**绘制步骤**

- 首先我们要准备好三角形点的数据

- 将这些数据传给WebGL

- 告诉WebGL根据这些点画出三角形

代码如下

<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;

    attribute float a\_PointSize;

    void main(){

        gl\_Position = a\_Position;

        gl\_PointSize = a\_PointSize;

    }

</script>

<!-- 片元着色器 -->

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

    // 导出颜色属性

    precision mediump float;

    uniform vec4 u\_FragColor;

    void main() {

        gl\_FragColor = u\_FragColor;

    }

</script>

<script>

    // 设置从缓存取数据

    function setVertexDatas(context, position) {

        // 生成数据缓存

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

        // 之前我们是手动给 position 赋值，示例：context.vertexAttrib2f(position, x, y);

        // 现在我们调用该方法，相对于告诉WebGL，画图时你从缓存中取2个值，调用 context.vertexAttrib2f(position, x, y) 取绘图

        // 2表示2个参数，context.FLOAT为参数类型，false  是否将顶点数据归一，0 相邻两个顶点间的字节数，0 从缓冲区的什么位置开始存储变量

        context.vertexAttribPointer(position, 2, context.FLOAT, false, 0, 0);

        // 开启顶点数据的批处理功能

        context.enableVertexAttribArray(position);

    }

...

    // 获取着色器属性点

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

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

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

    // 设置顶点大小

    context.vertexAttrib1f(a\_PointSize, 0);

    // 设置绘图颜色

context.uniform4f(u\_FragColor, 1, 1, 1, 1.0);

    // 设置顶点从缓存取数据

    setVertexDatas(context, a\_Position);

    // 顶点数据

    let datas = [

        // x, y

        0.0, 0.1,

        -0.1, -0.1,

        0.1, -0.1

    ];

    // 把数据放到缓存中

    context.bufferData(context.ARRAY\_BUFFER, new Float32Array(datas), context.STATIC\_DRAW);

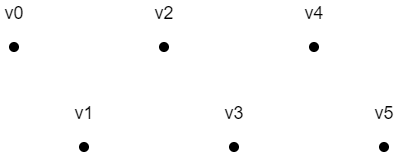
    // 绘制（绘制时顶点会从缓存中取出数据进行绘制）

    context.drawArrays(context.LINE\_LOOP, 0, 3);

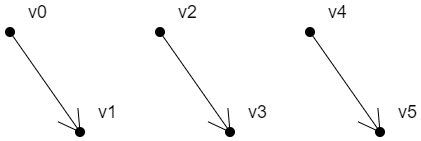
</script>

**可以绘制的图形**

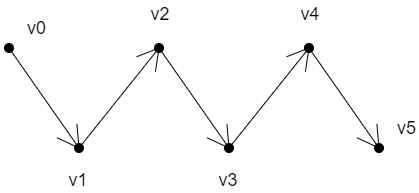
- POINTS 可视的点



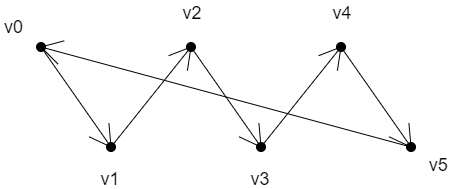
- LINES 单独线段



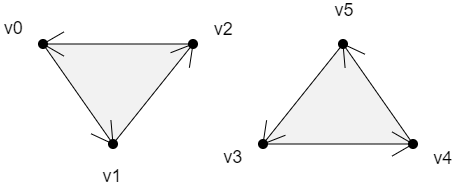
- LINE\_STRIP 线条



- LINE\_LOOP 闭合线条

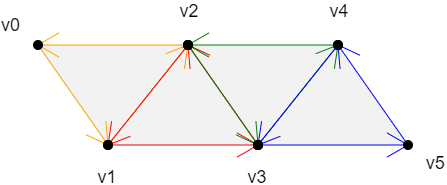


- TRIANGLES 单独三角形



- TRIANGLE\_STRIP 三角带

其绘图时绘图步骤为v0\v1\v2，v1\v2\v3，v2\v3\v4，以此类推



- TRIANGLE\_FAN 三角扇

