Student ID: 101062319
Name: 巫承威

Homework 4 - Report

Texture Mapping

• How to operate my program?

As the Figure 1. shown, there are several new header files and source files in my project, please make sure they are all under the project folder.

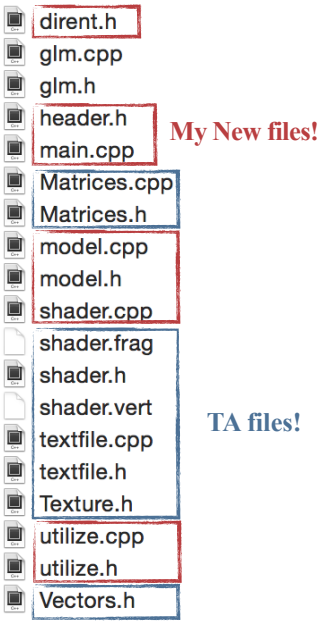

Figure 1.
Header Files and Source Files

Figure 2.
ScreenShot of Microsoft Visual Studio 2013

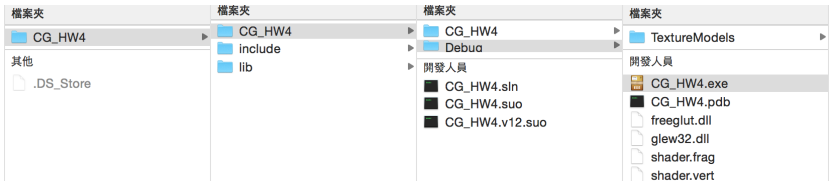

Figure 3.
Path of the "CG_HW4.exe"

All the user need to do to run my program is just open the .sln poject file via Microsoft Visual Studio 2013 (higher version may be fine). Then click the buttons "開始偵錯" or "啟動但不偵錯", and the execution window will appear. Or just execute the "CG_HW4.exe" as the Figure 3. shown. Like the example that TAs provided, user can type some button to control the display window and the object on it. These are the commands(note that I also allow the lowercase case letter to control the model as the uppercase letter does):

h / H - Show Help Menu
c / C - Clear Console
z Z ← / x X → - Switch Previous / Next model
Space - Trigger Wireframe / Fill Mode
r / R - Switch on / off Model Rotation
e / E - Start Eye Mode
p / P - Parallel / Perspective Projection

1 - Switch on / off Directional Light
2 - Switch on / off Point Light
3 - Switch on / off Spot Light
v / V - Switch Gouraud / Phong Shading
5 - Start Move Directional Light
6 - Start Move Point Light
7 - Start Move Spot Light
8 / 9 - Start Change Spot Light Cone / Exp

t / T - Trigger Texture Mapping
w / W - Switch TEXTURE_WRAP between GL_REPEAT / GL_CLAMP_TO_EDGE
M - Switch Mag_Filter between GL_LINEAR / GL_NEAREST
m - Switch Min_Filter between GL_LINEAR / GL_NEAREST

• Implementation and problems I met

At first, I uncomment several lines of codes provided by TA and add the parameters to the corresponding functions. And then based on my previous homework's framework, I add and insert the above functions into the correct location into my homework's architecture. Finally, I add some variables to control whether the final result contains the texture mapping or not, and send them to the vertex shader and fragment shader. Also, I replace some parameters sent to the function "glTexParameteri()" with my own variables which can be controlled by the user. Since I finished the per-pixel lighting in the previous homework, I only need to multiply the lighting values to the texture values and then get the correct final results.

Another interesting thing is that I remove my manipulating matrices this time. Due to the requirement in this homework, I need to translate, scale and rotate the model via the mouse. But after modifying the parameters sent to my manipulating matrices, the result isn't correct. So I try my best to find the solution to this problem. Fortunately, I try to translate, scale and rotate the model directly which means that I operate these matrices via the functions provided in "Matrices.cpp", and finally, I solve this tricky problem via the above steps.

• Other efforts I have done

Owing to the unfriendly codes which combine all the segment of the program into only one file, and also not so readable for the programmer, I decide to make it looked like the object-oriented program. Nevertheless, this idea cost me almost one afternoon to adjust them…. To finish this task, I need to figure out the operation between the model and the whole program and also how to manipulate each class. I think this is the most difficult part for transforming the original code to object-oriented type.

Another part I have done is to use the structure "DIR" and "dirent" which are already mentioned in above paragraph. It's more powerful and flexible than brute-force method because it can traverse all the files in the given folder's all subfolders. Also, I define new macro called "abs" to

help get the length of the model. To avoid the re-define error of so many new header files, I add the "#pragma once" to make compiler correctly include the header file only once when they need.

What's more, I add the feature that my model can be scaled by adjusting the GLUT window. It means that if I change the GL window, my object model will also change its size to fit the size of window.

• Screenshots

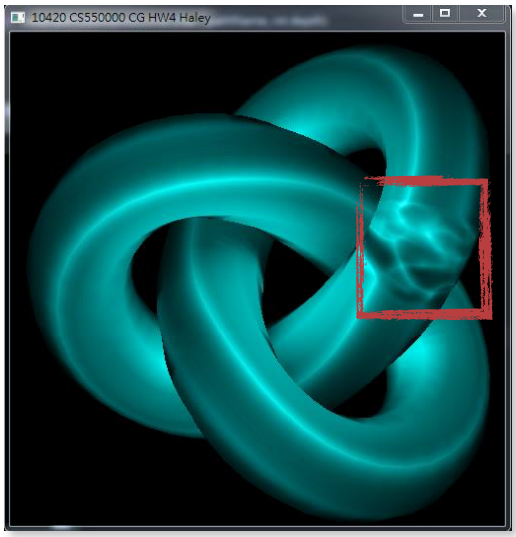

Figure 4.
Model TexturedKnot.obj (Clamp to Edge)

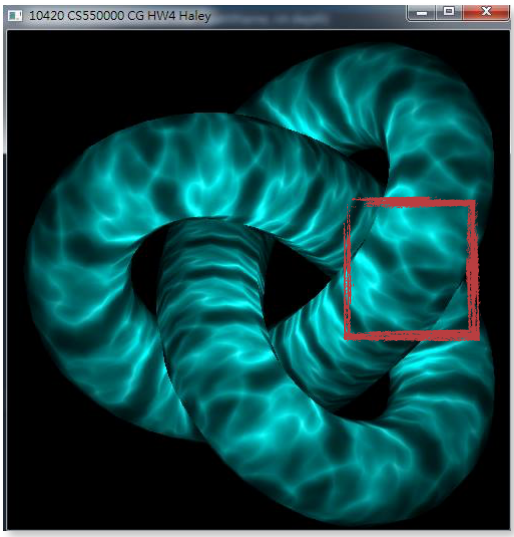

Figure 5.
Model TexturedKnot.obj (Repeat)

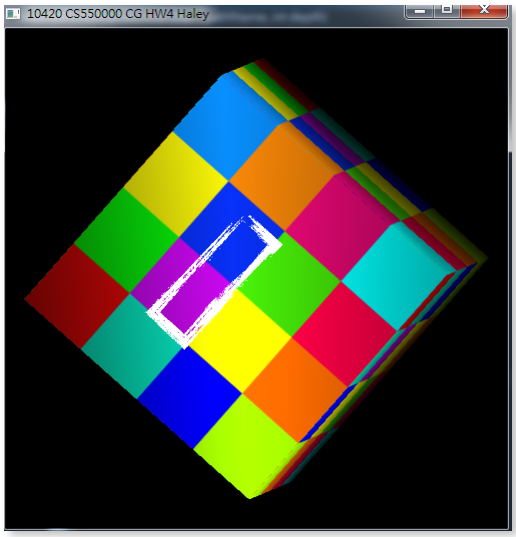

Figure 6.
Model Checker.obj (Mag_Filter_Linear)

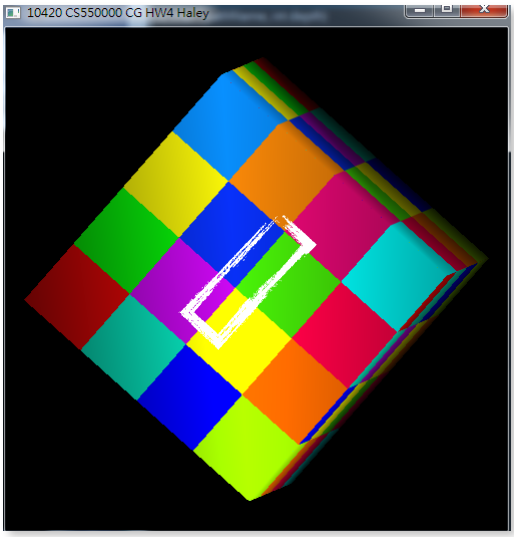

Figure 7.
Model Checker.obj (Mag_Filter_Nearest)

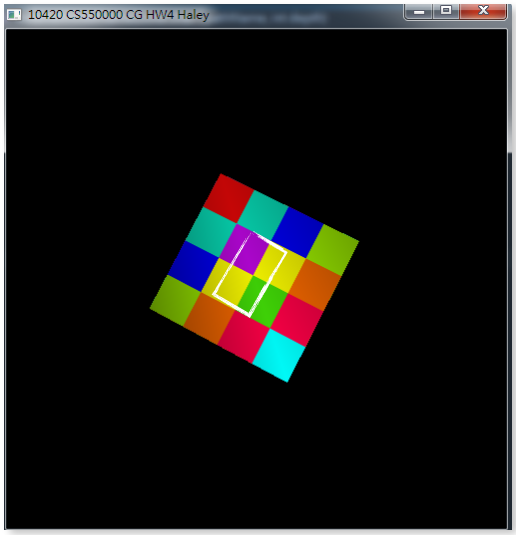

Figure 8.
Model Checker.obj (Min_Filter_Linear)

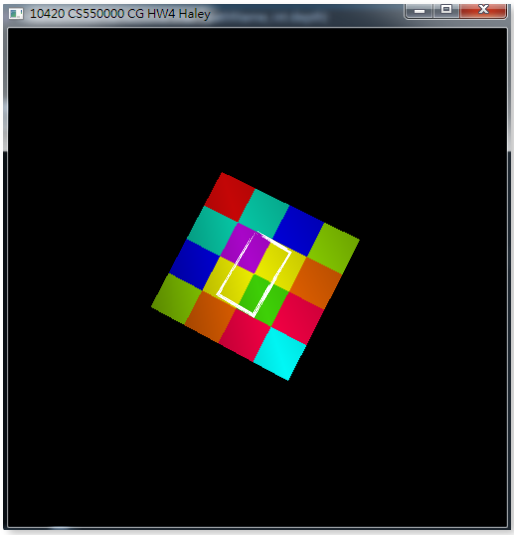

Figure 9.
Model Checker.obj (Min_Filter_Nearest)

- **Screenshots (Optional Model)**

Figure 10.
Model Leblanc.obj