CS334: Assignment #1 – Project It!
Linear Algebra + Perspective/Orthographic Projections

Out: Sept 1, 2017
Back (i.e., due): Sept 13, 2017

Now that you have a better understanding of graphics generation using OpenGL with FreeGLUT let’s proceed to a direct application of the linear algebra concepts and the camera projection mechanisms in a simple but interactive application.

Objective:
The objective is to implement a walking mechanism inside a “city” built of cubes. Using the keyboard the user can walk in four directions: forward, backward, to the left, and to the right. Similarly, the user can rotate the head for observing the landscape at different viewing angles, as is commonly done in First Person Shooter (FPS) games. Finally, the user can also select the camera projection type (i.e., orthographic or perspective) and the buildings will be animated in a simple way.

You are provided with the framework for this assignment; however invest some time in analyzing the code and its functionality in terms of 3D rendering and camera logic. YOU WILL NEED IT FOR LATER ASSIGNMENTS! The framework already contains the rendering logic for the city, a basic implementation of a walking mechanism (not quite perfect) and a default perspective projection camera. Understanding the basic concepts of linear algebra and projections is critical for the development of the assignment.

Important:
Run the provided framework in the PCs of the lab or in your own PC and check that it runs without problem. You can add your own .h and .cpp files to the project, but keep in mind not to modify the basic structure of the project (location of the GL folder and the freeglut.dll and freeglut.obj files) otherwise the integrity of the project will be compromised and may not run properly. The logic of the code can be grouped in three categories: 1) Rendering using OpenGL, 2) Events and 3) A simple UI. Check the code and the comments for additional details. More information concerning OpenGL and FreeGLUT will be given in the PSO sessions.

For walking in the city the user uses the arrow keys. The LEFT and RIGHT arrow keys are used for steps in the X axis, and UP and DOWN arrow keys are for walking forward or backwards (Z axis) – we assume Y points upwards in the scene. Similarly, rotations are implemented using the ‘w’ and ‘s’ keys for rotation about X-axis of the head and ‘a’ and ‘d’ keys for rotation about Y-axis. Z rotation is not required. Check the code for an example of this mechanism. For changing the projection mechanism, keys ‘p’ and ‘o’ should be used for selecting perspective or orthographic projections respectively. Perspective projection should be the default projection mechanism when the program is started.
Also, all buildings should be animated in some way – the easiest is to animate them all in the same fashion. For example, apply a changing-each-frame geometric transformation (e.g., combination of translation, rotation or scaling) to the cubes of the city in a reasonable way; i.e. the cubes rotate over their central pivot or they change their size periodically. The general structure of the city must remain intact and “buildings” still must be clearly identifiable.

**Note:** Using `gluLookAt` or any other `glu*` function is PROHIBITED for this assignment (except for `gluPerspective`).

Specifics (note: each item is labeled with a unique number and the percentage of the grade it accounts for)

1. (25%) Implement the walking mechanism using the arrow keys. Note that the walking direction and the viewing direction are uncoupled (i.e., you can look in some direction, but you walk in another).
2. (25%) Implement the rotation mechanism using the ‘a’, ‘d’, ‘w’ and ‘s’ keys. Note that the walking direction and the viewing direction are uncoupled (i.e., you can look in some direction, but you walk in another).
3. (25%) Implement the perspective and orthogonal projections. Perspective is activated when user presses ‘p’ and orthogonal is activated when user presses ‘o’. You choose the values for the projections (left, right, bottom, up, zNear and zFar), still they must be reasonable for good visualizations. Improve the approach given in the framework.
4. (25%) Implement a simple global building animation. You are free to choose whatever you like but the animation must be contiguous and reasonable in appearance.

**Bonus points (+10 points):**
Apply different animations for each building and have the animation react, or change, depending on the closeness of the camera viewpoint. Use your imagination!

**Code Template:**
For this assignment, the code framework is on the course website and in Blackboard.

**Handing-in Assignments:**
To submit the assignment, please use Purdue's Blackboard Learn. Your complete Visual Studio project (e.g., project file, data files, source code and precompiled executable) is due before class time on the due date. It is your responsibility to make sure the assignment is delivered on time. **Hint: don’t wait until the last moment to hand in the assignment!**
For grading, the program will be run with no command line parameters and tested against the above specifics. When appropriate, the code will be looked as well for proper functionality. If the program does not compile, *zero* points will be given.

**Copying/Plagiarism:**

As a reminder, ALL assignments (except the final assignment and upon instructor approval) are individual assignments. This means **YOU MUST DO YOU OWN PROGRAMMING.** If a copy case is found, both the provider and copier of the assignment will be given a grade of “0” for the assignment and the case will be reported to the University level office which will evaluate the case – expulsion from Purdue is a very likely outcome. Thus, please do your own work. You may consult the TAs, and your peers, for technical questions but not for copying solutions.

If you have more questions, please see myself or the TA.

Good luck!