CS1566 Assignment 3
Transformer

Algo Due:	Thu 10/3 5pm
Asgn Due:	Thu 10/9 11:59pm

1. Introduction

Having completed the Stitcher assignment, you are now intimately familiar with how three dimensional objects are created and displayed on the screen. However, you may have noticed that the last assignment overlooked something important… the program just displayed a shape in the center of the screen. And the shapes were always of the same size. And the rotation was a hack thrown in by the staff to aid you in completing the assignment correctly. Face it — you need some way of distorting and moving (transforming) your shapes so that you can actually use them for something other than decorating your screen. In this assignment you will be writing the tools necessary to move, rotate and scale your shapes. You will also take charge of the camera.

2. The Assignment

For this assignment you will write an interactive program that constructs, displays and maneuvers 3D shapes. As with your previous assignment, you will be using the code you write for most of the remaining assignments, so it will be really important that you really put your program through the paces and work out as many bugs as you can.

Expected behavior

Your OpenGL program should have the following functionality:

- At the start of the program, it displays the world axes and a "house" (see Stitcher assignment). The house should be placed in the world so that its center is at the world origin.

- When the '1' key is pressed, the house is replaced by a cube; key ‘2’ replaces the current object by a sphere, key ‘3’ replaces it by a cylinder etc (you get the picture; refer to your Stitcher assignment if in doubt). All shapes are centered at the origin.

- The local coordinate system for the current object is displayed using an RGB color scheme (similar to the global coordinate system). As the object changes (as a result of translation, etc.) the local coordinate system should change as well.

- When ‘T’/’t’ and ‘X’ are pressed together, the object physically translates along the **world** X axis, in the positive direction. Here and below, ‘physically’ means you should **not** let OpenGL handle the transformation (i.e., do not use glTranslate or glRotate; such OpenGL calls modify only the rendering of the object, not its physical location). To get any credit for this part of the assignment, you need to

modify the actual coordinates of the object, simulation-style. When ‘T’/’t’ and ‘x’ are pressed together, the object translates along the **world** X axis in the negative direction.

- When ‘T’/’t’ and ‘Y’, respectively ‘T’/’t’ and ‘y’ are pressed together, the rendered object translates along the **world** Y axis, in the positive, respectively negative direction.

- When ‘T’/’t’ and ‘Z’, respectively ‘T’/’t’ and ‘z’ are pressed together, the rendered object translates along the **world** Z axis, in the positive, respectively negative direction.

- When ‘R’/’r’ and ‘X’, respectively ‘R’/’r’ and ‘x’ are pressed together, the physical object rotates counterclockwise, respectively clockwise around the **world** X axis.

- When ‘R’/’r’ and ‘Y’, respectively ‘R’/’r’ and ‘y’ are pressed together, the physical object rotates counterclockwise, respectively clockwise around the **world** Y axis.

- When ‘R’/’r’ and ‘Z’, respectively ‘R’/’r’ and ‘z’ are pressed together, the physical object rotates counterclockwise, respectively clockwise around the **world** Z axis.

- When ‘S’/’s’ and ‘X’, respectively ‘S’/’s’ and ‘x’ are pressed together, the physical object scales up, respectively down along the **world** X axis.

- When ‘S’/’s’ and ‘Y’, respectively ‘S’/’s’ and ‘y’ are pressed together, the physical object scales up, respectively down along the **world** Y axis.

- When ‘S’/’s’ and ‘Z’, respectively ‘S’/’s’ and ‘z’ are pressed together, the physical object scales up, respectively down along the **world** Z axis.

- One left mouse click should enable the object to rotate counter-clockwise with respect to its central axis (by which we *always* mean the **object’s** (local) Y axis, e.g. in the case of the house object, the line from the tip of the pyramid to the center of the floor; in the case of the sphere, the North Pole to South Pole line etc). Note that the object could be simultaneously “spinning” around its own axis while "revolving" or translating along the world X, Y or Z-axis. A second left mouse click should disable the object from automatically rotating around its central axis.

  Notice carefully the behavior of your program for combined transformations. Once you look at these complex scenarios you can surely tell whether your transformation code is good or not. Your individual transformation and rotation may work. But when you combine them they may not work. For example if you apply translation and then rotation, the rotation should make the object rotate around the world X axis not spinning around its own axis. The grader is going to be a bit strict on this kind of cases.

You must implement the arbitrary axis rotation procedure you wrote in the algorithmic part of this assignment and use this arbitrary axis rotation to implement the spinning. You cannot use glRotate etc.

Note: Just because we ask for a spinning motion around the object main axis does not mean we will not check your solution on a different, completely arbitrary axis. To make sure your code is correct, you want to try picking an axis that goes diagonally through the house object, for example back bottom left corner through right front top corner, and the back bottom left corner as the arbitrary rotation point.

- Pressing 'O'/'o' should restore to the original setting. (Hint: you might want to just "re-make" the object as opposed to trying to undo a series of transformations)

- The right mouse button should toggle the appearance of shape normals. You will need to transform these normals with the shape.

- Pressing the 'q' key should quit out of the program.

Hint1: Do not attempt to replicate the hack rotation from Assignment 2 to accomplish rotations and translations around the main axes in this assignment. While calling glRotate may seem easy, telling OpenGL how to combine spinning and revolutions to obtain the correctly combined motion is quite difficult (involving repeated operations on the stack, and periodically saving the stack content).

To encourage you to start working early, we broke the assignment into two parts. The first part is algorithmic and is due in a week, in paper form, to the grad TA, to their mailbox or in their hand; it is worth 30% of your grade for this assignment, no late handins accepted. The second part is the actual program and is due in two weeks, via electronic submission; the coding component is worth 70% of your grade for this assignment.

Begin by reading and solving the 'Assignment 3 (part 1): Transformer Algorithm' handout (available under Assignments on the course website). When done, please move on to coding.

Download the Transformer support code package (available under Assignments on the course website). It contains a glmain.c and a glmain.h file, and a Readme.txt file. Make sure you can compile and run the support code. Read through the source code once, paying attention to comments. Concentrate on the functions labeled TODO. These are the main functions you will have to modify. You might want to start by integrating in your Stitcher code.

Note the grading sheet will include from now on points for code quality. Uncommented, hard-to-parse code will cost some points.

3. Extra Credit

EC1: Go ga-ga and use quaternions.

EC2: Any type of fun simulation.

3. Grading

Task	Points
Transformer algorithm handin	30
R/r and X/x, Y/y, Z/z rotate correctly the object without any OpenGL 'help'	15
T/t and X/x, Y/y, Z/z translate correctly the object (same as above)	5
S/s and X/x, Y/y, Z/z scale correctly the object (same)	5
'1'-'9' draws various objects	5
Local coordinate system is displayed correctly at all times	10
O/o resets the object	5
Mouse clicks enable spinning, correct behavior	15
Rotation around an arbitrary axis and point works correctly	5
README turned in and quality of the code	5
TOTAL	100

EC points are at the discretion of the grader, somewhere between 5 and 20 points are possible.

5. Support Code

• glmain.h — header file for the program
• glmain.c — main file for the program
• Readme.txt — text file containing your name, ID, description of Extra Credit work, and any additional comments.

6. Handing In

To hand in this assignment, follow the Submit link on the course website and upload the following files:
• your modified source files (glmain.c(pp) and glmain.h, any additional h/c/cpp files)
• your Makefile (if any)
• filled in Readme.txt file (please save it as plain text).

Do NOT submit any executables.
Use ftp only as an emergency backup plan (and notify the TAs immediately via email).
Enjoy!