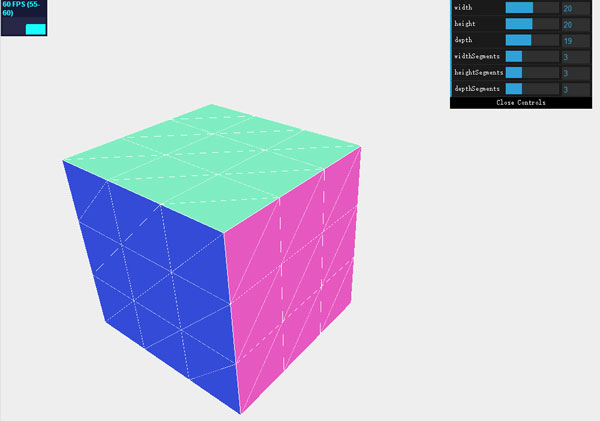
# 3.5 三维几何体——盒、球、圆柱

本文将讨论三维几何体，我们将从一个已经看到过多次的几何体（BoxGeometry）开始。

## BoxGeometry

BoxGeometry是一种非常简单的三维几何体，你只要指定宽度、高度和深度即可创建出一个方块。程序截图如下：



通过改变BoxGeometry对象的width（宽度）、height（高度）和depth（深度），你就可以控制网格的尺寸。这三个属性也是在创建一个方块时必须要提供的，如下所示：

new THREE.CubeGeometry(10,10,10);

你还可以定义方块的其他属性。下表是所有属性的说明：

|  |  |  |
| --- | --- | --- |
| 属性 | 是否必须 | 描述 |
| width | 是 | 定义方块的宽度，即方块沿x轴方向的长度。 |
| height | 是 | 定义方块的高度，即方块沿y轴方向的长度。 |
| depth | 是 | 定义方块的深度，即方块沿z轴方向的长度。 |
| widthSegments | 否 | 定义沿方块的x轴方向将面分成多少份。默认值是1。 |
| heightSegments | 否 | 定义沿方块的y轴方向将面分成多少份。默认值是1。 |
| depthSegments | 否 | 定义沿方块的z轴方向将面分成多少份。默认值是1。 |

通过增加各个分段（segment）属性，你可以将方块的6个大面分成很多小面。这在你用MeshFaceMaterial为方块的不同部分设置特定材质属性时比较有用。

### 完整代码

<!DOCTYPE html>

<html>

<head>

<title>示例03.05 - 三维几何体 - 盒</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 {

margin: 0;

overflow: hidden;

}

</style>

</head>

<body>

<div id="Stats-output">

</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 = -20;

camera.position.y = 30;

camera.position.z = 40;

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;

// 创建一个盒子网格

var cube = createMesh(new THREE.BoxGeometry(10, 10, 10));

scene.add(cube);

// 添加光源

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

spotLight.position.set(-40, 60, -10);

scene.add(spotLight);

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

var step = 0;

// 设置用户界面

var controls = new function () {

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

// 因为使用了是一个组合材质，所以要选取第一个子对象

this.width = cube.children[0].geometry.parameters.width;

this.height = cube.children[0].geometry.parameters.height;

this.depth = cube.children[0].geometry.parameters.depth;

this.widthSegments = cube.children[0].geometry.widthSegments;

this.heightSegments = cube.children[0].geometry.heightSegments;

this.depthSegments = cube.children[0].geometry.depthSegments;

this.redraw = function () {

// 移除先前的网格

scene.remove(cube);

// 创建一个新的网格

cube = createMesh(new THREE.BoxGeometry(controls.width, controls.height, controls.depth, Math.round(controls.widthSegments), Math.round(controls.heightSegments), Math.round(controls.depthSegments)));

scene.add(cube);

};

}

var gui = new dat.GUI();

gui.add(controls, 'width', 0, 40).onChange(controls.redraw);

gui.add(controls, 'height', 0, 40).onChange(controls.redraw);

gui.add(controls, 'depth', 0, 40).onChange(controls.redraw);

gui.add(controls, 'widthSegments', 0, 10).onChange(controls.redraw);

gui.add(controls, 'heightSegments', 0, 10).onChange(controls.redraw);

gui.add(controls, 'depthSegments', 0, 10).onChange(controls.redraw);

render();

function createMesh(geom) {

// 两种材质

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;

}

function render() {

stats.update();

// 使盒子旋转

cube.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;

}

});

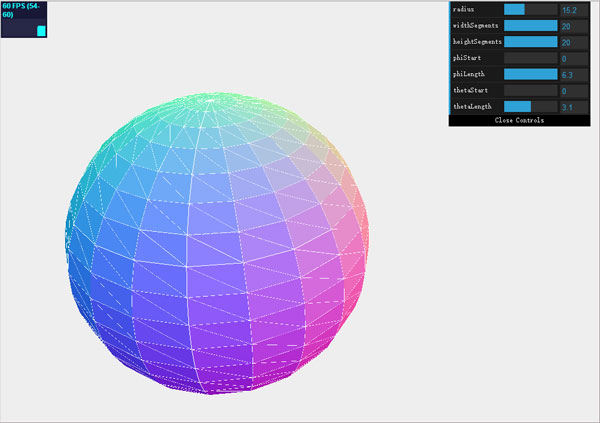
</script>

</body>

</html>

## SphereGeometry

通过SphereGeometry你可以创建一个三维球体。程序截图如下：



在前面这个截图里，我们展示的是半个打开的球，是基于SphereGeometry创建的。这个几何体非常灵活，可以用来创建所有跟球体相关的几何体。

下表中的属性可以用来调节结果网格的外观：

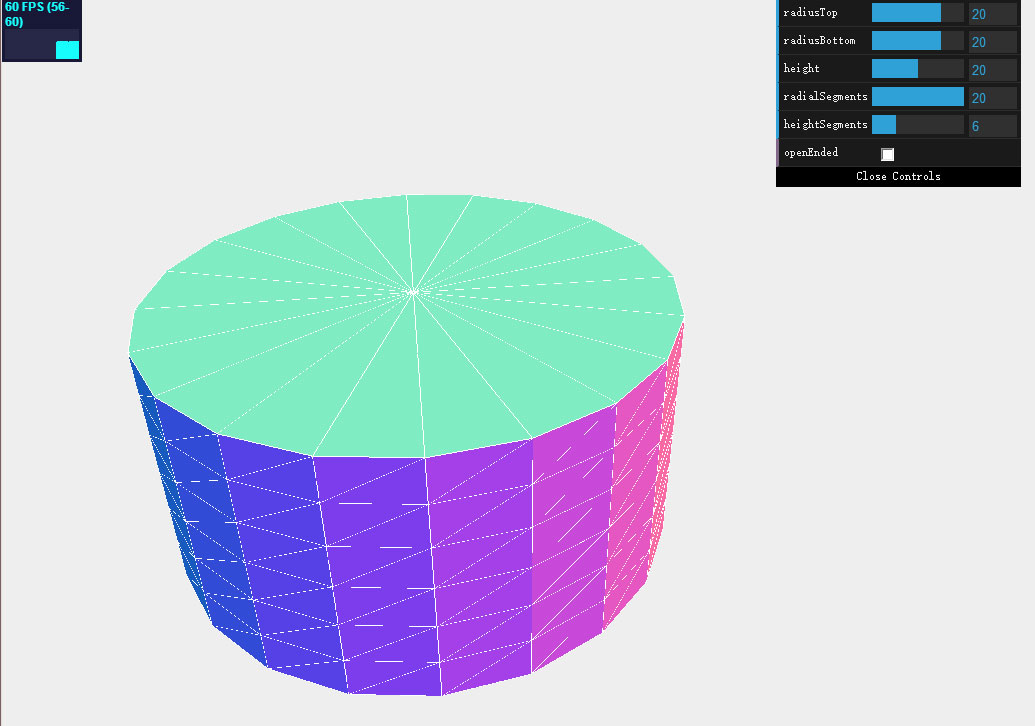
|  |  |  |
| --- | --- | --- |
| 属性 | 是否必须 | 描述 |
| radius | 否 | 设置球体的半径，默认是50。 |
| widthSegments | 否 | 指定竖直方向上的分段数。段数越多，球体的表面越光滑。默认值是8，最小值是3。 |
| heightSegments | 否 | 指定水平方向上的分段数。段数越多，球体的表面越光滑。默认值是6，最小值是2。 |
| phiStart | 否 | 指定从x轴的什么地方开始绘制。取值范围是0到2\*PI，默认是0。 |
| phiLength | 否 | 该属性用来指定从phiStart开始画多少。2\*PI是整球，0.5\*PI画的是一个四分之一球。 |
| thetaStart | 否 | 该属性用来指定从y轴的什么地方开始绘制。取值范围是0到PI，默认是0。 |
| thetaLength | 否 | 该属性用来指定从thetaStart开始画多少。PI是整球，0.5\*PI只会绘制上半球。 |

属性radius、widthSegments和heightSegments应该很清楚了，我们已经在别的例子里看过这些属性了。你可以通过程序示例理解一下属性phiStart、phiLength、thetaStart和thetaLength的用处。

完整代码略。

## CylinderGeometry

通过这个几何体我们可以创建圆柱和类似圆柱的物体。程序截图如下：



创建CylinderGeometry时，没有必须要提供的参数。所以你只要调用new THREE.CylinderGeometry()即可创建出一个圆柱。你也可以提供几个参数，用来修改圆柱的外观。这些属性列在下表中：

|  |  |  |
| --- | --- | --- |
| 属性 | 是否必须 | 描述 |
| radiusTop | 否 | 该属性设置圆柱顶部的尺寸，默认是20。 |
| radiusBottom | 否 | 该属性设置圆柱底部的尺寸，默认是20。 |
| height | 否 | 该属性设置圆柱的高度，默认是100。 |
| radialSegments | 否 | 该属性设置沿x轴分成多少段，默认是8。这个数字越大，圆柱越光滑。 |
| heightSegments | 否 | 该属性设置沿y轴分成多少段，默认是1。分段越多，意味着面越多。 |
| openEnded | 否 | 该属性指定网格的顶部和底部是否封闭。默认是false。 |

这些都是配置圆柱体的基础属性。你可以在顶部（或底部）使用值为负数的半径。如果这么设置，你就可以用这个几何体创建一个类似沙漏的图形。需要注意的是：圆柱的上半部分内外翻转了。如果你用的材质不是设置成THREE.DoubleSide，你就看不到上半部分。

完整代码略。