CS 770/870
p0: Installing OpenGL and Drawing Shapes

September 3, 2014

Due: Friday, 9/12 (by 23:59:59)

Lateness: Sat-Sun -3, Mon -10, Tue -15

The goals of this assignment are to insure that you gain access to a working OpenGL environment right away, and to introduce you to the most basic output features of OpenGL within a simple object-oriented context for representing objects that can be displayed.

A. Get a working OpenGL environment on your machine

1. If you plan to use Java, install JOGL (from jogamp.org) or LWJGL (from lwjgl.org). See notes below.
2. Make sure you can access OpenGL and GLUT and *make* on both agate and whatever machine you plan to use for most of your code development. These are already part of any Linux installation. They are also included in MacOS X but you have to install the XCode development environment – even if you don't use XCode for development. If you work on Windows, you might want to install *make* on Windows; if you don't, be sure you've tested the *make* on agate before submitting.
3. Get the *basic* demo code (either C++ or Java) to work on your machine**. You should be able to make NO changes to the Makefile. If you must change it, send me mail as soon as you do with an explanation of what didn't work and your correction.**

B. Extend the *basic* program

4. Add *Rectangle, Quad*, and *Polygon* classes that extend the *Shape* class. A *Rectangle* object should be specified by a location and *width, height.* A *Quad* object constructor should accept an array of 4 *Point*. A *Polygon* object constructor can take information defining any number of points. Feel free to define multiple constructors in order to simplify the task of creating objects. You should use a *Point* class that encapsulates *x* and *y* instead have having to define separate arrays for *x* and *y*.
5. Build a scene using multiple copies of all your classes as well as the Triangle class.
6. Expand the Shape class to support defining, changing and drawing the boundary/fill attributes of openGL; this will, of course, also require new code in the children of Shape. In *OpenGL*, you implement this by drawing the "polygon" twice; for the border use *glBegin(GL_LINE_LOOP)*.

Point allocation

50 Implementation and display of 3 classes that inherit from Shape that draw 3 distinct shapes using multiple colors, locations, and scales.

30 Implement the boundary/fill functionality.

20 Very clear demonstration from the visual results that you have implemented all the required features.

Points are earned by correctly implementing features robustly. Points are deducted for bugs, incorrect implementation, poor style, poor commenting, poor design decisions, non-functioning make file, etc.

Submission information will be available soon.

1. Overall implementation issues

In order to maintain portability between your implementation and ours, you must follow some basic constraints. These should **not** be needed for this assignment, but will soon.

1. Create a directory in your home called *CS770* (not *cs770,* not *CS870*, or anything else).
2. Create subdirectories of *CS770* called *include, lib,* and *jars* (if you are using Java).
3. Download the C++ *basic* demo from agate *~cs770/public/cppDemos/basic* or from the course web page, **OR**
4. Download the *BasicJOGL* demo from agate *~cs770/public/joglDemos/BasicJOGL* or from the course web page.

You must store the demo code in a folder with the same name I used (*basic* or *BasicJOGL*). The Makefile conventions require this!

2. C++ implementation/installation issues

Linux installation: There should be nothing you have to install for this assignment.

Mac OS X installation: 1) Install XCode 2) **From XCode Preferences, install "Command Line Tools".**

Windows: OpenGL comes with windows. The biggest headache with Windows is insuring that the same code will run everywhere. Since *make* is a critical component of (my attempt at) portability, you might want to install from http://gnuwin32.sourceforge.net/packages/make.htm.

Eclipse: You are not required to use Eclipse as your IDE, but you really should use an IDE. If you are not already familiar with VisualStudio or XCode or NetBeans, I recommend Eclipse. As much as I dislike Eclipse, it is available on all platforms and it has an option to manage a project that has a Makefile already. You will need to install the CDT plugin for C++.

3. Java implementation/installation issues

JOGL: Regardless of your platform, you'll need to install the *jogl* and *gluegen* packages from http://jogamp.org. I recommend you install the "current stable" version which you can get by following links or directly at http://jogamp.org/deployment/jogamp-current/archive/ . Once you get to this page, you only NEED to download jogamp-all-platforms.7z. After unpacking this (just need to click on it), you only need the stuff in its *jar* subdirectory and you only need the versions that apply to your system. For my Mac, this was (your sizes will not match):

```
-rw-r--r--  2 rdb  rdb     4008 Jul 20 14:31 gluegen-rt-natives-macosx-universal.jar
-rw-r--r--  1 rdb  rdb   257194 Jul 20 14:31 gluegen-rt.jar
-rw-r--r--  1 rdb  rdb   656087 Jul 20 14:31 gluegen.jar
-rw-r--r--  1 rdb  rdb   368859 Jul 20 14:49 jogl-all-natives-macosx-universal.jar
-rw-r--r--  1 rdb  rdb  4351913 Jul 20 14:49 jogl-all.jar
-rw-r--r--  1 rdb  rdb  1761939 Jul 20 14:49 jogl-test.jar
```

There are 2 versions of the generic jogl code; one with AWT, one without. I deleted the latter.

```
-rw-r--r--  1 rdb  rdb  4079626 Jul 20 14:49 jogl-all-noawt.jar  ← This isn't needed
```

There are two zipped source files; these are useful for some error handling and debugging.

```
-rw-r--r--  1 rdb  rdb  3439714 Jul 20 14:31 gluegen-java-src.zip
-rw-r--r--  1 rdb  rdb 20409876 Jul 20 14:49 jogl-java-src.zip
```

There was a subdirectory, named *atomic*, but it doesn't seem to be needed for programs to run. I kept it, but deleted jar files in it that were not needed by Mac OSX.

Updating CLASSPATH on Linux and Mac OSX. If you have root access, it is common to put user-installed software like this in a subfolder in /sw on Mac OSX or /usr/local on Linux. However, it doesn't really matter. What does matter is that you set your *CLASSPATH* environment variable so that it contains all the jogl/gluegen jar files. You should do this in your *.login* file. Most of you probably use the *bash* shell. If you don't know how to set environment variables in *bash*, look it up on the web.