郭隆邦 2023-03-24

4. 解析外部模型关键帧动画

前面几节课，用到的关键帧动画，是借助threejs提供的两个类`KeyframeTrack`、`AnimationClip`自己写代码实现。不过实际开发的时候，很多时候会用三维建模软件，比如Blender，生成关键帧动画，导出包含动画的模型文件，加载模型后,你只需要播放关键帧动画，而不用手写代码创建关键帧动画。

下面就给大家讲解，如果加载解析外部模型文件中的关键帧动画数据。

课件源码中提供了一个美术用Blender编辑好的关键帧动画模型文件，你可以查看预览。

关键帧动画模型的父对象作为播放器`AnimationMixer`参数

前面讲解过，如果你想播放一个模型的关键帧动画，需要把模型作为播放器`AnimationMixer`的参数。

```js
//包含关键帧动画的模型对象作为AnimationMixer的参数创建一个播放器mixer
const mixer = new THREE.AnimationMixer(mesh);
```

即便你把`mesh`的父对象`group`作为播放器`AnimationMixer`的参数，播放器也能根据`KeyframeTrack`参数1包含的模型名字`.name`确定关键帧动画对应的模型对象。

```js
mesh.name = "Box";
const group = new THREE.Group();
group.add(mesh);
const posKF = new THREE.KeyframeTrack('Box.position', times, values);
const clip = new THREE.AnimationClip("test",6,[posKF]);
//包含关键帧动画的模型对象作为AnimationMixer的参数创建一个播放器mixer
const mixer = new THREE.AnimationMixer(group);
```

查看gltf模型动画数据

一般实际开发的时候，在三维建模软件中，创建生成动画相关数据，然后可以导出gltf、fbx等可以包含动画的文件，最后程序员通过threejs代码加载模型、解析模型包含的动画数据，下面

就以gltf模型文件为例给大家演示。

加载gltf模型，如果存在帧动画数据的话，可以通过加载返回gltf对象的动画属性`.animations`获取。

```js
const loader = new GLTFLoader();
loader.load("../工厂.glb", function (gltf) {
    console.log('控制台查看gltf对象结构', gltf);
    console.log('动画数据', gltf.animations);
})
```

`gltf.animations`是一个数组，如果没有帧动画数据，就是一个空数组，有帧动画数据的情况下，里面可能1个或多个Clip动画对象`AnimationClip`。

播放`AnimationClip`动画

```js
loader.load("../工厂.glb", function (gltf) {
    console.log('控制台查看gltf对象结构', gltf);
    // console.log('动画数据', gltf.animations);
    model.add(gltf.scene);

    //包含关键帧动画的模型作为参数创建一个播放器
    const mixer = new THREE.AnimationMixer(gltf.scene);
    //  获取gltf.animations[0]的第一个clip动画对象
    const clipAction = mixer.clipAction(gltf.animations[0]); //创建动画clipAction
    clipAction.play(); //播放动画

    // 如果想播放动画,需要周期性执行`mixer.update()`更新AnimationMixer时间数据
    const clock = new THREE.Clock();
    function loop() {
        requestAnimationFrame(loop);
        //clock.getDelta()方法获得loop()两次执行时间间隔
        const frameT = clock.getDelta();
        // 更新播放器相关的时间
        mixer.update(frameT);
    }
    loop();
})
```

下面是在渲染循环中更新播放器时间。

```js
let mixer = null; //声明一个播放器变量
loader.load("../工厂.glb", function (gltf) {
    model.add(gltf.scene);
     //包含帧动画的模型作为参数创建一个播放器
     mixer = new THREE.AnimationMixer(gltf.scene);
    //  获取gltf.animations[0]的第一个clip动画对象
     const clipAction = mixer.clipAction(gltf.animations[0]);//创建动画clipAction
     clipAction.play();//播放动画
})

 // 创建一个时钟对象Clock
 const clock = new THREE.Clock();
 function render() {
     requestAnimationFrame(render);
     if (mixer !== null) {
         //clock.getDelta()方法获得两帧的时间间隔
         // 更新播放器相关的时间
         mixer.update(clock.getDelta());
     }
 }
 render();
```

动画是否循环播放

人走路、跑步美术美术一般设置很短时间运动，如果你想一直看到运动动作，不用设置非循环。

```js
//不循环播放
clipAction.loop = THREE.LoopOnce;
// 物体状态停留在动画结束的时候
clipAction.clampWhenFinished = true
```

← 3. 动画播放(拖动任意时间状态)

5. 机械虚拟装配案例(播放)→