郭隆邦 2023-03-22

1. 关键帧动画

打开课件案例源码，你可以看到物体从一个位置移动到另一个位置的动画效果，移动过程中也出现过颜色变化。

课件源码效果具体描述，就是**0~3秒**物体逐渐从坐标**原点**移动端**x轴**100位置，然后**3~6秒**物体逐渐从**x轴**100移动到**z轴**100位置，同时**2~5秒**时间内，把物体从**红色**逐渐改变为**蓝色**。

关键帧动画解释

所谓关键帧动画，你可以理解为在时间轴上，选择几个关键的时间点，然后分别定义这几个时间点对应物体状态(比如位置、姿态、颜色等)，然后基于几个关键的**时间——状态**数据，生成连续的动画。

课件源码**位置**关键帧数据(**时间——状态**)

- 0秒：坐标原点
- 3秒：x轴上100坐标
- 6秒：z轴上100坐标

课件源码**颜色**关键帧数据(**时间——状态**)

- 2秒：红色
- 5秒：蓝色

1. 创建关键帧动画 `AnimationClip`

- 1.1 给需要设置关键帧动画的模型命名
- 1.2 设置关键帧数据 `KeyframeTrack`
- 1.3 基于关键帧数据 `KeyframeTrack` ，创建关键帧动画 `AnimationClip`

```js
// 给需要设置关键帧动画的模型命名
mesh.name = "Box";
const times = [0, 3, 6]; //时间轴上，设置三个时刻0、3、6秒
// times中三个不同时间点，物体分别对应values中的三个xyz坐标
const values = [0, 0, 0, 100, 0, 0, 0, 0, 100];
```

```
// 0~3秒，物体从(0,0,0)逐渐移动到(100,0,0),3~6秒逐渐从(100,0,0)移动到(0,0,100)
const posKF = new THREE.KeyframeTrack('Box.position', times, values);
// 从2秒到5秒，物体从红色逐渐变化为蓝色
const colorKF = new THREE.KeyframeTrack('Box.material.color', [2, 5], [1, 0, 0,
// 1.3 基于关键帧数据，创建一个clip关键帧动画对象，命名"test"，持续时间6秒。
const clip = new THREE.AnimationClip("test", 6, [posKF, colorKF]);
```

1.1 模型命名

你如果你想给一个模型对象设置关键帧动画，比如一个网格模型mesh、一个层级模型group，模型需要有一个名字，没有的话，可以通过`.name`属性命名。

```
mesh.name = "Box";
```

1.2 `KeyframeTrack`设置关键帧数据

`KeyframeTrack`参数1是一个字符串，字符串内容是模型对象的**名字.属性**构成，比如`Box.position`表示模型位置，比如`Box.material.color`表示模型颜色，参数2是时间轴上取的几个关键帧时间点，参数3是时间点对应的物体状态。

位置关键帧数据(**时间——状态**)

- 0秒：坐标原点
- 3秒：x轴上100坐标
- 6秒：z轴上100坐标

```
// 给名为Box的模型对象的设置关键帧数据KeyframeTrack
const times = [0, 3, 6]; //时间轴上，设置三个时刻0、3、6秒
// times中三个不同时间点，物体分别对应values中的三个xyz坐标
const values = [0, 0, 0, 100, 0, 0, 0, 0, 100];
// 创建关键帧，把模型位置和时间对应起来
// 0~3秒，物体从(0,0,0)逐渐移动到(100,0,0),3~6秒逐渐从(100,0,0)移动到(0,0,100)
const posKF = new THREE.KeyframeTrack('Box.position', times, values);
```

颜色关键帧数据(**时间——状态**)

- 2秒：红色
- 5秒：蓝色

```js
// 从2秒到5秒，物体从红色逐渐变化为蓝色
const colorKF = new THREE.KeyframeTrack('Box.material.color', [2, 5], [1, 0, 0,
```

1.3 创建关键帧动画 `AnimationClip`

基于关键帧数据 `KeyframeTrack` ，创建关键帧动画 `AnimationClip` ，这样就可以利用关键帧里面的数据生成一个关键帧动画，用于接下来的动画播放。

下面代码基于关键帧数据posKF、colorKF，创建一个clip关键帧动画对象 `AnimationClip` ，命名为 `test` ，动画持续时间6秒。

```js
// 1.3 AnimationClip表示一个关键帧动画，可以基于关键帧数据产生动画效果
// 创建一个clip关键帧动画对象，命名"test"，动画持续时间6s
// AnimationClip包含的所有关键帧数据都放到参数3数组中即可
const clip = new THREE.AnimationClip("test",6,[posKF, colorKF]);
```

2.1 `AnimationMixer` 播放关键帧动画 `AnimationClip`

前面代码,已经编辑好一个模型mesh的关键帧动画 `AnimationClip` ,如果你想播放动画，就要借助播放器 `AnimationMixer` 。

```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);
```

执行播放器 `AnimationMixer` 的 `.clipAction()` 方法返回一个 `AnimationAction` 对象, `AnimationAction` 对象用来控制如何播放，比如 `.play()` 方法。

```js
//AnimationMixer的`.clipAction()`返回一个AnimationAction对象
const clipAction = mixer.clipAction(clip);
//.play()控制动画播放，默认循环播放
clipAction.play();
```

2.2 `mixer.update()` 更新播放器 `AnimationMixer` 时间

如果想播放动画开始变化,需要周期性执行`mixer.update()`更新播放器`AnimationMixer`时间数据，比如你可以在`requestAnimationFrame`创建的可以周期性执行的函数中，更新播放器时间数据。

```js
function loop() {
    requestAnimationFrame(loop);
}
loop();
```

通过`Clock`对象辅助获取每次loop()执行的时间间隔。

```js
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();
}
loop();
```

执行`mixer.update()`更新播放器`AnimationMixer`时间数据

```js
function loop() {
    requestAnimationFrame(loop);
    const frameT = clock.getDelta();
    // 更新播放器相关的时间
    mixer.update(frameT);
}
loop();
```

如果你不想用`requestAnimationFrame`重新创建一个循环执行函数，也可以在index.js文件渲染循环中引入`mixer.update()`的代码

← 12. Sprite标签(Canvas作为贴图)　　2. 动画播放(暂停、倍速、循环)→