opengl es 笔记

创建流程如下：

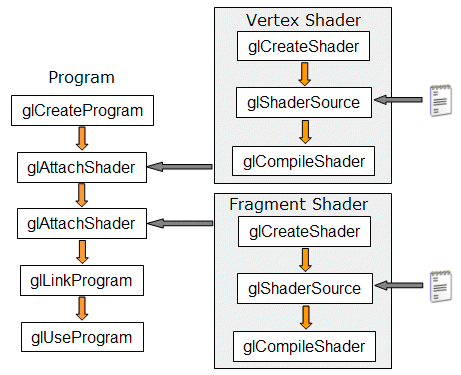


图1

loadvertex();//创建顶点着色器及三角形 即：骨骼

initShader(); //初始化着色器，即图1流程

loadtexture(); //加载纹理

之后就可以进行将画面绘制到界面上显示了。

**加载纹理的过程**：

生成纹理（GLE20.glGenTextures）->绑定纹理（GLE20.glBindTexture）->设置纹理参数（GLE20.glTextParameteri）->加载图像到纹理（texImage2D）

注：加载图像到纹理的过程，是将内存复制到显存的过程，所以加载完成后就可以将图像的内存释放掉了。

**GEL20.glGetAttribLocation()**

调用glGetAttribLocation来获取顶点着色器中，指定attribute名的index.以后就可以通过这个index向顶点着色器中传递数据。获取失败则返回-1.

**向顶点着色器传入数据**

在传递数据前先要调用GLE20.glUsePropram()方法。设置顶点属性数组启动。

**要启用或者禁用顶点属性数组**，调用glEnableVertexAttribArray和glDisableVertexAttribArray传入参数index

GLE20.glVertexAttribPointer()定义顶点属性数据

void glVertexAttribPointer (int indx,

int size,

int type,

boolean normalized,

int stride,

Buffer ptr )

参数含义：

index 指定要修改的顶点着色器中顶点属性的索引值

size 指定每个顶点属性的组件数量。必须为1、2、3或者4。如position是由3个（x,y,z）组成，而颜色是4个（r,g,b,a））

type 指定数组中每个组件的数据类型。可用的符号常量有GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT,GL\_UNSIGNED\_SHORT, GL\_FIXED, 和 GL\_FLOAT，初始值为GL\_FLOAT。

normalized指定当被访问时，固定点数据值是否应该被归一化（GL\_TRUE）或者直接转换为固定点值（GL\_FALSE）。

stride 指定连续顶点属性之间的偏移量。如果为0，那么顶点属性会被理解为：它们是紧密排列在一起的。初始值为0。如果normalized被设置为GL\_TRUE，意味着整数型的值会被映射至区间[-1,1](有符号整数)，或者区间[0,1]（无符号整数），反之，这些值会被直接转换为浮点值而不进行归一化处理。

Ptr顶点的缓冲数据

例：

mTriangleVerticesData数组为{

　　-1.0f, -0.5f, 0, -0.5f, 0.0f,

　　1.0f, -0.5f, 0, 1.5f, -0.0f,

　　0.0f, 1.11803399f, 0, 0.5f, 1.61803399f

} //x,y,z,u,v

GLES20.glVertexAttribPointer(maPositionHandle, 3, GLES20.GL\_FLOAT, false,//取3个数据x,y,z

20, mTriangleVertices);//跳转20个字节位(5个数据)再取另外3个数据，这是实现块状数据存储的关键，很多函数里都有这个参数，通常写作int stride

mTriangleVertices.position(3);//从索引3开始取数据

GLES20.glVertexAttribPointer(maTextureHandle, 2, GLES20.GL\_FLOAT, false,//取两个数据U,V

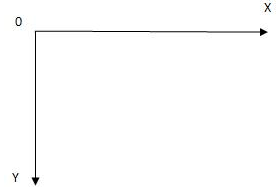
20, mTriangleVertices);//到此基本明晰了如何块状存储数据

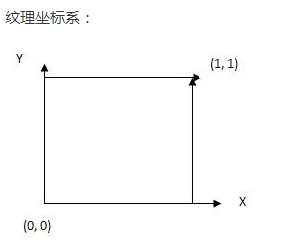
**设置活动纹理**

GEL20.glActiveTexture(),指定哪一个纹理单元被置为活动状态。

**坐标系**

屏幕坐标系





左下：（0，0）

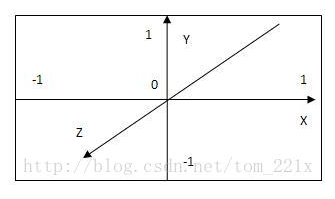
右下：（1，0）

左上：（0，1）

右上：（1，1）

1为全屏显示，半屏显示则是：0.5f

顶点坐标系：



左下角：（-1，-1）

右下角：（1，-1）

左上角：（-1，1）

右上角：（1，1）

顶点坐标以中心点（0，0）

1为全屏显示，半屏显示则是：0.5

**坐标的转换**

为了显示方便，要进行归一化处理，将顶点坐标转换成屏幕坐标，或屏幕坐标转换为顶点坐标显示。

例如将屏幕坐标转换成顶点坐标(全屏显示)： dX= 2.0\*（ screenX/screenWidth）

dY=0- 2.0\*(screenY/screenHeight)

其它情况类似

或者使用正交映射的方式进行归一化处理。

正交映射： Matrix.orthoM()，顶点着色器的范围使用第四象限进行处理。

如：void onsurfacechange(int w,int h){

Matrix.orthoM(0,width,0,height,-1,1);

Matrix.setLookAtM(0,0,1,0,0,0,0,1,0); //摄像机 眼的位置，观察者位置，摄像机方向360度

if (第一次加载）{

varyTools.translate(0, height, 0);

setMatrix(varyTools.getFinalMatrix());

GLES20.glUniformMatrix4fv(hMatrix, 1, false, matrix, 0);

进行平移纹理，且应用

}

}

顶点坐标与纹理坐标：

*private float*[] quadVertex = *new float*[]{  
 0f, 0f, 0.0f, *// Position 0* 0, 0f, *// TexCoord 0 0，1* 0f, (-1.0f), 0.0f, *// Position 1* 0f, 1f, *// TexCoord 1 0，0* 1.0f, (-1.0f), 0.0f, *// Position 2* 1.0f, 1f, *// TexCoord 2 1 0* 1.0f,0f, 0.0f, *// Position 3* 1.0f, 0f, *// TexCoord 3 1，1*};

以上有个问题，就是Y方向是反的，因为屏幕坐标与顶点坐标刚好Y方向是反的。

**常见问题**

* png图片显示的透明部分，没有透明显示

//打开混合功能，进行混合处理，透明

glEnable (GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA,GL\_ONE\_MINUS\_SRC\_ALPHA);