CS4052 Computer Graphics
Assignment #01

Due Date: 25th September 2018
Coursework %: approximately 5%

The purpose of this lab is to introduce you to programming in OpenGL. We will first step you through setting up a project in Visual studio. The provided source code will contain a basic OpenGL structure, with an: init, main, and display function. It will also contain a basic fragment and vertex shader, and functions to use these. Finally, it will contain a triangle, and how to setup this triangle in a VBO for OpenGL to be able to use it.

Outline

- Set up a working VS project, get it running, and complete the exercises
- You will be required to **show your working program** to the demonstrators on the due date and they will grade it.
- Submit a report on Blackboard by the due date. Your report should include a short written description and screen shots.
- This assignment is strictly **individual** (no groupwork).
- If you fail to show up for the lab or to submit your report on time, you will receive a grade of 0%.
- Do not wait until the last minute to start this assignment. This assignment should be straightforward, but it may not go as smoothly as you would like. Assume something unexpected will happen and give yourself time to deal with it. Be sure to attend labs and ask the demonstrators for help.
- Demonstrating a project that was not created by you is considered **cheating** and will be reported as such.

---

1. Setup- create a Visual C++ project for an OpenGL application as follows:

- Launch Microsoft Visual Studio 2017 (Not the Blend version)
- Click File -> New Project
- Choose Visual C++ Projects -> Empty Project in the template chooser
- enter a name for your new project and choose the directory where to create it
- This will create a directory named as your project, with default files in it
- A cpp file named as your project contains the default main function; you can see it in the solution explorer on the right.

- Replace it with the source files provided (in Blackboard)
- To do this,
    - Right-click on the file in the solution explorer and select remove.
    - copy the main.cpp file in Blackboard/Assignments/Lab1/Source Code into your project directory (using windows explorer). If you have a folder within your project directory of the same now, select this.
    - Right-click on the project name in the solution explorer and select Add -> Existing Item
    - select the main.cpp file and click Add
- Generally, this is how you will add pre-existing source files to your projects

The difference between the Debug and Release build:

- You can notice on the top of the Visual Studio Window a drop-box with the value "Debug". By clicking on it you can also select a Release build
    - In the Debug build the complete symbolic debug information is generated to help while debugging applications. Also, the code optimization is not taken into account.
      The application is then slower but can be debugged using breakpoints.
      (Advice: google some tutorials on how to use breakpoints to help debug your code)
    - In the “Release” build the symbolic debug info is not generated and the code execution is optimized.
      Therefore, the application is faster and the size of the final executable is lesser than a debug executable.
    - Note: One can expect to see errors in release builds due to compiler optimizations or differences in memory layout or initialization compared to the debug build. For instance, most compilers initialize automatically variables in debug (for instance integers are automatically initialized to 0), but not in release (random integer value).

Before you can build the project, you first need to install the freeglut and glew libraries:

- copy the freeglut and glew directories from Blackboard/Assignments/Lab1/Source Code to your project’s folder
- now, go back to Visual Studio and go to the solution Explorer and right-click on your project name (which is written in bold-text) -> Properties (the bottom item in the Project menu).
- Select the configuration: “Debug”
    - go to *Debugging*
    - go to the Working Directory field and type $(OutDir) instead of $(ProjectDir)
    - go to *Configuration Properties* -> C/C++ -> General
    - go to the *Additional Include Directories* field and first add the **include** directory of your freeglut folder, then add in another cell the **include** directory of your glew folder.
    - go to *Configuration Properties* -> Linker -> General

    - go to the *Additional Library Directories* field and first add the **lib** directory of your freeglut folder and then the **\lib\Release\Win32** directory of your glew folder (note glew does not have a debug version so we use the Release).
    - go to *Configuration Properties* -> Linker -> Input
    - go to the *Additional Dependencies* field and type " freeglut.lib;glew32.lib; " (but omit the " ")
- Select the configuration: Release
    - go to *Debugging*
    - go to the *Working Directory* field and type $(OutDir) instead of $(ProjectDir)
    - Repeat the previous instructions to set the **include** directory, but set the **glew-1.10.0\lib\Release\Win32** directory for the *Additional Library Directories* in order to use the Release build of freeglut instead of the Debug build.
    - Set the *Additional Dependencies* field and type "freeglut.lib; glew32.lib" (but omit the " ")

Build the project:

- To build, click Build -> Build project name. Do this in both Debug and Release mode as this will create a “Debug” and “Release” folder in your director, needed for adding your dlls.

Your application will also need freeglut.dll and glew32.dll to be included in your application folder. After building your project go into the freeglut folder and copy/paste:

- the freeglut.dll file in **freeglut\bin** to your “Debug” directory
- the freeglut.dll file in **freeglut\bin** to your “Release” directory
- Find the glew32.dll in the **glew-1.10.0\bin\Release** folder and place into the “Debug” and “Release” folders
- When you run the program, you should see a red triangle, occupying most of the space on the screen.

---

2. Exercises

- Now make the fragment shader use the colours specified in the vertex buffer object. (The resulting triangle should be multi-coloured) (~10% of the grade)
- Now try to draw 2 triangles next to each other using glDrawArrays by adding more vertices to your data. (~20% of the grade)
- Now create the same 2 triangles using two different VAOs and VBOs for their data. (~20% of the grade)
- Create two shader programs where the second program uses a different fragment shader that outputs the colour yellow; draw both triangles again where one outputs the colour yellow. (~30% of the grade).

Common mistakes:

*I can't log in" -> go to is-services now
*I can’t access blackboard -> go to is-services now
*Problems with visual studio/trying to load project that is sitting on network drive -> copy to a local folder

*If you are having problems with freeglut or glew (dll error, linking errors etc.), firstly, make sure that you are being consistent with the use of Win32 and Release/Debug, then ask for help from the demonstrators, or try downloading the latest versions (binaries): http://www.transmissionzero.co.uk/software/freeglut-devel/ and http://sourceforge.net/projects/glew/files/glew/1.10.0/glew-1.10.0-win32.zip/download

*x64 project --> if you want to use an x64 project, then you will have to go to the drop-box which says “Win32” on the top panel and select “Configuration Manager”-> Active Solution Platform -> <new> -> x64, and this will copy your project into a x64 version. Also, remember to use the x64 versions of freeglut and glew.
