Three.JS
University

Les matériaux Three.js

Modifions l’aspect visuel de nos objets 3D !

Il existe six types principaux de matériaux Three.js :

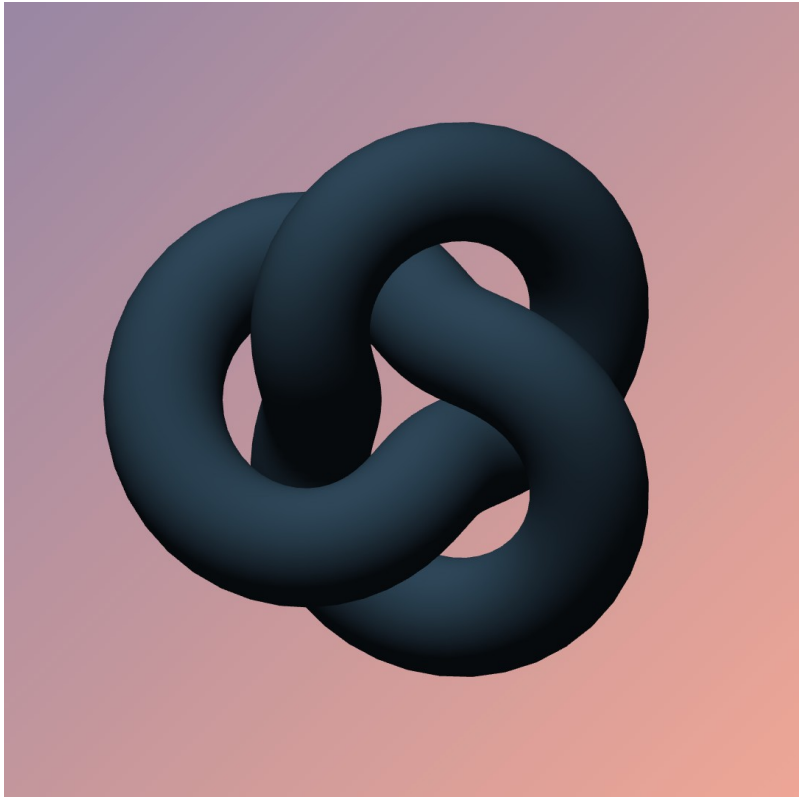
- MeshBasicMaterial
- MeshLambertMaterial
- MeshPhongMaterial
- MeshToonMaterial
- MeshStandardMaterial
- MeshPhysicalMaterial

MeshBasicMaterial

- La classe **MeshBasicMaterial** est utilisée pour dessiner un **Mesh** sans prise en compte de l'éclairage
- Rendu ultra-basique sans aucun *shading*
- *Prise en compte de la couleur*

```
const material_basic = new THREE.MeshBasicMaterial(
{
  color : 0x2c3e50
});
```

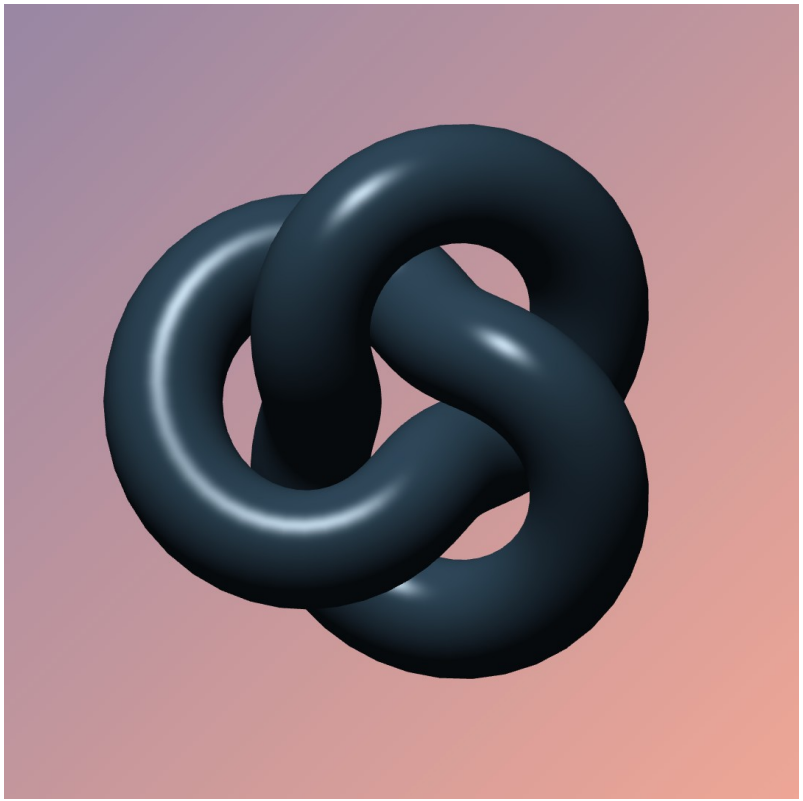
MeshLambertMaterial



- La classe **MeshLambertMaterial** propose un shading basé sur le modèle mathématique *Gouraud*
- Le **Material MeshLambertMaterial** est utilisé pour dessiner un **Mesh** avec des surfaces non-brillantes et sans reflets lumineux
- Le shading est calculé grâce aux sommets de la structure 3D, puis appliqué sur les faces de l'objet
- Prise en compte de la couleur et de l'éclairage

```
const material_lambert = new THREE.MeshLambertMaterial(
{
  color : 0x2c3e50
});
```

MeshPhongMaterial



- La classe **MeshPhongMaterial** se base sur le modèle mathématique *Blinn–Phong*
- Le modèle *Blinn–Phong* est une amélioration de *Gouraud*
- Le shading est calculé pour chaque pixel de l'objet, puis appliqué sur la surface
- Prise en compte de la couleur, de l'éclairage et de la spécularité

```
const material_phong = new THREE.MeshPhongMaterial(
{
  shininess : 80,
  color : 0x2c3e50
});
```

La propriété *shininess*

- La propriété **shininess** est utilisée pour définir la spécularité d'un **Material**
- Cette propriété est disponible uniquement pour les objets incluant une gestion de la spécularité (**MeshPhongMaterial**)

```
const material_phong = new THREE.MeshPhongMaterial(
{
  shininess : 80,
  color : 0x2c3e50
});
```

MeshToonMaterial

- La classe **MeshToonMaterial** est utilisée pour implémenter un effet *Toon Shading* (effet cartoon)
- **MeshToonMaterial** fonctionne sur les mêmes bases que **MeshPhongMaterial**, mais affiche un *shading* sans dégradé d'ombres, sur deux tons
- Prise en compte de la couleur, de l'éclairage et ajoute un effet *Toon shading*

```
const material_toon = new THREE.MeshToonMaterial(
{
  color : 0x2c3e50
});
```

MeshStandardMaterial

- La classe **MeshStandardMaterial** est basée sur le modèle de rendu réaliste : *Physically Based Rendering*
- Grâce à la prise en compte d'effets optiques supplémentaires, les rendus 3D proposés par cette classe sont beaucoup plus proches de la réalité
- Prise en compte de la couleur, de l'éclairage, et des concepts de réflexion optique

```
const material_standard = new THREE.MeshStandardMaterial(
{
  color : 0x2c3e50,
  metalness : 0.25,
  roughness : 0
});
```

MeshPhysicalMaterial

- La classe **MeshPhysicalMaterial** est basée sur le **Material MeshStandardMaterial**
- Cette classe ajoute la possibilité de simuler un vernis brillant : le *clearcoat*
- Prise en compte de la couleur, de l'éclairage, des concepts de réflexion optique et de l'effet vernis *clearcoat*

```
const material_physical = new THREE.MeshPhysicalMaterial(
{
  color : 0x2c3e50,
  metalness : 0.25,
  roughness : 0,
  clearcoat : 1
});
```

Les propriétés *metalness* et *roughness*

- La propriété **metalness** est utilisée pour donner un aspect métallique à un **Material**
- La propriété **roughness** est utilisée pour donner un aspect mat (non-brillant) à un **Material**
- Ces deux propriétés sont utilisables avec les classes **MeshStandardMaterial** et **MeshPhysicalMaterial**

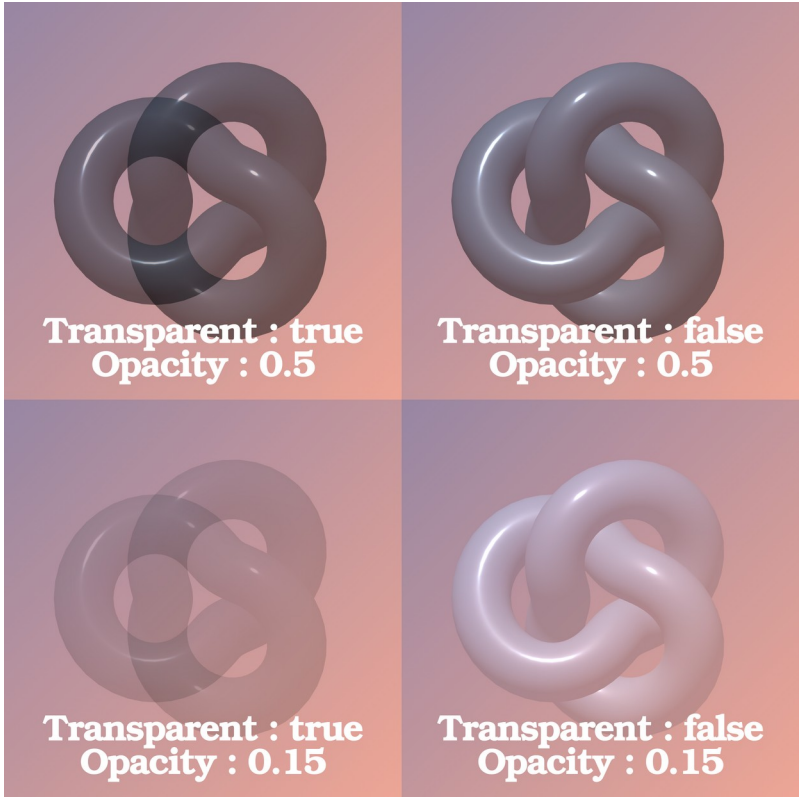
```
const material_standard = new THREE.MeshStandardMaterial(
{
  color : 0x2c3e50,
  metalness : 0.25,
  roughness : 0
});
```

Les propriétés clearcoat et *clearcoatRoughness*

- La propriété **clearcoat** est utilisée pour donner un aspect de verni brillant à un **Material**
- La propriété **clearcoatRoughness** est utilisée pour régler le niveau de non-brillance (aspect mat) du **clearcoat**
- Pour résumer, le verni sera a son niveau maximal de brillance avec un **clearcoat : 1** et un **clearcoatRoughness : 0**
- Ces deux propriétés sont utilisables avec la classe **MeshPhysicalMaterial**

```
const material_physical = new THREE.MeshPhysicalMaterial(
{
  color : 0x2c3e50,
  metalness : 0.25,
  roughness : 0.1,
  clearcoat : 0.75,
  clearcoatRoughness : 0.5
});
```

Les propriétés *transparent* et *opacity*



- La propriété **opacity** est utilisée pour régler l'opacité d'un **Material**
- Si la propriété **transparent** est réglée sur **true**, l'objet 3D sera partiellement transparent en fonction de son opacité
- Si la propriété **transparent** est réglée sur **false**, l'effet visuel du **Material** sera partiellement visible
- Cette propriété est disponible sur tous les types de matériaux

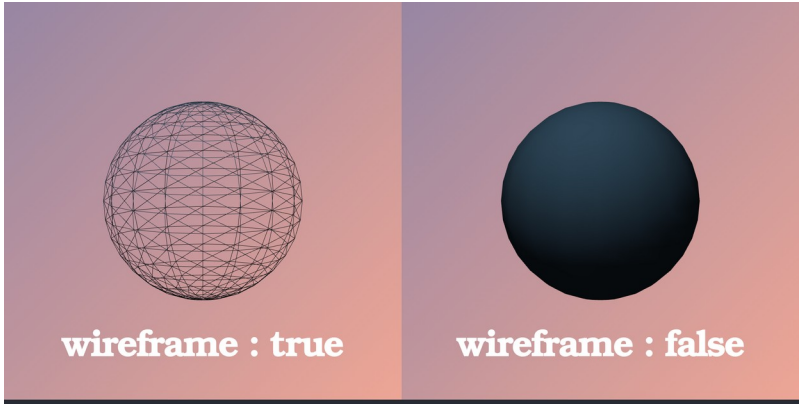
```
const material_physical = new THREE.MeshPhysicalMaterial(
{
  color : 0x2c3e50,
  metalness : 0.25,
  roughness : 0.1,
  clearcoat : 0.75,
  clearcoatRoughness : 0.5,
  transparent : false,
  opacity : 0.15
});
```

La propriété *emissive*

- La propriété **emissive** est utilisée pour régler la couleur émise d'un **Material**
- La couleur émise se comporte comme une couleur brute, visible sans éclairage (Similaire à la couleur d'un **MeshBasicMaterial**)

```
const material_standard = new THREE.MeshStandardMaterial(
{
  color : 0xffffff,
  metalness : 0.25,
  roughness : 0.5,
  emissive : 0xc0392b
});
```

La propriété *wireframe*




```
const material_lambert = new THREE.MeshLambertMaterial(
{
  color : 0x2c3e50,
  wireframe : true
});
```

- La propriété **wireframe** est utilisée pour afficher la structure d'un objet 3D au format « fil-de-fer »
- La valeur de **wireframe** est un **booléen**

Les matériaux Three.js

Si vous êtes prêts, passons à la pratique !