**基本数据类型**

bool，int，float

数据类型转换函数

| **转化** | **函数** | **规则** |
| --- | --- | --- |
| 浮点数转整型数 | int(10.1) | 直接把浮点数部分去掉，10.1转化为10 |
| 整型数转浮点数 | float(10) | 整型数直接转化为浮点数10.0 |
| 数子转布尔值 | bool(10) | 0或0.0转化为false，非0或非0.0转化为true |
| 布尔值转整型数 | int(true) | false转化为0，true转化为1 |
| 布尔值转浮点数 | float(true) | false转化为0.0，true转化为1.0 |

**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);

- 向量分量

| **向量vector** | **访问** |
| --- | --- |
| 第1个分量 | vector.x |
| 第2个分量 | vector.y |
| 第3个分量 | vector.z |
| 第4个分量 | vector.w |

示例

vec3 pos = vec3(10.2,9.6,3.9);

pos.x = pos.x+2.0;

pos.x = 20.0;

- 其他用法

// 声明一个四维向量变量，并赋值

vec4 v4 = vec4(10.0,9.0,3.0,1.0);

// 提取v4前三个分量，并赋值给一个三维向量变量

vec3 v3 = v4.xyz;

- 数学运算

如下演示向量的数学运算

// 加法

vec3 pos = vec3(1.0,2.0,3.0);

pos = pos+1.0;    //pos新的值为vec3(2.0,3.0,4.0)

// 减法

vec3 pos = vec3(1.0,2.0,3.0);

pos = pos-1.0;      //pos新的值为vec3(0.0,1.0,2.0)

// 乘法

vec3 pos = vec3(1.0,2.0,3.0);

pos = pos\*2.0;      //pos新的值为vec3(2.0,4.0,6.0)

// 除法

vec3 pos = vec3(1.0,2.0,3.0);

pos = pos/2.0;      //pos新的值为vec3(0.5,1.0,1.5)

// 向量间运算

vec3 v1 = vec3(1.0,2.0,3.0);

vec3 v2 = vec3(-1.0,-2.0,-3.0);

vec3 v3 = v1+v2;    //结果为vec3(0.0,0.0,0.0)

**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

);

- 对角矩阵

// 2.0 0.0 0.0 0.0

// 0.0 2.0 0.0 0.0

// 0.0 0.0 2.0 0.0

// 0.0 0.0 0.0 2.0

mat4 matrix = mat4(2.0);

- 访问矩阵元素

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

);

// 访问矩阵matrix4的第二列

vec4 v4 = matrix4[1];       //返回值vec4(1.2,2.2,3.2,4.2)

// 访问矩阵matrix4的第三列第四行对应的元素

float f = matrix4[2][3];    //返回4.3

- 矩阵的平移

// 声明一个四维向量pos，xyz的坐标是 0.0, 3.2, 6.8

vec4 pos = vec4(0.0, 3.2, 6.8, 1.0);

//创建平移矩阵(表示沿x轴平移-0.4)

//1   0   0  -0.4

//0   1   0    0

//0   0   1    0

//0   0   0    1

mat4 m4 = mat4(

    1,0,0,0,

    0,1,0,0,

    0,0,1,0,

    -0.4,0,0,1

);

// 算出平移后的向量newpos

vec4 newPos=m4\*pos; //平移后结果：newPos = vec4(-0.4,3.2,6.8,1.0)

- 矩阵的旋转

// 顶点齐次坐标pos

vec4 pos = vec4(0.0, 3.2, 6.8, 1.0);

float radian = radians(30.0);   //角度转弧度

float cos = cos(radian);        //求解旋转角度余弦值

float sin = sin(radian);        //求解旋转角度正弦值

//旋转矩阵mx，创建绕x轴旋转矩阵

mat4 mx = mat4(

    1,0,0,0,

    0,cos,-sin,0,

    0,sin,cos,0,

    0,0,0,1

);

//旋转矩阵my，创建绕y轴旋转矩阵

mat4 my = mat4(

    cos,0,-sin,0,

    0,1,0,0,

    sin,0,cos,0,

    0,0,0,1

);

//两个旋转矩阵、顶点齐次坐标连乘

vec4 newPos = mx\*my\*pos;

**数据类型 - 取样器**

取样器用于采集纹理

<!-- 顶点着色器源码 -->

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

    attribute vec2 a\_TexCoord;  // 纹理坐标

    varying vec2 v\_TexCoord;    // 插值后纹理坐标

    void main() {

      // 纹理坐标插值计算

      v\_TexCoord = a\_TexCoord;

    }

  </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highp

    precision highp float;

    // 接收插值后的纹理坐标

    varying vec2 v\_TexCoord;

    // 纹理图片像素数据

    uniform sampler2D u\_Sampler;

    void main() {

      // 采集纹素，逐片元赋值像素值

      gl\_FragColor = texture2D(u\_Sampler,v\_TexCoord);

    }

  </script>

**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];

**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