Computer Graphics DT 3025
Lab 1

Stevan Tomic and Martin Magnusson, November 2015
Based on material by Mathias Broxvall

Lab 1 - An introduction to OpenGL and GLSL

In this first lab you will first shortly get familiar with the operating system and your development environment and then solve a number of shorter OpenGL exercises of increasing difficulty. For a full specification of the OpenGL functions that are available to be used, please see www.opengl.org. Note that there is a list of useful OpenGL and GLU functions towards the end of this lab. Please look up these functions online and read what they does and how they can be used in your code. Also, it is highly advisable to read the tutorial on https://open.gl (or similar). This should increase your familiarity with OpenGL programming, and provide a basic background to deal with the tasks in the labs.

Exercise 1.1 Your first OpenGL/GLSL program(s).
Download, unpack and compile lab 1.1 from the course webpage. Launch the application. Congratulations, you now have your first OpenGL application running – showing a very interesting dark yellow screen that does nothing. Quit the application by pressing the escape key (or closing the window).

Your job now is to make the lab a bit more interesting.

If you take a look in the code you will note that there are a function `World::doRedraw` that is called periodically. This function already contains some OpenGL commands that launches two pieces of a *shader program* drawing a rectangle (consisting of two triangles) covering the full screen. The source code for the shader programs can be found in `vertexShader.vs` and `fragmentShader.fs` and corresponds to the *vertex shader* and *fragment shader*.

Start by opening `fragmentShader.fs` and try to change the colour that is drawn to the screen. This is done by changing the colour that is assigned to `fragmentColour`. You can for instance add the `position` variable to the `vertexColour` to get a gradient over the full screen. Note that you can make changes to the sourcecode of the shader files while the program is running, and recompile it on the fly by pressing the ’l’ key.

Next, you should continue by extending the *doRedraw* function of your main application so that it will draw some more interesting objects. Start by adding two more triangles looking like a rectangle that covers a part of the center of the screen. You can do this by extending the list of vertices that are given to OpenGL and by extending the list of graphic primitives and corresponding vertex indices that are given for the drawing command.

The rectangle appearing from these two triangles should have the pure colours *red, green, blue, yellow* in the four different corners and *there should be no visible edge* between the triangles that make up the rectangle. You should use `glDrawElements` for this and extend the list of vertices (and colours) by exactly four and the list of triangles (indices) by two..

Take a screenshot of your application (this should be included in your lab report).

Continue by changing the direction in which your rectangle is subdivided (eg. change the orientation of the diagonal from top-right to bottom-left by defining vertices in the clock-wise order), but keep the colours in the same positions on the screen (thus you will need to change the order in which you give the colours to OpenGL). Ideally this should look identical to your previous screenshot, *does it look the same?*.

Take yet another screenshot of your application and compare it with the first.

Finally, change your code to use `glDrawArrays` to draw both of the rectangles now on the screen. Take a screenshot and compare it to the two previous ones. What can you learn from this?

Question 1.
Explain *why* there is a difference between the interpolated colours in the different screenshots above.

Exercise 1.2 Additional drawing primitives.
Read online to see how the drawing primitives `POINTS`, `LINES`, `LINE_LOOP` and `TRIANGLE_STRIP` is working.

Augment your code from the previous exercises to draw at least one of each of the primitives above. Include a screenshot that shows all these priitives in the lab report.

Note that you are still NOT allowed to use depracated functions such as `glBegin` or `glVertex`.

Examination

To pass the lab you must hand in a report showing the screen-shots from your program and answering the question.

Deadline November 10