郭隆邦 2023-03-03

2. Raycaster(射线拾取模型)

上节课给大家介绍过射线`Ray`，下面给大家介绍另一个和射线相关的API射线投射器`Raycaster`。

射线投射器`Raycaster`

射线投射器`Raycaster`具有一个射线属性`.ray`，该属性的值就是上节课讲解的射线对象`Ray`。

```
const raycaster = new THREE.Raycaster();
console.log('射线属性',raycaster.ray);
```

```
// 设置射线起点
raycaster.ray.origin = new THREE.Vector3(-100, 0, 0);
// 设置射线方向射线方向沿着x轴
raycaster.ray.direction = new THREE.Vector3(1, 0, 0);
```

射线交叉计算(`.intersectObjects()`方法)

射线投射器`Raycaster`通过`.intersectObjects()`方法可以计算出来与自身射线`.ray`相交的网格模型。

`.intersectObjects([mesh1, mesh2, mesh3])`对参数中的网格模型对象进行射线交叉计算,未选中对象返回空数组[],选中一个对象，数组1个元素，选中多个对象，数组多个元素，如果选中多个对象，对象在数组中按照先后排序。

```
const raycaster = new THREE.Raycaster();
raycaster.ray.origin = new THREE.Vector3(-100, 0, 0);
raycaster.ray.direction = new THREE.Vector3(1, 0, 0);
// 射线发射拾取模型对象
const intersects = raycaster.intersectObjects([mesh1, mesh2, mesh3]);
```

```
console.log("射线器返回的对象", intersects);
```

`.intersectObjects()` 射线拾取返回信息

射线拾取返回的intersects里面的元素包含多种信息，你可以通过threejs文档查看，或者在浏览器控制台打印查看。

`.intersectObjects()` 和 `.intersectObject()` 功能相同，只是具体语法不同，`.intersectObjects()` 返回数组元素包含的信息，可以参考文档关于 `.intersectObject()` 的介绍。

```
console.log("射线器返回的对象", intersects);
// intersects.length大于0说明，说明选中了模型
if (intersects.length > 0) {
    console.log("交叉点坐标", intersects[0].point);
    console.log("交叉对象",intersects[0].object);
    console.log("射线原点和交叉点距离",intersects[0].distance);
}
```

射线选中的模型对象改变材质颜色

```
const intersects = raycaster.intersectObjects([mesh1, mesh2, mesh3]);
if (intersects.length > 0) {
    // 选中模型的第一个模型，设置为红色
    intersects[0].object.material.color.set(0xff0000);
}
```

← 1. 射线Ray

3. 屏幕坐标转标准设备坐标→