

首先你需要实例化 THREE.FontLoader() 来进行json格式的文字格式加载，在加载成功的回调函数里面进行创建网格。

然后通过THREE.TextBufferGeometry或者THREE.TextGeometry方法进行网格创建，并将需要设置的问题传入。

再设置一个纹理，通过THREE.Mesh()函数创建成图形添加到场景当中即可。

最后又调节了一下位置，就成了现在这个样子的代码。

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. var fontModel;
53. function initModel() {
54. var font;
55. var loader = new THREE.FontLoader();
56. loader.load("examples/fonts/gentilis\_regular.typeface.json", function (res) {
57. font = new THREE.TextBufferGeometry("fdsfasd", {
58. font: res,
59. size: 100,
60. height: 60
61. });
63. font.computeBoundingBox(); *// 运行以后设置font的boundingBox属性对象，如果不运行无法获得。*
64. *//font.computeVertexNormals();*
66. var map = new THREE.TextureLoader().load("examples/textures/UV\_Grid\_Sm.jpg");
67. var material = new THREE.MeshLambertMaterial({map:map,side:THREE.DoubleSide});
69. fontModel = new THREE.Mesh(font,material);
71. *//设置位置*
72. fontModel.position.x = - (font.boundingBox.max.x - font.boundingBox.min.x)/2; *//计算出整个模型的宽度的一半*
73. fontModel.position.y = - 50;
74. fontModel.position.z = - 30;
76. scene.add(fontModel);
77. });
78. }
80. *//初始化性能插件*
81. var stats;
82. function initStats() {
83. stats = new Stats();
84. document.body.appendChild(stats.dom);
85. }
87. *//用户交互插件 鼠标左键按住旋转，右键按住平移，滚轮缩放*
88. var controls;
89. function initControls() {
91. controls = new THREE.OrbitControls(camera, renderer.domElement);
93. *// 如果使用animate方法时，将此函数删除*
94. *//controls.addEventListener( 'change', render );*
95. *// 使动画循环使用时阻尼或自转 意思是否有惯性*
96. controls.enableDamping = true;
97. *//动态阻尼系数 就是鼠标拖拽旋转灵敏度*
98. *//controls.dampingFactor = 0.25;*
99. *//是否可以缩放*
100. controls.enableZoom = true;
101. *//是否自动旋转*
102. controls.autoRotate = false;
103. *//设置相机距离原点的最远距离*
104. controls.minDistance = 200;
105. *//设置相机距离原点的最远距离*
106. controls.maxDistance = 600;
107. *//是否开启右键拖拽*
108. controls.enablePan = true;
109. }
111. function render() {
112. renderer.render(scene, camera);
113. }
115. *//窗口变动触发的函数*
116. function onWindowResize() {
117. camera.aspect = window.innerWidth / window.innerHeight;
118. camera.updateProjectionMatrix();
119. render();
120. renderer.setSize(window.innerWidth, window.innerHeight);
122. }
124. function animate() {
125. *//更新控制器*
126. controls.update();
127. render();
129. *//更新性能插件*
130. stats.update();
131. requestAnimationFrame(animate);
132. }
134. function draw() {
135. initRender();
136. initScene();
137. initCamera();
138. initLight();
139. initModel();
140. initControls();
141. initStats();
143. animate();
144. window.onresize = onWindowResize;
145. }
146. </script>
147. </html>