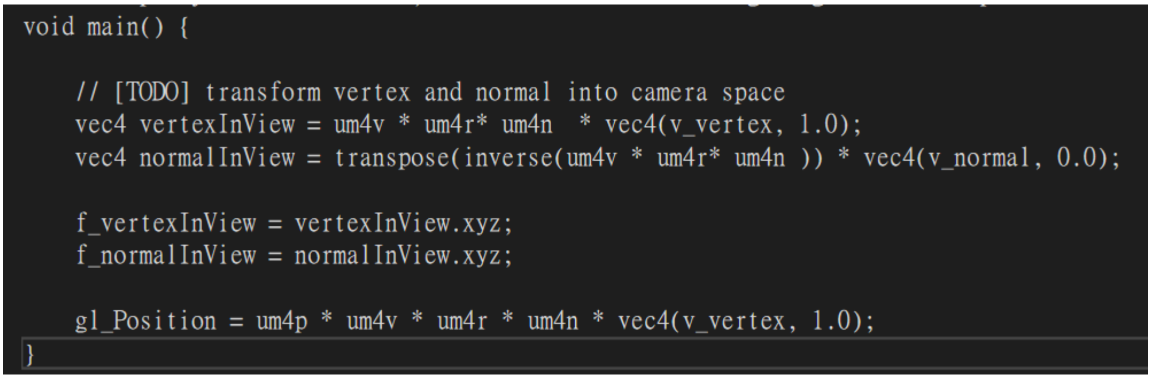
Computer Graphic HW2 Report 104062241 莫雅雯

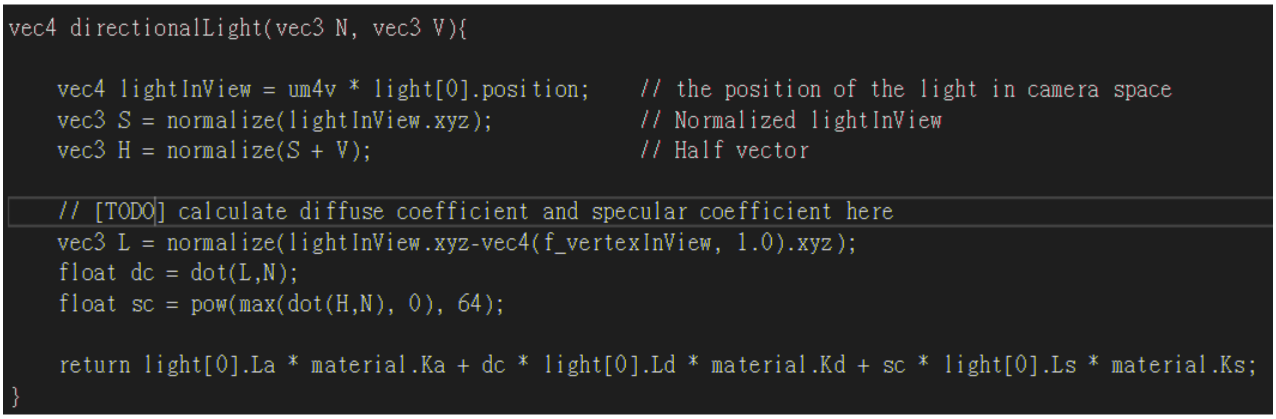
因為 spec 說 All the lighting calculations are performed in fragment shader，所以在 shader.vert 只要寫:



```
void main() {

    // [TODO] transform vertex and normal into camera space
    vec4 vertexInView = um4v * um4r* um4n  * vec4(v_vertex, 1.0);
    vec4 normalInView = transpose(inverse(um4v * um4r* um4n )) * vec4(v_normal, 0.0);

    f_vertexInView = vertexInView.xyz;
    f_normalInView = normalInView.xyz;

    gl_Position = um4p * um4v * um4r * um4n * vec4(v_vertex, 1.0);
}
```

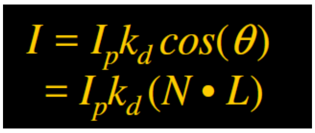
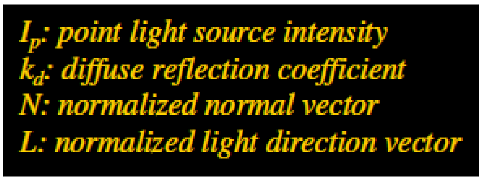
把 vertex in view 和 normal in view 傳到 shader.frag 即可。其他程式碼主要在 shader.frag 實作

一、Directional light



```
vec4 directionalLight(vec3 N, vec3 V){

    vec4 lightInView = um4v * light[0].position;    // the position of the light in camera space
    vec3 S = normalize(lightInView.xyz);            // Normalized lightInView
    vec3 H = normalize(S + V);                      // Half vector

    // [TODO] calculate diffuse coefficient and specular coefficient here
    vec3 L = normalize(lightInView.xyz-vec4(f_vertexInView, 1.0).xyz);
    float dc = dot(L,N);
    float sc = pow(max(dot(H,N), 0), 64);

    return light[0].La * material.Ka + dc * light[0].Ld * material.Kd + sc * light[0].Ls * material.Ks;
}
```

根據講義公式 ：Intensity=Ambient+Diffuse+Specular

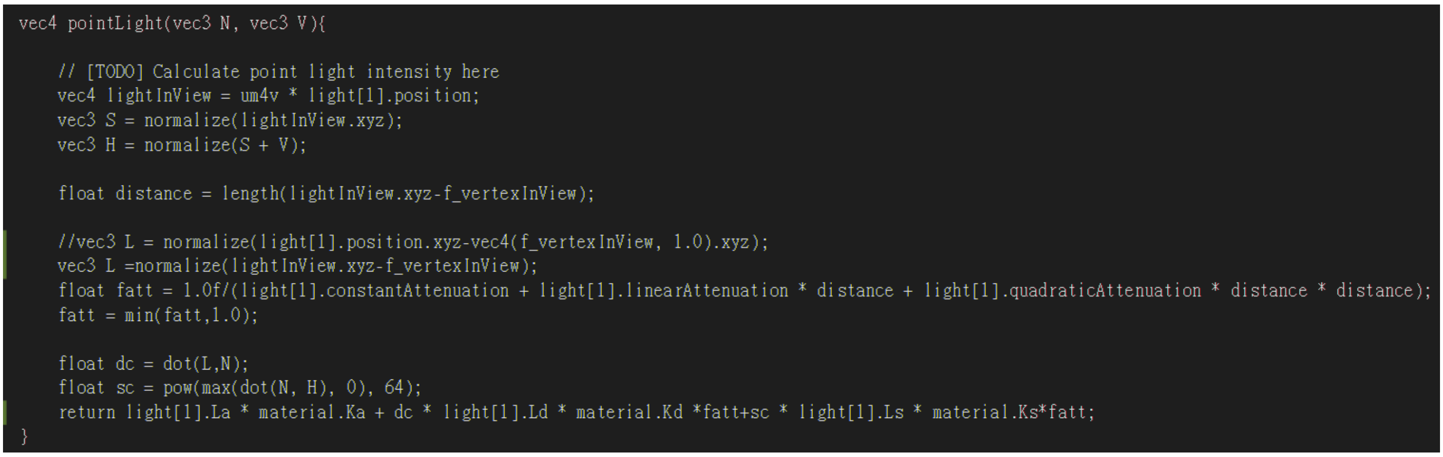
1.Ambient:	2.Diffuse:	3. Specular:
$I = I_a k_a$ I: resulting intensity I_a: ambient light intensity k_a: ambient reflection coefficient	 $I = I_p k_d cos(\theta)$ $= I_p k_d (N \bullet L)$  I_p: point light source intensity k_d: diffuse reflection coefficient N: normalized normal vector L: normalized light direction vector	$I = I_a k_a + \sum_{p=1}^{m} \boxed{f_p I_p} (k_d (N \cdot L_p) + \boxed{k_s (N \cdot H_p)^{n'}})$ 公式為紅框框部分。
	light direction vector(L)考慮到 light in view 和 vertex in view 將 2 者相減，再將其做 normalize，再跟 N 做 dot(內積) →得到 dc	light[0].Ls 就是 Ip(本題 light[0].Ld 跟 light[0].Ls 值都一樣。) 將 N 跟 H 做 dot(內積)再取 64 次方 →得到 sc

講義第 21 頁提到 Light Source Attenuation

$$I = I_a k_a + f_{att} I_p k_d (N \bullet L) \qquad f_{att} = \min(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1)$$

本題因為是 Directional light(Light source located at infinite far away)，故不用考慮 fatt。(fp 當成 1)

二、Point light



```
vec4 pointLight(vec3 N, vec3 V){

    // [TODO] Calculate point light intensity here
    vec4 lightInView = um4v * light[1].position;
    vec3 S = normalize(lightInView.xyz);
    vec3 H = normalize(S + V);

    float distance = length(lightInView.xyz-f_vertexInView);

    //vec3 L = normalize(light[1].position.xyz-vec4(f_vertexInView, 1.0).xyz);
    vec3 L =normalize(lightInView.xyz-f_vertexInView);
    float fatt = 1.0f/(light[1].constantAttenuation + light[1].linearAttenuation * distance + light[1].quadraticAttenuation * distance * distance);
    fatt = min(fatt,1.0);

    float dc = dot(L,N);
    float sc = pow(max(dot(N, H), 0), 64);
    return light[1].La * material.Ka + dc * light[1].Ld * material.Kd *fatt+sc * light[1].Ls * material.Ks*fatt;
}
```

根據講義公式 ：Intensity=Ambient+Diffuse+Specular

$$I = I_a k_a + \sum_{p=1}^{m} f_p I_p (k_d (N \cdot L_p) + k_s (N \cdot H_p)^{n'})$$

1.Ambient:這部分跟上一題一樣。

2.Diffuse:

講義第 21 頁提到 Light Source Attenuation

$$I = I_a k_a + f_{att} I_p k_d (N \bullet L) \qquad f_{att} = \min(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1)$$

(1)要先算出 fatt，再跟 diffuse 跟 specular 的部分相乘。

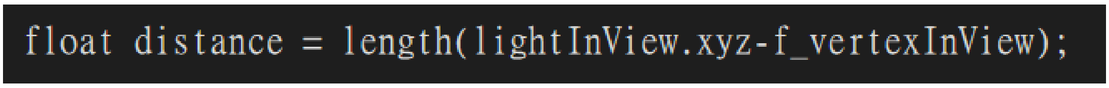
根據講義 45 頁:

$$\left(\frac{1}{k_c + k_l d + k_q d^2}\right)_l$$

可知道 c1 是 light[1].constantAttenuation，c2 是 light[1].linearAttenuation，c3 是 light[1].quadraticAttenuation。

(2)而其中 dL(distance)是 light source 跟 object 的距離

可以用這樣得到:



```
float distance = length(lightInView.xyz-f_vertexInView);
```

(3)帶入 fatt 公式得到 fatt

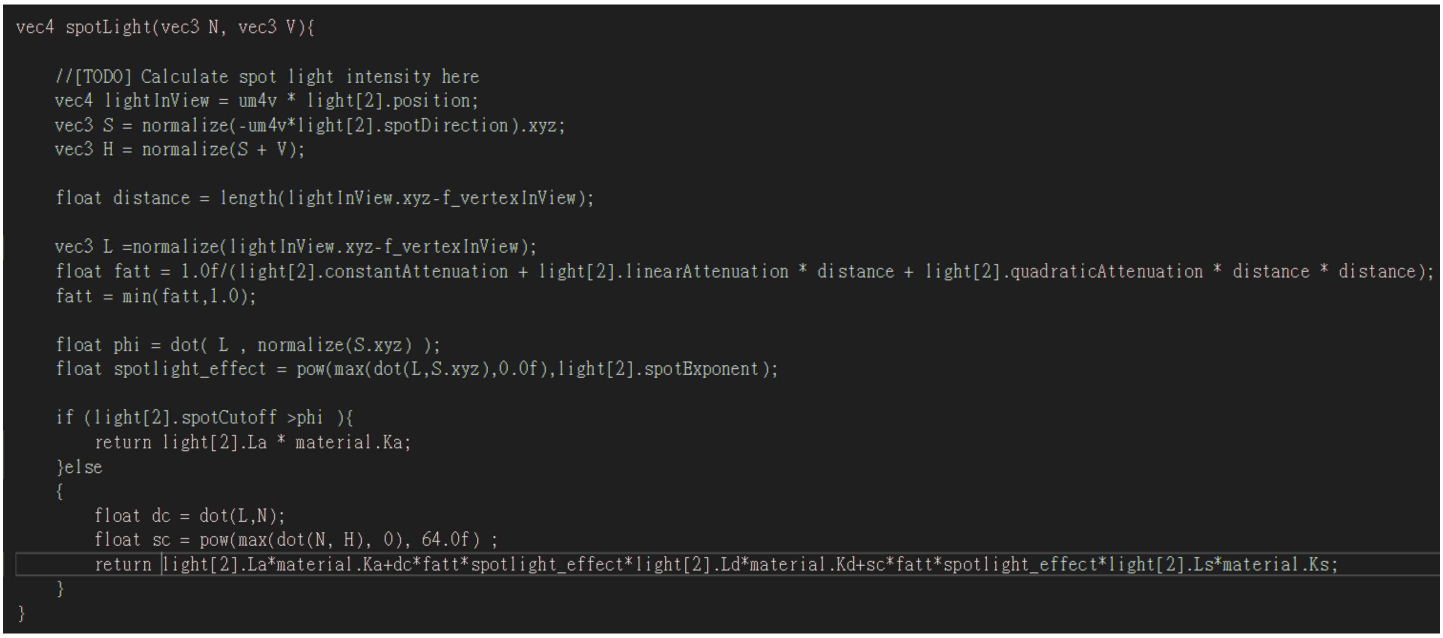
(4)dc 算法跟上題相同。

3. Specular:

$$I = I_a k_a + \sum_{p=1}^{m} \boxed{f_p I_p} (k_d (N \cdot L_p) + \boxed{k_s (N \cdot H_p)^{n'}})$$

跟上題差不多，只差在要多乘以 fatt

三、Spot light



```
vec4 spotLight(vec3 N, vec3 V){

    //[TODO] Calculate spot light intensity here
    vec4 lightInView = um4v * light[2].position;
    vec3 S = normalize(-um4v*light[2].spotDirection).xyz;
    vec3 H = normalize(S + V);

    float distance = length(lightInView.xyz-f_vertexInView);

    vec3 L =normalize(lightInView.xyz-f_vertexInView);
    float fatt = 1.0f/(light[2].constantAttenuation + light[2].linearAttenuation * distance + light[2].quadraticAttenuation * distance * distance);
    fatt = min(fatt,1.0);

    float phi = dot( L , normalize(S.xyz) );
    float spotlight_effect = pow(max(dot(L,S.xyz),0.0f),light[2].spotExponent);

    if (light[2].spotCutoff >phi ){
        return light[2].La * material.Ka;
    }else
    {
        float dc = dot(L,N);
        float sc = pow(max(dot(N, H), 0), 64.0f) ;
        return light[2].La*material.Ka+dc*fatt*spotlight_effect*light[2].Ld*material.Kd+sc*fatt*spotlight_effect*light[2].Ls*material.Ks;
    }
}
```

根據講義公式 ：Intensity=Ambient+Diffuse+Specular

$$I = I_a k_a + \sum_{p=1}^{m} f_p I_p (k_d (N \cdot L_p) + k_s (N \cdot H_p)^{n'})$$

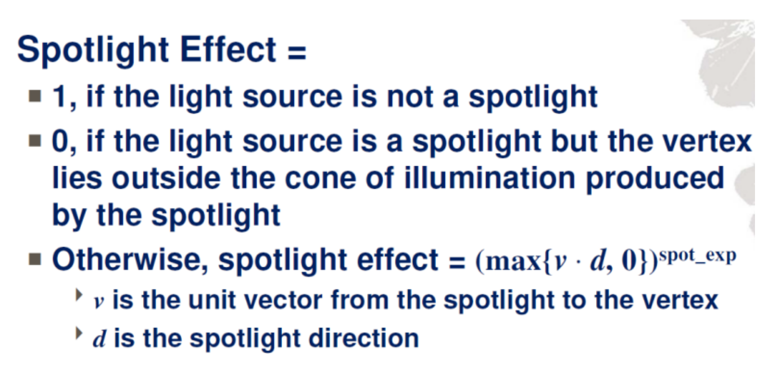
1.Ambient:這部分跟上一題一樣。

2.Diffuse:

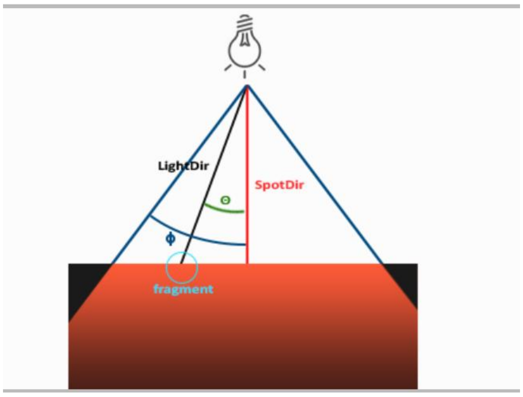
(1) fatt 部分同上一題。

(2)dc 算法跟上題相同。

(3) spotlight effect 部分參考講義 47

 **Spotlight Effect =** ■ **1, if the light source is not a spotlight** ■ **0, if the light source is a spotlight but the vertex lies outside the cone of illumination produced by the spotlight** ■ **Otherwise, spotlight effect = $(\max\{v \cdot d, 0\})^{spot\_exp}$** ‣ ***v* is the unit vector from the spotlight to the vertex** ‣ ***d* is the spotlight direction**	1.若不是 spot light: spotlight effect=1。所以前 2 題不用考慮 spotlight effect。 2.參見圖(一)，落在 phi(φ)之外設成 0。phi = dot(L , normalize(S.xyz)) 即 light direction vector(L)跟 spotDirection 做內積。 3.其他情形(代公式): spotlight_effect = pow(max(dot(L,S.xyz),0.0f),light[2].spotExponent)

圖(一)



(1)LightDir:fragement 指向 light source 的 vector

(2)SpotDir:spot light 所指向方向

(3)phi(φ):定義 spot light 半徑的切光角，每個落在這個角度之外的都會亮。

(4) θ : LightDir 向量和 SpotDir 向量之間的角度，應比 phi(φ)小才會在 spot light 內。

參考: https://learnopengl-cn.readthedocs.io/zh/latest/02%20Lighting/05%20Light%20casters/

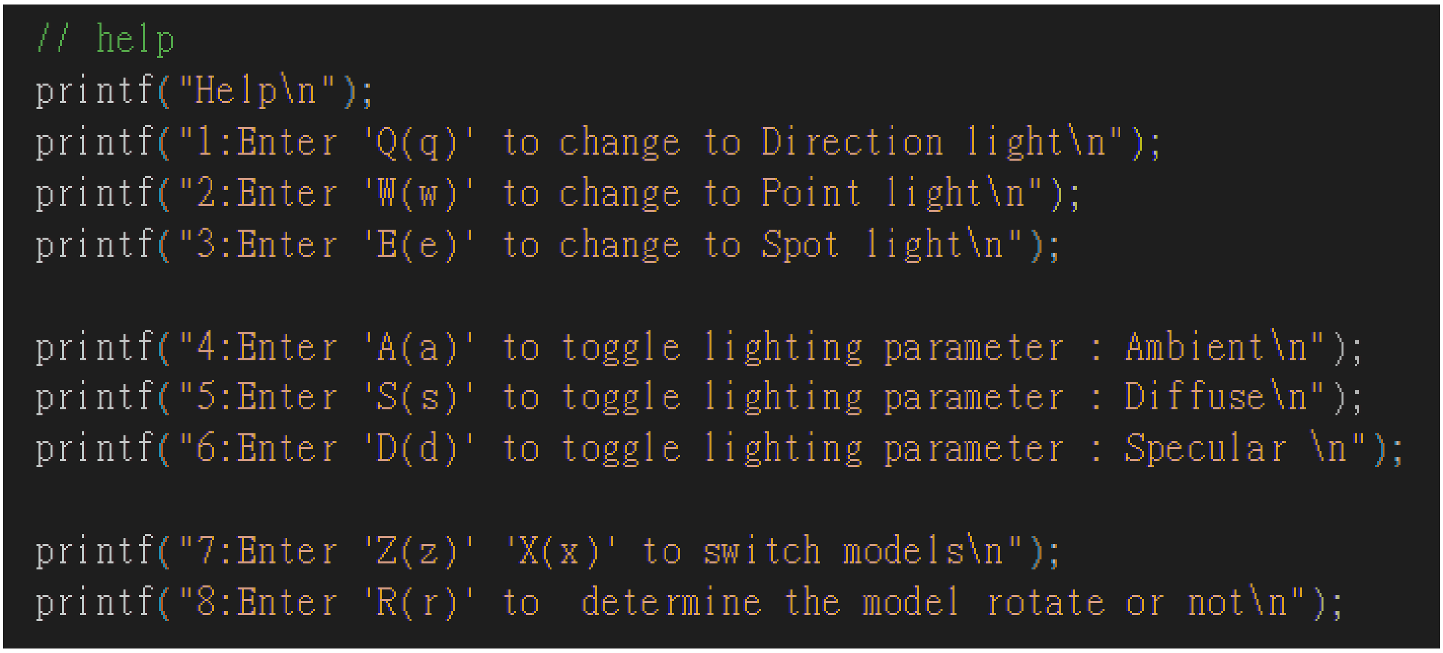
3. Specular:

跟上題差不多，只差在要多乘以 spotlight effect。

四、鍵盤

基本上我都照 spec 裡的要求實作。

1.按 h(H)會出現告訴使用者按什麼鍵的教學



```
// help
printf("Help\n");
printf("1:Enter 'Q(q)' to change to Direction light\n");
printf("2:Enter 'W(w)' to change to Point light\n");
printf("3:Enter 'E(e)' to change to Spot light\n");

printf("4:Enter 'A(a)' to toggle lighting parameter : Ambient\n");
printf("5:Enter 'S(s)' to toggle lighting parameter : Diffuse\n");
printf("6:Enter 'D(d)' to toggle lighting parameter : Specular \n");

printf("7:Enter 'Z(z)' 'X(x)' to switch models\n");
printf("8:Enter 'R(r)' to  determine the model rotate or not\n");
```

2.point light : 上下左右鍵 to move point light

3.spot light :

(1)滑鼠左鍵 : increase the EXP

(2)滑鼠右鍵 : decrease the EXP

(3) Scroll the mouse : modify the Cut-off angle

(4) Slide the mouse : change the position of lighting source