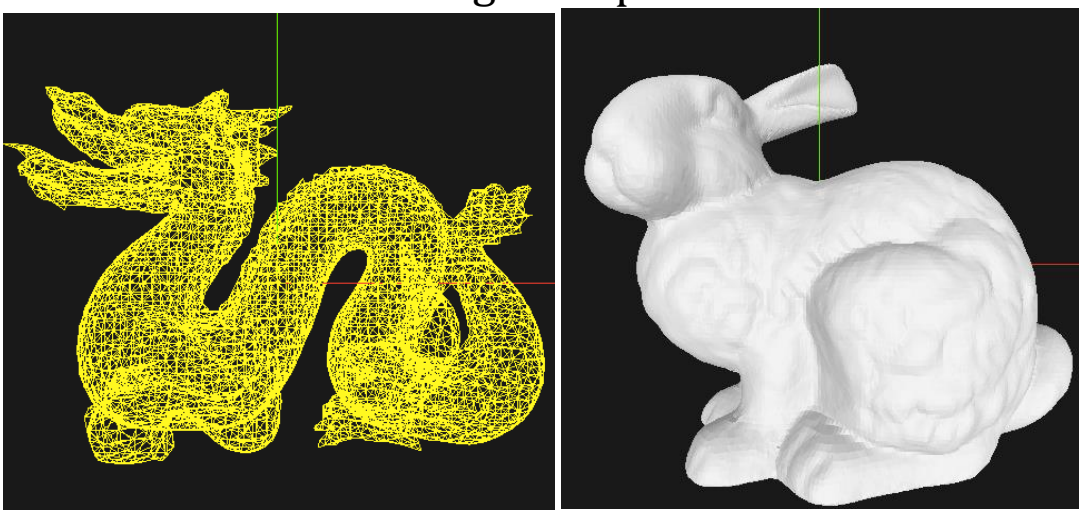
Loading a Shape File



Description:

You've learned that we can draw polygons to the screen by plotting them out one at a time, or even using mathematical functions to draw nice curves and geometric patterns. Often times to create more complex shapes, 3D objects will be scanned from the real world by lasers or modeled in 3D software. This data is then output in a 3D file format that can be read.

Your Task:

- You will be loading a file known as the PLY format. This format consists of a, header, vertex list, and a face list.
- Start by looking at the headers given to familiarize yourself with the functions provided. We have already provided the parser, and you can improve upon it on your own time. One way we will improve upon it is by implementing two functions:
  - render() – In ply.cpp
  - scaleAndCenter() – In ply.cpp
- Show the TA your running code rendering a 3d shape
  - Comment the top of your ply.cpp file with your name and a phrase the TA gives you
  - Submit on the linux servers $> provide comp175 lab01 ply.cpp

Files Given:

main.cpp – Main program interface (you do not have to modify this)
ply.h – The shape header file with data (start here)
ply.cpp – The shape class that loads the PLY model
geometry.h – Some data structures to store geometry information (you do not have to modify this).
entity.cpp and entity.h – Helper files to demonstrate inheritance

Data folder – These are the files you will be loading. Take a moment to load each in a text editor and familiarize yourself with this text-based 3D file format.

Compiling: (On the Mac)

Follow same pattern as lab 0, and be sure to include all of the .cpp files in the directory. Example posted below for Mac

```
g++ -Wall -Wextra main.cpp ply.cpp entity.cpp -
I/Library/Frameworks/GLUI.framework/Headers/ -framework OpenGL -
framework GLUT -framework GLUI -o ply
```

Running:

```
./ply
```

C++ Refresh -- Helper functions:

(for understanding the parser)

Opening a File

```
1. #include <fstream>
2. ifstream myfile ("filepath");
3. if (myfile.is_open()){
4.     string line;
       while(getline(myfile,line)){
5.         // some code
6.     }
7. }
```

Parsing a line

```
8. #include <stdio.h>
9. char* delimeter_pointer;
10. delimeter_pointer = strok(line," ")
11. while(delimeter_pointer != NULL){
12.     cout << delimeter_pointer;
13.     delimeter_pointer =
    strtok(NULL," ");
14. }
```

Example PLY File [1]:

```
ply
format ascii 1.0        { ascii/binary, format version number }
comment made by Greg Turk  { comments keyword specified, like all lines }
comment this file is a cube
element vertex 8        { define "vertex" element, 8 of them in file }
property float x        { vertex contains float "x" coordinate }
property float y        { y coordinate is also a vertex property }
property float z        { z coordinate, too }
element face 6          { there are 6 "face" elements in the file }
property list uchar int vertex_index { "vertex_indices" is a list of ints }
end_header              { delimits the end of the header }
0 0 0                   { start of vertex list }
0 0 1
0 1 1
0 1 0
1 0 0
1 0 1
1 1 1
1 1 0
4 0 1 2 3               { start of face list }
4 7 6 5 4
4 0 4 5 1
4 1 5 6 2
4 2 6 7 3
4 3 7 4 0
```

Going Further:
Did you enjoy this in class assignment? Take a look at some other 3D formats such as .md2(includes basic animation data). Look at some popular 3D modeling software like Maya, Blender3d (download for free!), and check out there file formats.
Think about other functions you can add to the ply.cpp. What interesting manipulations can you do with the file?

Some ideas:

- Highlight faces that have an area above a certain threshold
- Render surface normals from each face of the object(you will do this in A1)
- Create a noise function that adjusts the positions of vertices on the mesh.
- Add color to vertices above a certain height.
- Modify the rendering function to use a Display List

References:
[1] http://paulbourke.net/dataformats/ply/