Programming Assignment #1
CS410 : Introduction to Computer Graphics
Fall 2015

Transforming Coordinates
Due Tuesday, Sept. 22nd.

Motivation

Polygonal models exist for lots and lots of objects. To make a coherent scene, however, you need to be able to rotate, translate and scale these objects so that they are all in the right relative sizes and positions to each other, i.e. so that they are in the same global coordinate system. In this assignment you will take existing polygonal models (in PYL format) and transform them as indicated by the user.

Task

You will write a program that takes two arguments: an input file name and an output file name. The input file should be a polygonal model in ASCII PYL format. Your program should read the model file, print some information about the model to the terminal (see below), and then enter a read/eval loop in which the user may specify the following commands:

- S sx sy sz
- T tx ty tz
- R rx ry rz theta
- W

The S command is a command to scale the model; sx, sy and sz are the scale factors in the x, y and z dimensions respectively. Similarly, the T command is the translate command. (tx, ty, tz) is the translation vector. R is the rotate command, where (rx, ry, rz) is the axis to be rotated around, and theta is the angle of rotation.

The W command tells the system to write out the file and exit. The output file should be in ASCII PYL format. More specifically, the header of the output file should be the same as the header of the input file. The vertices in the output file should be transformed versions of the vertices of the input file. The faces should be the same as in the input file.

Note that the user may specify any number of S, R and T commands, before exiting with the W command.

Before entering the read/eval loop, your program should print out the number of vertices in the model, the number of faces in the model, the mean vertex and the bounding box of all the vertices. The mean vertex is just the average (x, y, z) coordinate of all the vertices in the model. (In other words, add up all the vertices [using vector addition] and then divide through by the number of vertices. The result should be a single vertex with x, y and z values.) The bounding box provides the minimum and maximum values observed for each of the three dimensions. The exact format for the mean vertex and bounding box coordinates is up to you, but it should be easily understood by the program user.

Test Data

The resources page of the class web site contains models in ASCII PYL format. The small models are simple geometric shapes (although the sphere isn't actually that small). The medium models are more complex shapes, but with fewer than 5,000 vertices each. The large model is, well, large.

Your program should be able to transform any of the small or medium models. Handling the large model is optional (but fun).

Hints

You are going to be rendering these models in the future. Be sure to build a model object class that includes not only vertices but faces, and make sure that your input is robust. Some of the models have three coordinates per vertex (x, y, z), but have additional values that can be ignored (and don't need to be written out), but should not cause input problems.

Submission/Grading

Make a tar file that includes your source files, a makefile if appropriate, and a README.txt file that explicitly tells us (1) how to compile your program and (2) how to execute it. Submit this tar file via the Checkin script on the class web site. The GTA will unpack your tar file, compile your program, and then test it on some subset of the small and medium models on the class web site. Note that your tar file should not contain executable or compiled files, just source files.

Reminder

There is no "late period". The program is due when it is due. All work you submit must be your own. You may not copy code from colleagues or the web or anywhere else. Cheating will not be tolerated, and will be handled in accordance with university and department policy.