郭隆邦 2023-03-22

12. Sprite标签(Canvas作为贴图)

上节课案例创建标签的方式，是把一张图片作为`Sprite`精灵模型的颜色贴图,本节给大家演示把Canvas画布作为`Sprite`精灵模型的颜色贴图，实现一个标签。

注意：本节课主要是技术方案讲解，默认你有Canvas基础，如果没有Canvas基础，可以学习之后再来学习本节课内容。

Canvas画布绘制一个标签

你可以使用Canvas绘制特定轮廓的标签，比如加上指引线或箭头，可以输入特定文字。

下面代码自动适配了不同长度的文字标注，文字符号越多，canvas画布越长。

```
// 生成一个canvas对象，标注文字为参数name
function createCanvas(name) {
    /**
     * 创建一个canvas对象，绘制几何图案或添加文字
     */
    const canvas = document.createElement("canvas");
    const arr = name.split(""); //分割为单独字符串
    let num = 0;
    const reg = /[\u4e00-\u9fa5]/;
    for (let i = 0; i < arr.length; i++) {
        if (reg.test(arr[i])) { //判断是不是汉字
            num += 1;
        } else {
            num += 0.5; //英文字母或数字累加0.5
        }
    }
    // 根据字符串符号类型和数量、文字font-size大小来设置canvas画布宽高度
    const h = 80; //根据渲染像素大小设置，过大性能差，过小不清晰
    const w = h + num * 32;
```

```
    canvas.width = w;
    canvas.height = h;
    const h1 = h * 0.8;
    const c = canvas.getContext('2d');
    // 定义轮廓颜色，黑色半透明
    c.fillStyle = "rgba(0,0,0,0.5)";
    // 绘制半圆+矩形轮廓
    const R = h1 / 2;
    c.arc(R, R, R, -Math.PI / 2, Math.PI / 2, true); //顺时针半圆
    c.arc(w - R, R, R, Math.PI / 2, -Math.PI / 2, true); //顺时针半圆
    c.fill();
    // 绘制箭头
    c.beginPath();
    const h2 = h - h1;
    c.moveTo(w / 2 - h2 * 0.6, h1);
    c.lineTo(w / 2 + h2 * 0.6, h1);
    c.lineTo(w / 2, h);
    c.fill();
    // 文字
    c.beginPath();
    c.translate(w / 2, h1 / 2);
    c.fillStyle = "#ffffff"; //文本填充颜色
    c.font = "normal 32px 宋体"; //字体样式设置
    c.textBaseline = "middle"; //文本与fillText定义的纵坐标
    c.textAlign = "center"; //文本居中(以fillText定义的横坐标)
    c.fillText(name, 0, 0);
    return canvas;
}
const canvas = createCanvas('设备A')
```

`CanvasTexture` 把canvas转化为纹理对象

`canvas` 画布作为 `CanvasTexture` 的参数创建一个纹理对象，本质上你可以理解为 `CanvasTexture` 把canvas画布当做图片，读取参数canvas画布上的像素值，创建纹理贴图 `Texture` 。

```
loader.load("../工厂.glb", function (gltf) {
  model.add(gltf.scene);
  const canvas = createCanvas('设备A');//创建一个canvas画布
  // canvas画布作为CanvasTexture的参数创建一个纹理对象
  // 本质上你可以理解为CanvasTexture读取参数canvas画布上的像素值
  const texture = new THREE.CanvasTexture(canvas);
  const spriteMaterial = new THREE.SpriteMaterial({
    map: texture,
```

```
  });
  const sprite = new THREE.Sprite(spriteMaterial);
})
```

精灵模型尺寸和位置设置

精灵模型尺寸和位置设置具体思路可以参考上节课讲解。

注意精灵模型**宽高比**和canvas画布宽高比保持一致即可。

```
const y = 4;//精灵y方向尺寸
// sprite宽高比和canvas画布保持一致
const x = canvas.width/canvas.height*y;//精灵x方向尺寸
sprite.scale.set(x, y, 1);// 控制精灵大小
sprite.position.y = y / 2; //标签底部箭头和空对象标注点重合
const obj = gltf.scene.getObjectByName('设备A标注'); // obj是建模软件中创建的一个空
obj.add(sprite); //tag会标注在空对象obj对应的位置
```

cavnas精灵标签封装(标注多个)

封装一个创建cavnas**精灵标签**的函数，可以根据需要调用，标注任何需要标注的地方。

```
import * as THREE from 'three';
import createCanvas from './canvas';
// 标注位置对应的模型对象obj
// name：标注文字
function createSprite(obj,name) {
    const canvas = createCanvas(name);//创建一个canvas画布
    // canvas画布作为CanvasTexture的参数创建一个纹理对象
    const texture = new THREE.CanvasTexture(canvas);
    const spriteMaterial = new THREE.SpriteMaterial({
      map: texture,
    });
    const sprite = new THREE.Sprite(spriteMaterial);
    // 控制精灵大小(sprite宽高比和canvas画布保持一致)
    const s = 0.05;//通过canvas宽高度缩放后，设置sprite.scale，避免图文宽高比变形
    const x = canvas.width*s;
    const y = canvas.height*s;
    sprite.scale.set(x, y, 1);
    sprite.position.y = y / 2; //标签底部箭头和空对象标注点重合
    obj.add(sprite); //tag会标注在空对象obj对应的位置
}
```

```
export default createSprite;
```

Canvas包含外部图片

如果Canvas包含外部图片作为背景，注意创建 `CanvasTexture` 的时候，不管你的代码结构怎么组织，主要要等图像加载完成再执行 `THREE.CanvasTexture(canvas)` ，如果还未加载完成，创建纹理时候，读取画布像素时候，会不包含图片。

```
// 生成一个canvas对象，标注文字为参数name
function createCanvas(img,name) {
    /**
     * 创建一个canvas对象，绘制几何图案或添加文字
     */
    const canvas = document.createElement("canvas");
    const w = 140; //根据渲染像素大小设置，过大性能差，过小不清晰
    const h = 80;
    canvas.width = w;
    canvas.height = h;
    const h1 = h * 0.8;
    const c = canvas.getContext('2d');
    c.fillStyle = "rgba(0,0,0,0.0)"; //背景透明
    c.fillRect(0, 0, w, h);
    c.drawImage(img, 0, 0, w, h);//图片绘制到canvas画布上
    // 文字
    c.beginPath();
    c.translate(w / 2, h1 / 2);
    c.fillStyle = "#ffffff"; //文本填充颜色
    c.font = "normal 32px 宋体"; //字体样式设置
    c.textBaseline = "middle"; //文本与fillText定义的纵坐标
    c.textAlign = "center"; //文本居中(以fillText定义的横坐标)
    c.fillText(name, 0, 0);
    return canvas;
}
```

```
  const img = new Image();
  img.src = "./标签箭头背景.png";
  img.onload = function () {
    const canvas = createCanvas(img,'设备A');//创建一个canvas画布
    // 图片加载完成后，读取canvas像素数据创建CanvasTexture
    const texture = new THREE.CanvasTexture(canvas);
    ...
    const sprite = new THREE.Sprite(spriteMaterial);
```

```
    ...
}
```

← 11. 精灵模型Sprite作为标签

1. 关键帧动画→