SCHOOL of ELECTRICAL ENGINEERING & COMPUTER SCIENCE
FACULTY of ENGINEERING & BUILT ENVIRONMENT
The UNIVERSITY of NEWCASTLE

Comp3320/6370 Computer Graphics

Semester 2, 2018

Lab Week 2
Jake Fountain
Version 1

Welcome to the COMP3320 Laboratories! These labs serve a dual purpose:

- Hands on instruction with low-level computer graphics programming (OpenGL)
- Additional help with mathematics and other theory topics

OpenGL is a popular library for creating low-level, real-time computer graphics. The first 6 labs will cover the basic programming techniques and mathematics which you will need for the major project. The labs will be using Windows, Visual Studio and C++ as the default platform to demonstrate simple OpenGL examples. Feel free to create your own implementations on other platforms and using different languages, but be aware that demonstrators will have limited knowledge for some platforms.

The "textbook" we will be using will be the set of tutorials at `www.open.gl`. The labs are meant to provide you with help and additional guidance as you complete key tutorials from this site. We don't have time to cover all of this in class, so some exercises will need to be done during the rest of the week (or as you implement features in your project).

Students are also encouraged to ask questions about math exercises from the lectures and resources section. The math content covered by the lectures and exercises may be assessed in exams (unless expressly stated otherwise), so ask questions if you need help understanding how to solve the problems!

1 Summary and Goals

This week's lab covers the basics of displaying graphics with simple geometry, colour and texture in OpenGL. We will be using vertex buffers to feed geometry data to the graphics card and then use basic vertex and fragment shaders for displaying colour and texture on the objects.

2 Glossary

Here are some key terms you will learn about today:

- **Vertex Buffer** - A memory object which stores a list of x, y and z coordinates (typically corresponding to an object we wish to draw) I.e. $(x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_N, y_N, z_N)$

- **Element Buffer** - A memory object which stores a list of indices, usually in triples, which define the connectivity of a set of vertices by grouping polygons, usually triangles. (e.g. (1,2,3) corresponds to the triangle with vertices $v_1 = (x_1, y_1, z_1), v_2 = (x_2, y_2, z_2), v_3 = (x_3, y_3, z_3)$ from the vertex buffer.)
- **Vertex Shader** - A parallel program which operates on a vertex buffer (and perhaps other per vertex information such as texture coordinates)
- **Fragment Shader** - A parallel program which operates per pixel to compute information such as colour and lighting
- **GL Shading Language (GLSL)** - The programming language used to program OpenGL shader programs. The language is similar to C, with some additional graphics-centric types and methods built in. GLSL is built at run time.

It is important to note that shaders operate in parallel on GPU (Graphics Processing Unit) hardware, hence the calculations are fast even for large datasets. In particular, this parallelisation can handle every pixel on a screen easily.

3 During the Lab: Draw a Triangle

This tutorial begins with an introduction to drawing shapes with vertex buffers, and compiling and using shaders. Because this requires a number of new programming concepts, it is expected you will not get much further than drawing a simple triangle in the first lab. **NOTE**: Ordering of the code is important and can be difficult to get right at first. To reduce headaches related to this, the base code includes TODOs comments which will guide the ordering. However, the TODOs are not meant to be completed in the order they appear in the code: follow the open.gl tutorial.

1. Download and extract the Visual C project file that comes with this lab. It contains code for a basic interface using GLFW (a cross-platform OpenGL windowing and input library), inclusion of the SOIL image library, as well as examples of how to perform some of the tasks such as loading and compiling basic shaders in C++.
2. Open the project `LabW2_Code_2018/GLFW_EXAMPLE/GLFW_EXAMPLE.sln` in Visual Studio 2015 and use it as a basis for following the tutorial found at: `http://open.gl/drawing`. If Visual Studio prompts you to login, you can click ignore. Compiling the code (with the green "play" arrow at the top of the page) should yield a blank OpenGL window with a blue background.
3. You should aim to finish through to the "Drawing" section of the open.gl tutorial by adding code to the example.

4 After the Lab: Continue to Uniforms, Element Arrays and Textures

When you have familiarised yourself with the basic drawing process, finish the rest of the drawing tutorial. Also try the texture buffer tutorial at: `http://open.gl/textures`. All of these concepts will be needed for your project and it is expected that you will have completed these tasks by the next lab.