

案例查看地址：<http://www.wjceo.com/blog/threejs/2018-02-09/6.html>

（1）创建了渲染器，并传值对象antialias设置为true，告诉Three.js要启用抗锯齿（antialiased）渲染。抗锯齿可以避免绘制物体边缘时产生的锯齿。

（2）创建场景。

（3）创建相机，设置相机的投影视角，并设置相机的位置。

（4）创建一个平行光光源（directional light），在这里我们设置了平行光的一个点，而光线的照射方向就是设置的这个点指向原点的方向。

（5）从48到60行，我们开始绘制立方体。首先我们使用了THREE.ImageUtils.loadTexture方法来加载纹理。然后改变了立方体的材质，MeshBasicMaterial用于定义属性简单的材质，比如固定颜色或是透明。这种材质不会对光照做出任何响应。所以我们用另一种类型的材质来代替它，这就是MeshPhongMaterial。这种类型的材质应用了一种相对简单、仿真度高而又性能优越的着色器模型，也就是著名的“Phong着色法”（Phong shading）。我们现在就可以分辨出立方体的边缘。立方体朝向光源方向的面将会更加明亮，而背对光源的面则会相对阴暗。

在51行，实例化材质的时候，我们将纹理传入进去。Phong内置的着色器结合光源的设置、材质颜色和纹理映射的像素值，使得每一个像素显示正确的颜色，并形成最后的图像效果。

然后我们又创建了一个立方体的集合体，将构造好的几何体、材质和纹理都放到了Three.js的网格中，并存储在变量cube中。

最后，把cube网格加入场景当中，立方体就可以直接显示了。

（6）在62到74行，我们生命一个rotationBool变量来代表模型是否旋转的变量。这里我们直接制作了一个动画，动画里面，每次渲染一次页面，判断是否动画旋转。

在Three.js中，每个物体都有位置（position）、旋转（rotation）和缩放（scale）属性。通过给网格cube的相关属性的赋值，即可修改物体的样式。（在为物体旋转变量赋值时，我们需要注意，大部分的3D图形系统都使用了弧度制（radians）来度量角度。弧度是只单位圆上相应角度的圆弧长度（例如，弧度制的2π就是角度值的360°）。Math.PI相当于180°，因此当我们赋值mesh.rotation.x = Math.PI/12的时候，实际上是绕着x轴旋转了15°）。

requestAnimationFrame()方法是为了重绘使用的。使用它，浏览器可以极大地优化动画的性能表现。因为它会综合考虑所有的绘制请求，把它们都放到同一个重绘步骤中。尤其是在多标签浏览器中，当动画页面处于后台时，浏览器将停止重绘以节省资源提高性能。这个函数呗装专门设置用于制作页面动画，当然也包括WebGL中的动画。

（7）最后一步是给body绑定了一个点击事件，只要点击body，动画就会暂停。

案例代码：

1. <!doctype html>
2. <html>
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport"
6. content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
7. <meta http-equiv="X-UA-Compatible" content="ie=edge">
8. <title>Document</title>
9. <script src="build/three.js"></script>
10. <style>
11. body {
12. margin: 0;
13. }
15. canvas {
16. width: 100%;
17. height: 100%;
18. display: block;
19. }
20. </style>
21. </head>
22. <body>
23. <script>
24. *//创建渲染器，添加到dom当中, antialias（是否启用抗锯齿）*
25. var renderer = new THREE.WebGLRenderer({antialias: true});
26. *//设置渲染器的尺寸*
27. renderer.setSize(window.innerWidth, window.innerHeight);
28. *//将渲染器放置到页面当中*
29. document.body.appendChild(renderer.domElement);
31. *//创建场景*
32. var scene = new THREE.Scene();
34. *//创建相机，设置位置*
35. var camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 1, 4000);
36. *//设置相机的位置*
37. camera.position.set(0, 0, 3);
39. *//创建一个平行光光源照射到物体上*
40. var light = new THREE.DirectionalLight(0xffffff, 1.5);
41. *//设置平型光照射方向，照射方向为设置的点照射到原点*
42. light.position.set(0, 0, 1);
43. *//将灯光放到场景当中*
44. scene.add(light);
46. *//创建一个接受光照并带有纹理映射的立方体，并添加到场景中*
47. *//首先，获取到纹理*
48. var map = THREE.ImageUtils.loadTexture("../three.js-master/examples/textures/disturb.jpg");
50. *//然后创建一个phong材质来处理着色，并传递给纹理映射*
51. var material = new THREE.MeshPhongMaterial({map: map});
53. *//创建一个立方体的几何体*
54. var geometry = new THREE.CubeGeometry(1, 1, 1);
56. *//将集合体和材质放到一个网格中*
57. var cube = new THREE.Mesh(geometry, material);
59. *//将立方体网格添加到场景中*
60. scene.add(cube);
62. *//声明一个判断是否旋转的变量*
63. var rotationBool = true;
65. (function animate() {
66. renderer.render(scene, camera);
68. if (rotationBool) {
69. cube.rotation.x += 0.02;
70. cube.rotation.y += 0.02;
71. }
73. requestAnimationFrame(animate);
74. })();
76. document.body.onclick = function () {
77. rotationBool = !rotationBool;
78. }
80. </script>
81. </body>
82. </html>