1. Steps of building: copy exist file tutorial\_06, move tutorial\_07 in; modify CMakeLists.txt; configure, generate; glClearColor函数是一个状态设置函数，而glClear函数则是一个状态应用的函数; (2018-3-11)
2. Tutorial 1: GLEW, OpenGL Extension Wrangler Library; GLFW, Graphics Library Framework, GLFW是一个专门针对OpenGL的C语言库，它提供了一些渲染物体所需的最低限度的接口; GLM, OpenCL mathematics; glfwCreateWindow; change do-while to while, done; move glfwPollEvents to loop start; (2018-3-11)
3. Tutorial 2: 顶点数组对象(Vertex Array Object, VAO)可以像顶点缓冲对象那样被绑定，任何随后的顶点属性调用都会储存在这个VAO中; GLuint vertexbuffer; 顶点缓冲对象(Vertex Buffer Objects, VBO), 它会在GPU内存(通常被称为显存)中储存大量顶点。使用这些缓冲对象的好处是我们可以一次性的发送一大批数据到显卡上，而不是每个顶点发送一次。从CPU把数据发送到显卡相对较慢，所以只要可能我们都要尝试尽量一次性发送尽可能多的数据; glVertexAttribPointer函数告诉OpenGL该如何解析顶点数据（应用到逐个顶点属性上）了; change do-while to while; change sequence of function; move functions out; OpenGL主要处理三角形, same as blender; (2018-3-11)
4. OpenGL CN: <https://learnopengl-cn.github.io/01%20Getting%20started/01%20OpenGL/>
5. 将上下文对象设回默认, glBindObject(GL\_WINDOW\_TARGET, 0); (2018-3-12)
6. 我们首先创建一个对象，然后用一个id保存它的引用（实际数据被储存在后台）。然后我们将对象绑定至上下文的目标位置（例子中窗口对象目标的位置被定义成GL\_WINDOW\_TARGET）。接下来我们设置窗口的选项。最后我们将目标位置的对象id设回0，解绑这个对象。(2018-3-12)
7. OpenGL的一大特性就是对扩展(Extension)的支持, this is GLEW; download GLEW, (2018-3-12)
8. Download GLFW; 从源代码编译库可以保证生成的库是兼容你的操作系统和CPU的，而预编译的二进制文件可能会出现兼容问题（甚至有时候没提供支持你系统的文件）; configure; include directories, library directories; find opengl32.lib; 然后编译的库glfw3.lib（注意我们用的是第3版）就会出现在src/Debug文件夹内;
9. Try GLAD, done; (2018-3-15)
10. --------------------------------------------------------------------------------------------------------------------------
11. 共同点: glfwTerminate();
12. 你好, 窗口; (2018-3-11)
13. glfwTerminate(): 当游戏循环结束后我们需要正确释放/删除之前的分配的所有资源。我们可以在main函数的最后调用glfwTerminate函数来释放GLFW分配的内存。(2018-3-11)
14. glfwPollEvents: 检查有没有触发什么事件（比如键盘输入、鼠标移动等），然后调用对应的回调函数（可以通过回调方法手动设置）。我们一般在游戏循环的开始调用事件处理函数。(2018-3-11)
15. While: 我们可不希望只绘制一个图像之后我们的应用程序就立即退出并关闭窗口。我们希望程序在我们明确地关闭它之前不断绘制图像并能够接受用户输入。(2018-3-11)
16. glfwSwapBuffers: 前缓冲保存着最终输出的图像，它会在屏幕上显示；而所有的的渲染指令都会在后缓冲上绘制。当所有的渲染指令执行完毕后，我们交换(Swap)前缓冲和后缓冲，这样图像就立即呈显出来，之前提到的不真实感就消除了。(2018-3-11)
17. 渲染（英语：render，或称“绘制”）在电脑绘图中，是指：用软件从模型生成图像的过程。模型是用语言或者数据结构进行严格定义的三维物体或虚拟场景的描述，它包括几何、视点、纹理、照明和阴影等信息。(2018-3-11)
18. 你好, 三角形; (2018-3-11)
19. 图形渲染管线可以被划分为两个主要部分：第一部分把你的3D坐标转换为2D坐标，第二部分是把2D坐标转变为实际的有颜色的像素。(2018-3-11)
20. 当今大多数显卡都有成千上万的小处理核心，它们在GPU上为每一个（渲染管线）阶段运行各自的小程序，从而在图形渲染管线中快速处理你的数据。这些小程序叫做着色器(Shader)。(2018-3-11)
21. -------------------------------------------------------------------------------------------------------------------
22. User-define shader: 对于大多数场合，我们只需要配置顶点和片段着色器就行了。几何着色器是可选的，通常使用它默认的着色器就行了; (2018-3-11)
23. Vertex shader: 顶点着色器主要的目的是把3D坐标转为另一种3D坐标（后面会解释），同时顶点着色器允许我们对顶点属性进行一些基本处理。(2018-3-11)
24. Fragment Shader: 片段着色器的主要目的是计算一个像素的最终颜色，这也是所有OpenGL高级效果产生的地方。通常，片段着色器包含3D场景的数据（比如光照、阴影、光的颜色等等），这些数据可以被用来计算最终像素的颜色。(2018-3-11)
25. OpenGL仅当3D坐标在3个轴（x、y和z）上都为-1.0到1.0的范围内时才处理它。(2018-3-11)
26. 三角形的位置数据不会改变，每次渲染调用时都保持原样，所以它的使用类型最好是GL\_STATIC\_DRAW。如果，比如说一个缓冲中的数据将频繁被改变，那么使用的类型就是GL\_DYNAMIC\_DRAW或GL\_STREAM\_DRAW，这样就能确保显卡把数据放在能够高速写入的内存部分。(2018-3-11)
27. OpenGL或GLSL中定义一个颜色的时候，我们把颜色每个分量的强度设置在0.0到1.0之间。比如说我们设置红为1.0f，绿为1.0f，我们会得到两个颜色的混合色，即黄色。(2018-3-11)
28. Check whether we can run without glad.c, we cannot, we must have glad.c; glad.c in listed outside all files; (2018-3-15)
29. Download VSCode in windows, install; download VSCode in ubuntu, done; (2018-3-15)
30. Download VPN gate, install; download CMake, install; error occurred when making answers; (2018-3-15)
31. ----------------------------------------------------------------------------------------------------------------
32. Check whether we can delete \n or not, no we cannot; (2018-3-15)
33. Install VS 2015, done; install C++; (2018-3-12)
34. Sign in microsoft account; (2018-3-11)
35. Shader filename extension can be whatever you want; (2018-3-15)
36. Try to delete shader\_s.h, not allowed; try to search shader\_s.h, find it; (2018-3-15)
37. Comment on shaders\_class.cpp; (2018-3-15)
38. separate shader in drawtriangle1, done; separate shader in drawtriangle2, done; separate shader in drawtriangle3, done; (2018-3-15)
39. Separate shader0, done; separate shader2, done; separate shaderex1, done; separate shaderex2, done; Try to separate shader from main program, done; (2018-3-16)
40. Try to use other key, use I and K, done; try to change with time, done; (2018-3-16)
41. -------------------------------------------------------------------------------------------------------------------
42. Try to run the answer of textures; done; (2018-3-17)
43. Check include, done; (2018-3-17)
44. Try to run transformation1, done; (2018-3-19)
45. Set light color via uniform, give up; (2018-3-20)
46. Download assimp3.3.1, done; (2018-3-20)
47. Cmake, could not find any instance of Visual Studio; try whether VS2017 can solve the problem or not; install VS2017, done; successfully build assimp; (2018-3-20)
48. Try to use VS2017 to run openGL, 不過我是在glad.c按右鍵點屬性，取消使用先行編輯的標頭檔; (2018-3-20)
49. Classify program according to their chapter, done; (2018-3-20)
50. Use vs2017 to implement previous work, done; (2018-3-21)
51. ------------------------------------------------------------------------------------------------------------------------
52. Try to run mirror light, done; (2018-3-21)
53. Add assimp.lib to library, done; (2018-3-21)
54. Try to run load model; add assimp to include, done; successful; (2018-3-21)
55. Unable to resolve function overload, add header, done; (2018-3-21)
56. 02/02
57. 在观察空间（而不是世界空间）中计算冯氏光照：参考解答。
58. 尝试实现一个Gouraud着色（而不是冯氏着色）。如果你做对了话，立方体的光照应该会看起来有些奇怪，尝试推理为什么它会看起来这么奇怪：
59. -