# Programming Assignment 4

Due: Wednesday, Nov 30

In this assignment you will learn to use the OpenGL API and also prepare for learning shader programming in the future.

You will be working in groups of at least 2 in this assignment. Maximum of **four** are allowed in a group. You are not allowed to work alone. If you prefer, you could keep the same group for the previous assignment.

C or C++ is the preferred language when using OpenGL or shader programming. I provide a zip file with one demonstration program. Create a project with Microsoft Visual C++, compile it and run it. For those who do not know how to compile and run the program, here are the instructions. However, most of you probably would be able to do this easily.

## Compile and Run

1. Download the source code.

2. Create a new project and import all source files in it. The approach might be IDE specific. In some IDE, you might only need to drag the files into the project. Please refer to online materials if needed.

3. Download the GLUT OpenGL library

For Mac users:

1. GLUT should have already been installed in Mac OS X, so you do not need to download anything. Yet, you might need to change some project setting to correctly include the glut library. (some IDE specified online instructions might be useful)
2. For example, in the Xcode IDE, you only need to change the "#include <GL/glut.h>" to "#include <GLUT/glut.h>", the "not found" error should be removed.

For Windows users:

1. For Windows users, download GLUT from http://osdn.net/projects/sfnet_colladaloader/downloads/colladaloader/colladaloader%201.1/glut-3.7.6-bin.zip/ and extract it to your local space.
2. Copy glut32.dll to C:\Windows\system32. Copy glut32.dll to C:\Windows\SysWOW64 if system32 does not exist. (This may happen for a 64-bit machine).
3. Copy glut32.lib to C:\ProgramFIles(x86)\Microsoft Visual Studio 11.0\VC\lib. (Assuming you are using VS 2012 and installed it in the default path)
4. Copy glut.h to C:\ProgramFIles(x86)\Microsoft Visual Studio 11.0\VC\include. Note: a few online tutorials may suggest you to copy it to …\VC\include\GL, which is not available by default for VS 2012.

For other IDE users, please search your IDE name along with glut for online help.

5. Once you have done this configuration, syntax errors you see from the previous part should have gone. Build and run your program, a window with a cube rendered in 3D should pop-up. Try possible interactions yourself.

6. Before you move on to the real assignment, it would be good to get familiar with the source files somehow. To do so, you might need to check some online OpenGL materials for reference.

## The Real Assignment

**Part A:** Goal: Write an animation using OpenGL

Write a program to display two cubes - Spinner and Revolver, that satisfy the following constraints:

a) Spinner should rotate about an axis through its center, parallel to any edge.
b) Revolver rotates about the same axis as Spinner revolving around Spinner.
c) Revolver DOES NOT rotate about an axis through its own center.
d) The frequency of Spinner rotating about its own axis should be the same as that of the rotating Revolver. This means, for every complete revolution of Revolver around Spinner, Spinner should itself have completed 360 degrees.
e) Spinner should be larger than Revolver.
f) Spinner and Revolver should NOT be of the same color.

Your program should support interactive viewpoint manipulation, i.e., the user should be able to use the mouse select viewpoint position in space. - Use glLookAt for this.

*Hints that MIGHT be useful*

1. Use glTranslate, glRotate, glScale - Look them up in the OpenGL reference
2. OpenGL uses a local co-ordinate system, and POST-MULTIPLIES successive transformations.
3. Think about how the mouse motion should affect the image. Look up glLookAt.
4. glutIdleFunc() can be used to trigger a refresh of the image every short period of time, and also update the animation. Read more in the URL: https://www.opengl.org/resources/libraries/glut/spec3/node63.html

**Part B:** The openGL calls that you used in Part A helps to generate the matrix for each call. However, in current version of OpenGL, all programming is done using shader programming. We did not do it in this course since it is too difficult to cover in just three weeks. However, now that you know OpenGL, it will be much easier to learn shader programming in your spare time or in a subsequent course. This part of the assignment is geared to make that process even easier.

In shader programming, you would not get the calls like glTranslate, glScale, glLookAt anymore. But, you have to code up generation of these matrices yourself. In this part of the assignment, you will write a library of functions *my_GL_X* (X can be translate, scale or lookAT etc) for all such calls you have used in Part A. This is exactly what you have to do if you are going to write a viewer in shader programming.

The advantage of writing your *my_GL* library in the setting of this assignment is that now you can verify it. Use your *my_GL* library calls in this part of the assignment and your output should be identical to what you had in Part A. This will guarantee that you have got the matrices set up right in your library.

**Part C:** Goal: Light your object
Use OpenGL lighting and material to light your object. You should be able add lights or change the position of lights easily. You should be able to create all of ambient, diffused and specular lighting. You should be able to demonstrate your animation with light ON or OFF.

In shader programming you will need to write your own illumination and triangle shading code. The ideal assignment would have been to get you to write the *my_GL* library for the illumination as well. However, this seems to be too much given the time constraints. I am hoping that by doing this part, at least you will get a feel of how lighting works and can pick it up in a subsequent course to learn how to do it in shader programming.

**Deliverables**

The TA would have an office hour (time to be fixed later on based on how the assignment goes) where each group will come and show the TA their working rendering for each of Part A, B and C.