CS 432 – Interactive Computer Graphics

Assignment 3

The goals of this assignment are

1. Render 3D objects.
2. Rotate 3D objects.
3. Implement a 3D camera interface.

Requirements

For each programming assignment you should submit a **zip file** containing the code (and MSVS/Xcode project/solution files) and a **report** stating: what you did; how you did it; any particular features you want to draw attention to; any problems with the program that you know about.

In addition:

1. Your code must be original. You may discuss approaches with your colleagues but not how to code it.
2. You must use GLSL (shaders) and vertex buffer objects (VBOs)
    a. Therefore you **may not** use glVertex*, etc..
3. Make sure you program complies and runs on either a Windows 7/8 machine with VC++, freeglut and glew or on a Mac with OSX and Xcode

What to Submit

For your submission, zip together the following.

- A README file with instructions on how to compile and run your program(s). This should also include the environment in which you are developing and running (OS version, Developing Environment version, Shader Version) and any issues/features you want to draw attention to.
- Your source files
- Your file(s) containing rules for compiling and running your program(s) (i.e. project files, make files, etc...)
- **A short screencast video that you talk over (5 minutes max)**

Part I: 3D Rendering

Take the cube sample code and augment it to draw a 3D polyhedron. This polygon should be rotating about its own y-axis one degree every 10 milliseconds. You can pause (and unpause) the animation via the spacebar. You can toggle between perspective and orthographic projections via the P/p key. You can also explore the scene using the "flying camera interface" discussed in class.

Polyhedron:

- The polyhedron must have at least eight polygonal sides. These MUST be rendering using triangles
- Each triangle should be a single flat color, except for the top and bottom of your object, if applicable.
- Use `flat` shading so that the triangles' fragment colors aren't interpolated.
- In addition to rendering the polyhedron, you must also render its wire frame in opaque black
- Vertices should be in homogenous coordinates.
- Rotation should be done in the shader. Just pass the updated matrices to shader and have it do the transformations.
- Place the object at (0,0,-2) in the world.
- Rotation should be about the object's origin.
- Your polyhedron should be wrapped in its own class, much like in HW2
- Hitting the spacebar toggle animation.

Camera:

- You should have a Camera class.
- Hitting the P/p key toggles between perspective and orthographic projection.
    - The orthographic parameters should be `left=bottom=-1, right=top=1, near=-10.0, far=10.0`
    - The perspective parameters should be `fov=65, aspect=width/height, near=1, far=100`
- **forward/backward arrow:** Move (slide camera) forward/backward (in/opposite the look-at vector **-n**)
- **right/left arrow:** Move (slide camera) left/right (in/opposite to the **u** vector)
- **Z/z:** Roll counterclockwise/clockwise in the xy-coordinate plane.
- **X/x:** Pitch up/down
- **C/c:** Yaw counterclockwsise/clockwise in the zx-coordinate plane.

Extra Credit:

- Draw using indices/elements

Grading:

- Draw 3D Polyhedron (20pts)
- Color correct (10pts)
- Wireframe correct (10pts)
- Animation correct (10pts)
- Toggle projection (10pts)
- Classes (10pts)
- Flying Camera Interface (30pts)
- Extra Credit (5pts)