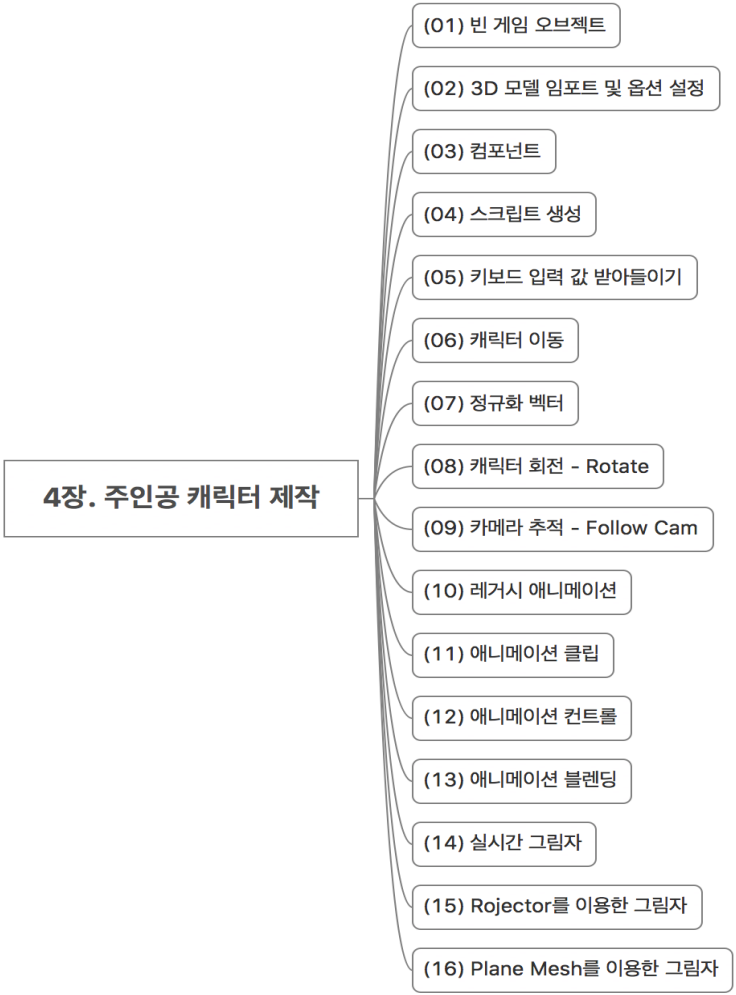
4일_유니티 좌표계 공부

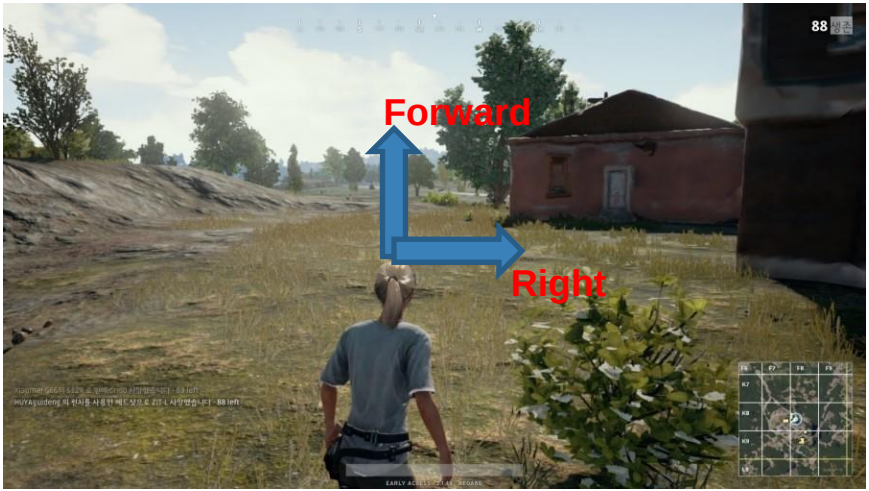
이준

학습 내용

- 주인공 캐릭터 제작 및
- 유니티 코딩 시작



캐릭터 이동 - 게임오브젝트의 이동



캐릭터 이동 - Translate() 이동 함수

- **이동 함수 원형**
  - void Translate(Vector3 direction, [Space relative]);
  - tr.Tanslate(Vector3.forwad);
    - Vector3.forward : Vector3(0, 0, 1)과 같음

- **이동 함수 처리**
  - tr.Translate(Vector3.forward*moveSpeed*v*Time.deltaTime, Space.Self);
  - 이동할 방향 * 속도 * 전진, 후진 변수(방향) * 시간
  - Time.deltaTime : 이전 프레임에서 현재 프레임가지 걸린 시간
    - Frame Rate에 상관 없이 지정한 속도로 이동 가능
    - Frame Rate가 다른 기기에서도 같은 속도로 이동할 수 있음

정규화 벡터

- 벡터는 크기와 방향을 가진 데이타 타입으로, 그 중에서 각 축의 크기가 1인 벡터를 단위 벡터(정규화된 벡터, Normalized Vector)라 한다. 즉, 방향만 표시하는 벡터

Vector3.forward	Vector3(0, 0, 1)
Vector3.back	Vector3(0, 0, -1)
Vector3.left	Vector3(-1, 0, 0)
Vector3.right	Vector3(1, 0, 0)
Vector3.up	Vector3(0, 1, 0)
Vector3.down	Vector3(0, -1, 0)
Vector3.one	Vector3(1, 1, 1)
Vector3.zero	Vector3(0, 0, 0)

- 유니티는 왼손 좌표계를 사용함 (DirectX와 동일)

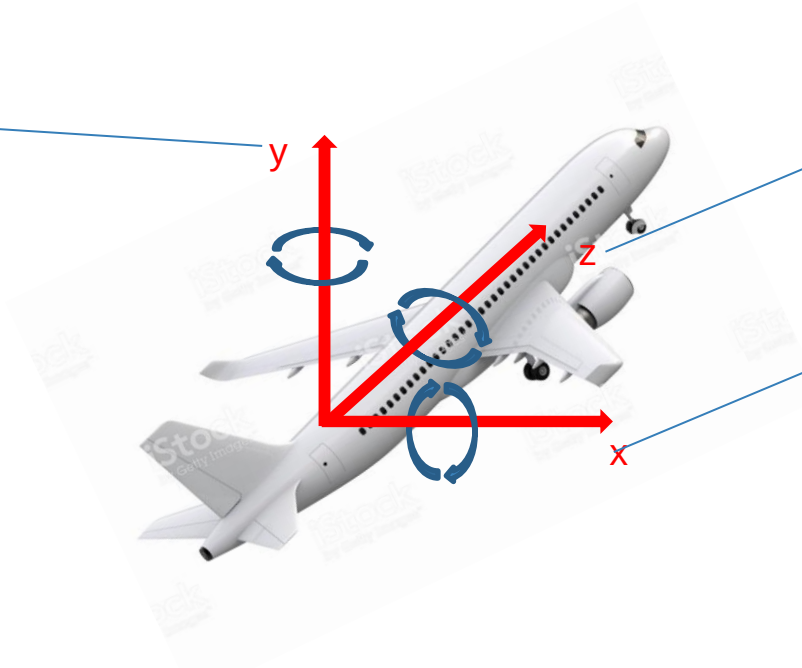
왼손 좌표계

- 공간 좌표에서 Z 축을 결정하기 위한 방법
- 유니티는 왼손 좌표계를 사용함 (DirectX와 동일)

캐릭터 회전 – Rotate() 이동함수

- 공간 좌표에서 Z 축을 결정하기 위한 방법
- 유니티는 왼손 좌표계를 사용함 (DirectX와 동일)

yaw :
y축을 중심으로 θ도 만큼
회전



Roll :
z축을 중심으로
ψ도 만큼 회전

Pitch :
X축을 중심으로
Φ 도 만큼 회전

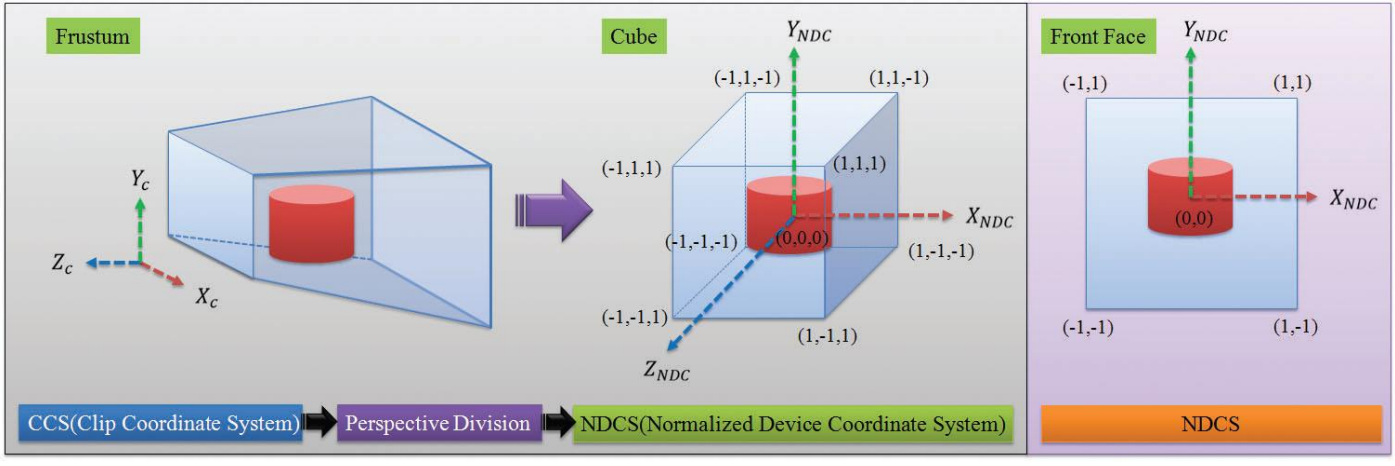
캐릭터 회전 – Rotate() 이동함수

- **오일러 앵글?**
  - 레온하르트 오일러라는 수학자가 1700년 경에 도입한 개념
  - 현대 기하학에서 활발하게 사용 됨

https://www.wolfram.com/language/11/core-geometry/roll-pitch-and-yaw-rotations.ko.html?product=language

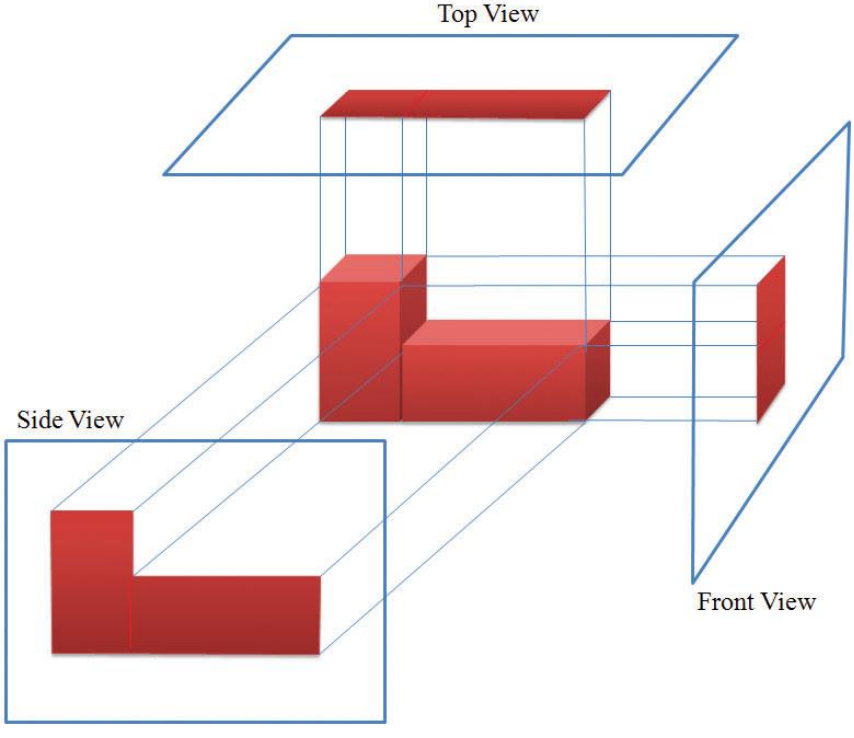
좌표 변환 Pipeline

NDCS(Normalized Device Coordinate Syste



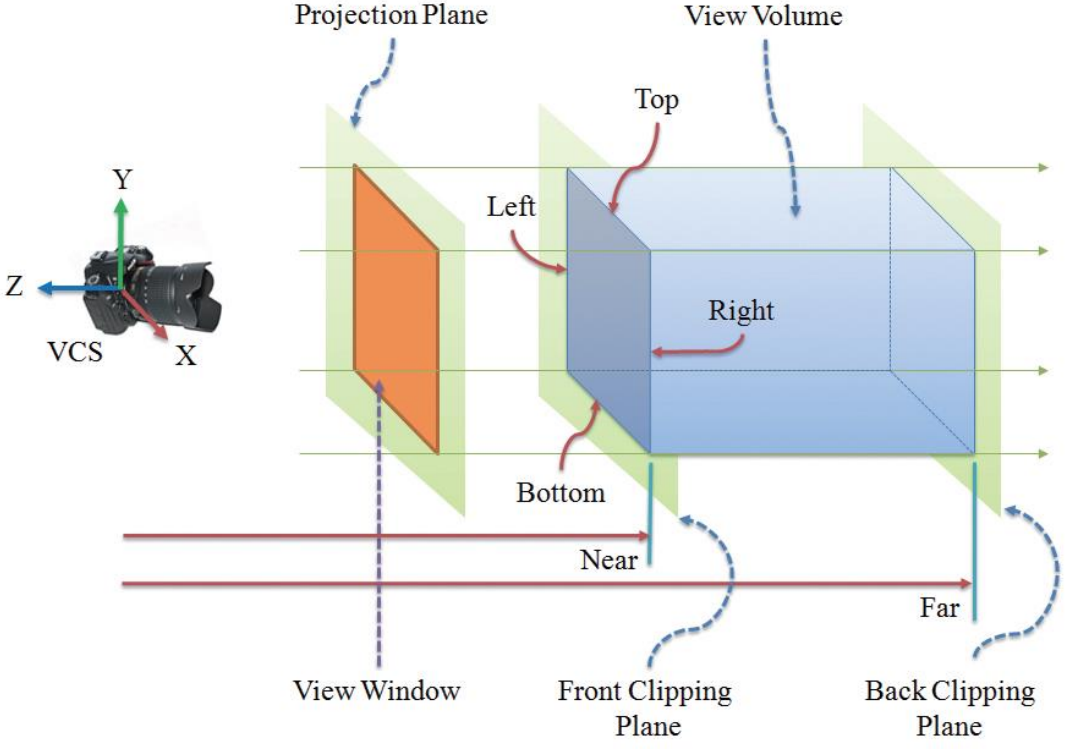
정사 투영

정사 투영의 3가지 종류



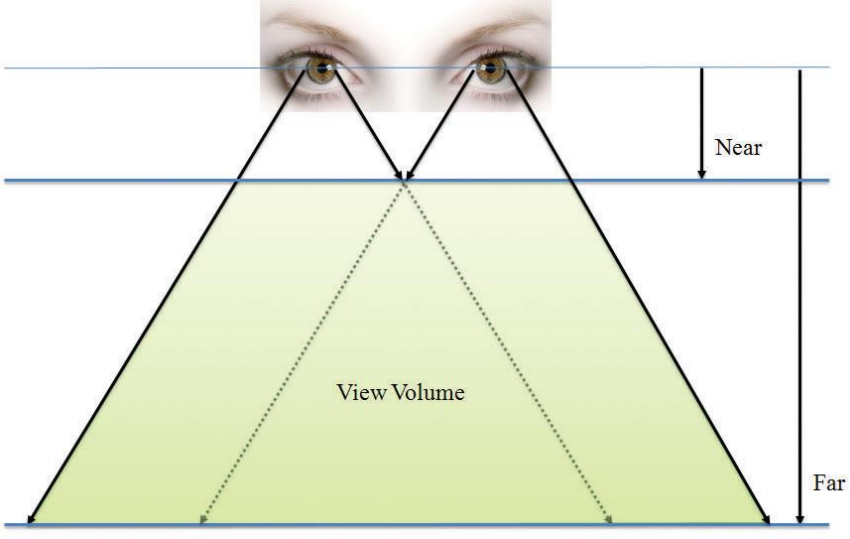
정사 투영

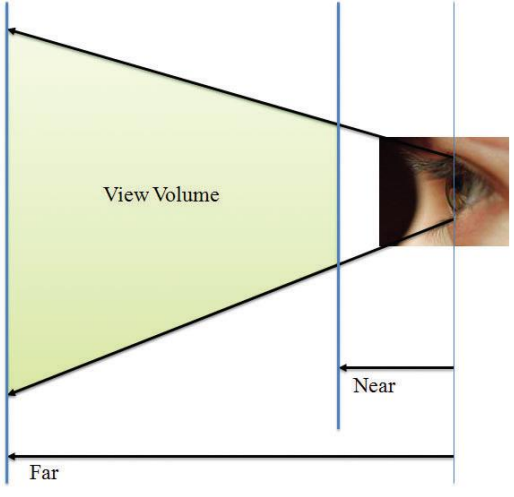
glOrtho 함수에 대한 구조



정사 투영

- 사람의 눈에 대한 두 개 시야의 각(Angle)





정사 투영

glOrtho 함수에 대한 Prototype

void glOrtho (GLdouble *left*, GLdouble *right*, GLdouble *bottom*, GLdouble *top*,
GLdouble *near*, GLdouble *far*);

Parameters Help	*left*	// 왼쪽 수직 절단면을 위한 X축 좌표
	right	// 오른쪽 수직 절단면을 위한 X축 좌표
	bottom	// 아래쪽 수평 절단면을 위한 Y축 좌표
	top	// 위쪽 수평 절단면을 위한 Y축 좌표
	near	// 전방 절단면을 위한 –Z축 방향의 거리(양수)
	far	// 후방 절단면을 위한 –Z축 방향의 거리(양수)

정사 투영

glOrtho Matrix로 표현하는 정사 투영

코드 4-1

glOrtho Matrix로 표현하는 정사 투영

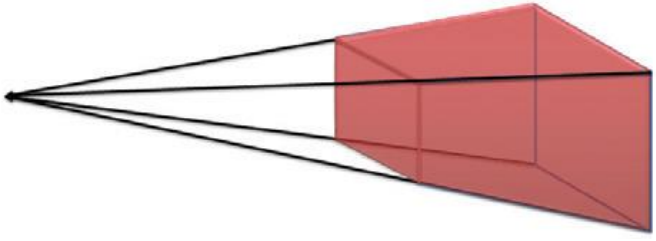
```
GLfloat l =-1.0, r =-1.0, b =-1.0, t =-1.0, n =-1.0, f =-1.0;

GLfloat Ortho_Matrix[4][4] = {
    { 2/(r-l), 0.0, 0.0, -((r+l)/(r-l)) },
    { 0.0, 2/(t-b), 0.0, -((t+b)/(t-b)) },
    { 0.0, 0.0, -2/(f-n), -((f+n)/(f-n)) },
    { 0.0, 0.0, 0.0, 1.0 }
};
...
glMultMatrixf((float*) Ortho_Matrix);
...
```

$$O_p = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\left(\frac{r+l}{r-l}\right) \\ 0 & \frac{2}{t-b} & 0 & -\left(\frac{t+b}{t-b}\right) \\ 0 & 0 & \frac{-2}{f-n} & -\left(\frac{f+n}{f-n}\right) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

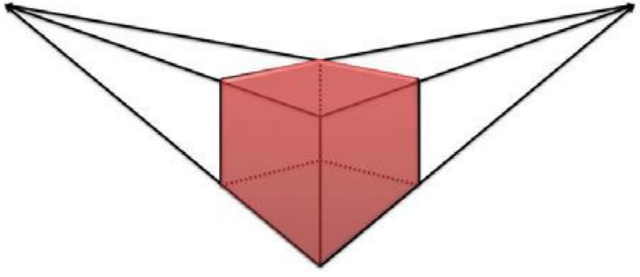
정사 투영(Orthographic Projection) Matrix

원근 투영

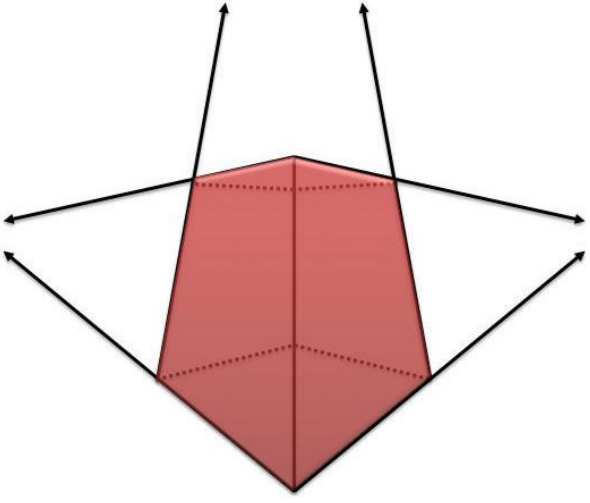
소실점의 개수에 따른 원근 투영 방법



(A) One-Point Projection



(B) Two-Point Projection



(C) Three-Point Projection

원근 투영

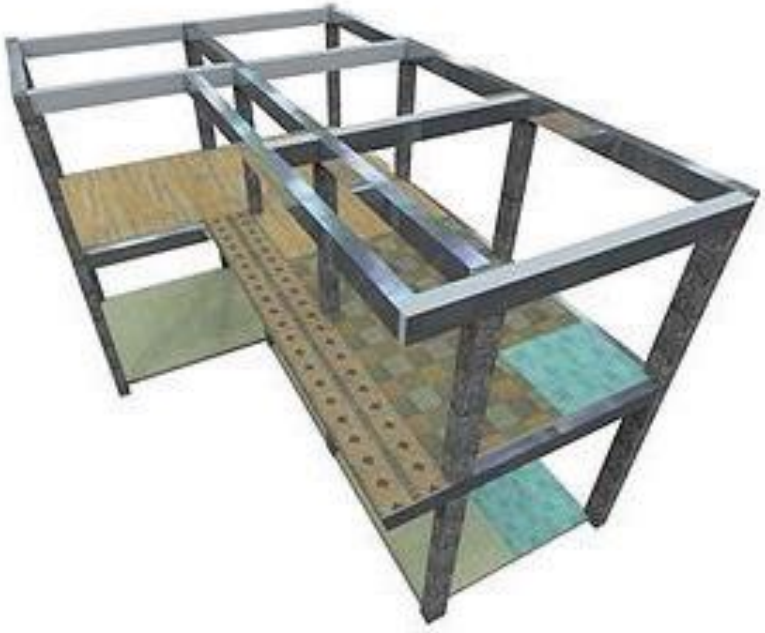
소실점의 개수에 따른 예시 – One

원근 투영

소실점의 개수에 따른 예시 – Two Point

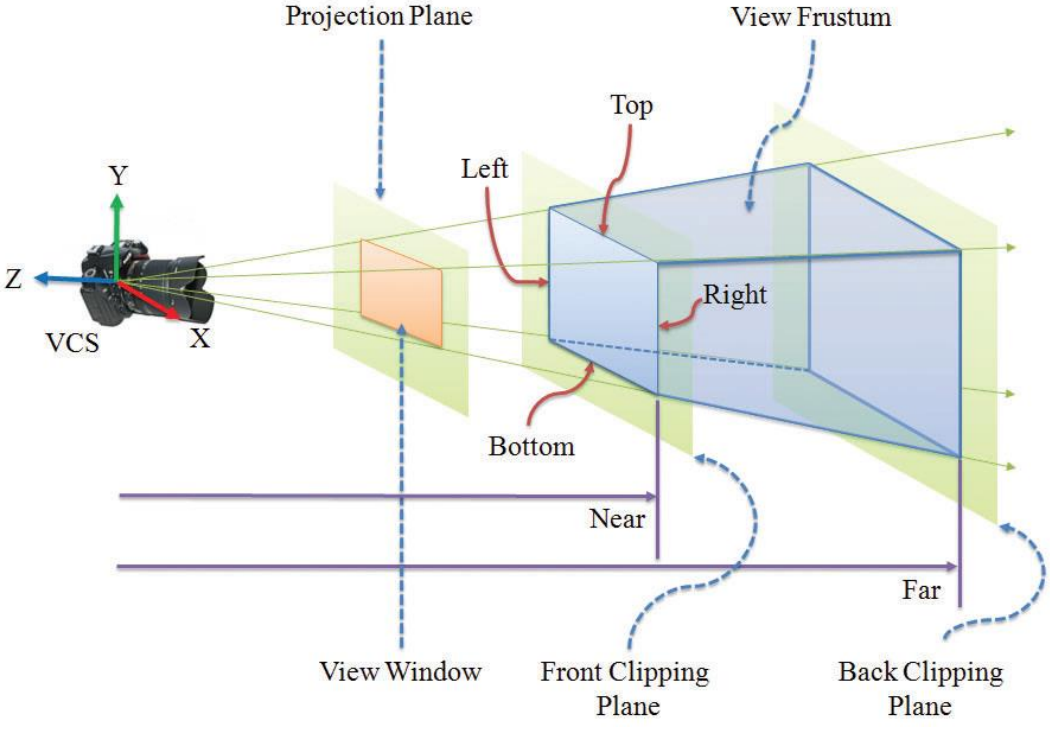
원근 투영

- 소실점의 개수에 따른 예시 – Three Point



원근 투영

카메라 함수의 구조



Thanks!

Any questions?

junlee@game.hoseo.edu