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。

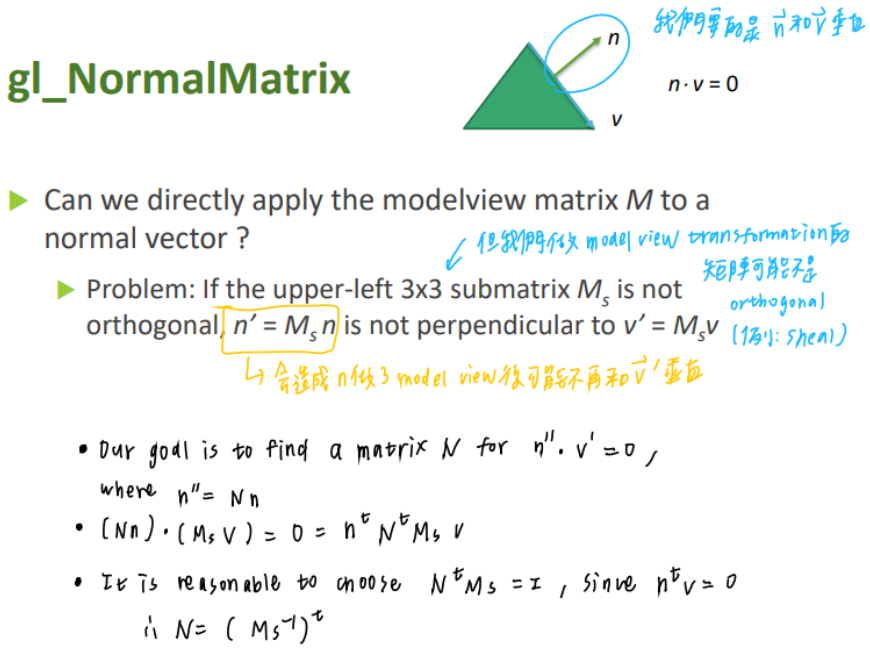
```
layout(location = 0) in vec3 in_position;
layout(location = 1) in vec3 in_normal;
layout(location = 2) in vec2 texcoord;
```

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;
```

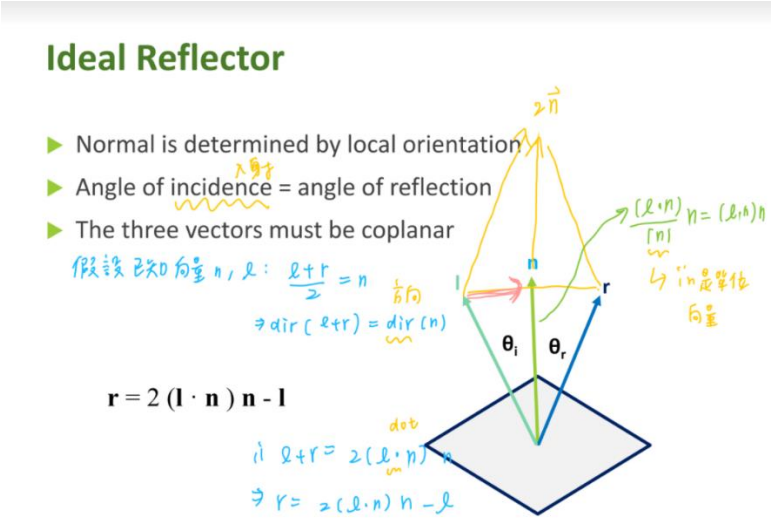
```
uniform float gloss;

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！