SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.017 Graphics and Visualization

Assignment 1 – OpenGL Mesh Viewer

Handout date: 2020.05.27

Submission deadline: 2020.06.03, 11:59 pm

Late submissions are not accepted

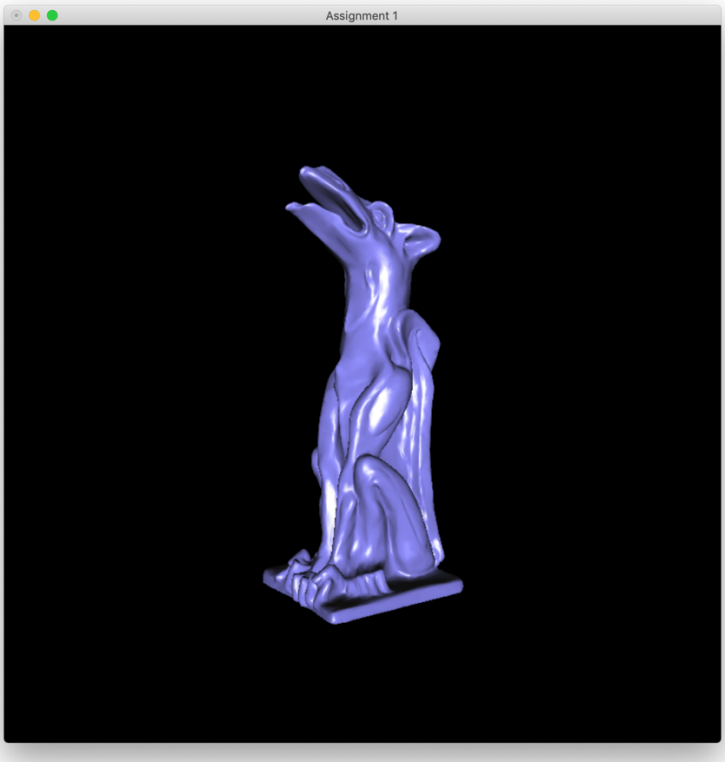

Figure 1. Expected program output by loading the garg.obj model.

In this assignment, you will load, render, and interact with a 3D mesh model saved as an OBJ file. The framework code for this assignment extends the one from last week. “TODO” comments have been inserted in main.cpp to indicate where you need to add your implementations.

Overview

This assignment consists of 4 parts:

1. Mesh Loading: Load a mesh model saved as an OBJ file

2. Mesh Display: Render the mesh model with OpenGL
3. Mesh Coloring: Change color of the mesh model
4. Mesh Transformation: Rotate and scale the mesh model

Three test datasets (garg.obj, mickey.obj, and sphere.obj) are provided in the code framework (data folder). Make sure that you're able to load, view, and interact with the three provided files without crashing; these are the only three files we'll test your program on.

Mesh Loading

In the code framework, we have provided several 3D meshes in OBJ format. It is your job to write the code to load and display these files. OBJ files are a fairly standard format that can describe all sorts of shapes, and you'll be handling a subset of their functionality.

Let's look at sphere.obj. It's a big file, but it can be summarized as follows:

```
#This file uses ...
...
v 0.148778 -0.987688 -0.048341
v 0.126558 -0.987688 -0.091950
...
vn 0.252280 -0.951063 -0.178420
vn 0.295068 -0.951063 -0.091728
...
f 22/23/1  21/22/2  2/2/3
f 1/1/4  2/2/3  21/22/2
...
```

Each line of this file starts with a token followed by some arguments. The lines that start with v define vertices, the lines that start with vn define normals, and the lines that start with f define faces. There are other types of lines, and your code should ignore these.

Your first task is to read in all of the vertices ("v") into an array (vecv) (or any other data structure that allows you to quickly reference the ith element). Then, do the same for the normals ("vn"), loading them into another array (vecn).

Understanding the faces ("f") is a little more difficult. Each face is defined using nine numbers in the following format: *a/b/c d/e/f g/h/i*. This defines a face with three vertices with indices a,d, g, and respective normal c, f, and i (you can ignore b, e, and h for this assignment). The general OBJ format allows faces with an arbitrary number of vertices; you'll just have to handle triangles in this assignment. Note that you'll need another array to store the faces (perhaps vecf).

In main.cpp, vecv is defined as an STL vector of Vectof3fs. An STL vector is simply a list of arbitrary objects. In this case, it is a list of Vector3f objects.

```
vector<Vector3f> vecv;
```

To add a new entry to this array, use push back:

```
vecv.push back(Vector3f(0,0,0));
```

There are several ways to iterate over an STL vector. Here's an example of using indices:

```
for (unsigned int i=0; i < vecv.size(); i++)
{
    Vec3d &v = vecv[i];
    // do something with v[0], v[1], v[2]
}
```

Important: fill in function LoadInput() in the code framework to achieve the above task of loading OBJ models; i.e., read the data in an OBJ file and save them in the vectors (vecv, vecn, and vecf).

Mesh Display

So let's say you have the vertices and normals stored in vecv and vecn. Then you'd draw the aforementioned triangle (defined using nine numbers in the following format: *a/b/c d/e/f g/h/i*.) using the following code:

```
glBegin(GL_TRIANGLES);
glNormal3d(vecn[c-1][0], vecn[c-1][1], vecn[c-1][2]);
glVertex3d(vecv[a-1][0], vecv[a-1][1], vecv[a-1][2]);
glNormal3d(vecn[f-1][0], vecn[f-1][1], vecn[f-1][2]);
glVertex3d(vecv[d-1][0], vecv[d-1][1], vecv[d-1][2]);
glNormal3d(vecn[i-1][0], vecn[i-1][1], vecn[i-1][2]);
glVertex3d(vecv[g-1][0], vecv[g-1][1], vecv[g-1][2]);
glEnd();
```

You may be wondering why there are all those minus-ones. It's because the faces index vertices and normals from 1, and C/C++ indexes from 0. If you have this implemented, the rest is fairly straightforward: you just have to loop over all the faces to draw the complete mesh.

Important: fill in function RenderModel() to achieve the above task of rendering a loaded OBJ model.

Mesh Coloring

Add the ability to change the color of the displayed model. Your task is to fill in function void SetDiffuseColor(int colorID) such that pressing the c key will toggle through four colors saved in the array colorTable.

Mesh Transformation

Add the ability to transform the displayed model, i.e., rotating or scaling the model, by using the keyboard or mouse. Your task is to fill in function void RotateModel(double angle, Vector3f axis) to rotate the model around its center and fill in function void ScaleModel(double scale) to scale the model around its center.

You can test your code by either using keyboard or mouse, which has been provided in the code framework.

- For rotation, pressing the r key will rotate the object around the y-axis (of the world coordinate) with a constant speed. You can also use mouse to rotate the model. Left mouse drag will rotate the object around a mapped axis based on the mouse motion (this technique is called “ArcBall Rotation”).
- For scaling, pressing the s key will make the object smaller while pressing the b key will make the object bigger. You can also use mouse to scale the model. Left mouse drag + holding shift key will scale the object to make it either smaller or bigger depending on the mouse moving direction.
- You can also reset view of the model by pressing the t key.

Grading

Each part of this assignment is weighted as follows:

- Mesh loading: 30%
- Mesh display: 30%
- Mesh coloring: 10%
- Mesh transformation: 30%

Note: in this assignment, you can complete all above tasks by filling in these five functions:

```
int LoadInput()
void RenderModel()
void SetDiffuseColor(int colorID)
void RotateModel(double angle, Vector3f axis)
void ScaleModel(double scale)
```

Assessment of this assignment will be based on your code in these functions.

Submission

A .zip compressed file renamed to AssignmentN_Name_I.zip, where N is the number of the current assignment, Name is your first name, and I is the number of your student ID. It should contain only:

- The **source code** project folder (the entire thing).
- A **readme.txt file** containing a description of how you solved each part of the exercise (use the same numbers and titles) and whatever problems you encountered. If you know there are bugs in your code, please provide a list of them, and describe what do you think caused it if possible. This is very important as we can give you partial credit if you help us understand your code better.
- A couple of **screenshots** clearly showing that you can display the three mesh models and that you can interact with them (change color and views) by using the mouse and/or keyboard.

Upload your zipped assignment to e-dimension. Late submissions receive 0 points!