Casting Shadows in OpenGL

Identify the Light Source Casting the Shadow

First, Render the Scene from that Light Source
1. Render a view from the light source – everything you cannot see must be in a shadow

Use the Z-buffer as a Depth Shadow Map
2. Generate a depth view from the light source

Second, Render the Scene as Normal, but Consult the Depth Map to Decide where Lighting Applies
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 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, not any colors.
In this grayscale image, dark colors are nearest to the eye, light colors are farther away.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
OpenGL Demo Program: Creating the Off-screen Depth Shadow Map Framebuffer

```glsl
glBindFramebuffer(GL_FRAMEBUFFER, 0);
glReadBuffer(GL_NONE);
glDrawBuffer(GL_NONE);
// force opengl to accept a framebuffer that doesn't have a color buffer in it:
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, GL_TEXTURE_2D, DepthTexture, 0);
gBindFramebuffer(GL_FRAMEBUFFER, DepthFramebuffer);
// attach texture to framebuffer as depth buffer:
gTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
gTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
gTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
GLuint DepthTexture = 0;
GLuint DepthFramebuffer = 0;
gluImageTexture2D(0, 0, 0, 0, 0, GL_RGB, 0, 0);
// create a texture object and a framebuffer:
```
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( ! 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( lighting, 1. );
}

How Did the Demo Program Render the 2D Shadow Map?
{glsl}

DisplayShadowMap.vert
in vec2 vST;

void main( )
{
    vST = gl_MultiTexCoord0.st;
    gl_Position = uProj * uView * uModel * gl_Vertex;
}

DisplayShadowMap.frag
uniform sampler2D uShadowMap;
in vec2 vST;

void main( )
{
    float gray = texture(uShadowMap, vST ).r;
    gl_FragColor = vec4( gray, gray, gray, 1. );
}