Final Project: System Objects

Introduction:

This is a simple simulation of object with texture and lighting. You can move straight, backward, right, left of viewpoint using arrow keys. The point also implemented to rotate viewpoint. And you can zoom in and zoom out by “Q” and “E” keys. There are some bug like the viewport is work well with window size of 500:458 (I defined it in main.cpp). This bug is not yet fixed because I did not fully undersood the way openGL 4.0 work (I will explain more below). And there are 1 more inconvenient thing is the mouse is fixed in middle of window space. It is used for the benefit of camera direction calculated. Therefore, if you want to close the program, Press button “Esc”.

- NOTE: if you just want to run my code, go straightforward to " Requirement to compile and run my program." part, skip all previous parts. There is no neccessary information in there.

Objective:

Simulation a static object with texture and lighting. It is simple and very concentrade on studying purpose. There is no new, no convolution, innovative or busy possible in here. I just try to apply what I studied with some come over difficult of openGL core profile. If you feel boring when using it, it is understandable because the worth value of this project is laying in lying in the source code and for who want to understand the openGL v4.0 only.

Guideline for Begginner

For study purpose, this project is develop from scratch but not from nothing, it following this tutorial. You can see many same idea or event same source code in there. But you should see also many difference in source code and example because the outdate of this tutorial. And the another source I want to according is khronos.org website where I spend most of time. You can search most openGL function detail and function explaination in there. If you want to understand carefully about openGL immediate mode (old version) and openGL core profile (new version from > 3.0), this is perfect website for you (with many searching in stackoverflow also).

Finally, after 2 day without sleeping, I can building the program which CAN BE RUN with implemented texture and light shading in fragment shader.

OpenGL Overview

I will not meantion about definition or Lib or something else, everyone can search it in internet. What I will talk is 2 mode of OpenGL which lead me to many painful experience in designing and understanding the way it work.

Immediate Mode (Legacy OpenGL)

As my understand, this is a bund of old definition and specification which used to implemente old function of openGL (they refer to openGL 1.0 and 2.0). The benefit of these specification is the abstraction. the function was designed to easy for interpreting and programming. Basically, most of function and technique was bounded into API and programmer will assembler them.

This is easy for the beginner when changing from math domain to programming domain. You can understand directly what the program do when reading it. This reason why my practical often was writen in openGL 2.0 (support many function of immediate mode).

And my first implementation final project use this mode. It is so simple. BUT the abstract is trade off with the flexible and performmance. Most of system mechanism was hidden, like the way vertices was store in momery, how can we program shader, how can we send data to GPU, how does GPU work, v.v. Because most of the task is automate behind sense, many configuration is fixed and the programmer lost the chance to discover and modify during program time.

In my case, when I program in openGL v2.0, most of function can be unserstandable, but I can not implement many technique like light shading or texturing. I dont know how to do it, there are function to open the lighting effect, but modify is not. And I panic for half of day !

Core profile (new opengGL)

Today, most of specification go in this way, instead of hiding everything, openGL provide many function allowing client to access more in memory, buffer, the transfer data between CPU and GPU, they way GPU work. In other word, the function is more and more technique than before. If you want to create the vertex, you dont call api and push value, you declare your own buffer in memory, tranfer it to VRAM (GPU memory). And so far with texturing, you write your all texture buffer, allocate it, feed data and bind to target. This is the first time I know the role of GPU in processing shader effect, how to program it with shader language and compile to run.

The thing become too difficult when change from the abstracted function of openGL 2.0 to mostly manual function in openGL 4.0. I spent 1.5 day without sleeping to read, search and understand the mechanism. My feeling is so terriable,

but it is worthy, absolutely. I can program small program with shader lighting and texturing.

Requirement to compile and run my program.

Standard lib and tool

- G++ (> ver 9.3.0) // compiler and standard lib for C++
- make

library requirement

- libglfw3 // for open window
- glm // lib for graphic mathimatic
- glew // provide many extension function and define

Some difficult from my friend

- Can not which kind of libglfw to install

There are 2 version: glfw-wayland (for wayland render system) and libglfw3 (for system X server). The libglfw3 is preference in all Window, MAC, LINUX. But if it can not run, try glfw-wayland.

Linking

- If you are familliar with make. You can read my makefile to know which LFLAG needed (Linux only). I dont support Window so if you use window, you will need to do it yourself.

Dependency

- All my dependency is on "common/" folder, and their header file is "common/shader.h"
- Make sure you link all "common" path when compiling

Folder explain:

- "common" : dependency lib
- "blender_model" : model obj file
- "image" : bmp and jpg file

File explain:

- makefile : my rule file to compile program -> output is run.o -> it work in my friend computer also (Linux only)
- main.cpp : main file of my project

- render_object.cpp : old main file of final project (openGL 2.0 orientation)
- simpleFrgShd.glsl, simpleVrtShd.glsl : fragment shader and vertex shader code.

Other

- If you want to add another bmp file not in my image folder, make sure it is "bmp 24bit color in format RBG" => this is very important for my poor implemented loader function.
- My objloader support only wareforn obj with f (face value) is positive and has 3 vertice in 1 f.