Computer Graphics HW3

Info

- Author: 0712238, Yan-Tong Lin
- Online Version: https://hackmd.io/b18vIHrDR76L4jQBdt5zzA
- Environment
  - OS: Windows 10
- Execution
  - open with `Visual Studio` (Note: not VS Code) and press `F5` to run

Task

- Illuminate eevee(3D object) with Phong Shading/Toon Shading/Edge Effect.
- Code the shaders and pass params to them.

Implementation

Explained with code and comments.

Switching between Programs

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

Passing Parameters through UniformPosition Binding

get position by `glGetUniformLocation({PROGRAM}, "{VAR_NAME}");` and use `glUniform{FOMAT}` to bind pointer

```

glm::vec3 WorldLightPos = glm::vec3(2, 5, 5);
glm::vec3 WorldCamPos = glm::vec3(7.5, 5.0, 7.5);

// K, params for material reflection
glm::vec3 Ka = glm::vec3(1., 1., 1.);
glm::vec3 Kd = glm::vec3(1., 1., 1.);
glm::vec3 Ks = glm::vec3(1., 1., 1.);

// L, params for light
glm::vec3 La = glm::vec3(0.2, 0.2, 0.2);
glm::vec3 Ld = glm::vec3(0.8, 0.8, 0.8);
glm::vec3 Ls = glm::vec3(0.5, 0.5, 0.5);

// Edge Color for Edge Effect
glm::vec4 edge_color = glm::vec4(0.0, 0.0, 1.0, 1.0);

float gloss = 25.;

// Pass all variable to shaders and trigger by Uniform
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);
glUniform4fv(glGetUniformLocation(program, "edge_color"), 1, &edge_color[0]);
```

Shared Vertex Shader

```
#version 430

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


uniform mat4 M, V, P;

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

void main() {
  vec4 worldPos4 = V * M * vec4(in_position, 1.0);
  // projected position
  gl_Position = P * worldPos4;
  // position in the object space
  worldPos = vec3(worldPos4) / worldPos4.w;
  //passing texture coordinate
  uv = texcoord;
  //calculate normal after translation
  //https://www.cs.upc.edu/~robert/teaching/idi/normalsOpenGL.pdf
  normal = mat3(transpose(inverse(M))) * normal_in;
}

```

Phong Shading

```glsl
#version 430

uniform sampler2D texture;

in vec2 uv;
in vec3 normal;
in vec3 worldPos;

uniform vec3 worldLightPos, worldCamPos;
uniform vec3 Ka, Kd, Ks, La, Ld, Ls;
uniform float gloss;

out vec4 color;

void main()
{
  //calculate normalized normal and to_light vector
  // normalize for computing cosine value
  vec3 N = normalize(normal);
  vec3 L = normalize(worldLightPos - worldPos);

  // Lambert's cosine law
  // If cos(N, L) < 0, theta is bigger than pi/2
  // the position cannot be illuminated
  float lambertian = max(dot(N, L), 0.0);
  float spec = 0.0;
  if(lambertian > 0.0) {
    vec3 R = normalize(reflect(-L, N));      // Reflected light vector
    vec3 V = normalize(worldCamPos-worldPos); // Vector to eye
    // Compute the specular term
    float specAngle = max(dot(R, V), 0.0);
    spec = pow(specAngle, gloss);
  }
  // three light sources combined is the resulting color of Phong shading
  vec4 obj_color = texture2D(texture, uv);
  vec4 ambient = vec4(La, 1.0) * vec4(La, 1.0) * obj_color;
  vec4 diffuse = vec4(Ld, 1.0) * vec4(Kd, 1.0) * obj_color * lambertian;
  vec4 specular = vec4(Ls, 1.0) * vec4(Ks, 1.0) * spec;
  color = ambient + diffuse + specular;
}
```

Toon Shading

```
#version 430

uniform sampler2D texture;

in vec2 uv;
in vec3 normal;
in vec3 worldPos;

uniform vec3 worldLightPos, worldCamPos;
uniform vec3 Ka, Kd, Ks, La, Ld, Ls;
uniform float gloss;

out vec4 color;

void main()
{
  vec3 N = normalize(normal);
  vec3 L = normalize(worldLightPos - worldPos);

  // Toon Shading considers cos(N,L)
  float level = dot(N, L);
  //assign intensities for different consine value ranges
  // this will make shading result "discrete" and seems "cartoon"
  float intensity;
  if(level > 0.95) intensity = 1.0;
  else if(level > 0.75) intensity = 0.8;
  else if(level > 0.5) intensity = 0.6;
  else if(level > 0.25) intensity = 0.4;
  else intensity = 0.2;

  color = intensity * vec4(Kd, 1.0) * texture2D(texture, uv);
}

```

Edge Effect

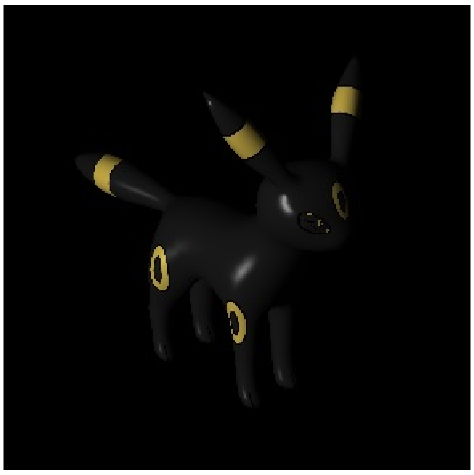
I tried two ways

- $intensity = |\cos(N,V)| < 0.01$ (discrete)
    - looks too discrete
- $intensity = e^{-10|\cos(N,V)|}$ (smooth)
    - this is smoother

```glsl
#version 430

uniform sampler2D texture;

in vec2 uv;
in vec3 normal;
in vec3 worldPos;

uniform vec3 worldLightPos, worldCamPos;
uniform vec3 Ka, Kd, Ks, La, Ld, Ls;
uniform float gloss;
uniform vec4 edge_color;

out vec4 color;

void main()
{
  vec3 N = normalize(normal);
  vec3 V = normalize(worldCamPos-worldPos);
  // Edge Effect, Show Color When normal and view is almost orthogonal
  /*float level = dot(N, V);
  float intensity = 0.0;
  if(level < 0.05 && level > -0.05) {
    intensity = 1.0;
  }else{
    intensity = 0.0;
  }*/
  float intensity = exp(-10*abs(dot(N, V)));
  //vec4 obj_color = texture2D(texture, uv);
  color = intensity*edge_color;
}
```

Results

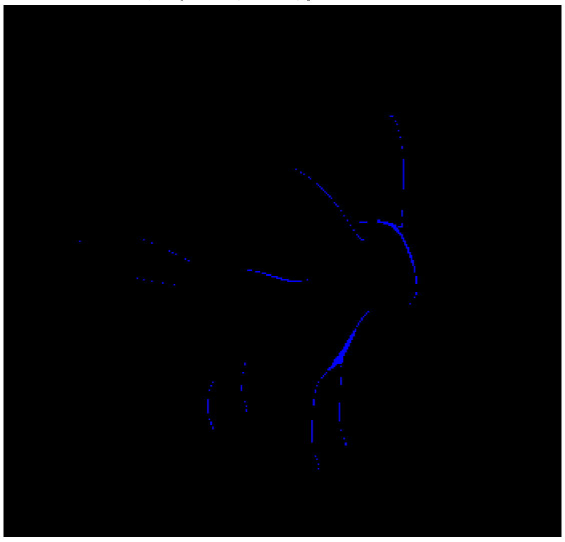
Phong Shading



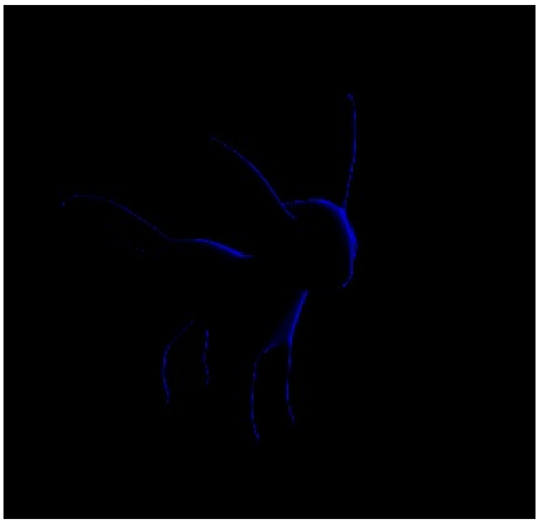
Toon Shading

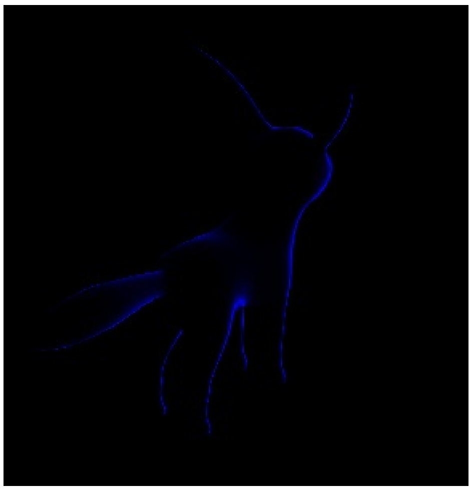
Edge Effect

- with intensity=$|\cos(N,V)|<0.01$ (discrete)



- with intensity=$e^{-10|\cos(N,V)|}$ (smooth)





Resources

- How to use unifrm bind
  - https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glUniform.xhtml
- Transformation of Normal
  - https://www.cs.upc.edu/~robert/teaching/idi/normalsOpenGL.pdf
- WebGL Phong Shading example
  - http://www.cs.toronto.edu/~jacobson/phong-demo/
- Toon Shading Reference
  - https://stackoverflow.com/questions/5795829/using-opengl-toon-shader-in-glsl
- Edge Effect Reference
  - https://computergraphics.stackexchange.com/questions/2450/opengl-detection-of-edges
  - https://en.wikibooks.org/wiki/GLSL_Programming/Unity/Toon_Shading#Outlines