0816028 周孟謙

Phongshading

Vertex shader:

```
normal = normalize((transpose(inverse(M)) * vec4(in_normal, 1.0)).xyz);
uv = texcoord;
worldPos = M * vec4(in_position, 1.0);

gl_Position = P * V * M * vec4(in_position, 1.0);
```

normal 的部分由(Nn)*(Mv)=I 推導出 N=(M^(-1))^T。
uv 則直接由 texcoord 帶入。
worldPos=modelview 乘上 in_position。
gl_Position 則與原本 code 一樣，沒做更改。

Fragment shader:

```
vec4 albedo = texture2D(texture, uv);
float edge_intensity = 0;

vec3 Light = normalize(WorldLightPos - worldPos.xyz);
vec3 Viewer = normalize(WorldCamPos - worldPos.xyz);
vec3 Halfway = normalize(Light + Viewer);

vec3 ambient = La * Ka * albedo.xyz;
vec3 diffuse = Ld * Kd * albedo.xyz * dot(Light, normal); // must > 0
vec3 specular = Ls * Ks * pow(dot(normal,Halfway), gloss);

float cos = dot(Viewer, normal);
if(cos < 0.2)
{
    edge_intensity = 1;
}
vec4 edge_color = vec4(1, 1, 1, 1)  * pow(edge_intensity,5);

// it's the color which only contain texture , you need to change the output to phong shading result
if(EdgeFlag)
{
    color = vec4(ambient + diffuse + specular + edge_color.xyz, 1.0);
}
else
{
    color = vec4(ambient + diffuse + specular, 1.0);
}
```

先將 Light vector, Viewer vector 用 pos 相減後得出再做 normalize，halfway 則依照公式 h = (l + v)/ | l + v |(l 為 Light vector，v 為 Viewer vector)算出。
Ambient, diffuse, specular 皆代入公式:
ambient = La * Ka * object_color;

diffuse = Ld * Kd * object_color * dot(L,N);
specular = Ls * Ks * pow(dot(V,R), gloss);

Edge effects 的部分則是當 Viewer vector 和 normal vector 之間的 cos 小於 0.2 時將 edge_intensity 設為 1，最後再用 EdgeFlag 判斷要不要將 edge_color 加到 color 裡。

Toon

Vertex shader:

```
normal = normalize((transpose(inverse(M)) * vec4(in_normal, 1.0)).xyz);
uv = texcoord;
worldPos = M * vec4(in_position, 1.0);

gl_Position = P * V * M * vec4(in_position, 1.0);
```

與 Phongshading 的 Vertex shader 一樣。

Fragment shader:

```
vec4 albedo = texture2D(texture, uv);
float edge_intensity = 0;
vec4 diffuse;

vec3 Light = normalize(WorldLightPos - worldPos.xyz);
vec3 Viewer = normalize(WorldCamPos - worldPos.xyz);

float intensity;
float level = dot(Light, normal);
if (level > 0.75) intensity = 0.8;
else if (level > 0.30) intensity = 0.6;
else intensity = 0.4;

diffuse = vec4(Kd * albedo.xyz * intensity, 1.0); // must > 0

float cos = dot(Viewer, normal);
if(cos < 0.2)
{
    edge_intensity = 1;
}
vec4 edge_color = vec4(1, 1, 1, 1)  * pow(edge_intensity,5);
// it's the color which only contain darker texture to show different shader, you need to change the output to toon shading result
if(EdgeFlag)
{
    color = vec4(diffuse.xyz + edge_color.xyz, 1.0);
}
else
{
    color = diffuse;
}
```

先將 Viewer vector 用 pos 相減後得出再做 normalize。

level 的部分則是用 Light vector 和 normal vector 之間 cos 的大小來作區間，>0.75, 0.75>= && >0.3, >=0.3 這三個區間分別設三種 intensity(0.8, 0.6, 0.4)。

diffuse 代入公式 Kd * object_color * intensity。
Edge effects 的部分則是與 Phongshading 相同，當 Viewer vector 和 normal vector 之間的 cos 小於 0.2 時將 edge_intensity 設為 1，最後再用 EdgeFlag 判斷要不要將 edge_color 加到 color 裡。

Describe the problems you met and how you solved them.
某些 vec3 和 vec4 的轉換沒有做好，只要將每個 vec 看清楚維度再做轉換就沒問題了。還有在不理解公式的情況下亂代入，解決辦法則是完全搞懂每個公式的意義以及它在幹甚麼，問題便迎刃而解。