Assignment #1 – OpenGL and JOGL

DUE DATE: Tuesday, February 14th (3 weeks)

Overview

This objective of this assignment is to insure that you are sufficiently familiar with the basic structure of applications written in JOGL, OpenGL, and GLSL (including vertex and fragment shaders), to build a system that includes both a simple GUI and graphical output. We will be using these techniques all semester, so it is important to master this framework.

The assignment is to make several relatively simple modifications to Program 2.6 from the textbook. Your program will draw an isosceles triangle that moves around the screen in various ways, and changes color, depending on input from the user (using keyboard, mouse, buttons, etc).

Java Program Structure

This program will become the framework for future assignments in the class. Therefore, it is important to design it from the beginning with *flexibility* and *performance* in mind. This means that your code should be clearly organized into appropriate modules, and that you should avoid allocating memory in the **display()** function.

As described in the Java GUI hints, your Java program will define a class that extends **JFrame** and implements **GLEventListener**. It will attach a **glCanvas** and some buttons to the **JFrame**.

Your JOGL (Java) program will need to incorporate an **Animator** to repeatedly invoke your **display()** method. Your **display()** function can also send additional information as needed to the vertex shader, using uniform variables as shown in the textbook. For example, it could send values that the shader can use to modify the location of the triangle's vertices, or its color(s).

GLSL Shaders

Your GLSL shaders must specify a version of "430", and they must be read in from text files. The vertex shader should hardcode a simple triangle, and may receive some input from the Java program. The fragment shader should output the desired color(s).

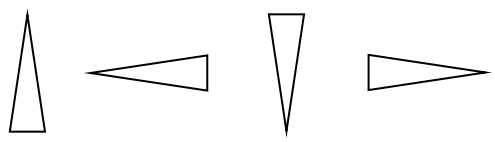
Program Requirements

Your program will make these modifications to Program 2.6 from the textbook:

- The triangle will be isosceles and fairly narrow, rather than the right triangle in Program 2.6.
- Movement must be based on elapsed time. Get the current time from Java, and use it to calculate the amount of time that has elapsed since the previous call to **display()**.
- Determine and display the current JOGL and OpenGL versions on the console at startup.

Your program will also add the following user controls:

- a button that causes the triangle to move in a circle around the **glCanvas** window.
- a button that toggles the triangle between a single solid color, and a gradient of three colors.
- the key (1) makes the triangle point up.
- the key (2) makes the triangle point to the left.
- the key (3) makes the triangle point down.
- the key (4) makes the triangle point to the right.



- mouse wheel control increases and decreases the size of the triangle.

The movement button, mouse wheel, and color toggle should all work correctly regardless of the direction that the triangle is pointing, and should not change the triangle direction.

You do **not** need to use transformation matrices to move the triangle. You can use the simple "offset" technique shown in Chapter 2 of the textbook. We will use transforms in homework #2.

You may add any additional features you like to the program, using additional buttons or keyboard input, so long as you document them in your PDF report. This is entirely optional.

Note that Swing menus *may not be compatible* with the JOGL **`GLCanvas`** object.

Java Packages

Your program code must be contained in a Java package named "**`a1`**" (lower case). Each source file must contain the statement **`package a1;`** at the beginning. If you use an IDE that puts your code in a different package (or the "default package"), you must either change your IDE settings or, when the final version is finished, modify the package statements and recompile everything from a command line. If you do your development directly from a command line, you must create all your source code in a subdirectory named "**`a1`**" and compile/execute it from the parent directory. Also, the "main" class in your program must be named "**`Code`**" (like in the textbook).

It must be possible to compile the program from a command prompt with: "**`javac a1/*.java`**"

Documentation

All code in assignments for this course must be *well-documented*, and you should follow standard Java coding conventions. Class names start with *uppercase* letters, package and variables start with *lowercase*, and names should use "CamelCase" ("**`MyClass`**", "**`myMethod`**", "**`myPackage`**").

Running under Microsoft Windows

Most Windows implementations try to take advantage of "Direct3D (d3d)" to accelerate OpenGL graphics. This needs to be disabled, by adding the following arguments to the "java" command:

```
java -Dsun.java2d.d3d=false -Dsun.java2d.uiScale=1 a1.Code
```

Deliverables

- This is an INDIVIDUAL assignment.
- Submit to Canvas TWO items:
  - a ZIP folder with your A1 submission, as described below
  - a TEXT file (.txt) indicating which RVR-5029 machine you used to test your program
- The ZIP folder contains the following:
  - Java source (.java) and compiled (.class) files, in the proper "**a1**" folder hierarchy.
  - GLSL vertex and fragment shader files
  - a .PDF report file consisting of the following numbered items:
    1. your name, "assignment 1", CSc-155, section number, and "Spring 2023".
    2. a screenshot of your running program
    3. a list of which of the eight program requirements you were able to fully implement
    4. a list of which of the eight program requirements you weren't able to fully implement
    5. instructions showing how to use your program, and which keys/buttons do what
    6. indicate on which RVR-5029 (remote) machine you tested your program

It is a requirement that your program run properly on at least one machine in the 5029 lab. Your report AND text file must BOTH indicate which lab machine your program runs on!