WebGL通过着色器进行绘图，来看一下如何通过JS改变着色器的属性

**WebGL数据类型**

如下列出了WebGL的数据类型

1. 基本数据类型

bool，int，float

1. 数据类型-向量

| **关键字** | **数据类型** |
| --- | --- |
| vec2 | 二维向量，具有xy两个分量，分量是浮点数 |
| vec3 | 三维向量 ，具有xyz三个分量，分量是浮点数 |
| vec4 | 四维向量 ，具有xyzw四个分量，分量是浮点数 |

- 变量声明

// 变量声明

vec3 direction;

// 赋值，这里 vec3 需要传如3个float值，vect4为4个

direction = vec3(1.0,0.0,0.0);

1. 数据类型 - 矩阵

矩阵的类型

| **关键字** | **数据类型** |
| --- | --- |
| mat2 | 2x2矩阵，4个元素 |
| mat3 | 3x3矩阵，9个元素 |
| mat4 | 4x4矩阵，16个元素 |

- 矩阵声明和赋值

// 需要表示的矩阵

// 1.1 1.2 1.3 1.4

// 2.1 2.2 2.3 2.4

// 3.1 3.2 3.3 3.4

// 4.1 4.2 4.3 4.4

// 注：webgl的矩阵是以列为主的方向

mat4 matrix4 = mat4(

    1.1,2.1,3.1,4.1,

    1.2,2.2,3.2,4.2,

    1.3,2.3,3.3,4.3,

    1.4,2.4,3.4,4.4

);

1. 数据类型 - 数组

WebGL仅支持一维数组

// 声明一个数组变量fArr，数组变量fArr有100个元素，元素的数据类型是浮点数

float arr[100];

// 声明一个长度20的三维向量数组变量v3Arr

vec3 v3Arr[20];

1. float精度设置

通过precision关键字可以设置float的精度，如

precision mediump float;

精度的类型

| **关键子** | **精度** |
| --- | --- |
| lowp | 低精度 |
| mediump | 中精度 |
| highp | 高精度 |

片元着色器的float没有设置默认精度，在使用片元着色器应该先设置默认精度

**导出着色器属性**

如下看代码

<!-- 顶点着色器 -->

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

    // 导出属性

    attribute vec4 a\_Position;

    void main(){

        gl\_Position = a\_Position;

        gl\_PointSize = 50.0;

    }

</script>

...

<script>

    // 获取着色器属性点

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

    // 修改属性点

    context.vertexAttrib3f(a\_Position, 0.0, 0.0, 0.0);

    // 绘制顶点

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

</script>

- 首先声明一个变量，该变量的类型是vec4，attribute为表示导出该变量（attribute只能用于顶点着色器，通常用于导出顶点相关的数据，该关键字导出的变量可以进行批处理）

attribute vec4 a\_Position;

- 通过上下文获取属性点，通过getAttribLocation方法可获取attribute导出的变量

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

- 通过上下文修改属性点，通过vertexAttrib3f方法可设置vertexAttrib3f导出的变量

context.vertexAttrib3f(a\_Position, 0.0, 0.0, 0.0);

扩展知识点：

vertexAttrib方法用于修改vertex着色器的属性，其方法有如下

vertexAttrib1f(location,v0)

vertexAttrib2f(location,v0,v1)

vertexAttrib3f(location,v0,v1,v2)

vertexAttrib4f(location,v0,v1,v2,v3)

其命名规则为vertexAttrib+参数个数+参数类型(f代表float除此之外还有i代表整型，v代表数字)，以后会遇到很多这种类型的方法

**导出颜色属性**

如下代码

<canvas id="canvas"></canvas>

...

<!-- 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>

    ...

    // 获取属性元素

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

    // 修改颜色

    context.uniform4f(u\_FragColor, Math.random(), Math.random(), Math.random(), 1.0);

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

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

</script>

- 导出颜色属性代码如下，uniform是导出变量（uniform可用于顶点着色器和片元着色器，通常导出非顶点相关的数据，该关键字导出的数据不能进行批处理）

uniform vec4 u\_FragColor;

- 第一行precision mediump float;是必须的，其定义了float的精度

- 通过getUniformLocation方法获取uniform导出的变量

- 通过uniform4f方法设置uniform导出的变量