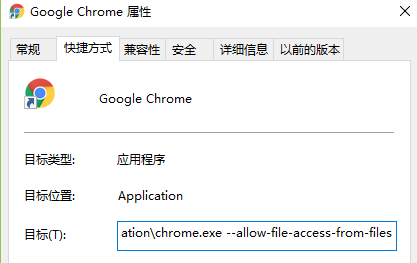
1. 提供了3个示例程序，都是导入的同一个obj文件（带法向、材质和纹理贴图，分别为obj文件、mtl文件和jpg文件），但3个示例程序采用不同的shader：
   1. 无光照无贴图：没有使用光照和纹理映射，只是利用了obj文件中的顶点坐标信息，绘制时使用白色，没有3D感；
   2. 有光照无贴图：使用了光照计算（在片元shader中计算光照），有3D感了；
   3. 有光照有贴图：进一步使用了纹理映射，最终的片元颜色根据光照计算的颜色以及纹理图像颜色计算得到，真实感更强了
2. 如果用Chrome运行示例程序，需要在Chrome的快捷方式上添加运行参数：



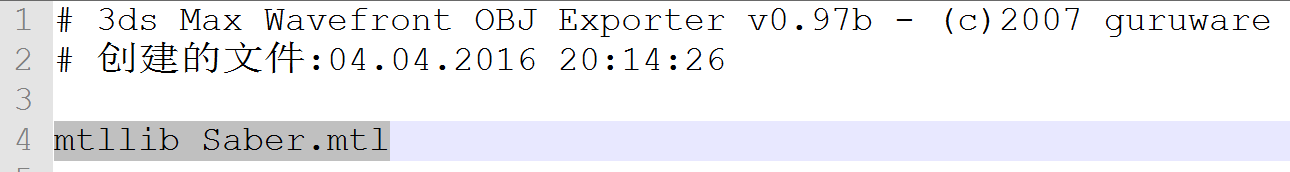
意思是允许访问文件，注意需要通过此快捷方式运行Chrome后，再打开示例程序html文件运行，如果直接右键点html文件选用Chrome打开，不会用到此快捷方式。

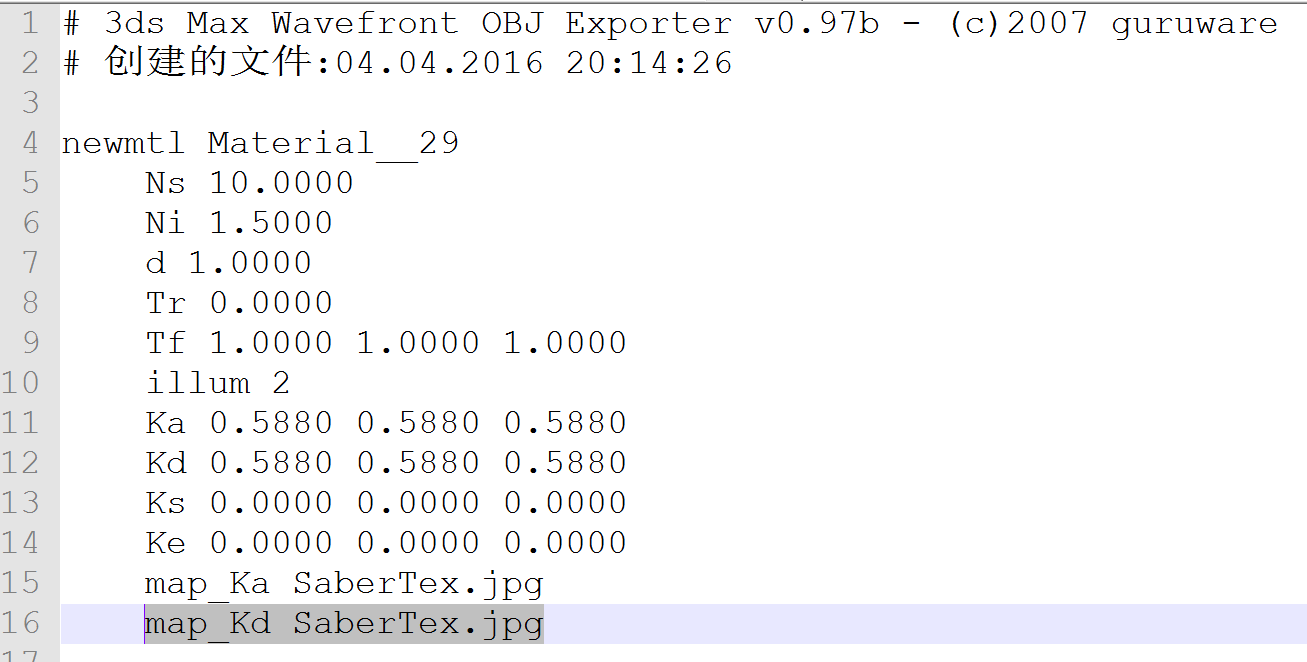
不同的浏览器设置不同，自行百度，有的浏览器（如ie）无需任何设置。

1. 在自己的程序中导入Obj文件的方法（请对照3个示例程序，查看以下步骤内容）：
   1. 在Common中新增js文件：ObjModel.js （示例程序中已提供）
   2. 在html文件中加载ObjModel.js：



* 1. 下载obj模型，或是自己用3dsMax或Maya之类的建模软件建模后导出为obj模型(注意纹理坐标设为UV展开形式)，用记事本打开obj文件和mtl文件，注意里面的文件路径是否正确，建议将obj、mtl连通纹理图像放在同一个文件夹下，然后obj和mtl中的文件路径只保留文件名即可：





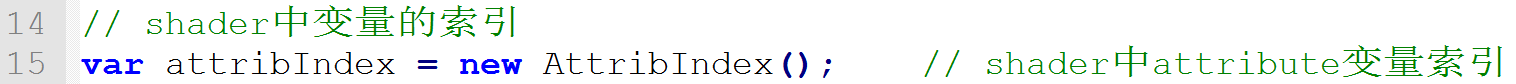
Mtl文件目前的库只使用了map\_Kd即漫反射贴图（但实际上在使用时不只影响到漫反射分量的颜色，计算方式可参见示例程序，原理会在后续课程中讲解）

* 1. 在js程序中定义全局变量，同时加载模型：

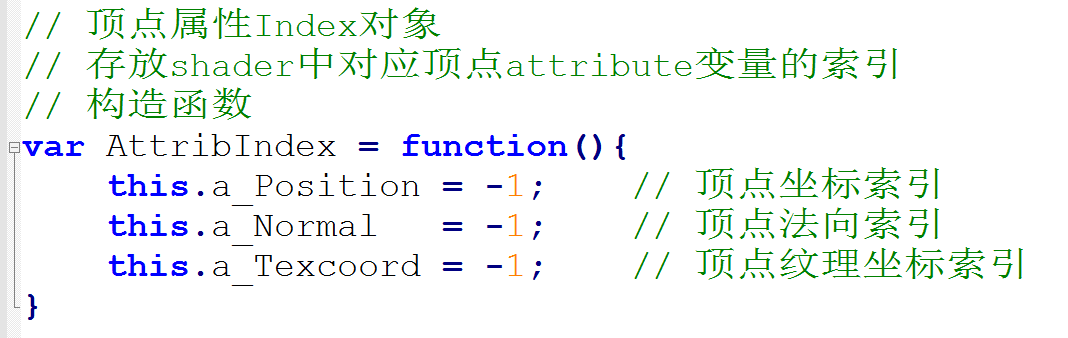


loadOBJ是ObjModel.js中提供的函数，注意参数中的相对路径是相对于js文件路径而言的（\\是转义字符，实际就是\），在此函数中会请求加载该obj文件及所有相关文件(mtl和纹理图像文件)，并读取所有文件内容，注意读取方式是异步方式，即js程序不会等文件读完，而是继续往下执行；

* 1. 定义全局AttribIndex对象：



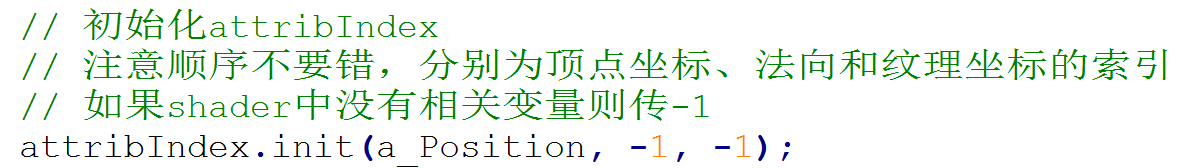
该对象主要用于存放shader中attribute变量索引，有3个分量，分别存放顶点坐标、顶点法向和纹理坐标的索引(AttribIndex是在ObjModel.js中定义的，下面是ObjModel.js中的定义代码)：



* 1. 在shader中获取相关shader变量的索引后，在main函数中对attribIndex进行初始化：



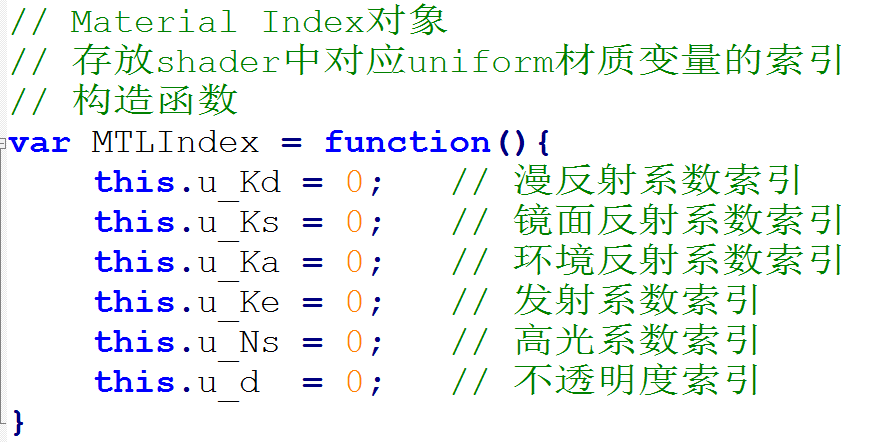
如果有的变量shader中没有，则调用init时传-1，例如“无光照无贴图”示例中是这样调用的：



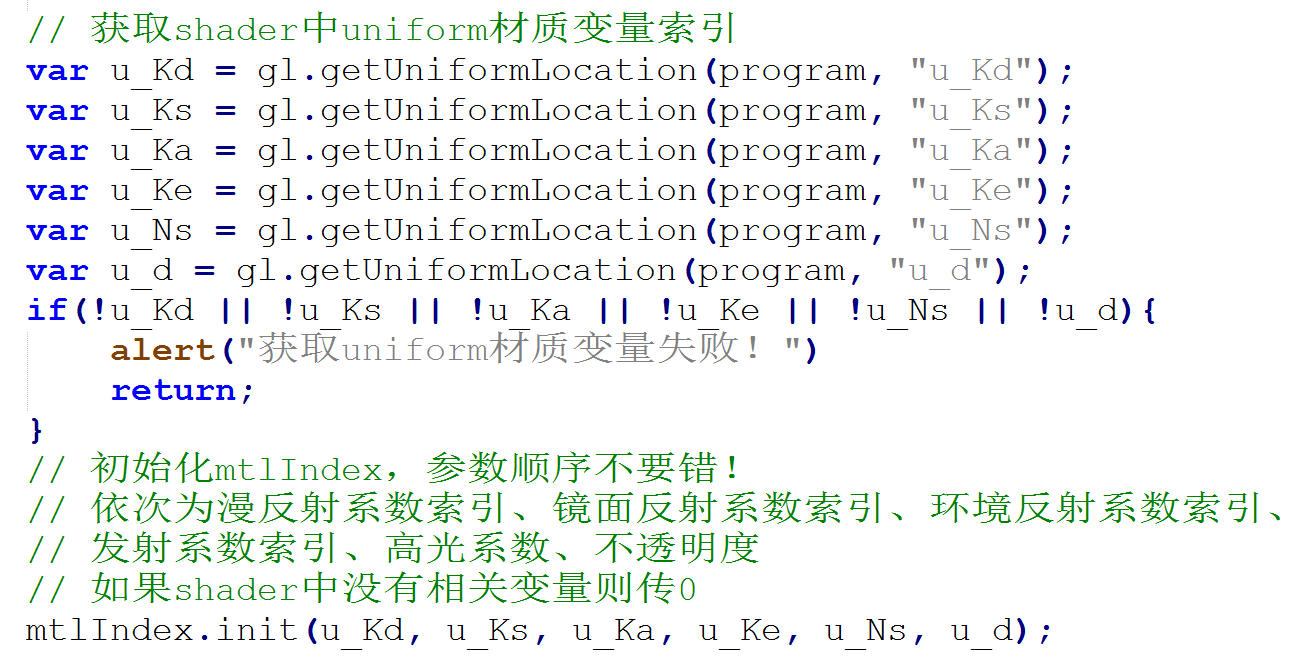
* 1. 如果有用到光照计算，则同样定义存放相关uniform shader变量索引的对象：



MTLIndex同样是在ObjModel.js中定义的：



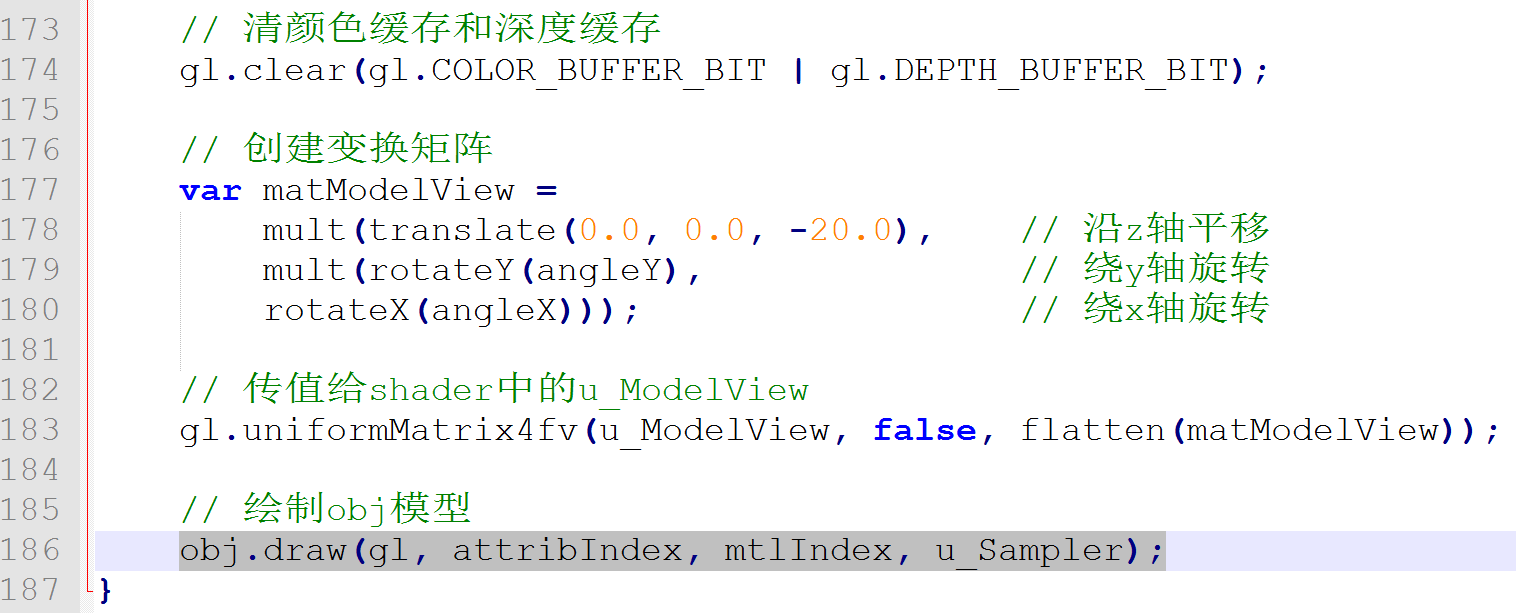
* 1. 同样在main函数中获取相关变量索引，并对mtlIndex进行初始化：



* 1. 在绘制函数render开始，检查obj对象是否一切就绪（含义是相关文件信息是否已全部读取完毕，且为绘制做的准备工作也做完了，例如buffer的初始化、纹理对象的初始化等等），如果读取完毕，则进一步处理，如果没有一切就绪，则请求重绘并返回，不进行后续绘制工作。(如果场景中包含很多对象，也可以先只绘制其他对象，而obj对象在读取完毕后再绘制)



* 1. 调用obj对象的draw函数进行绘制(注意draw之前的代码和obj导入无关，根据自己的需求编写，通常需要给shader中相关矩阵变量传值)：



draw函数也是在ObjModel.js中定义的，最后一个参数为shader中纹理采样器变量的索引，如果没有用光照计算，则第3个参数给null，如果没有用纹理贴图，则最后一个参数给0。在绘制obj模型时，自己不用管纹理相关变量的传值，也不用管材质变量和顶点属性变量的传值，全部封装好了，最主要的工作就是给矩阵传值，用来对模型进行实例化以及设置视角，注意下载的模型有的会很大或很小，需要缩放后才能看到。

1. 以上即是导入Obj模型的步骤，因为做了很多的封装，使用起来比较方便，感兴趣的同学可以看ObjModel.js来了解内部原理，此部分不要求掌握。
2. 下载的模型有的很大（一个几十M甚至上百M），这种模型不建议在实践作品中使用，因为在导入时需要太久。
3. 当导入的Obj模型比较多或是文件比较大时，可以尝试在导入Obj模型时绘制一个进度条，给大家一个思路：
   1. 在render中未一切就绪时，绘制进度条（两个矩形而已，一个不变，一个另外一种填充颜色的长度不断变化，另外使用不同的视域体(ortho2D即可)）
   2. 进度可根据两个方面：一是加载的Obj文件数量（根据数量将进度条划分成n等份），二是在调用obj对象的isAllReady时查询已解析的文本行数及完成读取的文件数量等信息，返回一个当前Obj相关文件的加载进度(此部分需要修改ObjModel.js了)