# 在我们安装好的库里面有下面的几何体，需要按需导入

|  |
| --- |
|  |
| three/examples/jsm/geometries/BoxLineGeometrythree/examples/jsm/geometries/ConvexGeometrythree/examples/jsm/geometries/DecalGeometrythree/examples/jsm/geometries/InstancedPointsGeometrythree/examples/jsm/geometries/ParametricGeometriesthree/examples/jsm/geometries/ParametricGeometrythree/examples/jsm/geometries/RoundedBoxGeometrythree/examples/jsm/geometries/TeapotGeometrythree/examples/jsm/geometries/TextGeometry |

## 1.我们把lesson1项目复制一份改名lesson2，然后修改一下包名，注意包名不要有数字。

|  |
| --- |
|  |

## 2.我们把这一次的学习分为三个部分，第一部分是基础的几何图形绘制，第二部分是shader效果。原来threejs在创建材质的时候可以传递shader程序。第三部分也是几何体。所以我们会使用函数part1，part2，part3来表示，下面是part1的功能

|  |
| --- |
|  |

## 3.然后在useEffect钩子函数里面调用part1函数

|  |
| --- |
|  |

### 效果：

|  |
| --- |
|  |

## 4.然后我们来在part2里面添加着色器程序

|  |
| --- |
| function part2() {    //创建three应用程序对象    let threeApp = new ThreeApp2("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    //定义着色器程序    function vertexShader() {      return `        varying float z;        uniform float u\_time;        void main(){          z = (cos(position.y+u\_time)+sin(position.x+u\_time))/4.0;          gl\_Position = projectionMatrix \* modelViewMatrix \* vec4(position.x,position.y,z+position.z,1.0);        }      `;    }    function fragmentShader() {      return `        void main(){          gl\_FragColor = vec4(1.0,0.0,0.0,1.0);        }      `;    }    let bGeo = new THREE.BoxGeometry(1, 1, 1, 1, 1, 1)    let bMat = new THREE.ShaderMaterial({      uniforms: threeApp.uniforms,      fragmentShader: fragmentShader(),      vertexShader: vertexShader(),      wireframe: true    })    let box = new THREE.Mesh(bGeo, bMat)    box.rotation.x = Math.PI / 2    threeApp.scene.add(box)  } |

## 5.然后在App.jsx里面调用part2函数，

|  |
| --- |
|  |

### 效果：会得到一个红色的会上下倾斜摆动的线框立方体

|  |
| --- |
|  |
|  |

## 6.把part2函数的调用注释掉，然后我们来写part3函数，代码如下

|  |
| --- |
| function part3() {    //创建three应用程序对象    let threeApp = new ThreeApp2("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let teapotGeo = new TeapotGeometry(0.5, 8)    let teapotMaterial = new THREE.MeshNormalMaterial({ wireframe: true })    let teapot = new THREE.Mesh(teapotGeo, teapotMaterial)    teapot.position.x = 3    threeApp.scene.add(teapot)    //DecalGeometry    let roundedBoxGeo = new RoundedBoxGeometry(1, 1, 1, 4, 0.1) //这个是threejs内置的几何体，不用额外导入    let roundedBoxMaterial = new THREE.MeshNormalMaterial({ wireframe: true })    let roundedBox = new THREE.Mesh(roundedBoxGeo, roundedBoxMaterial)    threeApp.scene.add(roundedBox)    function animate() {      window.requestAnimationFrame(animate)      roundedBox.rotation.x += 0.01      roundedBox.rotation.y += 0.01      teapot.rotateY(0.01)      teapot.rotateZ(0.05)    }    animate()  } |

## 7.在App.jsx里面都有part3函数

|  |
| --- |
|  |

### 效果：会得到一个旋转的削边立方体和一个旋转茶壶

|  |
| --- |
|  |
|  |

# 总结，在threejs里面也是可以使用着色器的，如果需要使用着色器，创建物体的时候需要使用ShaderMaterial材质，当然，还需要着色器程序源码字符串。

|  |
| --- |
|  |
|  |

# 这一节有更新的代码如下

## 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 ThreeApp2{      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 = 8 //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)          //定义一个uniform变量传递给glsl着色器          this.uniforms = {              u\_time:{type:'float',value:1.0},              colorB:{type:'vec3', value: new THREE.Color(0xfff000)},              colorA:{type:'vec3', value: new THREE.Color(0xffffff)},          }      }        //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.uniforms.u\_time.value += this.clock.getDelta()          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 ThreeApp2 from './lib/threeapp'  function part1() {    //创建three应用程序对象    let threeApp = new ThreeApp2("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let teapotGeo = new TeapotGeometry(0.5, 8)    let teapotMaterial = new THREE.MeshNormalMaterial({ wireframe: true })    let teapot = new THREE.Mesh(teapotGeo, teapotMaterial)    teapot.position.x = 3    threeApp.scene.add(teapot)    //DecalGeometry    let torusGeo = new THREE.TorusGeometry() //这个是threejs内置的几何体，不用额外导入    let torusMaterial = new THREE.MeshNormalMaterial({ wireframe: true })    let torus = new THREE.Mesh(torusGeo, torusMaterial)    torus.rotation.x = Math.PI / 5    torus.rotation.y = Math.PI / 6    threeApp.scene.add(torus)  }  function part2() {    //创建three应用程序对象    let threeApp = new ThreeApp2("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    //定义着色器程序    function vertexShader() {      return `        varying float z;        uniform float u\_time;        void main(){          z = (cos(position.y+u\_time)+sin(position.x+u\_time))/4.0;          gl\_Position = projectionMatrix \* modelViewMatrix \* vec4(position.x,position.y,z+position.z,1.0);        }      `;    }    function fragmentShader() {      return `        void main(){          gl\_FragColor = vec4(1.0,0.0,0.0,1.0);        }      `;    }    let bGeo = new THREE.BoxGeometry(1, 1, 1, 1, 1, 1)    let bMat = new THREE.ShaderMaterial({      uniforms: threeApp.uniforms,      fragmentShader: fragmentShader(),      vertexShader: vertexShader(),      wireframe: true    })    let box = new THREE.Mesh(bGeo, bMat)    box.rotation.x = Math.PI / 2    threeApp.scene.add(box)  }  function part3() {    //创建three应用程序对象    let threeApp = new ThreeApp2("myThreeJSCanvas")    //初始化场景    threeApp.initApp()    //调用动画渲染效果    threeApp.animate()    let teapotGeo = new TeapotGeometry(0.5, 8)    let teapotMaterial = new THREE.MeshNormalMaterial({ wireframe: true })    let teapot = new THREE.Mesh(teapotGeo, teapotMaterial)    teapot.position.x = 3    threeApp.scene.add(teapot)    //DecalGeometry    let roundedBoxGeo = new RoundedBoxGeometry(1, 1, 1, 4, 0.1) //这个是threejs内置的几何体，不用额外导入    let roundedBoxMaterial = new THREE.MeshNormalMaterial({ wireframe: true })    let roundedBox = new THREE.Mesh(roundedBoxGeo, roundedBoxMaterial)    threeApp.scene.add(roundedBox)    function animate() {      window.requestAnimationFrame(animate)      roundedBox.rotation.x += 0.01      roundedBox.rotation.y += 0.01      teapot.rotateY(0.01)      teapot.rotateZ(0.05)    }    animate()  }  function App() {    useEffect(() => {      // part1()      // part2()      part3()    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |