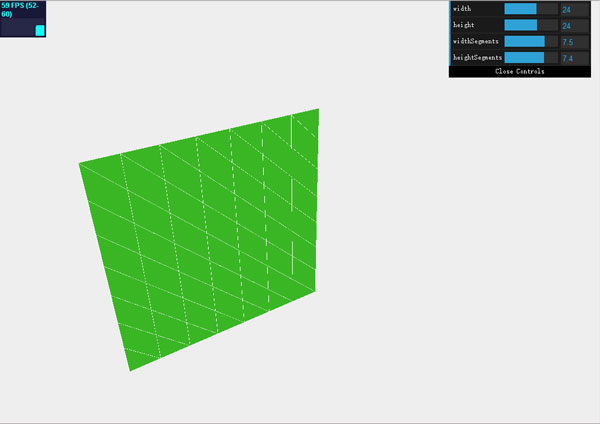
# 3.3 二维几何体——长方形、圆

Three.js中有几种几何体可以创建二维网格：CircleGeometry、PlaneGeometry和ShapeGeometry。二维几何体看上去是扁平的，顾名思义，它们只有两个维度。

## PlaneGeometry

PlaneGeometry可以用来创建非常简单的二维矩形。程序截图如下：



创建这种几何体非常简单，如下所示：

new THREE.PlaneGeometry(width,height,widthSegments,heightSegments);

在这个PlaneGeometry例子中，可以修改它的属性，并立即查看修改的结果。关于这些属性的解释列在下表中：

|  |  |  |
| --- | --- | --- |
| 属性 | 必须 | 描述 |
| width（宽度） | 是 | 矩形的宽度 |
| height（高度） | 是 | 矩形的高度 |
| widthSegments（宽度段数） | 否 | 矩形的宽度应该划分成几段 |
| heightSegmentss（高度段数） | 否 | 矩形的高度应该划分成几段 |

如果你想创建更多的面（比如，一种拥有多个方格的样式），你可以使用widthSegments属性和heightSegments属性，将这个矩形分成多个小面。

该示例中用到的材质需要做一个简短的说明。我们使用如下的方法在这些几何体上创建网格：

function createMesh(geom) {

// 两种材质

var meshMaterial = new THREE.MeshNormalMaterial();

meshMaterial.side = THREE.DoubleSide;

var wireFrameMat = new THREE.MeshBasicMaterial();

wireFrameMat.wireframe = true;

// 创建一个使用组合材质的网格对象

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

return plane;

}

在这个函数里，我们基于传入的几何体创建了一个拥有多个材质的网格。第一个材质是NormaIMaterial，这种材质基于物体表面的法向向量来计算它们的颜色。我们还把这种材质设置成双面的（THREE.DoubleSide）。如果不这么做，那么当物体的背面朝向相机的时候，就会看不到它。我们还添加了一个BasicMatreial材质，通过它来打开线框效果。这样我们就可以看到该物体在三维空间中形状以及该几何体创建的面。

## 完整代码

<!DOCTYPE html>

<html>

<head>

<title>示例03.03.01 - 二维几何体 - 平面</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;

// 添加平面

var plane = createMesh(new THREE.PlaneGeometry(10, 14, 4, 4));

scene.add(plane);

// 添加光源

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

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

scene.add(spotLight);

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

// 设置用户界面

var controls = new function () {

// 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 = plane.children[0].geometry.parameters.width;

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

this.widthSegments = plane.children[0].geometry.parameters.widthSegments;

this.heightSegments = plane.children[0].geometry.parameters.heightSegments;

this.redraw = function () {

// 首先移除先前的平面

scene.remove(plane);

// 根据参数再创建一个新平面

plane = createMesh(new THREE.PlaneBufferGeometry(controls.width, controls.height, Math.round(controls.widthSegments), Math.round(controls.heightSegments)));

scene.add(plane);

};

}

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

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

render();

function createMesh(geom) {

// 两种材质

var meshMaterial = new THREE.MeshNormalMaterial();

meshMaterial.side = THREE.DoubleSide;

var wireFrameMat = new THREE.MeshBasicMaterial();

wireFrameMat.wireframe = true;

// 创建一个使用组合材质的网格对象

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

return plane;

}

var step = 0;

function render() {

stats.update();

plane.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;

}

});

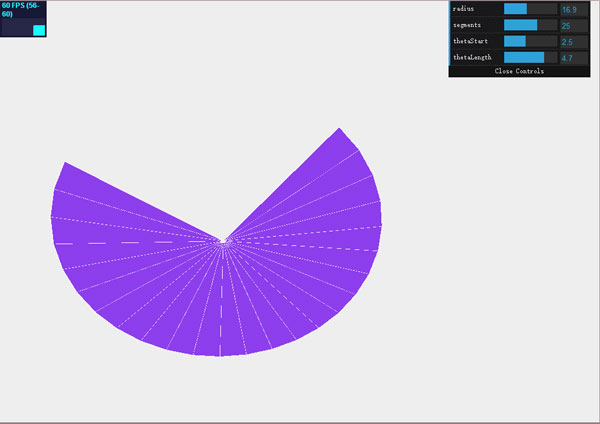
</script>

</body>

</html>

## CircleGeometry

通过这个几何体可以创建出简单的二维圆（或部分圆）。程序截图如下：



在这个示例里你可以观察、控制用CircleGeometry创建的网格。当你创建CircleGeometry对象时，可以指定几个属性，从而决定这个圆的样子。

|  |  |  |
| --- | --- | --- |
| 属性 | 是否必须 | 描述 |
| radius（半径） | 是 | 定义圆的半径，从而决定圆的大小。 |
| segments（分段） | 否 | 定义创建圆所用面的数量。最少3个，如果没有指定则默认8个。值越大，创建出的圆越光滑。 |
| thetaStart（起始角） | 否 | 定义从哪儿开始画圆。取值范围是0到2\*PI。 |
| thetaLength（角度） | 否 | 定义圆要画多大。如果没有指定，则默认为2\*PI（整圆）。如果你指定的是0.5\*PI，那么得到的将是一个四分之一圆。结合thetaStart属性和该属性即可定义圆的形状。 |

从代码角度来说，你可以用以下的代码片段创建出一个完整的圆：

new THREE.CircleGeometry(3,12);

如果你想创建一个半圆，那么你可以使用如下的代码：

new THREE.CircleGeometry(3,12,0,Math.PI);

Three.js创建这些二维对象时只使用x轴和y轴，所以它们都是“直立”的。这很符合逻辑，因为它们是二维对象。但是一般来讲，特别是用PlaneGeometry时，你会希望这些几何体躺下来，以便构建一种地面，好把其他对象放在上面。创建一个水平放置而不是竖直的二维对象最简单的方法就是：将这个网格沿x轴向后旋转四分之一圈（-PI/2）。

mesh.rotation.x=-Math.PI/2;

完整代码略。