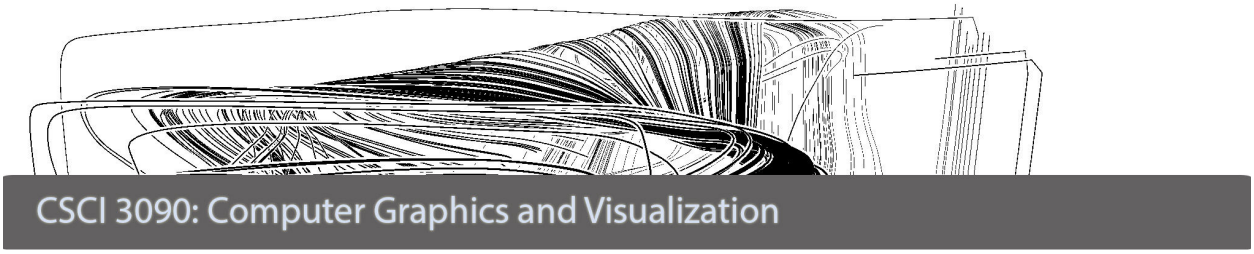

CSCI 3090U Assignment Two (6%)

OpenGL Lighting and Performance

Due: March 2, 2016 (11:59pm)

Introduction

In class we discussed the Phong lighting model and how it can be implemented in OpenGL. In this assignment you will measure the performance of the different ways of implementing the lighting models and the different types of lights. Measuring the performance of graphics programs is a bit of an art, due to the fact that most of the computations are performed on a different processor. We will use one of the standard techniques that does a reasonably good job of performance measurement.

What to Measure?

What is it that we are interested in knowing? We have three different types of light sources, each with a different computation. We could measure the time for each of them. We also have a choice of implementing the lighting models in the fragment shader or the vertex shader. We expect a vertex shader implementation to be faster but of lower quality. This gives us six different conditions that we could measure.

Methodology

You will need to prepare six pairs of vertex and fragment shaders for the six different experimental conditions. The ones where the lighting model is implemented in the fragment shader have been presented in class and are available in example 6. You will need to write the ones where the lighting model is implemented in the vertex shader. Note, you can have more than one shader program in an application. You can switch between shader programs using the glUseProgram() procedure. It is a good idea to load all six of your shader programs at the beginning of your program and then switch between them later.

Now we need to measure the time required to display the model. Basically we need to time the call the glDrawElements in the DisplayFunc procedure. The following function can be used to accurately determine the current time:

```
double getSeconds() {
    LARGE_INTEGER freq, val;
    QueryPerformanceFrequency(&freq);
    QueryPerformanceCounter(&val);
    return (double)val.QuadPart / (double)freq.QuadPart;
}
```

You can call this function once before calling glDrawElements (save the result) and then again after glDrawElements and compute the time difference. But, there is a problem here, glDrawElements normally returns immediately and the GPU executes the command independent of the CPU. The solution to this problem is to call glFinish() before the second call to getSeconds(). This procedure ensures that all the OpenGL commands have been processed before it returns. This function is only used for performance measurement; it is never used in a production program due to its impact on performance.

We now have way to measure the time required to draw the scene once. As good experimentalists we should repeat this 5 to 10 times and then compute the average time required to draw the scene. We can use our idle procedure to do this for us. In the idle procedure you can use a static variable to keep track of the number of times it has been called. Every 10 calls you can record the average time required to draw the scene and then switch to the next program. After you have run all six programs your program can quit. Remember to save the averages so you can include them in your report.

You can use the vase.obj object for testing your program since it loads and displays quite quickly. I have posted two larger models with the assignment that you can use for final testing. It takes some time for these models to load, so you should only use them after you have debugged your program.

You can use the example 6 program as a starting point for this assignment.

General Assignment Guidelines

Individual Work

This is an individual assignment. While you may discuss your progress with your classmates, each member of the class is expected to hand in their own work.

Please include this statement on the cover page of your report:

"I, <insert name>, certify that this work is my own, submitted for CSCI 3090U in compliance with the UOIT Academic Integrity Policy."

In this course, you MAY NOT review, copy or otherwise use any graded academic assignments from prior semesters or other sections of this course, in written, electronic, or verbal form, used in whole or part, including formatting of any assignment.

Report Formatting

Please hand in the following with every assignment in this course:

- A written report (PDF only) describing your solution, highlighting any features of your source code which you would like to bring to the instructor's attention, describing any known problems, answering any questions posed in the assignment handout, and confirming the academic integrity statement. This report should be well-formatted, clearly organized, and use correct grammar and spelling.
- Source code which can be compiled in Windows. If your code has special compilation requirements, include them in a `readme.txt` file.
- Images you produce should be embedded in your report. Keep a high resolution copy available in case it is requested for clarification.