**【Three.js入门】纹理及其常用属性、透明纹理、环境遮挡贴图与强度**

【使用 Three.js 实现的效果】

**一、纹理和材质**

纹理：简单来说就是给我们创建好的实体填上一层好看的皮囊。

1. 首先我们要导入纹理 THREE.TextureLoader()
2. 之后我们给纹理附上我们想要的图片（纹理图片很容易百度到）
3. 我们来创建物体 new THREE.BoxBufferGeometry()
4. 导入材质，给 map 赋值为我们的纹理 （map —> 颜色贴图）
5. 之后结合我们的物体和材质，添加到场景中即可

**关键代码：**

// 导入纹理

const textureLoader = new THREE.TextureLoader()

// 纹理加载器 添加纹理图片

const doorColorTexture = textureLoader.load('./textures/doors.png')

// 添加物体

const cubeGeometry = new THREE.BoxBufferGeometry(1, 1, 1)

// 材质

const basicMaterial = new THREE.MeshBasicMaterial({

// map 为颜色贴图

map: doorColorTexture

})

// 结合物体和材质

const cube = new THREE.Mesh(cubeGeometry, basicMaterial)

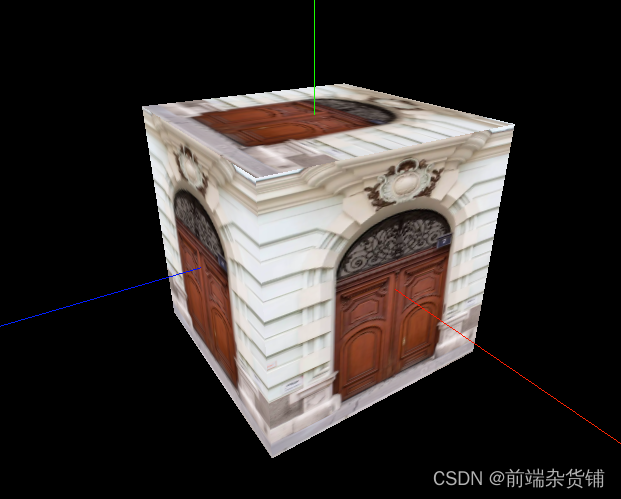
// 场景中添加物体

scene.add(cube)

此处的纹理是这样的（随便百度到的）：



渲染后的效果：



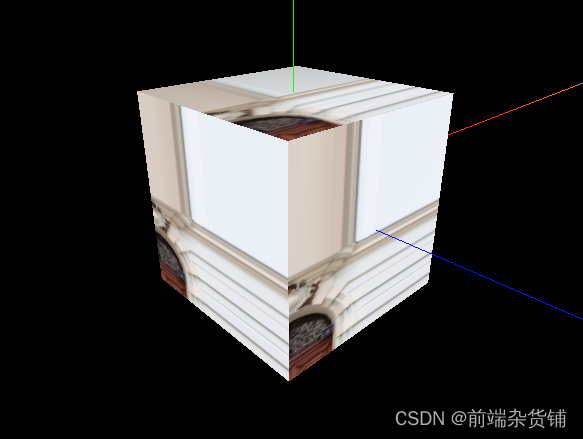
**二、纹理常用属性**

**1、偏移属性**

.offset 偏移属性：即让纹理贴图在物体上做偏移

// 设置偏移（范围0-1）

doorColorTexture.offset.set(0.5, 0.5, 0)

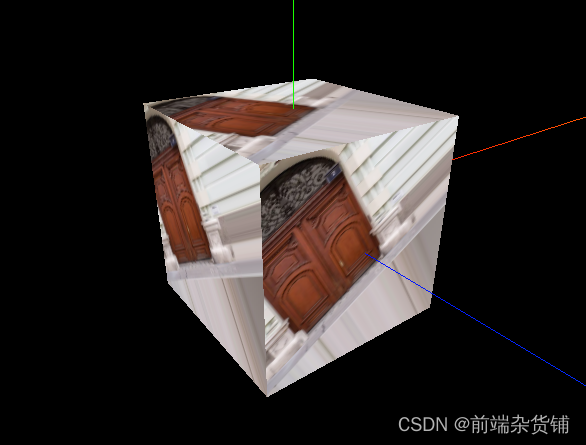


**2、旋转属性**

.rotation 纹理将围绕中心点旋转多少度，单位为弧度（rad），正值为逆时针旋转，默认值问为 0

// 设置旋转 30°（以 Math.PI 180° 为基准）

doorColorTexture.rotation = Math.PI / 6

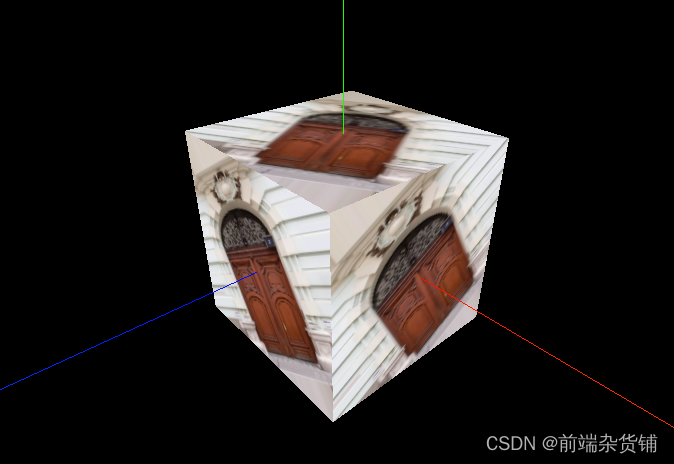


**3、旋转中心点**

.center 旋转中心点 (0.5, 0.5) 对应纹理的中心，默认为 (0, 0)

// 设置旋转中心点 (0.5, 0.5) 对应纹理的中心，默认为 (0, 0)

doorColorTexture.center.set(0.5, 0.5)



**4、纹理的重复**

.repeat 让纹理在物体上重复

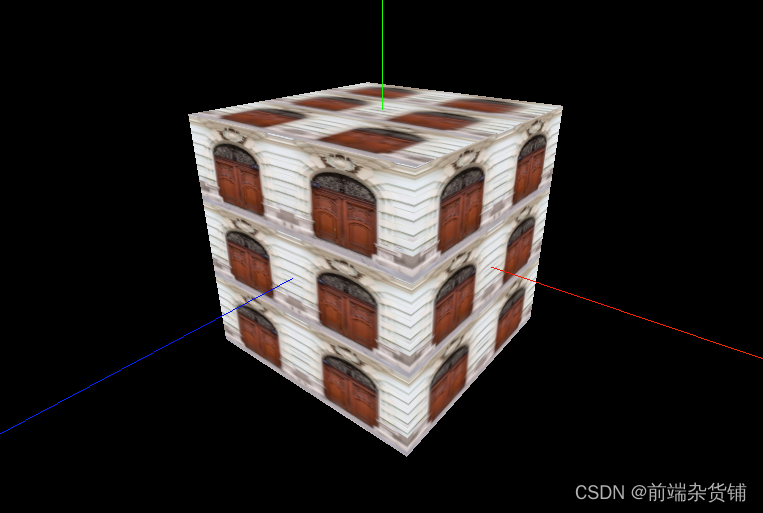
// 设置纹理的重复（x 轴方向重复2次，y 轴方向重复3次）

doorColorTexture.repeat.set(2, 3)

// 设置纹理重复的模式（重复到无穷大）

doorColorTexture.wrapS = THREE.MirroredRepeatWrapping

doorColorTexture.wrapT = THREE.RepeatWrapping



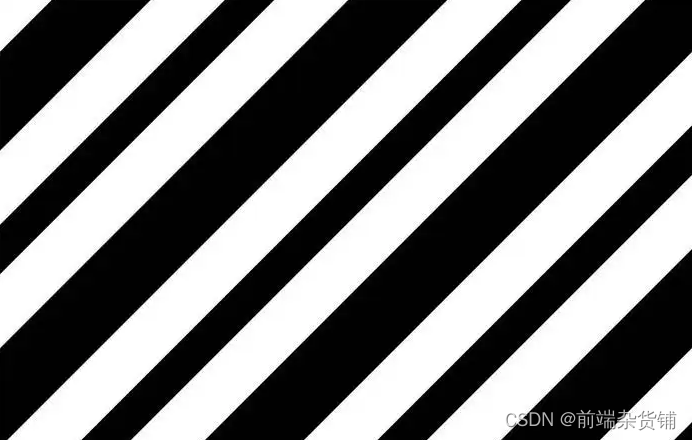
**三、透明纹理**

**1、添加透明纹理**

.alphaMap: Texture

alpha 贴图是一张灰色纹理，用于控制整个表面的不透明度（**黑色：完全透明；白色：完全不透明**）。默认值为 null。

我们在一个面一个门的基础上，添加透明纹理，透明纹理图片如下：



我们首先使用纹理加载器把透明纹理添加进来，之后再在基础材质中添加，注意还要设置透明属性

// 导入纹理

const textureLoader = new THREE.TextureLoader()

// 纹理加载器 添加纹理图片

const doorColorTexture = textureLoader.load('./textures/doors.png')

// 纹理加载器 添加透明纹理

const doorAplhaTexure = textureLoader.load('./textures/bwbk.png')

// 添加物体

const cubeGeometry = new THREE.BoxBufferGeometry(1, 1, 1)

// 材质

const basicMaterial = new THREE.MeshBasicMaterial({

// map 为颜色贴图

map: doorColorTexture,

// alphaMap 为透明纹理

alphaMap: doorAplhaTexure,

// 设置透明属性

transparent: true

})

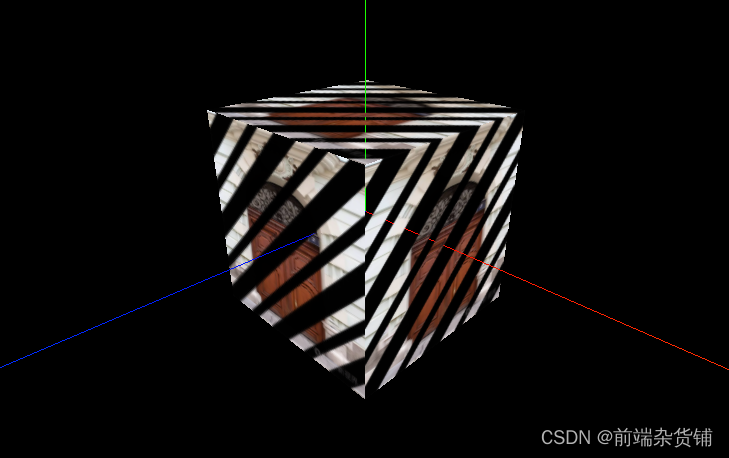
// 结合物体和材质

const cube = new THREE.Mesh(cubeGeometry, basicMaterial)

// 场景中添加物体

scene.add(cube)

https://csdnimg.cn/release/blogv2/dist/pc/img/newCodeMoreBlack.png



**2、添加平面（渲染前面）**

我们在网格中添加一个二维平面，设置为基础材质。注意设置一定的距离。

// 添加平面

const plane = new THREE.Mesh(

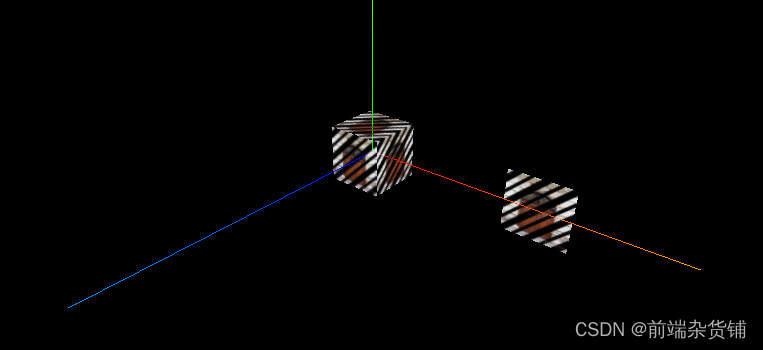
new THREE.PlaneBufferGeometry(1, 1),

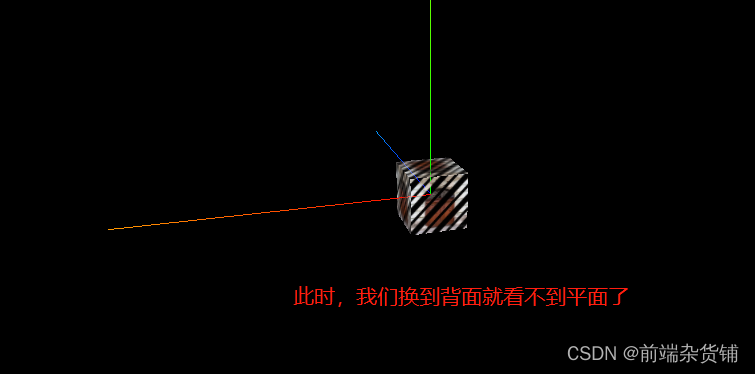
basicMaterial

)

// 设置一定的距离

plane.position.set(3, 0, 0)





**3、平面两面 / 背面渲染**

当我们想渲染两面或者只渲染背面的时候，我们只需要设置一个 side 属性即可

const basicMaterial = new THREE.MeshBasicMaterial({

// map 为颜色贴图

map: doorColorTexture,

// alphaMap 为透明纹理

alphaMap: doorAplhaTexure,

transparent: true,

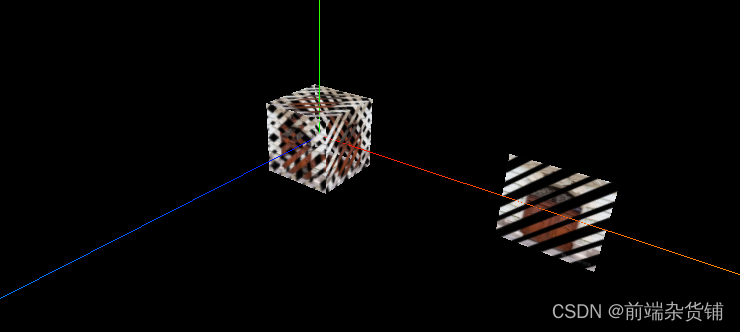
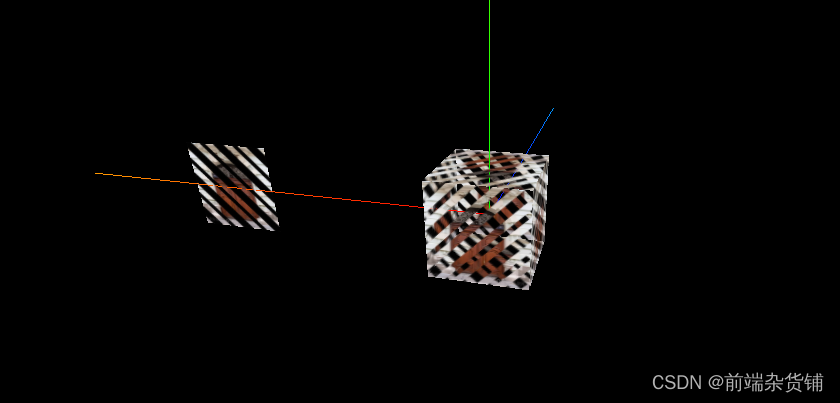
// 渲染两面

side: THREE.DoubleSide

// 渲染背面

// side: THREE.BackSide

})

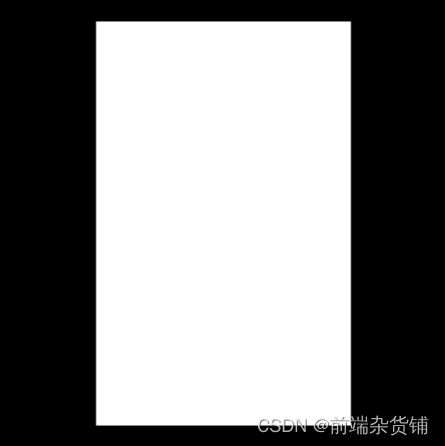
**四、环境遮挡贴图与强度**

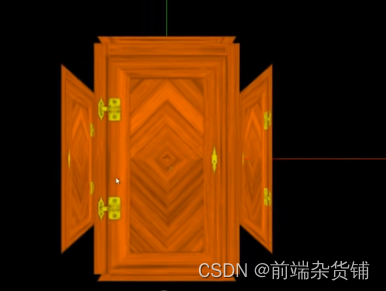
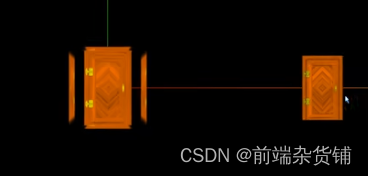
.aoMap 该纹理的红色通道用作环境遮挡贴图。默认值为 null。aoMap 需要第二组 UV。

【UV：纹理坐标通常具有U和V两个坐标轴，因此称之为UV坐标。U代表横向坐标上的分布、V代表纵向坐标上的分布。】

在此我们做一个演示，即实现我们给下面第一张图片添加上第二张图片的观感强度，从而实现第三四张图片的效果（立体逼真感）

**完整代码和图片资源可以通过下方方式加入进来自行获取：**

在这里，我们要使用 aoMap，注意此属性需要第二组 UV，所以我们需要使用自定义属性手动添加第二组 UV，参考代码如下：

// 导入纹理

const textureLoader = new THREE.TextureLoader()

// 纹理加载器 添加纹理图片

const doorColorTexture = textureLoader.load('./textures/doors.png')

// 纹理加载器 添加透明纹理

const doorAplhaTexure = textureLoader.load('./textures/bwbk.png')

// 纹理加载器 添加增强视觉效果的图片

const doorAoTexture = textureLoader.load('./textures/demo.png')

// 添加物体

const cubeGeometry = new THREE.BoxBufferGeometry(1, 1, 1)

// 材质

const basicMaterial = new THREE.MeshBasicMaterial({

// map 为颜色贴图

map: doorColorTexture,

// alphaMap 为透明纹理

alphaMap: doorAplhaTexure,

transparent: true,

aoMap: doorAoTexture

})

// 结合物体和材质

const cube = new THREE.Mesh(cubeGeometry, basicMaterial)

// 场景中添加物体

scene.add(cube)

// 给 cube 添加第二组 UV

cubeGeometry.setAttribute('uv2', new THREE.BufferAttribute(cubeGeometry.attributes.uv.array, 2))

// 添加平面

const planeGeometry = new THREE.PlaneBufferGeometry(1, 1)

const plane = new THREE.Mesh(planeGeometry, basicMaterial)

// 自带一组 UV（Object-geometry-attributes-uv）

console.log(plane)

// 设置一定的距离

plane.position.set(3, 0, 0)

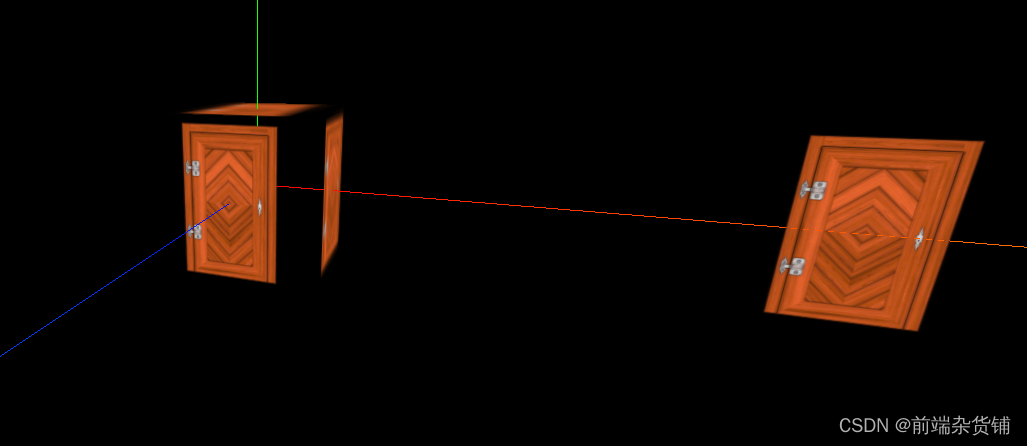
// 添加到场景中

scene.add(plane)

// 给平面设置第二组 UV

planeGeometry.setAttribute('uv2', new THREE.BufferAttribute(planeGeometry.attributes.uv.array, 2))

https://csdnimg.cn/release/blogv2/dist/pc/img/newCodeMoreBlack.png



**五、写在最后（附源码和图片资源）**

这篇文章看完是不是又收获满满呢，接下来进行一下实操练习吧。

相关资源可以通过下面方式加入进来，自行获取。