郭隆邦 2023-06-22

4. 四元数乘法运算

下面给大家介绍四元数的**乘法运算**，Three.js四元数`Quaternion`提供了多个用于四元数乘法运算的方法，比如`.multiply()`、`.multiplyQuaternions()`、`.premultiply()`，这些方法本质上都一样，只是语法细节不同，本节课以`.multiply()`为例给大家讲解。

四元数乘法`.multiply()`含义

两个四元数分别表示一个旋转，如果相乘，会得到一个新的四元数，新四元数表示两个旋转的组合旋转。

```
// 在物体原来姿态基础上，进行旋转
const q1 = new THREE.Quaternion();
q1.setFromAxisAngle(new THREE.Vector3(1, 0, 0), Math.PI / 2);
fly.quaternion.multiply(q1);
// 在物体上次旋转基础上，进行旋转
const q2 = new THREE.Quaternion();
q2.setFromAxisAngle(new THREE.Vector3(0, 1, 0), Math.PI / 2);
fly.quaternion.multiply(q2);
// 在物体上次旋转基础上，进行旋转
const q3 = new THREE.Quaternion();
q3.setFromAxisAngle(new THREE.Vector3(0, 0, 1), Math.PI / 2);
fly.quaternion.multiply(q3);
```

四元数乘法顺序

四元数乘法不满足交换律，`q1.clone().multiply(q2)`和`q2.clone().multiply(q1)`表示的旋转结果不同。

```
// 在物体原来姿态基础上，进行旋转
const q1 = new THREE.Quaternion();
q1.setFromAxisAngle(new THREE.Vector3(1, 0, 0), Math.PI / 2);
fly.quaternion.multiply(q1);
// 在物体上次旋转基础上，进行旋转
const q2 = new THREE.Quaternion();
```

```
q2.setFromAxisAngle(new THREE.Vector3(0, 1, 0), Math.PI / 2);
fly.quaternion.multiply(q2);
```

先变换q1,后变换q2，和上面代码效果一样

```
const q1 = new THREE.Quaternion();
q1.setFromAxisAngle(new THREE.Vector3(1, 0, 0), Math.PI / 2);
const q2 = new THREE.Quaternion();
q2.setFromAxisAngle(new THREE.Vector3(0, 1, 0), Math.PI / 2);
const newQ= q1.clone().multiply(q2);
fly.quaternion.multiply(newQ);
```

先变换q2,后变换q1，和上面代码效果不一样，`q2.clone().multiply(q1)`与`q1.clone().multiply(q2)`的表示旋转过程顺序不同

```
// 先变换q2,后变换q1，和上面代码效果不一样，
// q2.clone().multiply(q1)与q1.clone().multiply(q2)表示的旋转过程顺序不同
const q1 = new THREE.Quaternion();
q1.setFromAxisAngle(new THREE.Vector3(1, 0, 0), Math.PI / 2);
const q2 = new THREE.Quaternion();
q2.setFromAxisAngle(new THREE.Vector3(0, 1, 0), Math.PI / 2);
const newQ= q2.clone().multiply(q1);
fly.quaternion.multiply(newQ);
```

`.multiply()`与`.copy()`总结

`A.multiply(B)`表示A乘以B，结果赋值给A，在A的基础上旋转B。

`A.copy(B)`表示用B的值替换A的值，A表示的旋转会被B替换。

可以先通过欧拉角改变物体的姿态，先物体一个初始的角度状态。

```
//改变物体欧拉角，四元数属性也会同步改变
fly.rotation.x = Math.PI/2;
```

创建一个四元数表示一个旋转过程。

```
const quaternion = new THREE.Quaternion();
quaternion.setFromAxisAngle(new THREE.Vector3(0, 0, 1), Math.PI / 2);
```

执行`fly.quaternion.copy(quaternion)`，参数`quaternion`表示的旋转会完全覆盖已有的旋转`fly.quaternion`。无论物体原来的姿态角度是什么样，都会被参数`quaternion`表示新的姿态角度覆盖。

```
//quaternion表示旋转角度复制给物体.quaternion
fly.quaternion.copy(quaternion);
```

`.quaternion.multiply(quaternion)`表示在自身已有旋转的基础上，增加参数quaternion表示的旋转。

```
fly.quaternion.multiply(quaternion);
```

← 3. 四元数表示物体姿态

5. 四元数表示两个向量旋转→