Xilinx Zynq FPGA, TI DSP, MCU 프로그래밍 및 회로 설계 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

Transform Mathematics

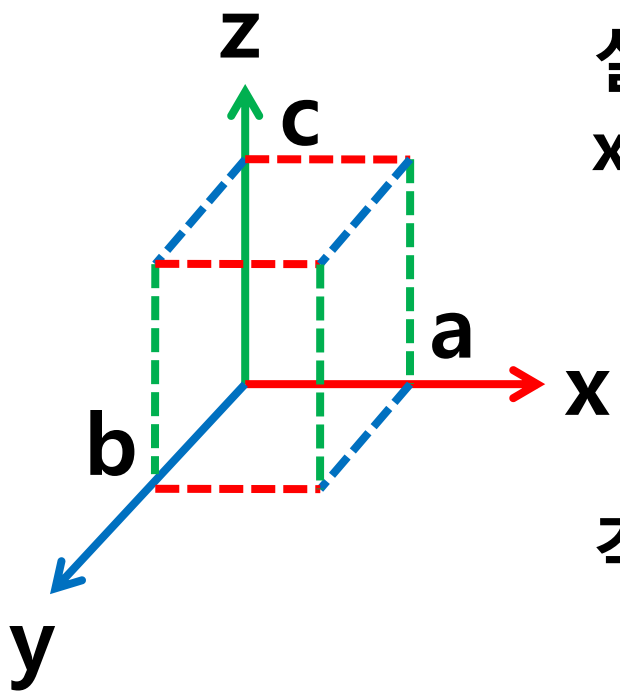
Transformation

마찬가지로 과거 3D 게임 프로그래밍 과정에서 사용했던 자료다.
쿼터니온을 이해하기 위해 필요한 최소한의 지식임으로 잠시 일부만을 활용해보도록 한다.

변환에서 가장 기본적인 것은 바로 좌표 이동이다.
중학생 때도 공부했던 바로 평행이동이다.

그럼 이제 변환을 본격적으로 들어가보자!

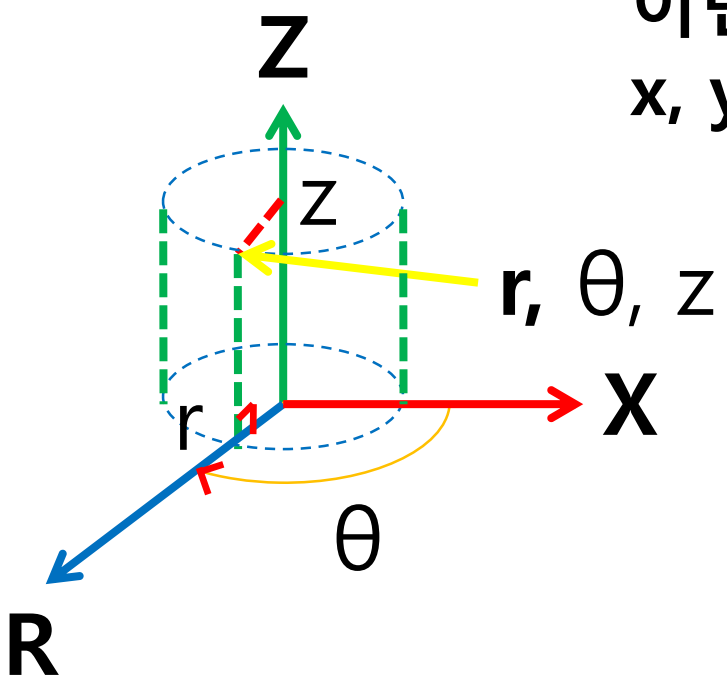
Cartesian Coordinates



실선 화살표를 좌표축이라 하며
x, y, z축이라고 불린다.

각축들의 접점은 원점이라고함

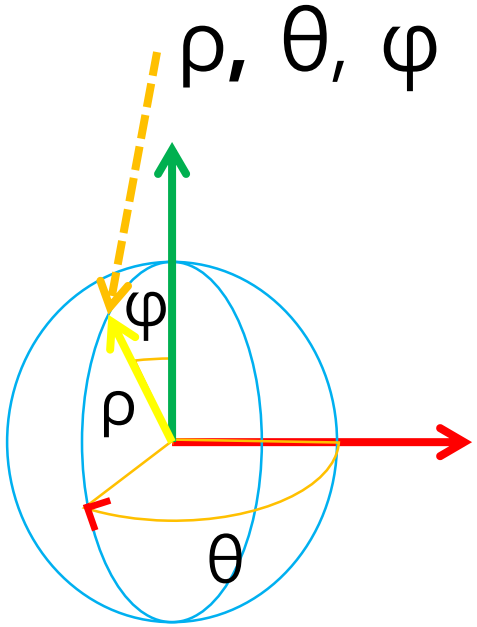
Cylindrical Coordinates



이런 원기둥 좌표계의 값을
x, y, z로 옮길 때는 아래와 같이함

$$x = r \cos \theta$$
$$y = r \sin \theta$$
$$z = z$$

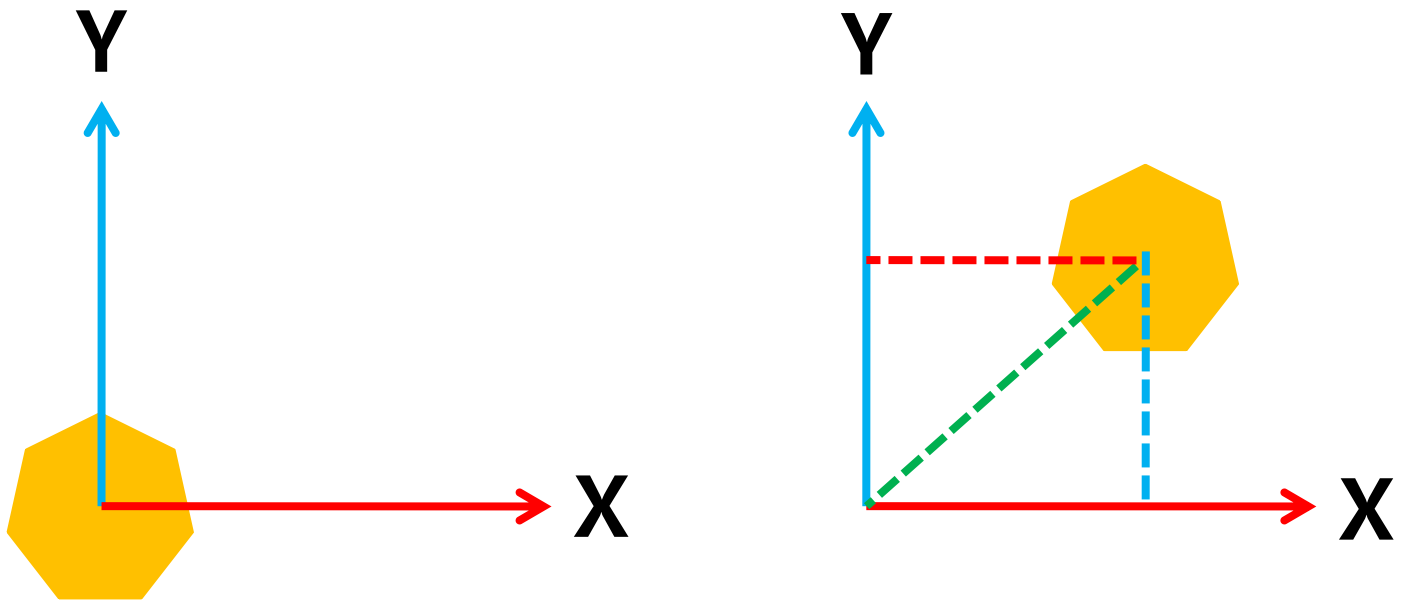
Spherical Coordinates



구면 좌표계의 값을
x, y, z로 옮길 때는 아래와 같이함

$$x = \rho \cos\theta \sin\theta$$
$$y = \rho \sin\theta \sin\varphi$$
$$z = \rho \cos\varphi$$

Parallel Transference



X축을 따라 x만큼
Y축을 따라 y만큼 평행 이동!

Rotational Transference

이동 변환은 각 좌표축에 대한
이동 크기를 더해줌으로써 수행되지만
회전 변환은 주어진 각도만큼
회전축을 기준으로 회전한다.

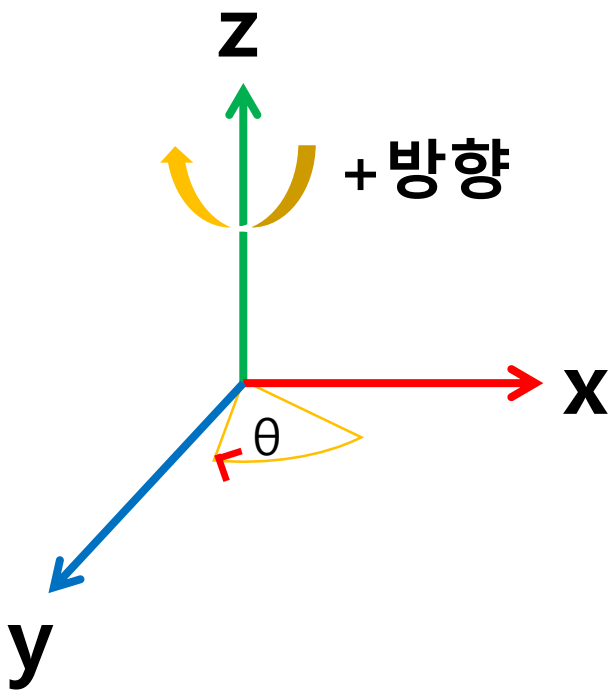
Plane Rotation

평면에서의 회전은 회전 행렬을
사용하여 쉽게 구할 수 있다.

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

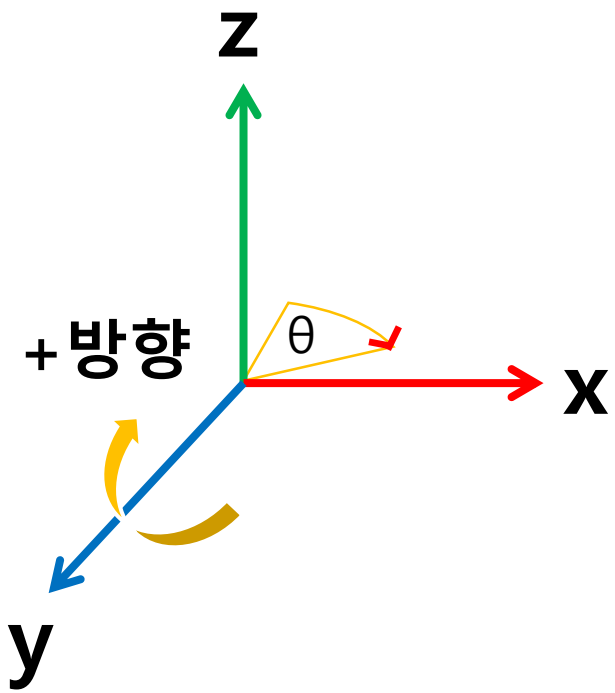
Θ는 반시계 방향으로 회전했을 때의 각도임

Plane Rotation in 3D



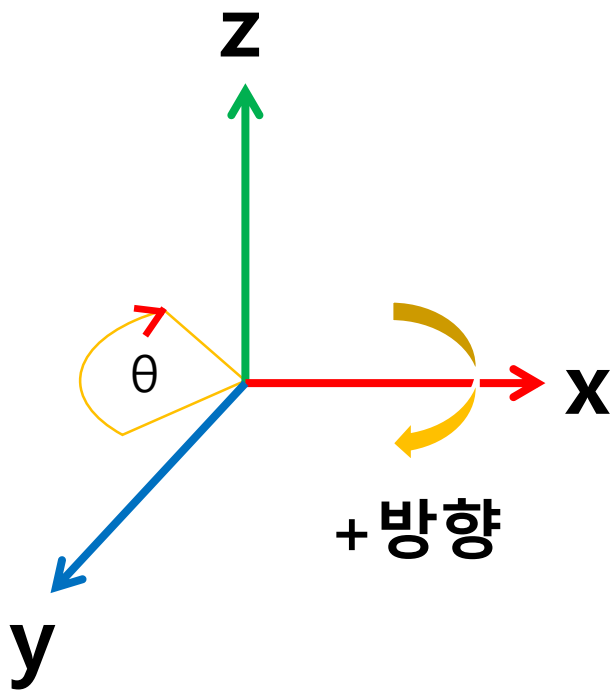
Z축 기준 회전

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



y축 기준 회전

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

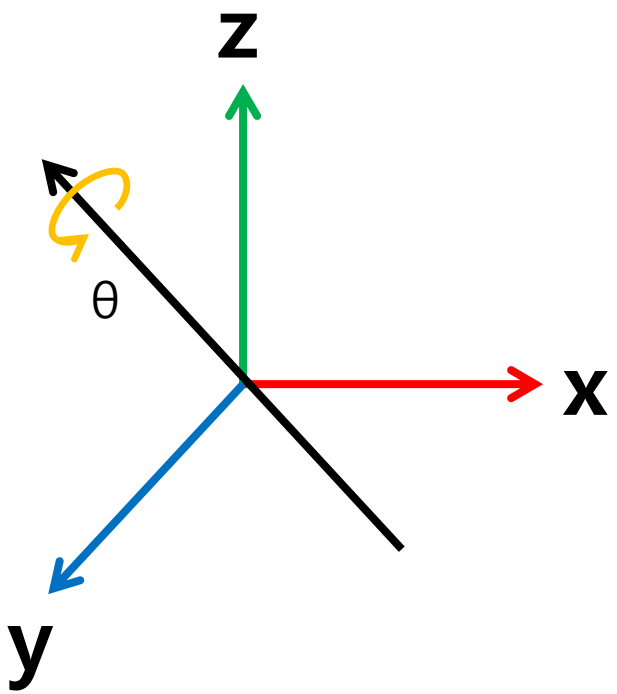


x축 기준 회전

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

3D Space Rotation

노란 축을 기준으로 θ만큼 회전



문제는 주어진 회전축과 회전각으로
어떻게 회전 변환 행렬을 만들어 주느냐는
만만하지 않다는 것이 문제다.

예로 x축을 기준으로 30도
y축을 기준으로 60도 회전 시켰을 경우를 살펴보자!

좌측이 y축에 대한 회전
우측이 x축에 대한 회전이다.

$$\begin{bmatrix} \cos 60 & 0 & \sin 60 \\ 0 & 1 & 0 \\ -\sin 60 & 0 & \cos 60 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 30 & -\sin 30 \\ 0 & \sin 30 & \cos 30 \end{bmatrix}$$

그 둘을 곱해야 최종적으로 원하는 결과를 얻을 수 있다.

Size Transformation

x축을 따라 길이 2
y축을 따라 길이 3인 직사각형을 2배로 키워보자!

현재 넓이는 6

x축을 따라 길이 4
y축을 따라 길이 6인 직사각형을 보자!

즉, 넓이는 24가 됨을 기억하자!

Homogenous Coordinate

여지 것 이동 변환, 회전 변환, 크기 변환 등을 수행했다.
이 모든 것을 한 번에 수행할 수 있는 것이 동차 좌표계다!

$$P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \qquad P = \begin{bmatrix} p_x \\ p_y \\ p_z \\ w \end{bmatrix}$$

위의 P가 점인지 Vector인지 알 방법이 없다.
그래서 Projective Geometry에서
매우 유용하게 쓰이는 w라는 녀석을 하나 가져와보자!

w는 우선 간단하게 일종의 균일한 크기 변환의 역할을 함
동차 좌표계로 P와 Q를 표현해보자!

$$P = \begin{bmatrix} p_x \\ p_y \\ p_z \\ w \end{bmatrix} \qquad Q = \begin{bmatrix} q_x \\ q_y \\ q_z \\ w \end{bmatrix}$$

위의 점 P의 실제 좌표는 아래와 같다.

$$P = \begin{bmatrix} p_x/w \\ p_y/w \\ p_z/w \\ 1 \end{bmatrix}$$

일반적으로 w = 1로 놓음

앞서 점에 대한 동차 좌표계를 알아봤으니 Vector로 해보자!

$$v = Q - P = \begin{bmatrix} q_x \\ q_y \\ q_z \\ w \end{bmatrix} - \begin{bmatrix} p_x \\ p_y \\ p_z \\ w \end{bmatrix} = \begin{bmatrix} q_x - p_x \\ q_y - p_y \\ q_z - p_z \\ 0 \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix}$$

일반적으로 점의 경우는 w가 1이며
Vector의 경우는 w가 0이다.

Transformation with Homogenous Coordinate

점 P에 변위 d를 더해 P′으로 옮길 경우를 생각해보자!

$$P' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad d = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

결과적으로 아래와 같은 형태가 된다.

$x' = x + d_x$, $y' = y + d_y$, $z' = z + d_z$

회전의 경우 3 by 3 행렬을 사용했지만,
동차 좌표계가 4 by 1 Vector이므로
이동 변환 행렬이 4 by 4가 되어야 할 것이다.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow P' = TP$$

여기서 T는 아래와 같고, 이를 Translation Matrix라 부른다.

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

동차 좌표계에서 회전을 생각해보자!

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

xy 평면에서 z축

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

xz 평면에서 y축

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

yz 평면에서 x축

동차 좌표계에서 크기 변환을 생각해보자!

x′ = ax, y′ = by, z′ = cz의 형태로 존재할 때
위의 형태를 동차 좌표 형태로 나타내면 아래와 같다.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 & d_x \\ 0 & b & 0 & d_y \\ 0 & 0 & c & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow P' = SP$$

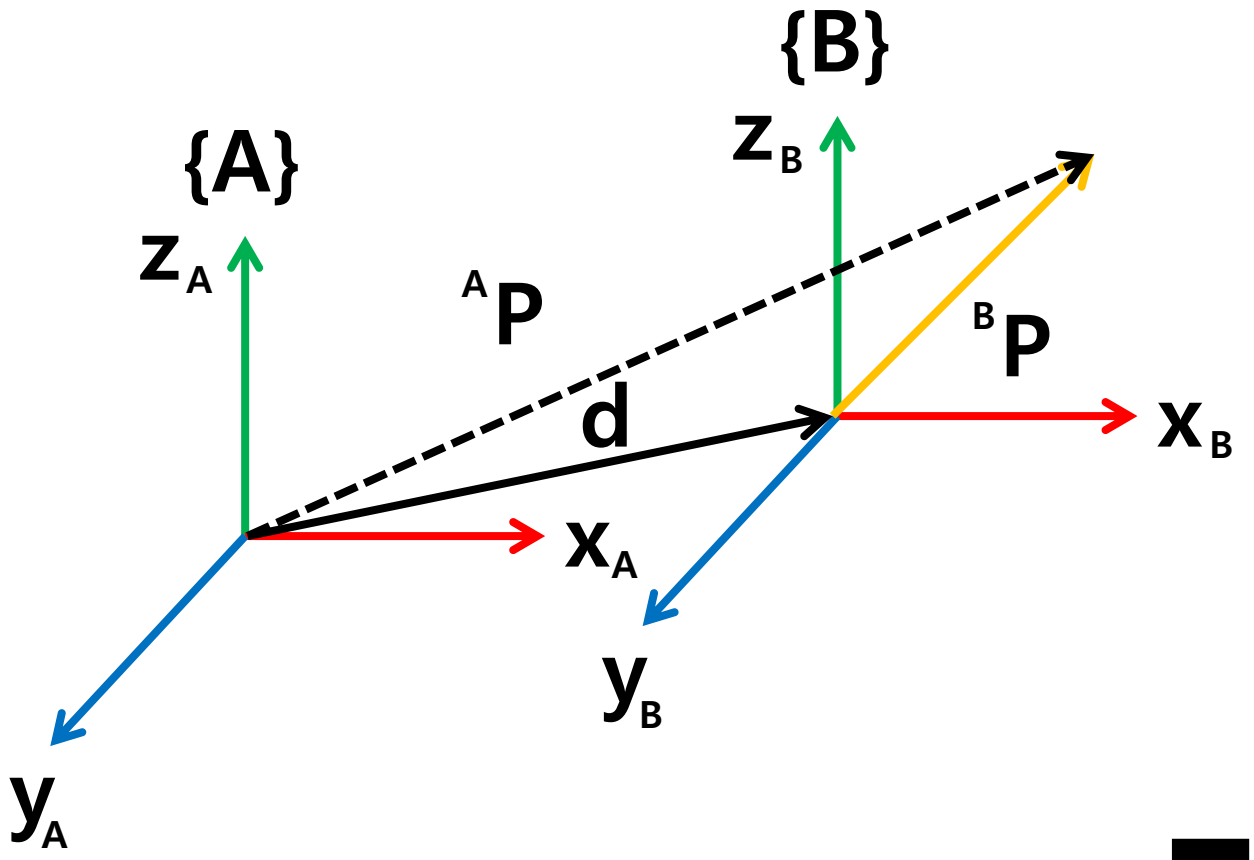
여기서 S는 Scaling Matrix라고 한다.

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

Coordinate Conversion

한 공간을 기준으로 여러 사물을 표현하기 보다
여러 좌표계를 두고 각 사물들을
관리하는 것이 보다 편리하다.

최종적으로 Rendering될 때 각각의 좌표계들은
Base 좌표계로 모두 변환된다.



{A}, {B}는 좌표계를 나타내고,
d는 다음과 같이 나타낼 수 있다.

$$d = \begin{bmatrix} d_x \\ d_y \\ d_z \\ 0 \end{bmatrix}$$

좌표계 {A}로부터 d만큼 이동 변환된 좌표계 {B}에서
표현된 점 ${}^{A}P$가 있을 때, 이 점을 좌표계 {A}에서 표현하면

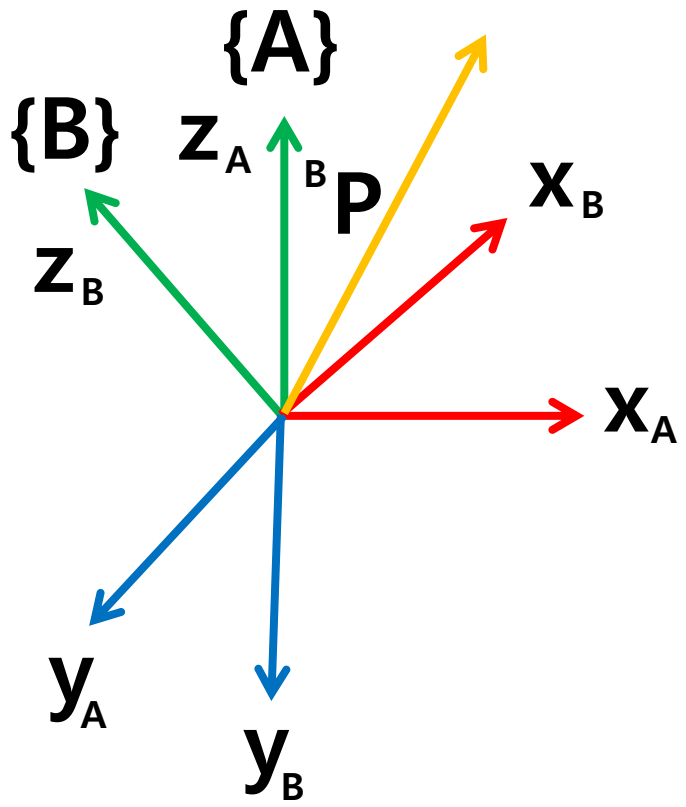
$$ {}^{A}P = {}^{B}P + d $$

즉, {B}좌표계에서 표현된 한점 P를 통째로 d만큼
이동 변환했음을 의미하므로 점의 이동과 마찬가지로
이동 변환 행렬 또한 아래와 같다.

$$ {}^{A}P = T + {}^{B}P $$

T는 아래와 같음

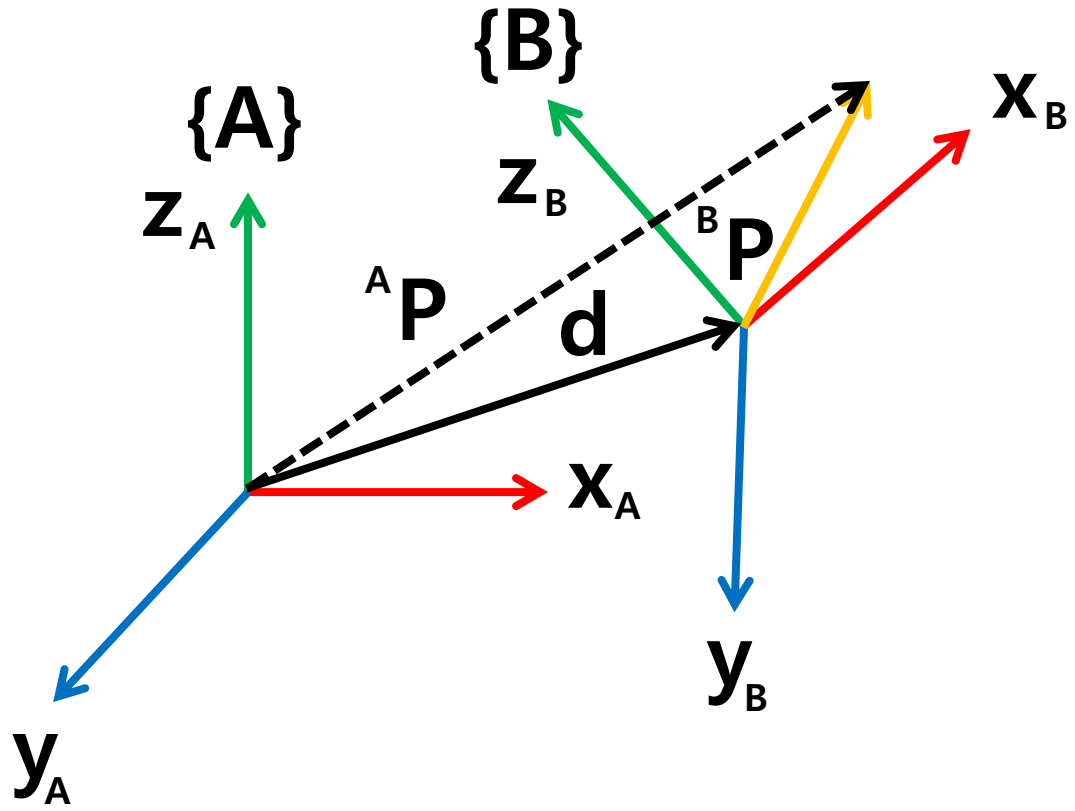
$$ \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



이번에는 {A}로부터 회전 변환 행렬
R만큼 회전 변환된 좌표계 {B}에서
표현된 점 P가 있을 때,
이 점을 좌표계 {A}에서 표현하면

$$^AP = R\,^BP$$

즉, 좌표계 {B}에서 표현된 한 점 ^BP를
통째로 R만큼 회전 변환한 후
좌표계 {A}에서 표현했다는 의미이다.



좌표계 {B}를 회전 행렬 R로 회전 변환을 수행 후,
d만큼 이동 변환한 것으로, 결과적으로 좌표계 {B}에서
표현된 점 ^{B}P를 좌표계 {A}에서 표현한 ^{A}P로 변환한 것이다.

위에서 봤던 변환을 식으로 표현하면 아래와 같다.

$$ {}^{A}P = R\,{}^{B}P + d $$

이를 동차 좌표계로 표현하면 아래와 같다.

$$ \begin{bmatrix} {}^{A}P_x \\ {}^{A}P_y \\ {}^{A}P_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}P_x \\ {}^{B}P_y \\ {}^{B}P_z \\ 1 \end{bmatrix} $$

빨간 박스는 회전 변환 부분이며
파랑 박스는 이동 변환 부분으로
회전과 이동을 모두 한 번에 다룰 수 있다.

OpenGL의 경우 4 by 4 행렬의 동차 변환을 사용!