这3个关键字用于数据传输

| **关键字(变量类型)** | **数据传递** | **声明变量** |
| --- | --- | --- |
| attribute | javascript——>顶点着色器 | 声明顶点数据变量 |
| uniform | javascript——>顶点、片元着色器 | 声明非顶点数据变量 |
| varying | 顶点着色器——>片元着色器 | 声明需要插值计算的顶点变量 |

**attribute类型变量**

attribute可以导出的变量如下

// attribute声明顶点位置变量

attribute vec4 position;

// 与顶点相关的浮点数

attribute float scale;

在js中获取导出的变量指针

const position = context.getAttribLocation(context.program, 'position');

const scale = context.getAttribLocation(context.program, 'scale');

在js中设置值

context.vertexAttrib4f(position, 0.0, 0.0, 0.0, 0.0);

context.vertexAttrib1f(scale, 0.0);

在js通过缓存设置值

// 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');

//缓冲对象

const sourceBuffer = context.createBuffer();

//绑定缓冲对象

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

//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);

// 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,

]);

//写入数据

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

**uniform类型变量**

可以导出的变量有

uniform vec4 u\_lightPosition;

uniform mat4 u\_matrix;

uniform sampler2D u\_Sampler;

uniform float arr[12];

在js中获取导出的变量指针

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

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

在js中设置值

context.uniform4f(u\_lightPosition, 0.0, 0.0, 0.0, 1.0);

const modelMatrix = new THREE.Matrix4();

context.uniformMatrix4fv(u\_matrix, false, mat.elements);

- 传递数组

声明

uniform float arr[12];

在js中设置

// 批量传递数组元素值

var arr =gl.getUniformLocation(program, "arr")

var typeArr = new Float32Array([

  0.6,-0.3,0.6,0.4,

  -0.8,-0.3,0.6,0.4,

  0.7,0.7,0.6,0.99,

])

gl.uniform1fv(arr, typeArr);

**varying类型变量**

varying用于顶点着色器向片元着色器传递变量

<!-- 顶点着色器源码 -->

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

  // attribute声明顶点颜色变量

  attribute vec4 a\_color;

  //varying声明顶点颜色插值后变量

  varying vec4 v\_color;

  void main() {

    //顶点颜色插值计算

    v\_color = a\_color;

  }

</script>

<!-- 片元着色器源码 -->

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

  // 所有float类型数据的精度是lowp

  precision lowp float;

  // 接收顶点着色器中v\_color数据

  varying vec4 v\_color;

  void main() {

    // 插值后颜色数据赋值给对应的片元

    gl\_FragColor = v\_color;

  }

</script>