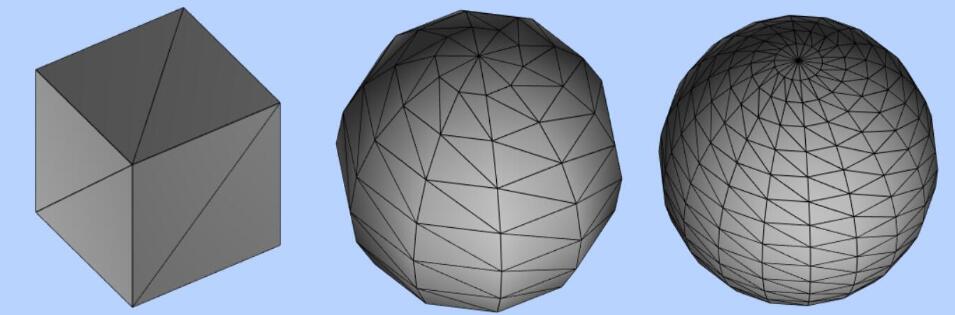
**顶点与几何体**

我们的几何体是由一个个的三角面组成的，而三角面是由3个顶点组成的，可以认为我们的几何体就是由顶点组成的



**BufferGeometry**

BufferGeometry代表一个几何体，我们从外部导入的模型或者前面讲的3角形球形等都是BufferGeometry

如下我们使用BufferGeometry生成一个几何体

function getObject() {

    // 创建一个几何体

    var geometry = new THREE.BufferGeometry();

    // 顶点数据

    var vertices = new Float32Array([

        0, 0, 0, //顶点1坐标

        50, 0, 0, //顶点2坐标

        0, 100, 0, //顶点3坐标

        0, 0, 10, //顶点4坐标

        0, 0, 100, //顶点5坐标

        50, 0, 10, //顶点6坐标

    ]);

    // 创建属性缓冲区对象，3表示3个为一组（即3个数据组成一个顶点）

    var attribue = new THREE.BufferAttribute(vertices, 3);

    // 设置几何体的顶点信息

    geometry.attributes.position = attribue;

    // 网格对象所需材质信息

    var material = new THREE.MeshBasicMaterial({

        color: 0x0000ff, //三角面颜色

        side: THREE.DoubleSide //两面可见

    });

    // 将几何体渲染为网格（即3角面）

var mesh = new THREE.Mesh(geometry, material);

    return mesh;

}

**点、线、网格**

几何体不仅可以生成网格，也可以生成点和线

- 点

    // 点材质

    var material = new THREE.PointsMaterial({

        color: 0xff0000,

        size: 10.0 //点对象像素尺寸

    });

    // 点

    var points = new THREE.Points(geometry, material);

- 线

    // 线条材质

    var material = new THREE.LineBasicMaterial({

        color: 0xff0000 //线条颜色

    });

    // 线

    var line = new THREE.Line(geometry, material);

**顶点颜色**

我们可以给顶点设置颜色，线条或网格渲染模式下，其中间的颜色会呈现过度状态

如下我们给顶点设置颜色

    //类型数组创建顶点颜色color数据

    var colors = new Float32Array([

        1, 0, 0, //顶点1颜色

        0, 1, 0, //顶点2颜色

        0, 0, 1, //顶点3颜色

        1, 1, 0, //顶点4颜色

        0, 1, 1, //顶点5颜色

        1, 0, 1, //顶点6颜色

    ]);

    // 设置几何体attributes属性的颜色color属性

    geometry.attributes.color = new THREE.BufferAttribute(colors, 3); //3个为一组,表示一个顶点的颜色数据RGB

材质颜色会覆盖顶点颜色，所以我们材质需要设置一下

    // 网格对象所需材质信息

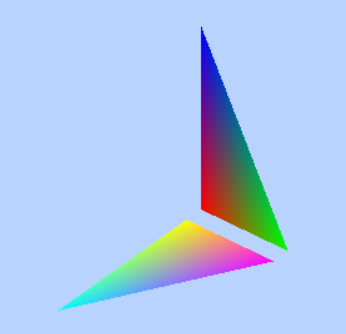
    var material = new THREE.MeshBasicMaterial({

        vertexColors: true, //以顶点颜色为准

        side: THREE.DoubleSide //两面可见

    });

效果如下



**顶点法向量**

顶点法向量影响光照效果，如下我们可以给顶点设置法向量

    var normals = new Float32Array([

        0, 0, 1, //顶点1法向量

        0, 0, 1, //顶点2法向量

        0, 0, 1, //顶点3法向量

        0, 1, 0, //顶点4法向量

        0, 1, 0, //顶点5法向量

        0, 1, 0, //顶点6法向量

    ]);

    // 设置几何体attributes属性的位置normal属性

    geometry.attributes.normal = new THREE.BufferAttribute(normals, 3); //3个为一组,表示一个顶点的法向量数据

只有材质才对光照由反应，所以更改一下材质

    // 网格对象所需材质信息

    var material = new THREE.MeshLambertMaterial({

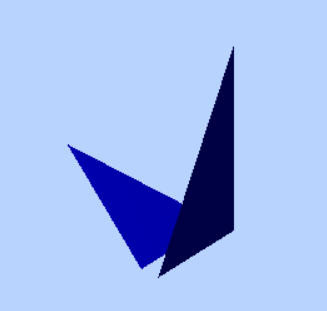
        color: 0x0000ff,

        // vertexColors: true, //以顶点颜色为准

        side: THREE.DoubleSide //两面可见

    });

效果如下



**顶点索引**

如下，我们建一个四边形

    // 创建一个几何体

    var geometry = new THREE.BufferGeometry();

    // 顶点数据

    var vertices = new Float32Array([

        0, 0, 0, //顶点1坐标

        80, 0, 0, //顶点2坐标

        80, 80, 0, //顶点3坐标

        0, 0, 0, //顶点4坐标   和顶点1位置相同

        80, 80, 0, //顶点5坐标  和顶点3位置相同

        0, 80, 0, //顶点6坐标

    ]);

    // 创建属性缓冲区对象，3表示3个为一组（即3个数据组成一个顶点）

    var attribue = new THREE.BufferAttribute(vertices, 3);

    // 设置几何体的顶点信息

    geometry.attributes.position = attribue;

    var normals = new Float32Array([

        0, 0, 1, //顶点1法向量

        0, 0, 1, //顶点2法向量

        0, 0, 1, //顶点3法向量

        0, 0, 1, //顶点4法向量

        0, 0, 1, //顶点5法向量

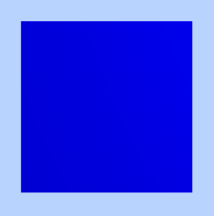
        0, 0, 1, //顶点6法向量

    ]);

    // 设置几何体attributes属性的位置normal属性

    geometry.attributes.normal = new THREE.BufferAttribute(normals, 3); //3个为一组,表示一个顶点的法向量数据

效果如下



可以看到我们需要6的顶点，6个法向量，但实际上有2个顶点是重复的，只需要4的顶点和4个法向量数据就可以了，所以我们引入了顶点索引，代码如下

    // 创建一个几何体

    var geometry = new THREE.BufferGeometry();

    // 顶点数据

    var vertices = new Float32Array([

        0, 0, 0, //顶点1坐标

        80, 0, 0, //顶点2坐标

        80, 80, 0, //顶点3坐标

        0, 80, 0, //顶点4坐标

    ]);

    // 创建属性缓冲区对象，3表示3个为一组（即3个数据组成一个顶点）

    var attribue = new THREE.BufferAttribute(vertices, 3);

    // 设置几何体的顶点信息

    geometry.attributes.position = attribue;

    var normals = new Float32Array([

        0, 0, 1, //顶点1法向量

        0, 0, 1, //顶点2法向量

        0, 0, 1, //顶点3法向量

        0, 0, 1, //顶点4法向量

    ]);

    // 设置几何体attributes属性的位置normal属性

geometry.attributes.normal = new THREE.BufferAttribute(normals, 3); //3个为一组,表示一个顶点的法向量数据

    // Uint16Array类型数组创建顶点索引数据

    var indexes = new Uint16Array([

        // 顶点 0, 1, 2, 构成一个三角形

        0, 1, 2,

        // 顶点 0, 2, 3, 构成一个三角形

        0, 2, 3,

    ])

    // 索引数据赋值给几何体的index属性

    geometry.index = new THREE.BufferAttribute(indexes, 1); //1个为一组

关于Uint16Array：你可以使用Uint8Array，Uint16Array，Uint32Array都可以，但他们占用的内存也会随之增大

**旋转、缩放、平移变换**

我们通过如下方法对几何体进行旋转、缩放、平移

// 旋转

rotateX(angle: number): BufferGeometry;

rotateY(angle: number): BufferGeometry;

rotateZ(angle: number): BufferGeometry;

// 移动

translate(x: number, y: number, z: number): BufferGeometry;

// 缩放

scale(x: number, y: number, z: number): BufferGeometry;