# 这个教程使用react+threejs的环境

## 1.项目的根目录是：F:\Projects\_f\Threejs\_projs\three-app-youtube，我们在这里用vscode打开终端，输入npm create vite@latest命令，会先下载vite，然后它提示项目名称

|  |
| --- |
|  |

## 2.输入three-setup回车，会问是否使用框架

|  |
| --- |
|  |
|  |

## 3.使用箭头键移动到React，回车，会问你使用react还是react-ts

|  |
| --- |
|  |
|  |

## 4.选择JavaScript，回车，项目创建完成

|  |
| --- |
|  |
|  |

## 5.根据提示进入three-setup文件夹，然后输入npm install，回车，就开始按照依赖，安装完后如下图

|  |
| --- |
|  |
|  |

### 测试：在终端中输入npm run dev，出现下面的界面，说明下面初始化成功

|  |
| --- |
|  |

### 然后按住ctrl点击链接，效果如下

|  |
| --- |
|  |

## 6.我们把项目中一些不需要的文件删除，然后App.jsx文件内容如下

|  |
| --- |
|  |

## 7.给index.html文件添加对tailwind css库的引用

|  |
| --- |
|  |

## 8.然后我们可以在App.jsx里面给里面的div设置tailwind的class样式

|  |
| --- |
|  |

### 注意：div里面的是class而不是className，因为这个样式不是react提供的

### 效果如下

|  |
| --- |
|  |

## 9.先把项目停了，然后在终端输入npm install three按住threejs,安装好后如下图

|  |
| --- |
|  |
|  |

## 10.然后再修改一些App.jsx代码，添加3D页面

|  |
| --- |
| import { useEffect } from 'react'  import './App.css'  import \* as THREE from 'three'  import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls'  function App() {    useEffect(() => {      //创建场景      let scene = new THREE.Scene()      //创建相机      let camera = new THREE.PerspectiveCamera(        50,        window.innerWidth / window.innerHeight,        0.1,        1000      )      camera.position.z = 96      //获取画布      let canvas = document.getElementById('myThreeJSCanvas')      //利用画布创建渲染对象      let renderer = new THREE.WebGLRenderer({        canvas,        antialias: true      })      //把渲染器添加到body中      renderer.setSize(window.innerWidth, window.innerHeight)      document.body.appendChild(renderer.domElement)      //创建环境光      let amientLight = new THREE.AmbientLight(0xffffff, 0.5)      scene.add(amientLight)      //创建方向光      let directionalLight = new THREE.DirectionalLight(0xffffff, 1)      directionalLight.position.set(0, 32, 64)      scene.add(directionalLight)      let cubeGeo = new THREE.BoxGeometry(16, 16, 16)      let material = new THREE.MeshNormalMaterial()      let cube = new THREE.Mesh(cubeGeo, material)      scene.add(cube)      let control = new OrbitControls(camera, renderer.domElement)      function animate() {        window.requestAnimationFrame(animate)        cube.rotation.x += 0.01        cube.rotation.y += 0.01        renderer.render(scene, camera)        control.update()      }      animate()      window.addEventListener('resize', () => {        camera.aspect = window.innerWidth / window.innerHeight        renderer.setSize(window.innerWidth, window.innerHeight)        camera.updateProjectionMatrix()        renderer.setPixelRatio(window.devicePixelRatio)      })    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |

### 运行程序，然后就会出现应该每个面的颜色都不一样的旋转立方体

|  |
| --- |
|  |
|  |

## 11.然后，为了方便学习，我们把这个项目复制一份，（注意：复制项目后，如果需要修改包名，注意包名不能有数字，否则报错）改名three-lession1，然后我们还可以使用面向对象的编程方法来改造这个程序

|  |
| --- |
|  |

## 12.我们在src文件夹里面新建一个lib文件夹，把App.jsx里面的threejs的主要代码封装到一个叫做threeapp.js文件里面的ThreeApp类，需要修改写法。

|  |
| --- |
| import \* as THREE from 'three'  import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls'  import Stats from 'three/examples/jsm/libs/stats.module';  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.定义环境光和方向光          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 = 48          //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，其实是平行光          this.directionalLight = new THREE.DirectionalLight(0xffffff,1)          //设置方向光的位置          this.directionalLight.position.set(0,32,64)          //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)      }  } |

## 13.然后我们需要在App.jsx里面导入这个类

|  |
| --- |
|  |

## 14.然后App.jsx的代码就可以背景精简了

|  |
| --- |
|  |

### 注意：这里我们没有条件使立方体旋转的代码，目的是为了测试轨道控制器是否能够工作

## 15.保存文件，项目会热加载，然后在浏览器中看到下面的效果

|  |
| --- |
|  |

## 16.然后我们用鼠标在立方体说明转动，效果如下

|  |
| --- |
|  |
|  |
|  |

## 窗口左上角也会显示帧率

|  |
| --- |
|  |

### 然后窗口也是响应式的

# 这一节就学习到这里，这一节的主要代码：

## 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';  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.定义环境光和方向光          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 = 48          //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，其实是平行光          this.directionalLight = new THREE.DirectionalLight(0xffffff,1)          //设置方向光的位置          this.directionalLight.position.set(0,32,64)          //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)      }  } |

## App.jsx

|  |
| --- |
| import { useEffect } from 'react'  import './App.css'  import \* as THREE from 'three'  import ThreeApp from './lib/threeapp'  function App() {    useEffect(() => {      //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()      //创建一个立方体并且添加到创建      let boxGeo = new THREE.BoxGeometry(16, 16, 16)      let material = new THREE.MeshNormalMaterial()      let cube = new THREE.Mesh(boxGeo, material)      threeApp.scene.add(cube)    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |

## index.html

|  |
| --- |
| <!doctype html>  <html lang="en">  <head>    <meta charset="UTF-8" />    <link rel="icon" type="image/svg+xml" href="/vite.svg" />    <meta name="viewport" content="width=device-width, initial-scale=1.0" />    <title>Vite + React</title>    <script src="https://cdn.tailwindcss.com"></script>  </head>  <body>    <div id="root"></div>    <script type="module" src="/src/main.jsx"></script>  </body>  </html> |