郭隆邦 2023-04-25

3. 点按钮,相机飞行靠近观察设备

继续上节课相机动画的讲解。

实际开发的的时候，一个较大的三维场景，有很多不同的设备或物品，你可能希望通过UI按钮点击切换到不同视角，观察某个区域，或者说放大观察某个特定的物品或设备。

按钮

切换相机位置和视角的按钮

```
<div class="pos">
    <div id="A" class="bu">设备A</div>
    <div id="B" class="bu" style="margin-left: 10px;">设备B</div>
    <div id="car" class="bu" style="margin-left: 10px;">停车场</div>
    <div id="all" class="bu" style="margin-left: 10px;">整体</div>
</div>
```

点击按钮A，相机运动到设备A附近

点击按钮A，相机运动到工厂中设备A附近，同时把相机观察目标，逐渐切换到设备A

```
import TWEEN from '@tweenjs/tween.js';
function render() {
    TWEEN.update();
    requestAnimationFrame(render);
}
render();
```

如果你希望相机移动到场景中某个位置附近，可以在Blender三维建模中，创建一个空对象进行标注，本节课模型用的是原来标注标签的空对象。当然你也可以直接读取某个模型的世界坐标。

获取某个对象世界坐标，作为相机lookAt指向的新目标观察点。

```
const A = model.getObjectByName('设备A标注');
const pos = new THREE.Vector3();
//获取三维场景中某个对象世界坐标
A.getWorldPosition(pos);
```

相机位置相对目标观察点，适当偏移，希望观察的范围大，就距离远一点，希望观察的设备显示效果大，就距离设备近一点。

```
// 向量的x、y、z坐标分别在pos基础上增加30
const pos2 = pos.clone().addScalar(30);
```

相机的位置逐渐改变，相机的观察目标也逐渐改变。

```
// 切换到设备A预览状态
document.getElementById('A').addEventListener('click', function () {
    const A = model.getObjectByName('设备A标注');
    const pos = new THREE.Vector3();
    A.getWorldPosition(pos); //获取三维场景中某个对象世界坐标
    // 相机飞行到的位置和观察目标拉开一定的距离
    const pos2 = pos.clone().addScalar(30);//向量的x、y、z坐标分别在pos基础上增加30
    // 相机从当前位置camera.position飞行三维场景中某个世界坐标附近
    new TWEEN.Tween({
            // 相机开始坐标
            x: camera.position.x,
            y: camera.position.y,
            z: camera.position.z,
            // 相机开始指向的目标观察点
            tx: 0,
            ty: 0,
            tz: 0,
        })
        .to({
            // 相机结束坐标
            x: pos2.x,
            y: pos2.y,
            z: pos2.z,
            // 相机结束指向的目标观察点
            tx: pos.x,
            ty: pos.y,
            tz: pos.z,
        }, 2000)
        .onUpdate(function (obj) {
```

```
            // 动态改变相机位置
            camera.position.set(obj.x, obj.y, obj.z);
            // 动态计算相机视线
            camera.lookAt(obj.tx, obj.ty, obj.tz);
        })
        .start();
})
```

考虑OrbitControls的影响

学下下面内容可以参考前面：6.4. OrbitControls辅助设置相机参数

如果你在项目中使用了相机控件 `OrbitControls` ，希望相机 `looAt()` 指向的目标改变以后，该相机控件让然可以正常使用。需要在动画结束 `.onComplete()` 的时候重新设置 `controls.target` ，或者 `.onUpdate()` 更新 `controls.target` 。

```
.onUpdate(function (obj) {
    ...
    camera.lookAt(obj.tx, obj.ty, obj.tz);
})
.onComplete(function(obj){
    controls.target.set(obj.tx, obj.ty, obj.tz);
    controls.update();
})
```

或者 `.onUpdate()` 中，设置 `controls.target` ，并执行 `controls.update()` ，`OrbitControls` 相机控件内部也会执行相机的 `.looAt()` 方法，完整相机视线重新计算，这样就不用执行 `camera.lookAt(obj.tx, obj.ty, obj.tz)` 。

实际开发，相机目标观察点初始状态，不一定就是坐标原点，再设置动画初始目标观察点的时候，可以直接访问 `controls.target` 的x、y、z属性获取。

```
.onUpdate(function (obj) {
    // 动态改变相机位置
    camera.position.set(obj.x, obj.y, obj.z);
    // 动态计算相机视线
    // camera.lookAt(obj.tx, obj.ty, obj.tz);
    controls.target.set(obj.tx, obj.ty, obj.tz);
    controls.update();
})
```

封装一个相机动画函数

这样所有的按钮点击后，都可以调用该函数。

动画开始的相机位置和目标观察点，不要手写具体数字，通过相机对象`camera.position`和相机控件对象读取`controls.target`，这样不管你点击那个按钮，动画开始状态都是上次相机动画结束的状态。

```
// 相机动画函数，从A点飞行到B点，A点表示相机当前所处状态
// pos: 三维向量Vector3，表示动画结束相机位置
// target: 三维向量Vector3，表示相机动画结束lookAt指向的目标观察点
function createCameraTween(endPos,endTarget){
    new TWEEN.Tween({
        // 不管相机此刻处于什么状态，直接读取当前的位置和目标观察点
        x: camera.position.x,
        y: camera.position.y,
        z: camera.position.z,
        tx: controls.target.x,
        ty: controls.target.y,
        tz: controls.target.z,
    })
    .to({
        // 动画结束相机位置坐标
        x: endPos.x,
        y: endPos.y,
        z: endPos.z,
        // 动画结束相机指向的目标观察点
        tx: endTarget.x,
        ty: endTarget.y,
        tz: endTarget.z,
    }, 2000)
    .onUpdate(function (obj) {
        // 动态改变相机位置
        camera.position.set(obj.x, obj.y, obj.z);
        // 动态计算相机视线
        // camera.lookAt(obj.tx, obj.ty, obj.tz);
        controls.target.set(obj.tx, obj.ty, obj.tz);
        controls.update();//内部会执行.lookAt()
    })
    .start();
}
```

设置设备A、设备B、停车场、整体预览四个按钮对应的相机动画，这样你可以在4个按钮之间，随意切换相机的观察状态。

```
// 切换到设备A预览状态
document.getElementById('A').addEventListener('click', function () {
    const A = model.getObjectByName('设备A标注');
    const pos = new THREE.Vector3();
    A.getWorldPosition(pos); //获取三维场景中某个对象世界坐标
    // 相机飞行到的位置和观察目标拉开一定的距离
    const pos2 = pos.clone().addScalar(30);
    createCameraTween(pos2, controls.target)
})
// 切换到设备B的预览状态
document.getElementById('B').addEventListener('click', function () {
    const B = model.getObjectByName('设备B标注');
    const pos = new THREE.Vector3();
    B.getWorldPosition(pos); //获取三维场景中某个对象世界坐标
    // 相机飞行到的位置和观察目标拉开一定的距离
    const pos2 = pos.clone().addScalar(30);
    // 相机从当前位置camera.position飞行三维场景中某个世界坐标附近
    createCameraTween(pos2, controls.target)
})

// 切换到设备停车场的预览状态
document.getElementById('car').addEventListener('click', function () {
    const car = model.getObjectByName('停车场标注');
    const pos = new THREE.Vector3();
    car.getWorldPosition(pos); //获取三维场景中某个对象世界坐标
    // 相机飞行到的位置和观察目标拉开一定的距离
    const pos2 = pos.clone().addScalar(30);
    // 相机从当前位置camera.position飞行三维场景中某个世界坐标附近
    createCameraTween(pos2, pos)
})

// 相机整体预览对应的位置和观察目标
const cameraPos0 = new THREE.Vector3(202, 123, 125)
const target0 = new THREE.Vector3(0, 0, 0);
// 切换整体预览状态
document.getElementById('all').addEventListener('click', function () {
    // 相机从当前位置camera.position回到整体预览状态
    createCameraTween(cameraPos0, target0)
})
```

← 2. tweenjs相机运动动画

4. 点击设备,相机靠近放大预览→