# threejs提供了渲染3D物体的功能，但是他没有提供物理相关功能，如果需要使用必须安装其他库如Ammo.js或者Connonjs。

|  |
| --- |
|  |

# 其实我们更推荐使用cannon-es，他是cannon.js的改进版本，他由react-three-fiber团队开发

## 安装cannon-es：npm install cannon-es

## 导入：import \* as CANNON from ‘cannon-es‘

## 还需要安装调试器cannon-es-debugger： npm install cannon-es-debugger

## 导入：import CannonDebugger from ‘cannon-es-debugger ‘

## 调用：

|  |
| --- |
|  |
|  |

# 练习项目

|  |
| --- |
|  |

## 1.安装cannon-es：打开终端定位到项目根目录，输入命令：npm install cannon-es

|  |
| --- |
|  |

## 2.安装cannon-es-debugger，在终端输入命令：npm install cannon-es-debugger

|  |
| --- |
|  |

## 3.我们用三个函数来学习cannon-es

### appfunc1.js

|  |
| --- |
| import \* as THREE from 'three'  import ThreeApp from '../lib/threeapp2'  // import \* as TWEEN from '@tweenjs/tween.js' //导入tweenjs  import \* as CANNON from 'cannon-es'  import CannonDebugger from 'cannon-es-debugger'  export function appFunc1(canvas){      let threeApp = new ThreeApp(canvas)      threeApp.initApp()      threeApp.animate()        let axesHelper = new THREE.AxesHelper(8)      threeApp.scene.add(axesHelper)        //利用CANNON库创建物理世界      let physicsWorld = new CANNON.World({        gravity: new CANNON.Vec3(0,-9.82,0)      })      // create a ground body with a static plane      let groundBody = new CANNON.Body({          type: CANNON.Body.STATIC,          shape: new CANNON.Plane()      })      // create a ground body with a static plane      groundBody.quaternion.setFromEuler(-Math.PI/2,0,0)      physicsWorld.addBody(groundBody)      //创建一个球体并且添加到物理世界      let radius=1      let sphere = new CANNON.Body({          mass:5,          shape: new CANNON.Sphere(radius)      })      //设置球体的位置      sphere.position.set(0,7,0)      physicsWorld.addBody(sphere)      // bind cannon debugger to the three.js scene + physics world      let cannonDebugger = new CannonDebugger(threeApp.scene,physicsWorld,{      })      const animate = () => {        physicsWorld.fixedStep()        cannonDebugger.update()        window.requestAnimationFrame(animate);      };      animate();  } |

### 效果：有一个线框求从y坐标轴上面下降到原点

### appfunc2.js

|  |
| --- |
| let threeApp = new ThreeApp(canvas)    threeApp.initApp()    threeApp.animate()      let axesHelper = new THREE.AxesHelper(8)    threeApp.scene.add(axesHelper)      //利用CANNON库创建物理世界    let physicsWorld = new CANNON.World({      gravity: new CANNON.Vec3(0,-9.82,0)    })    // create a ground body with a static plane    let groundBody = new CANNON.Body({        type: CANNON.Body.STATIC,        shape: new CANNON.Plane()    })    // create a ground body with a static plane    groundBody.quaternion.setFromEuler(-Math.PI/2,0,0)    physicsWorld.addBody(groundBody)    //创建一个球体并且添加到物理世界    let radius=1    let sphere = new CANNON.Body({        mass:5,        shape: new CANNON.Sphere(radius)    })    //设置球体的位置    sphere.position.set(0,7,0)    physicsWorld.addBody(sphere)    // bind cannon debugger to the three.js scene + physics world    let cannonDebugger = new CannonDebugger(threeApp.scene,physicsWorld,{    })    // combine the three.js game world with the physics world    let geo = new THREE.SphereGeometry(radius)    let spMat = new THREE.MeshNormalMaterial()    let sp = new THREE.Mesh(geo,spMat)    threeApp.scene.add(sp)    let animate=()=>{      physicsWorld.fixedStep()      cannonDebugger.update()      sp.position.copy(sphere.position)      sp.quaternion.copy(sphere.quaternion)      window.requestAnimationFrame(animate)    }    animate() |

### 效果：有一个线框求和一个彩色球体的组合体从y坐标轴上面下降到原点

### appfunc3.js

|  |
| --- |
| import \* as THREE from 'three'  import ThreeApp from '../lib/threeapp2'  import \* as CANNON from 'cannon-es'  import CannonDebugger from 'cannon-es-debugger'  export function appFunc3(canvas){    let threeApp = new ThreeApp(canvas)    threeApp.initApp()    threeApp.animate()      let axesHelper = new THREE.AxesHelper(8)    threeApp.scene.add(axesHelper)      //利用CANNON库创建物理世界    let physicsWorld = new CANNON.World({      gravity: new CANNON.Vec3(0,-9.82,0)    })    // create a ground body with a static plane    let groundBody = new CANNON.Body({        type: CANNON.Body.STATIC,        shape: new CANNON.Plane()    })    // create a ground body with a static plane    groundBody.quaternion.setFromEuler(-Math.PI/2,0,0)    physicsWorld.addBody(groundBody)    //创建一个球体并且添加到物理世界    let radius=1    let sphere = new CANNON.Body({        mass:5,        shape: new CANNON.Sphere(radius)    })    //设置球体的位置    sphere.position.set(0,7,0)    physicsWorld.addBody(sphere)    // bind cannon debugger to the three.js scene + physics world    let cannonDebugger = new CannonDebugger(threeApp.scene,physicsWorld,{    })    // combine the three.js game world with the physics world    let geo = new THREE.SphereGeometry(radius)    let spMat = new THREE.MeshNormalMaterial()    let sp = new THREE.Mesh(geo,spMat)    threeApp.scene.add(sp)    //添加一个立方体      let cannonBox = new CANNON.Body({         mass:5,         shape: new CANNON.Box(new CANNON.Vec3(1,1,1))      })      cannonBox.position.set(0,10,0)      physicsWorld.addBody(cannonBox)      const boxGeometry = new THREE.BoxGeometry(2,2,2);      const boxMaterial = new THREE.MeshNormalMaterial();      const box = new THREE.Mesh(boxGeometry, boxMaterial);      threeApp.scene.add(box);    let animate=()=>{      physicsWorld.fixedStep()      cannonDebugger.update()      box.position.copy(cannonBox.position)      box.quaternion.copy(cannonBox.quaternion)      sp.position.copy(sphere.position)      sp.quaternion.copy(sphere.quaternion)      window.requestAnimationFrame(animate)    }    animate()  } |

### 效果：一个球体和一个立方体一起用y轴上面下降到原点