20201222 CG hw3 Report

0716207 呂思函

- Shader 初始化 : shaderInit()

```
58  void shaderInit() {
59
60      GLuint vert = createShader("Shaders/Phongshading.vert", "vertex");
61      GLuint frag = createShader("Shaders/Phongshading.frag", "fragment");
62      Phongprogram = createProgram(vert, frag);
63
64      //// TODO: ////
65      // create the shaders and programs you need.
66      GLuint vert2 = createShader("Shaders/ToonShading.vert", "vertex");
67      GLuint frag2 = createShader("Shaders/ToonShading.frag", "fragment");
68      Toonprogram = createProgram(vert2, frag2);
69
70      GLuint vert3 = createShader("Shaders/EdgeEffects.vert", "vertex");
71      GLuint frag3 = createShader("Shaders/EdgeEffects.frag", "fragment");
72      Edgeprogram = createProgram(vert3, frag3);
73
74      // program 初始值(要讓執行的時候馬上有圖案)
75      program = Phongprogram;
76  }
```

  - 創建三個 shading program : Phong Shading, Toon Shading, Edge Effect
  - 並且初始化的 program 是 Phong Shading，讓程式一執行的時候就有模型

- Keyboard control

```
205     case '1':// phong
206     {
207     //// TODO: ////
208     // switch to the program which you want to use
209     program = Phongprogram;
210     break;
211     }
212     case '2':// toon
213     {
214     //// TODO: ////
215     // switch to the program which you want to use
216     program = Toonprogram;
217     break;
218     }
219     case '3':// edge
220     {
221     //// TODO: ////
222     // switch to the program which you want to use
223     program = Edgeprogram;
224     break;
```

  - 利用鍵盤切換不同的 program

- 畫伊布 : DrawUmbreon()，初始化各個參數

```
259     ///// TODO: ////
260     // Pass all variable to shaders and trigger by Uniform (like:WorldLightPos, WorldCamPos, Ka, La .....etc)
261     // glm::vec3 WorldLightPos = glm::vec3(2, 5, 5);
262     ModelMatrixID = glGetUniformLocation(program, "WorldLightPos");
263     glUniform3fv(ModelMatrixID, 1, &WorldLightPos[0]);
264     // glm::vec3 WorldCamPos = glm::vec3(7.5, 5.0, 7.5);
265     ModelMatrixID = glGetUniformLocation(program, "WorldCamPos");
266     glUniform3fv(ModelMatrixID, 1, &WorldCamPos[0]);
```

  - 需要注意的點是 WorldLightPos 和 WorldCamPos 是 3d vector

- 所以要用 glUniform3fv 傳參數到 shader 裡

```
ModelMatrixID = glGetUniformLocation(program, "Ka");
glUniform3fv(ModelMatrixID, 1, &Ka[0]);
glm::vec3 Kd = glm::vec3(1, 1, 1);// diffuse reflectivity
ModelMatrixID = glGetUniformLocation(program, "Kd");
glUniform3fv(ModelMatrixID, 1, &Kd[0]);
glm::vec3 Ks = glm::vec3(1, 1, 1);// specular reflectivity
ModelMatrixID = glGetUniformLocation(program, "Ks");
glUniform3fv(ModelMatrixID, 1, &Ks[0]);
```

```
glm::vec3 La = glm::vec3(0.2, 0.2, 0.2);// ambient intensity
ModelMatrixID = glGetUniformLocation(program, "La");
glUniform3fv(ModelMatrixID, 1, &La[0]);
glm::vec3 Ld = glm::vec3(0.8, 0.8, 0.8);// diffuse intensity
ModelMatrixID = glGetUniformLocation(program, "Ld");
glUniform3fv(ModelMatrixID, 1, &Ld[0]);
glm::vec3 Ls = glm::vec3(0.5, 0.5, 0.5);// specular intensity
ModelMatrixID = glGetUniformLocation(program, "Ls");
glUniform3fv(ModelMatrixID, 1, &Ls[0]);
glm::vec3 gloss = glm::vec3(25.0, 25.0, 25.0);// specular shininess factor
ModelMatrixID = glGetUniformLocation(program, "gloss");
glUniform3fv(ModelMatrixID, 1, &gloss[0]);
```

- 建立各種 3d vector 參數
- 因為找不到 float 要怎傳，所以 gloss 的部分還是用 3d vector 去傳

➢ Vertex shader

```
#version 430

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


uniform mat4 M, V, P;

out vec2 uv;
out vec3 normal;
out vec3 worldPos;

void main() {
 gl_Position = P * V * M * vec4(in_position, 1.0);// 設定model位置
 normal =  mat3(transpose(inverse(M))) * in_normal, 1.0;// 計算法向量經過模型變換後值
 uv= texcoord;
 worldPos = vec3(V * vec4(in_position, 1.0));// 在世界座標指定，傳座標給fragment shader
}
```

- 三個 shading 的 vertex shader 都相同
- 用 location 拿 main 的位置、法向量、材質的參數
- 利用 uniform 傳入 model(M)、model view(V)、projection(P)
- gl_Position 用來設定 model 的位置
- 要傳到 fragment shader 的 normal 必須先經過模型變換(先 inverse 再 transpose)，因為要讓 light 和 model 同在 world coordinate 或 model coordinate 才不會有衝突
- worldPos 這個變數用來儲存模型的世界座標，之後傳給 fragment shader

- PhongShading(fragment shader)

```
#version 430

uniform sampler2D texture;
uniform vec3 WorldLightPos, WorldCamPos;
uniform vec3 Ka, La, Kd, Ld, Ks, Ls, gloss;


in vec2 uv;
in vec3 normal;
in vec3 worldPos;
out vec4 frag_color;

vec3 lightDir = normalize(WorldLightPos - worldPos);// 光線要指向光源
vec3 viewDir = normalize(WorldCamPos - worldPos);
vec3 norm = normalize(normal);
vec3 R = normalize(reflect(-lightDir, norm));// 反射角
```

```
void main(){
  vec4 color = texture2D(texture, uv);

  float ambient = La[0] * Ka[0];
  float diffuse = Ld[0] * Kd[0] * dot(lightDir, norm);// must > 0
  float specular = Ld[0] * Ks[0] * pow(dot(viewDir, R), gloss[0]);

  frag_color = (ambient + diffuse + specular) * color;
}
```

  - 用 uniform 去接要使用的各種參數
  - 發現的問題 : fragment shader 的 out 只能有一個，不然會有衝突
  - main 裡的是計算 Phong Shading 各種因子的公式 : ambient、diffuse、specular(因為數值預設的關係所以 dot 不會小於 0，否則應該會有更進一步的判斷=>要取>0 的結果，且 gloss 必須>0)
  - 要把各種參數 normalize 的原因是 : 這些要計算的向量必須換成單位向量，否則計算結果可能會>0(燈光各種參數加起來應該會是 1，否則會爆亮過曝)

- ToonShading(fragment shader)

```
#version 430

uniform sampler2D texture;
uniform vec3 WorldLightPos, Kd;

in vec2 uv;
in vec3 normal;
in vec3 worldPos;
out vec4 frag_color;

vec3 lightDir = normalize(WorldLightPos - worldPos);// 光線要指向光源
vec3 norm = normalize(normal);
float level = dot(lightDir, norm);
float intensity;
```

```
void main(){
  vec4 color = texture2D(texture, uv);

  // Decide a level by calculating the included angles between light and normal vectors
  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;

  frag_color = vec4(Kd, 1.0) * intensity *color;
}
```

  - 上半部傳參數和變數宣告與 Phong Shading 類似
  - 底下 main 的地方是將顏色分成五層，利用公式將單位向量的 light vector 和 normal vector 內積
  - 得到對應的光強度之後，再與 diffuse 係數和 texture color 相乘

- EdgeEffect(fragment shader)

```
#version 430

uniform sampler2D texture;
uniform vec3 WorldCamPos;

in vec2 uv;
in vec3 normal;
in vec3 worldPos;
out vec4 frag_color;

vec3 viewDir = normalize(WorldCamPos - worldPos);
vec3 norm = normalize(normal);
vec4 color;
vec4 outline = vec4(0.0, 1.0, 1.0, 1.0);
```

```
void main(){
  float cos = dot(viewDir, norm) / (length(viewDir) * length(norm));

  if( cos<0.01 && cos>0 ) color = outline * (1 - cos);
  else   color = vec4(0.0, 0.0, 0.0, 0.0);

  frag_color =  color;
}
```

  - 上半部傳參數和變數宣告與 Phong Shading 及 Toon Shading 類似
  - 設定一個邊框顏色，在這邊我使用青色讓邊框清楚一點
  - 要取得圖案邊框，要從三角函數去取，這邊使用單位向量的光線和法向量的內積來得到 cos
  - 將 cos 值接近 0，盡量逼到圖案邊緣、以取得外框
  - 其他區塊就設成黑色