## 1.所谓的纹理，其实就是图片，加载纹理需要先创建纹理加载器textureLoader对象然后再调用它的load方法，开源把创建和调用代码一起写，如

|  |
| --- |
| let bgTexture = new THREE.TextureLoader().load('../assets/space.jpeg') |

## 2.开源设置纹理的处理方式

|  |
| --- |
| bgTexture.wrapS = THREE.RepeatWrapping   bgTexture.wrapT = THREE.RepeatWrapping |

## 3.可以给场景设置背景纹理

|  |
| --- |
| threeApp.scene.background = bgTexture |

# 完整实例1

|  |
| --- |
|  |

## 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.spotLight = 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.y = 9          this.camera.position.z = 36 //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)          //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(0xfafafa,0.5)          //设置方向光的位置          this.directionalLight.position.set(0,10,10) //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/App.jsx

|  |
| --- |
| import { useEffect } from 'react'  import './App.css'  import \* as THREE from 'three'  import { GUI } from 'three/examples/jsm/libs/lil-gui.module.min'  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()    //给场景添加背景纹理    let bgTexture = new THREE.TextureLoader().load('../assets/space.jpeg')    bgTexture.wrapS = THREE.RepeatWrapping    bgTexture.wrapT = THREE.RepeatWrapping    threeApp.scene.background = bgTexture  }  function App() {    useEffect(() => {      main1()    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |

### 效果

|  |
| --- |
|  |

# 实例2.创建一个立方体并且给他添加砖墙贴图

### 1）为了方便代码复用，我们把加载纹理的代码封装到一个函数中

|  |
| --- |
|  |

### 2）threeapp的代码和上面一样，我们这里只修改main1函数的代码，如下

|  |
| --- |
|  |

#### 注意：可以添加纹理的材质是标准材质：MeshStandardMaterial

## App.jsx的完整代码如下

|  |
| --- |
| import { useEffect } from 'react'  import './App.css'  import \* as THREE from 'three'  import { GUI } from 'three/examples/jsm/libs/lil-gui.module.min'  import ThreeApp from './lib/threeapp'  function loadTextures(app) {    let brickTexture = new THREE.TextureLoader().load('../assets/brick.jpeg')    let crateTexture = new THREE.TextureLoader().load('../assets/crate.jpeg')    let earthTexture = new THREE.TextureLoader().load('../assets/earth.jpeg')    let uvTexture = new THREE.TextureLoader().load('../assets/uv.png')    app.brickTexture = brickTexture    app.crateTexture = crateTexture    app.earthTexture = earthTexture    app.uvTexture = uvTexture  }  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()    //给场景添加背景纹理    loadTextures(threeApp)    //加载其他纹理图片    //创建一个立方体，并且贴图    let box1Geo = new THREE.BoxGeometry(15, 15, 15)    let box1Mat = new THREE.MeshStandardMaterial({      map: threeApp.brickTexture    })    let box1 = new THREE.Mesh(box1Geo, box1Mat)    box1.rotation.y = Math.PI / 4    box1.rotation.x = Math.PI / 8    threeApp.scene.add(box1)  }  function App() {    useEffect(() => {      main1()    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |

### 效果：

|  |
| --- |
|  |

# 实例3.加载其他纹理，主要是修改App.jsx,我们把代码封装在main2函数中

|  |
| --- |
| import { useEffect } from 'react'  import './App.css'  import \* as THREE from 'three'  import { GUI } from 'three/examples/jsm/libs/lil-gui.module.min'  import ThreeApp from './lib/threeapp'  function loadTextures(app) {    let brickTexture = new THREE.TextureLoader().load('../assets/brick.jpeg')    let crateTexture = new THREE.TextureLoader().load('../assets/crate.png')    let earthTexture = new THREE.TextureLoader().load('../assets/earth.jpeg')    let uvTexture = new THREE.TextureLoader().load('../assets/uv.png')    let uvTexture2 = new THREE.TextureLoader().load('../assets/uv.jpeg')    app.brickTexture = brickTexture    app.crateTexture = crateTexture    app.earthTexture = earthTexture    app.uvTexture = uvTexture    app.uvTexture2 = uvTexture2  }  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()    //给场景添加背景纹理    loadTextures(threeApp)    //加载其他纹理图片    //创建一个立方体，并且贴图    let box1Geo = new THREE.BoxGeometry(15, 15, 15)    let box1Mat = new THREE.MeshStandardMaterial({      map: threeApp.brickTexture    })    let box1 = new THREE.Mesh(box1Geo, box1Mat)    box1.rotation.y = Math.PI / 4    box1.rotation.x = Math.PI / 8    threeApp.scene.add(box1)  }  function main2() {    //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()    //给场景添加背景纹理    loadTextures(threeApp)    //加载其他纹理图片    //创建一个立方体，并且贴图    let box1Geo = new THREE.BoxGeometry(7, 7, 7)    let box1Mat = new THREE.MeshStandardMaterial({      map: threeApp.crateTexture    })    let box1 = new THREE.Mesh(box1Geo, box1Mat)    box1.rotation.y = Math.PI / 4    box1.rotation.x = Math.PI / 8    box1.position.x = -9;    box1.position.y = -5;    threeApp.scene.add(box1)    let box2Geo = new THREE.BoxGeometry(7, 7, 7)    let box2Mat = new THREE.MeshStandardMaterial({      map: threeApp.uvTexture2    })    let box2 = new THREE.Mesh(box2Geo, box2Mat)    box2.position.x = 9;    box2.position.y = -3;    threeApp.scene.add(box2)    //球体    let sp1Geo = new THREE.SphereGeometry(4)    let sp1Mat = new THREE.MeshStandardMaterial({      map: threeApp.earthTexture    })    let sphere = new THREE.Mesh(sp1Geo, sp1Mat)    sphere.position.x = -9    sphere.position.y = 5    threeApp.scene.add(sphere)    let sp2Geo = new THREE.SphereGeometry(4)    let sp2Mat = new THREE.MeshStandardMaterial({      map: threeApp.brickTexture    })    let sphere2 = new THREE.Mesh(sp2Geo, sp2Mat)    sphere2.position.x = 9    sphere2.position.y = 5    threeApp.scene.add(sphere2)  }  function App() {    useEffect(() => {      // main1()      main2()    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |

### 效果：

|  |
| --- |
|  |

# 扩展，其实可以添加光照效果，有空慢慢练习