Introduction to Computer Graphics HW3:Shader Implementation

0711239 李勝維

大綱

- main.cpp
- Edge effect
- Toon shading
- Phong shading
- 說明順序是因為我從Edge effect開始寫，後面的是由前面的修改而成，這樣比較順

main.cpp

Step 1 Create Program

```
void shaderInit()
{
    GLuint vert = createShader("Shaders/Phongshading.vert", "vertex");
    GLuint frag = createShader("Shaders/Phongshading.frag", "fragment");
    Phongprogram = createProgram(vert, frag);

    vert = createShader("Shaders/Toonshading.vert", "vertex");
    frag = createShader("Shaders/Toonshading.frag", "fragment");
    Toonprogram = createProgram(vert, frag);

    vert = createShader("Shaders/Edgeshading.vert", "vertex");
    frag = createShader("Shaders/Edgeshading.frag", "fragment");
    Edgeprogram = createProgram(vert, frag);

    program = Phongprogram;
}
```

使用createShader()和createProgram()

create出三組不同shader對應的program，方便之後切換

Step 2 Switch between programs

```
case '1':
{
    program = Phongprogram;
    break;
}
case '2':
{
    program = Toonprogram;
    break;
}
case '3':
{
    program = Edgeprogram;
    break;
}
```

(in void keyboard())

可以使用keyboard control來切換不同的shader 模式

Step 3-1 Pass variables by Uniform

```
// parameters for all three shaders
glUniform3fv(glGetUniformLocation(program, "worldCam"), 1, &WorldCamPos[0]);
glUniform3fv(glGetUniformLocation(program, "worldLight"), 1, &WorldLightPos[0]);

// parameters for edge
glm::vec4 blue(0, 0, 1, 1);
glUniform4fv(glGetUniformLocation(program, "blue"), 1, &blue[0]);
```

- 首先是所有模式都會用到的2組座標：worldCam和worldLight傳入，分別代表了攝影機和光源的三維世界座標
- 再來是一組vec4 blue，代表了藍色，將會使用在edge effect中（edge是藍色的）

Step 3-2 Pass variables by Uniform

```
// parameters for Phong
glm::vec3 Ka(1, 1, 1), Kd(1, 1, 1), Ks(1, 1, 1), La(0.2f, 0.2f, 0.2f), Ld(0.8f, 0.8f, 0.8f), Ls(0.5f, 0.5f, 0.5f);
float gloss = 25.0f;
glUniform3fv(glGetUniformLocation(program, "Ka"), 1, &Ka[0]);
glUniform3fv(glGetUniformLocation(program, "Kd"), 1, &Kd[0]);
glUniform3fv(glGetUniformLocation(program, "Ks"), 1, &Ks[0]);

glUniform3fv(glGetUniformLocation(program, "La"), 1, &La[0]);
glUniform3fv(glGetUniformLocation(program, "Ld"), 1, &Ld[0]);
glUniform3fv(glGetUniformLocation(program, "Ls"), 1, &Ls[0]);

glUniform1f(glGetUniformLocation(program, "gloss"), gloss);
```

- 最後將Phong shading會用到的Ka, Kd, Ks, La, Ld, Ls和gloss參數傳入

Edge Effect

Part 1 Vertex shader

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

uniform mat4 M, V, P;

out vec3 worldNormal;
out vec3 worldPos;

void main()
{
    gl_Position = P * V * M * vec4(in_position, 1.0);
    worldPos = mat3(P * V * M) * in_position;
    // for normal space convertion, see: https://computergr
    worldNormal = mat3(transpose(inverse(M))) * in_normal;
}
```

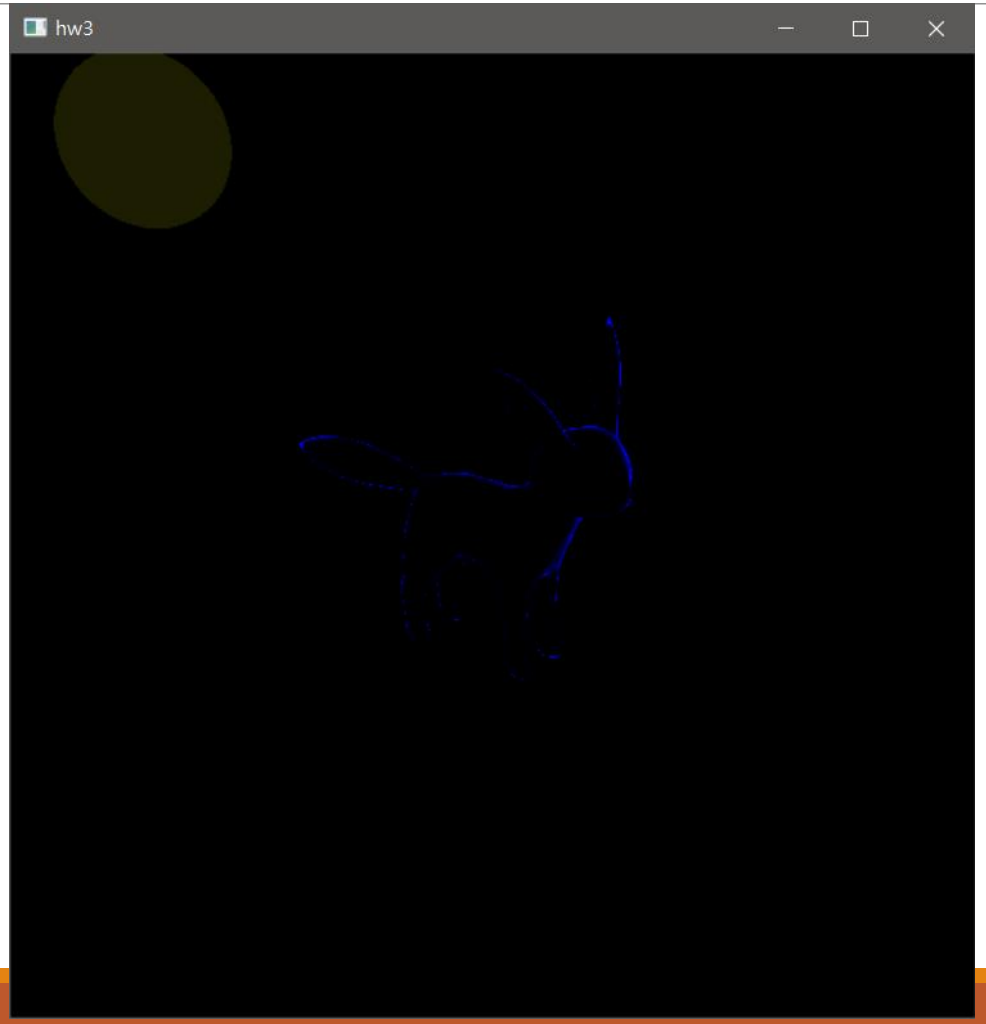
- 計算出gl_Position
- 將position和normal皆轉換到world space，並傳給fragment shader

Part 2 Fragment shader

```
in vec3 worldNormal;
in vec3 worldPos;

uniform vec3 worldCam;
uniform vec4 blue;

out vec4 color;

void main()
{
    vec3 to_cam = normalize(worldCam - worldPos);
    float intensity = 1.0f - dot(to_cam, worldNormal);
    intensity = pow(intensity , 10); // strengthening edges
    color = intensity * blue;
}
```

- 先用座標減法計算出指向攝影機的單位向量：to_cam
- 再利用內積計算to_cam和normal的cos值，該值越接近0，代表該點越接近邊緣
- 將計算出的intensity做10次方，可以eliminate比較小的值，變成只有邊緣有顏色（不然會有整片藍色的情況發生）
- 最後將藍色乘上光強度輸出

Result



Toon Shading

Part 1 Vertex shader

```
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 worldNormal;
out vec3 worldPos;

void main()
{
    gl_Position = P * V * M * vec4(in_position, 1.0);
    worldPos = mat3(P * V * M) * in_position;
    worldNormal = mat3(transpose(inverse(M))) * in_normal;
    uv = texcoord;
}
```

- 做的事情和Edge effect時幾乎一樣
- 將position和normal皆轉換到world space，並傳給fragment shader
- 再多傳入texture coordinate給fragment shader

Part 2-1 Fragment shader

```
in vec2 uv;
in vec3 worldNormal;
in vec3 worldPos;

uniform vec3 worldLight;
uniform sampler2D texture;

out vec4 color;

void main()
{
    vec3 to_light = normalize(worldLight - worldPos);
    float level = dot(to_light, worldNormal);
    float intensity;
```

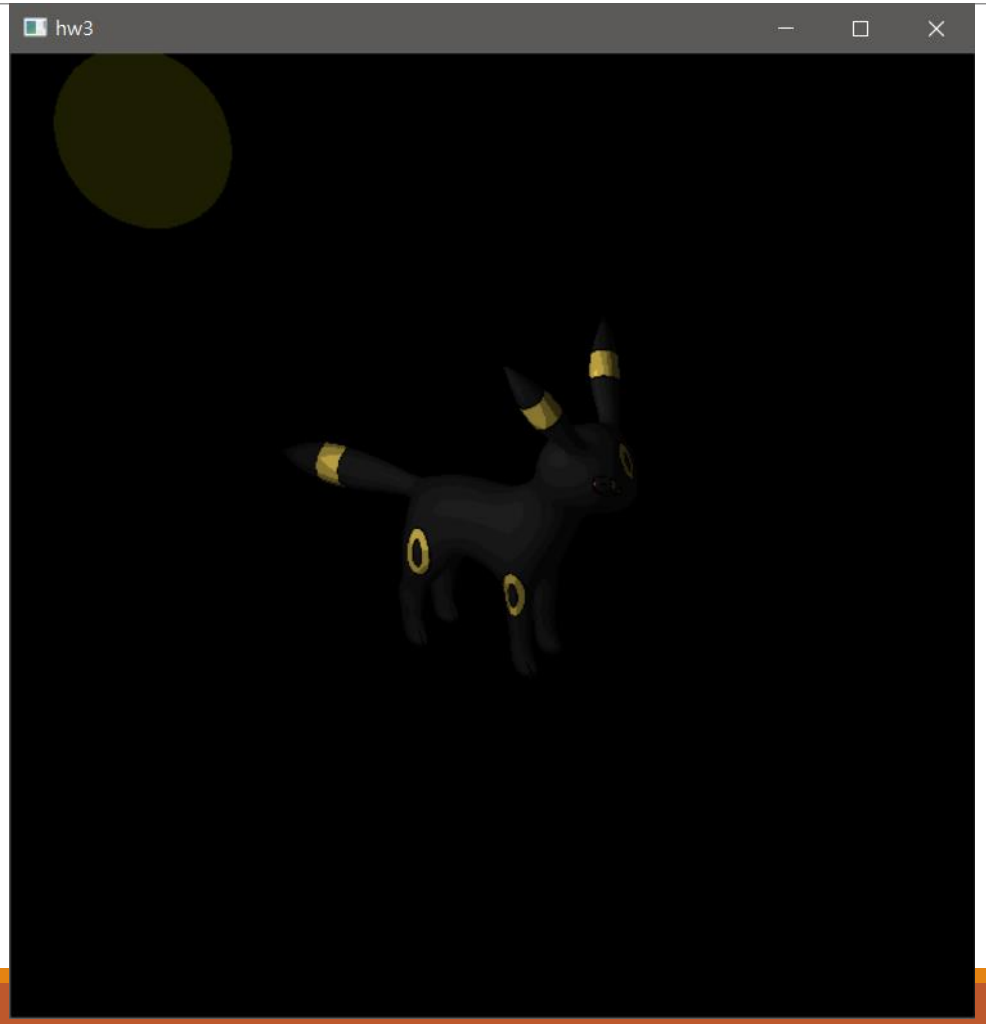
- 用座標減法算出指向光源的單位向量：to_light
- 再計算出to_light和normal夾角的cos值，當作Toon shading分層的依據

Part 2-2 Fragment shader

```
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;
color = texture2D(texture, uv) * intensity;
```

- 依據前面算出的cos值，level，來分成5層的光強度
- 將得到的texture乘上光強度輸出

Result



Phong Shading

Part 1 Vertex shader

```
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 worldNormal;
out vec3 worldPos;

void main()
{
    gl_Position = P * V * M * vec4(in_position, 1.0);
    worldPos = mat3(P * V * M) * in_position;
    worldNormal = mat3(transpose(inverse(M))) * in_normal;
    uv = texcoord;
}
```

- 做的事情和Toon shading時一樣
- 將position和normal皆轉換到world space，並傳給fragment shader
- 再傳入texture coordinate給fragment shader

Part 2-1 Fragment shader

```
in vec2 uv;
in vec3 worldNormal;
in vec3 worldPos;

uniform vec3 worldLight, worldCam, Ka, Kd, Ks, La, Ld, Ls;
uniform float gloss;
uniform sampler2D texture;

out vec4 color;

void main()
{
    vec3 to_light, to_cam, reflection, tmp, ambient, diffuse, specular;
    to_light = normalize(worldLight - worldPos);
    to_cam = normalize(worldCam - worldPos);
    reflection = 2.0f * dot(worldNormal, to_light) * worldNormal - to_light;
    tmp = texture2D(texture, uv).rgb;
```

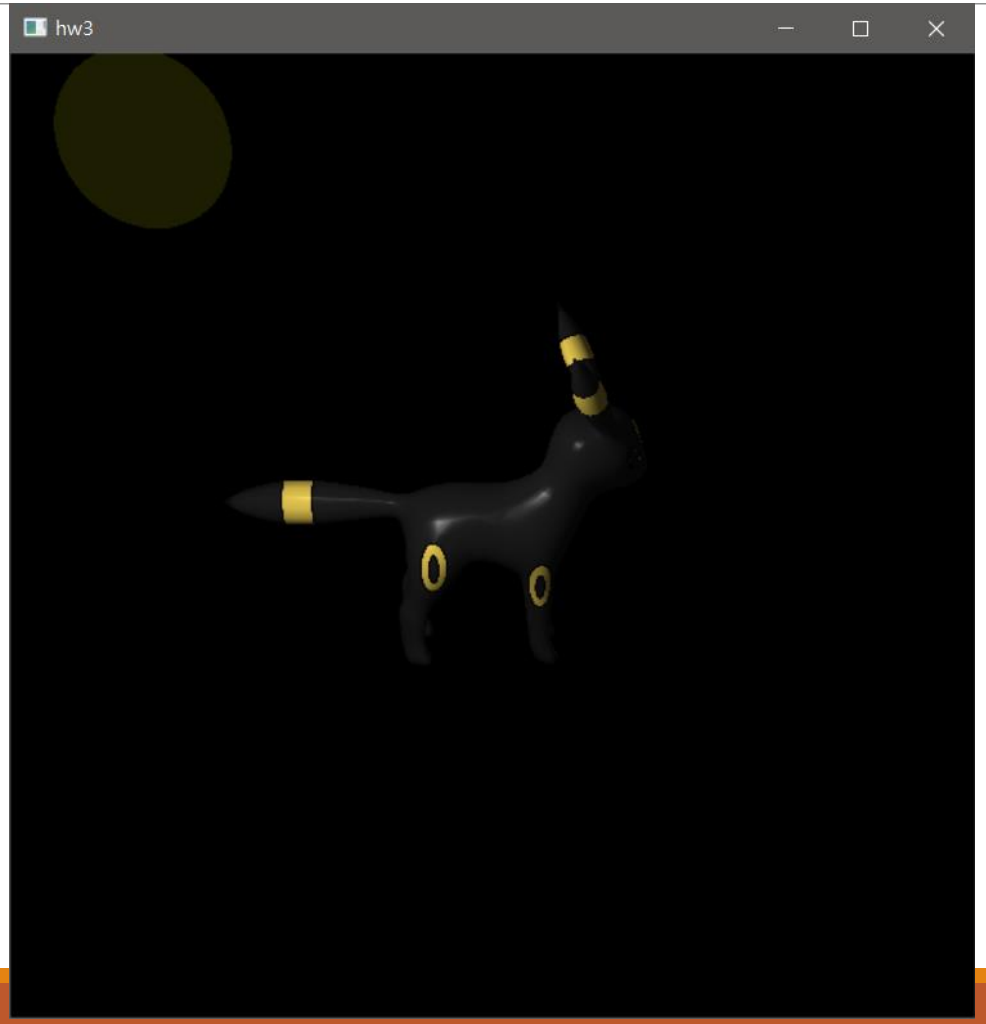
- 一樣先計算出分別指向攝影機和光源的單位向量，to_cam和to_light
- 再利用公式計算出反射角方向向量，reflection
- 最後利用一變數，tmp，儲存原始的texture顏色，方便後續計算

Part 2-2 Fragment shader

```
float cos1, cos2;
cos1 = max(0.0f, dot(to_light, worldNormal)); // angle between (l, n)
cos2 = max(0.0f, dot(to_cam, reflection));    // angle between (v, r)

ambient = La * Ka * tmp;
diffuse = Ld * Kd * tmp * cos1;
specular = Ls * Ks * pow(cos2, gloss);
color = vec4(ambient + diffuse + specular, 1.0f);
```

- 首先分別計算出(l, n)和(v, r)兩組夾角的cos值，其中利用了max()函數來去除負值
- 再將各種參數代入公式，分別計算出Phong shading 的 ambient, diffuse 和 specular三個分量
- 將三個分量加在一起並轉換成vec 4輸出

Result



Problems I’ve met

gl_Position.xyz

```
{
    gl_Position = P * V * M * vec4(in_position, 1.0);
    worldPos = mat3(P * V * M) * in_position;
    // for normal space convertion, see: https://computergraphics.stackexchange.com/questions/1502/why-is-the-transpo
    worldNormal = mat3(transpose(inverse(M))) * in_normal;
}
```

- 在Edgeshading.vert中 我本來是直接使用gl_Position.xyz來當作世界座標，但是其實不是
- 後來才知道應該要使用gl_Position.xyz/gl_Position.w來當作世界座標才對，因為gl_Position已經經過normalization了

沒了，謝謝助教