**【Three.js入门】图形用户界面GUI、BufferGeometry创建矩形、随机生成多个随机颜色的三角形**

【使用 Three.js 实现的效果】

**一、应用图形用户界面更改变量（GUI）**

简介：图形用户界面（Graphical User Interface，简称 GUI，又称图形用户接口）是指采用图形方式显示的计算机操作用户界面。

终端键入如下命令，安装 dat.gui

npm install --save dat.gui

导入 gui，创建 gui 实例，修改代码如下：

// 导入 dat.gui

import \* as dat from "dat.gui"

.......

// 创建 gui 实例

const gui = new dat.GUI()

// 在 x 轴移动（最小0，最大4，步数0.01，设置名字，改变时输出，完全停下来的时候也输出）

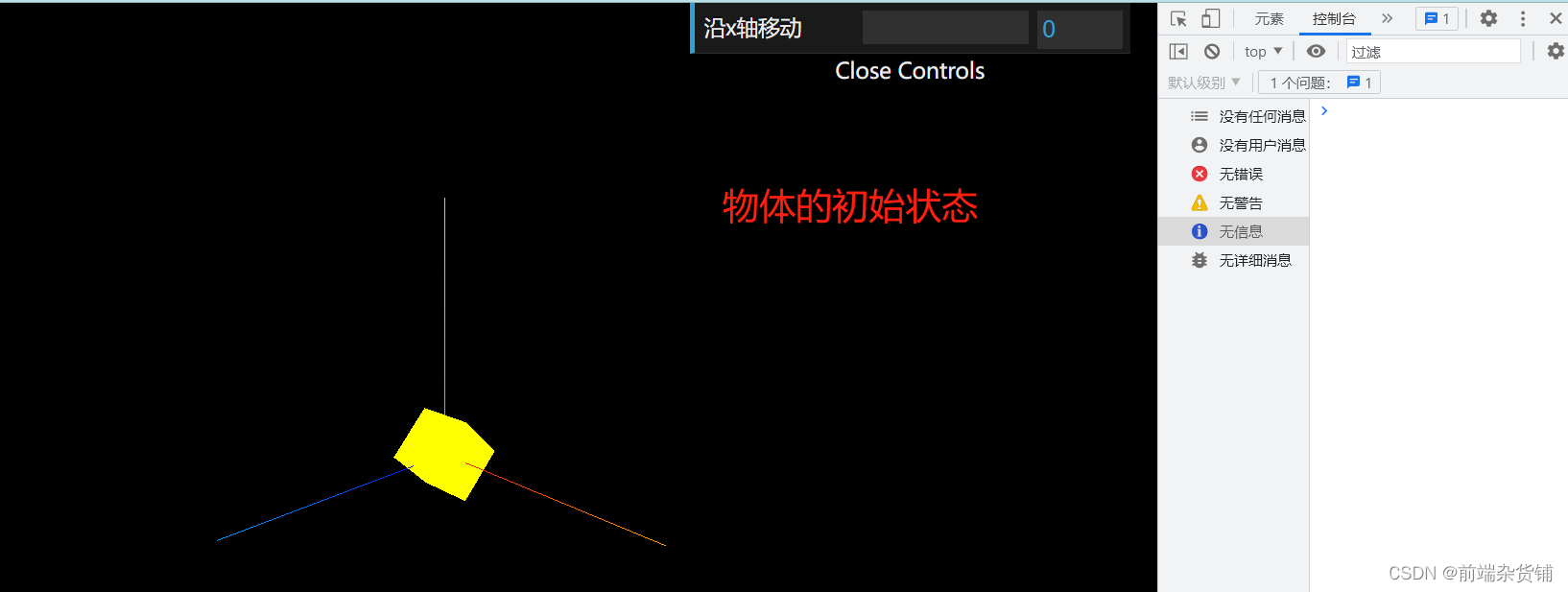
gui.add(cube.position, "x").min(0).max(4).step(0.01).name("沿x轴移动").onChange((value) => {

console.log("值被修改了：", value)

}).onFinishChange((value) => {

console.log("完全停下来：", value)

})





**接下来，深入了解一下 gui，完成修改物体颜色，显示/隐藏物体，让物体移动，以及添加文件夹显示边线的功能**

......

const params = {

// 修改物体的颜色

color: "#ffff00",

// 让立方体运动起来（2s移动到位置4），往返运动，无限循环

fn: () => {

gsap.to(cube.position, {

x: 4,

duration: 2,

yoyo: true,

repeat: -1

})

}

}

// 改变物体颜色

gui.addColor(params, "color").onChange((value) => {

cube.material.color.set(value)

})

// 设置选项框（显示/隐藏物体）

gui.add(cube, "visible").name('显示/隐藏物体')

// 设置按钮点击触发某个事件

gui.add(params, 'fn').name('立方体运动')

// 设置文件夹

let folder = gui.addFolder('设置立方体')

// 显示实体/边线

folder.add(cube.material, 'wireframe')

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

【GUI 用户界面更改变量】

**二、BufferGeometry 设置顶点创建矩形**

通过六个点，渲染出一个矩形

把原先 cube 相关内容删掉，我们通过创建六个点来渲染出一个矩形

import \* as THREE from 'three'

// 导入轨道控制器

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

// 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

// 创建矩形

const geometry = new THREE.BufferGeometry()

// 设置顶点

const vertices = new Float32Array([

-1.0, -1.0, 1.0,

1.0, -1.0, 1.0,

1.0, 1.0, 1.0,

1.0, 1.0, 1.0,

-1.0, 1.0, 1.0,

-1.0, -1.0, 1.0

])

// 设置每三个值为一个坐标

geometry.setAttribute('position', new THREE.BufferAttribute(vertices, 3))

// 创建材质

const material = new THREE.MeshBasicMaterial({

color: 0xffff00

})

// 根据几何体和材质创建物体

const mesh = new THREE.Mesh(geometry, material)

// 将物体添加到场景中

scene.add(mesh)

console.log(mesh)

// 修改物体的颜色

const params = { color: "#ffff00" }

// 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)

// 创建轨道控制器

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

// 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

// 添加坐标轴辅助器

const axesHelper = new THREE.AxesHelper(4)

// 添加到场景中

scene.add(axesHelper)

// 渲染函数

function render() {

controls.update()

renderer.render(scene, camera)

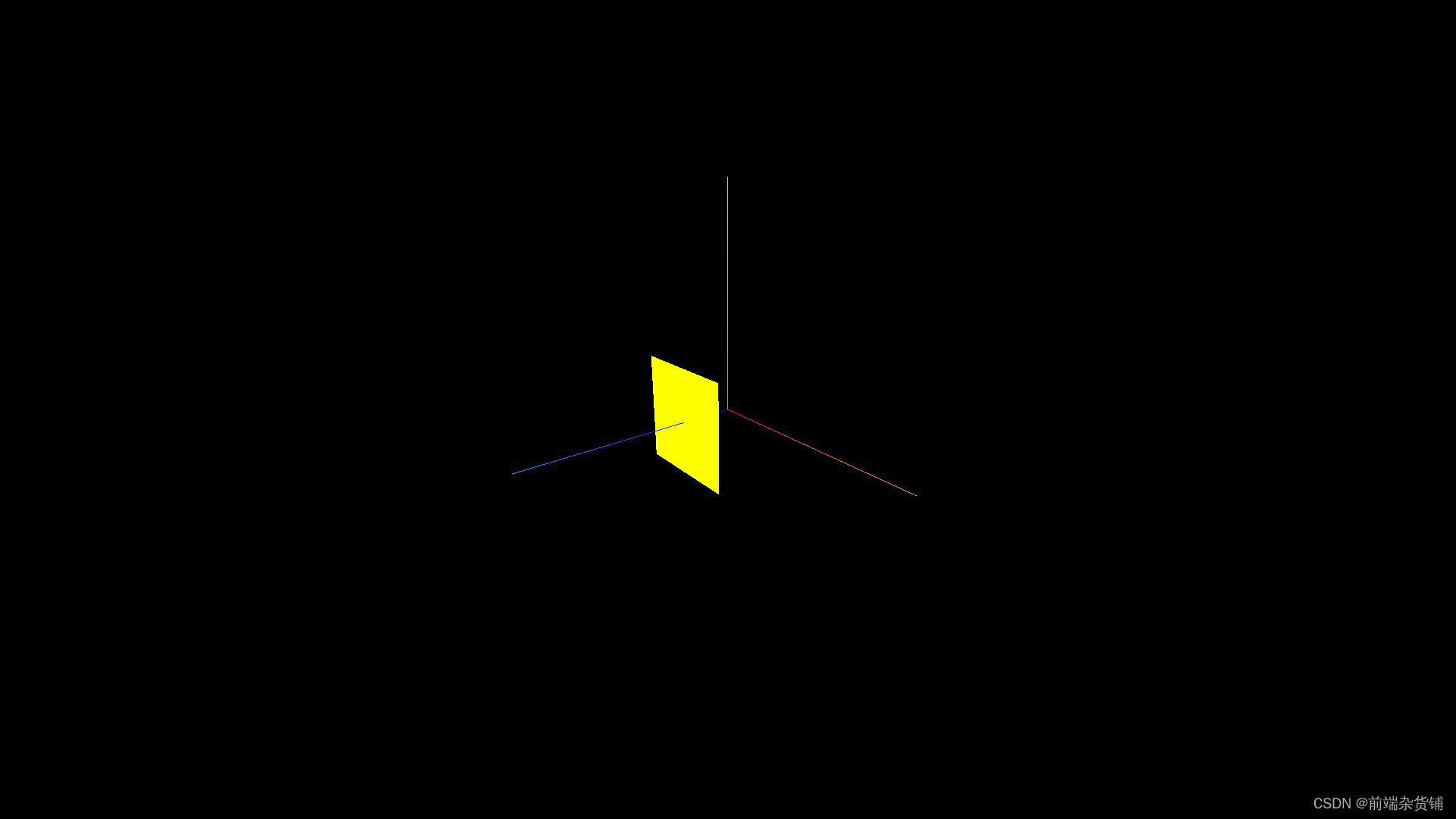
// 请求动画帧-下一帧的时候重新渲染

requestAnimationFrame(render)

}

render()

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



**三、随机生成多个三角形**

在上文知识的基础上，我们来实现 **随机生成多个不同颜色不同大小的三角形**

我们随机生成 45 个随机颜色的三角形，每xyz三个坐标确定一个点，每三个点确定一个三角形，随机生成它们

import \* as THREE from 'three'

// 导入轨道控制器

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

// 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

// 创建三角形

for (let i = 0; i < 45; i++) {

// 每个三角形，需要3个顶点，每个顶点需要3个值

const geometry = new THREE.BufferGeometry()

// 32 位的浮点数型数组，设置顶点

const positionArray = new Float32Array(9)

for (let j = 0; j < 9; j++) {

positionArray[j] = Math.random() \* 10 - 5

}

// 设置每三个值为一个坐标

geometry.setAttribute('position', new THREE.BufferAttribute(positionArray, 3))

// 随机生成颜色

let color = new THREE.Color(Math.random(), Math.random(), Math.random())

// 创建材质（设置颜色，透明度）

const material = new THREE.MeshBasicMaterial({

color: color,

// 设置透明度

transparent: true,

opacity: 0.6

})

// 根据几何体和材质创建物体

const mesh = new THREE.Mesh(geometry, material)

// 将物体添加到场景中

scene.add(mesh)

console.log(mesh)

}

// 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)

// 创建轨道控制器

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

// 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

// 添加坐标轴辅助器

const axesHelper = new THREE.AxesHelper(5)

// 添加到场景中

scene.add(axesHelper)

// 渲染函数

function render() {

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

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

// 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)

})

【随机生成多个随机颜色的三角形】