```glsl
#version 330 core

// Cook-Torrance Shader
// active inputs from ImGui; Ambient Light, Specular Light, Roughness

// 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_s_d_ratio;
uniform float u_roughness;
uniform float u_refraction;
//-----------------------------------

float PI = 3.141592653589793;


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

// frame buffer output
out vec4 fb_color;

void main() {

    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 = 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 D = (1.0 / (PI * u_roughness * u_roughness)) * pow(max(dot(halfway_vector, f_in.normal), 0.0), specular_power);

    float G_1 = (2 * dot(halfway_vector, f_in.normal) * dot(f_in.normal, eye)) / dot(eye, halfway_vector);
    float G_2 = (2 * dot(halfway_vector, f_in.normal) * dot(f_in.normal, light_direction)) / dot(eye, halfway_vector);
    float G = min( 1, min(G_1, G_2) );

    float F_0 = ((u_refraction - 1) * (u_refraction - 1)) / ((u_refraction + 1) * (u_refraction + 1));
    float F = F_0 + (1 - F_0) * pow((1 - (dot(eye, halfway_vector))), 5);


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

    // calculate cook torrance-------------------------
    float reflectance_diffuse = u_diffuse_coeff;
    //float reflectance_specular = (D * G * F) / (4.0 * dot(f_in.normal, light_direction) * dot(f_in.normal, eye));   // Full Cook-Torrance specular term
    float reflectance_specular = (D * G * F) / dot(f_in.normal, eye); // simplified version or visual preference


    vec3 color = vec3(u_ambient_coeff + light_color * (dot(f_in.normal, light_direction) * (u_s_d_ratio * reflectance_diffuse + (1 - u_s_d_ratio) * reflectance_specular)));
    color *= uColor;  // combine color of model with lighting's reflection
    //-------------------------------------------------

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

}

```