**【Three.js入门】处理动画、尺寸自适应、双击进入/退出全屏（Clock跟踪时间，Gsap动画库，自适应画面，进入/退出全屏）**

【使用 Three.js 实现的效果】

**一、Clock 跟踪时间处理动画**

接上篇文章，我们创建时钟对象，用于跟踪时间

// 设置时钟（该对象用于跟踪时间）

const clock = new THREE.Clock()

// 渲染函数

function render() {

// 获取自时钟启动后的秒数，同时将旧时间设置为当前时间

let time = clock.getElapsedTime()

let t = time % 4

cube.position.x = t \* 1

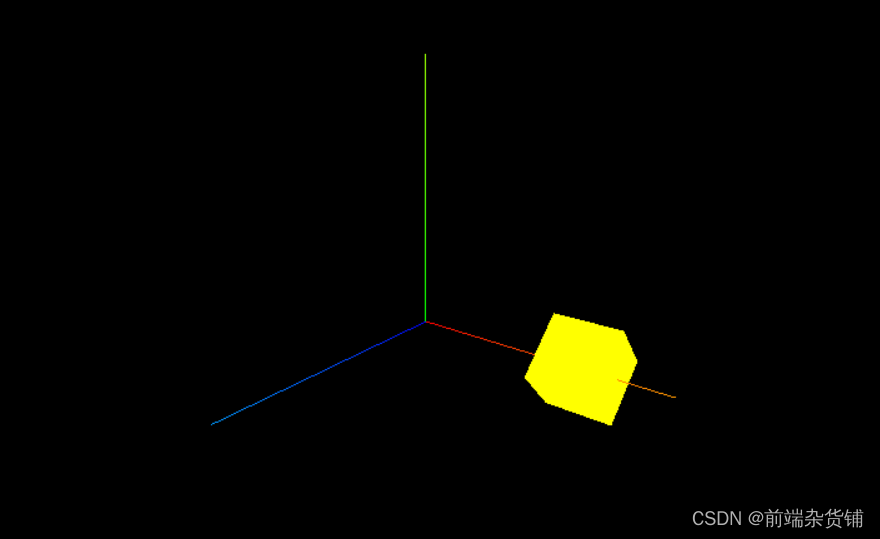
renderer.render(scene, camera)

// 请求动画帧-下一帧的时候重新渲染

requestAnimationFrame(render)

}

render()



**二、Gsap 动画库**

[Gsap 动画库官方网址](https://greensock.com/get-started/#easing)

终端键入如下命令，安装 gsap：

npm install gsap

我们删掉时钟的相关代码，修改如下：

import \* as THREE from 'three'

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

import gsap from "gsap"

......

// 设置动画（0-4的位置5s时间）

let animate = gsap.to(cube.position, {

// x 轴的位置

x: 4,

// 移动的时间

duration: 5,

// 移动方式

ease: "power1.inOut",

// 设置重复的次数，无限次循环 -1

repeat: 2,

// 往返运动

yoyo: true,

// 延迟两秒运动

delay: 2,

// 动画完成

onComplete: () => {

console.log("动画完成")

},

// 动画开始

onStart: () => {

console.log("动画开始");

}

})

// 设置旋转（5s旋转360°）

gsap.to(cube.rotation, {

x: 2 \* Math.PI,

duration: 5,

ease: "power1.inOut"

})

// 双击屏幕暂停/恢复

window.addEventListener("dblclick", () => {

if (animate.isActive()) {

// 动画暂停

animate.pause()

} else {

// 动画恢复

animate.resume()

}

})

// 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()

Gsap 控制动画属性和方法

**三、根据尺寸变化实现自适应画面**

**添加控制器阻尼：更好的转动效果（丝滑）**

根据上文，添加修改代码如下：

// 设置控制器阻尼，让控制器更有真实效果，必须在动画循环里调用.update()

controls.enableDamping = true

// 渲染函数

function render() {

// 在动画循环里调用.update()

controls.update()

renderer.render(scene, camera)

// 请求动画帧-下一帧的时候重新渲染

requestAnimationFrame(render)

}

render()

**监听画面的变化，更新渲染画面（尺寸变化，自动更新）**

在上文基础上，添加如下代码：

window.addEventListener('resize', () => {

console.log("画面变化了");

// 更新摄像头

camera.aspect = window.innerWidth / window.innerHeight

// 更新摄像机的投影矩阵

camera.updateProjectionMatrix()

// 更新渲染器

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

// 设置渲染器的像素比

renderer.setPixelRatio(window.devicePixelRatio)

})

阻尼+自适应画面

**四、控制画布进入&退出全屏**

进入全屏：renderer.domElement.requestFullscreen()，退出全屏：document.exitFullscreen()

注：去掉原先的动画和旋转的双击效果，修改代码如下：

// 双击屏幕进入/退出全屏

window.addEventListener("dblclick", () => {

// 获取全屏（非全屏输出为null，全屏输出为画布）

const fullScreenElement = document.fullscreenElement

if (!fullScreenElement) {

// 双击控制屏幕进入全屏

renderer.domElement.requestFullscreen()

} else {

// 退出全屏

document.exitFullscreen()

}

})



