配置VS：详B

***大致：***

***1、创建控制台新项目***

1. ***工具->NuGet包管理器->管理解决方案的Nu->搜索glfw->点该项目->安装***
2. ***项目名字右键->属性->链接器->输入->附加依赖项新增->输入“opengl32.lib”***
3. ***项目名字右键->属性->VC++目录->包含目录->编辑增加->找到glad所在位置加入include文件夹->加好后 源目录->编辑增加->找到glad所在位置加入src文件夹->应用->确定***
4. ***项目名字右键->添加->现有项->找到glad下src内的glad.c->添加***

还可？：

***1、创建控制台新项目***

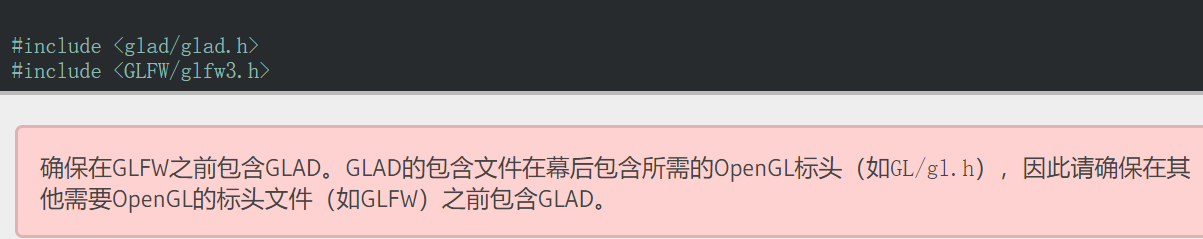
***2、工具->NuGet包管理器->管理解决方案的Nu->搜索nupengl->点该项目->安装***

***3、项目名字右键->添加->现有项->找到glad下src内的glad.c->添加***

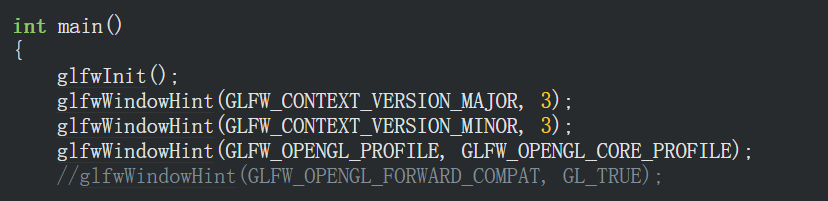
使用注意：

1. 基本框架

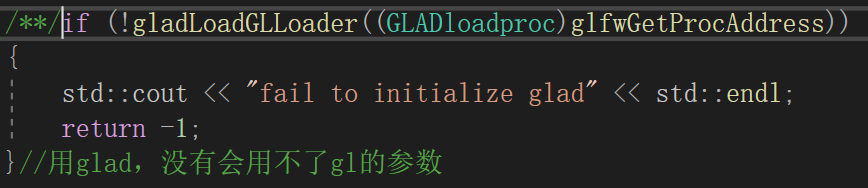
Include头：**GLAD在GLFW前**



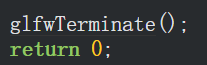
Main开头先初始化glfw库，注意参数



中间：必须要加入这一段，不然用不了后面的处理



结束：终止还原库



1. 一些基本函数

**·创建窗口**：

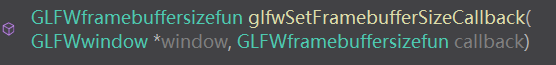


参数：长度、高度、名字

返回：窗口指针（GLFWwindow类型）

实例：glfwCreateWindow(800, 600, "Try my window", NULL, NULL);

**·回调BUFFERSIZE函数**

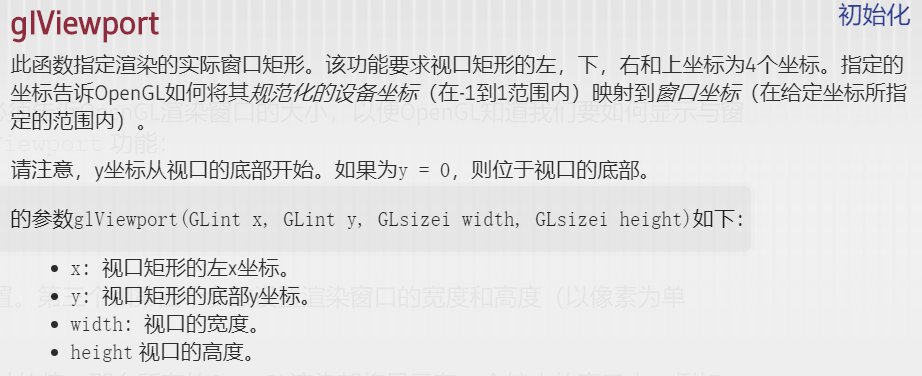


参数：窗口指针、回调函数名

返回：（可以先不管）

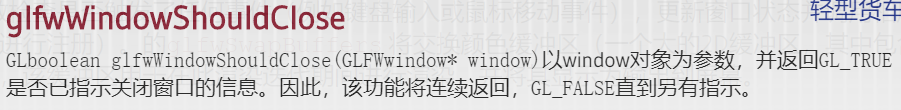
实例：glfwSetFramebufferSizeCallback(window, framebuffer\_size\_callback);

**·glViewport函数**



（**why没用？？？**）

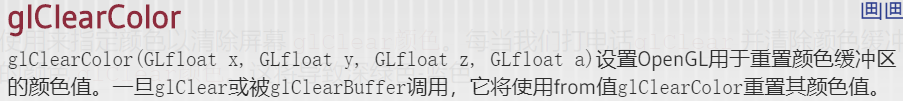
**·glfwWindowShouldClose及glfwSetWindowShouldClose**



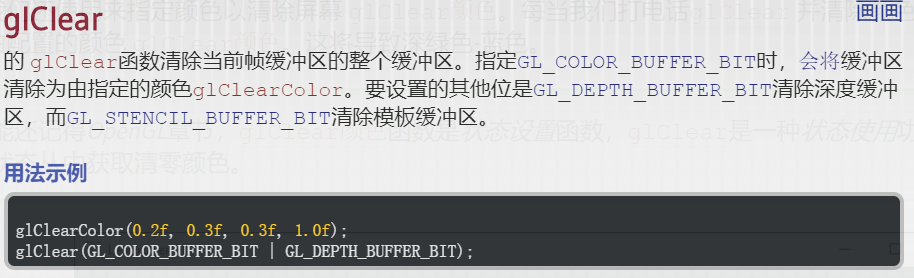
glfwSetWindowShouldClose(窗口指针, true/false);

设置要不要关，下次检查glfwWindowShouldClose就会判断关不关

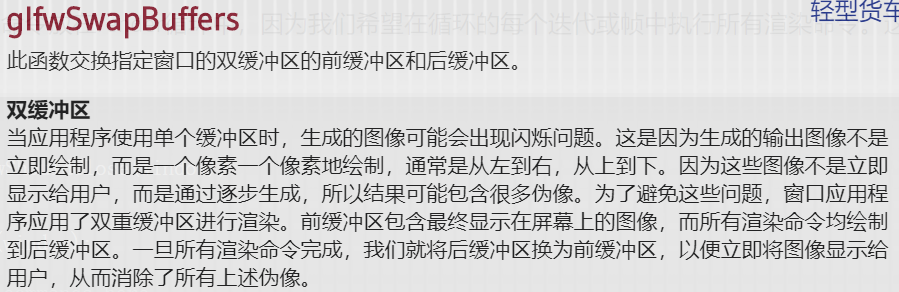
**·glClearColor与glClear**



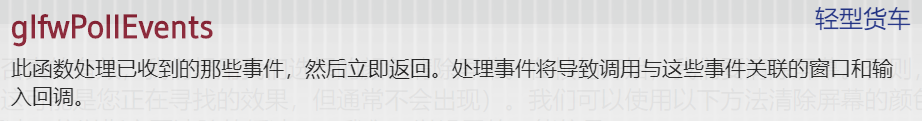
参数：R,G,B,透明度



**·glfwSwapBuffers(窗口指针)**



**·glfwPollEvents()**



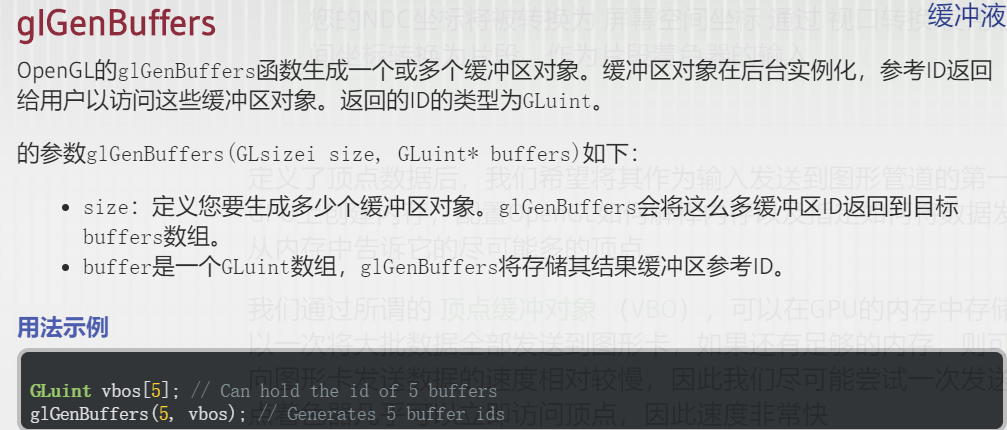
**·**

3、图形渲染管道（简单生成一个三角形）

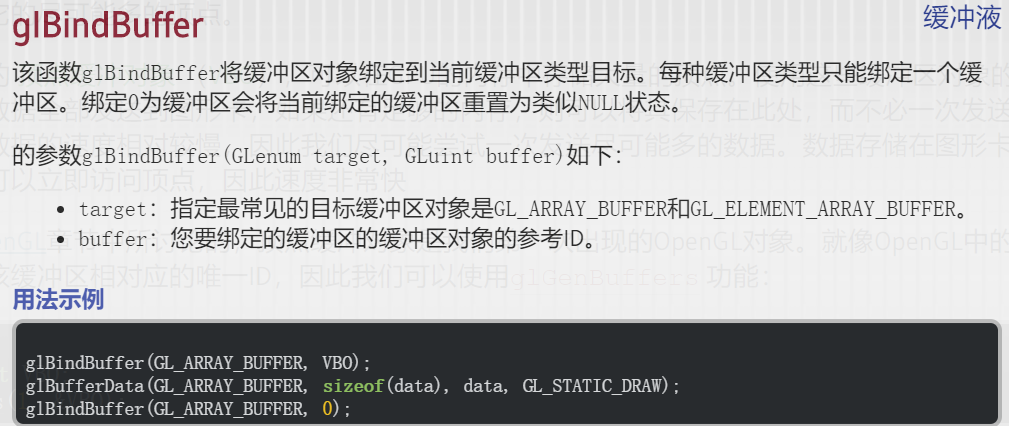
那几步，至少需要我们自己编写顶点着色器以及片段着色器

***····首先第一步要输入坐标，在GPU开辟缓冲空间然后传入数据***：

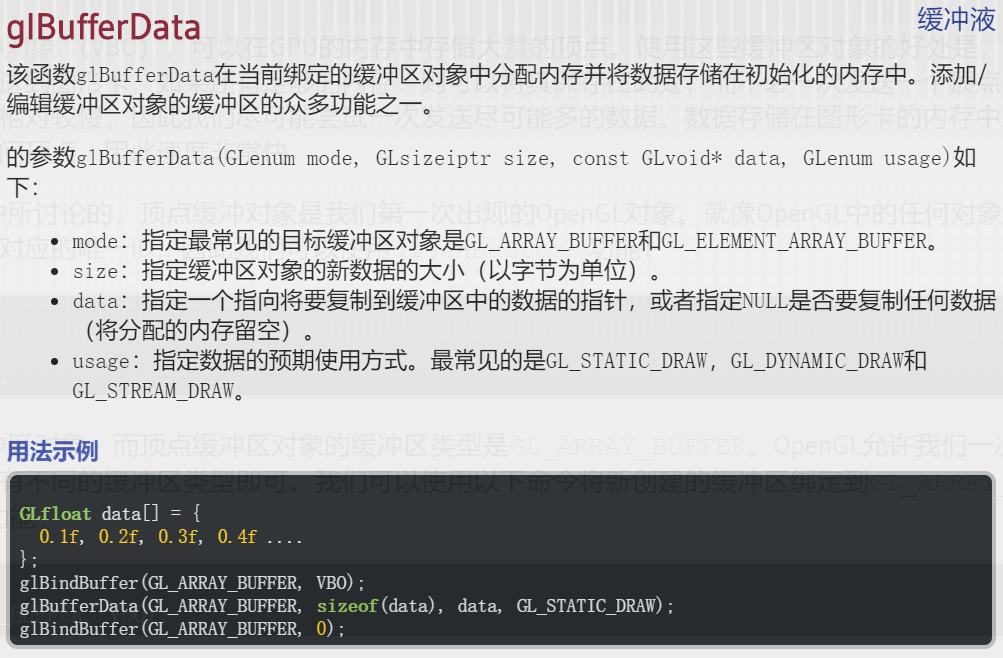
**·glGenBuffers**

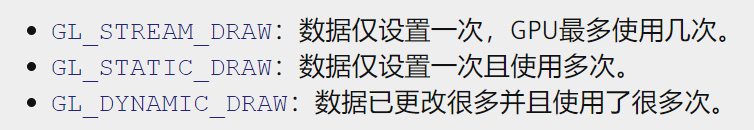


**·glBindBuffer**

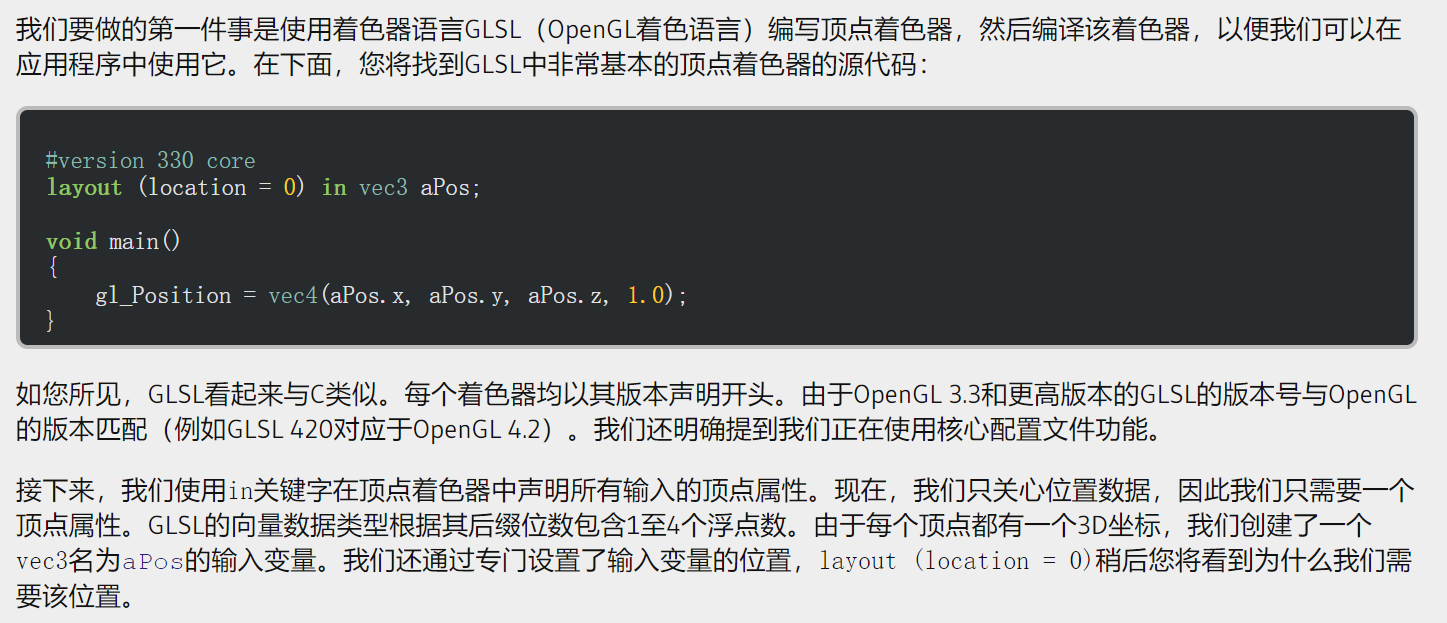


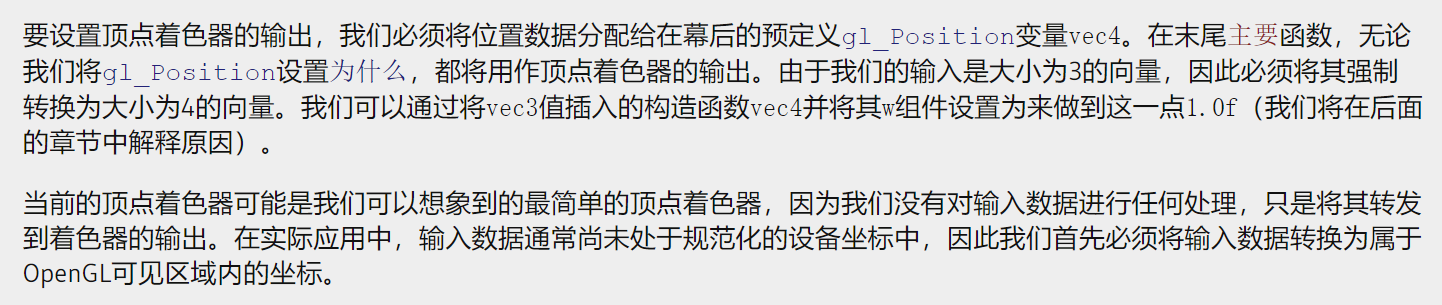
**·glBufferData**





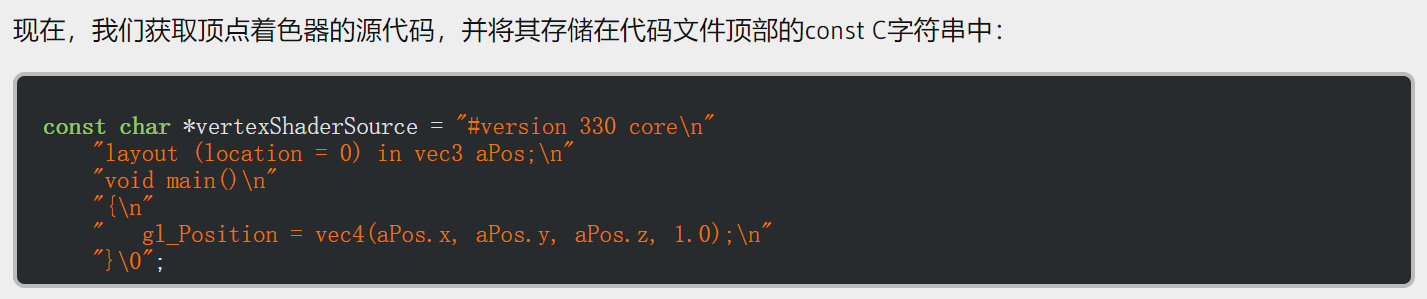
***····然后开始顶点着色器：***



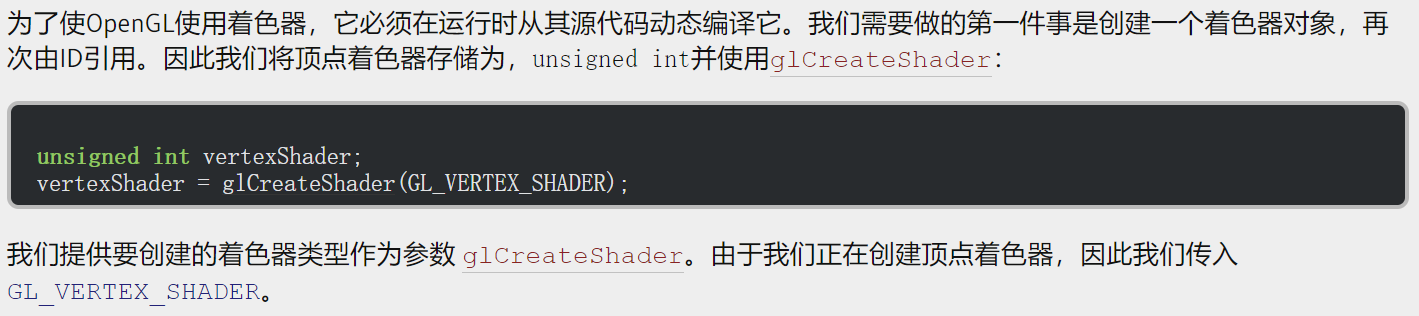


上面只是介绍，那么需要自己编辑这个顶点着色器的步骤：

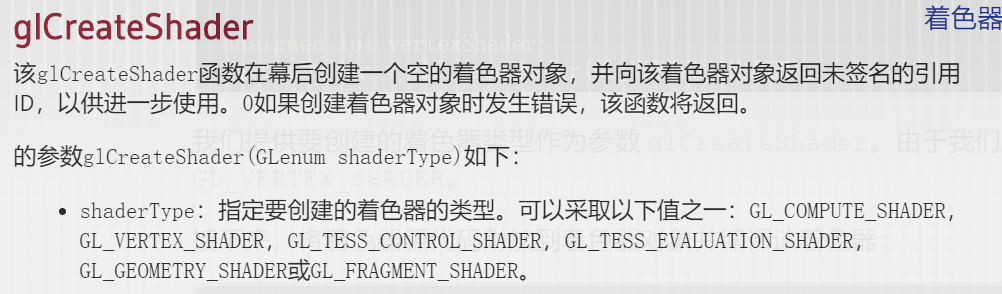
1）将源代码存入字符串



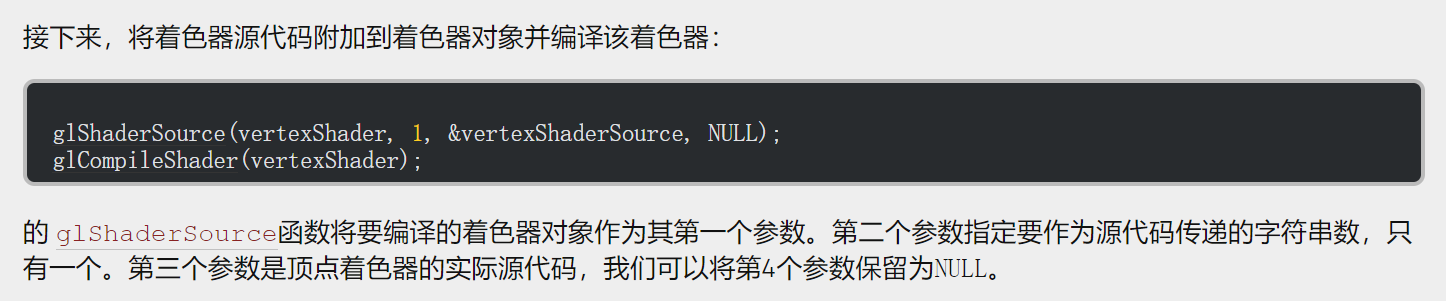
2）创建着色器对象



**·glCreateShader**



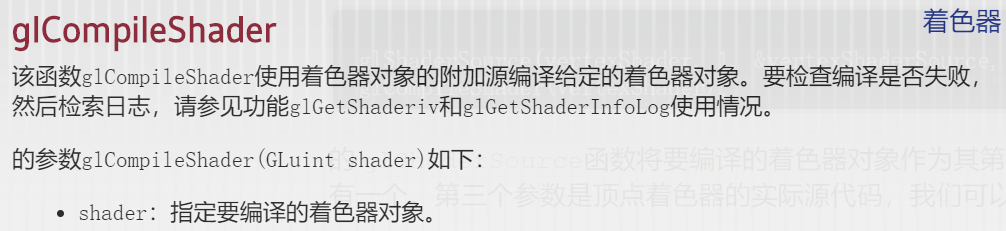
3）将源代码附加到着色器对象



**·glShaderSource**



**·glComplierShader**

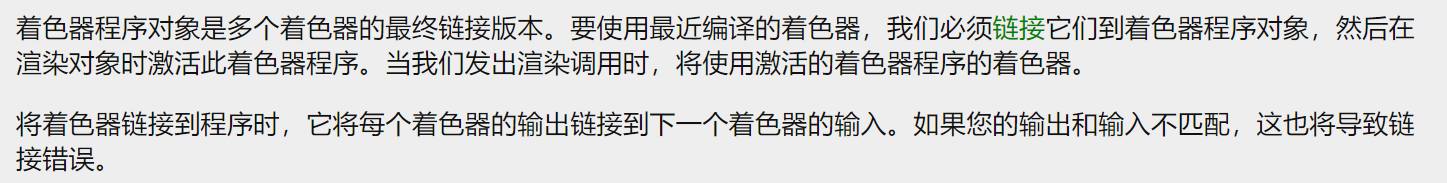


（可以检索是否出错详见网站）

***····然后开始片段着色器：***

和顶点着色器类似，函数也是类似，只是参数不一样（详见网站）

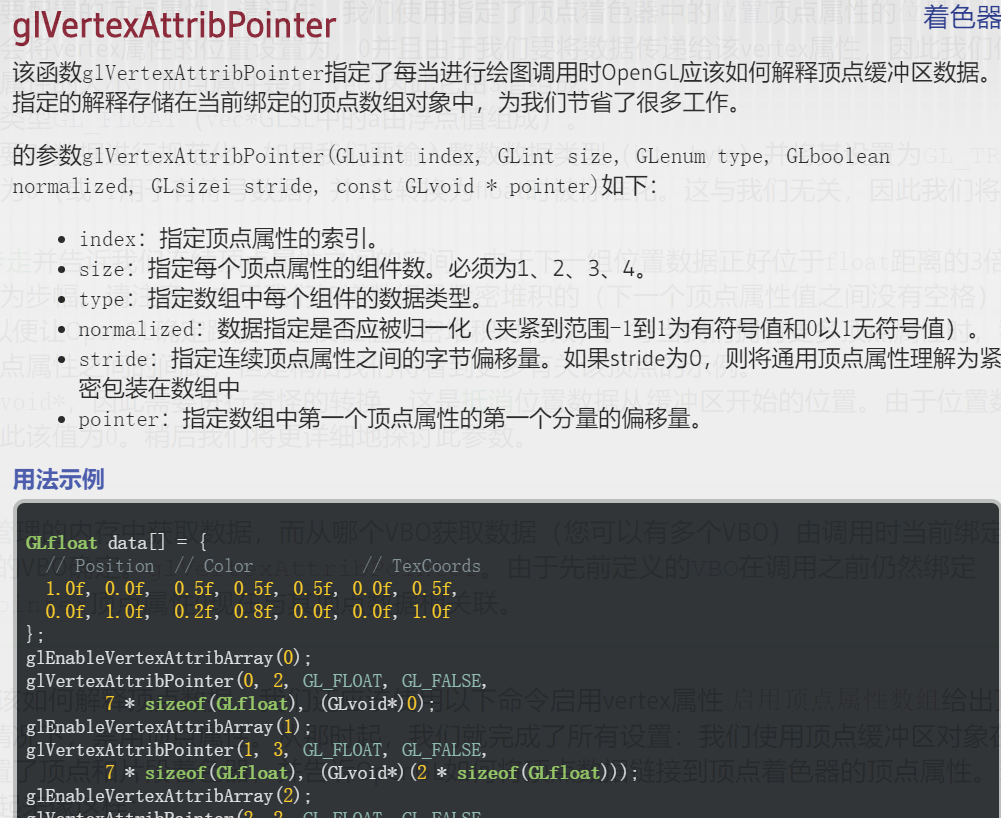
***····链接着色器：***

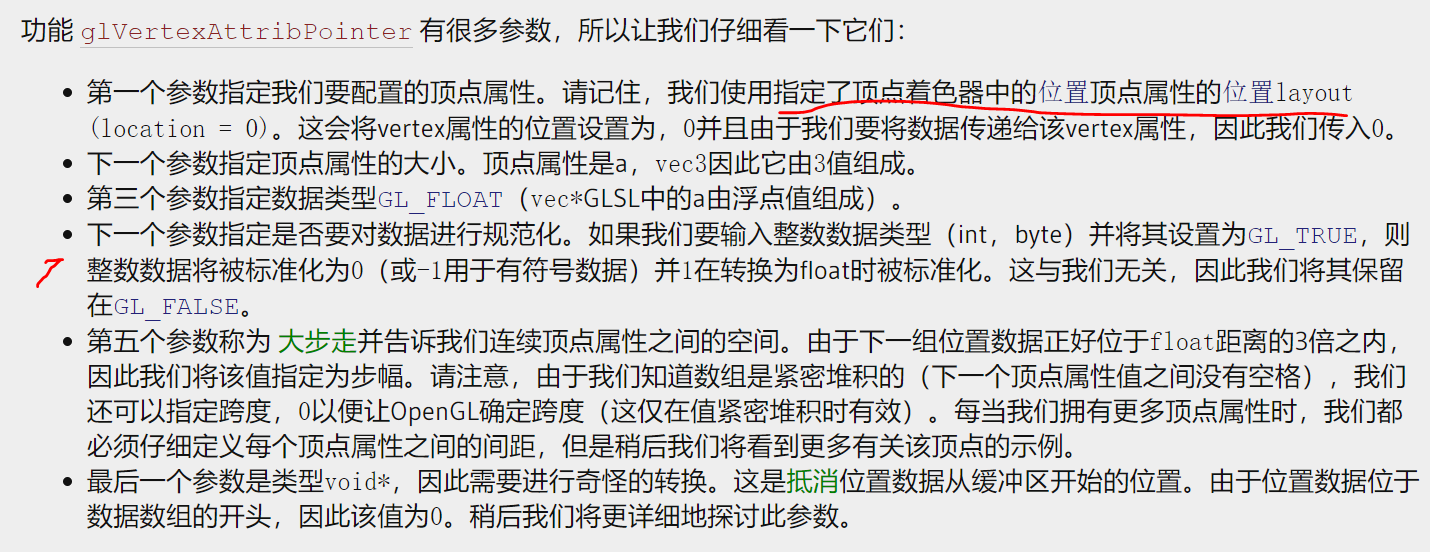




***····告诉如何解释顶点数据***

**·glVertexAttribPointer**



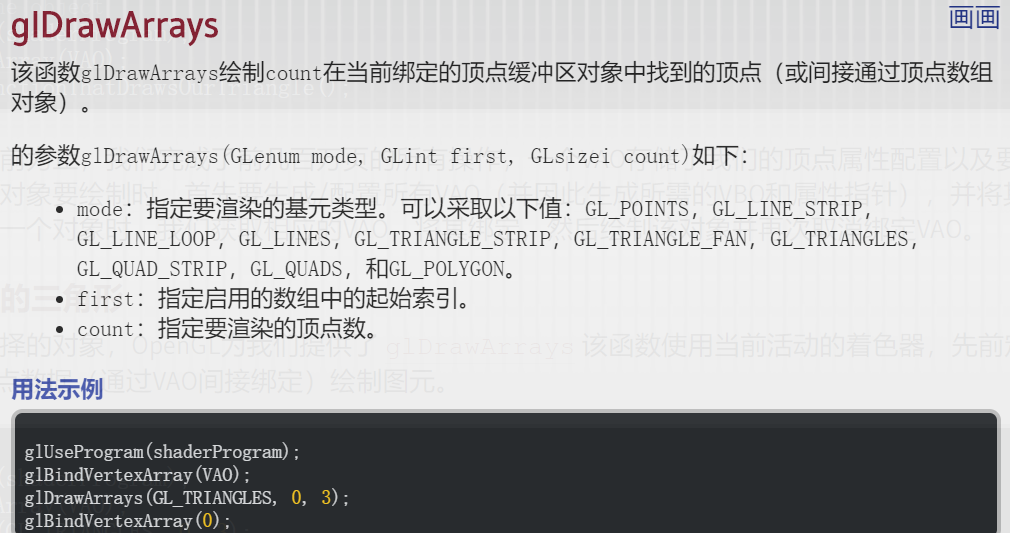


然后就可以enable这个链接

***····但是要每次生成VBO都要进行这个重复，因此加入VAO，绑定***

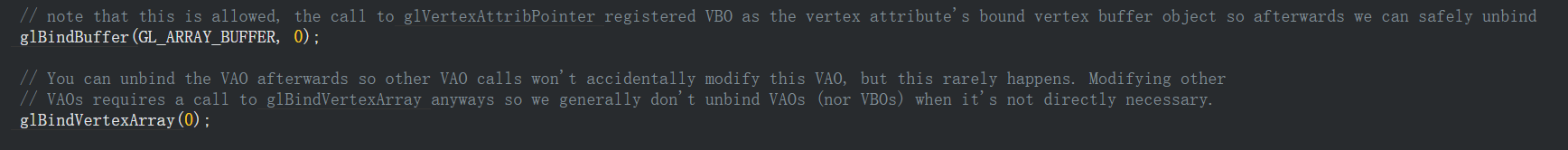


***····绘制***

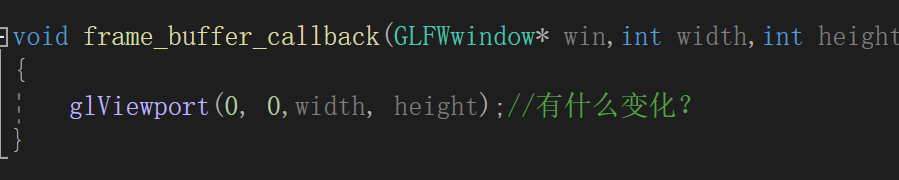


问：

1.有啥用？



2.没效果？



3.多加一个窗口为啥没显示了？

总体来讲：步骤分为：

（1）初始化，开个窗口……（详见第一）

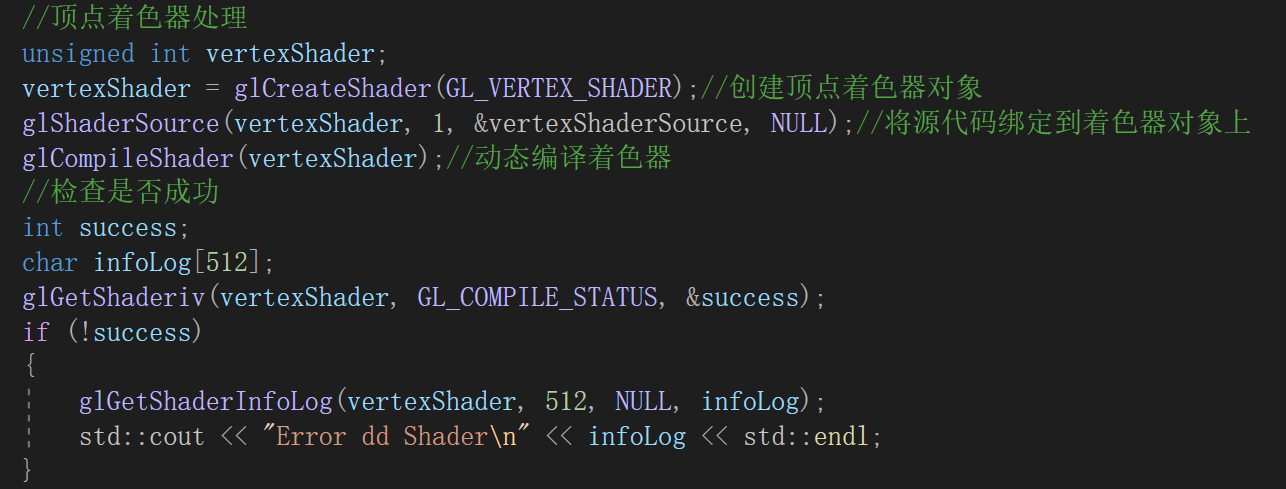
（2）着色器处理

着色器是驻留在GPU上的小程序

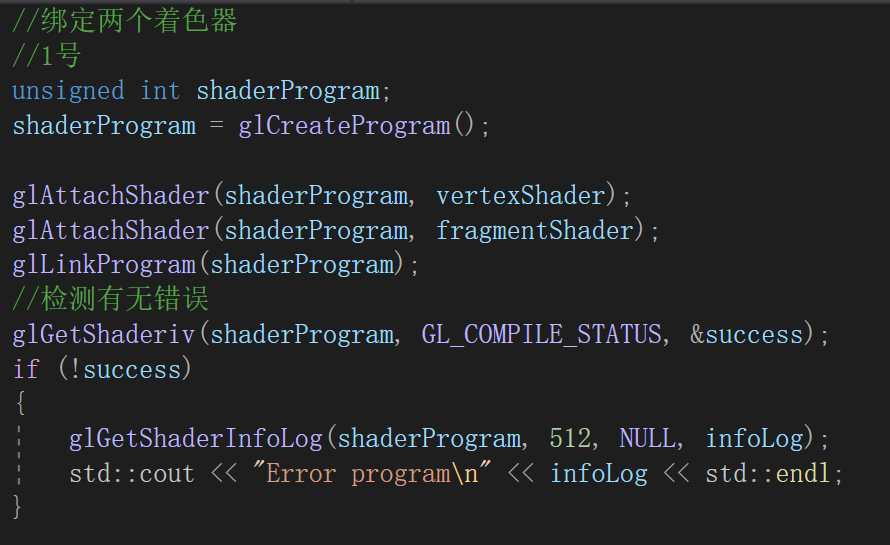
目前是顶点着色器以及片段着色器，它们分别使用字符串运行时编译然后连接形成一个着色器处理程序。实际上就是怎么处理数据点的过程。

顶点着色器处理输入的点集，片段着色器处理上色。

可以编写很多着色器代码，然后组合形成不一样的着色器（程序）。

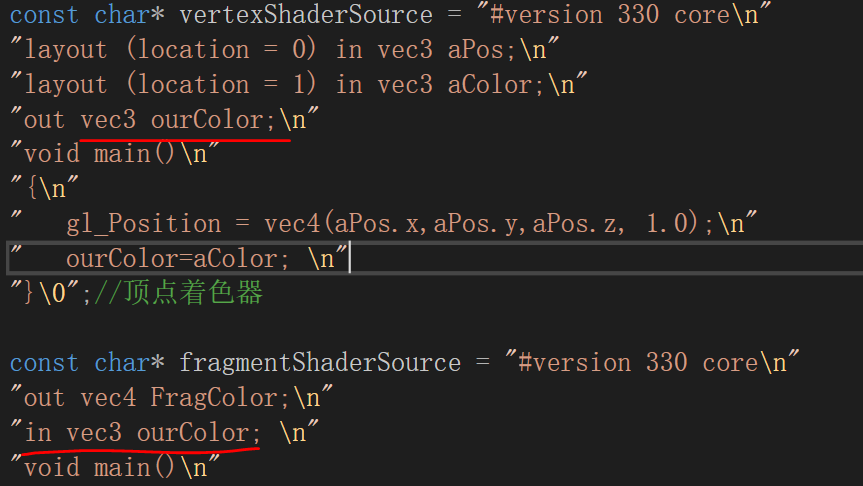


然后绑定一组着色器生成着色器过程程序：



不同的着色器程序可以处理得到不同的图形显示模式。

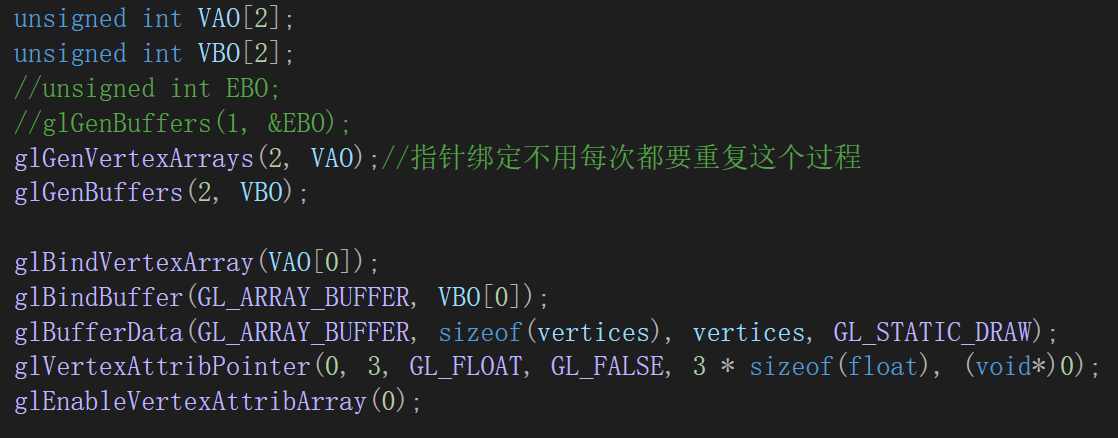
**注意！！从一个着色器输出给另一个着色器输入，这个变量名必须一样！！！**



（3）数据处理

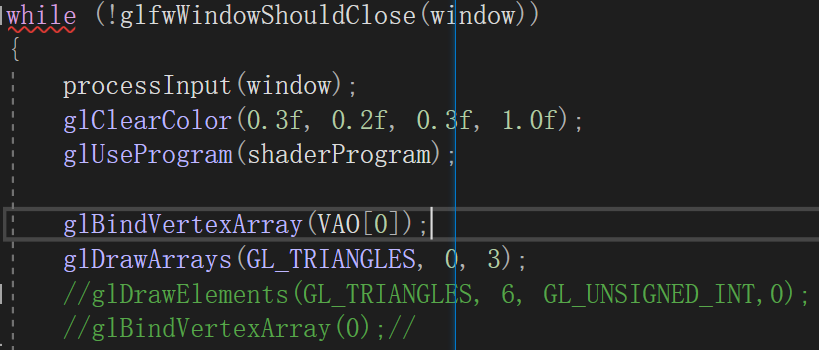
给出顶点的坐标之后还得告诉opengl怎么使用，so用VAO（一个指针？）、VBO（一个缓存）、EBO（当你想画矩形？给点和索引来画）

申请好缓冲区要bind绑定、向其中传入数据等等：



有多少个VBO、VAO就可以画出多少个不一样的图形。

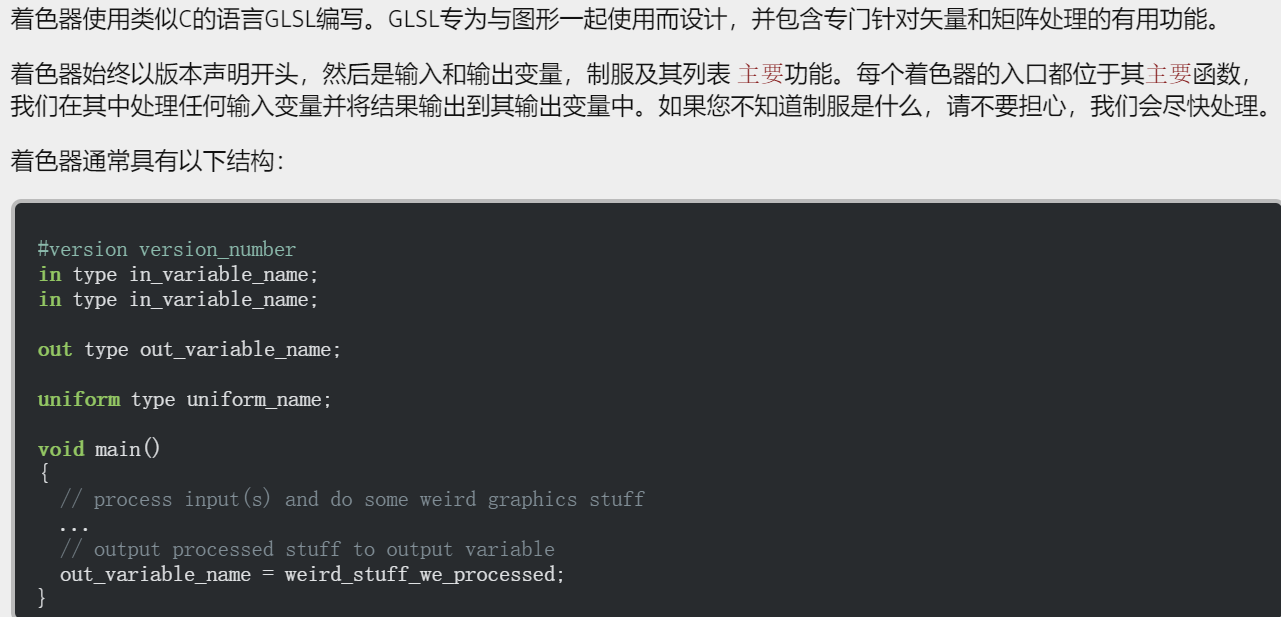
（4）开始显示





使用不同的着色器程序可以得到不同的效果结果

**着色器详解**：



~用uniform可以从CPU传输数据到GPU进行控制显示

~用in就得从输入赋值（要么是上一个着色器的输出，要么是你给的数组数据中赋值）

Layout是告诉它从哪个位置进行读取（因为给的数组可能有**多个元素**（位置、颜色vec）你得告诉从哪个位置开始是这个元素的输入，同时注意在VBOVAO也要进行改变，传输不同数据）

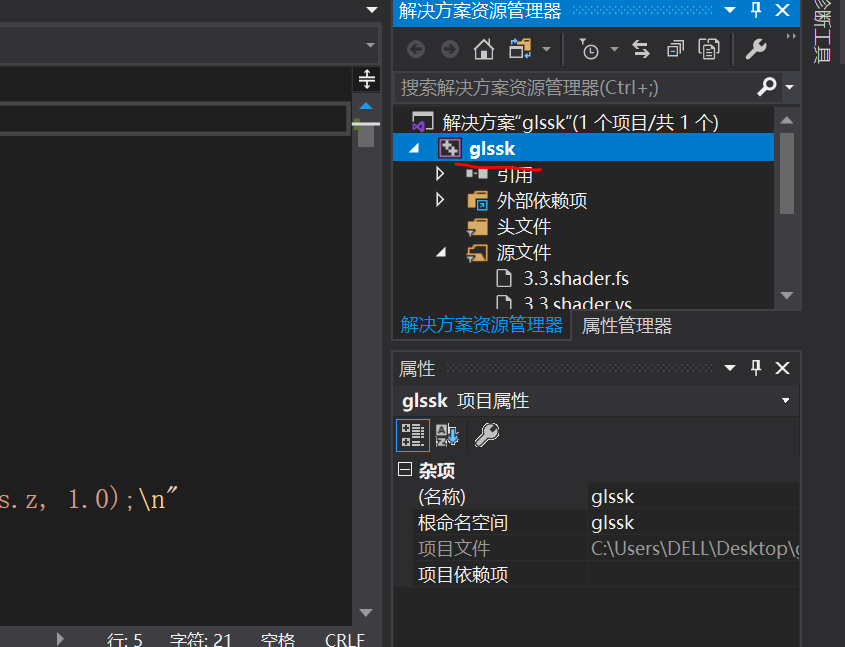
动态变化——Uniform（使得能够从程序动态控制着色器输入）

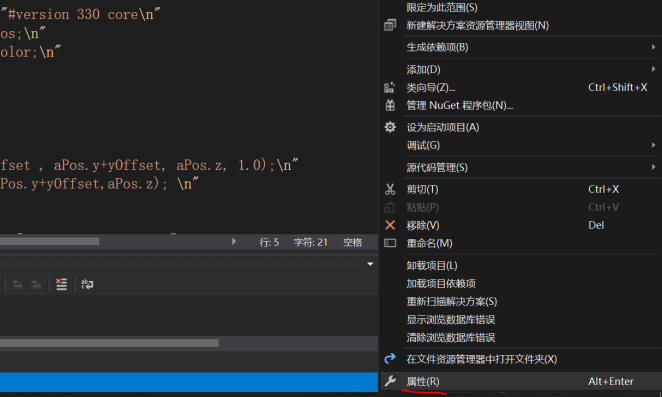
画多个图——多个着色器程序（so多个连接、多个VAOVBO）

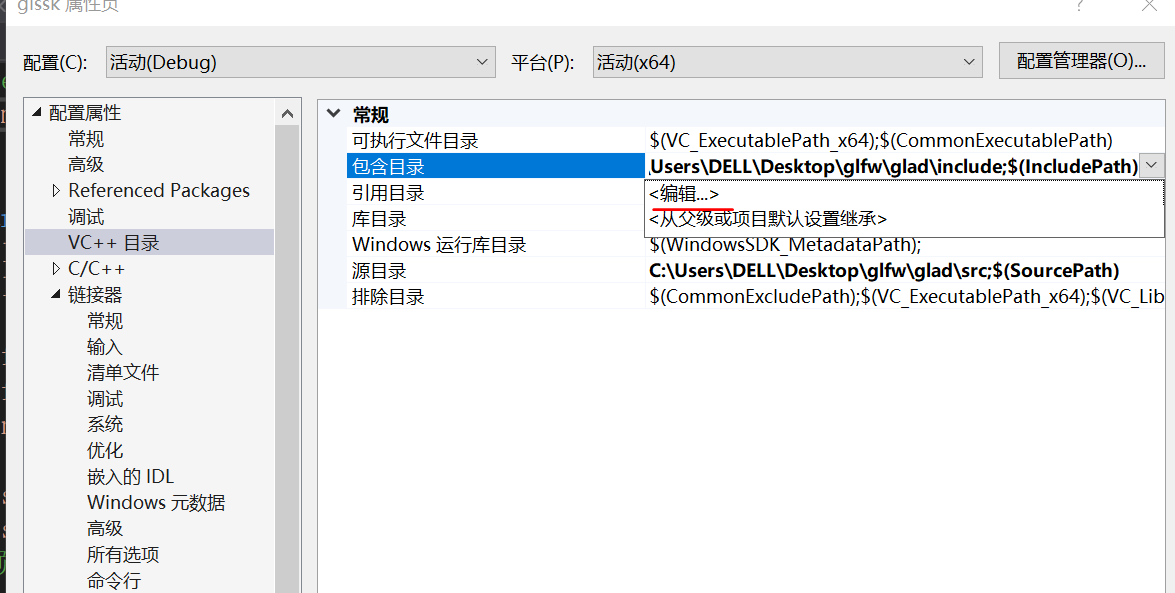
控制画出来的静态属性——在输入数组增加属性（so增加VBOVAO连接传输数据数（不是VAO个数）、着色器增加输入控制）

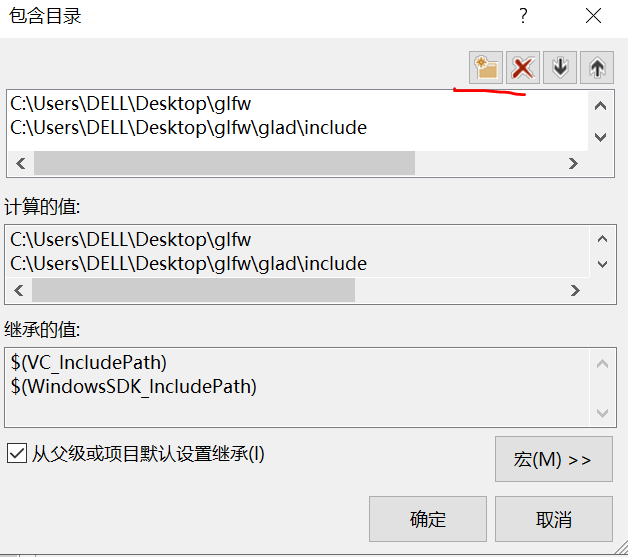
VAOVBO个数——对应多少个着色器程序——对应多少画出的对象

**\*\*\***着色器可以搞个头文件、类，注意要把它的路径添加到包含路径下（以后要使用某些文件都要先添加那个文件的路径）方法：

右键

属性

编辑

然后找你的文件在的目录添加就好了

然后搞好路径，再右键-添加项-现有项-找这个文件（这一系列不知道要不要）

**纹理**

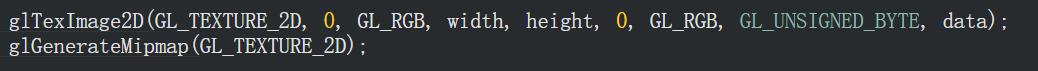
使用stb\_image.h头文件

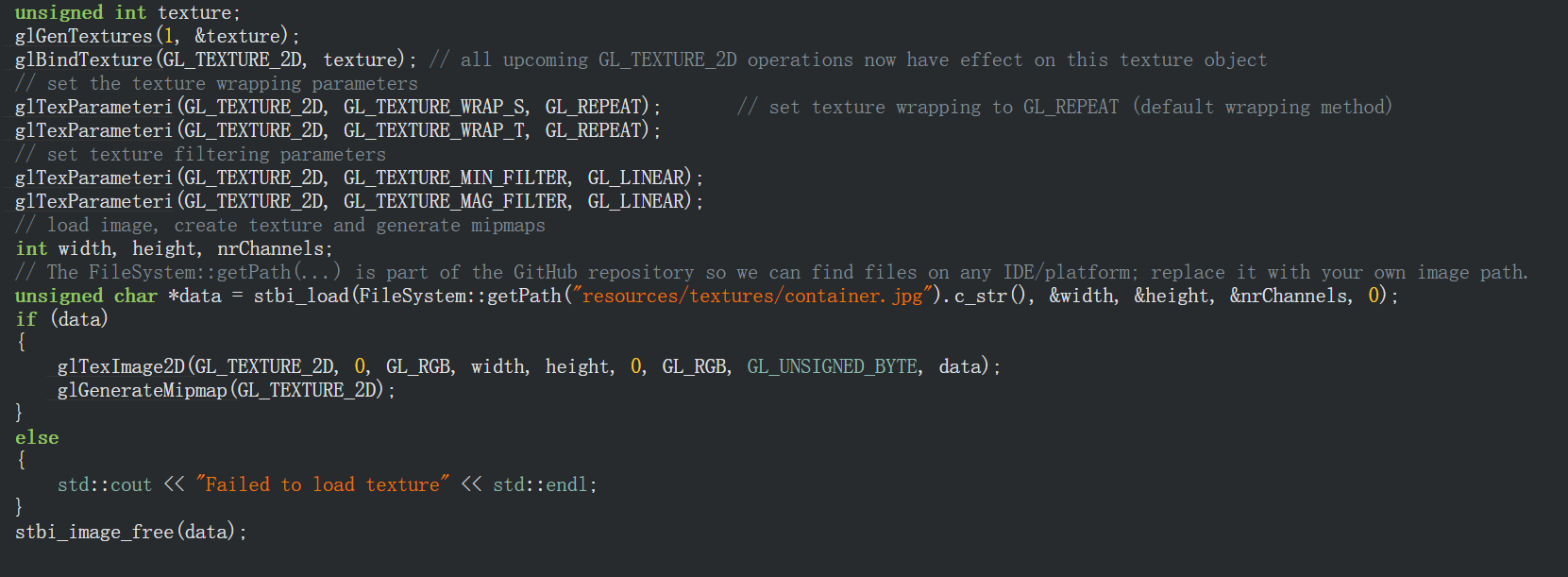
创建texture对象、绑定

glTexParameteri调一堆东西

打开贴图stbi\_load

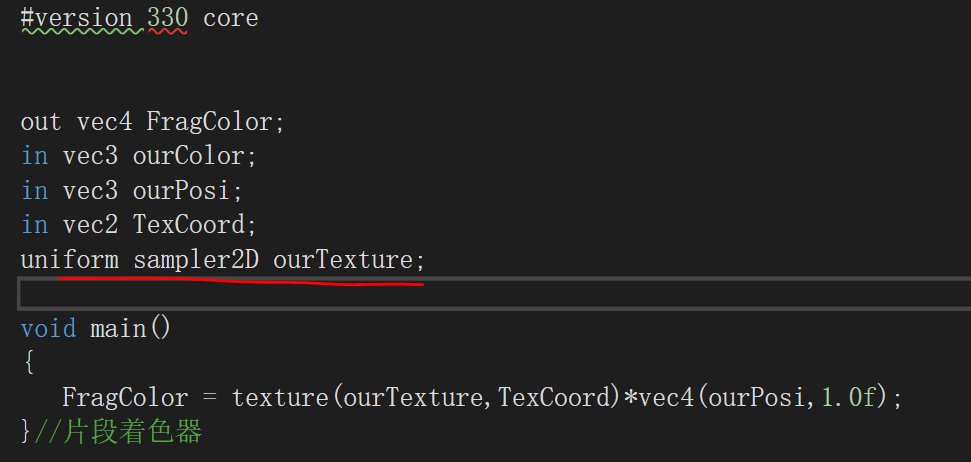
形成mipmap：





**问**：

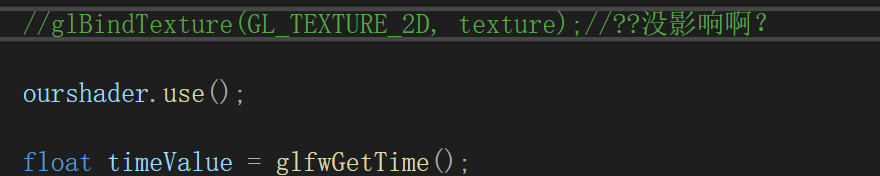
1、这个uniform从哪传进去的？



有默认值？

对，其实还是需要自己去赋值比较好

2.没影响？

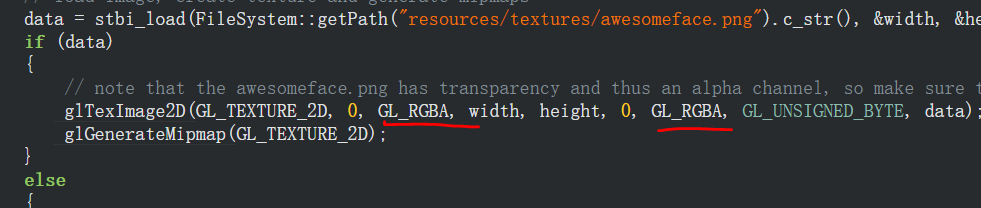


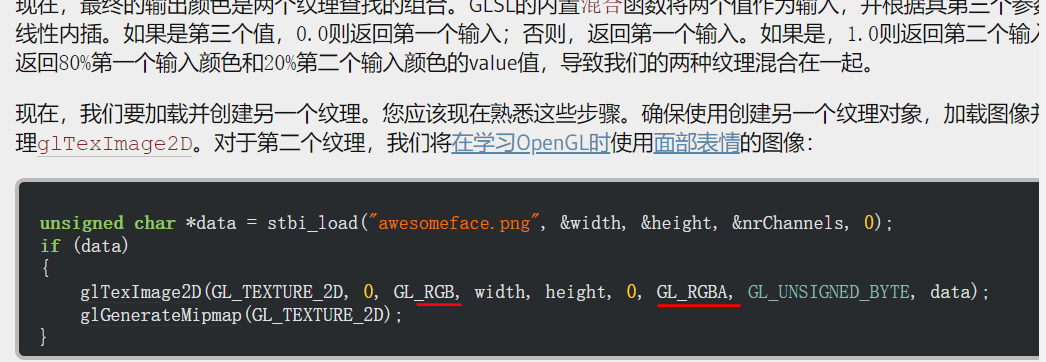
多个可能有影响

混合多个纹理——定义多个纹理对象，然后在片段着色器用mix混合

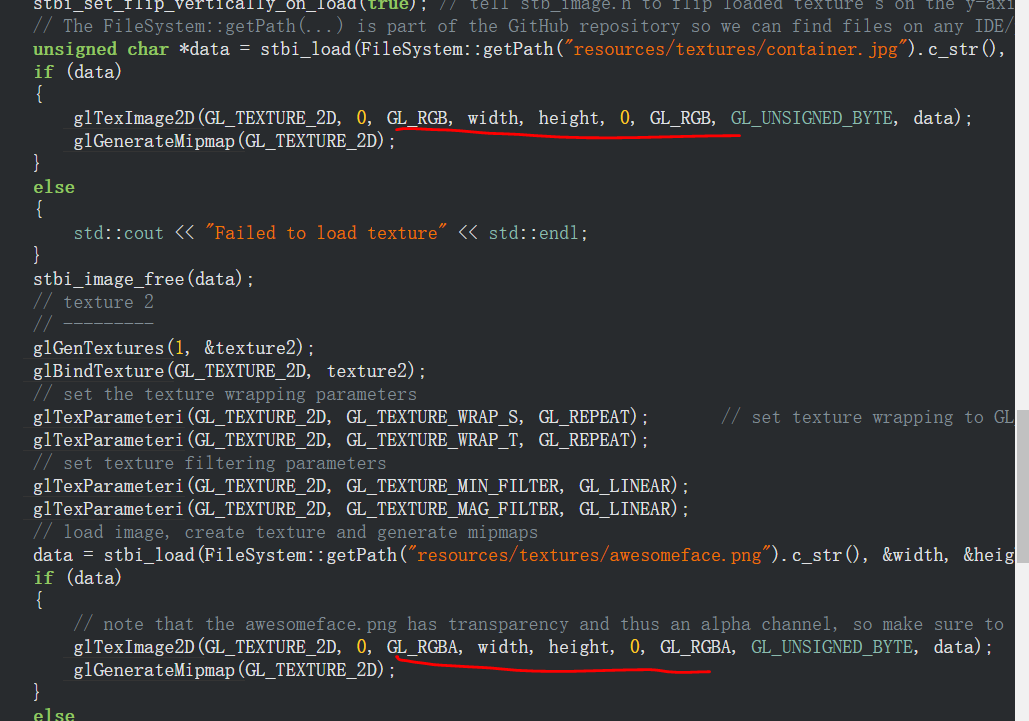
**出问题？？？**

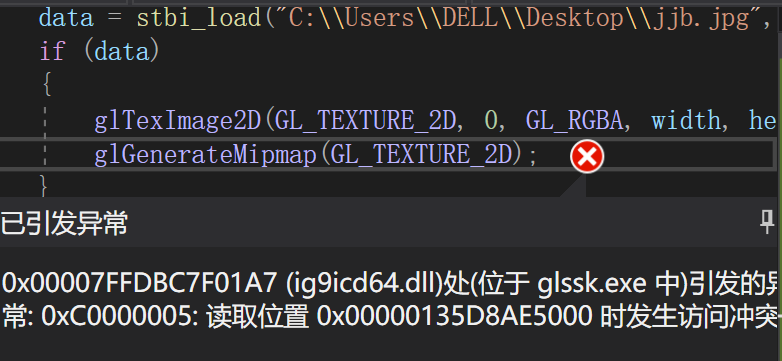
1、网站上



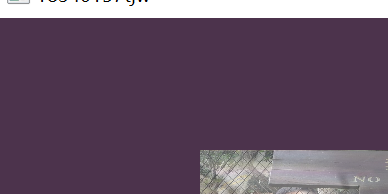


2.按网站的意思：第一个纹理用RGB、RGB，第二个用RGBA、RGBA但是报错

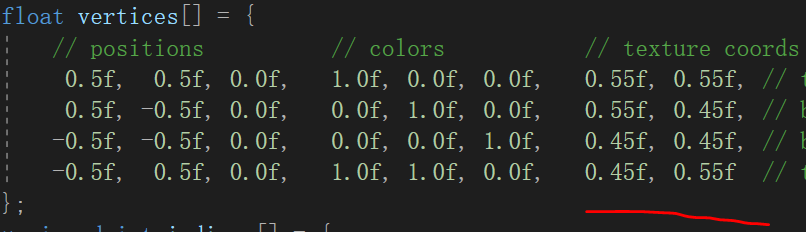




而我两个都用RGB就能运行：



3.这个贴图的坐标如果放成2就会那种重复图，放小成这样就是贴一部分？Why原理





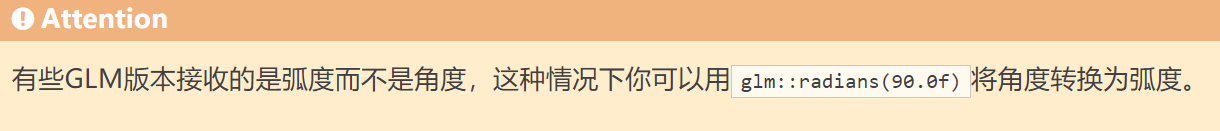
**glm**

Opengl没有自己的坐标变换，因此需要引入glm

以后加需要多包含glfw里面那个glm目录

然后代码加入：





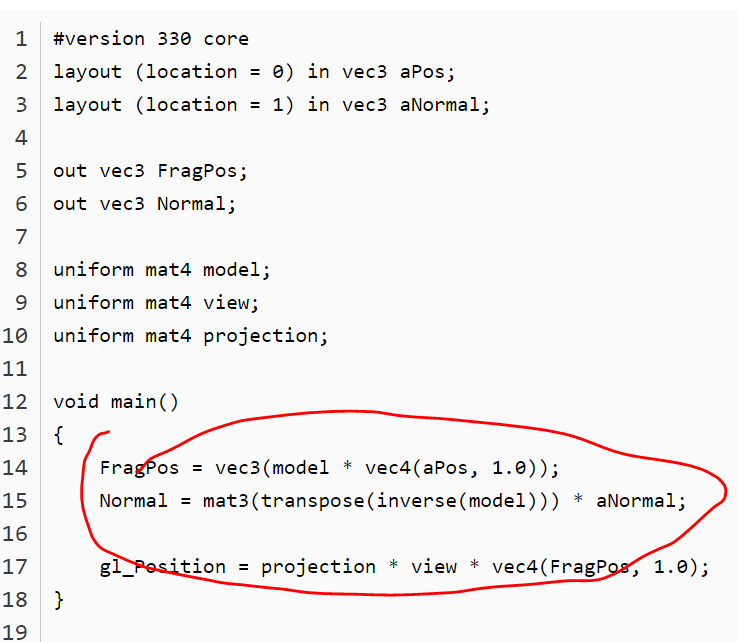
**·glBindBuffer**

**·glBindBuffer**

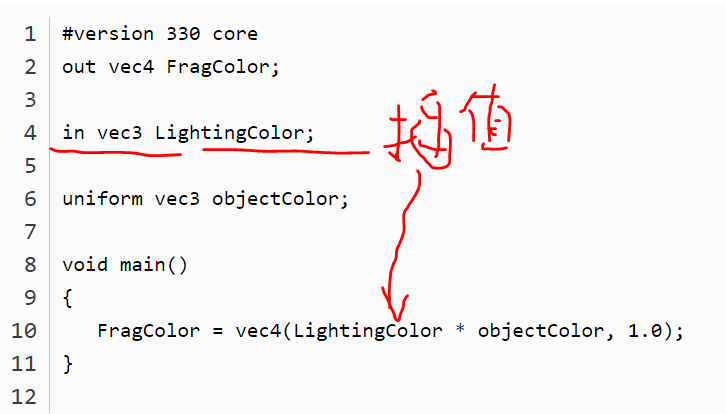
**········**

顶点着色器就是对顶点进行处理，片段着色器是对片段内（顶点围成的区域）的像素点进行处理。

顶点着色器——对顶点进行main函数内的处理



片段着色器——对这个片段，传入的in参数用顶点进行插值之后，再去对这个片段内**每个像素**进行main函数内的处理



so进行phon shading时候，原理是先给出片段上顶点的法向量，然后根据顶点的法向量插值片段内每个像素的法向量，然后每个像素进行phon光照模型得到像素颜色。so应该从顶点着色器传入顶点的法向量，然后在片段着色器main函数内根据传入的法向量计算phon光照明模型（因为传入的顶点法向量会根据顶点插值到片段的每个像素之和才执行main函数的计算，所以就正好符合phon shading）

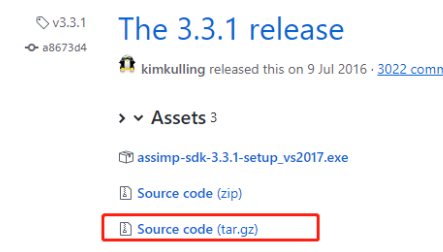
计算gouraud shading时候，原理是先根据phon光照明模型计算顶点颜色，然后再插值到顶点围成的片段的每个像素。so应该先在顶点着色器计算每个顶点的颜色，然后传入给片段着色器，片段着色器的main函数内就直接赋值颜色（因为传入的颜色在执行main函数之前会先根据顶点插值到每个片段内像素再执行main）

https://blog.csdn.net/qq\_36272282/article/details/89577774

**assimp加载模型库**

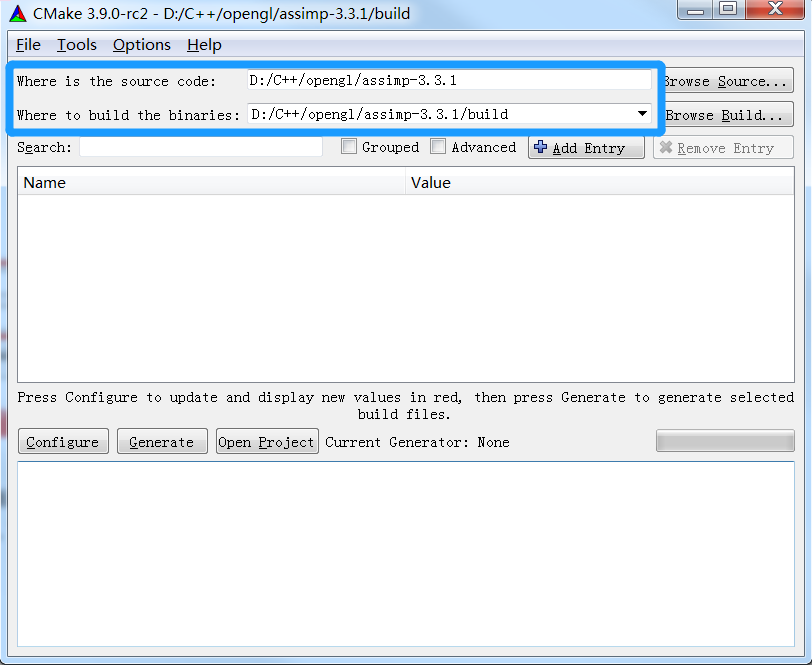
很难配……https://blog.csdn.net/wjl18270365476/article/details/102515966

1）准备好cmake；下载好assimp库https://github.com/assimp/assimp/releases/tag/v3.3.1/

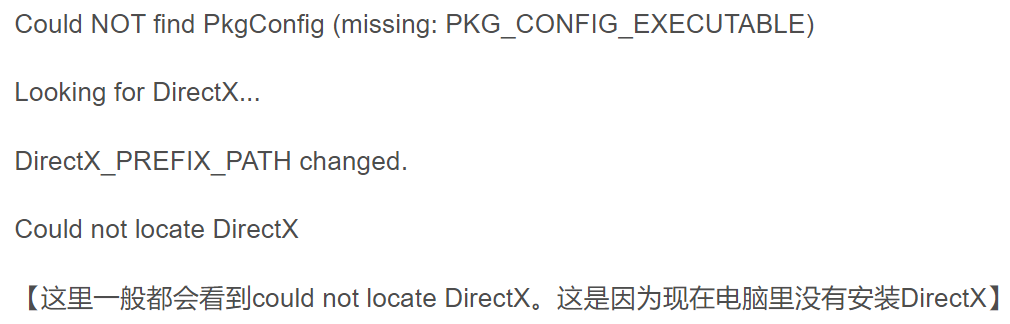


下载好后解压，并在文件夹内新建build文件夹

2）用CMake编译assimp工程



更改输入输出目录，输入为解压的大文件夹，输出为build文件夹；点击configure->选择自己的vs版本，finish之后就直接跑了，如果能成功就点generate。如果不能，就得安装DirectX！

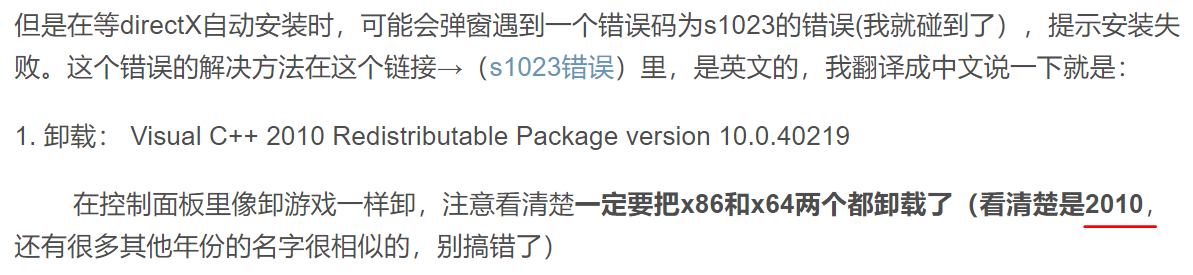


3）DirectX

下载：http://www.microsoft.com/en-us/download/details.aspx?id=6812

好像上面官网的，下不了，得自己搜下载：https://dl.pconline.com.cn/download/90643.html

下好之后就安装就好，但是安装时候又可能出现问题，解决：

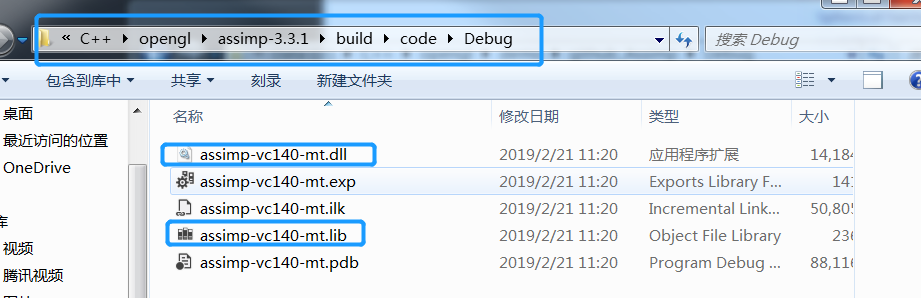


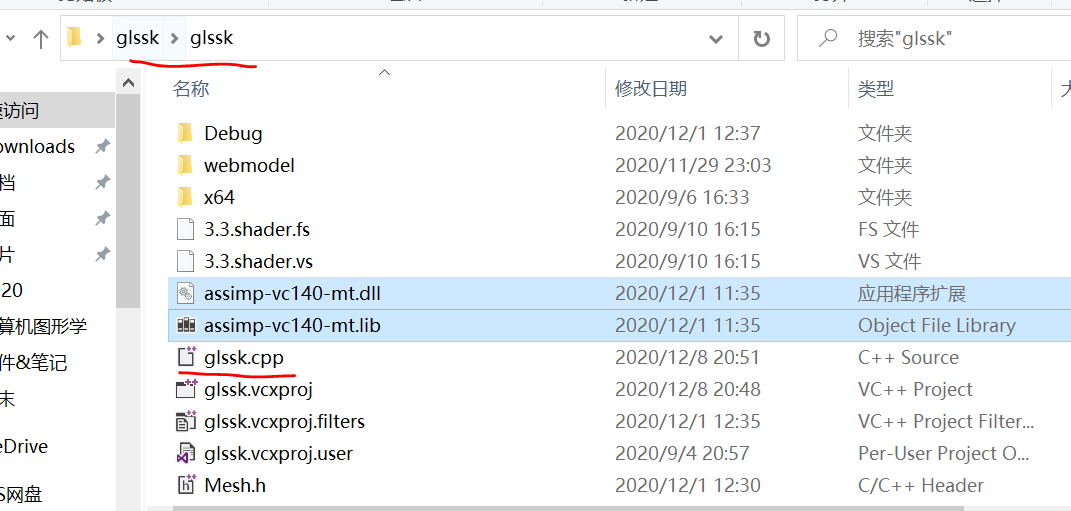
卸载好之后再安装就行了

4）装完之后再configure，然后generate就好了

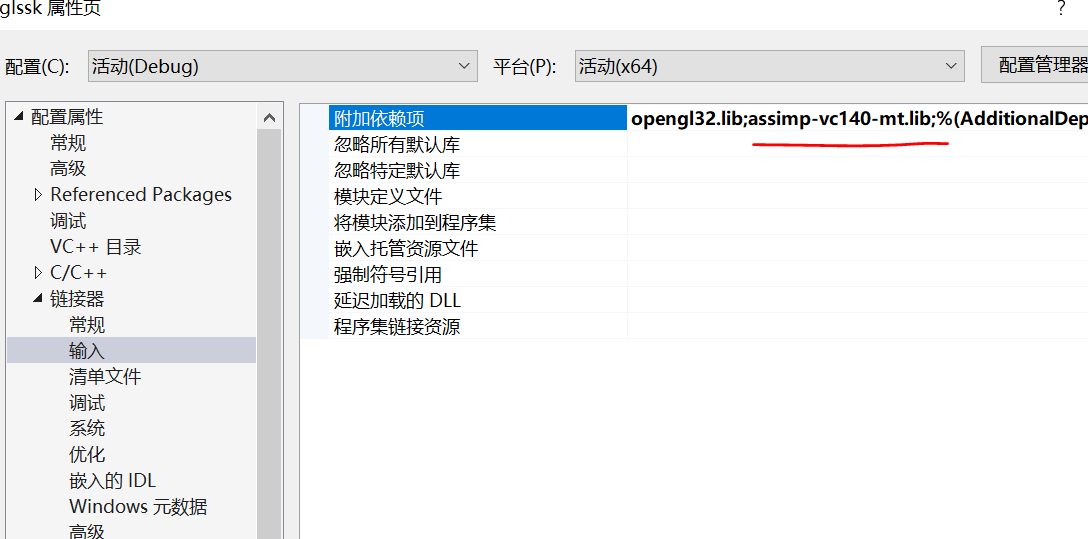
5）打开那个build文件夹，打开.sln文件，运行（就是绿色那个标志）；*这里好像我中途断掉了，但是也生成了dll和lib文件，不知道后面会不会有问题*

6）进入build文件夹下的code文件夹，把dll文件和lib文件复制到你的项目的文件夹，注意位置！！！！（应该是你的项目cpp文件的同个目录吧）





7）然后在vs内链接器，输入



（这里我也没按网上说的包含assimp目录，好像也执行了，不知道是不是对的）