CMSC427 Fall 2017
Project 1: Object viewer

Due: Nov. 16th at midnight
Submit online as combined PDF plus source files

Objectives of lab:

- Create Shape class hierarchy
- Create a virtual trackball for rotating the object
- Work with lighting
- Establish a working object viewer to investigate shaders

Requirements:

This assignment has multiple, relatively independent parts, so you can complete each part separately (and get partial credit for partial completion.)

The initial code will be in a folder Project1 exported from Eclipse with five source files: App.java, Code.java, Controller.java, ImportedModel.java and Mesh.java. There's also two shaders and a textured object (shuttle.obj and spstob_1.jpg). The latter two files should reside in the bin subdirectory of the project.

The initial code for this example is best called MV-C, or ModelView-Controller architecture, since the model and viewer are combined, and the controller is separate. In full Model-View-Controller all three are separate. The Controller.java file supports mouse, mouseMotion and keyboard events, with only one used now – the rest are there for you to use.

To do this project, first get it running. You should see a version of the space shuttle that you can rotate by clicking and dragging. Then look at the code to understand where the rotation of the object is set, where the texture file is read, and other aspects. A good exercise would be to try to change the image file that's used as texture – any jpg file would work (but may not look at good). Another exercise is to find another obj file and import it instead using the existing code.

Requirement 1: Shape class hierarchy.

Right now the code has a two stage shape hierarchy: class Mesh.java, and then class ImportedModel.java that extends Mesh. Add your shapes from the last lab to the hierarchy so you have classes Tetra.java and Cylinder.java that extend Mesh, and then add keyboard events to switch between the imported model, tetra and cylinder. Use another model than the shuttle.

The current pair of classes are not optimal. They are inefficient and inelegant. We'd like to improve them so they don't do extra work (each frame they reload the VBOs), and have a proper separate of purpose.

Requirement 2: Virtual trackball.

The current code uses the mouse location mx, my to set rotations around the X and Y axes. The angles set are fixed by location on the screen, so if you click anywhere you jump to that pair of angles. A more elegant solution would be to have a virtual trackball so the object rotates cleanly, and for each click-drag event, the object starts rotating from the last position. You are free to be imaginative here,

but there are standard solutions. This project has been done at other schools, and code is online, so the assignment is being given with that understanding. Document your sources.

http://www.cs.cornell.edu/courses/cs417/2003sp/Homeworks/Project4/trackball-notes.pdf

https://web.cse.ohio-state.edu/~crawfis.3/Graphics/VirtualTrackball.html

http://www.diku.dk/~kash/papers/DSAGM2002_henriksen.pdf

Requirement 3: Lighting

Add one positional light to the project and appropriate shaders to use the light. We'll talk about this part of the assignment in class, and you'll get additional example code.

Submit:

A. A header with CMSC427 fall 2017 Project 1 and your name.
B. A short narrative with what you found out in doing the project with the code, and in particular how you managed to implement the virtual trackball.
C. A copy of the Java source code in the PDF
D. Save as PDF and submit, along with a separate copy of your source files and the new obj file you used.

As a lab, the requirements for code are lightweight in that you don't have to validate your program against all possible inputs, or work on the most general solution. Consider your program a working prototype. You're free to extend, play with, revise, and otherwise make the assignment yours.