# 这个项目，我们来模拟太阳系，就是地球绕着太阳转，月亮绕着地球转

# 项目结构

|  |
| --- |
|  |

## 1.我们以前写程序都是先创建场景，相机和渲染器，然后创建物体添加到场景如这里我们,我们的appfunc1.js的代码。注意：我们把ThreeApp的构造函数修改了一下，传递cavans对象而不是ID,我们新建了一个threeapp2.js同时保留threeapp.js以防万一有id需求

### src/appFuncs/appfunc1.js

|  |
| --- |
| import \* as THREE from 'three'  import ThreeApp from '../lib/threeapp2'  export function appFunc1(canvas){      let threeApp = new ThreeApp(canvas)      threeApp.initApp()      threeApp.animate()      //加载纹理      let sunTexture = new THREE.TextureLoader().load('../../assets/sun.jpeg')      let earthTexture = new THREE.TextureLoader().load('../../assets/earth.jpeg')      let moonTexture = new THREE.TextureLoader().load('../../assets/moon.jpeg')      const sunGeometry = new THREE.SphereGeometry(4);      const sunMaterial = new THREE.MeshStandardMaterial({        map: sunTexture,      });      const sunMesh = new THREE.Mesh(sunGeometry, sunMaterial);      threeApp.scene.add(sunMesh);      const earthGeometry = new THREE.SphereGeometry(2);      const earthMaterial = new THREE.MeshStandardMaterial({        map: earthTexture,      });      const earthMesh = new THREE.Mesh(earthGeometry, earthMaterial);      earthMesh.position.x = 12;      threeApp.scene.add(earthMesh);      //part 2.1 - start rotating the earth around the sun      const animate = () => {        threeApp.scene.rotation.y += 0.005; //可以旋转整个场景        window.requestAnimationFrame(animate);      };      animate();  } |

### src/lib/threeapp2.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  //这个类的构造方法是传递canvas对象而不是id，需要从外面获取canvas传递进来  export default class ThreeApp{      constructor(canvas){          //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.canvas= canvas          //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.directionalLight = 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 //cameraz轴值越大，图像越小          //创建渲染器          this.renderer = new THREE.WebGLRenderer({               canvas:this.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)          //创建方向光          this.directionalLight = new THREE.SpotLight(0xffffff,1)          //设置方向光的位置          this.directionalLight.position.set(0,32,64) //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 { appFunc1 } from './appFuncs/appfunc1'  function App() {    useEffect(() => {      let canvas = document.getElementById("myThreeJSCanvas") //在外面创建canvas      appFunc1(canvas)    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |

### 效果：发现太阳和地球一起旋转，其实这是不对的，太阳是恒星，不会运动

## 2.我们编写appfunc2.js来尝试修正一下，只是让地球旋转太阳不动，而且地球还会自转

|  |
| --- |
| import \* as THREE from 'three'  import ThreeApp from '../lib/threeapp2'  export function appFunc2(canvas){      let threeApp = new ThreeApp(canvas)      threeApp.initApp()      threeApp.animate()      //加载纹理      let sunTexture = new THREE.TextureLoader().load('../../assets/sun.jpeg')      let earthTexture = new THREE.TextureLoader().load('../../assets/earth.jpeg')      let moonTexture = new THREE.TextureLoader().load('../../assets/moon.jpeg')      const sunGeometry = new THREE.SphereGeometry(4);      const sunMaterial = new THREE.MeshStandardMaterial({        map: sunTexture,      });      const sunMesh = new THREE.Mesh(sunGeometry, sunMaterial);      threeApp.scene.add(sunMesh);      const earthGeometry = new THREE.SphereGeometry(2);      const earthMaterial = new THREE.MeshStandardMaterial({        map: earthTexture,      });      const earthMesh = new THREE.Mesh(earthGeometry, earthMaterial);      earthMesh.position.x = 12;      threeApp.scene.add(earthMesh);        const animate = () => {        // 不要旋转最高场景，只让地球绕转        let t = threeApp.clock.getElapsedTime()        earthMesh.position.x = 12 \* Math.cos(t/2)        earthMesh.position.z = -12 \* Math.sin(t/2)        earthMesh.rotation.y = t        window.requestAnimationFrame(animate);      };      animate();  } |

### 效果：地球绕着太阳转和自转同时进行

## 3.这种方法虽然能够实现功能但是，如果再添加一个月球呢？就会有点复杂，所以我们可以使用threejs的物体分组的功能，我们写一个appfunc3.js，代码如下

|  |
| --- |
| import \* as THREE from 'three'  import ThreeApp from '../lib/threeapp2'  export function appFunc3(canvas){      let threeApp = new ThreeApp(canvas)      threeApp.initApp()      threeApp.animate()      //加载纹理      let sunTexture = new THREE.TextureLoader().load('../../assets/sun.jpeg')      let earthTexture = new THREE.TextureLoader().load('../../assets/earth.jpeg')      let moonTexture = new THREE.TextureLoader().load('../../assets/moon.jpeg')      //场景太阳系统组和地球轨道组      let solarSystemGroup = new THREE.Group()      let earthOrbit = new THREE.Group()      const sunGeometry = new THREE.SphereGeometry(4);      const sunMaterial = new THREE.MeshStandardMaterial({        map: sunTexture,      });      const sunMesh = new THREE.Mesh(sunGeometry, sunMaterial);      solarSystemGroup.add(sunMesh) //把太阳添加到太阳系组      threeApp.scene.add(solarSystemGroup);//把太阳系组添加到场景中      const earthGeometry = new THREE.SphereGeometry(2);      const earthMaterial = new THREE.MeshStandardMaterial({        map: earthTexture,      });      const earthMesh = new THREE.Mesh(earthGeometry, earthMaterial);      earthMesh.position.x = 12;      earthOrbit.add(earthMesh) //把地球添加到地球轨道组中      threeApp.scene.add(earthOrbit);//把地球轨道组添加到场景中        const animate = () => {        // 实现地球绕转太阳转        let t = threeApp.clock.getElapsedTime()        earthOrbit.rotation.y += 0.005 //转太阳转        earthMesh.rotation.y = t //注意地球需要自转的        window.requestAnimationFrame(animate);      };      animate();  } |

### 效果：地球自转同时绕太阳公转

## 4.现在我们添加月球，编写appfunc4.js,代码如下

|  |
| --- |
| import \* as THREE from 'three'  import ThreeApp from '../lib/threeapp2'  export function appFunc4(canvas){      let threeApp = new ThreeApp(canvas)      threeApp.initApp()      threeApp.animate()      //加载纹理      let sunTexture = new THREE.TextureLoader().load('../../assets/sun.jpeg')      let earthTexture = new THREE.TextureLoader().load('../../assets/earth.jpeg')      let moonTexture = new THREE.TextureLoader().load('../../assets/moon.jpeg')      //场景太阳系统组和地球轨道组      let solarSystemGroup = new THREE.Group()      let earthOrbit = new THREE.Group()//地球轨道组      let moonOrbit = new THREE.Group() //月球轨道组      const sunGeometry = new THREE.SphereGeometry(4);      const sunMaterial = new THREE.MeshStandardMaterial({        map: sunTexture,      });      const sunMesh = new THREE.Mesh(sunGeometry, sunMaterial);      solarSystemGroup.add(sunMesh) //把太阳添加到太阳系组      threeApp.scene.add(solarSystemGroup);//把太阳系组添加到场景中      const earthGeometry = new THREE.SphereGeometry(2);      const earthMaterial = new THREE.MeshStandardMaterial({        map: earthTexture,      });      const earthMesh = new THREE.Mesh(earthGeometry, earthMaterial);      earthMesh.position.x = 12;      earthOrbit.add(earthMesh) //把地球添加到地球轨道组中      //创建月球并且添加到月球轨道组中      const moonGeometry = new THREE.SphereGeometry(1);      const moonMaterial = new THREE.MeshStandardMaterial({        map: moonTexture,      });      let moonMesh = new THREE.Mesh(moonGeometry,moonMaterial)      moonOrbit.position.x = 12      moonMesh.position.x = 4      moonOrbit.add(moonMesh)      earthOrbit.add(moonOrbit)      threeApp.scene.add(earthOrbit);//把地球轨道组添加到场景中        const animate = () => {        // 实现地球绕转太阳转        let t = threeApp.clock.getElapsedTime()        //月亮绕转地球转        moonOrbit.rotation.y += 0.05        moonMesh.rotation.y = t        earthOrbit.rotation.y += 0.005 //转太阳转        earthMesh.rotation.y = t //注意地球需要自转的        window.requestAnimationFrame(animate);      };      animate();  } |

### App.jsx

|  |
| --- |
| import { useEffect } from 'react'  import './App.css'  import { appFunc1 } from './appFuncs/appfunc1'  import { appFunc2 } from './appFuncs/appfunc2'  import { appFunc3 } from './appFuncs/appfunc3'  import { appFunc4 } from './appFuncs/appfunc4'  function App() {    useEffect(() => {      let canvas = document.getElementById("myThreeJSCanvas") //在外面创建canvas      // appFunc1(canvas)      // appFunc2(canvas)      // appFunc3(canvas)      appFunc4(canvas)    }, [])    return (      <>        <div>          <canvas id="myThreeJSCanvas"></canvas>        </div>      </>    )  }  export default App |

### 效果：月亮会自转和绕地球公转，地球会自转和绕太阳公转，他们是同时进行的

# 总结，绕转如果是物体，需要使用它的position属性并且需要调整x和z轴的位置，如果是物体组，只需要调整rotation属性的y轴，自转需要调整物体的坐标轴，

# 物体组使用threejs的Group类

|  |
| --- |
|  |