Computer Graphics Lab1

Winter Semester 2015/2016

Date: 30/10/2015

Assistant Prof. Georgios Papaioannou

Teaching Assistant Konstantinos Vardis

Dept. of Informatics, Athens University of Economics and Business

Introduction

This lab introduces the basic concepts required to write an OpenGL application. The Windowing system is created and managed by FREEGLUT.

Getting started

1. Start Visual Studio and load the solution file (.sln). You should now have a window on the right (or left) hand side of your display that indicates that solution 'lab1' has loaded.
2. In the Solution Explorer (on your right or left) locate the Headers and Source filter. The first one contains the file Renderer.h which simply contains the definitions of the functions of Renderer.cpp we wish to expose. The Source filter contains the Main.cpp which is the main file of the application and the Renderer.cpp which contains the implementations of basic OpenGL functions. Double clicking on these files will display their contents the source code editor window.
3. The Shaders filter contains all the shader files necessary for each lab.
4. The ShaderGLSL class provides basic shader functionality (loading and compiling shaders)
5. The HelpLib files contain basic functions and includes in order to make your life easier :).
6. The solution produces both 32-bit and 64-bit executables.
7. **Make yourself comfortable with the code!**

Key Input

- **ESC**: exit the application

Exercises (Follow these in the order given!!!!!)

1. Familiarize yourself with the code. YES, it's important.
2. Make sure that the Output and Error List window are visible at the bottom panel. If not, enable them through the View menu.
3. Press F5 to run the project. This should give you an empty window. This is your first OpenGL program.

4. Comment out the code in the **Render** function which renders a triangle.
5. Locate the **BuildTriangleVAO** function and enable per-vertex coloring.
6. Modify the vertex and color values and see how this affects the rendering of the triangle.
7. Comment the triangle rendering code and uncomment the quad rendering code.
8. Again. enable per-vertex coloring for the quad.
9. Again, modify the vertex and color values and see how this affects the rendering of the quad.
10. Locate the **BasicGeometry.frag** file and see that the out_color value writes the vertex_color values you were using before. Change the vertex color value and use any color you want. Example: **out_color = vec4(1.0, 1.0, 0.0, 1.0)** should write the color yellow to the output buffer.