## 1.其实raycaster就是射线拾取模型，可以和鼠标配合，光线投射用于进行鼠标拾取（在三维空间中计算出鼠标移过了什么物体）。

## 2.实例：编写一个程序，当鼠标点击立方体，他会变为红色

|  |
| --- |
|  |

## 3.代码如下

### src/lib/threeapp.js

|  |
| --- |
| import \* as THREE from 'three'  import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls'  import Stats from 'three/examples/jsm/libs/stats.module';  //注意：这是第二课的class，内容有点不一样所以我们把class名称改为ThreeApp2  //帧率使用聚光灯而不是平行光  export default class ThreeApp{      constructor(canvasId){          //1.定义场景，相机，渲染器          this.scene = undefined          this.camera = undefined          this.renderer = undefined          //2.定义相机参数,但是保存在本类中          this.fov = 45          this.nearPlane = 1          this.farPlane = 1000          this.canvasId = canvasId          //3.定义额外组件          this.clock = undefined          this.controls = undefined          this.stats = undefined          //4.定义环境光和聚光灯          this.ambientLight = undefined          this.directionalLight = undefined      }      initApp(){          //创建场景对象并且赋值给成员变量          this.scene = new THREE.Scene()          //创建相机对象并且用相机成员变量接收          this.camera = new THREE.PerspectiveCamera(              this.fov,              window.innerWidth/window.innerHeight,              this.nearPlane,              this.farPlane          )          //调整相机位置          this.camera.position.z = 12 //cameraz轴值越大，图像越小          //根据传入的id获取画布对象          let canvas = document.getElementById(this.canvasId)          //创建渲染器          this.renderer = new THREE.WebGLRenderer({               canvas,               antialias:true           })            //设置渲染器的渲染尺寸          this.renderer.setSize(window.innerWidth,window.innerHeight)          //添加到body中          document.body.appendChild(this.renderer.domElement)          //创建时钟，轨道控制器，检测帧数(FPS)的工具          this.clock = new THREE.Clock()          this.controls = new OrbitControls(this.camera,this.renderer.domElement)          this.stats = Stats()          //将检测帧数(FPS)的工具添加到body中          document.body.appendChild(this.stats.domElement)          //创建环境光          this.ambientLight = new THREE.AmbientLight(0xffffff,0.5)          this.ambientLight.castShadow = true          //把环境光添加到场景中          this.scene.add(this.ambientLight)          //创建方向光          this.directionalLight = new THREE.SpotLight(0xffffff,1)          //设置方向光的位置          this.directionalLight.position.set(0,32,64) //位置也不一样            //添加方向光到场景中          this.scene.add(this.directionalLight)          //给window对象添加事件监听，用来实现窗口的响应式功能          window.addEventListener('resize',()=>this.onWindowResize(),false)      }        //定义animate成员函数，这个函数在外部调用      animate(){          window.requestAnimationFrame(this.animate.bind(this))          this.render()          this.stats.update()          this.controls.update()      }      render(){          this.renderer.render(this.scene,this.camera)      }      onWindowResize(){          //重新计算相机的宽高比          this.camera.aspect = window.innerWidth/window.innerHeight          //更新相机的投影矩阵          this.camera.updateProjectionMatrix()          //重新设置渲染器的渲染大小          this.renderer.setSize(window.innerWidth,window.innerHeight)      }  } |

### src/lib/addonutil.js

|  |
| --- |
| import {GUI} from 'three/examples/jsm/libs/lil-gui.module.min.js'  import {OrbitControls} from 'three/examples/jsm/controls/OrbitControls.js'  import {GLTFLoader} from 'three/examples/jsm/loaders/GLTFLoader.js'  import {FBXLoader} from 'three/examples/jsm/loaders/FBXLoader.js'  import {VOXLoader} from 'three/examples/jsm/loaders/VOXLoader.js'  import {TTFLoader} from 'three/examples/jsm/loaders/TTFLoader.js'  import { FontLoader } from 'three/examples/jsm/loaders/FontLoader';  import { TextGeometry } from 'three/examples/jsm/geometries/TextGeometry';  //很简单就是把这些类导入进来又导出，做的编写一次，使用多次的目的  export {    GUI,OrbitControls,GLTFLoader,FBXLoader,VOXLoader,TTFLoader,FontLoader,TextGeometry  } |

### src/App.jsx

|  |
| --- |
| import { useEffect } from 'react'  import './App.css'  import {GLTFLoader}from './lib/addonutil'  import \* as THREE from 'three'  import ThreeApp from './lib/threeapp'    function main1() {    //创建three应用程序对象    let threeApp = new ThreeApp("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let addNewBoxMesh = (x, y, z) => {      let boxGeo = new THREE.BoxGeometry(1, 1, 1)      let boxMat = new THREE.MeshPhongMaterial({ color: 0xfafafa })      let box = new THREE.Mesh(boxGeo, boxMat)      box.position.x = x      box.position.y = y      box.position.z = z      threeApp.scene.add(box)    }    // top rows    addNewBoxMesh(0, 2, 0);    addNewBoxMesh(2, 2, 0);    addNewBoxMesh(-2, 2, 0);    addNewBoxMesh(0, 2, -2);    addNewBoxMesh(2, 2, -2);    addNewBoxMesh(-2, 2, -2);    addNewBoxMesh(0, 2, 2);    addNewBoxMesh(2, 2, 2);    addNewBoxMesh(-2, 2, 2);    // middle rows    addNewBoxMesh(0, 0, 0);    addNewBoxMesh(2, 0, 0);    addNewBoxMesh(-2, 0, 0);    addNewBoxMesh(0, 0, -2);    addNewBoxMesh(2, 0, -2);    addNewBoxMesh(-2, 0, -2);    addNewBoxMesh(0, 0, 2);    addNewBoxMesh(2, 0, 2);    addNewBoxMesh(-2, 0, 2);    // bottom rows    addNewBoxMesh(0, -2, 0);    addNewBoxMesh(2, -2, 0);    addNewBoxMesh(-2, -2, 0);    addNewBoxMesh(0, -2, -2);    addNewBoxMesh(2, -2, -2);    addNewBoxMesh(-2, -2, -2);    addNewBoxMesh(0, -2, 2);    addNewBoxMesh(2, -2, 2);    addNewBoxMesh(-2, -2, 2);    let loadedModel    // let gltfLoader = new AddOn.GLTFLoader()    let gltfLoader = new GLTFLoader()    gltfLoader.load('../assets/shiba/scene.gltf', (gltfScene) => {      loadedModel = gltfScene      gltfScene.scene.position.y = 0.5      gltfScene.scene.position.z = 4      gltfScene.scene.scale.set(1, 1, 1)      threeApp.scene.add(gltfScene.scene)    })    //光线拾取操作    let pointer = new THREE.Vector2()    let raycaster = new THREE.Raycaster()    function onMouse(event) {      console.log(event);      pointer.x = (event.clientX / window.innerWidth) \* 2 - 1      pointer.y = -(event.clientY / window.innerHeight) \* 2 + 1      raycaster.setFromCamera(pointer, threeApp.camera)      //intersects      let intersects = raycaster.intersectObjects(threeApp.scene.children)      if (intersects.length > 0) {        intersects[0].object.material.color.set(0xff0000)      }    }    //添加鼠标移动事件监听    window.addEventListener('mousemove', onMouse)    // function animate() {    //   if (loadedModel) {    //     loadedModel.scene.rotation.x += 0.01    //     loadedModel.scene.rotation.y += 0.01    //     loadedModel.scene.rotation.z += 0.01    //   }    //   requestAnimationFrame(animate);    // }    // animate()  }  function App() {    useEffect(() => {      main1()    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |

### 效果：

### 点击前

|  |
| --- |
|  |

### 点击后，只要鼠标移动到上面就会有效果：

|  |
| --- |
|  |