项目使用Create-React-App+ts+Three

**项目创建**

1. 创建项目

yarn create react-app my-app --template typescript

1. 安装three

yarn add three @types/three

**示例**

这个示例让大家快速过一下代码，好了解three渲染的整个过程

1. 新建renderGL.ts文件，代码如下

import \* as THREE from 'three';

export default function () {

    /\*\*

   \* 创建场景对象Scene

   \*/

    var scene = new THREE.Scene();

    /\*\*

     \* 创建网格模型

     \*/

    var geometry = new THREE.BoxGeometry(100, 100, 100); //创建一个立方体几何对象Geometry

    var material = new THREE.MeshLambertMaterial({

        color: 0x0000ff

    }); //材质对象Material

    var mesh = new THREE.Mesh(geometry, material); //网格模型对象Mesh

    scene.add(mesh); //网格模型添加到场景中

    /\*\*

     \* 光源设置

     \*/

    //点光源

    var point = new THREE.PointLight(0xffffff);

    point.position.set(400, 200, 300); //点光源位置

    scene.add(point); //点光源添加到场景中

    //环境光

    var ambient = new THREE.AmbientLight(0x444444);

    scene.add(ambient);

    /\*\*

     \* 相机设置

     \*/

    var width = window.innerWidth; //窗口宽度

    var height = window.innerHeight; //窗口高度

    var k = width / height; //窗口宽高比

    var s = 200; //三维场景显示范围控制系数，系数越大，显示的范围越大

    //创建相机对象

    var camera = new THREE.OrthographicCamera(-s \* k, s \* k, s, -s, 1, 1000);

    camera.position.set(200, 300, 200); //设置相机位置

    camera.lookAt(scene.position); //设置相机方向(指向的场景对象)

    /\*\*

     \* 创建渲染器对象

     \*/

    var renderer = new THREE.WebGLRenderer();

    renderer.setSize(width, height);//设置渲染区域尺寸

    renderer.setClearColor(0xb9d3ff, 1); //设置背景颜色

    document.body.appendChild(renderer.domElement); //body元素中插入canvas对象

    //执行渲染操作   指定场景、相机作为参数

    renderer.render(scene, camera);

}

1. 在index.tsx中引入并调用该方法

import renderGL from './renderGL';

renderGL();

1. 执行yarn start运行项目

**基本对象**

上面的示例中，我们总结几个基本对象

1. 场景

    /\*\*

   \* 创建场景对象Scene

   \*/

    var scene = new THREE.Scene();

1. 网格模型

模型就是物体了，如下建立了一个模型，模型由几何体和材质组成

    /\*\*

     \* 创建网格模型

     \*/

    var geometry = new THREE.BoxGeometry(100, 100, 100); //创建一个立方体几何对象Geometry

    var material = new THREE.MeshLambertMaterial({

        color: 0x0000ff

    }); //材质对象Material

    var mesh = new THREE.Mesh(geometry, material); //网格模型对象Mesh

    scene.add(mesh); //网格模型添加到场景中

1. 点光源

点光源比较常见，比如灯泡等就是点光源

    //点光源

    var point = new THREE.PointLight(0xffffff);

    point.position.set(400, 200, 300); //点光源位置

    scene.add(point); //点光源添加到场景中

1. 环境光

    //环境光

    var ambient = new THREE.AmbientLight(0x444444);

    scene.add(ambient);

1. 相机

    /\*\*

     \* 相机设置

     \*/

    var width = window.innerWidth; //窗口宽度

    var height = window.innerHeight; //窗口高度

    var k = width / height; //窗口宽高比

    var s = 200; //三维场景显示范围控制系数，系数越大，显示的范围越大

    //创建相机对象

    var camera = new THREE.OrthographicCamera(-s \* k, s \* k, s, -s, 1, 1000);

    camera.position.set(200, 300, 200); //设置相机位置

    camera.lookAt(scene.position); //设置相机方向(指向的场景对象)

1. 渲染器

渲染器负债将场景渲染到canvas元素中

    /\*\*

     \* 创建渲染器对象

     \*/

    var renderer = new THREE.WebGLRenderer();

    renderer.setSize(width, height);//设置渲染区域尺寸

    renderer.setClearColor(0xb9d3ff, 1); //设置背景颜色

    document.body.appendChild(renderer.domElement); //body元素中插入canvas对象

    //执行渲染操作   指定场景、相机作为参数

    renderer.render(scene, camera);

**模型动起来**

如何让模型旋转起来，如下，在浏览器的每一帧动画我们都执行renderer.render重新渲染场景。

    (function ani() {

        // 旋转模型

        mesh.rotateY(0.01);

        // 渲染

        renderer.render(scene, camera);

        requestAnimationFrame(ani);

    })();

- 均匀的旋转

我们使用时间去计算即可

    let t0 = new Date().getTime();

    (function ani() {

        let t1 = new Date().getTime();

        // 旋转模型

        mesh.rotateY(0.001 \* (t1 - t0));

        t0 = t1;

        // 渲染

        renderer.render(scene, camera);

        requestAnimationFrame(ani);

    })();

**鼠标控制镜头**

通常情况下我们通过如下方式控制镜头

- 缩放：滚动—鼠标中键

- 旋转：拖动—鼠标左键

- 平移：拖动—鼠标右键

在\node\_modules\three\examples\js\controls目录下有一个OrbitControls.js文件，我们将该文件复制到我们的目录下面，因为我们使用的是ts，需要做一下改动

1. OrbitControls.js文件修改

- 把文件后缀修改为ts（即OrbitControls.ts）

- 在首行添加 // @ts-nocheck

- import \* as THREE from 'three';

- 导出class OrbitControls

修改后大概是这个样子

// @ts-nocheck

import \* as THREE from 'three';

...

export class OrbitControls extends THREE.EventDispatcher {

    ...

}

export class MapControls extends OrbitControls {

    ...

}

1. 镜头控制代码

代码只有简单的一行，当我们移动鼠标时，OrbitControls会更改camera的坐标，而我们的动画函数ani时时刻刻都在执行，就能把更新后的坐标渲染到canvas上了

import \* as THREE from 'three';

import {OrbitControls} from './OrbitControls';

export default function () {

    ...

    (function ani() {

        renderer.render(scene, camera);

        requestAnimationFrame(ani);

    })();

    // 控制镜头旋转

    var controls = new OrbitControls(camera, renderer.domElement);

}

**几何体**

如下three提供的几何体，可以修改尝试一下

//长方体 参数：长，宽，高

var geometry = new THREE.BoxGeometry(100, 100, 100);

// 球体 参数：半径60  经纬度细分数40,40

var geometry = new THREE.SphereGeometry(60, 40, 40);

// 圆柱  参数：圆柱面顶部、底部直径50,50   高度100  圆周分段数

var geometry = new THREE.CylinderGeometry( 50, 50, 100, 25 );

// 正八面体

var geometry = new THREE.OctahedronGeometry(50);

// 正十二面体

var geometry = new THREE.DodecahedronGeometry(50);

// 正二十面体

var geometry = new THREE.IcosahedronGeometry(50);

- 几何体常用方法

// 旋转

rotateX(angle: number): BufferGeometry;

rotateY(angle: number): BufferGeometry;

rotateZ(angle: number): BufferGeometry;

// 移动

translate(x: number, y: number, z: number): BufferGeometry;

// 缩放

scale(x: number, y: number, z: number): BufferGeometry;

**材质**

材质可以赋给模型，不同材质对光有不同的反应

| 材质类型 | 功能 |
| --- | --- |
| [MeshBasicMaterial](http://www.yanhuangxueyuan.com/threejs/docs/index.html" \l "api/zh/materials/MeshBasicMaterial) | 基础网格材质，不受光照影响的材质 |
| [MeshLambertMaterial](http://www.yanhuangxueyuan.com/threejs/docs/index.html" \l "api/zh/materials/MeshLambertMaterial) | Lambert网格材质，与光照有反应，漫反射 |
| [MeshPhongMaterial](http://www.yanhuangxueyuan.com/threejs/docs/index.html" \l "api/zh/materials/MeshPhongMaterial) | 高光Phong材质,与光照有反应 |
| [MeshStandardMaterial](http://www.yanhuangxueyuan.com/threejs/docs/index.html" \l "api/zh/materials/MeshStandardMaterial) | PBR物理材质，相比较高光Phong材质可以更好的模拟金属、玻璃等效果 |

- 示例高光效果

    var material = new THREE.MeshPhongMaterial({

        color:0x0000ff,

        specular:0x4488ee,

        shininess:12

    });

**光源**

Three提供了如下光源

| 光源 | 简介 |
| --- | --- |
| [AmbientLight](http://www.yanhuangxueyuan.com/threejs/docs/index.html" \l "api/zh/lights/AmbientLight) | 环境光 |
| [PointLight](http://www.yanhuangxueyuan.com/threejs/docs/index.html" \l "api/zh/lights/PointLight) | 点光源 |
| [DirectionalLight](http://www.yanhuangxueyuan.com/threejs/docs/index.html" \l "api/zh/lights/DirectionalLight) | 平行光，比如太阳光 |
| [SpotLight](http://www.yanhuangxueyuan.com/threejs/docs/index.html" \l "api/zh/lights/SpotLight) | 聚光源 |

- 点光源位置

var point = new THREE.PointLight(0xffffff);

point.position.set(400, 200, 300); //点光源位置