# 3.2 网格对象的函数和属性

本文主要介绍如何如何修改网格的位置、旋转和缩放。

我们已经知道，创建一个网格需要一个几何体，以及一个或多个材质。网格创建好之后，我们就可以把它添加到场景中，它就可以被渲染了。你可以利用网格的几个属性来改变它在场景中的位置，及其显示效果。在下面的示例里，我们会看一下如下的属性和函数：

|  |  |
| --- | --- |
| 函数/属性 | 描述 |
| position（位置） | 决定该对象相对其父对象的位置。多数情况下，一个对象的父对象是THREE.Scene()对象 |
| rotation（旋转） | 通过这个属性你可以设置对象绕任何一个轴的旋转弧度 |
| scale（比例） | 通过这个属性你可以沿着x、y和z轴缩放对象 |
| translateX(amount)（x轴平移） | 沿x轴将对象平移指定的距离 |
| translateY(amount)（y轴平移） | 沿y轴将对象平移指定的距离 |
| translateZ(amount)（z轴平移） | 沿z轴将对象平移指定的距离 |

示例截图如下：



通过下拉菜单你可以修改这些属性，而且立即就可以看到修改的结果。

position属性可以设置对象的x、y和z轴坐标。一个对象的位置是相对于其父对象而言的，而父对象则通常是你向其中添加对象的那个场景。我们可以用三种方式设置对象的位置：

直接设置坐标，如下所示：

cube.position.x=10;

cube.position.y=3;

cube.position.z=l;

也可以一次性地设置它们：

cube.position.set(10,3,1);

rotation（旋转）属性可以设置对象绕轴旋转的弧度。下面的代码片段展示的是如何设置旋转：

cube.rotation.x=0.5\*Math.PI;

cube.rotation.set(0.5\*Math.PI,0,0);

scale（缩放）可以沿着指定轴缩放对象。如果你设的缩放值小于1，这个对象就会缩小，如果这个值大于1，那么这个对象就会变大。

translate函数也可以改变对象的位置，但不是定义你想要将对象放在哪里的绝对位置，而是定义相对当前位置对象移动的量。假设你已经在场景中添加了一个球体，位置是(1,2,3)。下一步我们想让这个对象沿着x轴平移，translateX(4)。它的位置现在就是(5,2,3)。如果我们想让这个对象恢复到它原来的位置，可以调用translateX(-4)。在示例中有一个translate菜单项。通过它你可以试验这个功能。只要设置x、y和z轴方向平移的距离，点击translate按钮，你就会看到这个对象会依照这三个值移动到一个新位置。

## 完整代码

<!DOCTYPE html>

<html>

<head>

 <title>示例03.02 - 网格属性</title>

 <script src="../../../Scripts/jquery-2.1.3.min.js"></script>

 <script src="../../../Scripts/Threejs/three.min.js"></script>

 <script src="../../../Scripts/Threejs/stats.js"></script>

 <script src="../../../Scripts/Threejs/dat.gui.min.js"></script>

 <style>

 body{

 /\* 将margin设置为0，overflow设置为hidden，可让浏览器显示全屏 \*/

 margin: 0;

 overflow: hidden;

 }

 </style>

</head>

<body>

<div id="Stats-output">

</div>

<!-- 作为Canvas容器的div -->

<div id="WebGL-output">

</div>

<script type="text/javascript">

 // 页面加载完毕后，就可以运行Three.js了。

 $(function () {

 var stats = initStats();

 // 创建渲染器，并设置视口大小和清除色

 var renderer = new THREE.WebGLRenderer(); renderer.setClearColor(0xEEEEEE, 1.0);

 renderer.setSize(window.innerWidth, window.innerHeight);

 renderer.shadowMapEnabled = true;

 // 将WebGL的输出canvas放置到div中

 $("#WebGL-output").append(renderer.domElement);

 // 创建scene对象，用来容纳网格、相机、光源等对象

 var scene = new THREE.Scene();

 // 场景相机

 var camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 0.1, 1000);

 camera.position.x = -30;

 camera.position.y = 40;

 camera.position.z = 30;

 camera.lookAt(scene.position);

 // 创建一个Plane作为地面

 var planeGeometry = new THREE.PlaneBufferGeometry(60,40,1,1);

 var planeMaterial = new THREE.MeshLambertMaterial({color: 0xffffff});

 var plane = new THREE.Mesh(planeGeometry,planeMaterial);

 plane.receiveShadow = true;

 plane.rotation.x=-0.5\*Math.PI;

 plane.position.x=0

 plane.position.y=0

 plane.position.z=0

 scene.add(plane);

 // 添加环境光

 var ambientLight = new THREE.AmbientLight(0x0c0c0c);

 scene.add(ambientLight);

 // 添加一个聚光灯用于产生阴影

 var spotLight = new THREE.SpotLight( 0xffffff );

 spotLight.position.set( -40, 60, 020 );

 spotLight.castShadow = true;

 scene.add(spotLight);

 // 在场景中添加长方体

 var material = new THREE.MeshLambertMaterial({ color: 0x44ff44 });

 var geom = new THREE.BoxGeometry(5, 8, 3);

 var box = new THREE.Mesh(geom, material);

 box.position.y = 4;

 box.castShadow = true;

 scene.add(box);

 // 添加菜单

 var controls = new function() {

 this.scaleX = 1;

 this.scaleY = 1;

 this.scaleZ = 1;

 this.positionX = 0;

 this.positionY = 4;

 this.positionZ = 0;

 this.rotationX = 0;

 this.rotationY = 0;

 this.rotationZ = 0;

 this.scale = 1;

 this.translateX = 0;

 this.translateY = 0;

 this.translateZ = 0;

 this.translate = function() {

 box.translateX(this.translateX);

 box.translateY(this.translateX);

 box.translateZ(this.translateX);

 this.positionX = box.position.x;

 this.positionY = box.position.y;

 this.positionZ = box.position.z;

 }

 }

 var gui = new dat.GUI();

 guiScale = gui.addFolder('scale');

 guiScale.add(controls,'scaleX',0,5);

 guiScale.add(controls,'scaleY',0,5);

 guiScale.add(controls,'scaleZ',0,5);

 guiPosition = gui.addFolder('position');

 var contX = guiPosition.add(controls,'positionX',-10,10);

 var contY = guiPosition.add(controls,'positionY',-4,20);

 var contZ = guiPosition.add(controls,'positionZ',-10,10);

 contX.listen();

 contX.onChange(function(value) {

 box.position.x=controls.positionX;

 });

 contY.listen();

 contY.onChange(function(value) {

 box.position.y=controls.positionY;

 });

 contZ.listen();

 contZ.onChange(function(value) {

 box.position.z=controls.positionZ;

 });

 guiRotation = gui.addFolder('rotation');

 guiRotation.add(controls,'rotationX',-4,4);

 guiRotation.add(controls,'rotationY',-4,4);

 guiRotation.add(controls,'rotationZ',-4,4);

 guiTranslate = gui.addFolder('translate');

 guiTranslate.add(controls,'translateX',-10,10);

 guiTranslate.add(controls,'translateY',-10,10);

 guiTranslate.add(controls,'translateZ',-10,10);

 guiTranslate.add(controls,'translate');

 render();

 function render() {

 stats.update();

 box.rotation.x=controls.rotationX;

 box.rotation.y=controls.rotationY;

 box.rotation.z=controls.rotationZ;

 box.scale.set(controls.scaleX,controls.scaleY,controls.scaleZ);

 requestAnimationFrame(render);

 renderer.render(scene, camera);

 }

 function initStats() {

 var stats = new Stats();

 stats.setMode(0); // 0: fps, 1: ms

 // Align top-left

 stats.domElement.style.position = 'absolute';

 stats.domElement.style.left = '0px';

 stats.domElement.style.top = '0px';

 $("#Stats-output").append( stats.domElement );

 return stats;

 }

 });

</script>

</body>

</html>