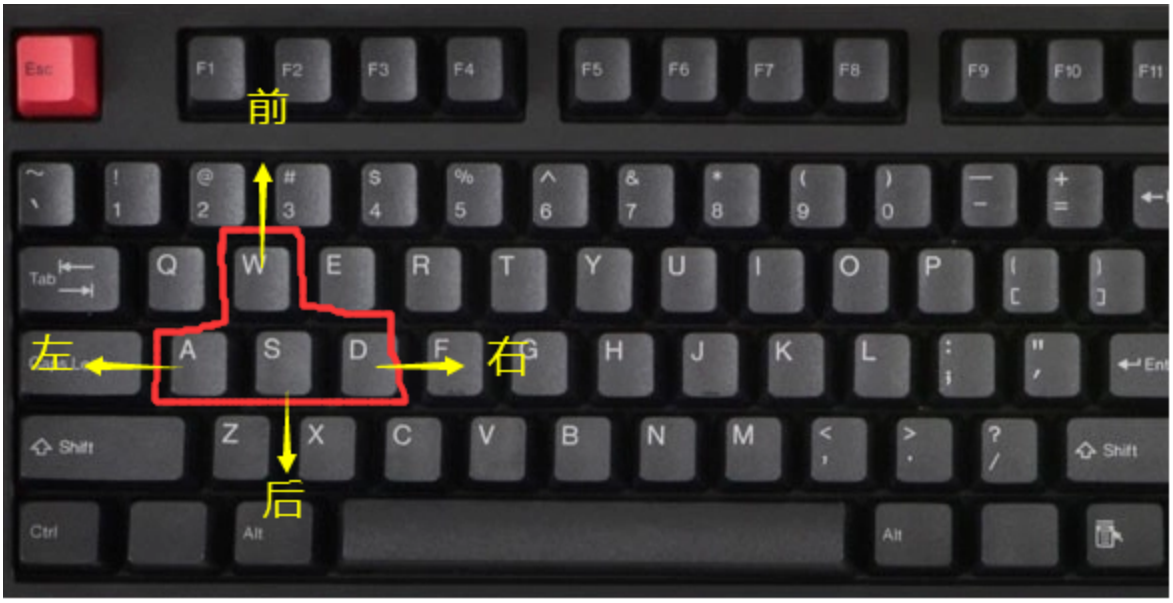
郭隆邦 2023-09-03

10. 玩家角色左右运动(叉乘)

前面给大家讲解过，通过W和S按键控制玩家角色的前后运动，本节课给大家讲解通过A和D键控制玩家的左右运动。



知识点回顾

8.8小节给大家讲解过，执行`player.getWorldDirection(front);`，可以获取玩家角色模型，当前视线正前方方向，用于W键和S键的前后运动控制。

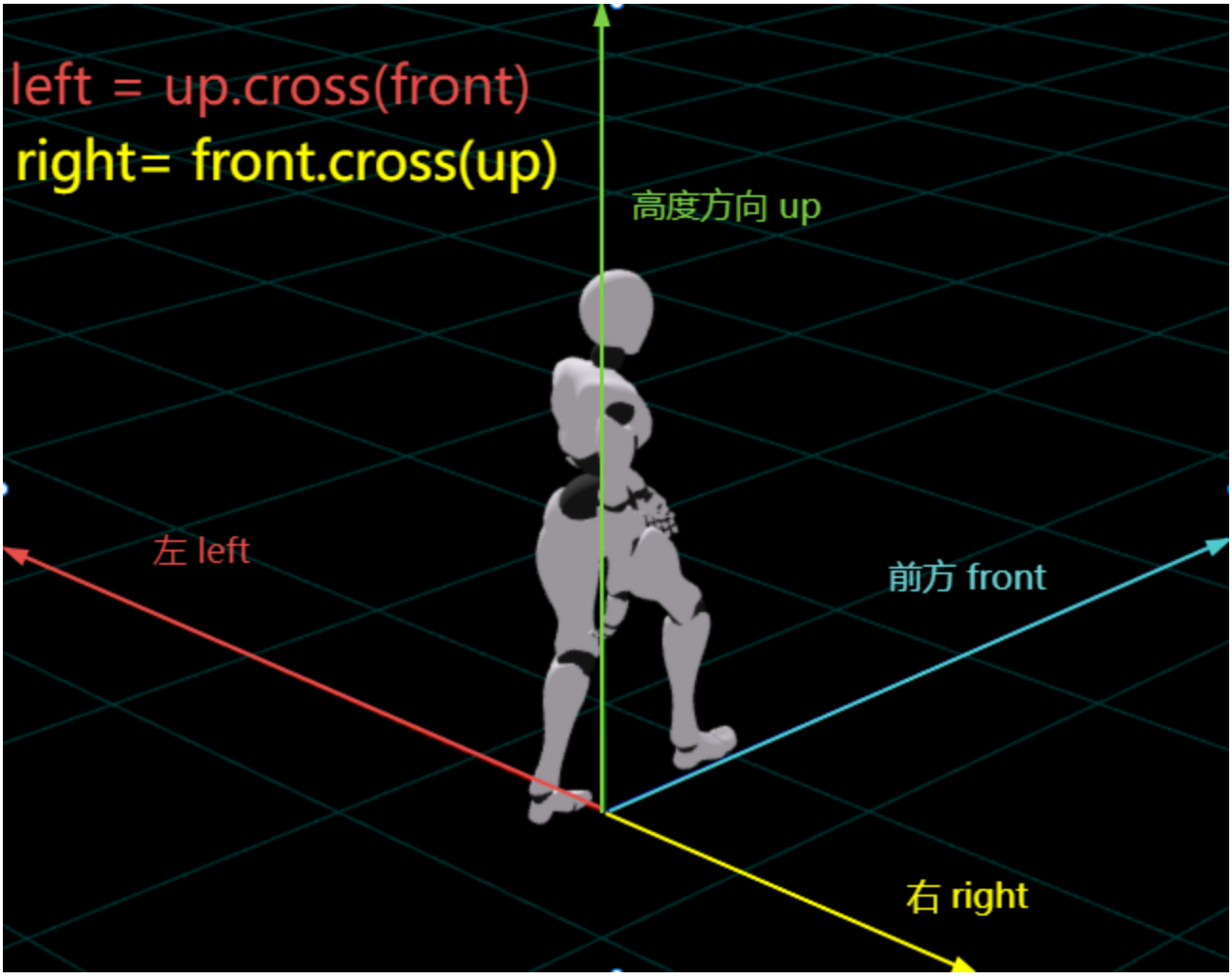
```js
if (keyStates.W) {
    const front = new THREE.Vector3();
    player.getWorldDirection(front);//获取玩家角色(相机)正前方
    v.add(front.multiplyScalar(a * deltaTime));
}
if (keyStates.S) {
    const front = new THREE.Vector3();
    player.getWorldDirection(front);
    // - a：与W按键反向相反
    v.add(front.multiplyScalar(- a * deltaTime));
}
```

叉乘计算左右方向

你可以先回顾下，前面3.5节 向量叉乘 的知识点。

两个向量a、b叉乘有一个特点，叉乘结果是一个同时垂直于a和b的向量。这就是说只要知道玩家角色模型人正前方方向和高度方向向量，就可以计算出来人的左右方向。

小技巧：叉乘获得垂直于向量up和front的向量,左右方向与叉乘顺序有关,左右方向，可以用右手螺旋定责判断，但是比较麻烦，如果你懒得思考，干脆不用右手螺旋定则判断，代码测试测试下最简单，先随便写一个叉乘顺序，如果不对，就把up和front叉乘顺序换下。



玩家角色的正前方

```js
const front = new THREE.Vector3();
player.getWorldDirection(front);
```

玩家角色的高度方向(竖直方向)

```js
const up = new THREE.Vector3(0, 1, 0);//y方向
```

A和D按键对应的方向计算代码。

```js
if (keyStates.A) {//向左运动
    const front = new THREE.Vector3();
    player.getWorldDirection(front);
    const up = new THREE.Vector3(0, 1, 0);//y方向

    const left = up.clone().cross(front);
    v.add(left.multiplyScalar(a * deltaTime));
}
if (keyStates.D) {//向右运动
    const front = new THREE.Vector3();
    player.getWorldDirection(front);
    const up = new THREE.Vector3(0, 1, 0);//y方向
    //叉乘获得垂直于向量up和front的向量 左右与叉乘顺序有关,可以用右手螺旋定则判断，也可以
    const right = front.clone().cross(up);
    v.add(right.multiplyScalar(a * deltaTime));
}
```

← 9. 鼠标上下移动只改变相机视角

11. 鼠标滑动改变视角(指针锁定模式)→