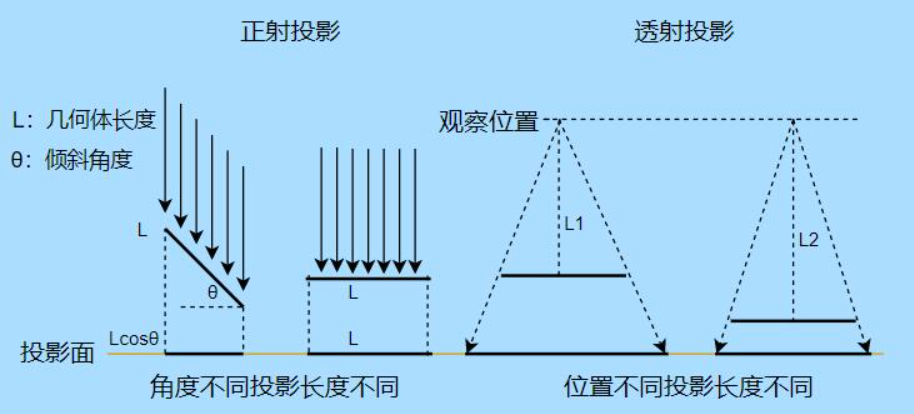
**正投影和透视投影**

正投影投影面大小是固定的，而透视投影投影面大小与观察位置远近有关



**正投影相机对象OrthographicCamera**

var width = window.innerWidth; //窗口宽度

var height = window.innerHeight; //窗口高度

var k = width / height; //窗口宽高比

var s = 150; //三维场景显示范围控制系数，系数越大，显示的范围越大

//创建相机对象

var camera = new THREE.OrthographicCamera(-s \* k, s \* k, s, -s, 1, 1000);

其构造函数如下

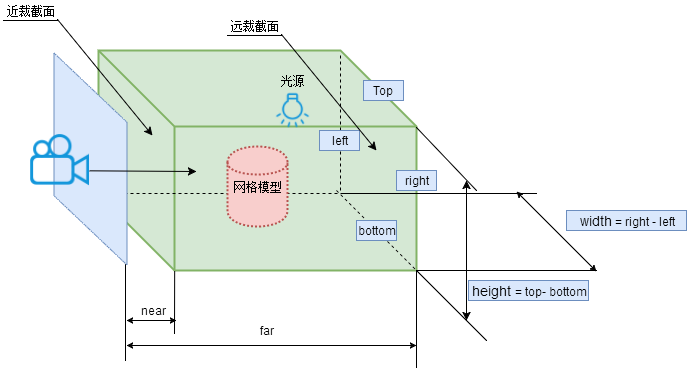
// 构造函数格式

OrthographicCamera( left, right, top, bottom, near, far )

注：相机的宽高比需要与canvas的宽高比一直，否者渲染的图形将会被拉伸

| 参数(属性) | 含义 |
| --- | --- |
| left | 渲染空间的左边界 |
| right | 渲染空间的右边界 |
| top | 渲染空间的上边界 |
| bottom | 渲染空间的下边界 |
| near | near属性表示的是从距离相机多远的位置开始渲染，一般情况会设置一个很小的值。 默认值0.1 |
| far | far属性表示的是距离相机多远的位置截止渲染，如果设置的值偏小小，会有部分场景看不到。 默认值1000 |

这些参数用于控制相机的一个渲染范围，如下超出绿色方体的部分将不会被渲染



**透视投影相机PerspectiveCamera**

/\*\*

 \* 透视投影相机设置

 \*/

 var width = window.innerWidth; //窗口宽度

 var height = window.innerHeight; //窗口高度

 /\*\*透视投影相机对象\*/

 var camera = new THREE.PerspectiveCamera(60, width / height, 1, 1000);

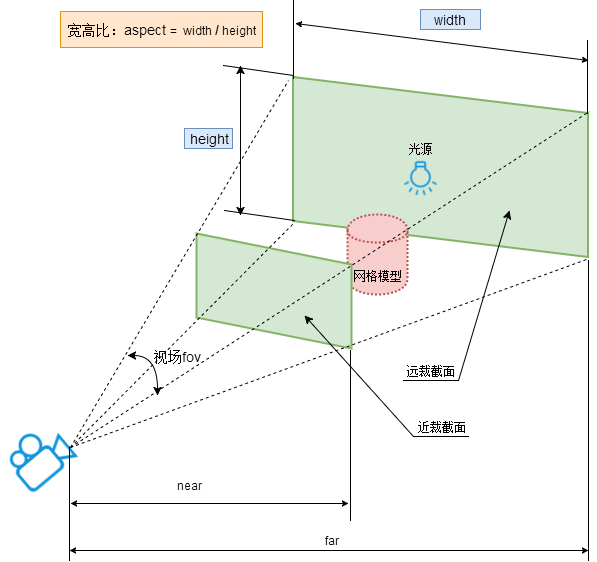
 camera.position.set(200, 300, 200); //设置相机位置

 camera.lookAt(scene.position); //设置相机方向(指向的场景对象)

其构造函数如下

PerspectiveCamera( fov, aspect, near, far )

| 参数 | 含义 | 默认值 |
| --- | --- | --- |
| fov | fov表示视场，所谓视场就是能够看到的角度范围，一般游戏会设置60~90度 | 45 |
| aspect | aspect表示渲染窗口的长宽比，如果一个网页上只有一个全屏的canvas画布且画布上只有一个窗口，那么aspect的值就是网页窗口客户区的宽高比 | window.innerWidth/window.innerHeight |
| near | near属性表示的是从距离相机多远的位置开始渲染，一般情况会设置一个很小的值。 | 0.1 |
| far | far属性表示的是距离相机多远的位置截止渲染，如果设置的值偏小，会有部分场景看不到 | 1000 |



**相机位置.posiiotn和拍摄位置.lookAt**

// 设置相机位置

camera.position.set(100,200,200);

// 设置相机拍摄的点

camera.lookAt(scene.position);

**浏览器窗口变化自适应**

浏览器窗口变化时我们需要重新设置渲染器和相机的渲染范围，不然会出现白色的范围

1. 正投影重新渲染

// onresize 事件会在窗口被调整大小时发生

window.onresize = function () {

    // 重置渲染器输出画布canvas尺寸

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

    // 重置相机投影的相关参数

    k = window.innerWidth / window.innerHeight;//窗口宽高比

    camera.left = -s \* k;

    camera.right = s \* k;

    camera.top = s;

    camera.bottom = -s;

    // 渲染器执行render方法的时候会读取相机对象的投影矩阵属性projectionMatrix

    // 但是不会每渲染一帧，就通过相机的属性计算投影矩阵(节约计算资源)

    // 如果相机的一些属性发生了变化，需要执行updateProjectionMatrix ()方法更新相机的投影矩阵

    camera.updateProjectionMatrix();

};

1. 透视投影重新渲染

window.onresize = function () {

    // 重置渲染器输出画布canvas尺寸

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

    // 全屏情况下：设置观察范围长宽比aspect为窗口宽高比

    camera.aspect = window.innerWidth / window.innerHeight;

    // 渲染器执行render方法的时候会读取相机对象的投影矩阵属性projectionMatrix

    // 但是不会每渲染一帧，就通过相机的属性计算投影矩阵(节约计算资源)

    // 如果相机的一些属性发生了变化，需要执行updateProjectionMatrix ()方法更新相机的投影矩阵

    camera.updateProjectionMatrix();

};