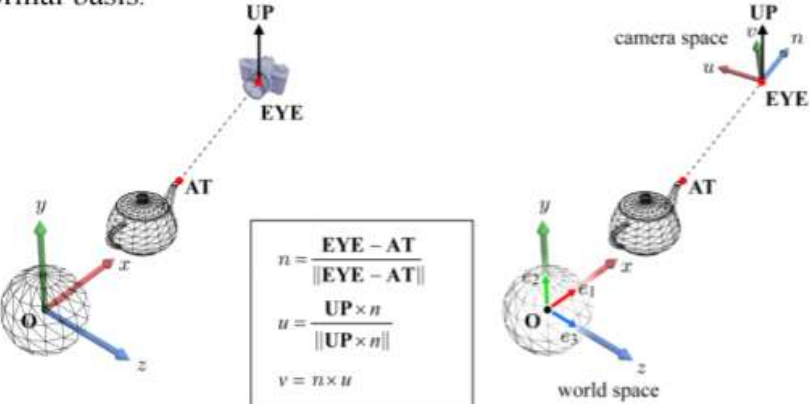
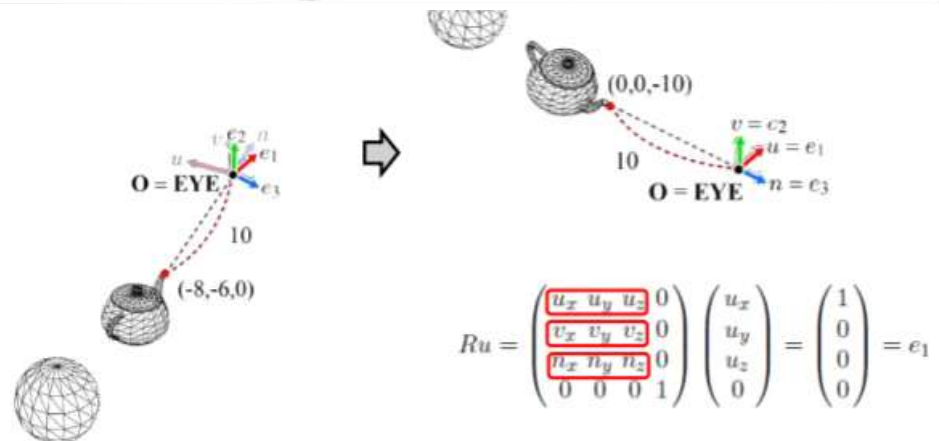
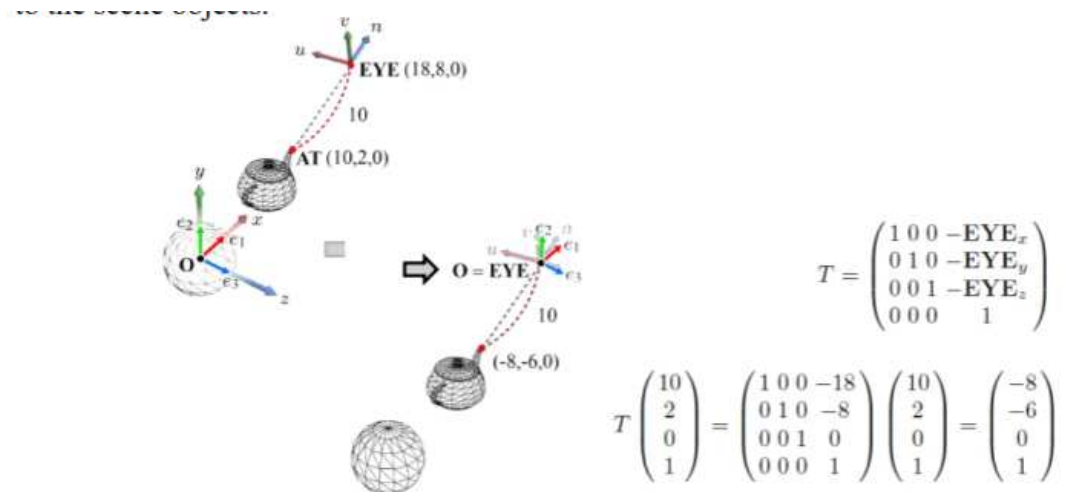


필기노트 - 컴퓨터 그래픽스 5장 정점 처리

	내 용
GPU Rendering Pipeline	<ul style="list-style-type: none"> ● vertex shader → rasterizer → fragment shader → output merger ● 이렇게 물 흐르듯이 순서대로 이루어진다고 Pipeline architecture라고 함
World Transform -법선벡터	<ul style="list-style-type: none"> ● 4장에서 다뤘던 변환 과정에서 vertex normal(법선벡터)는 어떻게 되는가? ● effin transform에서 우리는 $[L t] \cdot n$ 로 $Ln + t$인데 n이 벡터이므로 translation을 했을 때 normal은 변하지 않음 ● Vertex normal의 선형변환은 L이 아닌 $(L^{-1})^T = L^{-T}$ 을 Normal에 곱해준다.
Camera Space 카메라 좌표계	<p>Camera pose (position + orientation) specification in the world space</p> <ul style="list-style-type: none"> ▪ EYE: camera position ▪ AT: a reference point toward which the camera is aimed ▪ UP: view up vector that describes where the top of the camera is pointing. (In most cases, UP is set to the vertical axis, y-axis, of the world space.) <p>The camera space, $\{u, v, n, \text{EYE}\}$, can be created. Note that $\{u, v, n\}$ is an orthonormal basis.</p> <div style="text-align: center;">  <div style="border: 1px solid black; padding: 10px; width: fit-content; margin: 0 auto;"> $n = \frac{\text{EYE} - \text{AT}}{\ \text{EYE} - \text{AT}\ }$ $u = \frac{\text{UP} \times n}{\ \text{UP} \times n\ }$ $v = n \times u$ </div> </div> <ul style="list-style-type: none"> ● EYE : World space에서 카메라가 있는 위치 ● AT : 카메라가 초점을 맞출 위치 ● UP : 카메라의 수직으로 위를 향하는 벡터 ● n = AT에서부터 EYE로 향하는 단위벡터 ● u = UP에서 n의 외적 (UP부터 n까지 오른손 법칙) ● v = n에서 u의 외적 ● n, u, v는 각각 서로 orthonormal한 관계를 가지고 있음 <ul style="list-style-type: none"> - 외적의 결과물은 사용된 벡터와 항상 직교함 ● 결국 EYE를 원점으로 봤을 때 n, u, v가 3차원 좌표계를 형성 이 좌표계를 camera space라고 함
View Transform	<ul style="list-style-type: none"> ● world space → camera space 넘어가는 과정을 View Transform이라고 함 ● view transform에서는 월드공간좌표 → 카메라 공간 좌표로 바뀌어줘야 함.

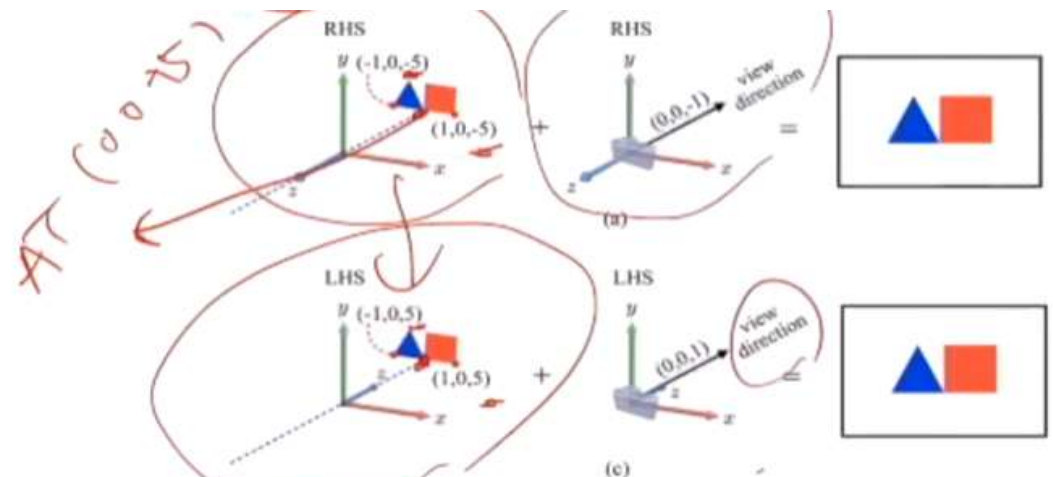


바꾸는 순서

1. camera space의 원점을 world space의 원점으로 이동시킴
2. $v = e_2$, $u = e_1$, $n = e_3$ 로 rotation으로 돌려줌

Left-hand system
왼손좌표계

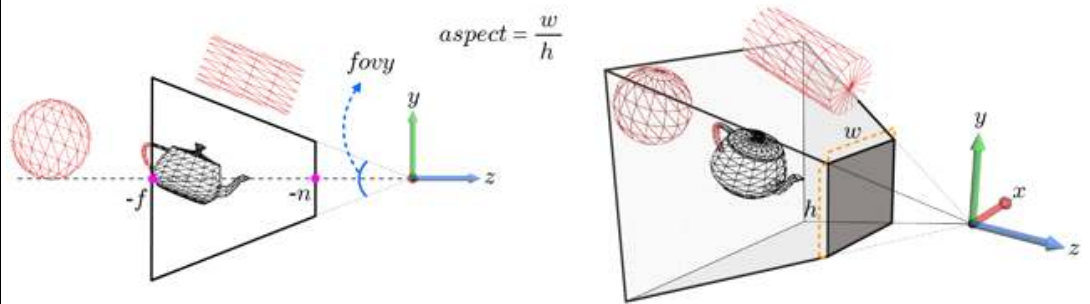
- 오른손 좌표계에서는 $x \rightarrow y$ 로 회전할 때 엄지가 z 축
- 이걸 왼손에서 적용한 것이 왼손좌표계
- Direct3D는 왼손좌표계를 사용함. OpenGL은 오른손좌표계 사용
- 좌표를 그대로 했을 때 오른손좌표계에서 만든 것을 왼손 좌표계로 넘길 때 좌우가 반전됨



그걸 해결하기 위해서는 Z좌표의 부호를 바꿔주면 된다.

View Frustum

- 외부 파라미터로 EYE, AT, UP가 있다
- 내부 파라미터로는 zoom-in, zoom-out, fovy(field of view y-axis), aspect - aspect = width/height

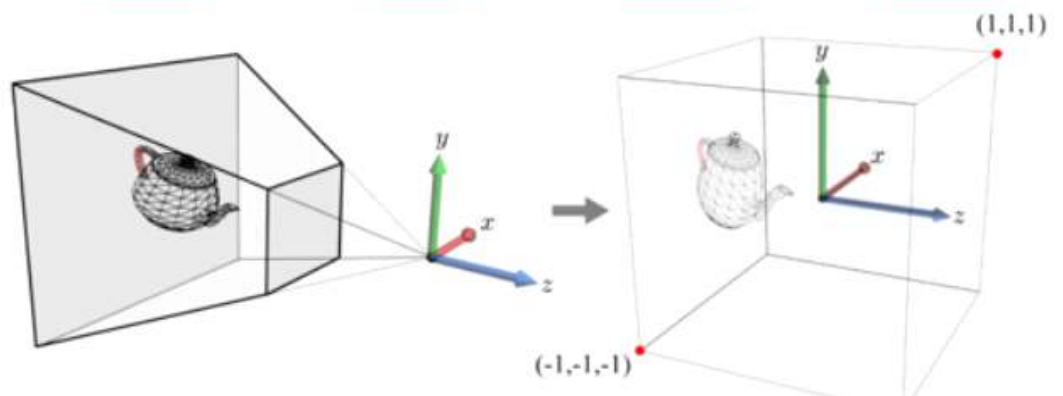
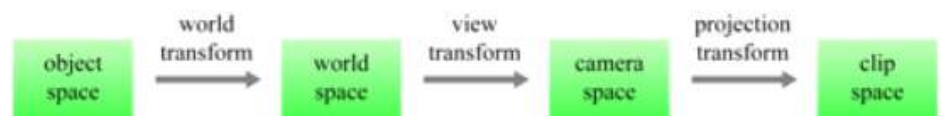


머리잡린 피라미드처럼 생긴 것이 View Frustum이고
View Frustum 안쪽에 있는 것만 시야에 들어오는 것

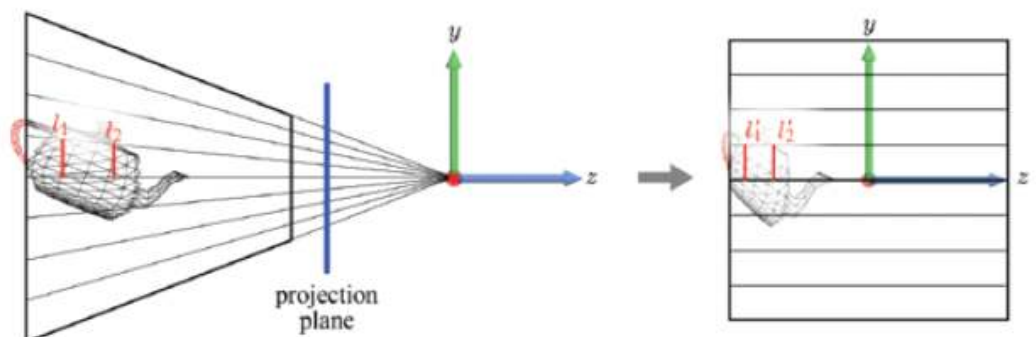
주전자에서 일부분이 View Frustum에서 벗어났을 경우에
그 부분을 제외하는 것을 clipping이라고 한다.

그런데 절두체(frustum)에서 그것을 수행하기 어려우므로
View Frustum을 정육면체로 변환하는 과정이 필요하다 ↓

- 클리핑을 위해 만들어진 공간이라 하여 이 정육면체 공간을 clip space라고 한다.



Projection transform 투영 변환



l1이 l2보다 길지만 원근법에 의해 보이는 길이는 같음

→ 투영변환 공식

$$\begin{pmatrix} \cot(\frac{fovy}{2}) & 0 & 0 & 0 \\ 0 & \cot(\frac{fovy}{2}) & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & \frac{2nf}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Vertex shader
→ rasterizer

- rasterizer는 왼손 좌표계 데이터만 입력 받을 수 있기 때문에 소프트웨어인 shader가 변경해서 넣어줘야 함
 - z좌표를 모두 바꾸어주면 된다.