Lab 1: VBOs and VAOs

Objective

Make a class in C++ which manages a **v**ertex **b**uffer **o**bject. At a minimum the class should allow you to set the following vertex attributes:

1. Vertex Positions
2. Vertex Normals
3. Texture Coordinates

The class must have the following functionalities:

1. Functions to set attribute data, e.g. addPositions(float *positionData, …)
2. A function to free memory allocated on the GPU, e.g. destroy(...)
3. Functions to bind, render and unbind a VAO

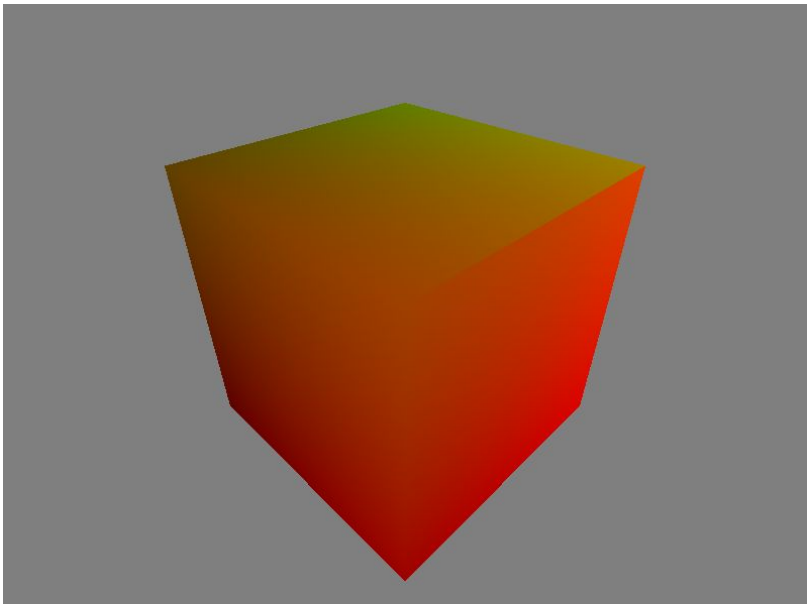
Note that the above functions are pseudo-code suggestions. Your implementations can vary as long as they perform the required functionalities.

Starting Point

Download “Lab1Starter.zip” from BlackBoard, open the solution and navigate to CubeData.h. Here you will find some arrays with data. These are the vertex attributes for a cube. Use this as test data for your class. Remember the task is to make a general VBO class so do not hardcode this. VertexBufferObject.h contains a skeleton of a potential VBO class. Your class does **not** have to look like this. Your implementation can be however you see fit. VertexBufferObject.cpp contains many comments which may be of use to you. Once you complete the VBO class you will need to tell the class to use the cube data provided, do this in the InitializeCube function found in main.cpp

Base Expectation

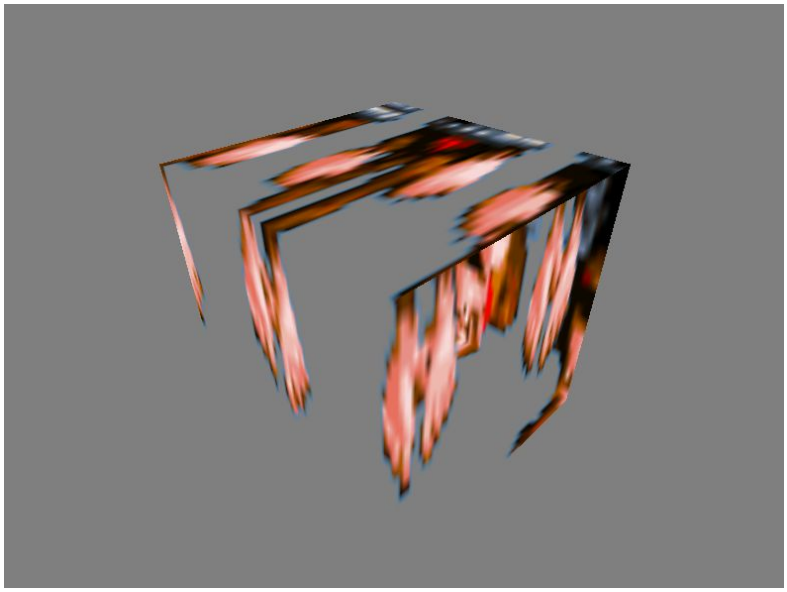
The base product should look like this:



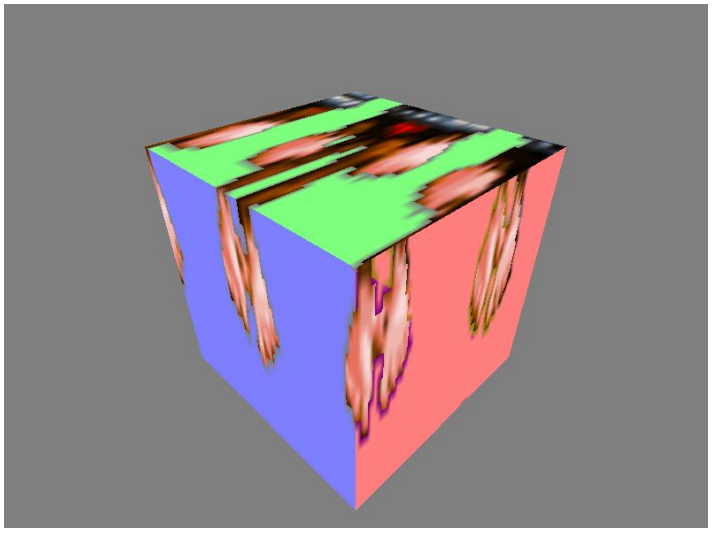
This is a visualization of the cube's UVs (texture coordinates). This is what the provided shader program does.

Additional Contribution

For the additional contribution, as a base implement texturing in the fragment shader:



Once you have texturing, add something (anything!) which makes use of the normals. Here is an example:



Another suggestion is simple shading. You can do anything, be creative :)

Grading

You are being graded on functionality and correctness.

Quality of core lab requirements (/10)
- Do you have a VBO class?
- Do you use a VAO?
- Does it work?
    - Does the cube appear correctly?

Quality of additional contribution (/4)
- Does the shader make use of a texture?
- Does the shader make use of the normals.

Project directory submitted and is clean (/2)
- Do not submit the intermediate directory and the .sdf file. If your submission is extraordinarily large it may not be marked and you will receive a grade of 0.

Project compiles and runs without error (/2)
- When project is run, it should compile successfully. If it does not the marker will try to fix it for no more than 2 minutes, if they cannot then significant marks will be lost.

Lab report with all explanations and requirements (/2)
- In no more than 200 words (can be less), here are some things you might talk about
    - What a VBO and a VAO is
    - The purpose of your VBO class.
    - The difference between immediate and retained modes
    - Talk about the shader effect for your additional contribution

Bonus

Add the ability to interleave the VBO attributes. **(/4)**

Hints

Key OpenGL functions you will need to use:

To create the VBO and VAO

1. glGenVertexArrays
2. glGenBuffers

To bind VBOs and the VAO

1. glBindVertexArray

To send data to the GPU

1. glEnableVertexAttribArray
2. glBufferData

To draw the VAO

1. glDrawArrays

To free GPU memory

1. glDeleteVertexArrays
2. glDeleteBuffers

The documentation for the above function can be found here: http://docs.gl/