# TDT4195: Visual Computing Fundamentals

**Computer Graphics - Assignment 1**

September 27, 2016

Bart van Blokland
Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU)

- **Delivery deadline: October 14th, 2016 by 22:00.**
- **This assignment counts towards 3% of your final grade.**
- You can work on your own or in groups of two people.
- Deliver your solution on *itslearning* before the deadline.
- Use the Gloom project as your starting point.
- Do not include any additional libraries apart from those provided with Gloom.
- Upload your report as a single PDF file.
- Upload your code as a single ZIP file solely containing the *src* and *shaders* directories found in the Gloom project. All modifications must be done to files in these directories.
- Your code must be runnable on the lab computers at IT-S 015.
- All tasks must be completed using C++.
- Use only functions present in OpenGL revision 4.0 Core or higher. If possible, version 4.3 or higher is recommended.
- The delivered code is taken into account with the evaluation. Ensure your code is documented and as readable as possible.

Questions which should be answered in the report have been marked with a **[report]** tag.

**Objective:** Familiarise yourself with C/C++ and OpenGL. Basic rendering of primitives, trying out Shaders.

## Task 0: Preparation [0 points]

a) Ensure the following tools are installed:

- CMake
- Git
- A C/C++ compiler, such as GCC or MSVC++.

These have already been installed for you on the lab machines.

On Microsoft Windows, it may be easiest to install Microsoft Visual Studio Express, which can be downloaded for free and comes with the MSVC++ compiler, and it is the easiest to work with in combination with CMake. Alternatively, the full version of Microsoft Visual Studio can be downloaded for free through MSDNAA (Guru).

b) Clone the Gloom repository using the command:

```
git clone --recursive https://github.com/senbon/gloom.git
```

**Note**: the additional `--recursive` flag is required here for downloading the third party libraries Gloom depends upon.

If you forgot to include the `--recursive` flag when cloning the repository, use the following command:

```
git submodule update --init
```

c) Compile the project.

For compilation instructions, see:

https://github.com/senbon/gloom#compilation

d) Run the project.

You should see an empty window if compilation is successful and your GPU supports the OpenGL version required for these labs (4.0 or higher).

## Task 1: Primitive Rendering [2 points]

Depending on your situation, it may be beneficial to answer subtasks in this task out of order if you get stuck on answering one.

a) **[0.7 point]** Create a function which sets up a Vertex Array Object (VAO) containing triangles.

The function must take an array of three dimensional vertex coordinates (`float*`) and an array of indices (`int*`).

The integer ID of an OpenGL VAO set up to draw the input arrays must be returned.

The contents of the buffer can be assumed to exclusively contain triangles, specified by floating point coordinates.

Refer to chapter 2 of the OpenGL guide for detailed information on how to accomplish the objective of this task.

OpenGL can be extremely "picky" about values and parameters it is given. A single incorrect parameter or input value can cause nothing to be rendered, and no error to be reported. If you experience problems, check your code carefully line by line. It is also recommended to come to the designated lab hours for help.

The Gloom project includes the `printGLError()` function. This function prints the last OpenGL error that occurred, but only if one occurred in the first place. You can place it after OpenGL function calls to verify whether any errors occurred.

b) **[0.3 point]** **[report]** Define and instantiate a VAO containing at least 5 distinct triangles using the function you defined in a).

All triangles must be visible. Put a screenshot of the resulting scene in your report.

Hint: these coordinates will give a visible triangle when you perform the drawing correctly:

$$v_0 = \begin{bmatrix} -0.6 \\ -0.6 \\ 0 \end{bmatrix}, v_1 = \begin{bmatrix} 0.6 \\ -0.6 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} 0 \\ 0.6 \\ 0 \end{bmatrix}$$

c) **[0.5 point]** **[report]** While drawing one or more triangles, try to change the order in which the vertices of a single triangle are drawn by modifying the index buffer. A significant change in the appearance of the triangle(s) should occur.

i) What happens?

ii) Why does it happen?

iii) What is the condition under which this effect occurs? Determine a rule.

d) **[0.5 point]** **[report]** Draw a single triangle passing through the following vertices:

$$v_0 = \begin{bmatrix} 0.6 \\ -0.8 \\ -1.2 \end{bmatrix}, v_1 = \begin{bmatrix} 0 \\ 0.4 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} -0.8 \\ -0.2 \\ 1.2 \end{bmatrix}$$

Put a screenshot of the result in your report.

This shape does not entirely look like a triangle.

Explain in your own words:

i) What is the name of this phenomenon?

ii) When does it occur?

iii) What is its purpose?

Hint: try to move vertex coordinates around and see what happens.

## Task 2: Simple Shaders [1 point]

It is highly recommended you read through chapter 3 of the OpenGL guide prior to answering the questions in this task.

a) **[0.3 point]** **[report]** Explain the following in your own words:

i) What is a Shader?

ii) OpenGL makes a distinction between a Program and a Shader. What is the difference?

iii) What are the two most commonly used types of Shaders? What are the responsibilities of each of them?

b) **[0.2 point]** In the handout code, you'll find a pair of basic shaders in the folder "gloom/shaders".

Use the code in Shader.hpp to load the shader pair from the source files. Apply the shaders on your scene containing at least one triangle.

Instructions for using this class can be found here:

https://github.com/senbon/gloom#shader-class

c) **[0.2 point]** **[report]** Include a screenshot showing the effect of the shader pair when enabled.

Explain in your own words:

i) What is the purpose of layout qualifiers in the OpenGL Shading Language (GLSL)?

ii) What is the difference between a vertex attribute and a uniform variable in a GLSL shader?

d) **[0.3 point]** **[report]** Modify the source code of the shader pair to:

i) Mirror the whole scene horizontally and vertically at the same time.

ii) Change the colour of the drawn triangle(s) to a different colour.

Put a of each effect in your report.

Explain briefly how you accomplished each effect.

Before you submit your work: ensure that each OpenGL function call (those of the form `gl..()`) is present in the OpenGL 4.0 Core library. You can use the website `http://docs.gl/` to only show GL4 functions, and search that list. If you have exclusively used the guide, this should not be necessary.