CAP 4730 Computer Graphics
Dr. Corey Toler-Franklin

Program #1 Mesh Manipulation
DUE: January 23rd, 11:59 pm

Overview | Details | Resources | Getting Help | Submitting

Overview

In graphics, triangle meshes are commonly used to approximate object surfaces. As presented in class and chapter 12 of your textbook, an indexed triangle set is a mesh data structure consisting of groups of triangles with shared vertices, and indices that indicate how to connect them together to produce a surface. Mesh structures enable quick access to adjacency information for applications that require mesh operations like subdivision, mesh editing and mesh compression.

You will create meshes of simple primitives and implement processing functions to alter their appearance. You will gain experience with data structures and file formats for storing meshes. Grading is based on the following criteria; (1) Source code completion and correctness 30% (2) Warm-up 10% (3) Create Mesh 20% (4) Apply Textures 10% (5) Adjust camera and lights 5% (6) Smooth surfaces 15% (7) One page report 10%.

Details

Getting Started

Extract the file proj1_mesh.zip to your local folder. The source code framework is in the *src* folder. Datasets (meshes and textures) are found in the *data* folder. Other helpful materials are in the *docs* folder. Keep the same directory structure throughout the assignment.

To build the proj1_mesh application on linux, type *make* in the *src/mainsrc* folder. This also generates a visual studio c++ solution file. Use any platform but make sure your code compiles and runs on Ubuntu Virtual Box before you submit (see *docs* for instructions). Included in the package is libst, an open source openGL wrapper, and tinyojloader (by Syoyo Fujita), an OBJ file object loader. The program is executed from the main function in the file mainsrc/main.cpp. Examine the functions in main.cc. The titles reflect their function.

Run proj1_mesh. It will load a 3-D model. Look at the function *MouseMotionCallback*, to see how to navigate the scene using the mouse. **Search the project files for *TO DO*. In the following sections, you will insert or alter code at these locations to complete the assignment.**

Warm-Up

[h]

Primitives are represented as vertex positions (3-D vectors of x, y and z coordinates), Normals, textures and other parameters. Use an indexed triangle set to create a square pyramid, and a cube centered at the origin $(0,0,0)$ by typing vertices and indices in *.obj* files (*mycube.obj*, *mypyramid.obj*). *OBJ* files are text files that list one vertex per line followed by the ordered vertex indices for connection.

```
v -1 0 -1
v  1 0 -1
v  1 0  1
v -1 0  1
f 4 3 2 1
```

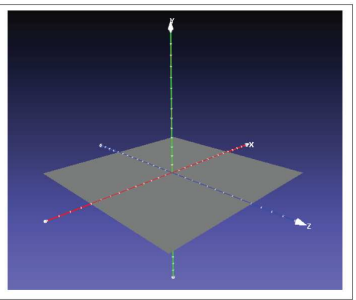

Figure 1: OBJ File Format

The cube has six square faces and 8 vertices. The pyramid has a square in the x —z plane, going from $-1-1$ on each axis and an apex at $(0,1,0)$. When specifying your vertex indices, use a counter clockwise direction with normals pointing outward. An example is shown in figure 1. Note that indices begin at 1 (rather than zero). The *.obj* file format is described in *docs/objfileformat.pdf*. For additional examples, use a text editor to open one of the *.obj* files in the *data* folder. Save a screen shot of your model (see the *KeyCallback* function to see how to do this).

Create Mesh

Next, write code to create a triangle mesh of a unit sphere centered at the origin $(0,0,0)$. Part of the work has been done for you. An icosphere is a type of geodesic dome where triangles are distributed evenly. As shown in Figure 2, the base of the icosphere is an icosahedron (a regular polyhedron with 20 equilateral triangles). Create an icosahedron (see *docs/icosahedron/*), then recursively divide it into triangles by splitting each triangle into 4 smaller triangles. Follow comments in the function *createSphere()*. Your sphere is saved as *mysphere.obj* which you will submit. You will need to add a keypress event to *KeyCallback* invoke your function. Remember to load and view your sphere to test it.

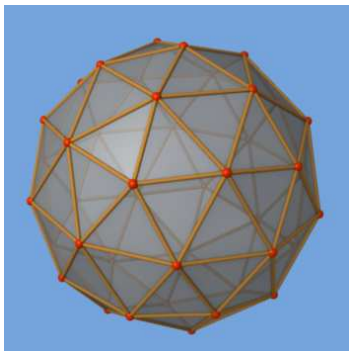

Figure 2: Unit Shpere

Mesh Processing

Add a new keypress event in *KeyCallback* to call the function gManualTriangleMesh->LoopSubdivide() (function provided). Toggle the number of subdivision surfaces of your sphere. Observe how the level of detail in the surface approximation changes. Save screen shots of a course and a smooth subdivision.

Apply Materials and Textures

Texture mapping applies 2-D images to 3-D geometry to add detail information without increasing geometric complexity. Use the textures and models in the *data* folder to create a 3-D textured model. You will use a two-stage mapping process that incorporates an intermediate shape to apply the texture. You are provided source code for the function CalculateTextureCoordinatesViaSphericalProxy() which computes the texture coordinates using a sphere intermediate object centered at the origin. The spatial coordinates of the model vertices are first mapped from Cartesian coordinates to spherical coordinates using the following equations:

$$R = \sqrt{x^2+y^2+z^2}, \qquad \phi = \tan^{-1}\left(\frac{y}{x}\right), \qquad \theta = \cos^{-1}\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right) \tag{1}$$

and finally texture coordinates are parameterized in u and v:

$$u = \frac{\phi}{2\pi}, \qquad v = \frac{\theta}{\pi} \tag{2}$$

Comments in the code explain implementation details. You will need to adjust the camera position, and lighting parameters for the best results. The next section explains how.

Adjust Camera and Lights

OpenGL uses a camera defined by an *up*, *lookat*, and *position* vector to view the scene. Search the main function for *camera*. Try to understand how the camera is manipulated to change the view. Use the mouse to position the camera and take a snap shot of the best view of your textured model. Lights make surfaces visible. You may adjust the ambient, diffuse and specular lighting parameters

(search for these). Save a screenshot of the best view of the model texture.

Report Write a one page document that includes the following:

- List any difficulties you had with the assignment.
- Include all screen shots from the assignment.
- Add any special instructions I need to run your code.
- Explain any challenges (e.g. Why your code does not run. Bugs?)

Resources

The openGL Programming Guide, available online https://www.opengl.org/sdk/ is a good reference. Appendix A also provides information about GLUT and GLEW. Chapter 12 of the textbook, Fundamentals in Computer Graphics, covers meshes. Information about the *.obj* file format is in the *docs* folder.

If you are not in the CISE department, and would like computer lab access, you can register for a CISE account at https://www.cise.ufl.edu/help/account. The source packet proj1_mesh.zip will run on machines in the computer labs on the first floor of the CSE building. Labs are open 24 hours (see https://www.cise.ufl.edu/help/access). **Remember to back-up your work regularly!!!** Use version control to store your work. DO NOT PUBLISIZE SOLUTIONS.

Getting Help

Source Code Please do not re-invent the wheel. Use the source code framework (and comments) and review the notes in the docs folder.

Discussion Group Post questions to Canvas (everyone benefits in this case), or send me an email at ctoler@cise.ufl.edu. I will check both daily.

Office Hours Stop by my office, CSE 332 (or lab CSE 319) during office hours MWF 11:45am to 12:30pm.

Collaborating Work independently. Remember to always credit outside sources you use in your code. University policies on academic integrity must be followed.

Submitting

Upload your proj1_mesh.zip fie to Canvas. It should include:

- Source code.
- Make file.
- One page report.
- Screen shots of your mesh and processing results.
- *mycube.obj*, *mypyramid.obj* and *mysphere.obj*
- Other information you would like us to know (bugs, difficulties)