郭隆邦 2023-03-10

6. 射线拾取层级模型(模型描边)

前面几节课演示过，通过射线投射器`Raycaster`的`.intersectObjects()`方法可以拾取Mesh模型对象。

如果一个层级模型包含**多个**网格网格模型Mesh，使用`.intersectObjects()`方法拾取的时候，返回的结果默认是层级模型的后代Mesh，没办法整体选中该层级模型。

下面以案例：射线拾取工厂设备添加发光描边，给大家演示如何解决含多Mesh的层级模型如何拾取。

`.intersectObjects()`参数的元素是层级模型

通过Blender查看工厂模型，你可以发现存储罐`cunchu.children`里面有两个子对象**设备A**、**设备B**，这些子对象本身都是由多个Mesh构成的父对象。

执行`.intersectObjects(cunchu.children)`进行射线拾取计算，返回结果并不是**设备A**或**设备B**父对象，而是他们的某个子对象Mesh。

```
const cunchu = model.getObjectByName('存储罐');
// 射线拾取模型对象(包含多个Mesh)
// 射线交叉计算拾取模型
const intersects = raycaster.intersectObjects(cunchu.children);
```

课件源码已经设置好后处理`OutlinePass`,为了方便测试查看，那个Mesh被选中了，你直接给射线选中模型添加发光描边即可。

```
outlinePass.selectedObjects = [intersects[0].object];
```

`.intersectObjects()`拾取层级模型解决方法

`.intersectObjects([父对象A,父对象B...])`

给需要射线拾取父对象的所有子对象Mesh自定义一个属性`.ancestors`，然后让该属性指向需要射线拾取父对象。

```
const cunchu = model.getObjectByName('存储罐');
// 射线拾取模型对象(包含多个Mesh)
// 可以给待选对象的所有子孙后代Mesh，设置一个祖先属性ancestors,值指向祖先(待选对象)
for (let i = 0; i < cunchu.children.length; i++) {
    const group = cunchu.children[i];
    //递归遍历chooseObj，并给chooseObj的所有子孙后代设置一个ancestors属性指向自己
    group.traverse(function (obj) {
        if (obj.isMesh) {
            obj.ancestors = group;
        }
    })
}
// 射线交叉计算拾取模型
const intersects = raycaster.intersectObjects(cunchu.children);
console.log('intersects', intersects);
if (intersects.length > 0) {
    // 通过.ancestors属性判断那个模型对象被选中了
    outlinePass.selectedObjects = [intersects[0].object.ancestors];
}
```

完整代码

鼠标单击选中工厂某个设备，并添加高亮发光描边后处理效果。

```
addEventListener('click', function (event) {
    const px = event.offsetX;
    const py = event.offsetY;
    //屏幕坐标转标准设备坐标
    const x = (px / window.innerWidth) * 2 - 1;
    const y = -(py / window.innerHeight) * 2 + 1;
    const raycaster = new THREE.Raycaster();
    //.setFromCamera()在点击位置生成raycaster的射线ray
    raycaster.setFromCamera(new THREE.Vector2(x, y), camera);
    const cunchu = model.getObjectByName('存储罐');
    // 射线拾取模型对象(包含多个Mesh)
    // 可以给待选对象的所有子孙后代Mesh，设置一个祖先属性ancestors,值指向祖先(待选对象)
    for (let i = 0; i < cunchu.children.length; i++) {
        const group = cunchu.children[i];
        //递归遍历chooseObj，并给chooseObj的所有子孙后代设置一个ancestors属性指向自己
```

```
            group.traverse(function (obj) {
                if (obj.isMesh) {
                    obj.ancestors = group;
                }
            })
        }
        // 射线交叉计算拾取模型
        const intersects = raycaster.intersectObjects(cunchu.children);
        console.log('intersects', intersects);
        if (intersects.length > 0) {
            // 通过.ancestors属性判断那个模型对象被选中了
            outlinePass.selectedObjects = [intersects[0].object.ancestors];
        }
    })
```

← 5. Canvas尺寸变化(射线坐标计算)

7. 射线拾取Sprite控制场景→