521140S Computer Graphics (2019) Programming Assignment I

General information

In this assignment, the goal is to get a working OpenGL environment for development and learn about the basics of OpenGL rendering (how data is stored, shaders, framebuffers etc.). The same environment and libraries will be used for all future programming assignments as well.

Deadline for returning the work is 2019-04-09 24:00. If you need an extension for any reason, please apply for it before the deadline.

The maximum number of points for this assignment is 8, the minimum requirement is to get 5 points to pass the assignment. Missing or incorrect functionality will reduce the points.

Introductions in the task guidelines are based on Windows platform. Programming on Linux or OS system is possible. Please find the Platform-Specific Information in Common additional material page at Noppa. A submission working on one platform is enough.

Assignment is to be finished alone. Feel free to discuss about the problems but sharing code is not allowed.

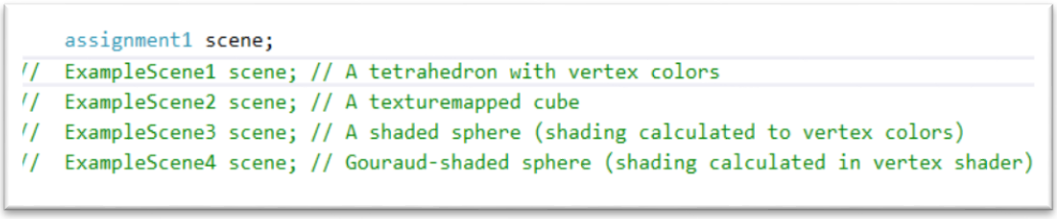
Do not use any existing libraries to solve the rendering part of the assignment outside what are used in the framework itself. The purpose of this course is to learn how to implement graphics engines and not using existing ones.

Workstations. Framework has all the required libraries and a project file for MS Visual Studio 2013+ included for both 32- and 64-bit platforms. PC classrooms (TS135 and TS137) have Visual Studio 2015 installed so it is possible to work there.

Tasks

In this assignment, you are expected to render a simple scene, where you implement the code to draw a cube, to render with shaders, and to finish basic interactions with input devices.

1. Get included examples to work.
   a. Download example code package from Noppa.
   b. Check the *main.cpp* file. Open each scene class, build and run the code respectively (Figure 1). Take a screenshot of each example scene (*i.e.*, ExampleScene1, ExampleScene2, ExampleScene3, and ExampleScene4) and include them in the documentation. Each example scene provides a different example and you may reuse the code in your own work. (1 p)



```
    assignment1 scene;
//  ExampleScene1 scene; // A tetrahedron with vertex colors
//  ExampleScene2 scene; // A texturemapped cube
//  ExampleScene3 scene; // A shaded sphere (shading calculated to vertex colors)
//  ExampleScene4 scene; // Gouraud-shaded sphere (shading calculated in vertex shader)
```

Figure 1

2. Render a simple cube.
   a. The code framework for the assignment has been created in *assignment1.cpp* and *assignment1.h*.
   b. Replace the code "ExampleScene scene;" with "assignment1 scene;" in the *main.cpp* file, and render your own scene in assignment1. You should add all the code in *assignment1.cpp* and *assignment1.h* for submission.
   c. Create a cube in void assignment1::createCube(). You may take the ExampleScene1 and ExampleScene2 as the references. (2 p)

Hints: The cube is composed of 6 faces and the rendering should be based on triangles. Thus, there should be totally 12 triangles for rendering one cube. In assignment1::createCube(), an example of rendering one triangle of the front face is provided. Please complete this part with the full set of vertices. Also, always make sure that the vertices are ordered correctly such that the normal of the faces always points outside (Figure 2). You may also learn this part in tutorials (i.e., http://www.opengl-tutorial.org/beginners-tutorials/tutorial-4-a-colored-cube/).

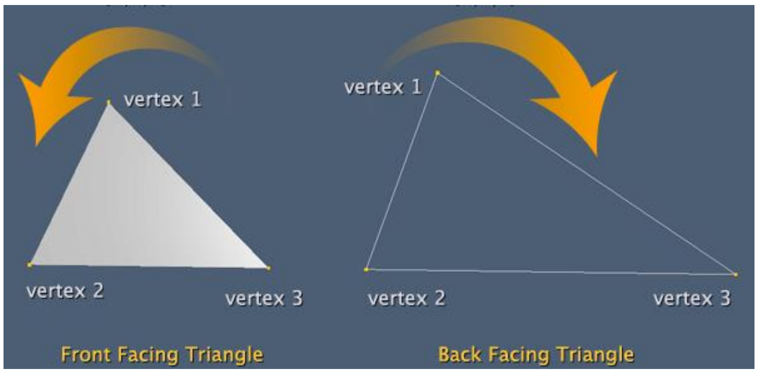


Figure 2

3. Change the cube color by mouse clicking.
    a. Learn SDL handle events and add a SDL_Event event in bool assignment1::handleEvent(const SDL_Event &e) in the file *assignment1.cpp*. (1 p)
    b. You may get the color of the mouse clicking point by the function glReadPixels(). Note that the actual pixel position is (x, windowheight-y) as you should match screen coordinate to window coordinate system. In window coordinate, the origin point is the left lower corner. (1 p)
    c. Use GLSL for shading. Create your own fragment shader *assignment1.fs* and vertex shader *assignment1.vs* for color shading. (1p)
    d. Check the `glUniform4f()`/`glUniform4fv()`/`glUniform3f()`/`glUniform3fv()` function, and set the color property here to change the color. Together, you should change the fragment shader, and add "uniform vec4 cubeColor" as the input variable to the shader, and set the cubeColor as color property to `glUniform4fv()`. (2 p)

Hints: For SDL handle events, here is an example of getting SDL_PollEvent, https://docs.huihoo.com/sdl/sdl-1.0-intro-en/usingevents.html. There are also example code from line 234 in the *main.cpp* file. You may catch the SDL handle event about mouse button down, get the button location, and use glReadPixels() to obtain the corresponding color. You may use uniforms to pass data from the code to the shaders. The flow is that you set the cube color->pass the cube color to shader with `glUniformXX()->the shader get the color setting and execute shading on GPU.` You can find a nice example in https://learnopengl.com/Getting-started/Shaders.

What to return

Please send the following to any one of the TAs using **title format “CG1 _<student id>_ <given name> _<family name>”** during the given deadline date at the latest (Deadline date is at first page, assume Finnish time).

The submission should contain the following documents:

1. A PDF document describing your programming work and how you solved your assignment. Make sure your document contains at least the following information:
    a. Your name and student id number
    b. What operating system(s) was/were used to develop and test the software.
    c. Number of hours needed to finish the assignment
    d. Description of how you solved the programming tasks (General description of the method, any used external information sources used etc.) Please include information of any problems you encountered and how you managed to solve or work around them.
    e. Any feedback you wish to provide.
2. A zip file containing **assignment1.cpp, assignment1.h, assignment1.vs** and **assignment1.fs.**

Contacts

Please contact the TAs for submission or for help if you face with problems. The TAs for the programming assignments are as follow:

Tuomas Varanka: tuomas.varanka@oulu.fi (Finnish & English)
Yingyue Xu: yingyue.xu@oulu.fi (English)
Nhat Vo: nhat.vo@oulu.fi (English)