Computer Graphics Fall, 2014

TP 1

Submission date: 26 September, 2359h. *3 October*

Problem. The main goal of this assignment is for you to become familiar with basic OpenGL. You will have to do the following things:

1. Learn how the basic OpenGL/GLUT/Shaders setup works. Download a basic OpenGL program, understand and compile it. See the use of the library `GLM`.
2. Learn how to draw using object buffers. You will write functions to read a .obj mesh file into a simple float data array, the indices into an index array, and then draw them using object buffers.
3. Then write a function to compute the normals of each vertex (as discussed in class), and using glBegin/glEnd, draw the normals of the model.
4. Compute the silhouette of the mesh in the shaders using the normals.

I have already created empty functions where you will have to write the code. All code will be written either in main.cpp or myobject3d.h file. Places where you have to write code are indicated by text ADD CODE.

1. Running OpenGL

Download startingcode.zip from the course webpage. This contains the VSC++ project for a very basic OpenGL example. Compile and run it. We covered all the code present here in class, but you should go over the entire code, and make sure you understand it. You will build on this starting code for the rest of the course!

2. Drawing a Mesh

There is a mostly empty class **myObject3D** meant to store data and functions for 3D mesh objects. You will have to add all of the following functions to this class:

- `readMesh`. Read a mesh from a obj file into vertices, indices array.
- `createObjectBuffers`. Generate buffer IDs, give data location to object buffers for both vertex and index arrays.
- `displayObject`. Draw the object in the object buffers created earlier.
- `computeNormals`. Write a function to compute the normals of each vertex. Store these normals in a GLfloat array in the same way the vertex data is stored. Write functions to draw these normals on the object. You will need to pass the normal matrix to the shaders; it is computed as: `transpose(inverse(view_matrix))`.

3. Silhouette computation in the Shaders

Write code to draw a silhouette of the mesh on the screen. Do not change the main or vertex-shader. You should only make changes to the fragment shader to implement this feature. The key 's' should toggle the silhouette. Pass the view matrix, as well as the normal matrix to the fragment shader. Then recall that a fragment p lies on the silhouette if the `dot product` of normal at p with the direction from the camera to p is small (say less than 0.1).