇의 3차원 가시화 프로그램 작성 예 (7)

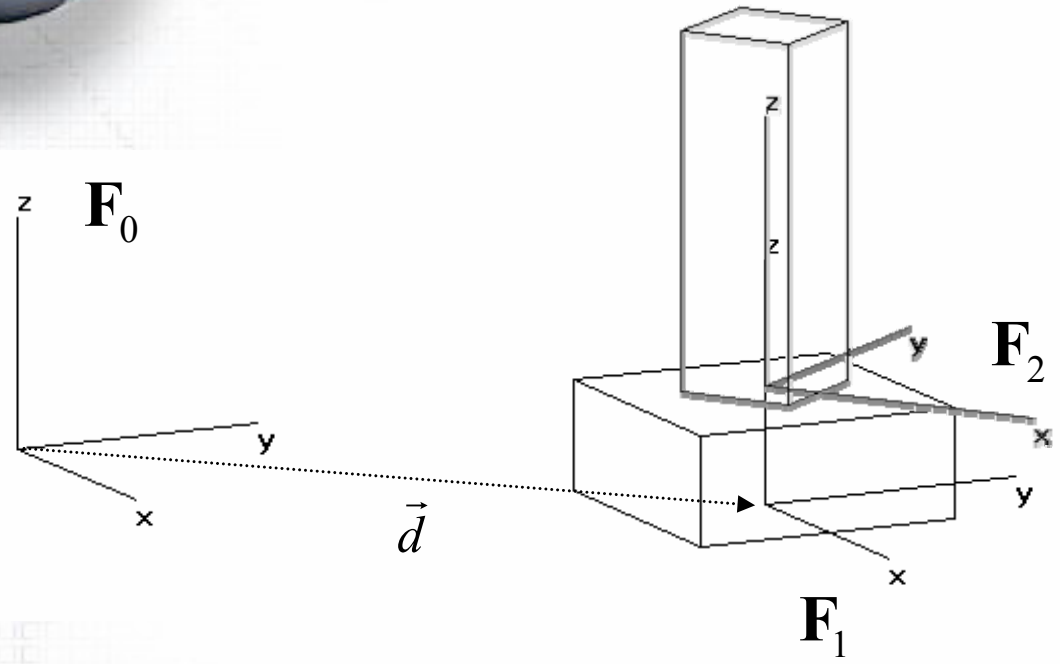


$$\mathbf{F}_{1,0} = Trans(\vec{d}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ d_x & d_y & d_z & 1 \end{bmatrix}$$

```
// For Robotics
m_BasePosition.Set(200,200,0);
m_Base[0].Set(50,50,0);
m_Base[1].Set(-50,50,0);
m_Base[2].Set(-50,-50,0);
m_Base[3].Set(50,-50,0);
m_Base[4].Set(50,50,40);
m_Base[5].Set(-50,50,40);
m_Base[6].Set(-50,-50,40);
m_Base[7].Set(50,-50,40);
```

```
// For Robot
m_Vis.SetIdentityModelingMatrix();
m_Vis.SetTranslationMatrix(m_BasePosition.x, m_BasePosition.y, m_BasePosition.z);

   m_Vis.Cube(pDC, m_Base);
```

5.3.5 로봇의 3차원 가시화 프로그램 작성 예 (8)



$$\mathbf{F}_{2,1} = Trans(z, l_1) \cdot Rot(z, \theta_1)$$

$$\mathbf{F}_{2,0} = \mathbf{F}_{2,1} \cdot \mathbf{F}_{1,0}$$
$$= Trans(z, l_1) \cdot Rot(z, \theta_1) \cdot Trans(d)$$

```
m_length1 = 40;
m_Link1[0].Set(20,20,0);
m_Link1[1].Set(-20,20,0);
m_Link1[2].Set(-20,-20,0);
m_Link1[3].Set(20,-20,0);
m_Link1[4].Set(20,20,120);
m_Link1[5].Set(-20,20,120);
m_Link1[6].Set(-20,-20,120);
m_Link1[7].Set(20,-20,120);
m_theta1 = 0;
```

```
// For Robot
m_Vis.SetIdentityModelingMatrix();
m_Vis.SetTranslationMatrix(m_BasePosition.x, m_BasePosition.y, m_BasePosition.z);
    m_Vis.Cube(pDC, m_Base);

m_Vis.SetRotationMatrix_Z(m_theta1);
m_Vis.SetTranslationMatrix(0, 0, m_length1);
    m_Vis.Cube(pDC, m_Link1);
```

참고 문헌

- **박종우**, A First Course in Robot Mechanics, **서울대학교 기계학공공학부 2005년** Introduction to Robotics **강의 교재**
- J.J. Craig, Introduction to Robotics: Mechanics and Control, Prentice Hall, 2004
- S.B. Niku, Introduction to Robotics: Analysis, Systems, Applications, Prentice Hall, 2001
- **이현찬, 채수원,최영 공역, 컴퓨터 그래픽스 및 형상 모델링, 시그마프레스**, 1996