# 3.6 三维几何体——圆环、环结、多面体

本文介绍三种三维几何体：环、环结和多面体，其中多面体使用起来非常灵活。

## TorusGeometry

Torus（圆环）是一种简单的图形，看上去像是甜甜圈。程序截图如下：



创建TorusGeometry时没有必须提供的参数。下表列出的是创建这个几何体时可以指定的参数：

|  |  |  |
| --- | --- | --- |
| 属性 | 是否必须 | 描述 |
| radius | 否 | 设置完整圆环的尺寸，默认值是100。 |
| tube | 否 | 设置管子的半径。该属性的默认值是40。 |
| radialSegments | 否 | 设置沿圆环长度方向分成的段数。默认是8。 |
| tubularSegments | 否 | 设置的是沿圆环宽度方向分成的段数。默认是6。 |
| arc | 否 | 控制是否绘制一个完整的圆环。默认值是2\*PI。 |

大部分都是你已经见过的基础属性。但是其中的arc是一个有趣的属性。通过这个属性你可以指定绘制一个完整的圆环，还是部分圆环。试验一下这个属性，你可以用它来创建出一些很有趣的网格。

完整代码略。

## TorusKnotGeometry

通过TorusKnotGeometry你可以创建一个环面纽结。环面纽结是一种比较特别的结，看上去就像是一根管子绕着它自己转了几圈。程序截图如下：



修改属性p和q，你就可以创建出各种各样漂亮的几何体。p属性定义该结绕其轴多久旋转一次。p属性定义该结绕其内部旋转多少次。你不需要理解这些属性也能创建出漂亮的结（如果对细节感兴趣，Wikipedia上关于这个主题有篇很不错的文章，网址是<http://en.wikipedia.org/wiki/Torus_knot>）。

下表列出的是创建这个几何体时可以指定的参数：

|  |  |  |
| --- | --- | --- |
| 属性 | 是否必须 | 描述 |
| radius | 否 | 这个参数设置的是完整圆环的尺寸，默认值是100。 |
| tube | 否 | 这个参数设置的是管子（真正的甜甜圈）的半径。该属性的默认值是40。 |
| radialSegments | 否 | 这个参数设置的是沿圆环长度方向分成的段数。默认是8。 |
| tubularSegments | 否 | 这个参数设置的是沿圆环宽度方向分成的段数。默认是6。 |
| p | 否 | 该属性定义结的形状。默认值是2。 |
| q | 否 | 该属性定义结的形状。默认值是3。 |
| heightScale | 否 | 通过这个属性你可以拉伸这个环面纽结。默认值是l。 |

完整代码略。

## PolyhedronGeometry

使用这个几何体，可以很容易地创建多面体。多面体是只有平面和直边的几何体。多数情况下，你不会直接使用这种几何体。Three.js提供了几种特定的多面体，你可以直接使用，不必直接设置PolyhedronGeometry的顶点和面。如果你的确想要直接使用PolyhedronGeometry，那么你不得不指定各个顶点和面。例如，要创建一个类似金字塔的多面体：

var vertices = [

 1, 0, 1,

 1, 0, -1,

 -1, 0, -1,

 -1, 0, 1,

 0, 1, 0

];

var indices = [

 0, 2, 1,

 3, 2, 0,

 0, 1, 4,

 1, 2, 4,

 2, 3, 4,

 3, 0, 4

];

polyhedron = createMesh(new THREE.PolyhedronGeometry(vertices, indices, controls.radius, controls.detail));

要创建一个PolyhedronGeometry对象，我们需要传入vertices（顶点）、indices（索引）、radius（半径）和detail参数。

程序截图如下（在type下拉菜单中选择选择Custom）：



当你创建多面体时，可以传入下面的四个属性：

|  |  |  |
| --- | --- | --- |
| 属性 | 是否必须 | 描述 |
| vertices | 是 | 定义构成多面体的顶点，每三个数字表示一个顶点的坐标。 |
| indices | 是 | 指定由vertices的索引创建出的三角形，每三个数字表示一个三角形。 |
| radius | 否 | 该属性指定多面体的大小，默认值是1。 |
| detail | 否 | 通过这个属性可以给这个多面体添加额外的细节。如果设为1，这个多面体上的每个三角形都会分成4个小三角形。如果设为2，那么那些4个小三角形中的每一个都会继续分成4个小三角形，以此类推。 |

Three.js还提供了几个开箱即用的多面体。所有这些多面体都可以在示例中通过选择下拉菜单获得。

### 1．IcosahedronGeometry

通过IcosahedronGeometry（正20面体）可以创建出一个有20个相同三角形面的多面体，这些三角形面是从12个顶点创建出来的。创建这个多面体时，你要做的只是指定radius和detail，如下图所示：



### 2．Tetrahedron

Tetrahedron（正四面体）是最简单的多面体。这个多面体只有4个三角形面，而这些面是从4个顶点创建出来的。创建TetrahedronGeometry跟创建Three.js提供的其他多面体一样，只要指定radius和detail即可，如下图所示：



### 3．Octahedron

Octahedron正八面体。顾名思义，这个多面体有8个面。这些面是从6个顶点中创建出来的，如下图所示：



### 完整代码

<!DOCTYPE html>

<html>

<head>

 <title>示例03.06 - 三维几何体 - 多面体</title>

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

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

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

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

 $(function () {

 var stats = initStats();

 // 创建scene对象，用来容纳网格、相机、光源等对象

 var scene = new THREE.Scene();

 // 创建相机

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

 camera.position.x = -30;

 camera.position.y = 40;

 camera.position.z = 50;

 camera.lookAt(new THREE.Vector3(10, 0, 0));

 // 创建渲染器，并设置视口大小和清除色

 var webGLRenderer = new THREE.WebGLRenderer();

 webGLRenderer.setClearColor(0xEEEEEE, 1.0);

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

 webGLRenderer.shadowMapEnabled = true;

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

 // 添加多面体

 var polyhedron = createMesh(new THREE.IcosahedronGeometry(10, 0));

 scene.add(polyhedron);

 // 创建用户界面

 var controls = new function () {

 console.log(polyhedron.children[0].geometry);

 this.radius = 10;

 this.detail = 0;

 this.type = 'Icosahedron';

 this.redraw = function () {

 // 移除前面创建的多边形

 scene.remove(polyhedron);

 // 创建一个新的多边形

 switch (controls.type) {

 case 'Icosahedron':

 polyhedron = createMesh(new THREE.IcosahedronGeometry(controls.radius, controls.detail));

 break;

 case 'Tetrahedron':

 polyhedron = createMesh(new THREE.TetrahedronGeometry(controls.radius, controls.detail));

 break;

 case 'Octahedron':

 polyhedron = createMesh(new THREE.OctahedronGeometry(controls.radius, controls.detail));

 break;

 case 'Custom':

 var vertices = [

 1, 0, 1,

 1, 0, -1,

 -1, 0, -1,

 -1, 0, 1,

 0, 1, 0

 ];

 var indices = [

 0, 2, 1,

 3, 2, 0,

 0, 1, 4,

 1, 2, 4,

 2, 3, 4,

 3, 0, 4

 ];

 polyhedron = createMesh(new THREE.PolyhedronGeometry(vertices, indices, controls.radius, controls.detail));

 break;

 }

 // 将新的多边形添加到场景中

 scene.add(polyhedron);

 };

 }

 var gui = new dat.GUI();

 gui.add(controls, 'radius', 0, 40).step(1).onChange(controls.redraw);

 gui.add(controls, 'detail', 0, 3).step(1).onChange(controls.redraw);

 gui.add(controls, 'type', ['Icosahedron', 'Tetrahedron', 'Octahedron', 'Custom']).onChange(controls.redraw);

 render();

 function createMesh(geom) {

 // 两个材质

 var meshMaterial = new THREE.MeshNormalMaterial();

 meshMaterial.side = THREE.DoubleSide;

 meshMaterial.shading = THREE.FlatShading;

 var wireFrameMat = new THREE.MeshBasicMaterial();

 wireFrameMat.wireframe = true;

 // 创建一个使用组合材质的网格

 var mesh = THREE.SceneUtils.createMultiMaterialObject(geom, [meshMaterial, wireFrameMat]);

 return mesh;

 }

 var step = 0;

 function render() {

 stats.update();

 polyhedron.rotation.y = step += 0.01;

 requestAnimationFrame(render);

 webGLRenderer.render(scene, camera);

 }

 function initStats() {

 var stats = new Stats();

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

 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>