Lab 3: OpenGL Template

The minimum of an OpenGL application requires a lot of lines of code. So, it is a good idea for you to have a template source code for your future labs and projects. In doing so, you can simply copy your template source code and modify it.

What To Do?

This instruction assumes that the current directory is `/[YOUR_DIR]/src` where `/[YOUR_DIR]/` can be any such as `/home/username/cs1566/`. Note that you can use whichever directory structure you want according to your style. This instruction also assume that your linear algebra library from Lab 2 are `linear_alg.h` and `linear_alg.c`. Again, they are just names for this instruction. You do not have to change the names of your files to be the same as this instruction.

It is a good idea to have your library and some common libraries in a separate directory. Create the directory named `mylib` (again, can be any names) under the directory `/[YOUR_DIR]/src`. This directory will contain your linear algebra library as will as `initShader`. Now, copy the following files into `/[YOUR_DIR]/src/mylib`:

- `linear_alg.h`
- `linear_alg.c`
- `initShader.h` (can be downloaded from Lab 3 or use the one from Lab 1)
- `initShader.c` (can be downloaded from Lab 3 or use the one from Lab 1)

Next, create the directory named `template` under the directory `/[YOUR_DIR]/src`. Again, it can be any names you like to use. Now, download the following files into the `/[YOUR_DIR]/src/template` directory:

- `template.c`
- `vshader.glsl`
- `fshader.glsl`
- `makefile`

There are a couple places that you need to modify according to the names of directories and files in your programming environment. The following is the code snippet of `template.c`:

```
/*
 * template.c
 *
 * An OpenGL source code template.
 */

#include <GL/glew.h>
#include <GL/freeglut.h>
#include <GL/freeglut_ext.h>
#include <stdio.h>

#include "../mylib/initShader.h"
```

```
#include "../mylib/linear_alg.h"

#define BUFFER_OFFSET( offset )   ((GLvoid*) (offset))

vec4 vertices[6] =
{{ 0.0,  0.5,  0.0, 1.0},        // top
 {-0.5, -0.5,  0.0, 1.0},        // bottom left
 { 0.5, -0.5,  0.0, 1.0},        // bottom right
 { 0.5,  0.8, -0.5, 1.0},        // top
 { 0.9,  0.0, -0.5, 1.0},        // bottom right
 { 0.1,  0.0, -0.5, 1.0}};        // bottom left

vec4 colors[6] =
{{1.0, 0.0, 0.0, 1.0},        // red   (for top)
 {0.0, 1.0, 0.0, 1.0},        // green (for bottom left)
 {0.0, 0.0, 1.0, 1.0},        // blue  (for bottom right)
 {0.0, 1.0, 0.0, 1.0},        // blue  (for bottom right)
 {0.0, 1.0, 0.0, 1.0},        // blue  (for bottom right)
 {0.0, 1.0, 0.0, 1.0}};        // blue  (for bottom right)
```

For the file `template.c` shown above, you need to modify the following:

- The `include` statement: Replace `mylib` by the name of your directory that contains your linear algebra libbrary and `initShader.h` and `initShader.c`.
- The array of `vec4` is initialized according to structure implementation. If you use array of `float`, you **may or may not** need to modify how the program should initialize them.

Next, you also need to modify the `makefile`:

```
CC       = gcc
CFLAGS   = -O3 -Wall
LIBS      = -lXi -lXmu -lglut -lGLEW -lGLU -lm -lGL
OBJDIR   = ../mylib
OBJS     = $(OBJDIR)/initShader.o $(OBJDIR)/linear_alg.o

template: template.c $(OBJS)
        $(CC) -o template template.c $(OBJS) $(CFLAGS) $(LIBS)

$(OBJDIR)/%.o: %.c %.h
        $(CC) -c @< -o $@ $(CFLAGS)
```

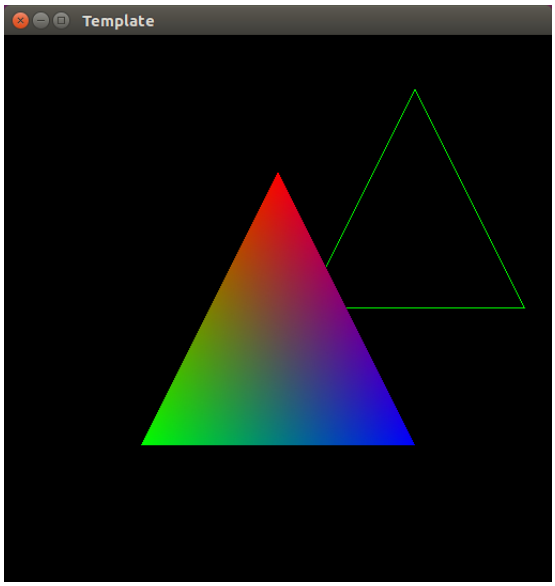
Again, change `mylib` to the directory that contains your linear algebra library. Also change the name `linear_alg.o` to the name of your linear algebra library (again with `.o`).

Testing Your Template

Execute the following command:

```
make template
```

If all go well, you should have the executable file named `template`. Run it and you should see the familiar two triangles:



If there are problems during compilation and linking, most likely they are related to names, directories, and array initialization. Try to fix them until you are able to run the `template` program. Contact the instructor or TA if you cannot solve the problems.

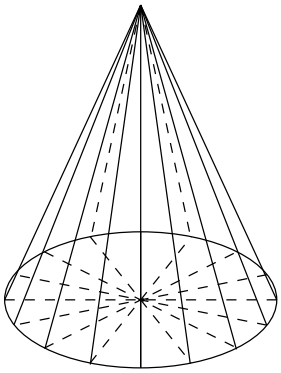
Demo

Once your template is working properly, perform the following steps:

1. Create a new directory named `lab03` under `/[YOUR_DIR]/src` directory
2. Copy the content of `/[YOUR_DIR]/src/template` to `/[YOUR_DIR]/src/lab03`. Make sure that you actually use **COPY** not **MOVE**.
3. Rename the file `template.c` to `lab03.c`
4. Edit the `makefile` by renaming all occurrence of the string `template` by `lab03`
5. Execute the command:

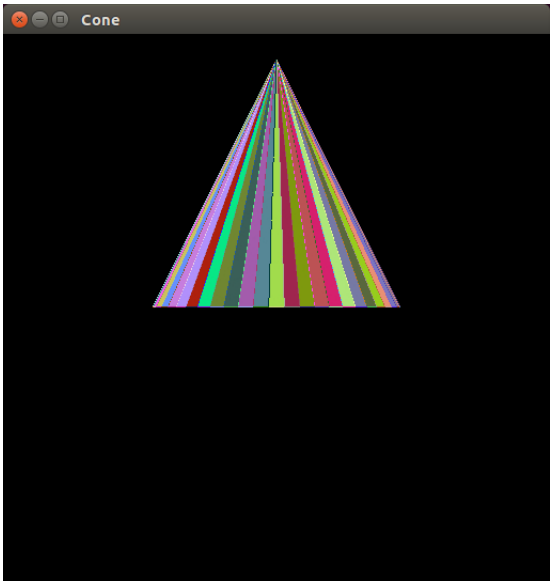
   ```
   make lab03
   ```

6. If all go well, you should get a new executable file named `lab03`. Run it and you should see the familiar two triangles
7. Edit the file `lab03.c` that display a solid 3D cone as shown below:



A solid cone consists of two parts, the bottom and the side. As you may be able to imagine that the bottom part is just a simple circle where its front side is facing down (from the above picture). Recall that a simple circle can be generated by a number of triangles. The side of a cone is simply a number of triangles as shown in the above picture. Again, do not forget about the order of vertices so that each triangle is facing outward from the cone.

For this part, you are required to **implement two functions**. The first function will be used to generate an array of vertices for a cone and the second function will be used to generate an array of random colors (**one color per triangle**). Note that when you generate a cone, it can be in any size and in any orientation. Pick the one that you are comfortable with. For example, one may decide to create a cone with height 2 unit where the base is on the plane $y = -1$ and the tip is at the coordinate $(0, 1, 0)$. One may decide to create a cone with height 1 where the base is on the plane $y = 0$ and the top is at the coordinate $(0, 1, 0)$. There is no restriction on this part. Make sure that you cone looks (kind of) smooth. Remember that the more triangles that you use the smoother the cone. Here is an example:



Note that you are not allowed to manually define each vertex of your cone. Your cone must be generated using some kind of formula using trigonometry theory.

Submission

Zip all files related to this lab into the file named `lab03.zip`. You also need to show to your TA that you accomplished the following:

1. Have a separate library directory to store your linear algebra library and `initShader` files
2. Have a template directory and show that you are able to execute the following commands under your template directory:

   ```
   make template
   ./template
   ```

3. Have a `lab03` directory and show that you are able to execute the following commands under your template directory:

   ```
   make lab03
   ./lab03
   ```

   If you did not use a UNIX system for your OpenGL application, show the TA that you are able to generate the solid 3D cone.