Lab 4

Code Generation

Overview This lab involves generating ARB fragment shader assembly [1]. Your compiler will compile code into a text file (`frag.txt`). You will be provided with two MiniGLSL and OpenGL programs that you can use for testing.

Starter Files No starter files (aside from test programs and documentation) will be provided. You will build on your assignment 3 directly.

Fragment Shaders Fragments are data structures that represent potential pixels on the screen. A fragment shader is a programmable processing unit on the GPU that takes as input fragment information and OpenGL scene information, and outputs pixels. A fragment shader program runs on each fragment independently. If the fragment shader program decides to output the fragment to the screen, it calculates it's new color and writes it to a pixel on the screen (equivalent to updating the `gl_color` variable in MiniGLSL).

For example, a programmer can make a surface look like a net by discarding fragments that map to odd pixel coordinates.

In this course we are going to use the ARB fragment program assembly language to program the fragment shader. Please refer to the ARBFragmentProgram document (included in starter files) or to the link in the footnote for a detailed description of the ARB fragment program assembly language.

The starter files also contains a document about ARB vertex programs. This documentation might be helpful as there are many similarities between fragment and vertex programs. Note: you are to compile *fragment* programs, not vertex programs.

Demo / Test Code You are provided with two demo OpenGL programs. Both demos loads and executes a fragment shader program from a file called `frag.txt` (your compiler must output this file). Both demos include the original MiniGLSL code in a separate file with extension `*.frag`. Your job is to compile the miniGLSL code for each demo into a frag.txt file, and make sure that the demo executes properly.

The demos are suppose to do the following:

Demo1	Display a sphere. Initially the sphere is dark blue, when the shader is turned on, the color of the sphere should change as you move up the x and y axises.
Demo2	Initially a regular lighting model will be applied. When you turn the shader on, the lighting model should change to phong lighting. You should notice that the shade boundaries on the object are much sharper.

How to run each demo:

1. Go to the directory of the demo that you want to run.

[1] See the ARB_vertex/fragment_program section: http://www.renderguild.com/gpuguide.pdf

2. Compile the MiniGLSL code (from the `*.frag` file) into a `frag.txt` file.
3. Build and run the executable.
4. Initially the fragment shader is disabled (the GPU does not run the code from `frag.txt`).
5. You can start executing the shader code by pressing `f`. You can also change other options by following the instructions in the execution terminal.

Debugging and Testing In order to debug your assembly code you will write your own MiniGLSL benchmarks and compile them into a `frag.txt` file. You can use either one of the demos to load and execute your shader assembly code (Just follow the instructions from the previous section). If your assembly code is syntactically wrong, the OpenGL demo program will output an error message when you try to run it.

What to Hand In As well as handing in all requested documentation for Lab 3, you should also include a clear description of the methodology you used to generate the code. Specifically you should describe:

- How did you implement non-trivial math operations.
- How did you handle boolean types.
- How did you implement if statements.
- How did you implement constants.
- How the code for each type of node was generated.