Casting Shadows in OpenGL

Identify the Light Source Casting the Shadow

First, Render the Scene from the Point of View of that Light Source

1. Render a view from the light source – everything you cannot see must be in a shadow

Second, use the Z-buffer as a Depth Shadow Map

2. Generate a depth view from the point of view of the light source (in OpenGL, this means extracting the z-buffer)

Third, Render the Scene as you Normally Would, but Consult the Depth Map to Decide where Lighting Can’t Reach and Shouldn’t Be Used

3. Put the eye back where it really belongs. Render that view. Every time you create a pixel in the scene, compare its 3D location against the depth map. If the light-position camera could not see it before, don’t allow lighting to be applied to it now.

OpenGL Shadow Demo Program: The Depth (Z-buffer) Shadow Map

The depth shadow map is created from the point of view of the light source.

The rendering is done into an off-screen framebuffer and only renders the depth (z-buffer), not any colors.

In this grayscale image, dark colors are nearest to the eye, light colors are farther away.
OpenGL Demo Program: Creating the off-screen Depth Shadow Map Framebuffer

1. Create a framebuffer object and a depth texture object:

```glBindFramebuffer(GL_FRAMEBUFFER, DepthFramebuffer);
// attach texture to framebuffer as depth buffer:
```

2. Create a texture that will be the framebuffer’s depth buffer:

```glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT, SHADOW_WIDTH, SHADOW_HEIGHT,
             GL_FLOAT, NULL);
```
void main() {
    vec3 normal = normalize(vNs);
    vec3 light = normalize(vLs);
    vec3 eye = normalize(vEs);
    float d = 0.;
    float s = 0.;
    vec3 lighting = KA * uColor;
    bool isInShadow = IsInShadow(vFragPosLightSpace);
    if (uShadowsOn != 0)
        isInShadow = false;  // if in ShadowOff mode, nothing should be considered in a shadow
    if (!isInShadow)
    {
        d = dot(normal, light);
        if (d > 0.)
        {
            vec3 diffuse = KD * d * uColor;
            lighting += diffuse;
            vec3 refl = normalize(reflect(-light, normal));
            float dd = dot(eye, refl);
            if (dd > 0.)
            {
                s = pow(dd, SHININESS);
                vec3 specular = KS * s * SPECULAR_COLOR;
                lighting += specular;
            }
        }
    }
    gl_FragColor = vec4(lightning, 1.);
}

DisplayWithShadows.frag