如下我们将下面3张图合并为一张，第一张为主图，第二张为图案，第三张为蒙版





**导出取样器**

首先我们需要导出3个取样器

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

        precision mediump float;

        // 主取样器

        uniform sampler2D u\_Sampler;

        // 图案取样器

        uniform sampler2D u\_Pattern;

        // 蒙版取样器

        uniform sampler2D u\_Mark;

        varying vec2 v\_Pin;

        void main() {

            vec4 s=texture2D(u\_Sampler, v\_Pin);

            vec4 p=texture2D(u\_Pattern, v\_Pin);

            vec4 m=texture2D(u\_Mark, v\_Pin);

            gl\_FragColor =  m.r == 1.0 ? s\*p : s;

        }

    </script>

- 颜色取值

颜色取值的代码为gl\_FragColor =  m.r == 1.0 ? s\*p : s; ，意思是，如果蒙版颜色为白色，则取主图和图案的混合，反之蒙版颜色为黑色，则只取主图

**加载所有图片**

// image 对象

        let img1;

        let img2;

        let img3;

        new Promise((rv, rj) => {

            function loaded() {

                if (img1.complete && img2.complete && img3.complete) {

                    rv();

                }

            }

            img1 = new Image();

            img1.src = './images/dress.jpg';

            img1.onload = loaded;

            img2 = new Image();

            img2.src = './images/pattern1.jpg';

            img2.onload = loaded;

            img3 = new Image();

            img3.src = './images/mask-dress.jpg';

            img3.onload = loaded;

        })

**生成纹理并绑定取样器**

            // 使用图片生成纹理

            createTexture(img1, 0);

            createTexture(img2, 1);

            createTexture(img3, 2);

            //将0号纹理分配给着色器，0是纹理单元编号

            context.uniform1i(u\_Sampler, 0);

            context.uniform1i(u\_Pattern, 1);

            context.uniform1i(u\_Mark, 2);

**整体代码**

<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;

        // 主取样器

        uniform sampler2D u\_Sampler;

        // 图案取样器

        uniform sampler2D u\_Pattern;

        // 蒙版取样器

        uniform sampler2D u\_Mark;

        varying vec2 v\_Pin;

        void main() {

            vec4 s=texture2D(u\_Sampler, v\_Pin);

            vec4 p=texture2D(u\_Pattern, v\_Pin);

            vec4 m=texture2D(u\_Mark, v\_Pin);

            gl\_FragColor =  m.r == 1.0 ? s\*p : s;

        }

    </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;

        }

        // 生成纹理

        // img 要生成纹理的图片，index 放入哪个纹理单元

        function createTexture(img, index) {

            // 激活纹理单元

            context.activeTexture(context[`TEXTURE${index}`]);

            // 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);

            //配置纹理图像

            context.texImage2D(

                context.TEXTURE\_2D,

                0,

                context.RGB,

                context.RGB,

                context.UNSIGNED\_BYTE,

                img

            );

            //配置纹理参数

            context.texParameteri(

                context.TEXTURE\_2D,

                context.TEXTURE\_WRAP\_S,

                context.CLAMP\_TO\_EDGE

            );

            context.texParameteri(

                context.TEXTURE\_2D,

                context.TEXTURE\_WRAP\_T,

                context.CLAMP\_TO\_EDGE

            );

            context.texParameteri(

                context.TEXTURE\_2D,

                context.TEXTURE\_MIN\_FILTER,

                context.LINEAR

            );

        }

        // 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);

        context.enable(context.CULL\_FACE);

        context.enable(context.DEPTH\_TEST);

        // 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');

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

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

        //模型矩阵

        const modelMatrix = new THREE.Matrix4();

        const mx = new THREE.Matrix4().makeRotationX(0.02)

        const my = new THREE.Matrix4().makeRotationY(0.02)

        //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 points = [

            -0.4, 0.8, 0, 1,

            -0.4, -0.8, 0, 0.0,

            0.4, 0.8, 1.0, 1,

            0.4, -0.8, 1.0, 0.0,

        ];

        const source = new Float32Array(points);

        // 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);

        // image 对象

        let img1;

        let img2;

        let img3;

        new Promise((rv, rj) => {

            function loaded() {

                if (img1.complete && img2.complete && img3.complete) {

                    rv();

                }

            }

            img1 = new Image();

            img1.src = './images/dress.jpg';

            img1.onload = loaded;

            img2 = new Image();

            img2.src = './images/pattern1.jpg';

            img2.onload = loaded;

            img3 = new Image();

            img3.src = './images/mask-dress.jpg';

            img3.onload = loaded;

        }).then(() => {

            // 使用图片生成纹理

            createTexture(img1, 0);

            createTexture(img2, 1);

            createTexture(img3, 2);

            //将0号纹理分配给着色器，0是纹理单元编号

            context.uniform1i(u\_Sampler, 0);

            context.uniform1i(u\_Pattern, 1);

            context.uniform1i(u\_Mark, 2);

            //渲染

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

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

        })

    </script>

</body>

</html>