**概念**

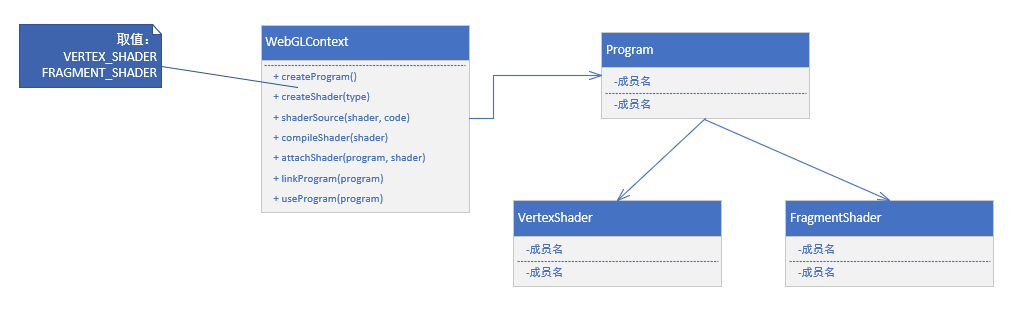
WebGL上线文：WebGL有一套独立的语言，JS无法直接调用，所有调用都需要通过WebGL上下文进行

Program（程序）：用于绘图的程序

VertexShader（顶点着色器）:用于指定绘图的点

FragmentShader（片段着色器）：在指定绘图的点后WebGL会根据顶点生成片元，然后调用片元着色器

他们的关系如下



**初始化操作**

如下是初始化的流程

- 创建Context

- 创建Program、创建VertexShader、创建FragmentShader

- 为Program指定VertexShader和FragmentShader

- 链接Program到上下文，并启动Program

- 使用Context开始绘图

上代码：

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

    <canvas id="canvas"></canvas>

    <!-- 顶点着色器 -->

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

        void main() {

            gl\_Position = vec4(0.0, 0.0, 0.0, 1.0);

            gl\_PointSize = 100.0;

        }

    </script>

    <!-- 片元着色器 -->

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

        void main() {

            gl\_FragColor = vec4(1.0, 1.0, 0.0, 1.0);

        }

    </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, 1.0);

        // 清理绘图区

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

        // 绘制顶点

        context.drawArrays(context.POINTS, 0, 1);

    </script>

</body>

**顶点着色程序**

我们看一下顶点着色程序的代码

    <!-- 顶点着色器 -->

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

        void main() {

            gl\_Position = vec4(0.0, 0.0, 0.0, 1.0);

            gl\_PointSize = 100.0;

        }

    </script>

指定type="x-shader/x-vertex"后，浏览器不认识该类型，所有不会执行这个script

gl\_Position：是一个全局指针变量，WebGL通过访问gl\_Position获取顶点位置

gl\_PointSize：是一个全局指针变量，WebGL通过访问gl\_PointSize获取顶点大小

void main() {…… } 是入口函数

**片元着色程序**

看一下片元着色程序的代码

    <!-- 片元着色器 -->

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

        void main() {

            gl\_FragColor = vec4(1.0, 1.0, 0.0, 1.0);

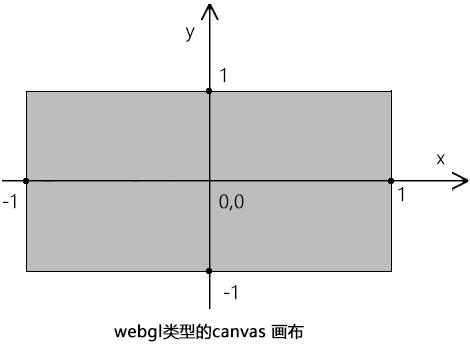
        }

    </script>

gl\_FragColor是一个颜色指针变量，WebGL通过访问gl\_FragColor获取颜色

**WebGL坐标**

如下是WebGL的坐标，坐标由xyz组成，z轴的正方向指向屏幕外，这里说明一下xy，(0,0)表示中心点，(1,1)表示右上角，(-1,-1)表示左下角



**Html位置转WebGL坐标**

如下函数

        // 转GL坐标

        function toGLPosition(clientX, clientY){

            // 获取canvas的位置和宽高

            const { left, top, width, height } = document.getElementById('canvas').getBoundingClientRect();

            // 计算canvas x y 的长

            const xLenght = width / 2;

            const yLenght = height / 2;

            // 计算canvas中心点坐标

            const xCenter = left + xLenght;

            const yCenter = top + yLenght;

            // 计算坐标

            const x = (clientX - xCenter) / xLenght;

            const y = (clientY - yCenter) / yLenght;

            return [x, -y];

        }