University of Toronto
CSC467F Compilers and Interpreters, Fall 2018

Assignment 4: Code Generation

Introduction

This assignment involves generating ARB fragment shader assembly. The details for the ARB assembly can be found on the course website. Your compiler will compile code into a text file (`frag.txt`). You will be provided with two MiniGLSL and OpenGL demo programs that you can use for testing, which are also accessible from the course website.

You will build this assignment directly on top of your Assignment 3 implementation.

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 lab, we are going to use the ARB fragment program assembly language to program the fragment shader. Please refer to the ARB Assembly documentation or to the online documentation[1] for a detailed description of the ARB fragment program assembly language.

Note that you may also find online documentation about ARB vertex programs. There are many similarities between fragment and vertex programs, but in this lab you are only to compile *fragment* programs, not vertex programs.

Demo Program

You are provided with two demo OpenGL programs. Both demos load and execute a fragment shader program from a file called `frag.txt` and your compiler must output this file. Both demos include the original MiniGLSL source 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 two demos are supposed to do the following:

- **Demo 1** 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 axes.
- **Demo 2** 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.

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

How to run each demo:

1. Go the the directory of the demo that you want to run.
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, which means the GPU does not run the code from `frag.txt`.
5. You can start executing the shader code by pressing key 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 by just following the instructions in 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.

Submission

Pack up your code and submit your assignment by typing the following command on one of the UG EECG machines:

```
tar czvf lab4.tar.gz compiler467
submitcsc467f 4 lab4.tar.gz
```

Please be careful to make sure that you only submit the compiler code and do not submit the demo code.