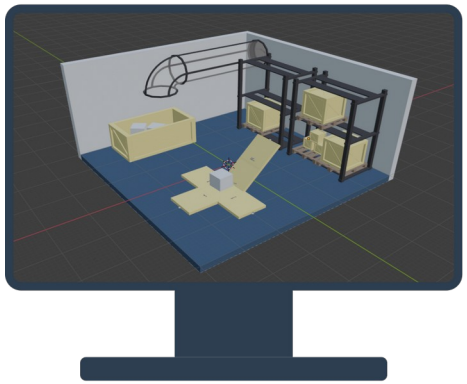
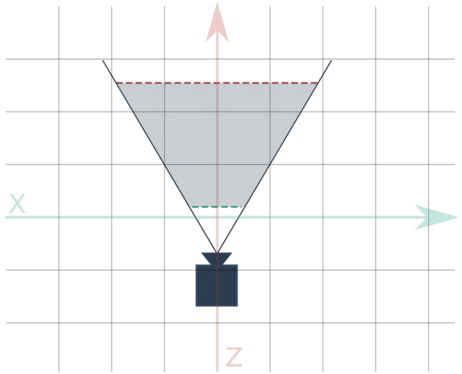
Three.JS University

Camera – Perspective & Orthographic

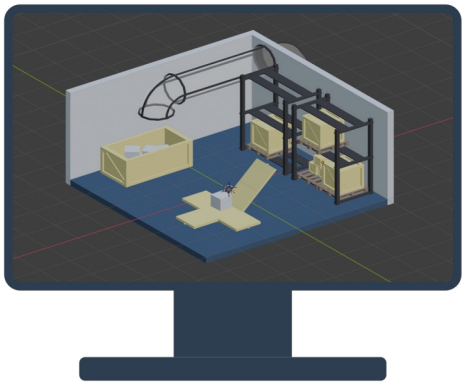
Camera – *Perspective & Orthographic*

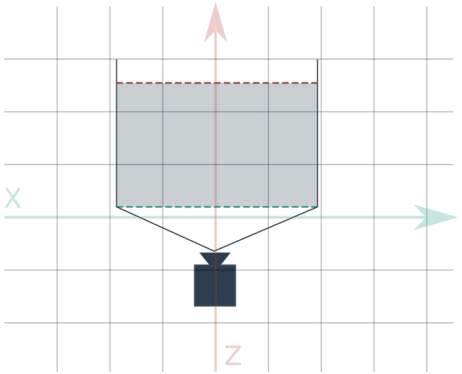
Perspective





Orthographic



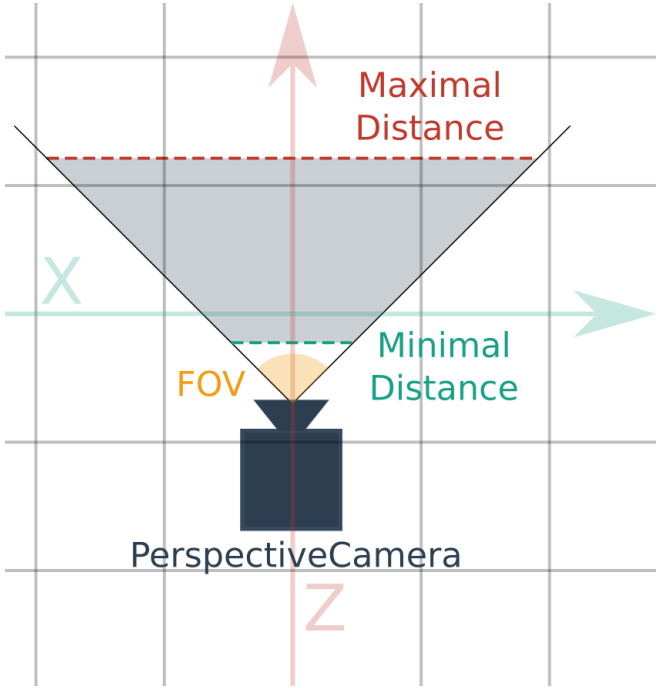


Dans le monde de la 3D, il existe deux principaux mode de projection :

- La projection « *Perspective* »
- La projection « *Orthographic* »

Dans Three.js, les classes **PerspectiveCamera** et **OrthographicCamera** permettent de recréer ces deux modes de projection

PerspectiveCamera



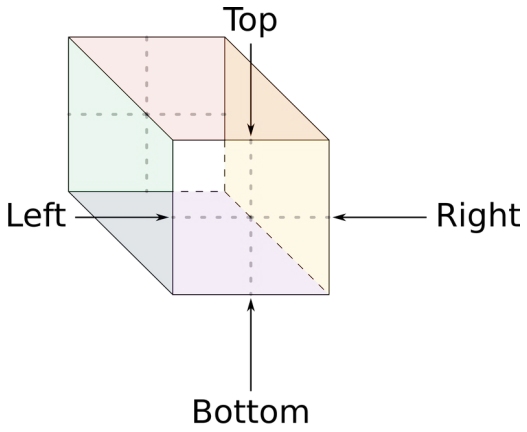
Projection basée sur la perspective, similaire à la vision humaine

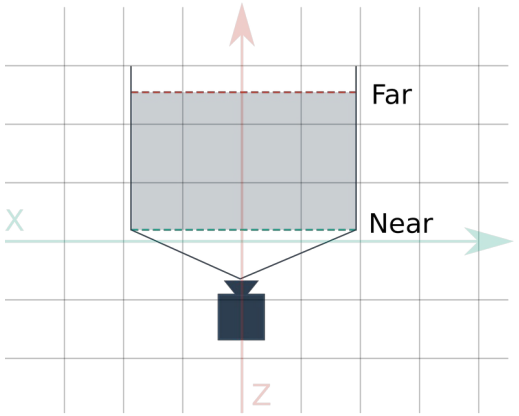
Le constructeur de **PerspectiveCamera** accepte quatre paramètres :

- **FOV** – Field of View (Ouverture de l'objectif)
- **Aspect** – Le ratio de projection
- **Distance minimale** – Distance de vue minimale de la caméra
- **Distance maximale** – Distance de vue maximale de la caméra

```
camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 1, 10000);
```

OrthographicCamera



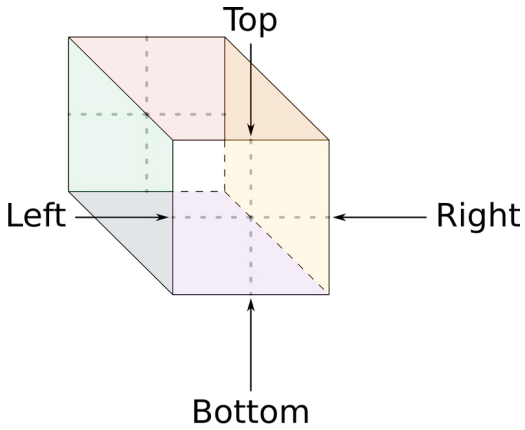


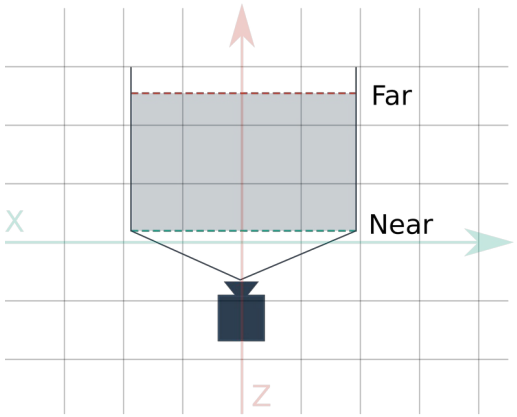
Caméra basée sur les règles d'une projection orthogonale – Ici, la taille d'un objet de dépend pas de sa distance vis à vis de la caméra

Le constructeur de **OrthographicCamera** accepte six paramètres :

- **Left** – Limite gauche du tronc de vue
- **Right** – Limite droite du tronc de vue
- **Top** – Limite haut du tronc de vue
- **Bottom** – Limite bas du tronc de vue
- **Near** – Distance de vue minimale de la caméra
- **Far** – Distance de vue maximale de la caméra

OrthographicCamera





Le constructeur de **OrthographicCamera** accepte six paramètres :

```
const aspect = window.innerWidth / window.innerHeight;
const frustumSize = 10;

camera = new THREE.OrthographicCamera(
frustumSize*aspect/-2,
frustumSize*aspect/2,
frustumSize/2,
frustumSize/-2,
1,
1000 );
```

Camera – Perspective & Orthographic

Si vous êtes prêts, passons à la pratique !