# 这一节学习用cannon-es库来创建一辆刚体车辆

## 项目结构

|  |
| --- |
|  |

## 注意，需要先安装cannon-es库和cannon-es-debugger库

## 1.在appfunc1.js中我们显示一辆线框刚体车，appfunc1.代码如下

|  |
| --- |
| 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CannonDebugger实例对象      const cannonDebugger = new CannonDebugger(threeApp.scene, physicsWorld);      //创建一个车体      let carBody = new CANNON.Body({        mass:5,        position:new CANNON.Vec3(0,6,0),        shape:new CANNON.Box(new CANNON.Vec3(4,0.5,2))      })      //利用车体创建刚体车辆      let vehicle = new CANNON.RigidVehicle({         chassisBody:carBody      })      //创建轮子      let mass = 1      let axisWidth = 5      let wheelShape = new CANNON.Sphere(1)      let wheelMaterial = new CANNON.Material('wheel')      let down = new CANNON.Vec3(0,-1,0)      let wheelBody1 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody1.addShape(wheelShape)      wheelBody1.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody1,        position:new CANNON.Vec3(-2, 0, axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })      let wheelBody2 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody2.addShape(wheelShape)      wheelBody2.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody2,        position:new CANNON.Vec3(-2, 0, -axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })      let wheelBody3 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody3.addShape(wheelShape)      wheelBody3.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody3,        position:new CANNON.Vec3(2, 0, axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })      let wheelBody4 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody4.addShape(wheelShape)      wheelBody4.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody4,        position:new CANNON.Vec3(2, 0, -axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })      vehicle.addToWorld(physicsWorld)      const animate = () => {        physicsWorld.fixedStep();        cannonDebugger.update();        window.requestAnimationFrame(animate);      };      animate();  } |

## 2.在App.jsx中调用appfunc1

|  |
| --- |
| import { useEffect } from 'react'  import './App.css'  import { appFunc1 } from './appFuncs/appfunc1'  import { appFunc2 } from './appFuncs/appfunc2'  import { appFunc3 } from './appFuncs/appfunc3'    function App() {    useEffect(() => {      let canvas = document.getElementById("myThreeJSCanvas") //在外面创建canvas      appFunc1(canvas)      // appFunc2(canvas)      // appFunc3(canvas)    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |

### 效果：

|  |
| --- |
|  |

## 3.然后我们编写appfunc2.js代码，在里面添加按键设计监听功能，实现用方向键或者a，d，w，s键扩展车辆的运动方向

|  |
| --- |
| 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 appFunc2(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CannonDebugger实例对象      const cannonDebugger = new CannonDebugger(threeApp.scene, physicsWorld);      //创建一个车体      let carBody = new CANNON.Body({        mass:5,        position:new CANNON.Vec3(0,6,0),        shape:new CANNON.Box(new CANNON.Vec3(4,0.5,2))      })      //利用车体创建刚体车辆      let vehicle = new CANNON.RigidVehicle({         chassisBody:carBody      })      //创建轮子      let mass = 1      let axisWidth = 5      let wheelShape = new CANNON.Sphere(1)      let wheelMaterial = new CANNON.Material('wheel')      let down = new CANNON.Vec3(0,-1,0)      let wheelBody1 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody1.addShape(wheelShape)      wheelBody1.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody1,        position:new CANNON.Vec3(-2, 0, axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })      let wheelBody2 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody2.addShape(wheelShape)      wheelBody2.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody2,        position:new CANNON.Vec3(-2, 0, -axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })      let wheelBody3 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody3.addShape(wheelShape)      wheelBody3.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody3,        position:new CANNON.Vec3(2, 0, axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })      let wheelBody4 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody4.addShape(wheelShape)      wheelBody4.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody4,        position:new CANNON.Vec3(2, 0, -axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })        vehicle.addToWorld(physicsWorld)      // move car based on user input      document.addEventListener('keydown',(event)=>{        let maxSteerVal = Math.PI/8        let maxForce = 10        switch (event.key) {          case 'w':          case 'ArrowUp':            vehicle.setWheelForce(maxForce,0)            vehicle.setWheelForce(maxForce,1)              break;          case 's':          case 'ArrowDown':            vehicle.setWheelForce(-maxForce/2,0)            vehicle.setWheelForce(-maxForce/2,1)            break;          case 'a':          case 'ArrowLeft':            vehicle.setSteeringValue(maxSteerVal,0)            vehicle.setSteeringValue(maxSteerVal,1)            break;          case 'd':          case 'ArrowRight':            vehicle.setSteeringValue(-maxSteerVal,0)            vehicle.setSteeringValue(-maxSteerVal,1)            break;        }      })      document.addEventListener('keyup',(event)=>{        switch (event.key) {          case 'w':          case 'ArrowUp':            vehicle.setWheelForce(0,0)            vehicle.setWheelForce(0,1)            break;          case 's':          case 'ArrowDown':            vehicle.setWheelForce(0,0)            vehicle.setWheelForce(0,1)              break;          case 'a':          case 'ArrowLeft':            vehicle.setSteeringValue(0,0)            vehicle.setSteeringValue(0,1)            break;          case 'd':          case 'ArrowRight':            vehicle.setSteeringValue(0,0)            vehicle.setSteeringValue(0,1)            break;        }      })      const animate = () => {        physicsWorld.fixedStep();        cannonDebugger.update();        window.requestAnimationFrame(animate);      };      animate();  } |

## 4.在App.jsx中调用appFunc2.

|  |
| --- |
| import { useEffect } from 'react'  import './App.css'  import { appFunc1 } from './appFuncs/appfunc1'  import { appFunc2 } from './appFuncs/appfunc2'  import { appFunc3 } from './appFuncs/appfunc3'    function App() {    useEffect(() => {      let canvas = document.getElementById("myThreeJSCanvas") //在外面创建canvas      // appFunc1(canvas)      appFunc2(canvas)      // appFunc3(canvas)    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |

### 效果：按w或者↑车辆前进，按s或者↓车辆后台，按a或者←往左转，按d或者→键转右，不过这里有一个问题，就是按键弹起了，车辆还是会运动不会停下来

## 5.编写appfunc3.js中的appFunc3函数，把线框体和threejs中是物体结合起来

|  |
| --- |
| 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CannonDebugger实例对象      const cannonDebugger = new CannonDebugger(threeApp.scene, physicsWorld);      //创建一个车体      let carBody = new CANNON.Body({        mass:5,        position:new CANNON.Vec3(0,6,0),        shape:new CANNON.Box(new CANNON.Vec3(4,0.5,2))      })      //利用车体创建刚体车辆      let vehicle = new CANNON.RigidVehicle({         chassisBody:carBody      })      //创建轮子      let mass = 1      let axisWidth = 5      let wheelShape = new CANNON.Sphere(1)      let wheelMaterial = new CANNON.Material('wheel')      let down = new CANNON.Vec3(0,-1,0)      let wheelBody1 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody1.addShape(wheelShape)      wheelBody1.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody1,        position:new CANNON.Vec3(-2, 0, axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })      let wheelBody2 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody2.addShape(wheelShape)      wheelBody2.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody2,        position:new CANNON.Vec3(-2, 0, -axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })      let wheelBody3 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody3.addShape(wheelShape)      wheelBody3.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody3,        position:new CANNON.Vec3(2, 0, axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })      let wheelBody4 = new CANNON.Body({        mass,        wheelMaterial      })      wheelBody4.addShape(wheelShape)      wheelBody4.angularDamping = 0.4      //创建轮子后，需要添加到钢体车辆里面      vehicle.addWheel({        body:wheelBody4,        position:new CANNON.Vec3(2, 0, -axisWidth / 2),        axis:new CANNON.Vec3(0,0,1),        direction:down      })        vehicle.addToWorld(physicsWorld)      // move car based on user input      document.addEventListener('keydown',(event)=>{        let maxSteerVal = Math.PI/8        let maxForce = 10        switch (event.key) {          case 'w':          case 'ArrowUp':            vehicle.setWheelForce(maxForce,0)            vehicle.setWheelForce(maxForce,1)              break;          case 's':          case 'ArrowDown':            vehicle.setWheelForce(-maxForce/2,0)            vehicle.setWheelForce(-maxForce/2,1)            break;          case 'a':          case 'ArrowLeft':            vehicle.setSteeringValue(maxSteerVal,0)            vehicle.setSteeringValue(maxSteerVal,1)            break;          case 'd':          case 'ArrowRight':            vehicle.setSteeringValue(-maxSteerVal,0)            vehicle.setSteeringValue(-maxSteerVal,1)            break;        }      })      document.addEventListener('keyup',(event)=>{        switch (event.key) {          case 'w':          case 'ArrowUp':            vehicle.setWheelForce(0,0)            vehicle.setWheelForce(0,1)            break;          case 's':          case 'ArrowDown':            vehicle.setWheelForce(0,0)            vehicle.setWheelForce(0,1)              break;          case 'a':          case 'ArrowLeft':            vehicle.setSteeringValue(0,0)            vehicle.setSteeringValue(0,1)            break;          case 'd':          case 'ArrowRight':            vehicle.setSteeringValue(0,0)            vehicle.setSteeringValue(0,1)            break;        }      })      // sync game world with physics world      let boxGeo = new THREE.BoxGeometry(8,1,4)      let boxMat = new THREE.MeshNormalMaterial()      let box = new THREE.Mesh(boxGeo,boxMat)      threeApp.scene.add(box)      let sphereGeo = new THREE.SphereGeometry(1)      let sphereMat = new THREE.MeshNormalMaterial()      let sphere1 = new THREE.Mesh(sphereGeo,sphereMat)      threeApp.scene.add(sphere1)      let sphere2 = new THREE.Mesh(sphereGeo,sphereMat)      threeApp.scene.add(sphere2)      let sphere3 = new THREE.Mesh(sphereGeo,sphereMat)      threeApp.scene.add(sphere3)      let sphere4 = new THREE.Mesh(sphereGeo,sphereMat)      threeApp.scene.add(sphere4)        const animate = () => {        physicsWorld.fixedStep();        cannonDebugger.update();        //把 CANNON创建的线框物体和Threejs创建的物体组合起来，需要把线框的位置和四元数复制给threejs文件的物体        box.position.copy(carBody.position)        box.quaternion.copy(carBody.quaternion)        sphere1.position.copy(wheelBody1.position)        sphere1.quaternion.copy(wheelBody1.quaternion)        sphere2.position.copy(wheelBody2.position)        sphere2.quaternion.copy(wheelBody2.quaternion)        sphere3.position.copy(wheelBody3.position)        sphere3.quaternion.copy(wheelBody3.quaternion)        sphere4.position.copy(wheelBody4.position)        sphere4.quaternion.copy(wheelBody4.quaternion)          window.requestAnimationFrame(animate);      };      animate();  } |

### 效果：和上面appFunc2函数的相关是一样的

# 总结：这个系列的教程比较简单，只是一个入门教程，只学习了最简单的代码，如果想进一步提高，需要学习一下Threejs，Tweenjs和Cannon-es的官方示例以及网上其他示例和教程，这里有一个比较好的地方是使用vscode的webgl glsl插件来写着色器源码，和使用vite-plugin-glsl来导入glsl文件，这样子比较方便。