

# iOS开发-OpenGL ES入门教程2



([u/815d10a4bdce](#))([/u/815d10a4bdce](#)) [+ 关注](#)

2016.03.28 16:56\* 字数 1370 阅读 6582 评论 61 喜欢 24 赞赏 1 阅读 6582 评论 61 喜欢 24 赞赏 1

## 教程

OpenGL ES入门教程1-Tutorial01-GLKit (<http://www.jianshu.com/p/750fde1d8b6a>)  
这次的是**shader**编译链接、**glsl**入门和简单图形变换。

OpenGL ES系列教程在这里 (<http://www.jianshu.com/notebooks/2135411/latest>)。  
OpenGL ES系列教程的代码地址 (<https://github.com/loyinglin/LearnOpenGLES>) - 你的star和fork是我的源动力，你的意见能让我走得更远。

## 效果展示



## 核心思路

不采用GLKBaseEffect，编译链接自定义的着色器（shader），用简单的glsl语言来实现顶点和片元着色器，并对图片用简单的图形变换。

## 具体细节

### 1、shader编译

- c语言编译流程：预编译、编译、汇编、链接
- glsl的编译过程类似c语言，主要有glCompileShader、glAttachShader、glLinkProgram三步；

```

- (GLuint)loadShaders:(NSString *)vert frag:(NSString *)frag {
    GLuint verShader, fragShader;
    GLint program = glCreateProgram();

    //编译
    [self compileShader:&verShader type:GL_VERTEX_SHADER file:vert];
    [self compileShader:&fragShader type:GL_FRAGMENT_SHADER file:frag];

    glAttachShader(program, verShader);
    glAttachShader(program, fragShader);

    //释放不需要的shader
    glDeleteShader(verShader);
    glDeleteShader(fragShader);

    return program;
}

- (void)compileShader:(GLuint *)shader type:(GLenum)type file:(NSString *)file {
    //读取字符串
    NSString* content = [NSString stringWithContentsOfFile:file encoding:NSUTF8StringEncoding error:nil];
    const GLchar* source = (GLchar *)[content UTF8String];

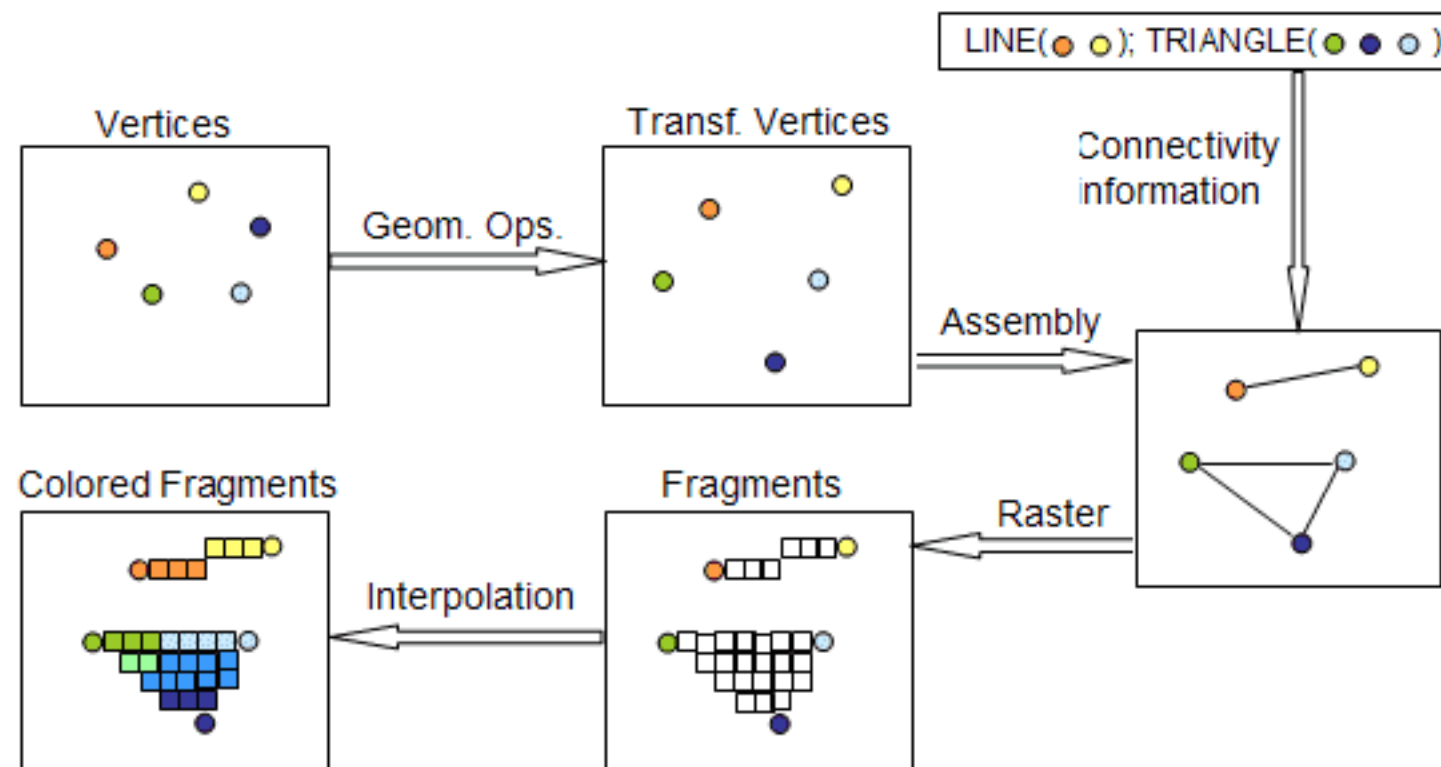
    *shader = glCreateShader(type);
    glShaderSource(*shader, 1, &source, NULL);
    glCompileShader(*shader);
}

```

## 2、glsl入门

glsl是OpenGL的着色器语言，有c基础可以很快上手，注意以下几点：

- 着色器有顶点着色器和片元着色器两种；参考下图，顶点着色器在第一个，片元着色器在最后一个；**注意**，在顶点着色器中处理顶点，片元着色器处理像素点颜色，那么对于一条线段，顶点着色器只会处理俩个顶点的坐标、颜色等信息，线段上的点会由插值生成。



- 如下，是一个顶点着色器。出现了attribute、uniform、varying这类修饰符，遇到这些可以看 [这里](http://www.tuicool.com/articles/yEBFvmA) (<http://www.tuicool.com/articles/yEBFvmA>)，有详细的概念介绍。需要注意的是，glsl是严格的类型匹配，int和float进行运算会出错。

**顶点着色器的目标是输出顶点，所以gl\_Position必须赋值**

```

attribute vec4 position;
attribute vec2 textCoordinate;
uniform mat4 rotateMatrix;
varying lowp vec2 varyTextCoord;
void main()
{
    varyTextCoord = textCoordinate;

    vec4 vPos = position;

    vPos = vPos * rotateMatrix;

    gl_Position = vPos;
}

```

- 如下，这是一个片元着色器。注意，在片元着色器，数字变量都要有类似lowp的精度描述。

片元着色器的目标是输出像素颜色，gl\_FragColor必须赋值

```
varying lowp vec2 varyTextCoord;
uniform sampler2D colorMap;
void main()
{
    gl_FragColor = texture2D(colorMap, varyTextCoord);
}
```

这里有一个详细的博客 (<http://blog.csdn.net/racehorse/article/details/6593719>), 讲得很好。

### 3、简单图形变换

几何变换有比例、旋转、平移、对称、错切，这里我们介绍简单的旋转变换。  
先给出结论：对于一个图形进行旋转变换，相当于对每个顶点乘以一个旋转变换矩阵。  
矩阵如下：

$$(x' \ y' \ z' \ 1) = (x \ y \ z \ 1) \begin{bmatrix} \cos \gamma & \sin \gamma & 0 & 0 \\ -\sin \gamma & \cos \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

绕z轴旋转

旋转矩阵.png

对于顶点的变换，我们可以放在OC代码里面来实现，把顶点变换完成后，把顶点输入到OpenGLES；也可以在glsl代码实现，把顶点变换交给gpu来完成。这里我们采用的是后者。  
如下：

```
GLuint rotate = glGetUniformLocation(self.myProgram, "rotateMatrix");

float radians = 10 * 3.14159f / 180.0f;
float s = sin(radians);
float c = cos(radians);

//z轴旋转矩阵
GLfloat zRotation[16] = { //
    c, -s, 0, 0.2, //
    s, c, 0, 0, //
    0, 0, 1.0, 0, //
    0.0, 0, 0, 1.0 //
};

//设置旋转矩阵
glUniformMatrix4fv(rotate, 1, GL_FALSE, (GLfloat *)&zRotation[0]);
```

细心的开发者会发现，这里的z轴旋转矩阵和上面给出来的旋转矩阵并不一致。  
究其原因就是OpenGLES是列主序 ([http://blog.lazybee.me/d3dopengl\\_matrix/](http://blog.lazybee.me/d3dopengl_matrix/))矩阵，对于一个一维数组表示的二维矩阵，会先填满每一列（a[0][0]、a[1][0]、a[2][0]、a[3][0]）。

把矩阵赋值给glsl对应的变量，然后就可以在glsl里面计算出旋转后的矩阵。

### 思考题

- 1、为什么熊猫的反的？要如何解决？
- 2、在这个样例中，顶点着色器调用次数和片元着色器调用次数哪个多？
- 3、glsl里面的变量可以通过glUniform进行赋值，那么是否可以在编译成功后或者链


接成功后直接进行赋值？

## 总结

这一篇的内容作为教程2难度有点大，特别是shader和glsl语言容易让人兴趣直接降到谷底，如果觉得难，可以暂时不用管glsl语言。  
待熟悉GLKBaseEffect后，再回来学习glsl也不迟。  
代码点我 (<https://github.com/loyinglin/LearnOpenGLS/tree/master/Tutorial02-shader%E5%85%A5%E9%97%A8>)

## 思考题答案

- 1、纹理坐标系的(0, 0)在左下角；
- 2、片元着色器。顶点着色器调用次数与顶点数量有关，片元着色器调用与像素多少有关系。
- 3、一个一致变量在一个图元的绘制过程中是不会改变的，所以其值不能在glBegin/glEnd中设置。一致变量适合描述在一个图元中、一帧中甚至一个场景中都不变的值。一致变量在顶点shader和片断shader中都是只读的。首先你需要获得变量在内存中的位置，这个信息只有在连接程序之后才可获得。



落影loyinglin (/u/815d10a4bdce)

写了 170235 字，被 4748 人关注，获得了 2741 个喜欢

(/u/815d10a4bdce)写了 170235 字，被 4748 人关注，获得了 2741 个喜欢

+ 关注

工程师一枚，喜欢思考，喜欢游戏，喜欢运动。做过什么已经不重要，未来的方向以及当下的准备是生活的...

赞赏是支持别人，也是加深自己的记忆。

赞赏支持



(/u/15893823363f)

 喜欢 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-like-button)

| 24







更多分享

(<http://cwb.assets.jianshu.io/notes/images/3360526>



后发表评论

(/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-comment-form)


61条评论 

只看作者

按喜欢排序

按时间正序

按时间倒序



Somnus\_chh (/u/2894ac97cddd)

16楼 · 2016.12.01 09:57

(/u/2894ac97cddd)

虽不明但觉厉，第一篇文章熊猫对称，没弄出来，看了这篇竟然把对称做出来了，。。。😏



👍 1人赞    💬 回复



陈\_某\_某 (/u/64bc83898735)  
2楼 · 2016.06.02 14:23

(/u/64bc83898735)  
感觉有点难，教程1看了半天，感觉不理解，后来结合其它博客才勉强理解了。看了这个demo完全傻掉了。。。

👍 赞    💬 回复

落影loyinglin (/u/815d10a4bdce): @\_iOSer (/users/64bc83898735) 我尽量简化多余的代码以及添加注释。光看代码不够，教程1后面附带的链接有更详细的解析。

2016.06.02 14:42    💬 回复

✎ 添加新评论



陈\_某\_某 (/u/64bc83898735)  
3楼 · 2016.06.02 16:24

(/u/64bc83898735)  
1.熊猫是反的，因为纹理坐标系（0，0）在左下角。。  
3.应该不可以吧，glsl跟c语言差不多。不是面向对象？？

👍 赞    💬 回复



落影loyinglin (/u/815d10a4bdce) 作者  
4楼 · 2016.06.02 21:46

(/u/815d10a4bdce)  
一个一致变量在一个图元的绘制过程中是不会改变的，所以其值不能在glBegin/glEnd中设置。一致变量适合描述在一个图元中、一帧中甚至一个场景中都不变的值。一致变量在顶点shader和片断shader中都是只读的。  
首先你需要获得变量在内存中的位置，这个信息只有在连接程序之后才可获得。

👍 赞    💬 回复



魁拔2015 (/u/5f4bea531f06)  
5楼 · 2016.06.25 21:03

(/u/5f4bea531f06)  
你好 那个熊猫的图标相反的原因是纹理坐标系的原因 那如果解决呢？ 我没有找到解决方案 希望指点一下。

👍 赞    💬 回复

落影loyinglin (/u/815d10a4bdce): @魁拔2015 (/users/5f4bea531f06) 用GLKit的话，读取的时候可以设置对应的参数即可，在后面教程有写。

2016.06.25 21:31    💬 回复

魁拔2015 (/u/5f4bea531f06): @落影loyinglin (/users/815d10a4bdce) 那如果不用GLKit呢 是否能够解决？

2016.06.25 21:35    💬 回复

落影loyinglin (/u/815d10a4bdce): @魁拔2015 (/users/5f4bea531f06) 如果不用GLKit，可以在设置纹理坐标的时候，上下的坐标颠倒即可。也可以用正常的坐标，但是要修改变化矩阵。

2016.06.25 22:06    💬 回复

✎ 添加新评论    |    还有8条评论， 展开查看



陈阿票 (/u/2db58466d30f)  
6楼 · 2016.09.05 15:56

(/u/2db58466d30f)  
太难了，看晕了

👍 赞    💬 回复



junghao (/u/645ff0691817)

7楼 · 2016.09.20 09:59

(/u/645ff0691817)

請問為什麼

<https://github.com/loyinglin/LearnOpenGLS/blob/master/Tutorial02-shader>

(<https://github.com/loyinglin/LearnOpenGLS/blob/master/Tutorial02-shader>)入門/LearnOpenGLS/LearnView.m#L215

這裡產出是self.myColorFrameBuffer，但是在Bind時卻是self.myColorRenderBuffer？而且似乎不管改成哪個對於結果都沒影響？

謝謝。



赞



回复

落影loyinglin (/u/815d10a4bdce)： @junghao (/users/645ff0691817) 用不太严谨的描述 framebuffer=colorBuffer+depthBuffer+STENCILBuffer

2016.09.20 10:40 回复

junghao (/u/645ff0691817)： @落影loyinglin (/users/815d10a4bdce) 我是說 self.myColorFrameBuffer = buffer; // 设置为当前 framebuffer glBindFramebuffer(GL\_FRAMEBUFFER, self.myColorRenderBuffer);

第一行指派給myColor-FRAME-Buffer 但是下一行bind的時候，bind的是 myColor-RENDER-buffer 整個.m的其他地方也沒有用到myColor-FRAME-Buffer 所以是否筆誤了？

2016.09.21 06:12 回复

落影loyinglin (/u/815d10a4bdce)： @junghao (/users/645ff0691817) 你好，是笔误。正确的写法应该是glBindFramebuffer(GL\_FRAMEBUFFER, self.myColorFrameBuffer); 感谢指正。

2016.09.21 10:19 回复



添加新评论

还有2条评论， [展开查看](#)



YapheeetS (/u/291824c9920c)

8楼 · 2016.10.27 16:33

(/u/291824c9920c)

我也知道纹理坐标原点是(0, 0)。。但是熊猫是反的,应该如何解决啊？？



赞



回复

YapheeetS (/u/291824c9920c)： 刚开始学，还望多多请教啊

2016.10.27 16:45 回复

落影loyinglin (/u/815d10a4bdce)： @YapheeetS (/users/291824c9920c) 考虑把纹理坐标翻转下

2016.10.27 18:20 回复

YapheeetS (/u/291824c9920c)： @落影loyinglin (/users/815d10a4bdce) 翻转一下确实正了，但是在教程1和教程2中设置纹理坐标的方式是一样的，为什么教程1中的图片是正的呢？是因为系统的GLKBaseEffect对纹理坐标进行了翻转吗？

2016.10.28 10:59 回复



添加新评论

还有3条评论， [展开查看](#)



此晨有亮 (/u/4d62ff3796e1)

9楼 · 2016.10.28 16:27

(/u/4d62ff3796e1)

自定义的 看不懂 例如 attribute vec4 position;

attribute vec2 textCoordinate;

uniform mat4 rotateMatrix;

varying lowp vec2 varyTextCoord;

```
void main()
{
varyTextCoord = textCoordinate;

vec4 vPos = position;

vPos = vPos * rotateMatrix;

gl_Position = vPos;
}
```

函数里只用到了position rotateMatrix 为什么还要vec4 vPos = position varyTextCoord = textCoordinate不理解 为什么 二维等于四维

赞 回复

小山Sam (/u/bbabb44509eb): @Captain\_FL (/users/4d62ff3796e1) 一行四列的矩阵，乘以四行四列的矩阵，结果就是一行四列的矩阵

2016.11.04 19:30 回复

添加新评论



张霸天 (/u/6c46e1ac4ee0)

10楼 · 2016.10.31 20:13

(/u/6c46e1ac4ee0)  
0.5f, -0.5f, -1.0f, 1.0f, 0.0f,  
-0.5f, 0.5f, -1.0f, 0.0f, 1.0f,  
-0.5f, -0.5f, -1.0f, 0.0f, 0.0f,  
0.5f, 0.5f, -1.0f, 1.0f, 1.0f,  
-0.5f, 0.5f, -1.0f, 0.0f, 1.0f,  
0.5f, -0.5f, -1.0f, 1.0f, 0.0f,

我觉得你这个应该要注释下，因为新手很难理解为什么坐标需要这么写，有重复的地方，为了节省index

赞 回复

落影loyinglin (/u/815d10a4bdce): @zhangdazuiba (/users/6c46e1ac4ee0) 好建议

2016.10.31 21:38 回复

添加新评论



小山Sam (/u/bbabb44509eb)

11楼 · 2016.11.04 19:41

(/u/bbabb44509eb)  
“0.5f, -0.5f, -1.0f, 1.0f, 0.0f,”

为啥第三个数字是-1.0f？改成0效果一样，改成-1.1就看不到熊猫了

赞 回复



小山Sam (/u/bbabb44509eb)

12楼 · 2016.11.05 13:07

(/u/bbabb44509eb)  
@落影loyinglin (/users/815d10a4bdce) 我加了一个滤镜效果，  
https://github.com/chencan/LearnOpenGL/commit/9ba16ebcaab4de63b9590fe63d6da7cb504d6267  
(https://github.com/chencan/LearnOpenGL/commit/9ba16ebcaab4de63b9590fe63d6da7cb504d6267) 但glsl编译不过，能帮忙看看吗

赞 回复

落影loyinglin (/u/815d10a4bdce): @小山Sam (/users/bbabb44509eb) 像素着色器

2016.11.05 13:26 回复

落影loyinglin (/u/815d10a4bdce): @小山Sam (/users/bbabb44509eb) 精度没问题

2016.11.05 13:26

回复

小山Sam (/u/bbabb44509eb):


@落影loyinglin (/users/815d10a4bdce) 是说精度有问题吗?

2016.11.05 14:44

回复

添加新评论

还有4条评论， 展开查看

 哎疯 (/u/899e2d74bdeb)

13楼 · 2016.11.14 21:55

(/u/899e2d74bdeb)

请问问题1的答案是不是在加载纹理的时候（0，0）坐标在右上角，而纹理坐标的（0，0）是在左下角？导致图片颠倒？

赞

回复

落影loyinglin (/u/815d10a4bdce):

@哎疯 (/users/899e2d74bdeb) gl的坐标系 和 ios的坐标系不同导致

2016.11.14 23:47

回复

添加新评论

 初心\_媛 (/u/12db32328291)

14楼 · 2016.11.16 16:36

(/u/12db32328291)

楼主你好，最近在学习GPUImage,但是好多概念都不懂，需要从OpenGL 开始学吗？

赞

回复


落影loyinglin (/u/815d10a4bdce):

@初心\_媛 (/users/12db32328291) 不需要

2016.11.16 17:22

回复

添加新评论

 GSD\_iOS (/u/4e4c71af3fae)

15楼 · 2016.11.27 15:43

(/u/4e4c71af3fae)

很不错! 🍻🍻

赞







回复

1

2

下一页

被以下专题收入，发现更多相似内容

-  iOS Dev... (/c/3233d1a249ca?utm\_source=desktop&utm\_medium=notes-included-collection)
-  iOS入的那些坑 (/c/2ac262be8a53?utm\_source=desktop&utm\_medium=notes-included-collection)
-  OpenGL ... (/c/044a5240577d?utm\_source=desktop&utm\_medium=notes-included-collection)
-  iOS开发专题 (/c/c258bc0ea6bd?utm\_source=desktop&utm\_medium=notes-included-collection)
-  OPenGL ... (/c/408442c9c764?utm\_source=desktop&utm\_medium=notes-included-collection)
-  OpenGL+... (/c/5d2c87603bd3?)




utm\_source=desktop&utm\_medium=notes-included-collection)




IOS个人开发 (/c/b76f9973bf0f?

utm\_source=desktop&utm\_medium=notes-included-collection)

展开更多

 登录/注册  
为你个性化推荐内容

(/sign\_in?utm\_source=desktop&utm\_medium=notes-included-collection)

 下载简书App  
随时随地发现和创作内容

(/apps/download?utm\_source=desktop&utm\_medium=click-note-bottom-bind)