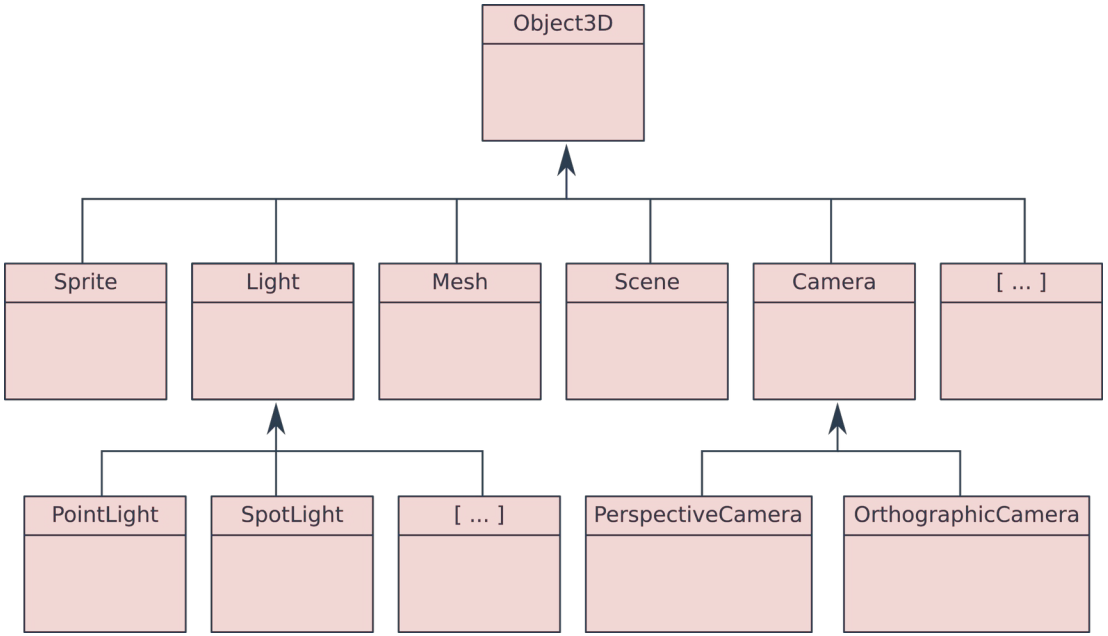
Three.JS University

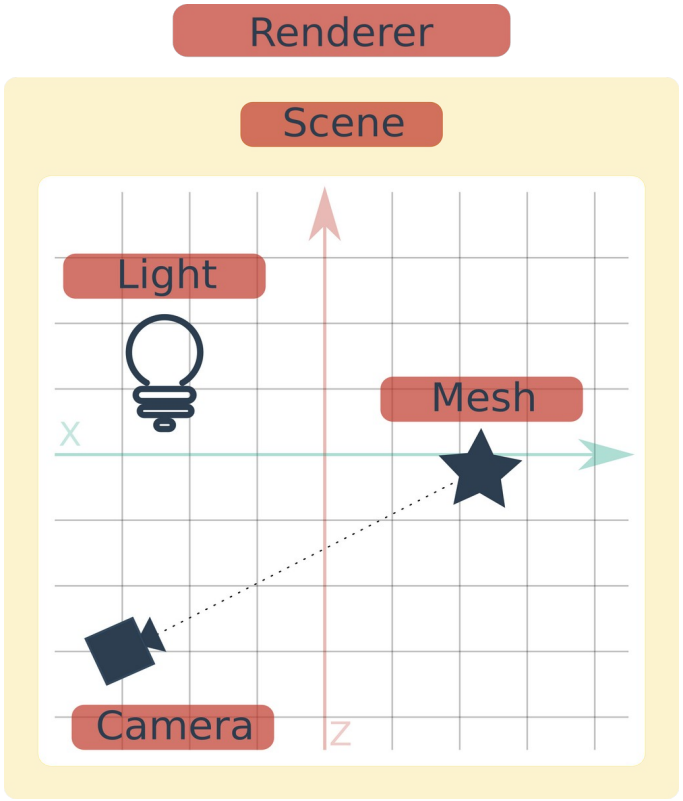
Object3D – Les bases

La classe Object3D



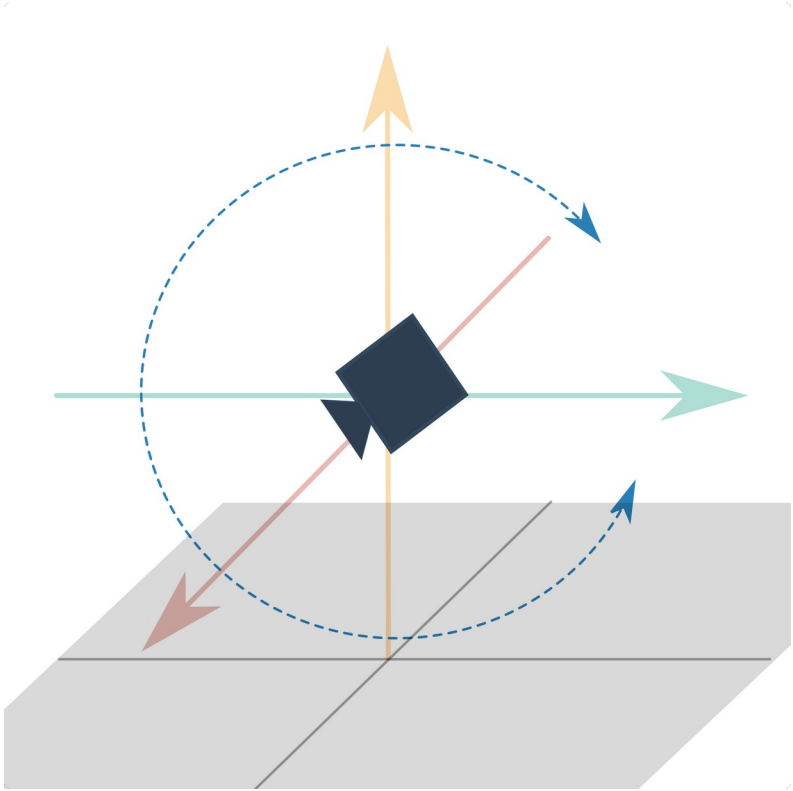
Object3D représente la classe de base de la plupart des objets de Three.js

La classe Object3D



- Les classes **Mesh, PerspectiveCamera, Scene, DirectionalLight** et **AmbientLight** utilisées dans les premiers chapitres sont liées à **Object3D** !
- **Object3D** fournit un ensemble de propriétés et de méthodes pour manipuler des objets dans un espace 3D

Object3D – id et *name*



- Chaque instance de classe **Object3D** possède un identifiant unique, attribué lors de la création de l'objet
- Cet identifiant est stocké dans la propriété de classe **id (Integer)**
- **id** est une valeur en lecture seule

```
console.log(cube);
```

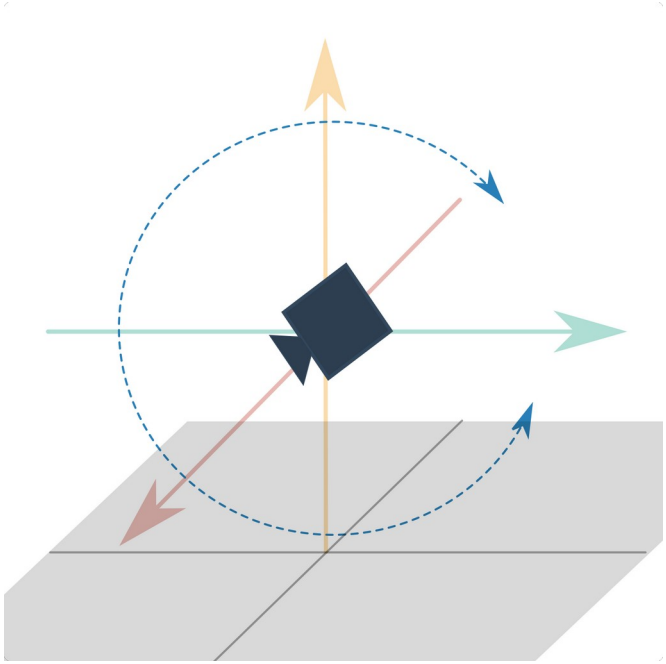
```
visible: true
id: 81
modelViewMatrix
```

- Il est également possible d'identifier un objet grâce à la propriété **name** (**String**) , facultative et librement modifiable

```
cube.name = "My Cube";
console.log(cube);
```

```
matrixWorldNeedsU
name: "My Cube"
parent: Scene {uu
```

Object3D – position, rotation et *scale*



Object3D met à disposition trois propriétés de classe :

- **position** – Position de l'objet sur les axes **X**,**Y** et **Z**
- **rotation** – Rotation de l'objet sur les axes **X**,**Y** et **Z**
- **scale** – Échelle de l'objet sur les axes **X**,**Y** et **Z**

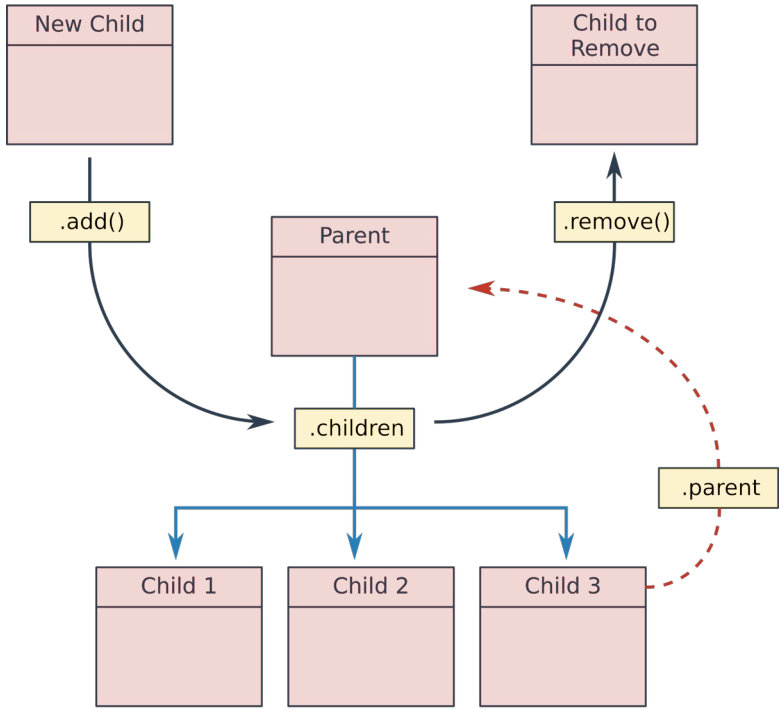
La méthode **set** permet de définir la valeur souhaitée sur les axes **X**,**Y** et **Z** :

```
cube.position.set(3, -4, 10);
cube.rotation.set(0, Math.PI/8, 0);
cube.scale.set(2, 2, 2);
```

Il est également possible d'utiliser les propriétés **x**, **y** ou **z** :

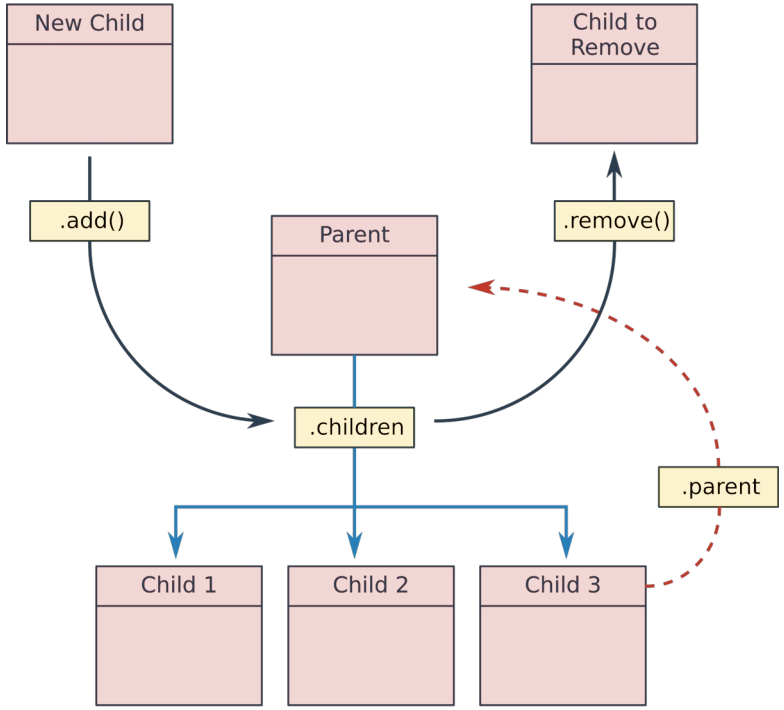
```
cube.position.z = 2;    cube.rotation.x = Math.PI/12;    cube.scale.y = 2;
cube.position.x = -4;   cube.rotation.y = Math.PI/2;     cube.scale.z = 2;
```

Object3D – parent et *children*



- Three.js propose un système de relation parent/enfant entre les éléments
- Chaque instance de classe **Object3D** dispose d'une propriété **children** – Il est ainsi possible de créer une arborescence d'éléments **Object3D**
- Cette propriété est utilisée pour stocker les enfants d'un objet
- Les méthodes **add** et **remove** permettent d'ajouter et retirer des éléments de la liste **children** d'un objet
- Chaque instance de classe **Object3D** dispose d'une propriété **parent** représentant l'instance de son parent dans l'arborescence

Object3D – parent et *children*



Les méthodes **add** et **remove** acceptent un paramètre : l'instance de **Object3D** à ajouter ou supprimer

```
var cube = new THREE.Object3D();
cube.add( child_1 );
cube.add( child_2 );
console.log(cube.children);
```

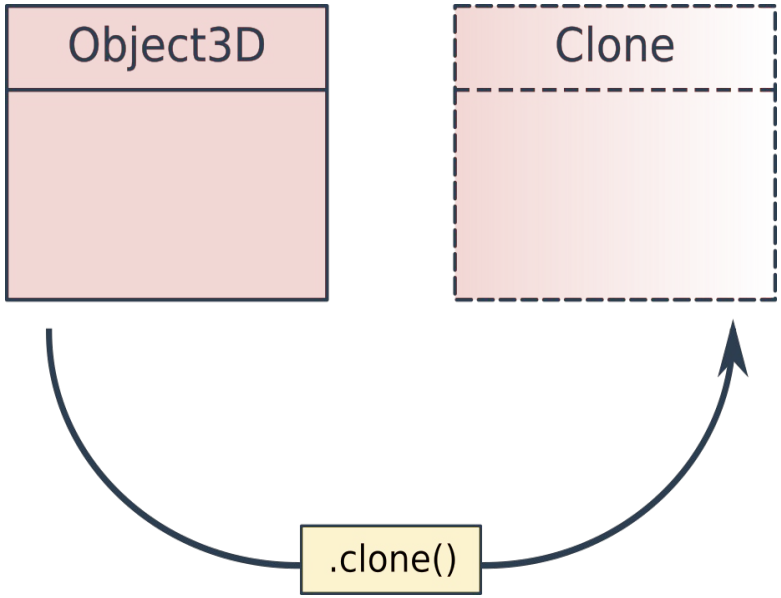
```
▼ (2) [Group, Group]
  ▶ 0: Group {uuid: 'BA59FC46-4808-4843-9E3A-8
  ▶ 1: Group {uuid: '741638BB-42AB-4403-854C-A
    length: 2
  ▶ [[Prototype]]: Array(0)
```

```
cube.remove( child_1 );
```

```
console.log( child_2.parent );
```

```
Object3D {uuid: '79EBC9B6-74D4-4901-8
▼ e: '', type: 'Object3D', parent: null
  (1), …}
  ▶ animations: []
    castShadow: false
  ▶ children: [Group]
```

Object3D – clone



La méthode **clone** de classe **Object3D** est utilisée pour dupliquer un élément

Cette méthode accepte un paramètre facultatif :

- **recursive** : **Booléen** – Valeur **false** si nous ne souhaitons pas inclure l'arborescence des éléments **children** dans le clone (**true** par défaut)

```
var cube = new THREE.Object3D();
var newcube = cube.clone();
```

Object3D – Les bases

Si vous êtes prêts, passons à la pratique !