姓名：成文瑄 學號：0716004

先由phong shading的部分開始，首先是要在main.cpp中把shader建起來和連起來（phong shading的助教已經寫好了，我加上了toon shading的edge effect的）。

    GLuint vert1 = createShader("Shaders/Phongshading.vert", "vertex");

    GLuint frag1 = createShader("Shaders/Phongshading.frag", "fragment");

    Phongprogram = createProgram(vert1, frag1);

    //// TODO: ////

    // create the shaders and programs you need.

    GLuint vert2 = createShader("Shaders/Toonshading.vert", "vertex");

    GLuint frag2 = createShader("Shaders/Toonshading.frag", "fragment");

    Toonprogram = createProgram(vert2, frag2);

    GLuint vert3 = createShader("Shaders/Edgeeffects.vert", "vertex");

    GLuint frag3 = createShader("Shaders/Edgeeffects.frag", "fragment");

    Edgeprogram = createProgram(vert3, frag3);

接著就是要寫vertex shader和fragment shader。

首先在vertex shader中需要關於頂點的位置、normal資訊，原本助教給的沒有把normal傳進來，所以我把它加上了。

layout(location = 0) in vec3 in\_position;

layout(location = 1) in vec3 in\_normal;

layout(location = 2) in vec2 texcoord;

再來是關於output的變數：

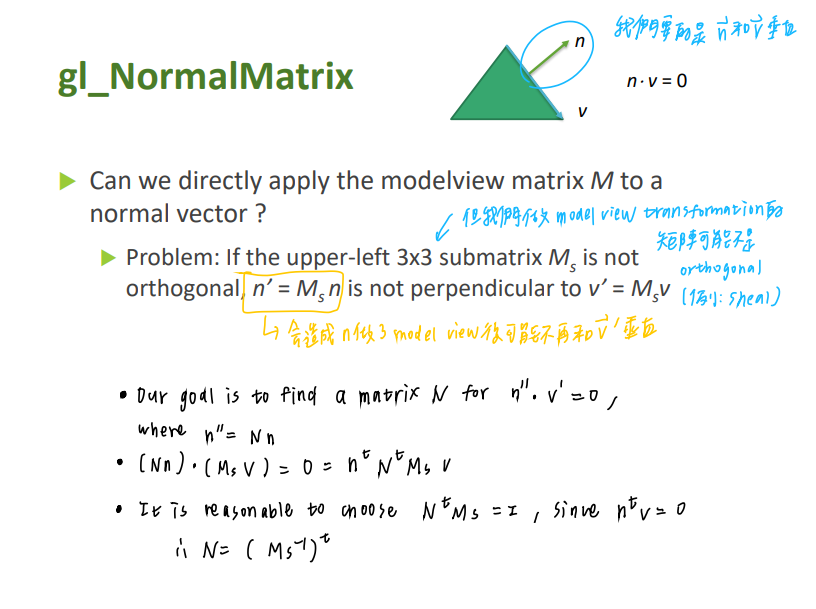
  gl\_Position = P \* V \* M \* vec4(in\_position, 1.0);

  worldPos = (M \* vec4(in\_position, 1.0)).xyz;

  normal = normalize((transpose(inverse(M)) \* vec4(in\_normal, 1.0)).xyz);

  uv= texcoord;

uv是要拿的texture的座標、gl\_position是原本的model在經過model view、projection等等的transformation後，在“螢幕”上要顯示的位置。但在fragment shader中，是在world coordinate做shading的，所以這裡的另一個output worldPos就是model在world coordinate的位置（所以只要原來的in\_position做一個M的transform就好），然後output的normal是因為model經過transform了所以原本的in\_normal應該也會改變（但是in\_normal不能像in\_position一樣直接乘一個M矩陣，因為M矩陣可能不是orthogonal，會造成乘完之後的nomal和worldPos可能不垂直，由上課的ppt中可證明出in\_normal要乘的矩陣是(M^-1)^t，最外面的normalize是為了要把乘出來的normal向量變成單位向量）。



再來是fragment shader的部分：

在fragment shader裡首先需要的就是phong shading的各個參數（包括反射率Kads、光強度Lads、gloss、法向量N、視角V、光的入射向量L、反射向量R等等），反射率、光強度、gloss是固定的所以從main.cpp用uniform傳進來，剩下的那些向量是每個頂點不一樣的所以要自己算。

Fragment shader

uniform sampler2D texture;

uniform vec3 WorldLightPos;

uniform vec3 WorldCamPos;

uniform vec3 Ka;

uniform vec3 Kd;

uniform vec3 Ks;

uniform vec3 La;

uniform vec3 Ld;

uniform vec3 Ls;

uniform float gloss;

in vec2 uv;

in vec3 normal;

in vec3 worldPos;

main.cpp

    GLuint PosID = glGetUniformLocation(program, "WorldLightPos");

    glUniform3fv(PosID, 1, &WorldLightPos[0]);

    PosID = glGetUniformLocation(program, "WorldCamPos");

    glUniform3fv(PosID, 1, &WorldCamPos[0]);

    glm::vec3 Ka = glm::vec3(1, 1, 1);

    glm::vec3 Kd = glm::vec3(1, 1, 1);

    glm::vec3 Ks = glm::vec3(1, 1, 1);

    glm::vec3 La = glm::vec3(0.2, 0.2, 0.2);

    glm::vec3 Ld = glm::vec3(0.8, 0.8, 0.8);

    glm::vec3 Ls = glm::vec3(0.5, 0.5, 0.5);

    float gloss = 25;

    GLuint ParameterID = glGetUniformLocation(program, "Ka");

    glUniform3fv(ParameterID, 1, &Ka[0]);

    ParameterID = glGetUniformLocation(program, "Kd");

    glUniform3fv(ParameterID, 1, &Kd[0]);

    ParameterID = glGetUniformLocation(program, "Ks");

    glUniform3fv(ParameterID, 1, &Ks[0]);

    ParameterID = glGetUniformLocation(program, "La");

    glUniform3fv(ParameterID, 1, &La[0]);

    ParameterID = glGetUniformLocation(program, "Ld");

    glUniform3fv(ParameterID, 1, &Ld[0]);

    ParameterID = glGetUniformLocation(program, "Ls");

    glUniform3fv(ParameterID, 1, &Ls[0]);

    ParameterID = glGetUniformLocation(program, "gloss");

    glUniform1f(ParameterID, gloss);

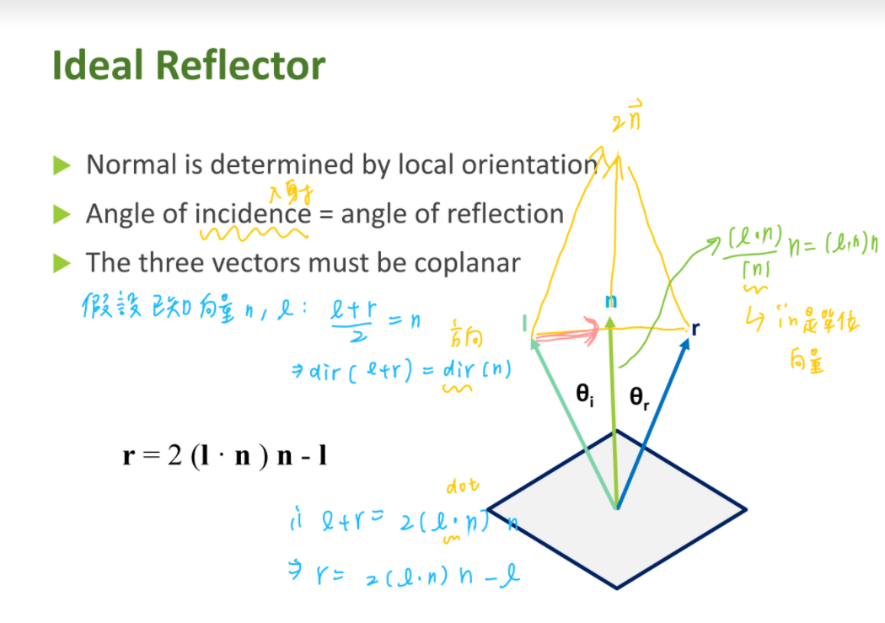
N就是剛才在vertex shader中算好的，L的話我用uniform傳進來光的位置再把它和物體位置相減後做normalize可以得到，V的話我用uniform傳進來相機的位置再把它和物體位置相減後做normalize可以也可以得到，R就是利用算出來的L和N再由ppt上的公式可以算出來（這些向量都是單位向量）。

    vec3 L = normalize(WorldLightPos - worldPos);

    vec3 V = normalize(WorldCamPos - worldPos);

    vec3 N = normal;

    vec3 R = 2 \* dot( L, N) \* N - L;



再來做phong shading時顏色是由ambient+diffuse+specular。

ambient是Ka\*La，diffuse是Kd\*Id\*dot(L,N)（這裡的內積是因為diffuse和光的入射角度有關，關係是cos(theta)，因為L和N都是單位向量所以內積結果就是cos(theta)），specular則是Ks\*Ls\*(dot(V,R))^a（這裡的內積是因為specular和人看的角度有關，人看的角度約接近完美反射的地方應該越亮，關係也是cos(theta)，alpha則是代表specular的衰減係數，看他亮的範圍可以多大），output的color就是把三者加起來，多加的那一維是透明度。

    vec3 object\_color = texture2D(texture, uv).rgb;

    vec3 ambient = La \* Ka \* object\_color;

    vec3 diffuse = Ld \* Kd \* object\_color \* max( dot(L, N), 0);

    vec3 specular = Ls \* Ks \* pow( dot( V, R), gloss);

    color = vec4( ambient + diffuse + specular, 1);

再來是toon shading的部分：

vertex shader的部分和phong shading一樣所以就不再說明，所以直接說明fragment shader的部分。toon shading是用diffuse的概念，但不像phong shading那邊是每個位置的cos都不一樣，這裡是cos在一個範圍內的都定成同一個值（這裡叫intensity)，所以這裡就把原本phong shading中diffuse的cos（dot(L,N)）叫做level然後根據它們的大小給他們新的intensity值，然後和算diffuse一樣把它們乘起來。

    float intensity;

    float level = dot( L, N);

   if( level > 0.95) intensity = 1;

   else if( level > 0.75) intensity = 0.8;

   else if( level > 0.50) intensity = 0.6;

   else if( level > 0.25) intensity = 0.4;

   else intensity = 0.2;

    vec3 object\_color = texture2D(texture, uv).rgb;

    color = vec4(Kd \* object\_color \* intensity, 1);

最後是edge effect：

vertex shader 也是和前面一樣，所以直接說明fragment shader的部分。因為edge effect是只想要畫出邊(dot(N,V)~0），用以往直接乘的話會乘到0是畫不出來的，所以我用1減掉這個dot，這樣在接近邊的地方值就會接近1，而其他面的部分值就會是0.多，因為現在只想畫邊，所以需要邊的地方趨近1其他地方趨近0，所以我把剛才的1-dot去取了8次方，因為面的地方是0.多，乘很多次就會趨近0，邊的地方因為幾乎是1，乘了很多次還是1，再把它乘上我想要的邊的顏色（是從外面用uniform傳進來的）就可以了。

    vec3 L = normalize(WorldLightPos - worldPos);

    vec3 V = normalize(WorldCamPos - worldPos);

    vec3 N = normal;

    vec3 R = 2 \* dot( L, N) \* N - L;

    color = vec4( object\_color \* pow(1- dot(N, V), 8), 1);

object\_color:

fragment shader

uniform vec3 object\_color;

main.cpp

    glm::vec3 object\_color = glm::vec3(0, 0, 1);

    ParameterID = glGetUniformLocation(program, "object\_color");

    glUniform3fv(ParameterID, 1, &object\_color[0]);

最後就是在按鍵的那邊指定program是不同的shading方式。

    case '1':

    {

        //// TODO: ////

        // switch to the program which you want to use

        program = Phongprogram;

        break;

    }

    case '2':

    {

        //// TODO: ////

        // switch to the program which you want to use

        program = Toonprogram;

        break;

    }

    case '3':

    {

        //// TODO: ////

        // switch to the program which you want to use

        program = Edgeprogram;

        break;

    }

遇到的問題是一開始不太懂spec上面寫的那個座標轉換的東西，後來問了別人，看了上課的ppt之後就懂了！