threejs的几个常用组件

轨道控制器, OrbitControls

允许用户通过鼠标、触摸屏或键盘操作来旋转、缩放和平移相机，从而从不同角度观察场景。

```
import { OrbitControls } from "@lib/three-r173/jsm/controls/OrbitControls.js";

  // 创建控制器
  const controls = new OrbitControls(camera, renderer.domElement);
  // // 禁止选中
  controls.enableRotate = false;
  // // 将平移从右键修改为左键
  controls.mouseButtons.LEFT = THREE.MOUSE.RIGHT;

  // 启用触摸控制
  controls.touches.ONE = THREE.TOUCH.PAN; // 单指触摸用于平移
  controls.touches.TWO = THREE.TOUCH.DOLLY_ROTATE; // 双指触摸用于缩

  // 最后，需要在动画循环中一直更新
  // controls.update();
```

可以设置宽度线条

MeshLine 和 MeshLineMaterial 是用于在 three.js 中创建具有宽度和复杂效果的线条的工具，它们通常一起使用，以替代 three.js 内置的 THREE.Line，因

```
import { MeshLine, MeshLineMaterial } from "three.meshline";

// 创建边框
function createThickLine(points, color, lineWidth) {
  var geometry = new THREE.BufferGeometry();
  geometry.setFromPoints(points.map((point) => new THREE.Vector3(point.x, point.y, 0)));

  var line = new MeshLine();
  line.setGeometry(geometry);
```

```
  var material = new MeshLineMaterial({
    color: color,
    lineWidth: lineWidth,
    transparent: true,
    opacity: 1.0,
    blending: THREE.NormalBlending,
    depthTest: false,
    depthWrite: false
  });

  var mesh = new THREE.Mesh(line.geometry, material);
  return mesh;
}



  // 绘制边框
  var line = createThickLine(points, 0x0099ff, 0.05);
  line.position.set(object.x * scale, -(object.y * scale), 0);
  line.renderOrder = 11;
  group.add(line);
```

射线发射器

用于检测鼠标、触摸或其他交互事件与场景中物体的交点。它通过发射光线（Ray）并与场景中的几何体进行相交检测，从而实现诸如鼠标悬停高亮、点击选中物体等功能

```
  // 创建射线投射器
  var raycaster = new THREE.Raycaster();
  var mouse = new THREE.Vector2();
  // 监听鼠标点击事件
  window.addEventListener("click", onMouseClick, false);


// 监听鼠标事件
function onMouseClick(event) {

    let mouse = _mouse;
    let raycaster = _raycaster;
```

```
    let camera = _camera;
    let scene = _scene;

    // □□□□□□□□□[-1, 1]□□
    mouse.x = (event.clientX / window.innerWidth) * 2 - 1;
    mouse.y = -(event.clientY / window.innerHeight) * 2 + 1;

    // □□□□□□□
    raycaster.setFromCamera(mouse, camera);

    // □□□□□□□□□□□□□□
    var intersects = raycaster.intersectObjects(scene.children, true);

    // □□□□□□□□□□□□□
    if (intersects.length > 0) {
        intersects.forEach(intersected => {
           // □□□□□□
        });
    }
}
```

□□□ #3
□ □□□ □□□ 18 □□ 2025 14:25:37
□ □□□ □□□ 18 □□ 2025 14:38:14