CS1566 Assignment 4 (part 1)
Camera Modeler Algorithm

Out: Thu 10/11
Due: Tue 10/16 5:00pm

Grading:

Problem	Points
1	15
2	15
3	15
4	15
5	15
6	15
7	10
Total	100

1. Consider an orthographic (parallel) camera with parameters: *posx, posy, posz, lookAtx, lookAty, lookAtz, upx, upy, upz*. Give the pseudocode to:
[Please consult the course notes or the gluLookAt function in the OpenGL Blue Book linked from the course website http://vis.cs.pitt.edu/teaching/cs1566/resources.php to remember what the parameters mean]

1a. translate your camera by 2 units along the *Z* world axis

1b. translate the camera by 2 units along the *X* world axis

1c. point the camera to a new *lookAt*, located 2 units lower on the *Y* world axis than the original *lookAt* point

2. Given the parameters listed in problem 1, give the pseudocode for computing the following (you may use a cross-product auxiliary function if you wish):

2a. the *w* camera unit local axis.

2b. the *v* camera unit local axis.

2c. the *u* camera unit local axis.

[Hint: remember to normalize and check against division by zero, and that your 'up' and 'look' are not collinear. Remember that 'lookAt' is a point, not a vector.]

3. Given the *u, v, w* vectors in problem 2 and the camera arguments in problem 1, give the pseudocode to:

3a. move the camera to its left (aka along *u*) by *t* units. How about moving to the right? You need to move the *lookAt* point as well.

3b. move the camera higher/lower (aka along its *v*).

3c. move the camera closer/further from the *lookAt* point (aka along *w*).

4. Consider the lens parameters of the camera: *left, right, bottom, top, near, far*. Give the pseudocode to:
[Please consult the course notes or the glOrtho function in the OpenGL Blue Book linked from the course website http://vis.cs.pitt.edu/teaching/cs1566/resources.php to remember what the parameters mean]

4a. move the near clipping plane closer to/further from the camera. Make sure you don't go **past** the camera, though, or past the far clipping plane in the other direction.

4b. move the far clipping plane closer to/further from the camera. Make sure you don't go **nearer** than the near clipping plane, though.

4c. double the width and height of the view volume (effectively changing the lens).

5. For this problem, you need to write pseudocode for building the translation, rotation, and scaling matrices for an orthographic camera. Use 4×4 matrices of type GLdouble and also keep in mind that OpenGL matrices use column-major format. Given the parameters in problem 1, respectively computed in problem 2:

5a. give the pseudocode for building the camera viewing translation matrix.

5b. give the pseudocode for building the camara viewing rotation matrix.

5c. give the pseudocode for building the camera viewing scaling matrix.

6. The following is example OpenGL code for setting up an orthographic projection. Replace the gluLookAt call with your own code (to load your problem 5 matrices to the stack).

```
//assuming: glMatrixMode(GL_MODELVIEW)
glLoadIdentity();
gluLookAt(posX, posY, posZ, lookAtX, lookAtY, lookAtZ, upX, upY, upZ);
//all other ModelView transformations follow this
```

7. The following is example OpenGL code for setting up the orthographic projection. Replace the glOrtho call with your own code (to load the scaling matrix computed in problem 5 to the correct stack).

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
//if parallel
glOrtho(left, right, bottom, top, near, far);
glMatrixMode(GL_MODELVIEW);
```

EC

1. Give the pseudocode for rotating the camera around the world *X* axis.
2. Give the pseudocode for rotating the camera around its local *w* axis.
3. Give the pseudocode for spinning the camera around some other object's main axis *a*.
4. Give the pseudocode for moving the camera along a sinusoidal path along the *look* direction (thus the cameraman's head would be bobbing up and down as they walk).