**控制器、坐标轴辅助器、移动缩放旋转）**

【使用 Three.js 实现的效果】

**一、前言**

**什么是 Three.js ？**

Three.js 是 JavaScript 编写的 WebGL 第三方库。提供了非常多的 **3D 显示功能。**

你可以用它创建各种三维场景，包括了 **摄影机、光影、材质等** 各种对象。

**Three.js 的应用场景 ？**

开发工业软件、智慧城市、轻量的3D展示、建筑家装，以及其他 3D 展示效果。

**几个概念 ？**

场景：场景能够让你在什么地方、摆放什么东西来交给three.js来渲染，这是你放置物体、灯光和摄像机的地方。

摄相机：摄像机的抽象基类。在构建新摄像机时，应始终继承此类。

渲染器：WebGL Render 用WebGL渲染出你精心制作的场景。

**二、使用 Three.js 渲染第一个场景及物体**

**1、基本步骤**

创建场景——>创建相机——>创建物体——>把物体放进场景——>通过渲染器进行渲染

**2、实例展示**

（1）首先导入进来 three 文件（这里是创建了一个项目，通过 npm install 安装的依赖）  
（2）创建场景  
（3）创建相机对象（透视相机）  
（4）设置相机的位置  
（5）在场景里面添加相机  
（6）创建几何体和材质  
（7）根据几何体和材质创建物体  
（8）将几何体添加到场景中  
（9）初始化渲染器  
（10）设置渲染的尺寸大小  
（11）将 webgl 渲染的 canvas 内容添加到 body  
（12）使用渲染器，通过相机将场景渲染进来

// 导入 three 文件

import \* as THREE from 'three'

// 创建场景

const scene = new THREE.Scene()

// 创建相机对象（角度，宽高比，近端，远端）

const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000)

// 设置相机的位置（x，y，z）

camera.position.set(0, 0, 10)

// 场景里面添加相机

scene.add(camera)

// 添加物体

// 创建几何体

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

// 创建材质

const cubeMaterial = new THREE.MeshBasicMaterial({color: 0xffff00})

// 根据几何体和材质创建物体

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

// 将几何体添加到场景当中

scene.add(cube)

// 初始化渲染器

const renderer = new THREE.WebGLRenderer()

// 设置渲染的尺寸大小

renderer.setSize(window.innerWidth, window.innerHeight)

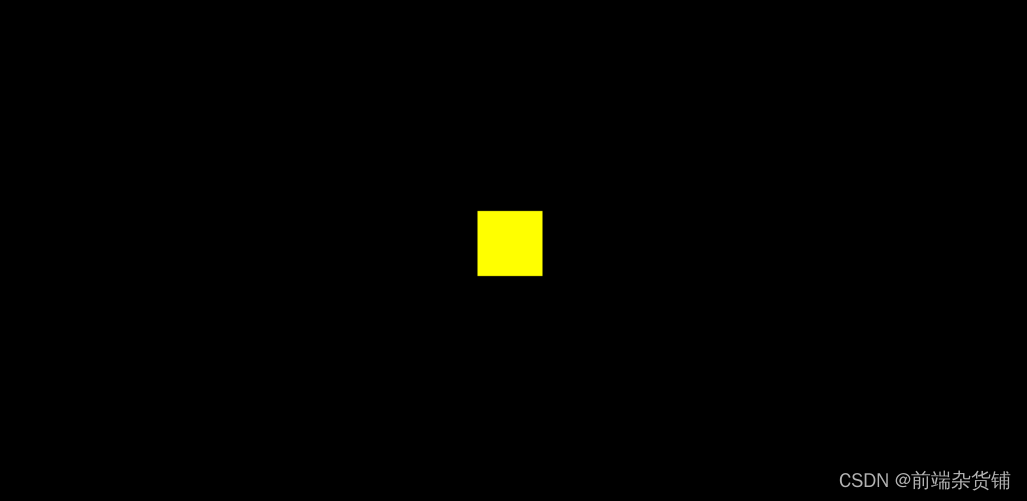
// console.log(renderer)

// 将 webgl 渲染的 canvas 内容添加到 body

document.body.appendChild(renderer.domElement)

// 使用渲染器，通过相机将场景渲染进来

renderer.render(scene, camera)



**三、轨道控制器**

Orbit controls（轨道控制器）可以使得相机围绕目标进行轨道运动。

在上文的基础上进行如下改动，添加轨道控制器，使得物体可旋转

import \* as THREE from 'three'

// 导入轨道控制器

import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls'

// 使用控制器查看3d物体

......

// 使用渲染器，通过相机将场景渲染进来

// renderer.render(scene, camera)

// 创建轨道控制器 （domElement->用于事件监听的 HTML元素）

const controls = new OrbitControls(camera, renderer.domElement)

// 渲染函数

function render() {

renderer.render(scene, camera)

// 请求动画帧-下一帧的时候重新渲染

requestAnimationFrame(render)

}

render()

【使用轨道控制器效果图】

**四、坐标轴辅助器**

红色代表 X 轴，绿色代表 Y 轴，蓝色代表 Z 轴。

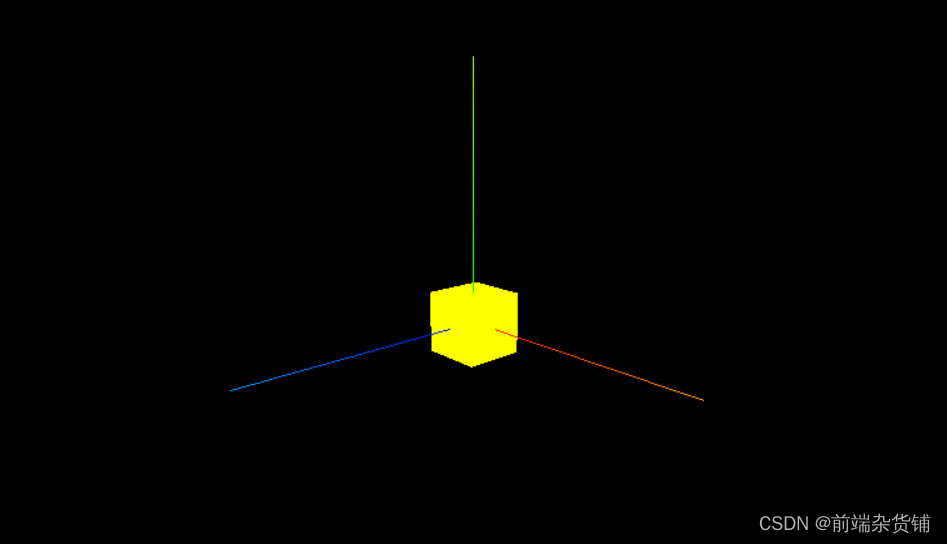
在上文的基础上添加如下代码，添加坐标轴辅助器

// 添加坐标轴辅助器

const axesHelper = new THREE.AxesHelper(4)

// 添加到场景中

scene.add(axesHelper)



**五、设置物体移动**

修改 cube 的 position 即可（x、y、z）

在上文的基础上把渲染函数的代码进行如下改动，就可以让物体移动起来

// 渲染函数

function render() {

cube.position.x += 0.01

if (cube.position.x > 4) {

cube.position.x = 0

}

renderer.render(scene, camera)

// 请求动画帧-下一帧的时候重新渲染

requestAnimationFrame(render)

}

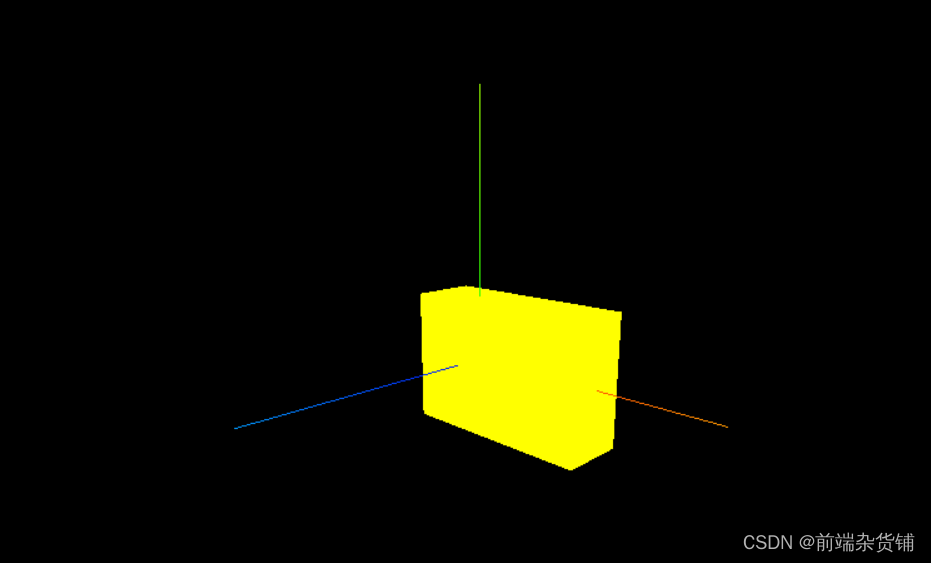
【设置物体的移动】

**六、设置物体的缩放和旋转**

修改 cube 的 scale 即可进行对物体的缩放（x、y、z）

// x 三倍，y 两倍，z 一倍

cube.scale.set(3, 2, 1)



修改 cube 的 rotation 即可进行对物体的旋转（x、y、z）

Math.PI为180°，下面代码即沿 x 轴旋转 45°

cube.rotation.set(Math.PI / 4, 0, 0)

