Grading Structure of the “Second” Assignment (Modeling)

Open World RPG Game Concept

You will be given several tasks(quests) every week. They are divided into two main categories, “***main***” and “***optional***”. The main type is what you need to proceed onto the (some of the) tasks of the following weeks, and optional type is some “side quests”.

Each task has a maximum mark. If you try it, you will be given a mark ranging from 0 to that maximum mark.

Final Boss

At Week 12, we will announce several BIG tasks options, and you can choose one (and only one) to do. Of course, this task will have more marks comparing to previous tasks.

Final Grading

We will gather and grade your submissions on IVLE, but we will arrange a time to meet you personally during the reading week also. To discuss and understand what you have done.

Lab 1 Picking up Trist Structure

We have introduced the Trist structure for mesh in our lecture. We will build the structure one step at a time. In this session, we will introduce a subset of the functionalities without the `fnext` operation first.

OBJ Wavefront Files

For a starter, we will create a mesh by reading in OBJ files. OBJ files are first created by Wavefront technology and adopted by many other 3D software as a file standard. The file type stores a lot of information such as geometry and texture, etc. However, we will only use the vertex and triangle information for this assignment.

Vertices are stored with a line beginning with the character "v" and followed by the three numbers for its x,y and z coordinates. (But "`vt`" and "vn" are storing other information, such as vertex normal vectors.) Basically, each vertex is given an index according to its order of appearance in the file, starting with 1 (instead of 0). So the indices of the vertices will be ranging from 1 to n, i.e. the number of vertices.

Faces are stored with a line beginning with the character "`f`" and followed by some integers. The simplest form is an "`f`" followed by three integers, in which, representing the three vertices by the vertex indices. OBJ files allow a face with more than 3 vertices, but we will use those files with *triangles only* in this assignment.

So, these are all we need to know for now. In this assignment, we will assume the .obj files are manifolds consist of triangles. If you would like to understand more about OBJ files, you can google "`obj wavefront file`" or visit some website such as http://www.fileformat.info/format/wavefrontobj/egff.htm.

Skeleton Code

No matter you are using Windows or Mac, there are three c++ files for you to code, main.cpp and mesh.{h,cpp}. You can download the project files, or only the .{h,cpp} files and create your own project.

The `main.cpp` is basically Lab 3 in CS3241. It provides you with some basic visualization for some 3D objects and allows some keyboard I/O. Basically, you should not need to change any code in it, except if you need to add more testing code or new functionalities of your own.

The mesh.{h,cpp} are the files you will use for this half of the semester. Let's see what is the header file first. For simplicity and efficiency, we put all the code for an **OBJ file** and the **Trist** structure together. Namely, we did not separate the code or write two classes for them.

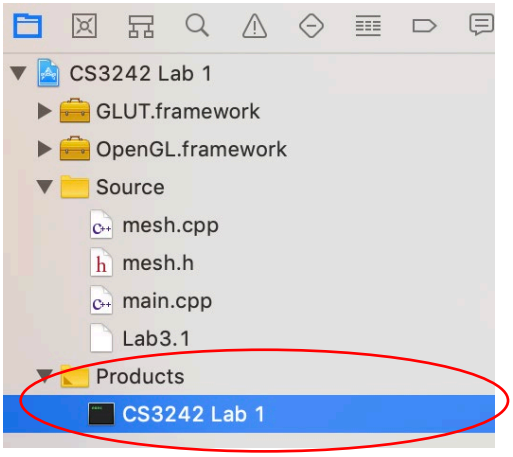
```
class myObjType {
    int vcount = 0;
    int tcount = 0;
    double vlist[MAXV][3];   // vertices list
    int tlist[MAXT][3];      // triangle list
    int fnlist[MAXT][3];     // fnext list for future (not this assignment)
    double nlist[MAXT][3];   // storing triangle normals
    double lmax[3];          // the maximum coordinates of x,y,z
    double lmin[3];          // the minimum coordinates of x,y,z

    int statMinAngle[18]; // each bucket is  degrees has a 10 degree range from 0 to 180 degree
    int statMaxAngle[18];
public:
    myObjType() { vcount = 0; tcount = 0; };
    void readFile(char* filename);  // assumming file contains a manifold
    void writeFile(char* filename);
    void draw();
    void computeStat();
};
```

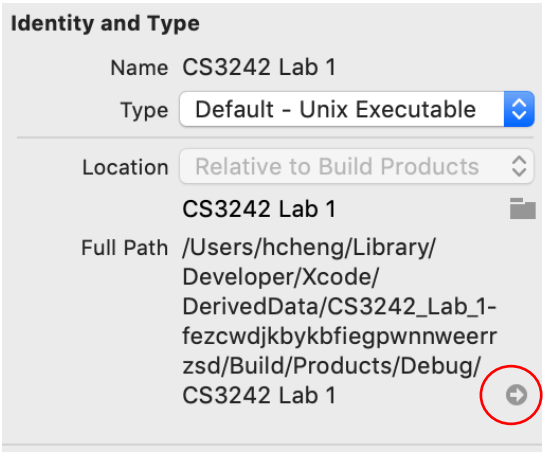
Member functions

The member function `readfile()` will read in an .obj file and store all the vertices into `vlist` and the vertex indices of a triangle into `flist`. And the numbers of vertices and triangles are stored in the variables `vcount` and `tcount` respectively. In the process of reading, `lmin[3]` will store the minimums of x,y,z values of all the vertices, and `lmax[3]` for the maximums. Following the conventions for the vertices, the indices for the triangles will be ranging from 1 to `tcount`, in which, `flist[0]` is not used.

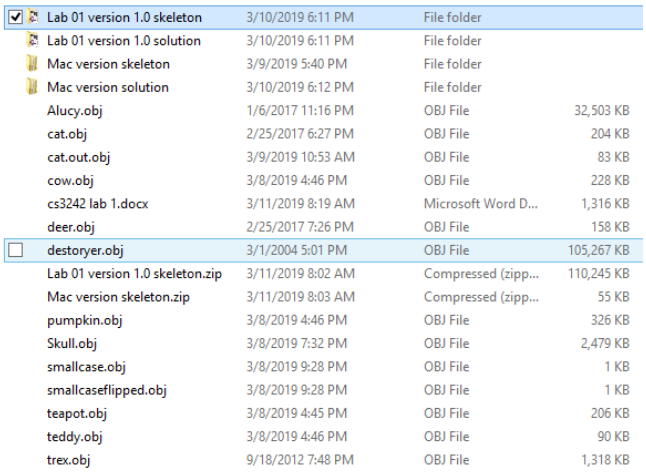
Note that main.cpp will ask you to input a filename in the console. Some may have trouble to locate the files because you do not know which directory is your executable running in.

For **Mac** users, you can click the "Products" folder in your XCode project. And on the right, you will find the "Full Path". After clicking the little right grey arrow on the right, the location/folder of the executable will be opened. And you can copy your .obj file to this folder to be opened by your program.





For **Windows** users, you may copy the file into your executable folder like Mac users. However, I found it not so ideal because those .obj files should not be a part of the code/project. My suggestion is to place the .obj files at the same folder with your project file. So if you want to open a file, e.g. `cat.obj`, you can type "`../../cat.obj`" to open it.



Lab 01 version 1.0 skeleton	3/10/2019 6:11 PM	File folder	
Lab 01 version 1.0 solution	3/10/2019 6:11 PM	File folder	
Mac version skeleton	3/9/2019 5:40 PM	File folder	
Mac version solution	3/10/2019 6:12 PM	File folder	
Alucy.obj	1/6/2017 11:16 PM	OBJ File	32,503 KB
cat.obj	2/25/2017 6:27 PM	OBJ File	204 KB
cat.out.obj	3/9/2019 10:53 AM	OBJ File	83 KB
cow.obj	3/8/2019 4:46 PM	OBJ File	228 KB
cs3242 lab 1.docx	3/11/2019 8:19 AM	Microsoft Word D...	1,316 KB
deer.obj	2/25/2017 7:26 PM	OBJ File	158 KB
destoryer.obj	3/1/2004 5:01 PM	OBJ File	105,267 KB
Lab 01 version 1.0 skeleton.zip	3/11/2019 8:02 AM	Compressed (zipp...	110,245 KB
Mac version skeleton.zip	3/11/2019 8:03 AM	Compressed (zipp...	55 KB
pumpkin.obj	3/8/2019 4:46 PM	OBJ File	326 KB
Skull.obj	3/8/2019 7:32 PM	OBJ File	2,479 KB
smallcase.obj	3/8/2019 9:28 PM	OBJ File	1 KB
smallcaseflipped.obj	3/8/2019 9:28 PM	OBJ File	1 KB
teapot.obj	3/8/2019 4:45 PM	OBJ File	206 KB
teddy.obj	3/8/2019 4:46 PM	OBJ File	90 KB
trex.obj	9/18/2012 7:48 PM	OBJ File	1,318 KB

At the end of `readfile()`, the function `computeStat()` will be called. The function `computeStat()` will compute some statistics of the mesh, such as angles of the triangles, but it is empty now because you are supposed to implement them as one of the tasks.

The function `draw()` will draw all the stored triangles by OpenGL. However, it will only draw the mesh without shading effect. If you press “W”, you will see the wireframe of your file. However, every lighting environment is already set up for you in the code, except the normal vectors of the triangles of the mesh. You will compute them as one of the tasks, as well as the function `writefile()`.

Quests (Tasks)

Here is a list of tasks that you can do. Some are the "main" quests and some "side" quests. And they should be pretty easy too… at least for now.

Computing Normal Vectors (main, 20 marks)

Compute all normal vectors of each triangle and store them into `nlist` with the same index. Uncomment the code "`glNormal3dv`" in the `draw()` function so that your mesh will be drawn with correct shading. The normal computations should be done in the function `readfile()`.

Compute Angle Statistics (main, 30 marks)

For each triangle, it has a maximum angle and a minimum angle. This task is to record the frequencies of them into the two arrays `statMaxAngle` and `statMinAngle`. Each of the array has 18 buckets for angles from 0 to 180 degrees, i.e. each bucket has a 10-degree range. For example, if there are three triangles with minimum angles 9, 3 and 55, then `statMinAngle[0] = 2` and `statMinAngle[5] = 1`. The statistic will be printed at the end of `computeStat()`.

Write an OBJ file (main, 20 marks)

Just write the current mesh into another obj file. If you press "O", the file main.cpp will call your `writefile()`. You only have to produce the "`v`" and "`f`" lines. You should be able to read back your file with `readfile()`.

Read Some Other Type of Files Other Than OBJ (optional, 20 marks)

There are other files such as STL, 3DS, etc. Some of them are also quite easy to be read also. Implement some new member functions to read these files. We suggest you to add a new command in the keyboard function in `main.cpp` to call your new reading function(s) to test out this new reading functionality.

What's Next?

Next week, we will implement `fnext`. For those who are adventurous and and want to try it next week, we want you to know that we will implement a ***simplified*** version of the Trist. Namely, we will assume the mesh is a *manifold*. This means that, each edge will be shared by at most two triangles instead of the "book page" triangles in the lecture notes. So, we do not need 6 extra entries for `fnext`, 3 will be sufficient instead.