# 老师的教程使用dat.gui,其实threejs自带lil-gui,我来使用lil-gui看看

## 准备工作：把lesson2项目复制一份改名lesson3

|  |
| --- |
|  |

## 1.修改src/lib/threeapp.js.把ThreeApp2改为ThreeApp3,然后把代码修改如下

|  |
| --- |
| 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 ThreeApp3{      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.z = 48          this.camera.position.z = 96 //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)          //创建聚光灯          this.spotLight = new THREE.SpotLight(0xffffff,1)          this.spotLight.castShadow = true //显示阴影          //设置方向光的位置          this.spotLight.position.set(0,64,32) //位置也不一样          //添加方向光到场景中          this.scene.add(this.spotLight)          //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)      }  } |

## 2.修改App.jsx,我们这里打算使用4给函数，先来看part1函数，就是老师的part1+part2，代码如下

|  |
| --- |
| function part1() {    //创建three应用程序对象    let threeApp = new ThreeApp3("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let boxGeometry = new THREE.BoxGeometry(24, 24, 24)    let boxMaterial = new THREE.MeshPhongMaterial({ color: 0xff0000 })    let box = new THREE.Mesh(boxGeometry, boxMaterial)    threeApp.scene.add(box)    //创建GUI对象    let gui = new GUI()    //把box的一些属性添加到gui中    gui.add(box.rotation, "x", 0, Math.PI).name('Rotate X Axis');    gui.add(box.rotation, "y", 0, Math.PI).name('Rotate Y Axis');    gui.add(box.rotation, "z", 0, Math.PI).name('Rotate Z Axis');    gui.add(box.scale, "x", 0, 2).name('Scale X Axis');    gui.add(box.scale, "y", 0, 2).name('Scale Y Axis');    gui.add(box.scale, "z", 0, 2).name('Scale Z Axis');    () => {      return gui.destroy()    }  } |

### 效果：

|  |
| --- |
|  |

## 3.现在，我们把part1的调用注释掉，我们来写part2函数。代码如下，其实就是在part1的基础上添加一些代码

|  |
| --- |
| function part2() {    //创建three应用程序对象    let threeApp = new ThreeApp3("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let boxGeometry = new THREE.BoxGeometry(24, 24, 24)    let boxMaterial = new THREE.MeshPhongMaterial({ color: 0xff0000 })    let box = new THREE.Mesh(boxGeometry, boxMaterial)    threeApp.scene.add(box)    //创建GUI对象    let gui = new GUI()    //把box的一些属性添加到gui中    gui.add(box.rotation, "x", 0, Math.PI).name('Rotate X Axis');    gui.add(box.rotation, "y", 0, Math.PI).name('Rotate Y Axis');    gui.add(box.rotation, "z", 0, Math.PI).name('Rotate Z Axis');    gui.add(box.scale, "x", 0, 2).name('Scale X Axis');    gui.add(box.scale, "y", 0, 2).name('Scale Y Axis');    gui.add(box.scale, "z", 0, 2).name('Scale Z Axis');    //修改立方体的颜色    let materialParams = {      boxColor: box.material.color.getHex()    };    //打开和关闭线框功能    gui.add(box.material, "wireframe");    gui.addColor(materialParams, 'boxColor').onChange((value) => {      box.material.color.set(value)    });    () => {      return gui.destroy()    }  } |

## 4.在App.jsx里面调用part2函数

|  |
| --- |
|  |

### 效果如下：

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

## 5，把这个函数的调用注释了，我们来写part3函数，用另外一种方法来使用GUI，就是文件夹方式

|  |
| --- |
| function part3() {    //创建three应用程序对象    let threeApp = new ThreeApp3("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let boxGeometry = new THREE.BoxGeometry(24, 24, 24)    let boxMaterial = new THREE.MeshPhongMaterial({ color: 0xff0000 })    let box = new THREE.Mesh(boxGeometry, boxMaterial)    threeApp.scene.add(box)    //创建GUI对象    let gui = new GUI()    //先创建一个组：geoFolder    let geoFolder = gui.addFolder("Mesh Geometry")    geoFolder.open() //创建几何文件夹    //添加旋转文件夹    let rotationFolder = geoFolder.addFolder("Rotation")    rotationFolder.add(box.rotation, "x", 0, Math.PI).name('Rotate X Axis');    rotationFolder.add(box.rotation, "y", 0, Math.PI).name('Rotate Y Axis');    rotationFolder.add(box.rotation, "z", 0, Math.PI).name('Rotate Z Axis');    //添加缩放文件夹    let scaleFolder = geoFolder.addFolder('Scale')    scaleFolder.add(box.scale, "x", 0, 2).name('Scale X Axis');    scaleFolder.add(box.scale, "y", 0, 2).name('Scale Y Axis');    scaleFolder.add(box.scale, "z", 0, 2).name('Scale Z Axis');    scaleFolder.open()    //添加材质文件夹    let materialFolder = geoFolder.addFolder("Material")    //修改立方体的颜色    let materialParams = {      boxColor: box.material.color.getHex()    };    //打开和关闭线框功能    materialFolder.add(box.material, "wireframe");    materialFolder.addColor(materialParams, 'boxColor').onChange((value) => {      box.material.color.set(value)    });    () => {      return gui.destroy()    }  } |

## 6.然后在App.jsx中调用part3函数，

|  |
| --- |
|  |

### 效果：可以把各种属性分门别类，比较有规律，是比较推荐的用法

|  |
| --- |
|  |

## 7.把part3的调用代码注释了，还有part4，它是在part3的基础上添加自定义函数功能

|  |
| --- |
| function part4() {    //创建three应用程序对象    let threeApp = new ThreeApp3("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let boxGeometry = new THREE.BoxGeometry(24, 24, 24)    let boxMaterial = new THREE.MeshPhongMaterial({ color: 0xff0000 })    let box = new THREE.Mesh(boxGeometry, boxMaterial)    threeApp.scene.add(box)    //创建GUI对象    let gui = new GUI()    //先创建一个组：geoFolder    let geoFolder = gui.addFolder("Mesh Geometry")    geoFolder.open() //创建几何文件夹    //添加旋转文件夹    let rotationFolder = geoFolder.addFolder("Rotation")    rotationFolder.add(box.rotation, "x", 0, Math.PI).name('Rotate X Axis');    rotationFolder.add(box.rotation, "y", 0, Math.PI).name('Rotate Y Axis');    rotationFolder.add(box.rotation, "z", 0, Math.PI).name('Rotate Z Axis');    //添加缩放文件夹    let scaleFolder = geoFolder.addFolder('Scale')    scaleFolder.add(box.scale, "x", 0, 2).name('Scale X Axis');    scaleFolder.add(box.scale, "y", 0, 2).name('Scale Y Axis');    scaleFolder.add(box.scale, "z", 0, 2).name('Scale Z Axis');    scaleFolder.open()    //添加材质文件夹    let materialFolder = geoFolder.addFolder("Material")    //修改立方体的颜色    let materialParams = {      boxColor: box.material.color.getHex()    };    //打开和关闭线框功能    materialFolder.add(box.material, "wireframe");    materialFolder.addColor(materialParams, 'boxColor').onChange((value) => {      box.material.color.set(value)    });    //添加自定义函数文件夹    let customFolder = geoFolder.addFolder("Custom Function")    customFolder.open()    let customParam = {      customFunction: false    }    customFolder.add(customParam, 'customFunction')      .name("print'Hello'")      .onChange((value) => {        if (value === true) {          alert('Hello')        }      });    () => {      return gui.destroy()    }  } |

## 8.然后在App.jsx中都有part4函数

|  |
| --- |
|  |

### 效果：会出现一个print‘Hello’复选框，添加复选框会弹出一条Hello消息，然后复选框会选中

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

# 下一节学习完成，这一节的主要代码如下

## 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 ThreeApp3{      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.z = 48          this.camera.position.z = 96 //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)          //创建聚光灯          this.spotLight = new THREE.SpotLight(0xffffff,1)          this.spotLight.castShadow = true //显示阴影          //设置方向光的位置          this.spotLight.position.set(0,64,32) //位置也不一样          //添加方向光到场景中          this.scene.add(this.spotLight)          //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 { TeapotGeometry } from 'three/examples/jsm/geometries/TeapotGeometry'  import { RoundedBoxGeometry } from 'three/examples/jsm/geometries/RoundedBoxGeometry';  import { GUI } from 'three/examples/jsm/libs/lil-gui.module.min'  import ThreeApp3 from './lib/threeapp'  function part1() {    //创建three应用程序对象    let threeApp = new ThreeApp3("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let boxGeometry = new THREE.BoxGeometry(24, 24, 24)    let boxMaterial = new THREE.MeshPhongMaterial({ color: 0xff0000 })    let box = new THREE.Mesh(boxGeometry, boxMaterial)    threeApp.scene.add(box)    //创建GUI对象    let gui = new GUI()    //把box的一些属性添加到gui中    gui.add(box.rotation, "x", 0, Math.PI).name('Rotate X Axis');    gui.add(box.rotation, "y", 0, Math.PI).name('Rotate Y Axis');    gui.add(box.rotation, "z", 0, Math.PI).name('Rotate Z Axis');    gui.add(box.scale, "x", 0, 2).name('Scale X Axis');    gui.add(box.scale, "y", 0, 2).name('Scale Y Axis');    gui.add(box.scale, "z", 0, 2).name('Scale Z Axis');    () => {      return gui.destroy()    }  }  function part2() {    //创建three应用程序对象    let threeApp = new ThreeApp3("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let boxGeometry = new THREE.BoxGeometry(24, 24, 24)    let boxMaterial = new THREE.MeshPhongMaterial({ color: 0xff0000 })    let box = new THREE.Mesh(boxGeometry, boxMaterial)    threeApp.scene.add(box)    //创建GUI对象    let gui = new GUI()    //把box的一些属性添加到gui中    gui.add(box.rotation, "x", 0, Math.PI).name('Rotate X Axis');    gui.add(box.rotation, "y", 0, Math.PI).name('Rotate Y Axis');    gui.add(box.rotation, "z", 0, Math.PI).name('Rotate Z Axis');    gui.add(box.scale, "x", 0, 2).name('Scale X Axis');    gui.add(box.scale, "y", 0, 2).name('Scale Y Axis');    gui.add(box.scale, "z", 0, 2).name('Scale Z Axis');    //修改立方体的颜色    let materialParams = {      boxColor: box.material.color.getHex()    };    //打开和关闭线框功能    gui.add(box.material, "wireframe");    gui.addColor(materialParams, 'boxColor').onChange((value) => {      box.material.color.set(value)    });    () => {      return gui.destroy()    }  }  function part3() {    //创建three应用程序对象    let threeApp = new ThreeApp3("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let boxGeometry = new THREE.BoxGeometry(24, 24, 24)    let boxMaterial = new THREE.MeshPhongMaterial({ color: 0xff0000 })    let box = new THREE.Mesh(boxGeometry, boxMaterial)    threeApp.scene.add(box)    //创建GUI对象    let gui = new GUI()    //先创建一个组：geoFolder    let geoFolder = gui.addFolder("Mesh Geometry")    geoFolder.open() //创建几何文件夹    //添加旋转文件夹    let rotationFolder = geoFolder.addFolder("Rotation")    rotationFolder.add(box.rotation, "x", 0, Math.PI).name('Rotate X Axis');    rotationFolder.add(box.rotation, "y", 0, Math.PI).name('Rotate Y Axis');    rotationFolder.add(box.rotation, "z", 0, Math.PI).name('Rotate Z Axis');    //添加缩放文件夹    let scaleFolder = geoFolder.addFolder('Scale')    scaleFolder.add(box.scale, "x", 0, 2).name('Scale X Axis');    scaleFolder.add(box.scale, "y", 0, 2).name('Scale Y Axis');    scaleFolder.add(box.scale, "z", 0, 2).name('Scale Z Axis');    scaleFolder.open()    //添加材质文件夹    let materialFolder = geoFolder.addFolder("Material")    //修改立方体的颜色    let materialParams = {      boxColor: box.material.color.getHex()    };    //打开和关闭线框功能    materialFolder.add(box.material, "wireframe");    materialFolder.addColor(materialParams, 'boxColor').onChange((value) => {      box.material.color.set(value)    });    () => {      return gui.destroy()    }  }  function part4() {    //创建three应用程序对象    let threeApp = new ThreeApp3("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let boxGeometry = new THREE.BoxGeometry(24, 24, 24)    let boxMaterial = new THREE.MeshPhongMaterial({ color: 0xff0000 })    let box = new THREE.Mesh(boxGeometry, boxMaterial)    threeApp.scene.add(box)    //创建GUI对象    let gui = new GUI()    //先创建一个组：geoFolder    let geoFolder = gui.addFolder("Mesh Geometry")    geoFolder.open() //创建几何文件夹    //添加旋转文件夹    let rotationFolder = geoFolder.addFolder("Rotation")    rotationFolder.add(box.rotation, "x", 0, Math.PI).name('Rotate X Axis');    rotationFolder.add(box.rotation, "y", 0, Math.PI).name('Rotate Y Axis');    rotationFolder.add(box.rotation, "z", 0, Math.PI).name('Rotate Z Axis');    //添加缩放文件夹    let scaleFolder = geoFolder.addFolder('Scale')    scaleFolder.add(box.scale, "x", 0, 2).name('Scale X Axis');    scaleFolder.add(box.scale, "y", 0, 2).name('Scale Y Axis');    scaleFolder.add(box.scale, "z", 0, 2).name('Scale Z Axis');    scaleFolder.open()    //添加材质文件夹    let materialFolder = geoFolder.addFolder("Material")    //修改立方体的颜色    let materialParams = {      boxColor: box.material.color.getHex()    };    //打开和关闭线框功能    materialFolder.add(box.material, "wireframe");    materialFolder.addColor(materialParams, 'boxColor').onChange((value) => {      box.material.color.set(value)    });    //添加自定义函数文件夹    let customFolder = geoFolder.addFolder("Custom Function")    customFolder.open()    let customParam = {      customFunction: false    }    customFolder.add(customParam, 'customFunction')      .name("print'Hello'")      .onChange((value) => {        if (value === true) {          alert('Hello')        }      });    () => {      return gui.destroy()    }  }  function App() {    useEffect(() => {      // part1()      // part2()      // part3()      part4()    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |