**纹理坐标**

纹理有自己的一套st坐标（也可以叫uv坐标），一个纹理它的宽度和高度都是1



**着色器代码**

    <!-- 顶点着色器 -->

    <script id="vertexShader" type="x-shader/x-vertex">

        // 导出属性

        attribute vec4 a\_Position;

        // 纹理坐标

        attribute vec2 a\_Pin;

        // 声明全局指针

        varying vec2 v\_Pin;

        void main(){

            gl\_Position = a\_Position;

            // 将v\_Pin指向a\_Pin

            v\_Pin = a\_Pin;

        }

    </script>

    <!-- 片元着色器 -->

    <script id="fragmentShader" type="x-shader/x-fragment">

        precision mediump float;

        // 二维取样器

        // 这里需要导出取样器

        uniform sampler2D u\_Sampler;

        varying vec2 v\_Pin;

        void main() {

            // 设置颜色从纹理中进行获取

            gl\_FragColor = texture2D(u\_Sampler, v\_Pin);

        }

    </script>

- 声明并导出一个纹理坐标

attribute vec2 a\_Pin：

- 指定gl\_FragColor颜色指针从纹理中读取颜色

gl\_FragColor = texture2D(u\_Sampler, v\_Pin);

**建立纹理数据并指定指针属性**

        // 指定绘图的顶点坐标和对应的纹理坐标

        const source = new Float32Array([

            -0.5, 0.5, 0.0, 1.0,

            -0.5, -0.5, 0.0, 0.0,

            0.5, 0.5, 1.0, 1.0,

            0.5, -0.5, 1.0, 0.0,

        ]);

        // 写入数据

        context.bufferData(context.ARRAY\_BUFFER, source, context.STATIC\_DRAW);

        // 设置a\_Position的指针属性，以便其能够从缓存中正确提前数据

        context.vertexAttribPointer(

            a\_Position,

            2,

            context.FLOAT,

            false,

            4 \* 4,

            0 \* 4

        );

        // 赋能-批处理

        context.enableVertexAttribArray(a\_Position);

        // 设置a\_Pin的指针属性，以便其能够从缓存中正确提前数据

        context.vertexAttribPointer(

            a\_Pin,

            2,

            context.FLOAT,

            false,

            4 \* 4,

            2 \* 4

        )

        //赋能-批处理

        context.enableVertexAttribArray(a\_Pin);

**装载纹理**

        // 对纹理图像垂直翻转

        context.pixelStorei(context.UNPACK\_FLIP\_Y\_WEBGL, 1);

        // 激活纹理单元

        context.activeTexture(context.TEXTURE0);

        // 创建纹理对象

        const texture = context.createTexture();

        // 把纹理对象装进纹理单元里

        context.bindTexture(context.TEXTURE\_2D, texture);

纹理单元：纹理对象需要送到纹理单元中才能被取样器读取，WebGL提供了9个纹理单元，分别是TEXTURE0|1|2|3|4|5|6|7|8，使用前需要先激活

图像垂直翻转：Image对象遵守的是栅格坐标系，其坐标y是朝下的，但我们前面说的st坐标是朝上的，所以这里需要翻转一下

**加载图片到纹理对象**

            //配置纹理图像

            context.texImage2D(

                context.TEXTURE\_2D,

                0,

                context.RGB,

                context.RGB,

                context.UNSIGNED\_BYTE,

                image

            );

            //配置纹理参数

            context.texParameteri(

                context.TEXTURE\_2D,

                context.TEXTURE\_MIN\_FILTER,

                context.LINEAR

            );

            //设置取样其为0，这样取样器就会从0号纹理单元取样

            context.uniform1i(u\_Sampler, 0);

- 配置纹理图像

texImage2D(type, level, internalformat, format, type, pixels)

type：纹理类型

level：基本图像等级

internalformat：纹理中的颜色组件

format：纹理数据格式，必须和internalformat 一样

type：纹理数据的数据类型

UNSIGNED\_BYTE 无符号字节

pixels：图像源

- 配置纹理参数

texParameteri(type, pname, param)

type：纹理类型

TEXTURE\_2D 二维纹理

pname：纹理参数的名称

TEXTURE\_MIN\_FILTER 纹理缩小滤波器

param：与pname相对应的纹理参数值

gl.LINEAR 线性

- 设置取样器取样的纹理单元

context.uniform1i(u\_Sampler, 0);

0：对应的0号纹理单元

**整体代码如下**

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <script src="http://www.yanhuangxueyuan.com/versions/threejsR92/build/three.js"></script>

</head>

<body style="margin: 0px;">

    <canvas id="canvas"></canvas>

    <!-- 顶点着色器 -->

    <script id="vertexShader" type="x-shader/x-vertex">

        // 导出属性

        attribute vec4 a\_Position;

        // 纹理坐标

        attribute vec2 a\_Pin;

        // 声明全局指针

        varying vec2 v\_Pin;

        void main(){

            gl\_Position = a\_Position;

            // 将v\_Pin指向a\_Pin

            v\_Pin = a\_Pin;

        }

    </script>

    <!-- 片元着色器 -->

    <script id="fragmentShader" type="x-shader/x-fragment">

        precision mediump float;

        // 二维取样器

        // 这里需要导出取样器

        uniform sampler2D u\_Sampler;

        varying vec2 v\_Pin;

        void main() {

            // 设置颜色从纹理中进行获取

            gl\_FragColor = texture2D(u\_Sampler, v\_Pin);

        }

    </script>

    <script>

        // 初始化

        function init(context) {

            //创建程序对象

            const program = context.createProgram();

            //创建顶点着色器

            const vertexShader = context.createShader(context.VERTEX\_SHADER);

            // 获取顶点着色器代码

            const vsSource = document.getElementById('vertexShader').innerText;

            //将着色器源文件传入着色器对象中

            context.shaderSource(vertexShader, vsSource);

            //编译着色器对象

            context.compileShader(vertexShader);

            //把顶点着色对象装进程序对象中

            context.attachShader(program, vertexShader);

            //创建片元着色器

            const fragmentShader = context.createShader(context.FRAGMENT\_SHADER);

            // 获取片元着色器代码

            const fsSource = document.getElementById('fragmentShader').innerText;

            //将着色器源文件传入着色器对象中

            context.shaderSource(fragmentShader, fsSource);

            //编译着色器对象

            context.compileShader(fragmentShader)

            //把片元着色对象装进程序对象中

            context.attachShader(program, fragmentShader);

            //连接webgl上下文对象和程序对象

            context.linkProgram(program);

            //启动程序对象

            context.useProgram(program);

            //将程序对象挂到上下文对象上

            context.program = program;

        }

        // canvas画布

        const canvas = document.getElementById('canvas');

        canvas.width = document.body.clientWidth;

        canvas.height = document.body.clientHeight;

        // webgl上下文

        const context = canvas.getContext('webgl');

        // 初始化上下文

        init(context);

        // 指定将要用来清理绘图区的颜色

        context.clearColor(0.0, 0.0, 0.0, 1.0);

        // 清理绘图区

        context.clear(context.COLOR\_BUFFER\_BIT);

        // 获取坐标指针

        const a\_Position = context.getAttribLocation(context.program, 'a\_Position');

        // 获取纹理坐标指针

        const a\_Pin = context.getAttribLocation(context.program, 'a\_Pin');

        // 获取取样器

        const u\_Sampler = context.getUniformLocation(context.program, 'u\_Sampler');

        //缓冲对象

        const sourceBuffer = context.createBuffer();

        //绑定缓冲对象

        context.bindBuffer(context.ARRAY\_BUFFER, sourceBuffer);

        // 指定绘图的顶点坐标和对应的纹理坐标

        const source = new Float32Array([

            -0.5, 0.5, 0.0, 1.0,

            -0.5, -0.5, 0.0, 0.0,

            0.5, 0.5, 1.0, 1.0,

            0.5, -0.5, 1.0, 0.0,

        ]);

        // 写入数据

        context.bufferData(context.ARRAY\_BUFFER, source, context.STATIC\_DRAW);

        // 设置a\_Position的指针属性，以便其能够从缓存中正确提前数据

        context.vertexAttribPointer(

            a\_Position,

            2,

            context.FLOAT,

            false,

            4 \* 4,

            0 \* 4

        );

        // 赋能-批处理

        context.enableVertexAttribArray(a\_Position);

        // 设置a\_Pin的指针属性，以便其能够从缓存中正确提前数据

        context.vertexAttribPointer(

            a\_Pin,

            2,

            context.FLOAT,

            false,

            4 \* 4,

            2 \* 4

        )

        //赋能-批处理

        context.enableVertexAttribArray(a\_Pin);

        // 对纹理图像垂直翻转

        context.pixelStorei(context.UNPACK\_FLIP\_Y\_WEBGL, 1);

        // 激活纹理单元

        context.activeTexture(context.TEXTURE0);

        // 创建纹理对象

        const texture = context.createTexture();

        // 把纹理对象装进纹理单元里

        context.bindTexture(context.TEXTURE\_2D, texture);

        // image 对象

        const image = new Image();

        image.src = './images/erha.jpg';

        image.onload = function () {

            //配置纹理图像

            context.texImage2D(

                context.TEXTURE\_2D,

                0,

                context.RGB,

                context.RGB,

                context.UNSIGNED\_BYTE,

                image

            );

            //配置纹理参数

            context.texParameteri(

                context.TEXTURE\_2D,

                context.TEXTURE\_MIN\_FILTER,

                context.LINEAR

            );

            //设置取样其为0，这样取样器就会从0号纹理单元取样

            context.uniform1i(u\_Sampler, 0);

            //渲染

            context.clear(context.COLOR\_BUFFER\_BIT);

            // 建立倒带上角形

            context.drawArrays(context.TRIANGLE\_STRIP, 0, 4);

        }

    </script>

</body>

</html>