郭隆邦 2023-01-28

3. 加载.gltf文件(模型加载全流程)

本节课，以gltf格式为例，给大家讲解加载外部三维模型的整个过程。

场景、光源、渲染器、相机控件等前面说过基础代码，本节课不专门讲解，主要是把下面三部，给大家全流程演示一遍。

1. gltf模型加载器 `GLTFLoader.js`
2. 相机参数根据需要设置
3. 加载gltf的时候，webgl渲染器编码方式设置

1.1.引入 `GLTFLoader.js`

你在three.js官方文件的**examples/jsm/**子文件**loaders/**目录下，可以找到一个文件 `GLTFLoader.js` ，这个文件就是three.js的一个扩展库，专门用来加载gltf格式模型加载器。

```js
// 引入gltf模型加载库GLTFLoader.js
import { GLTFLoader } from 'three/addons/loaders/GLTFLoader.js';
```

1.2.gltf加载器 `new GLTFLoader()`

执行 `new GLTFLoader()` 就可以实例化一个gltf的加载器对象。

```
// 创建GLTF加载器对象
const loader = new GLTFLoader();
```

1.3.gltf加载器方法 `.load()`

通过gltf加载器方法 `.load()` 就可以加载外部的gltf模型。

执行方法 `.load()` 会返回一个gltf对象，作为参数2函数的参数，改gltf对象可以包含模型、动画等信息，本节课你只需要先了解gltf的场景属性 `gltf.scene` ,该属性包含的是模型信息，比如几何体BufferGometry、材质Material、网格模型Mesh。

```
loader.load( 'gltf模型.gltf', function ( gltf ) {
  console.log('控制台查看加载gltf文件返回的对象结构',gltf);
  console.log('gltf对象场景属性',gltf.scene);
  // 返回的场景对象gltf.scene插入到threejs场景中
  scene.add( gltf.scene );
})
```

相机选择(正投影 `OrthographicCamera` 和透视投影 `PerspectiveCamera`)

如果你想预览一个三维场景，一般有**正投影相机** `OrthographicCamera` 和**透视投影相机** `PerspectiveCamera` 可供选择。不过大部分3D项目，比如一般都是使用**透视投影相机** `PerspectiveCamera` ，比如游戏、物联网等项目都会选择**透视投影相机** `PerspectiveCamera` 。

如果你希望渲染的结果符合人眼的**远小近大**的规律，毫无疑问要选择**透视**投影相机，如果不需要模拟人眼远小近大的投影规律，可以选择**正投影**相机。

尺寸概念

项目开发的时候，程序员对一个模型或者说一个三维场景要有一个尺寸的概念，不用具体值，要有一个大概印象。

一般通过三维建模软件可以轻松测试测量模型尺寸，比如作为程序员你可以用三维建模软件blender打开gltf模型，测量尺寸。

单位问题

three.js的世界并没有任何单位，只有数字大小的运算。

obj、gltf格式的模型信息只有尺寸，并不含单位信息。

不过实际项目开发的时候，一般会定义一个单位，一方面甲方、前端、美术之间更好协调，甚至你自己写代码也要有一个尺寸标准。比如一个园区、工厂，可以m为单位建模，比如建筑、人、相机都用m为尺度去衡量，如果单位不统一，就需要你写代码,通过 `.scale` 属性去缩放。

设置合适的相机参数

通过gltf加载完成，模型后，你还需要根据自身需要，设置合适的相机参数，就好比你拍照，你想拍摄一个石头，肯定要把相机对着石头，如果希望石头在照片上占比大，就要离石头近一些。

相机位置怎么设置，你就类比你的眼睛，如果你想模拟人在3D场景中漫游，那么很简单，你把相机放在地面上，距离地面高度和人身高接近即可。

如果你想看到工厂的全貌，你可以理解为你坐着无人机向下俯瞰，简单说，相比人漫游工厂，整体预览工厂相机距离工厂距离更远一些，否则你也看不到全貌，当然过于远了，你就看不清工厂了。

以课程工厂为例，先设定一个小目标，我们希望工厂能够居中显示在canvas画布上，并且保证可以整体预览。

下面以**透视投影相机** `PerspectiveCamera` 为例说明。

2.1.相机位置 `.position`

工厂尺寸范围大概200米数量级，那么如果想整体预览观察工厂所有模型，那很简单，第一步，把 `camera.position` 的xyz值统统设置为几百即可，比如 `(200, 200, 200)` 。

具体xyz值，你可以通过OrbitControls可视化操作调整，然后浏览器控制台记录相机参数即可。

```
camera.position.set(200, 200, 200);
```

2.2 某位置在canvas画布居中

你需要工厂那个位置在canavs画布上居中，直接把 `camera.lookAt()` 指向哪个坐标。

如果美术建模，把工厂整体居中，也就是说模型的几何中心，大概位于世界坐标原点。你设置 `camera.lookAt(0,0,0)` ,相机视线指向坐标原点。

```
camera.lookAt(0, 0, 0);
```

注意相机控件OrbitControls会影响lookAt设置，注意手动设置OrbitControls的目标参数

```
camera.lookAt(100, 0, 0);
```

```
// 设置相机控件轨道控制器OrbitControls
const controls = new OrbitControls(camera, renderer.domElement);
// 相机控件.target属性在OrbitControls.js内部表示相机目标观察点，默认0,0,0
// console.log('controls.target', controls.target);
controls.target.set(100, 0, 0);
controls.update();//update()函数内会执行camera.lookAt(controls.targe)
```

2.3.远裁截面 far 参数

近裁截面near和远裁截面far，要能包含你想渲染的场景，否则超出视锥体模型会被剪裁掉，简单说near足够小，far足够大，主要是far。

```
PerspectiveCamera(fov, aspect, near, far)
```

测量工厂尺寸大概几百的数量级，这里不用测具体尺寸，有个大概数量级即可，然后far设置为3000足够了。

```
const camera = new THREE.PerspectiveCamera(30, width / height, 1, 3000);
```

3.纹理贴图颜色偏差解决

three.js加载gltf模型的时候，可能会遇到three.js渲染结果颜色偏差，对于这种情况，你只需要修改WebGL渲染器默认的编码方式 .outputEncoding 即可

```
//解决加载gltf格式模型纹理贴图和原图不一样问题
renderer.outputEncoding = THREE.sRGBEncoding;
```

注意！！！！！！！最新版本属性名字有改变。渲染器属性名 .outputEncoding 已经变更为 .outputColorSpace 。

查WebGL渲染器文档，你可以看到 .outputColorSpace 的默认值就是SRGB颜色空间 THREE.SRGBColorSpace ，意味着新版本代码中，加载gltf，没有特殊需要，不设置 .outputColorSpace 也不会引起色差。

```
//新版本，加载gltf，不需要执行下面代码解决颜色偏差
renderer.outputColorSpace = THREE.SRGBColorSpace;//设置为SRGB颜色空间
```

← 2. GLTF格式简介 (Web3D领域JPG)　　4. OrbitControls辅助设置相机参数→