# webgl入门

## hello webgl

### 绘图流程

1. 获取canvas元素
2. 获取webgl绘图上下文
3. 设置背景色 gl.clearcolor
4. 清空canvas gl.clear clear函数的参数为bugger，

Buffer的类型包括颜色缓冲区，深度缓冲区，模板缓冲区，可以清空多个buffer

### 绘制第一个点—直接绘制

在清空canvas之后，可以开始绘制点；

绘制点之前要指定顶点着色器和片元着色器，那么必须先初始化shader

Webgl坐标系为右手，其坐标原点对应canvas区域的中心点

### 绘制第一个点—使用变量传递

在webgl系统外部，通过Gl.getAttribLocation函数获取相应attribute的地址，并调用对应的函数该地址赋值，webgl系统内部即可接收到该值。

getAttribLocation获取的是顶点着色器程序中main函数中的变量地址。

同理可以修改绘制点的大小

### 点击绘制点---自己动手实现

绘制函数执行后，颜色缓冲区会被重置

### 改变点的颜色

## 绘制三角形

### 使用缓冲区对象向顶点着色器传入多个顶点着色器的过程：

1. 创建缓冲区对象 gl.createBuffer
2. 绑定缓冲区对象 gl.bindBuffer
3. 将数据写入缓冲区对象 gl.bufferData
4. 将缓冲区对象分配给一个attribute变量 gl.vertexAttribPointer
5. 开启attribute变量 gl.enableVertexAttribArray

## 绘制矩形

如果是三角带的方式，第一个点在右下角

如果是三角扇的方式，第一个点在左下角

## 平移旋转缩放---暂时跳过

## 颜色与纹理

### 非顶点数据传入顶点着色器—varying

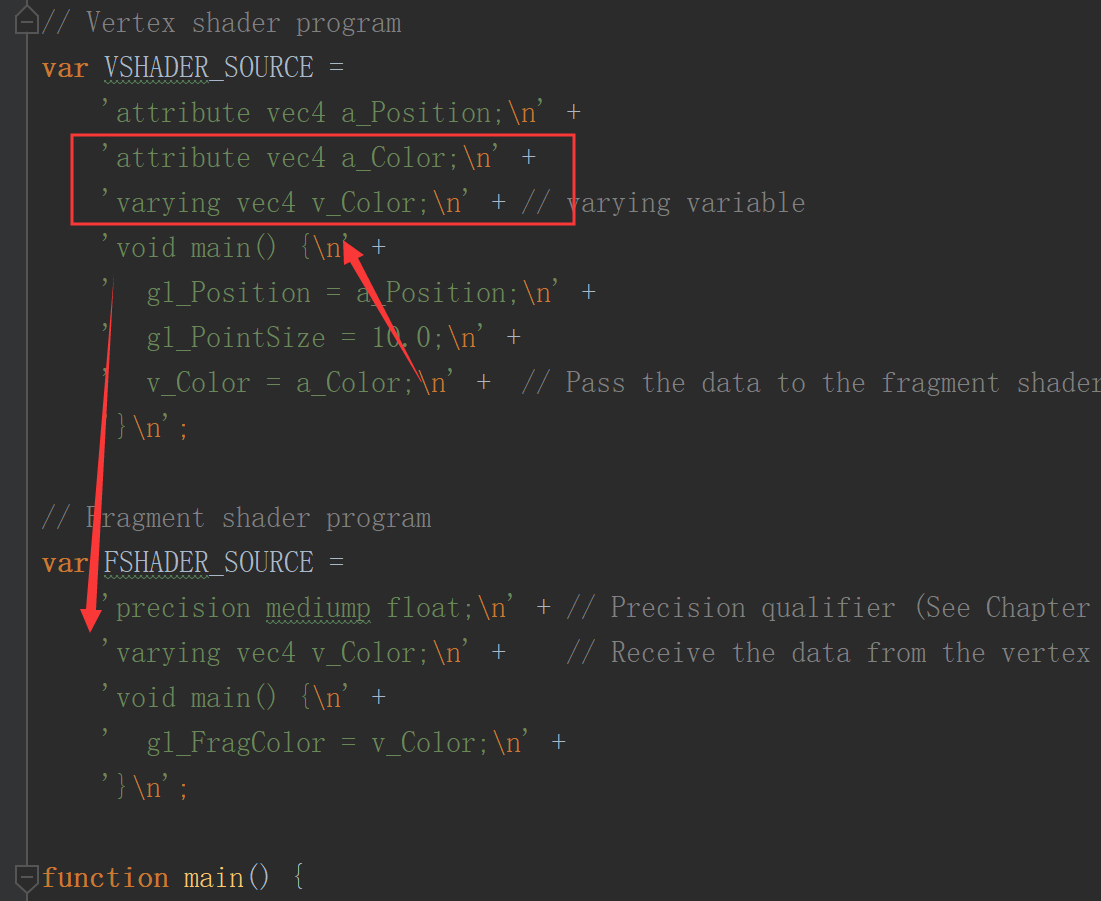
，即顶点的尺寸，颜色等

顶点的尺寸属性可以通过对缓冲区对象数据的切分来实现。详见gl.vertexAttribPointer

颜色：

### 修改颜色

在顶点着色器中写一个varing变量来接收，片元着色器中写一个相同得变量，webgl会自动赋值



### 彩色三角形

### 紋理贴图—跳过

1准备好纹理图片

2为图形配置纹理映射方式 纹理坐标—图元坐标 如何映射的逻辑，纹理坐标在左下角

3加载纹理图像，对其进行一些配置

4在片元着色器中将相应的文素抽取出来，并赋值给片元

## 着色器

### 数据类型：

Int float bool

Vec矢量

Ivec

Bvec

Mat2,mat3,mat4 矩阵

构造矩阵的方式：

依次指定

通过向量构造

通过指定+向量

### 矩阵

[x,y,z,w

R,g,b,a

S,t,p,q]

顶点坐标分量-颜色分量，纹理映射分量

## 进入三维世界

### 视图矩阵

EyeX,eyeY,eyeZ 指定视点

atX,atY,atZ, 指定观察点

upX,upY,upZ 上方向

webgl中：观察者默认状态，

视点位于坐标原点 0,0,0

视线为z轴负方向，观察点位于0 0 -1

上方向为Y 轴负方向 即0 1 0 相对于canvas坐标系而言容易理解

实际上，改变视图矩阵和改变图元的顶点坐标位置是等价的过程

因此在改变视图矩阵时，可以理解为对顶点坐标进行仿射变换

### 模型矩阵

由于对每一个顶点坐标而言，视图矩阵与仿射变换矩阵都是相同的，因此不需要每一次都进行计算

视图矩阵与放射变换矩阵的乘积即为模型矩阵

### 可视范围（正射类型）

水平视角为200°

### 可视空间

长方体可视空间—盒状空间-------正射投影产生

四棱锥/金字塔可视空间-----------透视投影 ---------通常所用-真实的状态

### 定义盒状可视空间

Matrix.setOrtho(left,right,bottom,tp,near,far)设置投影矩阵

正射投影矩阵

### 可视空间—透视投影

Matrix.setperspective(fov,aspect,near,far)

第一个参数为垂直视角，即可视空间顶面与底面间的夹角，必须大于0 ，人有仰视角和俯视角度，此角度单指一个角度

第二个参数，指定近裁剪面的宽高比例

Near，far指定远近裁剪面的值

### 隐藏面消除

Webgl绘制按传入坐标的先后顺序而来，可能导致被遮挡的物体却显示在最前面

Gl.enable(gl.DEPTH\_TEST)

在绘制前清除深度缓冲区

Gl.clear(gl.DEPTH\_BUFFER\_BIT)

### 深度冲突

当两个物体非常接近时，深度缓冲区的精度无法检测，导致无法区分哪个在前/后

// 开启多边形偏移，开启代码在绘制之前

gl.enable(gl.POLYGON\_OFFSET\_FILL);

// 设置偏移的参数，偏移代码在绘制之后

gl.polygonOffset(1.0, 1.0); // Set the polygon offset

二次绘制时就不会产生深度冲突问题了

### 立方体

## 光照

### 光源类型

点光源，平行光源，环境光（二次反射光）

环境光不需要指定光源位置和方向，仅需要指定光源的颜色

反射类型：

漫反射，镜面反射（环境反射）

### 漫反射----点光源/平行光源

漫反射的颜色=入射光颜色\*表面基底颜色\*cos(a)

由于cos a=光线方向\*法线方向 光线方向是指入射方向的反方向，实际上，光线方向无论指入射方向还是反射方向，由于光线在于镜面平行时，反射光的强度为0，因此仅需要计算在垂直于镜面方向的光线强度即可。

由向量的点积的几何含义：即a向量在向量b方向的值

a\*b=|a||b|\*cos(a) 当a与b均为单位向量时，即可得到cos a的值

将ab矢量的长度调整为一同时保持方向不变的过程称为归一化，GLSL ES提供了内置函数，可直接使用

法线：垂直于表面的方向

### 环境反射

环境反射=入射光颜色\*表面基底颜色

相比漫反射而言，不需要再乘角度了

当两种反射同时存在时，表面反射光的颜色=漫反射颜色+环境反射颜色

### 逆转置矩阵—逆矩阵转置

对顶点进行变换的---模型矩阵

将变换之前的法向量乘模型矩阵-----------逆转置矩阵

将变换矩阵求逆再转置，乘原来的法向量再归一化即可

### 点光源

动态计算光源方向

### 逐片元光照-----顶点逻辑，片元实现

将顶点着色器中的计算反射光颜色移动到片元着色器中

## 层次模型

### 着色器和着色器程序对象

初始化着色器执行顺序

创建着色器对象

填充着色器代码

编译着色器

创建程序对象

为程序分配着色器

链接程序对象

使用程序对象

着色器对象：管理一个顶点/片元着色器

程序对象：管理着色器对象的容器

## 高级技术

### 鼠标旋转

### 选中物体

### 选中一个表面

### HUD平视显示器—添加标注等

Canvas重叠

### 雾化效果

### 使用W分量

### 绘制圆形的点

### α混合