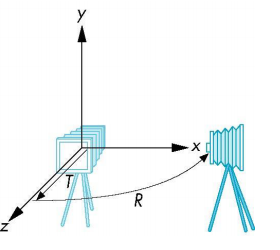
Part A

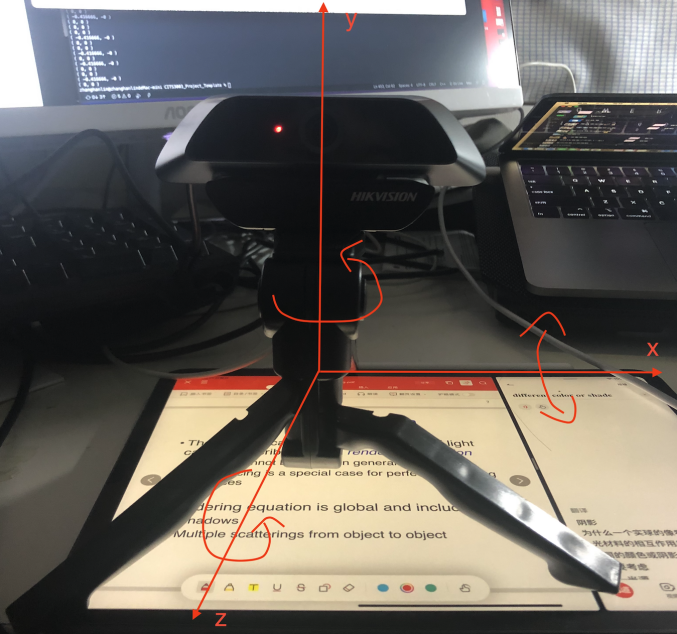
For part A, as we can see in the question request, it said we need to move and rotate the camera follow the action of the left and middle button on the mouse.

Looking through the lecture 14 p8, we get this picture:



So it means we need to rotate the camera on the x and y axis.

Here are some pictures to make more sense:



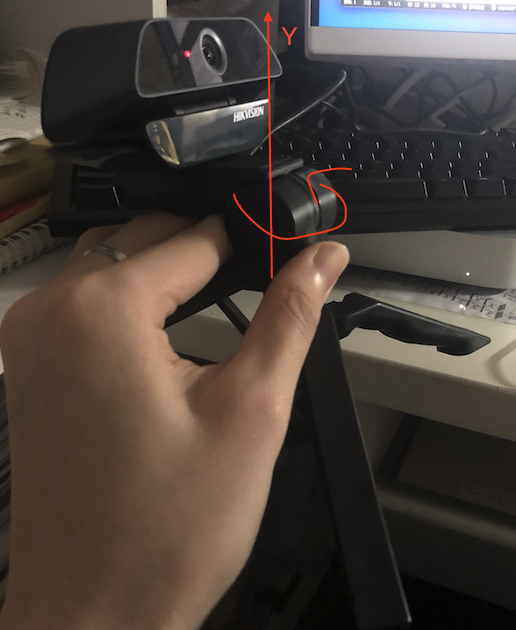
Rotate X:

The camera rotate on x-axis if you compare with the origin picture



Rotate Y:

The camera rotate on y-axis if you compare with the origin picture



Rotate Z:

The camera rotate on z-axis if you compare with the origin picture





Also, from the lecture 14, we learn a formula at p8:

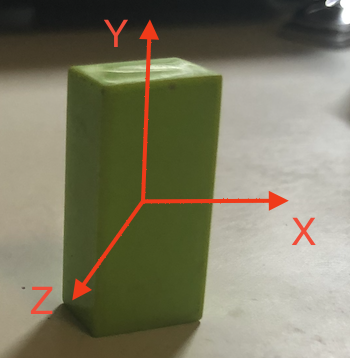
Since we need to rotate an x-axis and y-axis, so we get the code below:



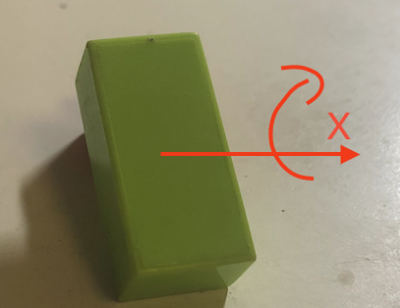
Part B

Similar to part A, but this time we need to change the angle of the object (rotate the object).

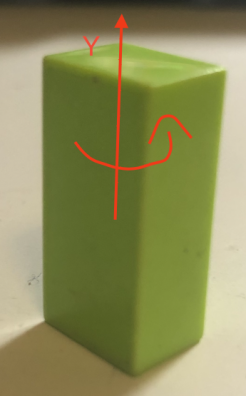
Original object:



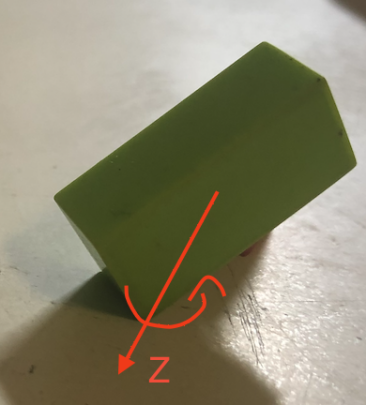
Rotate X:



Rotate Y:



Rotate Z:



So in this part, we need to consider all of the situations, rotate x-axis, y-axis, z-axis.

As we know, inside of the structure “SceneObject”, it contains a float list named angles which contains the rotations around X,Y and Z axes.

Finally, based on the last part and the information we get from this part, we get code below:

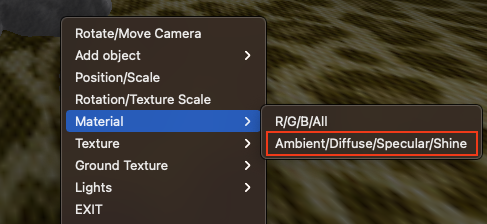


Later, we found it is necessary to change the fragment shader, change 2.0 to textScale, to let the textScale change with the camera.



Part C

Same with what we know from the question request, it ask us to modify the function below:

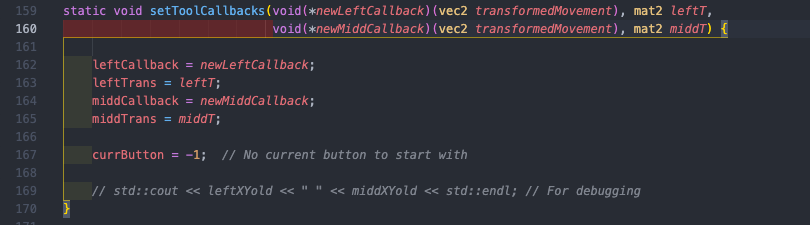


So, we need to go to the function “makeMenu” first. According to this, we know ID for “R/G/B/ALL” is 10 and ID for “Ambient/Diffuse/Specular/Shine” is 20 (name of this function already been changed to match the demo video). Then we go back to the function “materialMenu”.

First, we need to follow the code for “R/G/B/ALL” part, so that is why we add a else if statement with condition “id == 20”, it is for “Ambient/Diffuse/Specular/Shine” part.

Set the toolObj to be the current object, next we need to add a “setToolCallbacks” function.

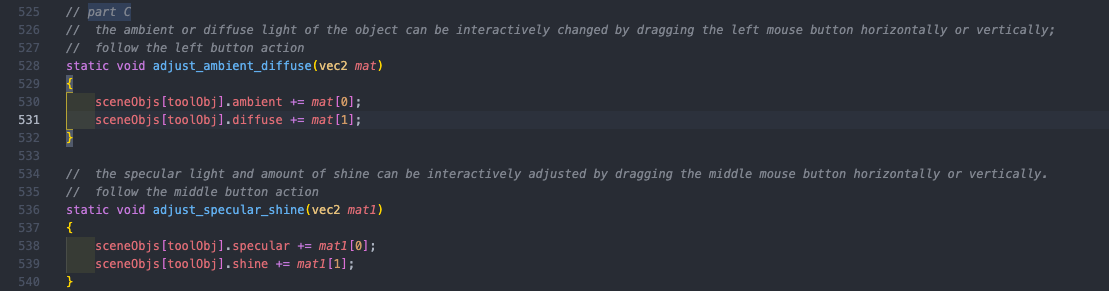
After checking the file “gnatidread.h”, we get this below from the file:

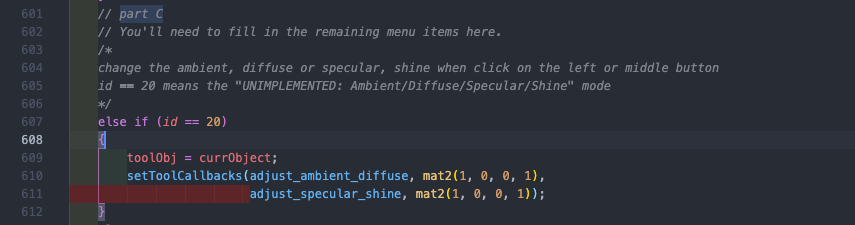


The func1 which call inside of “setToolCallbacks” follow the action of mouse left button and func2 follow the action of mouse middle button.

Now it is time to make func1 and fun2. func1 to adjust ambient and diffuse, func2 to adjust specular and shine.

Follow the both call functions of setToolCallbacks: “adjustRedGreen” and “adjustBlueBrightness” for part “R/G/B/ALL” in function materialMenu, we can know how to build the function “adjust\_ambient\_diffuse” and “adjust\_specular\_shine”. Finally we get the code below:





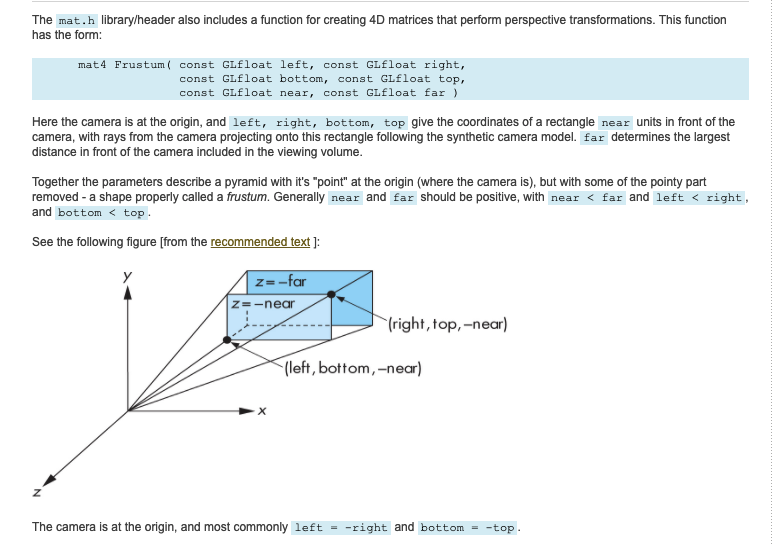
Change in function “makeMenu” is only change the function name of the “UNDEFINED: Ambient/Diffuse/Specular/Shine”, so no screenshot post here.

Part D

Just makes the near distance smaller, that is all.

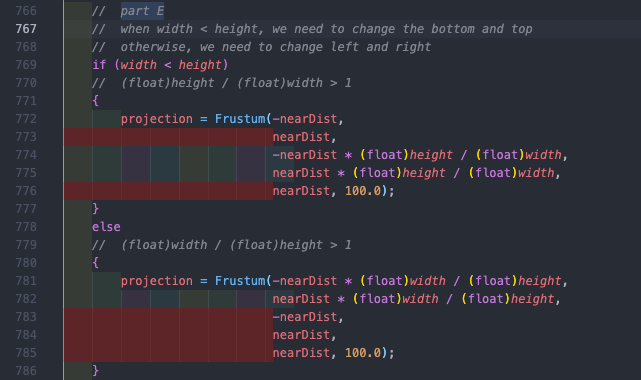
Part E

We did the similar thing in lab 5 (week 6), so we now has this image:



Then we know we need if-else statement for width < height and width >= height. For width < height, we need to change bottom and top value, otherwise we need to change left and right value.

Based on this, we get:



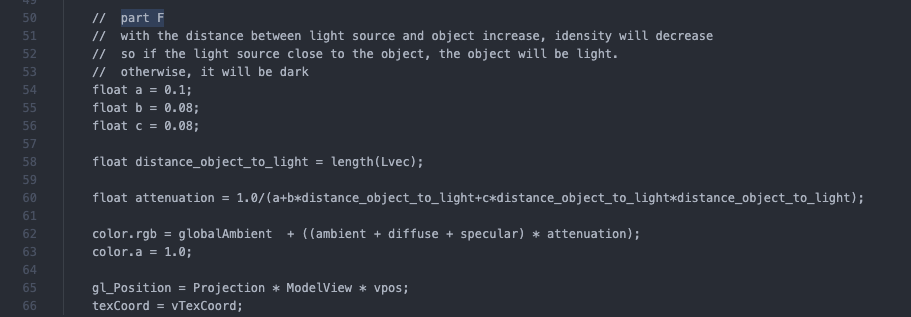
Part F

For this one, we need to modify the vertex shader so that the light reduces with distance. To achieve this, we go back to lecture 15 p15.

Now we get the formula below (where *d*𝑑 is the distance from light source):



After adjust the number of a, b, c to make the result better, we get code below:



With the distance between light source and object increase, idensity will decrease. So, if the light source close to the object, the object will be light. Otherwise, it will be dark.

We found a, b, c means constant, linear and quadratic. Constant should be 1.0 normally and Linear should larger than quadratic. But we are not too sure why for a = 1.0, the lightness of the light will be really low. To make it lighter, we can only decrease a.

Part G

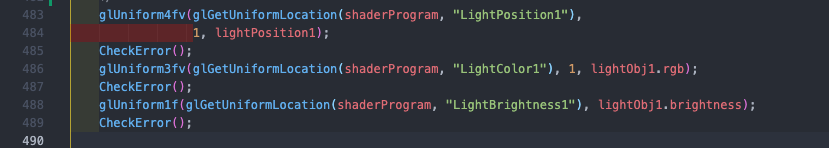
The lighting calculations in the fragment shader, so that the directions are calculated for individual fragments rather than for the vertices.

Nothing need to change in .cpp file, only need to change vertex shader and fragment shader. Just copy and paste the part from vertex shader to fragment shader and pass the variables from vertex shader to fragment shader.

Part H

Not too sure do we need to change the sceneObjs[nObjects].rgb[0],[1],[2] to be 1.0 inside of the addObject( ) function, since the white light color is 1.0, 1.0, 1.0 not 0.7, 0.7, 0.7. But in the request it said “Generally specular highlights tend towards white”, maybe no need?

For this part, we need to get the color and brightness of the light, same with how we get the light position:

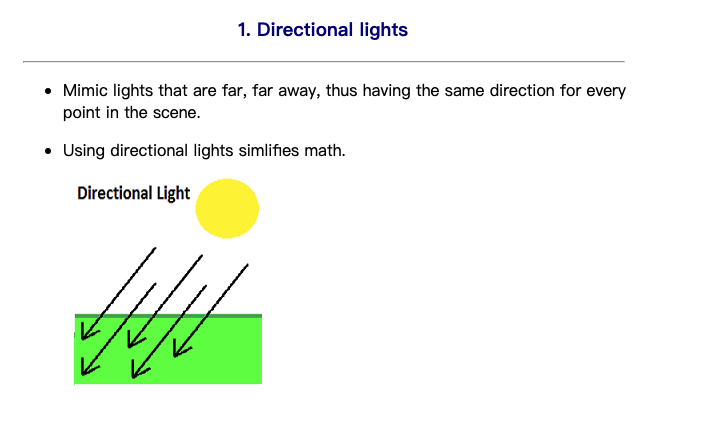


After doing these, we need to modify the fragment shader.

Part I

The second light should be directional, i.e., the direction that the light hits each surface is constant and independent of the position of the surface. So, we only need to consider the camera rotation, no transmission required.

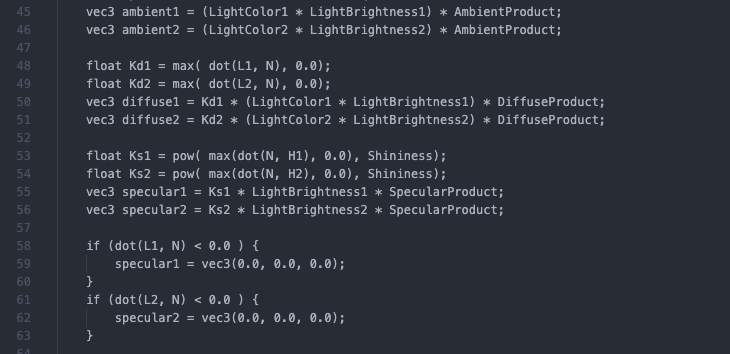
Then, the light reduction is not suitable for light 2.



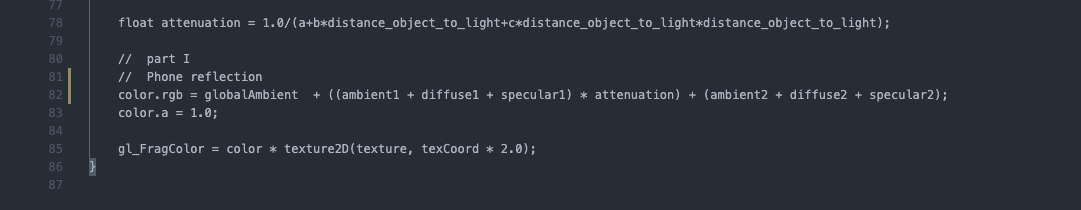
But actually it will has influence.

For ambient and diffuse, both of them are related to LightColor and LightBrightness. No relation between specular and LightColor, so we only need to multiple LightBrightness.

Change in fragment shader:

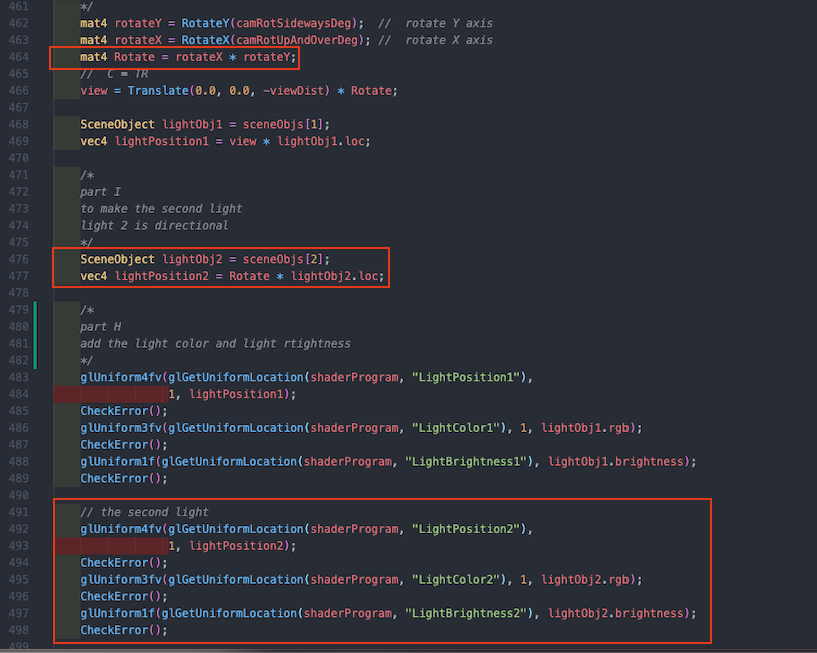


When we were working on the fragment shader color part, we get confused, since some people compute specular lighting separately with ambient and diffuse, other people compute specular lighting with ambient and diffuse. To solve this, we tried to search some source, see in the reference.



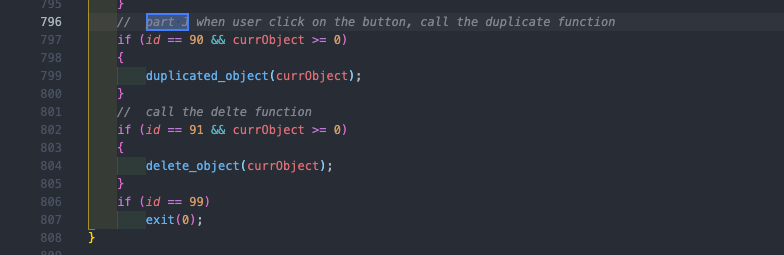
Change in .cpp file:

Add a new light



Part J

For the duplicated, no need to change the vertex or fragment shader, just simply add a new function in .cpp file and add a menu into the menu box.

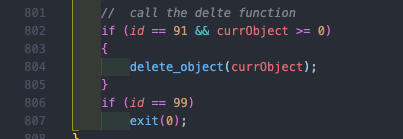




Due to we plan to add a spotlight in later part, so we set if statement for nObjects == 4, because if nObjects == 4, it means we only get three lights and a ground, no object can be duplicated.

After set up a new object at the same place with origin one, we need to redisplay the window.

To delete the object, it is a little bit similar with duplicated, no need to change the vertex or fragment shader, just simply add a new function in .cpp file and add a menu into the menu box.

If nObjects == 4, means no object to delete, it will return and quit delete function.

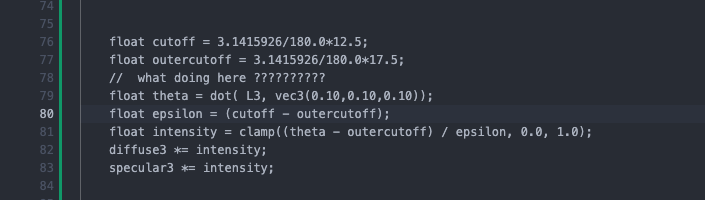
Also, redisplay the window after one remove an object.

For the spotlight, we add new menu to move and change “R/G/B/All Light”, just like what we done in the previous parts. Similar with how we add a new light (light 2) into the window, we add a new spotlight by editing .cpp file. After doing so, we need to modify the fragment shader to calculate the light of spotlight. As we know, the spotlight looks like a cone. We tried to search some resources related with this part, but no much help. Most of them use another lib which called glm, but actually we get some error when we try to modify the makefile and makefilelist, so we decided to use another way.

Looking through the example code from the website, we found they only use glm to compute the convert between radians and angles, so we decide to compute this by formula.

radians = π / 180.0 \* angles

Finally, we contributed all the things and found something wrong inside. But due to the shallow understand of others’ source code, we do not know how to fix that. The issue might inside of the fragment shader, the code below:



After checking with the open source code from the internet, we think now it is should be better now.

Also thanks for tutor’s help, we use the buid in c++ functions to avoid import the glm lib and get some idea to fix the bug.

Reflection

We found part A - part F is not that difficult with what I thought. When we first time to look through the files, we even do not know what we need to do. But with in-depth study, we found most of things we can not understand become clearly. But for the parts after part F, they become harder, especially for part J.

Finally, we know we still have much things need to learn since we spend more time than we expect on this project. Also, this project give us much fun and we get delighted after we finish a part or fix tricky bugs.

Reference

1. How glGetUniformLocation works? (For part H)

<https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glGetUniformLocation.xhtml>

2. GLSL Programming/GLUT/Specular Highlights (For part G)

<https://en.wikibooks.org/wiki/GLSL_Programming/GLUT/Specular_Highlights>

1. RGB color examples (For part G)

<https://www.tug.org/pracjourn/2007-4/walden/color.pdf>

1. Diffuse and Specular Light (For part I)

<http://www.c-jump.com/bcc/common/Talk3/OpenGL/Wk06_light/Wk06_light.html#W01_0010_directional_lights>

1. Phong reflection model (For part I)

<https://en.wikipedia.org/wiki/Phong_reflection_model>

1. Ambient lighting (For part I)

<http://ogldev.atspace.co.uk/www/tutorial17/tutorial17.html>

1. Diffuse Lighting (For part I)

<http://ogldev.atspace.co.uk/www/tutorial18/tutorial18.html>

1. Specular Lighting (For part I)

<http://ogldev.atspace.co.uk/www/tutorial19/tutorial19.html>

1. Lighting (For part I)

<https://learnopengl.com/Lighting/Basic-Lighting>

10. Phong lighting: add specular lighting separately or with ambient and diffuse?

<https://stackoverflow.com/questions/48160165/phong-lighting-add-specular-lighting-separately-or-with-ambient-and-diffuse>

1. Use glutPostRedisplay( ) (For part J)

<https://www.opengl.org/resources/libraries/glut/spec3/spec3.html>

12. Why directional lights not related with light reduction (For part I)

<https://www.youtube.com/watch?v=gFZqzVQrw84>

1. What are the coefficients in attenuation formula? (For part F)

<http://wiki.ogre3d.org/tiki-index.php?page=-Point+Light+Attenuation>

1. Make a spotlight (For part J)

<https://www.youtube.com/watch?v=tmCOMzAA4rc>

1. Spotlight introduction (For part J)

<https://github.com/SonarSystems/Modern-OpenGL-Tutorials>

1. 16. Spotlight source code (For part J)

<https://learnopengl.com/code_viewer_gh.php?code=src/2.lighting/5.3.light_casters_spot/light_casters_spot.cpp>