```glsl
#version 330 core

// Blinn-Phong Shader with updated power exponent to respond to roughness
// Blinn-Phong diffuse replaced with Oren-Nayar


// uniform data
uniform mat4 uProjectionMatrix;
uniform mat4 uModelViewMatrix;
uniform vec3 uColor;

// I created these-------------------
uniform float u_ambient_coeff;
uniform float u_diffuse_coeff;
uniform float u_specular_coeff;
uniform float u_power_specular_coeff;
uniform vec3 u_light_position;
uniform float u_roughness;
//-----------------------------------


// viewspace data (this must match the output of the fragment shader)
in VertexData {
    vec3 position;
    vec3 normal;
    vec2 textureCoord;
} f_in;

// framebuffer output
out vec4 fb_color;

void main() {

    // Oren-Nayar Diffuse----------------------------------
    float A = 1.0 - 0.5 * (u_roughness * u_roughness) / (u_roughness * u_roughness + 0.33);
    float B = 0.45 * (u_roughness * u_roughness) / (u_roughness * u_roughness + 0.09);
    float cos_theta_n_l = max(0.0, dot(f_in.normal, light_direction));
    float cos_theta_n_v = max(0.0, dot(f_in.normal, eye));
    float alpha = acos(dot(normalize(-light_direction + eye), f_in.normal));
    float cos_diff_n_l_n_v = cos(max(cos_theta_n_l, cos_theta_n_v) * alpha);
    float sin_max_theta = sin(max(cos_theta_n_l, cos_theta_n_v));

    float O_N_reflectance = A + B * cos_theta_n_l * cos_theta_n_v * sin_max_theta / max(cos_theta_n_l, cos_theta_n_v) * cos_diff_n_l_n_v;

    //-------------------------------------------------

    vec3 eye = normalize(-f_in.position);   // camera's perspective
    float light = abs(dot(normalize(f_in.normal), eye));

    vec3 light_color_1 = vec3(0.8, 0.8, 0.8);  // white light
    vec3 light_color = uColor * light_color_1;

    // calculated variables from user-defined values-----------------
    vec3 light_direction = normalize(u_light_position - f_in.position);
    vec3 halfway_vector = normalize(light_direction + eye);
    float specular_power = (2 / u_roughness) - 2;
    //--------------------------------------------------------------


    // float reflectance_diffuse = u_diffuse_coeff * dot(f_in.normal, light_direction);
    float reflectance_diffuse = O_N_reflectance;

    float reflectance_specular = u_specular_coeff * pow(dot(halfway_vector, f_in.normal), specular_power);

    vec3 color = vec3(u_ambient_coeff + (light_color * (reflectance_diffuse + reflectance_specular)));
    color *= uColor; // combine color of model with light reflectance
    //-----------------------------------------------------

    // output to the frambuffer
    fb_color = vec4(color, 1);

}
```