下面颜色如何给每个顶点赋予颜色

整体代码如下：

<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;

        // 导出矩阵

        uniform mat4 u\_Matrix;

        // 导出视图矩阵

        uniform mat4 u\_ViewMatrix;

        // 导出大小

        attribute float a\_PointSize;

        // 导出颜色

        attribute vec4 a\_Color;

        // 指定一个全局变量

        varying vec4 v\_Color;

        void main(){

            gl\_Position = u\_ViewMatrix\*u\_Matrix\*a\_Position;

            gl\_PointSize = a\_PointSize;

            // 将全局变量指向a\_Color

            v\_Color = a\_Color;

        }

    </script>

    <!-- 片元着色器 -->

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

        // 导出颜色属性

        precision mediump float;

        // 指定一个全局变量

        varying vec4 v\_Color;

        void main() {

            // 将gl\_FragColor指向v\_Color

            gl\_FragColor = v\_Color;

        }

    </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);

        // 获取着色器属性点

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

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

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

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

        // 视图矩阵

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

        // 设置顶点大小

        context.vertexAttrib1f(a\_PointSize, 0);

        //缓冲对象

        const sourceBuffer = context.createBuffer();

        //绑定缓冲对象

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

        // 设置顶点从缓存取数据

        const source = new Float32Array([

            0, 0.2, 0, 1, 0, 0, 1,

            -0.2, -0.1, 0, 0, 1, 0, 1,

            0.2, -0.1, 0, 0, 0, 1, 1,

        ]);

        //修改attribute 变量

        context.vertexAttribPointer(

            a\_Position,

            3,

            context.FLOAT,

            false,

            7 \* 4,

            0

        );

        context.enableVertexAttribArray(a\_Position);

        //修改attribute 变量

        context.vertexAttribPointer(

            a\_Color,

            4,

            context.FLOAT,

            false,

            7 \* 4,

            3 \* 4

        )

        //赋能-批处理

        context.enableVertexAttribArray(a\_Color);

        //写入数据

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

        // 物体运动矩阵

        let objm = new THREE.Matrix4();

        context.uniformMatrix4fv(u\_Matrix, false, objm.elements);

        // 视图矩阵

        const viewMatrix = new THREE.Matrix4();

        context.uniformMatrix4fv(u\_ViewMatrix, false, viewMatrix.elements);

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

        context.drawArrays(context.TRIANGLES, 0, 3);

    </script>

</body>

1. 首先我们导出颜色

<!-- 顶点着色器 -->

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

    // 导出颜色

    attribute vec4 a\_Color;

    // 指定一个全局变量

    varying vec4 v\_Color;

    void main(){

        // 将全局变量指向a\_Color

        v\_Color = a\_Color;

    }

</script>

<!-- 片元着色器 -->

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

    precision mediump float;

    // 指定一个全局变量

    varying vec4 v\_Color;

    void main() {

        // 将gl\_FragColor指向v\_Color

        gl\_FragColor = v\_Color;

    }

</script>

这里导出了a\_Color，然后声明了一个全局变量v\_Color，v\_Color指向a\_Color，然后在片元着色器中我们又将颜色gl\_FragColor指向v\_Color，这就等于gl\_FragColor指向了a\_Color，我们只需要给a\_Color赋值就可以控制颜色了

1. 在js中获取a\_Position和a\_Color

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

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

1. 构建数据类型

        // 设置顶点从缓存取数据

        const source = new Float32Array([

            0, 0.2, 0, 1, 0, 0, 1,

            -0.2, -0.1, 0, 0, 1, 0, 1,

            0.2, -0.1, 0, 0, 0, 1, 1,

        ]);

其中每一行代表一个顶点，每一行前面3个是顶点的位置，后面4个是顶点的颜色

1. 设置a\_Position和a\_Color的指针属性，以便他们能够正确的从缓存中取得数据

        //修改attribute 变量

        context.vertexAttribPointer(

            // 要设置的指针

            a\_Position,

            // 每次取数据的个数

            3,

            // 数据的类型

            context.FLOAT,

            // 是否归一化

            false,

            // 一行数据的字节数，我们一行又7个数据，Float32是4个字节，所以一行数据的字节数为 7\*4

            7 \* 4,

            // 每行数据跳过的自己数

            // a\_Position是从第0个数据开始取的，所以这里填0\*4

            0 \* 4

        );

        //赋能-批处理

        context.enableVertexAttribArray(a\_Position);

        //修改attribute 变量

        context.vertexAttribPointer(

            a\_Color,

            4,

            context.FLOAT,

            false,

            7 \* 4,

            3 \* 4

        )

        //赋能-批处理

        context.enableVertexAttribArray(a\_Color);

1. 绘图

context.drawArrays(context.TRIANGLES, 0, 3);

取出3行数据进行绘图