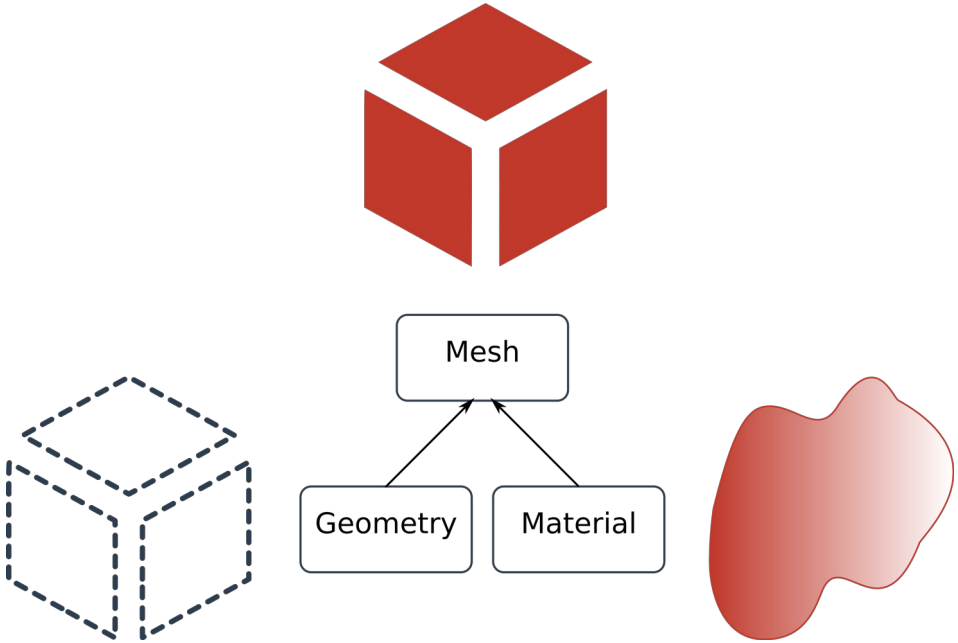
Three.JS University

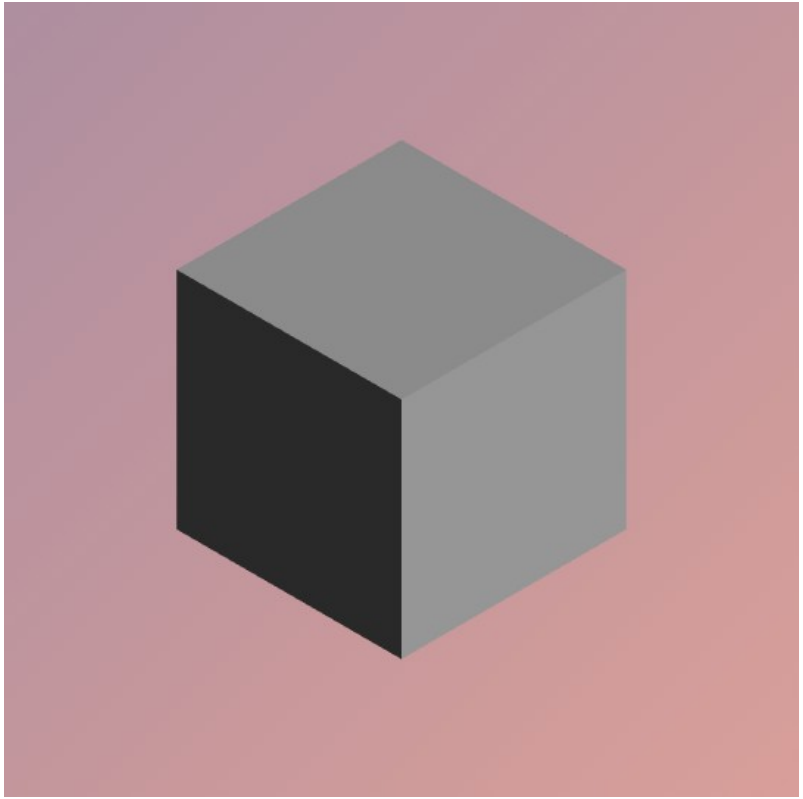
Les structures Geometry primitives de Three.js

Les classes Geometry



- Dans le chapitre précédent, nous avons utilisé **BoxGeometry** pour créer un cube
- Il beaucoup d'autres classes **Geometry** nativement proposées par Three.js

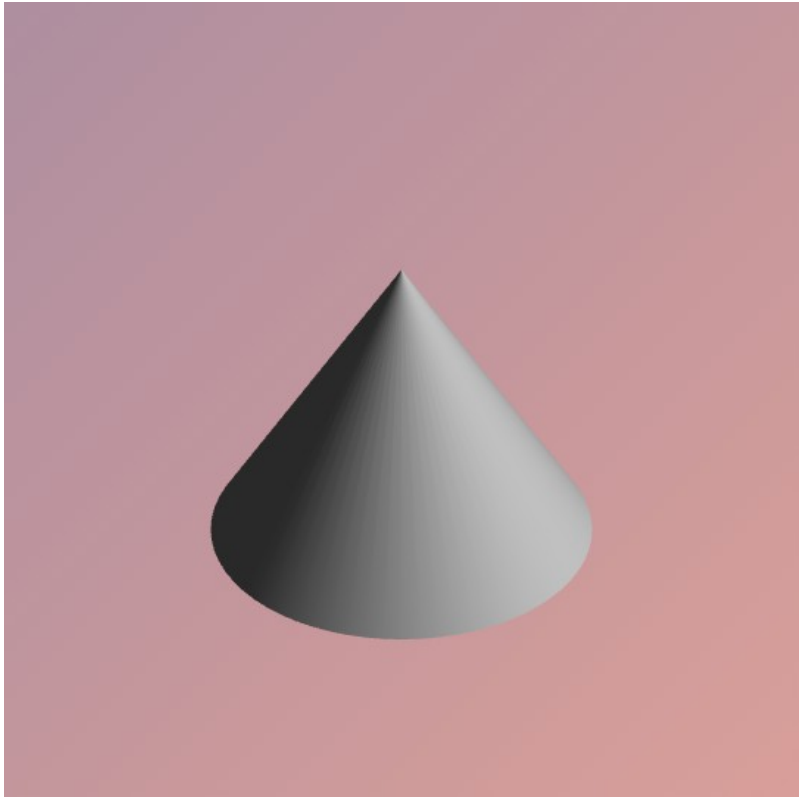
BoxGeometry



- La classe **BoxGeometry** représente une structure parallélépipédique
- Le constructeur de cette classe accepte trois paramètres :
    1. X – Taille de la structure sur l'axe **X**
    2. Y – Taille de la structure sur l'axe **Y**
    3. Z – Taille de la structure sur l'axe **Z**

```
const geometry = new THREE.BoxGeometry( 2, 2, 2 );
```

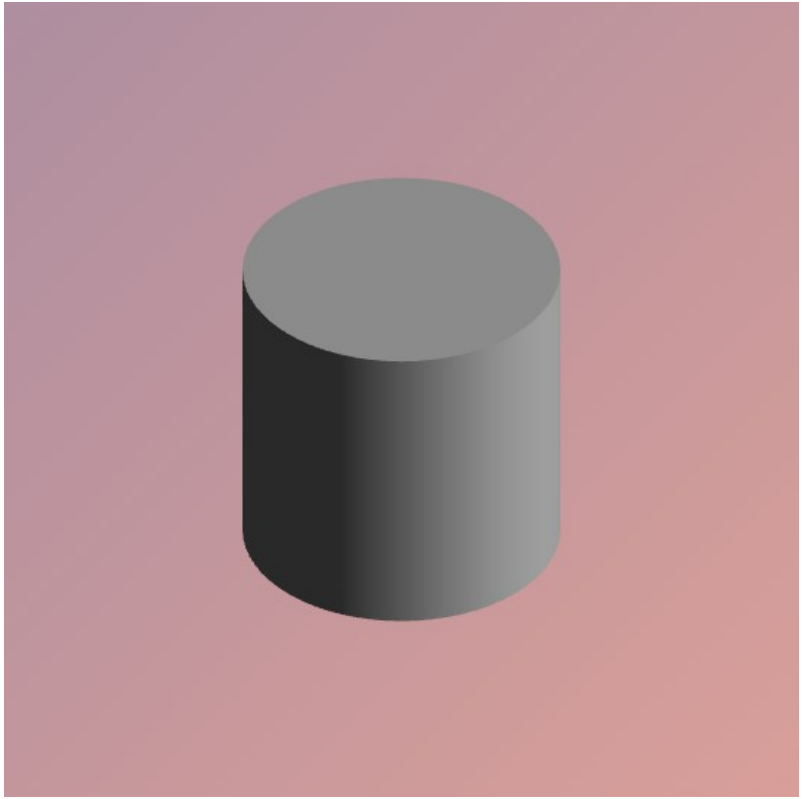
ConeGeometry



- La classe **ConeGeometry** représente une structure conique
- Le constructeur de cette classe accepte trois paramètres :
    1. Rayon – Taille de rayon du cône
    2. Hauteur – Hauteur du cône
    3. Complexité de la structure – Nombre de segments du rayon

```
const geometry_cone = new THREE.ConeGeometry( 1.2, 2, 64 );
```

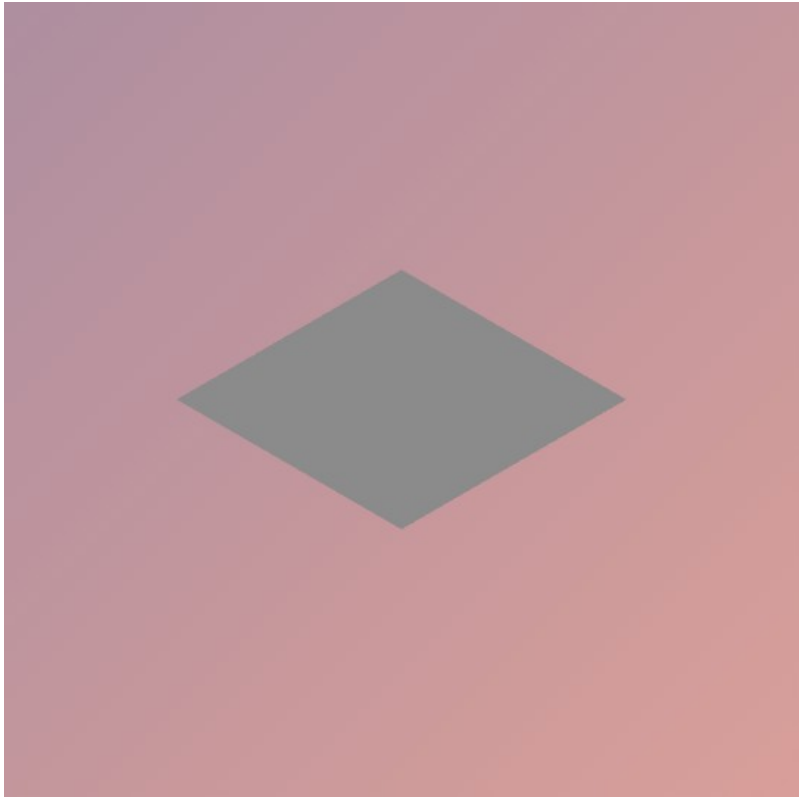
CylinderGeometry



- La classe **CylinderGeometry** représente une structure cylindrique
- Le constructeur de cette classe accepte cinq paramètres :
    1. Rayon supérieur – Taille de rayon de la face supérieure de la structure
    2. Rayon inférieur – Taille de rayon de la face inférieure de la structure
    3. Hauteur – Hauteur du cylindre
    4. Segments rayon – Nombre de segments rayons de la structure
    5. Segments hauteur – Nombre de segments sur la hauteur de la structure

```
const geometry_cylinder = new THREE.CylinderGeometry( 1, 1, 2, 64, 32 );
```

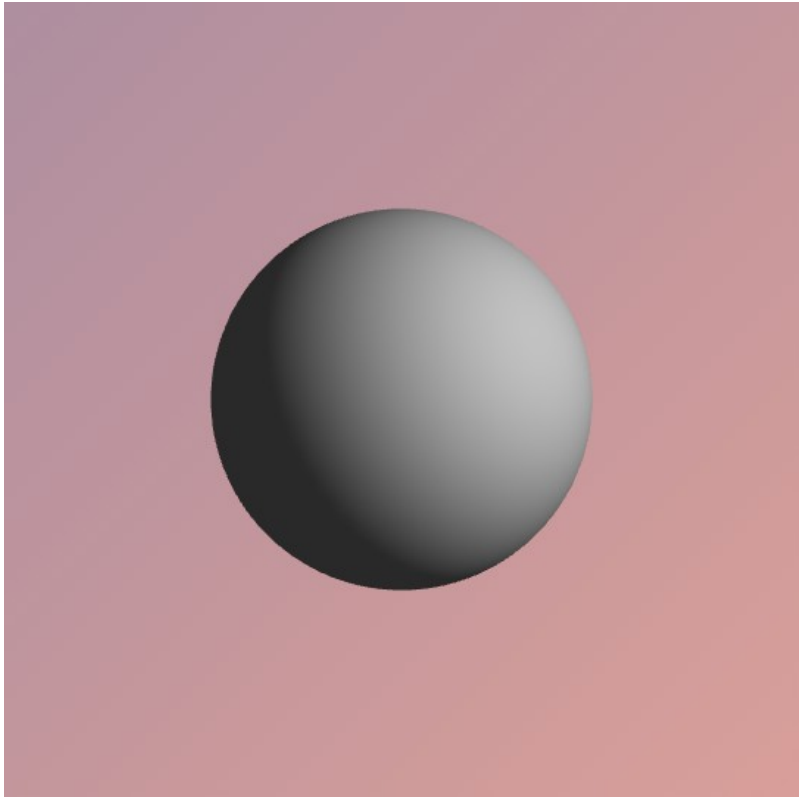
PlaneGeometry



- La classe **PlaneGeometry** représente la structure d'un plan 2D
- Le constructeur de cette classe accepte quatre paramètres :
    1. Largeur – Largeur de la structure sur l'axe **X**
    2. Hauteur – Hauteur de la structure sur l'axe **Y**
    3. Nombre de segments sur la largeur
    4. Nombre de segments sur la hauteur

```
const geometry_plane = new THREE.PlaneGeometry( 2, 2, 2, 2 );
```

SphereGeometry



- La classe **SphereGeometry** représente une structure sphérique
- Le constructeur de cette classe accepte trois paramètres :
    1. Rayon – Taille de rayon de la sphère
    2. Complexité Largeur – Nombre de segments sur la largeur
    3. Complexité Hauteur – Nombre de segments sur la hauteur

```
const geometry_sphere = new THREE.SphereGeometry( 1.2, 64, 64 );
```

IcosahedronGeometry

- La classe **IcosahedronGeometry** représente une structure d’icosaèdre
- Le constructeur de cette classe accepte deux paramètres :
    1. Rayon – Taille de rayon de la structure
    2. Complexité – **Facultatif** (Définir une valeur supérieure à zéro ajoute des sommets, la structure ne sera donc pas un icosaèdre)

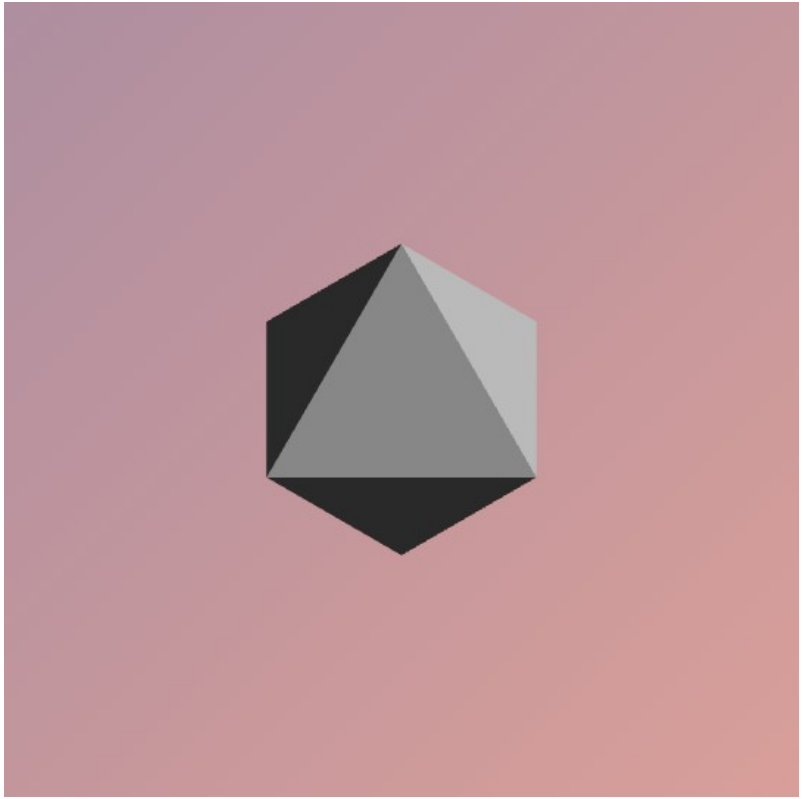
```
const geometry_ico = new THREE.IcosahedronGeometry( 1.2 );
```

TetrahedronGeometry

- La classe **TetrahedronGeometry** représente une structure de tétraèdre
- Le constructeur de cette classe accepte deux paramètres :
    1. Rayon – Taille de rayon de la structure
    2. Complexité – **Facultatif** (Définir une valeur supérieure à zéro ajoute des sommets, la structure ne sera donc pas un tétraèdre)

```
const geometry_tetra = new THREE.TetrahedronGeometry( 1.2 );
```

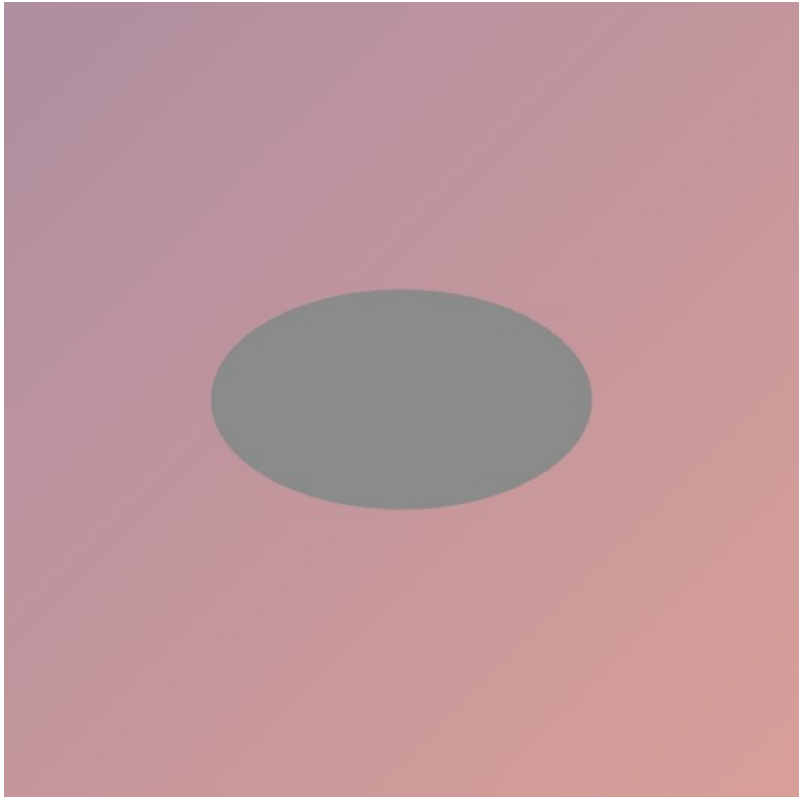
OctahedronGeometry



- La classe **OctahedronGeometry** représente une structure d’octaèdre
- Le constructeur de cette classe accepte deux paramètres :
    1. Rayon – Taille de rayon de la structure
    2. Complexité – Facultatif (Définir une valeur supérieure à zéro ajoute des sommets, la structure ne sera donc pas un octaèdre)

```
const geometry_oct = new THREE.OctahedronGeometry( 1.2 );
```

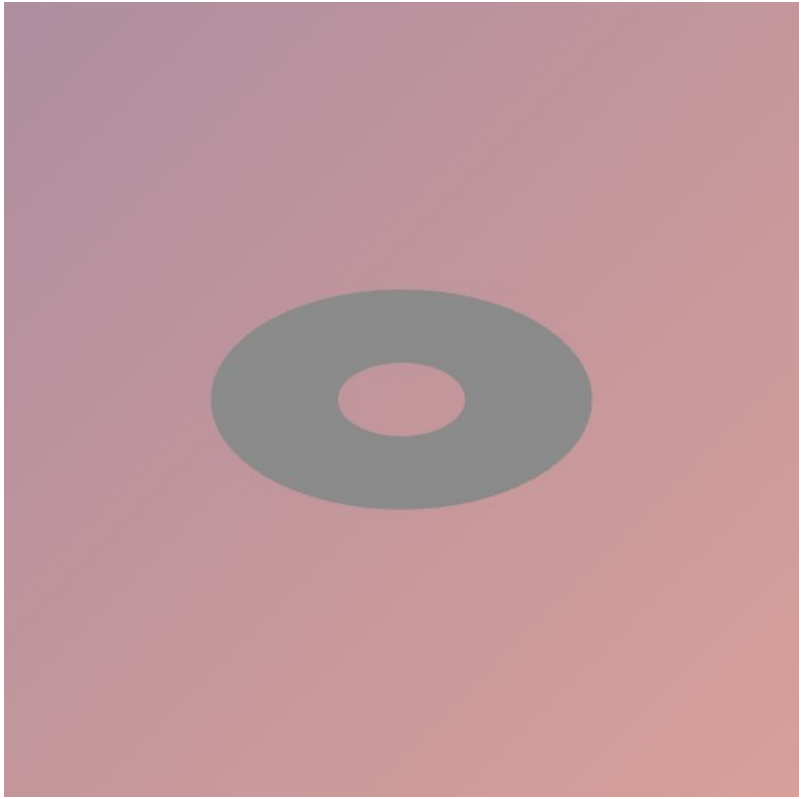
CircleGeometry



- La classe **CircleGeometry** représente une structure de cercle 2D
- Le constructeur de cette classe accepte quatre paramètres :
    1. Rayon – Taille de rayon de la structure
    2. Complexité – Nombre de subdivisions de la structure
    3. ThetaStart – **Facultatif** – Angle de départ du premier segment
    4. ThetaLength – **Facultatif -** Angle central (angle *theta*) de la structure - (*Math.PI *2* pour un cercle complet)

```
const geometry_circle = new THREE.CircleGeometry( 1.2, 64, 0, Math.PI * 2 );
```

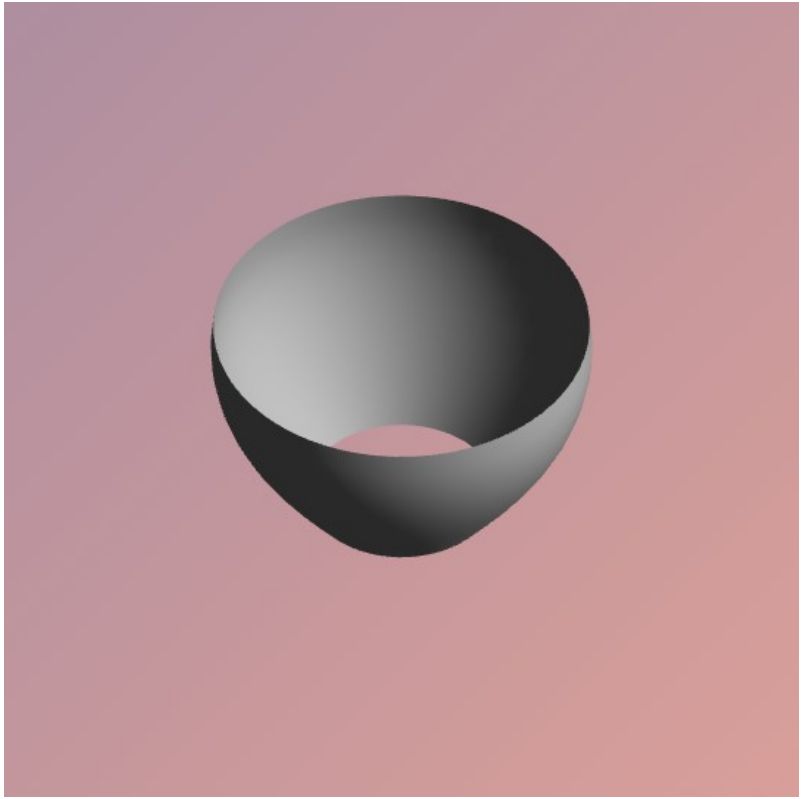
RingGeometry



- La classe **RingGeometry** représente une structure d'anneau 2D
- Le constructeur de cette classe accepte six paramètres :
    1. Rayon intérieur – Taille de rayon intérieur de la structure
    2. Rayon extérieur – Taille de rayon extérieur de la structure
    3. Complexité *théta* – Nombre de subdivisions *théta* de la structure
    4. Complexité *phi* – Nombre de subdivisions *phi* de la structure
    5. ThetaStart – **Facultatif** – Angle de départ du premier segment
    6. ThetaLength – **Facultatif -** Angle central (angle *theta*) de la structure - (*Math.PI *2* pour un cercle complet)

```
const geometry_ring = new THREE.RingGeometry( 0.4, 1.2, 64, 5, 0, Math.PI * 2 );
```

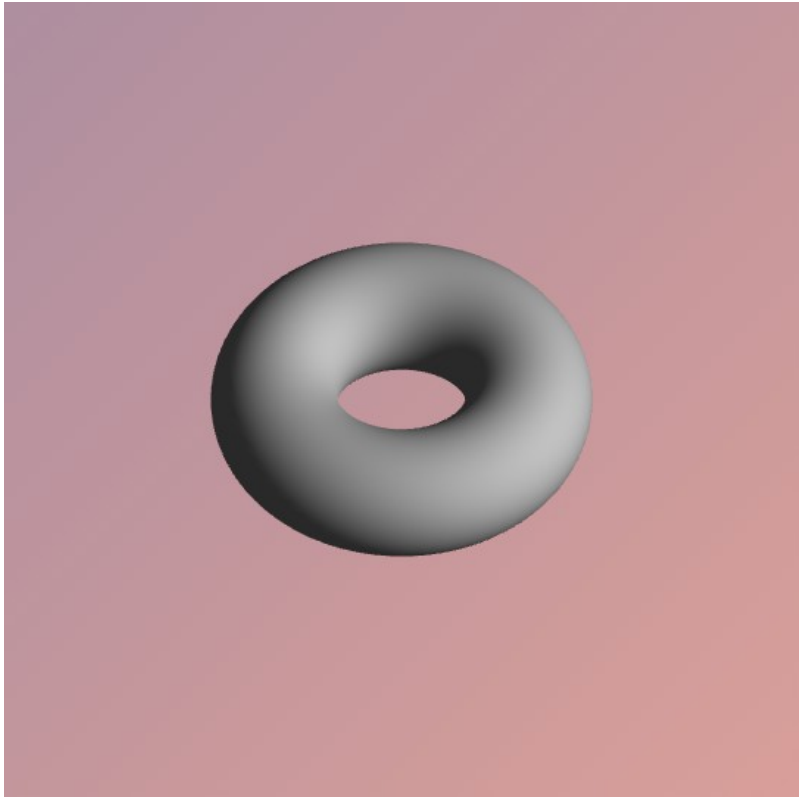
LatheGeometry



- La classe **LatheGeometry** représente une structure créée à partir d'une symétrie axiale mise en rotation, comme un vase
- La rotation est effectuée sur l'axe Y
- Le constructeur de cette classe accepte quatre paramètres :
    1. Points – **Array** de **Vector2** – Coordonnée **X** de chaque point de l'échantillon utilisé pour créer le vase
    2. Complexité - Nombre de subdivisions de la structure
    3. PhiStart – **Facultatif** – Angle radiant de départ du premier segment
    4. PhiLength – **Facultatif** – L'angle radiant de la structure créée - (*Math.PI *2* pour un vase complet)

```
const geometry_lathe =  new THREE.LatheGeometry( points, 64 );
```

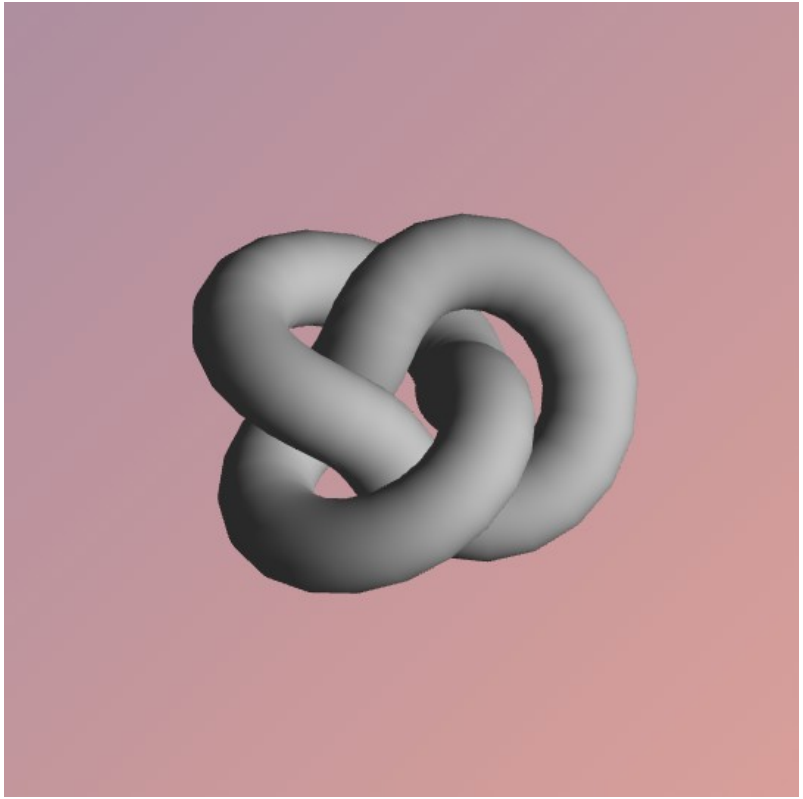
TorusGeometry



- La classe **TorusGeometry** représente une structure d'anneau en 3D
- Le constructeur de cette classe accepte cinq paramètres :
    1. Rayon – Taille de rayon de la structure
    2. Tube – Taille de rayon du tube
    3. Complexité *radiale* - Nombre de subdivisions *radiales* (du rayon de la structure)
    4. Complexité *tubulaire* - Nombre de subdivisions *tubulaires* (du tube)
    5. Arc – **Facultatif** - Angle central de la structure - (*Math.PI *2* pour un cercle complet)

```
const geometry_torus = new THREE.TorusGeometry( 0.8, 0.4, 64, 64 )
```

TorusKnotGeometry



- La classe **TorusKnotGeometry** représente une structure spéciale de type *Torus Knot*
- Le constructeur de cette classe accepte six paramètres :

1. Rayon – Taille de rayon de la structure
2. Tube – Taille de rayon du tube
3. Complexité *tubulaire* - Nombre de subdivisions *tubulaires* (du tube)
4. Complexité *radiale* - Nombre de subdivisions *radiales* (du rayon de la structure)
5. P – **Facultatif** – Le nombre de fois que la géométrie s'enroule autour de son axe de symétrie de rotation
6. Q – **Facultatif** – Le nombre de fois que la géométrie s'enroule autour d'un cercle à l'intérieur de la structure

```
const geometry_torusknot = new THREE.TorusKnotGeometry( 0.8, 0.3, 64, 64 )
```

Les structures Geometry primitives de Three.js

Si vous êtes prêts, passons à la pratique !