案例查看效果地址：<http://www.wjceo.com/blog/threejs/2018-02-09/12.html>

这个效果相对于第八节的轨迹球插件使用上感觉要好，虽然轨迹球插件可以来回的滚动，但是容易分辨不清楚上下左右的关系，容易混乱，适合调试，而轨道控制插件orbit则适合客户使用，还不会产生混乱效果。下面讲一下使用。

（1）首先引入插件，文件地址在官方案例的examples/js/controls/OrbitControls.js。

（2）然后实例化函数，把相机和渲染器的dom传入，并设置相关设置。

1. *//用户交互插件 鼠标左键按住旋转，右键按住平移，滚轮缩放*
2. var controls;
3. function initControls() {
5. controls = new THREE.OrbitControls( camera, renderer.domElement );
7. *// 如果使用animate方法时，将此函数删除*
8. *//controls.addEventListener( 'change', render );*
9. *// 使动画循环使用时阻尼或自转 意思是否有惯性*
10. controls.enableDamping = true;
11. *//动态阻尼系数 就是鼠标拖拽旋转灵敏度*
12. *//controls.dampingFactor = 0.25;*
13. *//是否可以缩放*
14. controls.enableZoom = true;
15. *//是否自动旋转*
16. controls.autoRotate = true;
17. *//设置相机距离原点的最远距离*
18. controls.minDistance = 200;
19. *//设置相机距离原点的最远距离*
20. controls.maxDistance = 600;
21. *//是否开启右键拖拽*
22. controls.enablePan = true;
23. }

（3）最后，在animate函数内调用orbit的update()更新。

1. function animate() {
2. *//更新控制器*
3. controls.update();
4. render();
6. *//更新性能插件*
7. stats.update();
8. requestAnimationFrame(animate);
9. }

就实现了相关效果。下面是全部案例代码：

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Title</title>
6. <style type="text/css">
7. html, body {
8. margin: 0;
9. height: 100%;
10. }
12. canvas {
13. display: block;
14. }
16. </style>
17. </head>
18. <body onload="draw();">
20. </body>
21. <script src="build/three.js"></script>
22. <script src="examples/js/controls/OrbitControls.js"></script>
23. <script src="examples/js/libs/stats.min.js"></script>
24. <script>
25. var renderer;
26. function initRender() {
27. renderer = new THREE.WebGLRenderer({antialias:true});
28. renderer.setSize(window.innerWidth, window.innerHeight);
29. document.body.appendChild(renderer.domElement);
30. }
32. var camera;
33. function initCamera() {
34. camera = new THREE.PerspectiveCamera(45, window.innerWidth/window.innerHeight, 1, 10000);
35. camera.position.set(0, 0, 400);
36. }
38. var scene;
39. function initScene() {
40. scene = new THREE.Scene();
41. }
43. var light;
44. function initLight() {
45. scene.add(new THREE.AmbientLight(0x404040));
47. light = new THREE.DirectionalLight(0xffffff);
48. light.position.set(1,1,1);
49. scene.add(light);
50. }
52. function initModel() {
53. var map = new THREE.TextureLoader().load("examples/textures/UV\_Grid\_Sm.jpg");
54. var material = new THREE.MeshLambertMaterial({map:map});
56. var cube = new THREE.Mesh(new THREE.BoxGeometry(100, 200, 100, 1, 1, 1), material);
57. scene.add(cube);
58. }
60. *//初始化性能插件*
61. var stats;
62. function initStats() {
63. stats = new Stats();
64. document.body.appendChild(stats.dom);
65. }
67. *//用户交互插件 鼠标左键按住旋转，右键按住平移，滚轮缩放*
68. var controls;
69. function initControls() {
71. controls = new THREE.OrbitControls( camera, renderer.domElement );
73. *// 如果使用animate方法时，将此函数删除*
74. *//controls.addEventListener( 'change', render );*
75. *// 使动画循环使用时阻尼或自转 意思是否有惯性*
76. controls.enableDamping = true;
77. *//动态阻尼系数 就是鼠标拖拽旋转灵敏度*
78. *//controls.dampingFactor = 0.25;*
79. *//是否可以缩放*
80. controls.enableZoom = true;
81. *//是否自动旋转*
82. controls.autoRotate = true;
83. *//设置相机距离原点的最远距离*
84. controls.minDistance = 200;
85. *//设置相机距离原点的最远距离*
86. controls.maxDistance = 600;
87. *//是否开启右键拖拽*
88. controls.enablePan = true;
89. }
91. function render() {
92. renderer.render( scene, camera );
93. }
95. *//窗口变动触发的函数*
96. function onWindowResize() {
97. camera.aspect = window.innerWidth / window.innerHeight;
98. camera.updateProjectionMatrix();
99. render();
100. renderer.setSize( window.innerWidth, window.innerHeight );
102. }
104. function animate() {
105. *//更新控制器*
106. controls.update();
107. render();
109. *//更新性能插件*
110. stats.update();
111. requestAnimationFrame(animate);
112. }
114. function draw() {
115. initRender();
116. initScene();
117. initCamera();
118. initLight();
119. initModel();
120. initControls();
121. initStats();
123. animate();
124. window.onresize = onWindowResize;
125. }
126. </script>
127. </html>