**HW3 Report**

110550071 田松翰

1. **Blinn-Phong shading**

*Vertex shader:*

Get MVP and camera (viewPos) from main using uniform.

worldPos = M\*vec4(aPos, 1.0)

Get gl\_Position by P\*V\*worldPos

Output texCoord, normal, worldPos, viewPos.

*Fragment shader:*

color = vec3(texture(ourtexture, texCoord))

Ambient = La \* Ka \* color

Diffuse = Ld \* Kd \* color \* (0.0, dot(light, normal)) // normalize light and normal.

half\_vec = lightPos + viewPos

Specular = Ls \* Ks \* pow(max(0.0, dot(normal, half\_vec)), a) // normalize normal and half\_vec

Fragcolor = vec4(ambient+diffuse+specular, 1.0)

1. **Gouraud shading**

*Vertex shader:*

Apply Phong shading on each vertex.

Get MVP and camera (viewPos) from main using uniform.

worldPos = M\*vec4(aPos, 1.0)

Get gl\_Position by P\*V\*M\*ve4(aPos, 1.0).

Calculate N, L, V, R for Phong.

N = normalize(normal)

L = normalize(lightPos - worldPos.xyz)

V = normalize(viewPos – worldPos.xyz)

R = normalize(reflect(-lightPos, normal)

Ambient = La\*Ka

Diffuse = Ld\*Kd\*max(0.0, dot(L, N))

Specular = Ls\*Ks\*pow(max(0.0, dot(V, R), a)

Output texCoord, ambient, diffuse, specular to fragment shader.

*Fragment shader:*

Get texture(color) by uniform sampler2D from main.

Fragcolor = vec4(ambient\*color + diffuse\*color + specular), 1.0)

1. **Flat shading:**

*Vertex shader:*

Get MVP, camera, gl\_Position and normal as the methods in Part1.

Due to geometry shader, output a structure VS\_OUT with texCoord, normal, worldPos inside.

*Geometry shader:*

Get the VS\_OUT from vertex shader as input, and output fragNormal, texCoord, worldPos to next shader.

Using layout (triangle) as in, and layout(triangle\_strip, max\_vertices=3) as out.

fragNormal = sum of normal of 3 vertices and normalize.

worldPos uses the first vertex worldPos.

gl\_Position and texCoord uses those from each vertex.

*Fragment shader:*

Like Part2, calculate N, L, V, R for Phong.

Change “normal” to “fragNormal” getting from geometry shader.

Get the result by ambient\*color + diffuse\*color + specular and return FragColor.

1. **Toon shading:**

*Vertex shader:*

Do the same thing as Part3.

*Fragment shader:*

Like Part2, calculate N, L, V, R for Phong.

Calculate dot(N, L) as cos, and specular = Ls\*Ks\*pow(max(0.0, dot(V, R), a)

If cos<0.4, apply low intensity.

Else if specular.x or y or z > 0.02, apply high intensity.

Else, apply medium intensity.

// 0.4 and 0.02 just a threshold I found.

1. **Border shading:**

*Vertex shader:*

Do the same thing as Part4.

*Fragment shader:*

Calculate angle between normal and view\_dir as dot(normal, view\_dir).

Return FragColor = 0.9(1-angle) + angle\*vec4(color, 1.0).

1. **Dissolve shading**

*Vertex shader:*

Do the same thing as Part4.

Additionally, send a xPos as the x position to fragment shader.

*Fragment shader:*

Set an increasing threshold (“x\_dissolve”) that will increase with time and send it from main to this shader by uniform.

When xPos >= x\_dissolve, the object will be presented.

Otherwise, discard the vertex so that it will not be showed on the window.

**Problems I met:**

1. I didn’t know the correct method to use the geometry shader, and I was not cleared for the mission for each shader works. By the way, I didn’t know how to use vs\_out as the structure to send variables between shaders.

* I check for lots of websites to learned and asked my classmate to understand more details for it.

1. I thought that I can only send uniform to the vertex shader, so I sent lots of variables that only used on geometry shader or fragment shader.

* I know that I can call uniform on geometry or fragment shader to use them.

**Result:**

一張含有 哺乳動物, 卡通, 鹿, 動物的鼻子 的圖片

自動產生的描述 一張含有 哺乳動物, 動物玩偶, 動物的鼻子, 鹿 的圖片

自動產生的描述 一張含有 哺乳動物, 動物玩偶, 鹿, 卡通 的圖片

自動產生的描述 一張含有 哺乳動物, 動物的鼻子, 鹿 的圖片

自動產生的描述 一張含有 圖畫, 寫生, 動物玩偶, 卡通 的圖片

自動產生的描述 一張含有 哺乳動物, 羚羊, 幼鹿, 麆鹿 的圖片

自動產生的描述