CMSC427 Fall 2017
Lab 3 – Transformations (part B): Camera

Due by midnight Tuesday, Oct. 10th
Online as combined PDF plus a PDE file

Objectives of lab:

- Implement a Camera class to support camera motion
- Understand the Camera matrix and its computation

Requirements:

Take the Lab3Camera example code from the course web site, and complete the Camera so you can:

1. Move forward and backwards.
2. Slide left and right.
3. Slide up and down.
4. Do a roll, left and right.
5. Do a pitch, up and down.
6. Do a yaw, left and right.
7. Home back to the initial, default camera position
8. Optional: view from top, view from side (alternatives to the default position)

Implement keystroke bindings for each of these so you can navigate the camera through a simple wireframe scene. Make the scene out of several primitives, your choice. The scene only has to be complex enough to make motion interesting.

Before you implement the motions you need to complete the computeCameraMatrix method which is the equivalent of the **`camera`** method in Processing (or gluLookAt in some implementations of OpenGL.) Because you're re-implementing an existing routine you can test yours with the printMatrix() method – when your version of computeCameraMatrix is correct, you should see the same matrix as when you use the built-in **`camera`** method.

The calling sequence is to first compute the components of the Camera coordinate system, which become the elements of the Camera matrix, and then set the matrix in the OpenGL pipeline.

```
// Compute the camera transformation matrix
cam.computeCameraMatrix();
// Apply the camera transformation matrix
cam.setCameraMatrix();
```

Note that there are possible confusions here because, if you look up how to make the camera matrix on the internet and in textbooks, you'll find different variable names for the camera coordinate axes (in our PowerPoint, xc, yc, zc and d – elsewhere n, u, v and eye, or simply x, y, z and eye), and you'll find different versions of the matrix (either with xc, yc and zc as columns, or with them transposed as rows). Part of this lab is to navigate these notational challenges and find a solution that works.

The previous lab had this link which gives some hints towards camera motion:
https://learnopengl.com/#!Getting-started/Camera

Submission

To submit you should create a Word or other document that includes the following:

A. A header with CMSC427 fall 2017 Lab 3 and your name.
B. A short narrative on how you solved the lab problems. One page overall will be enough.
C. A copy of the PDE files (Lab3Camera.pde and Camera.pde),
D. One screen capture of your scene
E. Save the document as PDF and submit, along with a separate copy of your PDE files

Remember to put your name in every file, and properly comment your code.